Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Захарова Ігоря Володимировича*
(ПІБ)

академічної групи *122-19-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка мобільного додатку (Android та iOS*

"Електронний журнал оцінок студентів" з використанням фреймворку Flutter.

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Приходченко С.Д.*			
розділів:				
спеціальний	*доц. Приходченко С.Д.*			
економічний	*проф. Касьяненко Л.В*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу
бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-19-1* (група) *Захарова Ігоря Володимировича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка мобільного додатку* (Android та iOS) *"Електронний журнал оцінок студентів" з використанням фреймворку Flutter.*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *доц. Приходченко С.Д.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Захаров І.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 140 с., 18 рис., 3 дод., 21 джерел.

Об'єкт розробки: Розробка мобільного додатку (Android та iOS) "Електронний журнал оцінок студентів" з використанням фреймворку Flutter.

Мета кваліфікаційної роботи: розробити мобільний (Android, iOS) додаток для запису та перегляду оцінок студентів в режимі реального часу за допомогою фреймворку Flutter.

У початковій частині роботи проводиться дослідження сучасного стану проблеми, конкретизується сформульоване завдання, визначається мета кваліфікаційної роботи та область, в якій вона застосовується, а також надається аргументація актуальності обраної теми.

У першому розділі роботи здійснюється дослідження об'єкта дослідження та наявних рішень, встановлюється значущість поставленого завдання і призначення розробки, формулюється постановка завдання.

У другому розділі обирається платформа для розробки, виконується проектування програми і її розробка, наводиться опис алгоритму і структури функціонування системи, визначаються вхідні і вихідні дані, наводяться характеристики складу параметрів технічних засобів, описується робота програми.

В економічному розділі визначається трудомісткість розробленого програмного продукту, проводиться підрахунок вартості роботи по створенню застосунку та розраховується час на його створення.

Практичне значення полягає у розробці мобільного додатку для запису та перегляду оцінок студентів за допомогою фреймворку Flutter та мови програмування Dart.

Актуальність програмного продукту визначається зручним і інтуїтивно зрозумілим інтерфейсом для студентів та викладачів, можливістю швидкого оновлення та надання оцінок в режимі реального часу, а також можливістю легкої інтеграції з існуючими системами університетів для автоматизації процесу оцінювання. З використанням Flutter, який забезпечує швидку розробку та високу продуктивність, "Електронний журнал оцінок студентів" може бути ефективним та актуальним рішенням для університетів та освітніх установ.

Список ключових слів: ДОДАТОК, МОБІЛЬНИЙ, FLUTTER, DART, FIREBASE, ПРОДУКТИВНІСТЬ, КОДОВА БАЗА, ІНТЕРФЕЙС КОРИСТУВАЧА.

ABSTRACT

Explanatory note: 140 p., 24 figs., 3 appx., 21 sources.

Development of a mobile application (Android and iOS) "Electronic Journal of Student Grades" using the Flutter framework..

The purpose of the qualification work: to develop a mobile (Android, iOS) application for recording and viewing student grades in real time using the Flutter framework.

In the initial part of the work, a study of the current state of the problem is carried out, the task is specified, the purpose of the qualification work and the area in which it is applied are determined, and the relevance of the chosen topic is argued.

In the first section of the work, the research object and existing solutions are studied, the significance of the task and the purpose of the development are established, and the task statement is formulated.

The second section selects the platform for development, performs program design and development, describes the algorithm and structure of the system, defines input and output data, provides characteristics of the composition of technical means parameters, and describes the program operation.

In the economic section, the labor intensity of the developed software product is determined, the cost of the work on creating the application is calculated, and the time for its creation is calculated.

The practical significance is to develop a mobile application for recording and viewing student grades using the Flutter framework and the Dart programming language.

The relevance of the software product is determined by a user-friendly and intuitive interface for students and teachers, the ability to quickly update and provide grades in real time, and the ability to easily integrate with existing university systems to automate the grading process. With the use of Flutter, which provides fast development and high performance, the "Electronic Student Gradebook" can be an effective and relevant solution for universities and educational institutions.

List of keywords: APPLICATION, MOBILE, FLUTTER, DART, FIREBASE, PERFORMANCE, CODE BASE, USER INTERFACE.

ЗМІСТ

РЕФЕРАТ 3
ABSTRACT 4
СПИСОК УМОВНИХ ПОЗНАЧЕНЬ 7
ВСТУП 8
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ 9
1.1. Загальні відомості з предметної галузі 9
1.2. Призначення розробки та галузь застосування 9
1.3. Підстави для розробки 10
1.4. Постановка завдання 10
1.5. Вимоги до програми або програмного виробу 11
1.5.1. Вимоги до функціональних характеристик 11
1.5.2. Вимоги до інформаційної безпеки 11
1.5.3. Вимоги до складу та параметрів технічних засобів 12
1.5.4. Вимоги до інформаційної та програмної сумісності 12
РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ 13
2.1. Загальні відомості з предметної галузі 13
2.2. Опис застосованих математичних методів 14
2.3. Опис використаних технологій та мов програмування 14
2.4. Опис структури системи та алгоритмів її функціонування 32
2.5. Обґрунтування та організація вхідних та вихідних даних програми 36
2.6. Опис розробленої системи 37
2.6.1. Вимоги до функціональних характеристик 37
2.6.2. Використані програмні засоби 38
2.6.3. Виклик та завантаження програми 39
2.6.4. Опис інтерфейсу користувача 40
РОЗДІЛ 3 ЕКОНОМІЧНИЙ РОЗДІЛ 51
3.1 Розрахунок трудомісткості та вартості розробки програмного продукту 51

3.2. Рахунок витрат на створення програми 54
ВИСНОВКИ 56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 57
ДОДАТОК А 59
ДОДАТОК Б 139
ДОДАТОК В 140

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ОС - операційна система;

UI - інтерфейс користувач;

CI/CD - (Continuous Integration/Continuous Deployment) практика автоматизації інтеграції змін в коді та постійної поставки нових версій програмного продукту;

SDK - набір для розробки програмного забезпечення.

ВСТУП

В сучасних умовах розвитку інформаційних технологій актуальністю стає автоматизація процесів, пов'язаних з освітою. Однією з найважливіших складових цих процесів є система оцінювання студентів, яка дозволяє визначати рівень знань та компетенцій студентів. З метою поліпшення цього процесу та забезпечення більш ефективного взаємодії між студентами та викладачами, у цій дипломній роботі розглядається розробка мобільного додатку "Електронний журнал оцінок студентів" з використанням фреймворку Flutter.

У процесі роботи будуть розглянуті вимоги до функціональності додатку, розроблена архітектура, реалізовано інтерфейс користувача, виконано тестування та проведена оцінка продуктивності додатку.

Результатом роботи є ефективне та зручне рішення для автоматизації процесу оцінювання студентів, яке може бути використано в університетах та інших освітніх установах.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

У сучасному освітньому середовищі електронні журнали оцінок студентів стали необхідним інструментом для збору, зберігання та обробки даних про академічні досягнення студентів. Традиційні паперові журнали вже не задовольняють потреби університетів та вищих навчальних закладів у швидкому доступі до актуальних даних та ефективному спілкуванні між студентами, викладачами та адміністраторами.

Електронні журнали оцінок надають можливість вести облік оцінок, присутності, коментарів та інших релевантних даних, що дозволяє студентам та їх батькам отримувати актуальну інформацію щодо навчального процесу.

Проте, існуючі рішення на ринку часто є платними, складними у використанні або не задовольняють потребам розгортання на різних мобільних платформах. Тому розробка мобільного додатку "Електронний журнал оцінок студентів" з використанням фреймворку Flutter стає актуальною задачею, яка може забезпечити ефективну та зручну систему управління оцінками студентів, що працюватиме на різних платформах та пропонуватиме нові можливості для взаємодії учасників навчального процесу.

1.2. Призначення розробки та галузь застосування

Розробка додатку має на меті полегшити процес ведення журналу оцінок студентів для вчителів та дати змогу студентам відстежувати свій академічний прогрес. Окрім того, додаток може зберігати додаткову інформацію про кожного студента, таку як контактні дані батьків, список предметів, розклад занять та інші важливі дані.

Галузь застосування додатку охоплює освітні заклади будь-якого рівня, де необхідно вести журнал оцінок студентів, а також батьків студентів, які хочуть бути в курсі академічного прогресу своїх дітей. Додаток може бути корисним як в школах, так і в університетах, коледжах та інших навчальних закладах. Також, додаток може бути використаний в позашкільних закладах, таких як музичні школи, спортивні секції та інші організації, де необхідно вести журнал оцінок та відстежувати прогрес учнів.

1.3. Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- освітня програма 122 «Комп’ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;
- завдання на кваліфікаційну роботу на тему “ Розробка мобільного додатку "Електронний журнал оцінок студентів" з використанням фреймворку Flutter”

1.4. Постановка завдання

Завданням даної роботи є розробка мобільного додатку "Електронний журнал оцінок студентів" з використанням фреймворку Flutter.

В результаті необхідно спроєктувати та розробити відлажений та протестований додаток для перегляду та внесення оцінок студентів.

Поставлена задача буде досягнена при виконанні таких умов:

- вивчення предметної галузі;
- планування та розробка UI;
- написання коду програми з дотриманням SOLID принципів та використанням Clean Architecture;
- написання тестів (unit testing, widget testing, integration testing);
- написання yml файлів, використання CI/CD за допомогою GitHub Actions;
- перевірка програми на роботоздатність.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт повинен дотримуватися наступних функціональних вимог:

- зрозумілий інтерфейс користувача;
- авторизація та ролі користувачів за допомогою Firebase Authentication;
- можливість додавати нових користувачів за допомогою Firebase Authentication;
- перегляд та зберігання оцінок студентів за допомогою Firebase Cloud Storage;
- перегляд профілю користувача;
- можливість додавання зображення профіля користувача у базу даних Firebase Firestore;
- експорт таблиць з оцінками у Excel та PDF.

1.5.2. Вимоги до інформаційної безпеки

Для забезпечення інформаційної безпеки користувача повинні бути забезпечені такі умови:

- обробка виняткових ситуацій;

- перевірка пошти та паролю на правильність за допомогою регулярних виразів при авторизації та реєстрації користувача;
- функціонал реєстрації нового користувача наданий лише адміністрації.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для запуску додатку необхідно мати:

- Мобільний пристрій з ОС Android з мінімальною версією SDK 21

АБО

- Мобільний пристрій з ОС iOS

1.5.4. Вимоги до інформаційної та програмної сумісності

Вимоги до інформаційної та програмної сумісності для мобільного додатку "Електронний журнал оцінок студентів" з використанням фреймворку Flutter включають наступні пункти:

- Підтримка різних розмірів екрану та їх роздільної здатності на пристроях;
- Підтримка різних мов та локалей для відображення інтерфейсу та введення даних;
- Підтримка різних форматів збереження та відображення даних.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Загальні відомості з предметної галузі

Результатом даної кваліфікаційної роботи має бути створений відлагоджений мобільний додаток створений на мові програмування Dart з використанням фреймворку Flutter, за допомогою якого вчителі зможуть взаємодіяти з таблицями оцінок за допомогою CRUD (Create, Read, Update, Delete) операцій, вносити оцінки під час освітнього процесу. Студенти матимуть функціонал огляду створених вчителями таблиць.

При взаємодії с таблицею користувач повинен мати можливість у будь який момент завантажити файл Excel до внутрішнього сховища пристрою, в якому міститеться утворена програмою таблиця оцінок студентів. Функціонал завантаження PDF не є реалізований з оглядом на те, що конвертація в PDF вже реалізована безпосередньо програмою Excel і за бажанням користувач зможе самостійно отримати таблицю у PDF файлі.

Для взаємодією з таблицями користувачу попередньо необхідно авторизуватися. Авторизація налагоджена за допомогою сервісу Google Firebase Authentication. Всі створені таблиці користувача будуть зберігатись в NoSQL базі даних Cloud Firestore.

Додаток призначений для спрощення процесу ведення журналів оцінок в освітніх закладах, і не вимагає від користувачів навичок роботи з Excel.

2.2. Опис застосованих математичних методів

У цій програмі використовуються методи сортування, які є важливими для ефективного упорядкування і організації даних. Методи сортування використовуються для розташування елементів у заданому порядку, такому як за зростанням або спаданням значень. Вони дозволяють швидко знаходити, доступатися і аналізувати інформацію, полегшуючи роботу з великими наборами даних. У програмі можуть бути використані різноманітні методи сортування, такі як алгоритми бульбашкового сортування, сортування вставками, сортування обміном або швидке сортування. Використання цих методів допомагає забезпечити ефективну та організовану обробку даних у додатку.

В іншому для роботи додаток не використовує складні математичні методи і заснований виключно на простих алгоритмічних операціях, таких як додавання, віднімання, ділення та множення.

2.3. Опис використаних технологій та мов програмування

У даній кваліфікаційній роботі були використані такі технології та мови програмування:

- Мова програмування Dart;
- Фреймворк Flutter;
- Системи управління станів Flutter;
- Система контролю версій Git;
- GitHub;
- Firebase Authentication;
- База даних Firebase Cloud Firestore;
- Кодогенерація за допомогою бібліотеки freezed;
- CI/CD за допомогою GitHub Actions.

Переглямо ці технології більш детально:

Dart

Dart - це сучасна та ефективна мова програмування, розроблена компанією Google. Її головна мета - створити мову, яка б дозволяла писати надійне та ефективне програмне забезпечення для різних платформ.

Одним з найбільших переваг Dart є його простий і зрозумілий синтаксис, що сприяє швидкому вивченню та розробці. Він має схожість з C-style мовами програмування, що робить його знайомим для багатьох програмістів. Dart підтримує сучасні функції програмування, такі як асинхронність і стрілкові функції, що дозволяють писати зрозумілий і компактний код.

Ще однією важливою особливістю Dart є його вбудована система управління станом, відома як "Flutter State Management". Це дозволяє розробникам ефективно керувати станом додатків і забезпечити відповідну реакцію на зміни даних.

Крім того, Dart має широку підтримку відкритих інструментів і розширень, таких як редактори коду, IDE та пакетний менеджер Pub, що спрощують розробку та підтримку проектів на цій мові.

Одна з ключових областей, де Dart виявляється дуже потужним і популярним, це розробка крос-платформенних мобільних додатків. Завдяки фреймворку Flutter, побудованому на Dart, розробники можуть створювати високоякісні мобільні додатки для платформ Android та iOS з одним кодом. Dart і Flutter забезпечують швидке виконання, гарну продуктивність та багатий набір інтерфейсних компонентів, що дозволяють розробникам швидко і ефективно створювати красиві та функціональні мобільні додатки.

Google Ads є одним з найбільш відомих прикладів мобільного додатку, розробленого з використанням Dart та Flutter. Як платформа рекламного маркетингу Google, Google Ads дозволяє рекламодавцям керувати своїми

рекламними кампаніями та отримувати максимальну вигоду з рекламного бюджету.

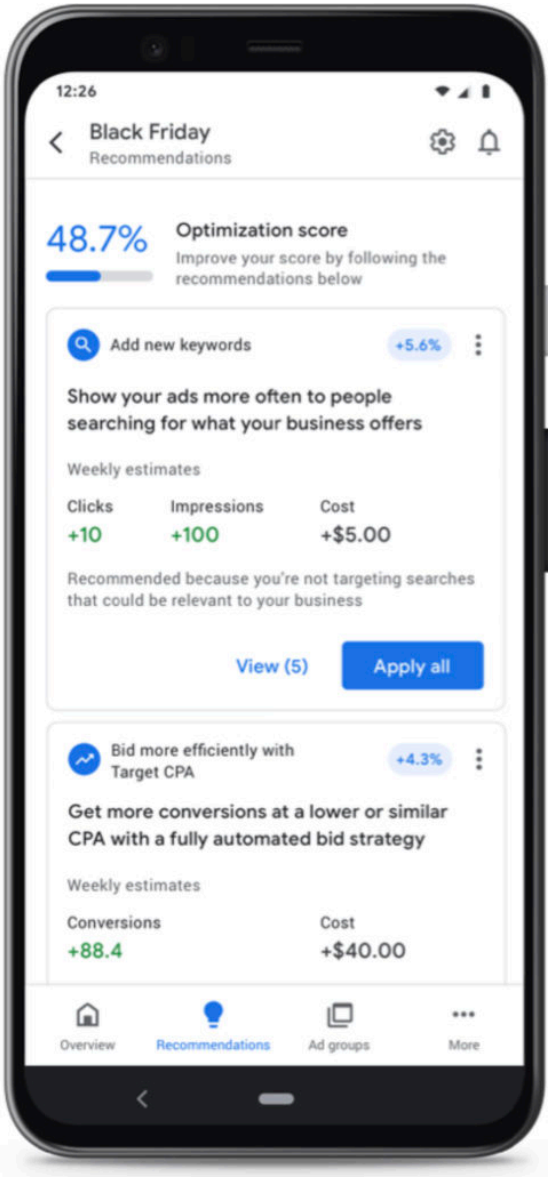

Рис. 2.1. Приклад мобільного додатку написаного на Dart та Flutter – Google Ads

Крім мобільної розробки, Dart також використовується для створення веб-додатків. Він може бути використаний як мова програмування для розробки фронтенду та бекенду веб-додатків. За допомогою фреймворку AngularDart, розробники можуть створювати потужні та масштабовані веб-додатки з використанням компонентної архітектури.

Також Dart може бути використаний для розробки серверних додатків і мікросервісів. Завдяки своїй продуктивності та простоті використання, Dart може бути ефективним вибором для створення швидких та масштабованих серверних додатків.

Крім того, Dart широко використовується в галузі розробки ігор. Завдяки бібліотеці Flame, яка побудована на Dart, розробники можуть створювати прості та захоплюючі ігри для різних платформ.

Flutter

Flutter - це потужний відкритий фреймворк для розробки крос-платформенних мобільних, веб- та настільних додатків. Розроблений компанією Google, він відрізняється своєю унікальною архітектурою та особливостями, що роблять його популярним серед розробників.

Архітектура Flutter - це одна з головних причин, чому цей фреймворк є таким потужним та ефективним для розробки крос-платформенних додатків. Вона базується на концепції "все є віджетом" (everything is a widget), що дозволяє розробникам будувати додатки за допомогою компонентів, які складаються з віджетів.

Головним елементом архітектури Flutter є віджети (widgets), які представляють найменші будівельні блоки додатку. Віджети можуть бути простими, такими як кнопка або текстове поле, або складними, такими як екран або списки. Вони описують як виглядає та поводиться кожен елемент інтерфейсу користувача.

У Flutter кожен аспект користувацького інтерфейсу, будь то кнопка, текстове поле чи навіть весь екран, є віджетом. Віджети можуть бути простими або складними, і вони можуть бути вкладені один в одного для створення багатошарового інтерфейсу. Це дозволяє зручно організовувати та керувати елементами UI.

Flutter використовує декларативний підхід до розробки, що означає, що ви описуєте бажаний стан та вигляд додатку, і Flutter самостійно забезпечує оновлення інтерфейсу, якщо стан змінюється. Це дозволяє зосередитися на логіці додатку, не турбуючись про пряме керування візуальними елементами.

Архітектура Flutter також включає "одну шину подій" (single event loop), що забезпечує швидку та ефективну обробку подій та оновлення інтерфейсу. Вона також підтримує гаряче перезавантаження (hot reload), яке дозволяє розробникам швидко бачити зміни у реальному часі під час розробки, що сприяє швидкості та продуктивності.

Flutter також має вбудовану підтримку для структуризації коду за допомогою різних архітектурних шаблонів, таких як BLoC (Business Logic Component), Provider та GetX. Це дозволяє розробникам організувати свій код у логічні блоки, що полегшує його розуміння та підтримку.

Flutter також славиться своєю високою продуктивністю та швидкістю. Він використовує власний двигун для рендерингу, що дозволяє створювати візуально привабливі та плавні інтерфейси користувача. Flutter також надає доступ до багатого набору вбудованих віджетів та бібліотек, що допомагає розробникам швидко будувати складні інтерфейси та функціональність.

Двигун для рендерингу, який використовується в фреймворку Flutter, називається Skia. Skia є потужним і швидким двигуном для малювання 2D графіки, розробленим компанією Google. Він використовується для створення візуально привабливих та плавних інтерфейсів користувача в додатках, написаних з використанням Flutter.

Skia забезпечує високу продуктивність та ефективність, що дозволяє рендерити графіку швидко та плавно навіть при великій кількості елементів на екрані. Він використовує прискорення апаратного забезпечення для

максимальної швидкості та оптимізації роботи з графічними ресурсами пристрою.

Skia також має вбудовану підтримку для різних ефектів та анімацій, таких як тіні, градієнти, перетини, масштабування та обертання. Це дозволяє розробникам створювати багатошарові та інтерактивні інтерфейси, що відповідають сучасним дизайнерським тенденціям.

Завдяки Skia, фреймворк Flutter може забезпечити високу якість візуального відображення додатків на різних пристроях та платформах. Разом з іншими технологіями та компонентами Flutter, Skia допомагає розробникам створювати естетично привабливі та динамічні додатки, які надають користувачам незабутній досвід.

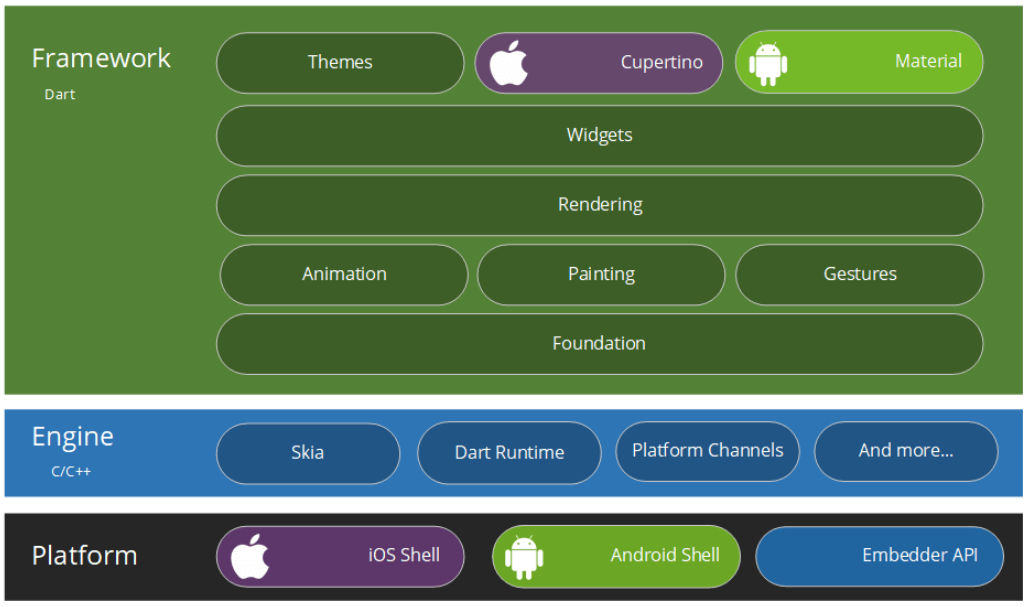

Рис. 2.2. Складові фреймворку Flutter

Загалом, Flutter є потужним інструментом для розробки крос-платформенних додатків з вражаючими можливостями. Його швидкість, продуктивність та гнучкість роблять його відмінним вибором для розробників, які прагнуть створити сучасні та інноваційні додатки для різних платформ.

Системи управління станів Flutter

У фреймворку Flutter існує кілька популярних систем управління станами, серед яких виділяються BLoC, Cubit та Provider. Ці системи надають розробникам потужні інструменти для ефективного керування станом додатків.

Одним з вбудованих механізмів управління станами в Flutter є метод setState(). Цей метод є простим способом оновлення стану віджета та перебудови його інтерфейсу користувача.

Коли використовується setState(), Flutter викликає функцію-зворотний виклик (callback), яка містить оновлення стану віджета. Після оновлення стану, Flutter автоматично викликає метод build() віджета, що приводить до перебудови інтерфейсу користувача з урахуванням нового стану.

Одна з особливостей методу setState() полягає в тому, що він працює на рівні одного віджета. Це означає, що для кожного віджета, який має свій власний стан, потрібно використовувати окремий виклик setState() для оновлення цього стану.

Метод setState() дозволяє реагувати на події та зміни стану додатка, що дозволяє розробникам динамічно оновлювати інтерфейс користувача при необхідності. Він є простим у використанні та зручним для невеликих проектів або випадків, коли стан віджета не є складним або не потребує глобального управління.

Однак, для складних проектів або коли потрібно керувати станом в багатьох віджетах, можуть бути більш потужні системи управління станами, такі як BLoC, Cubit або Provider, які надають більше контролю та організації коду.

BLoC (Business Logic Component) є одним з найпопулярніших шаблонів управління станами в Flutter. Він базується на розділенні бізнес-логіки від інтерфейсу користувача та використовує потоки (streams) для передачі та

обробки даних. BLoC дозволяє розробникам організовувати код у логічні блоки, що спрощує управління станом та забезпечує його повторне використання. Він також підтримує асинхронні операції та дозволяє реагувати на зміни стану для оновлення інтерфейсу користувача.

Cubit є відносно новою системою управління станами, яка була представлена Flutter командою. Вона базується на концепції BLoC, але надає спрощений та більш прямолінійний підхід до управління станом. Cubit надає однонаправлену потокову модель для керування станом та реагування на події. Вона покликана спростити розробку додатків та зменшити складність коду.

Provider - це легкий та простий у використанні механізм управління станами в Flutter. Він базується на концепції "інверсії залежності" (dependency inversion) та "розповсюдження контексту" (context propagation). Provider дозволяє розробникам легко створювати та управляти провайдерами (providers), які постачають дані у різних частинах додатку. Вона спрощує передачу даних між компонентами та забезпечує ефективне управління станом.

Вибір між BLoC, Cubit та Provider залежить від особистих вподобань, потреб проекту та складності додатку. Кожна система має свої переваги та може бути застосована для різних сценаріїв управління станом в Flutter-додатках.

Система контролю версій Git

Система контролю версій Git є потужним інструментом, який дозволяє розробникам ефективно керувати та відстежувати зміни вихідного коду проекту. Вона забезпечує збереження історії змін файлів, спрощує спільну роботу над проектом та дозволяє відновлювати попередні версії коду в разі потреби.

Однією з головних переваг Git є його розподілена архітектура. Кожен розробник має повну копію репозиторію проекту на своєму комп'ютері, що дозволяє працювати незалежно навіть без доступу до мережі. Це також сприяє швидкості та ефективності роботи, оскільки багато операцій можуть бути виконані локально без необхідності взаємодії з централізованим сервером.

Основна перевага Git полягає в його розподіленій природі. Кожен розробник має повну копію репозиторію з історією змін, що дозволяє працювати над проектом навіть у відсутність з'єднання з центральним сервером. Це забезпечує гнучкість та надійність в роботі з Git.

Одним з ключових понять у Git є коміт (commit), який представляє собою фіксацію змін в репозиторії. Кожен коміт має унікальний ідентифікатор і зберігає зміни, внесені в файли проекту. Це дозволяє розробникам відстежувати історію змін, переходити до попередніх версій коду та знаходити причини виникнення проблем.

Git надає багато потужних функцій для керування версіями коду. Це включає можливість створювати гілки (branches) для розробки функціональності незалежно одна від одної та об'єднувати їх пізніше, злиття (merge) змін з однієї гілки в іншу, відстеження стану файлів, розв'язання конфліктів та багато іншого. Ці функції роблять Git потужним інструментом для розробників, що працюють над складними проектами або в команді.

Одним з найбільш використовуваних сервісів для збереження та спільної роботи з Git є GitHub. Він надає хмарне сховище для репозиторіїв, інструменти для спільної роботи, відстеження проблем (issues), а також можливості інтеграції з іншими сервісами.

Git став стандартом в індустрії розробки програмного забезпечення і є незамінним інструментом для керування версіями коду. Він допомагає зберегти цінну інформацію про зміни, спрощує спільну роботу та полегшує

відлагодження проблем. Незалежно від розміру проекту або команди розробників, Git є незамінним помічником у процесі розробки програмного забезпечення.

GitHub

GitHub є однією з найпопулярніших веб-платформ для розробників, яка надає зручні інструменти для спільної роботи над проектами. Ця платформа базується на системі контролю версій Git і надає розширені можливості для збереження та керування версіями програмного коду.

Одна з основних переваг GitHub полягає в його соціальному аспекті. Він дозволяє розробникам публікувати свої проекти, співпрацювати з іншими розробниками, отримувати зворотний зв'язок та вносити внески до відкритих проектів. Це створює потужну спільноту розробників, яка сприяє обміну знаннями та співпраці.

GitHub також надає розширені можливості для керування проектами. Він дозволяє створювати задачі (issues) для відстеження проблем та завдань, що потребують вирішення. Крім того, можна створювати проекти (projects) для організації роботи над конкретними функціональностями або виправленнями помилок. Це допомагає управляти процесом розробки та тримати все під контролем.

Дана кваліфікаційна робота, як і багато інших проектів, використовує переваги GitHub та її код є збереженим окремим репозиторієм на сервісі та налічує 173 комітів.

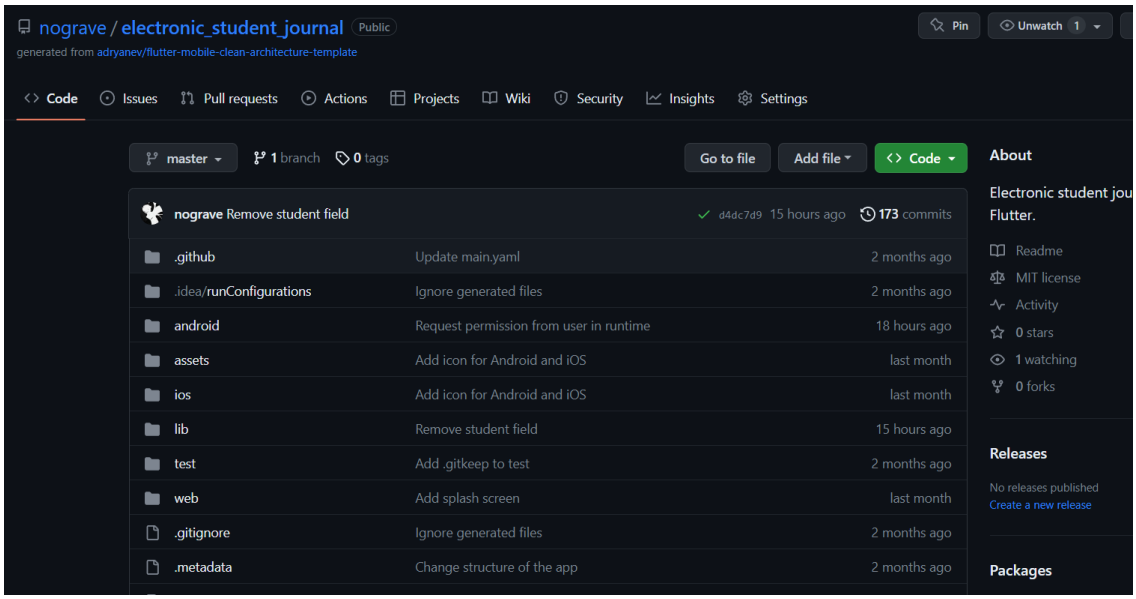

Рис. 2.3. Вигляд репизорію проекту

GitHub також надає можливість для автоматичної інтеграції та розгортання (CI/CD) проектів. CI/CD дозволяє автоматизувати процес тестування, збирання та розгортання програмного коду. Це покращує якість розробки та забезпечує швидку ітерацію проекту.

Загалом, GitHub є не тільки системою контролю версій, але і платформою, яка сприяє співпраці, обміну знаннями та ефективному управлінню проектами. Він є важливим інструментом для розробників, допомагаючи їм покращити якість своїх проектів, спілкуватися зі спільнотою та розвиватися в своїй кар'єрі.

Firebase Authentication

Firebase Authentication - це сервіс аутентифікації, який надає потужні інструменти для безпечного та простого управління процесом аутентифікації в додатках. Він дозволяє розробникам швидко впровадити функціональність аутентифікації, що дозволяє користувачам реєструватися, увійти та аутентифікуватися в додатку за допомогою різних методів входу.

Firebase Authentication підтримує широкий спектр методів аутентифікації, включаючи електронну пошту та пароль, аутентифікацію через соціальні мережі (такі як Google, Facebook, Twitter), номер телефону, аутентифікацію з використанням постачальників OpenID та інші. Це надає розробникам гнучкість у виборі методу аутентифікації, що найкраще відповідає потребам їхнього додатку та його цільової аудиторії.

Один з головних переваг Firebase Authentication - його безпека. Сервіс надає високий рівень захисту ваших користувачів, включаючи захист від атак перехоплення сесій, зламу паролів та інших загроз безпеки. Він автоматично керує хешуванням паролів, перевіркою електронної пошти та номерів телефонів, а також надає можливість налаштувати додаткові правила безпеки за допомогою Firebase Security Rules. Firebase Authentication також надає розробникам інструменти для управління користувачами, включаючи можливість створення та збереження додаткових користувацьких атрибутів, надання адміністративних привілеїв, використання мультифакторної аутентифікації та багато іншого. Це дозволяє розробникам зручно керувати користувачами та налаштовувати функціонал аутентифікації під свої потреби.

В ході створення проекту ми використаємо сервіс Firebase Authentication для авторизації та регістрації учнів та вчителей. При створенні аккаунту дані користувачів будуть зберігатися в базі даних Firebase Cloud Firestore.

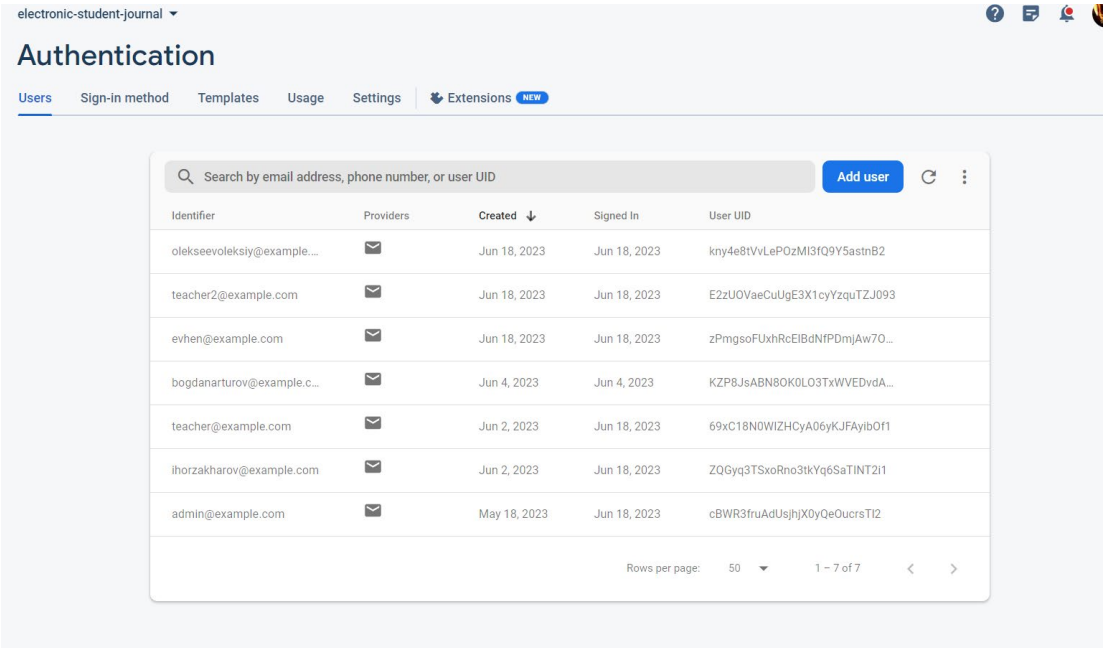

Рис. 2.4. Вигляд зареєстрованих користувачів в сервісі Firebase Authentication

Загалом, Firebase Authentication - це потужний та надійний сервіс аутентифікації, який допомагає розробникам швидко впровадити безпечну функціональність аутентифікації в своїх додатках. Він спрощує процес реєстрації та входу користувачів, забезпечує високий рівень безпеки та надає гнучкість у налаштуванні. Firebase Authentication є важливою складовою Firebase-екосистеми і дозволяє розробникам зосередитися на основних аспектах розробки додатків, маючи впевненість в безпеці та надійності аутентифікаційного процесу.

База даних Firebase Cloud Firestore

Firebase Cloud Firestore - це гнучка та масштабована NoSQL база даних, яка пропонує розробникам широкі можливості зберігання та синхронізації даних для їх мобільних, веб- та серверних додатків. Вона входить до складу Firebase-платформи і надає розробникам зручні інструменти для роботи зі структурованою і неструктурованою інформацією.

Однією з ключових переваг Firebase Cloud Firestore є його висока масштабованість. База даних автоматично масштабується вгору або вниз в залежності від навантаження, що дозволяє обробляти як невеликі, так і дуже великі обсяги даних. Вона також забезпечує миттєву синхронізацію даних між різними клієнтськими пристроями, що дозволяє користувачам бачити оновлені дані в режимі реального часу.

Ціноутворення на використання бази даних Firebase Cloud Firestore базується на кількох факторах. При розрахунку вартості використання Firestore враховується кількість операцій читання, запису та запитів, обсяг передачі даних та простір для зберігання даних.

Firebase Cloud Firestore пропонує безкоштовний план "Spark", який має обмеження на кількість операцій читання, запису та запитів, а також на обсяг передачі даних. Цей безкоштовний план відмінний для розробників, які працюють над невеликими проектами або хочуть протестувати функціонал Firestore. Проект кваліфікаційної роботи є невеликим за обсягом проектом, де необхідні незначні кількості читання та запису інформації, тож план "Spark" ідеально підійде для цього проекту.

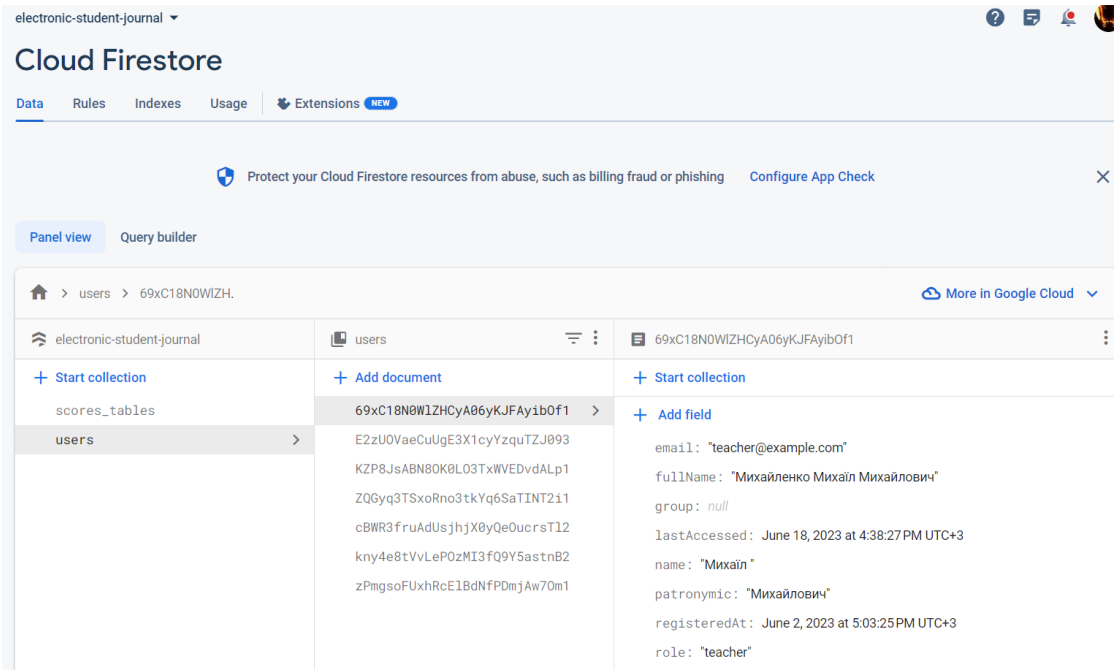

Рис. 2.5. Вигляд даних програми які збережені у Firebase Cloud Firestore

Крім безкоштовного плану, Firebase також пропонує платний план під назвою "Blaze". За платний план розрахунок вартості здійснюється на основі кількості операцій читання, запису, запитів, обсягу передачі даних та простору для зберігання. Це дає розробникам більшу гнучкість та можливість масштабувати свої проекти залежно від їх потреб.

Ціна за використання Firestore залежить від регіону, в якому розташовані сервери Firebase, а також від кількості та типу операцій, що виконуються. Наприклад, операції читання коштують менше, ніж операції запису. Додатково, вартість передачі даних в межах Firebase-екосистеми (наприклад, між Firestore та іншими сервісами Firebase) може бути безкоштовною або має знижені тарифи. Важливо врахувати, що ціноутворення на використання Firebase Cloud Firestore може змінюватися з часом, оскільки Firebase постійно вдосконалює свої послуги та пропонує нові функції.

Іншою важливою особливістю Firebase Cloud Firestore є його зручність в роботі з даними. Він пропонує документ-орієнтовану модель, де дані зберігаються у вигляді колекцій документів. Кожен документ може містити різні поля та значення, і їх можна швидко отримувати, оновлювати або видаляти за допомогою зрозумілих API. Firebase Cloud Firestore також підтримує запити, сортування та фільтрацію даних, що дозволяє розробникам легко отримувати необхідну інформацію з бази даних.

Безпека є ще однією важливою складовою Firebase Cloud Firestore. Вона забезпечує механізми автентифікації та авторизації для захисту даних від несанкціонованого доступу. Розробники можуть налаштовувати правила доступу до даних на основі різних критеріїв, таких як роль користувача, і забезпечувати безпеку своїх даних.

Firebase Cloud Firestore також інтегрується з іншими сервісами Firebase, що дозволяє розробникам використовувати його разом з іншими функціями, такими як аутентифікація, зберігання файлів, повідомлення в реальному часі та багато інших. Це робить його потужним інструментом для розробки повноцінних додатків з різноманітними функціями та можливостями.

Кодогенерація за допомогою бібліотеки freezed

Кодогенерація за допомогою бібліотеки freezed є потужним інструментом, який спрощує розробку моделей даних в мові програмування Dart. Вона дозволяє автоматично генерувати код для створення імутабельних (immutable) класів, які містять поля даних та методи серіалізації та десеріалізації.

Одна з основних переваг кодогенерації з використанням freezed полягає в тому, що вона зменшує рутинну роботу розробника та ризик помилок. Замість того, щоб вручну створювати класи даних, поля, методи серіалізації та

десеріалізації, freezed використовує аннотації та аннотаційний препроцесор, щоб згенерувати весь необхідний код автоматично.

Це дозволяє розробникам зосередитися на логіці додатку, замість витрачання часу на рутинне програмування. Код, згенерований freezed, забезпечує безпечну та незмінну модель даних, що допомагає уникнути помилок, пов'язаних з неправильними змінами даних.

Крім того, freezed підтримує серіалізацію та десеріалізацію даних в різні формати, такі як JSON або бази даних. Вона автоматично генерує методи для перетворення об'єктів на рядки JSON і навпаки, що значно спрощує роботу з обміном даними з сервером або збереженням даних в базі даних.

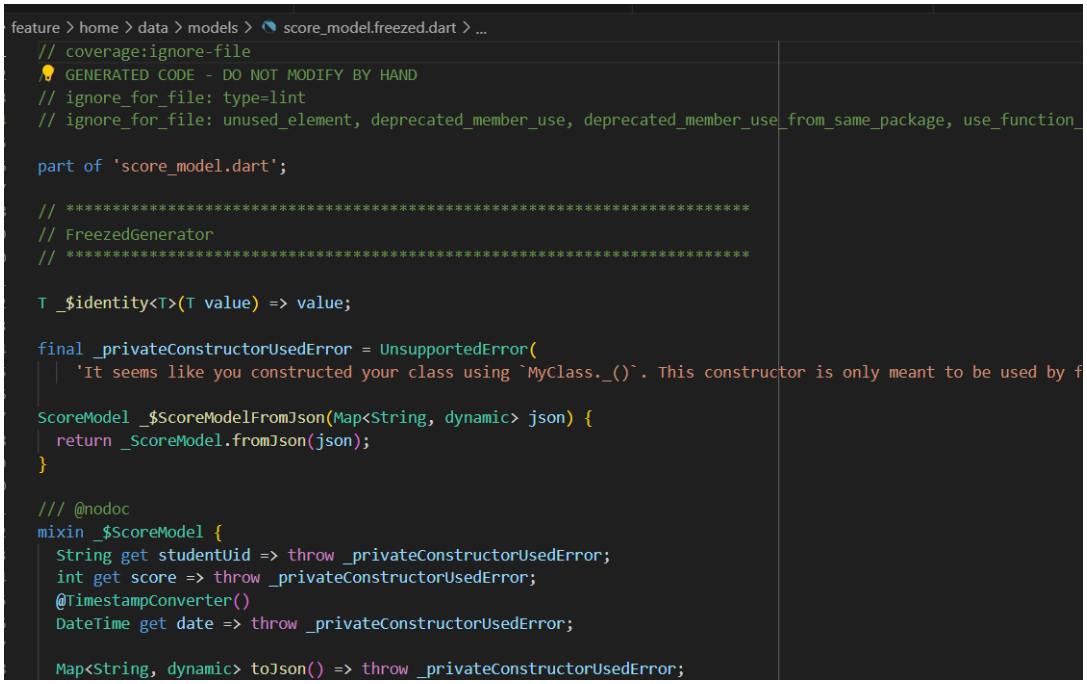

Рис. 2.6. Приклад коду, згенерованого за допомогою бібліотеки freezed

Freezed є популярною бібліотекою серед розробників Dart і знаходить широке застосування в різних типах проектів, від мобільних додатків до веб-додатків та серверних програм. Вона допомагає покращити швидкість розробки, підвищити якість коду та зменшити кількість потенційних помилок.

Окрім кодогенерації, freezed також надає можливості для роботи з імутабельними структурами даних, такими як списки та мапи, і має вбудовану підтримку різних паттернів стану, таких як BLoC або Provider.

CI/CD за допомогою GitHub Actions

Continuous Integration/Continuous Deployment (CI/CD) є незмінною складовою розробки програмного забезпечення. Ця методологія сприяє автоматизації процесу збирання, тестування та розгортання програмного коду, забезпечуючи швидкі та безперервні зміни в проекті.

Одним з популярних інструментів для реалізації CI/CD є GitHub Actions. Це сервіс, який надається GitHub і дозволяє розробникам створювати та налаштовувати автоматизовані робочі процеси для їх репозиторіїв. За допомогою GitHub Actions можна визначати набір кроків, які автоматично виконуються при кожному коміті або пул-реквесті. Ці кроки можуть включати збирання проекту, запуск автоматичних тестів, розгортання програми на серверах та інші дії, необхідні для підтримки якості та безперервності розробки.

GitHub Actions має багато переваг. Особливою є його інтеграція з екосистемою GitHub. Розробники можуть легко налаштувати робочі процеси за допомогою файлу конфігурації, який знаходиться прямо у їх репозиторії. Це спрощує управління та збереження конфігурації і дозволяє команді розробників спільно працювати над покращенням робочих процесів.

GitHub Actions також надає гнучкість у реалізації різних сценаріїв CI/CD. Завдяки широкому спектру доступних дій та підтримці різних мов програмування, розробники можуть виконувати різні дії, включаючи встановлення залежностей, компіляцію коду, автоматичне тестування та створення звітів про покриття коду. Більш того, GitHub Actions інтегрується з популярними сервісами, такими як Docker, AWS, Google Cloud та багатьма

іншими. Це дає розробникам можливість використовувати широкий спектр інструментів для розгортання та управління своїми додатками.

Наявність логів та статусів в GitHub Actions дозволяє розробникам отримувати швидку зворотну інформацію про стан їх процесу CI/CD. Це сприяє виявленню та виправленню проблем швидше, полегшує співпрацю між розробниками та сприяє постійному вдосконаленню розробки.

Використання GitHub Actions для CI/CD допомагає розробникам автоматизувати та прискорити процес розробки програмного забезпечення. Це забезпечує стабільність та високу якість коду, полегшує співпрацю в команді розробників та дозволяє швидко реагувати на зміни та вдосконалення. Тому GitHub Actions є одним з найпопулярніших інструментів для впровадження CI/CD у проектах розробки програмного забезпечення.

2.4. Опис структури системи та алгоритмів її функціонування

Опис структури системи та алгоритмів її функціонування з використанням Clean Architecture відбувається на основі певних принципів та концепцій, які спрямовані на створення модульної, легко змінної та тестованої системи програмного забезпечення.

Clean Architecture - це підхід до організації структури програмного забезпечення, який сприяє його модульності, розширюваності та тестованості. Використовуючи Clean Architecture в Flutter, ми можемо створити добре структуровану систему, яка легко підтримується та розвивається з часом.

Структура системи з використанням Clean Architecture в Flutter може включати наступні шари:

Presentation Layer:

Цей шар відповідає за представлення даних та взаємодію з користувачем. В Flutter це можуть бути віджети, які відображають інтерфейс користувача та відповідають за обробку введення. В Presentation Layer також можуть бути включені BLoC (Business Logic Component) або Provider, які відповідають за управління станом додатку та обробку подій.

Domain Layer:

Цей шар містить бізнес-логіку та моделі домену. Він повинен бути незалежним від фреймворків та зовнішніх бібліотек. В Flutter це можуть бути класи, які відповідають за логіку додатку, обробку даних та розрахунки. Domain Layer визначає контракти (інтерфейси) для взаємодії з Data Layer та Presentation Layer.

Data Layer:

Цей шар відповідає за доступ до джерел даних, таких як бази даних, мережеві запити, файлові системи тощо. В Flutter це можуть бути репозиторії або класи, які відповідають за взаємодію зі зовнішніми джерелами даних. Data Layer реалізує контракти, визначені в Domain Layer, та забезпечує збереження та отримання даних.

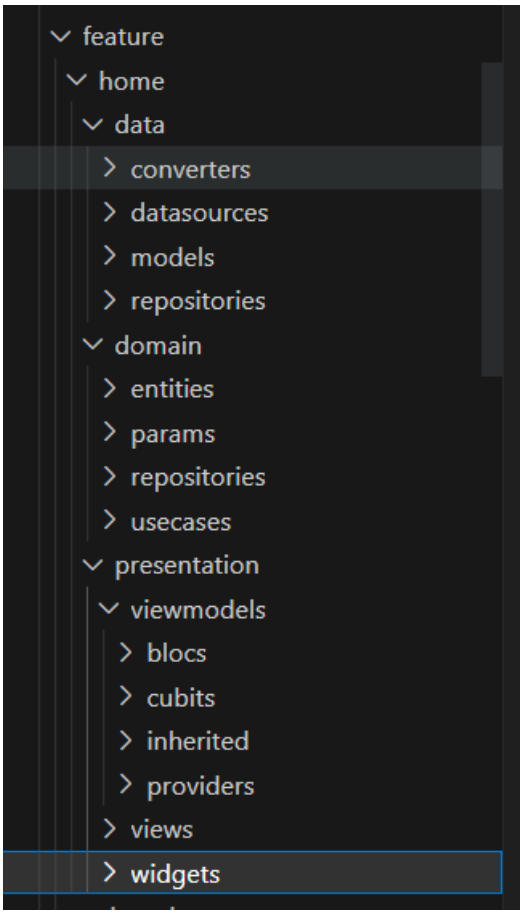

Рис. 2.7. Зміст Data, Domain та Presentation шарів

Алгоритми функціонування системи з Clean Architecture в Flutter можуть бути наступними:

1. Користувач взаємодіє з Presentation Layer, наприклад, вводить дані або натискає на кнопку.
2. Presentation Layer обробляє введення та ініціює відповідну бізнес-логіку у Domain Layer.
3. Domain Layer виконує необхідні операції, обчислення та маніпуляції з даними, використовуючи сервіси та репозиторії, визначені в цьому шарі.
4. Якщо потрібно отримати або зберегти дані, Domain Layer взаємодіє з Data Layer через визначені контракти.
5. Data Layer забезпечує доступ до джерел даних, виконує запити до бази даних або мережі та повертає результати до Domain Layer.

6. Після обробки запиту та отримання результатів, Domain Layer повертає відповідні дані до Presentation Layer.
7. Presentation Layer оновлює відповідні віджети або стан додатку, щоб відобразити оновлені дані та взаємодію з користувачем.

Цей цикл повторюється залежно від взаємодії користувача та потреб системи. Clean Architecture дозволяє зберегти логіку домену незалежною від фреймворку та легко міняти Presentation Layer або Data Layer без впливу на решту системи.

Окрім цього, в бібліотеці lib також знаходитимуться допоміжні бібліотеки core, gen та utils та два файли з розширенням .dart. Папка core зберігає головний віджет програми my_app.dart, файл для ін’єкції залежностей, app_router для навігації між сторінками програми, файли локалізації та ін. В бібліотеці gen знаходяться згенеровані freezed бібліотекою файли. В utils – допоможні файли extension для розширювання базових класів необхідним функціоналом. В файлі firebase_options міститься інформація для Firebase. А main.dart є вхідною точкою програми.

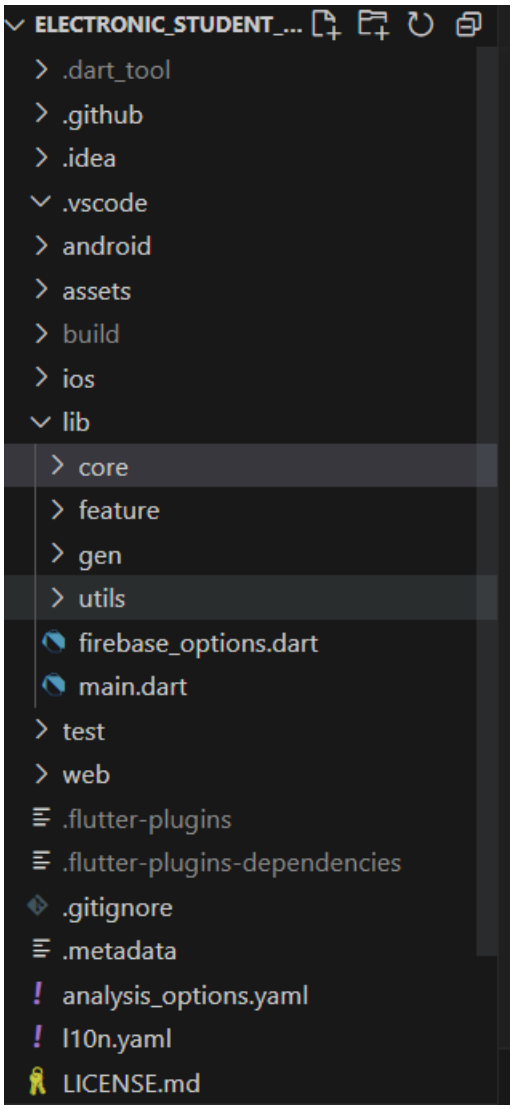

Рис. 2.8. Загальний зміст папки програми

2.5. Обґрунтування та організація вхідних та вихідних даних програми

У даній системі вхідною інформацією є різноманітні дані, які вносяться користувачем за допомогою клавіатури та миші. Під час авторизації це включає дані користувача такі як прізвище, ім’я, по-батькові, назва освітнього закладу, група (у випадку студента) та ін. При взаємодії з таблицями це включає дані оцінок студентів, дати виставлення оцінок.

З вихідних даних важливою є інформація, яка надається користувачеві під час відображення даних додатком. Це можуть бути заповнені таблиці з оцінками

або інформація користувача. Також вихідними даними може бути Excel таблиці утворені програмою.

Вхідні дані, внесені користувачем, дозволяють системі збирати та обробляти необхідну інформацію для забезпечення коректної роботи та взаємодії з користувачем. Вихідні дані, надані системою користувачеві, створюють зручний інтерфейс, де користувач може отримати потрібну інформацію та виконати необхідні дії.

2.6. Опис розробленої системи

2.6.1. Вимоги до функціональних характеристик

Для розробки була використана електронно-обчислювальна машина(ноутбук) з нижченаведеними характеристиками:

- OS: Windows 10
- CPU: AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
- GPU: AMD Radeon Vega 8 Graphics
- Memory: SSD 256 GB
- RAM: 8 GB.

Також для перевірки коректності роботоздатності програми був використаний реальний мобільний пристрій з наступними характеристиками:

- OS: Android 11
- CPU: Qualcomm Snapdragon 860
- GPU: Adreno 640
- Memory: 256 GB
- RAM: 8 GB.

2.6.2. Використані програмні засоби

Microsoft Visual Studio Code (VS Code) - це безкоштовний і надзвичайно популярний текстовий редактор, розроблений компанією Microsoft. Він став вибором багатьох розробників завдяки своїм потужним можливостям, легкості використання та великому спектру розширень.

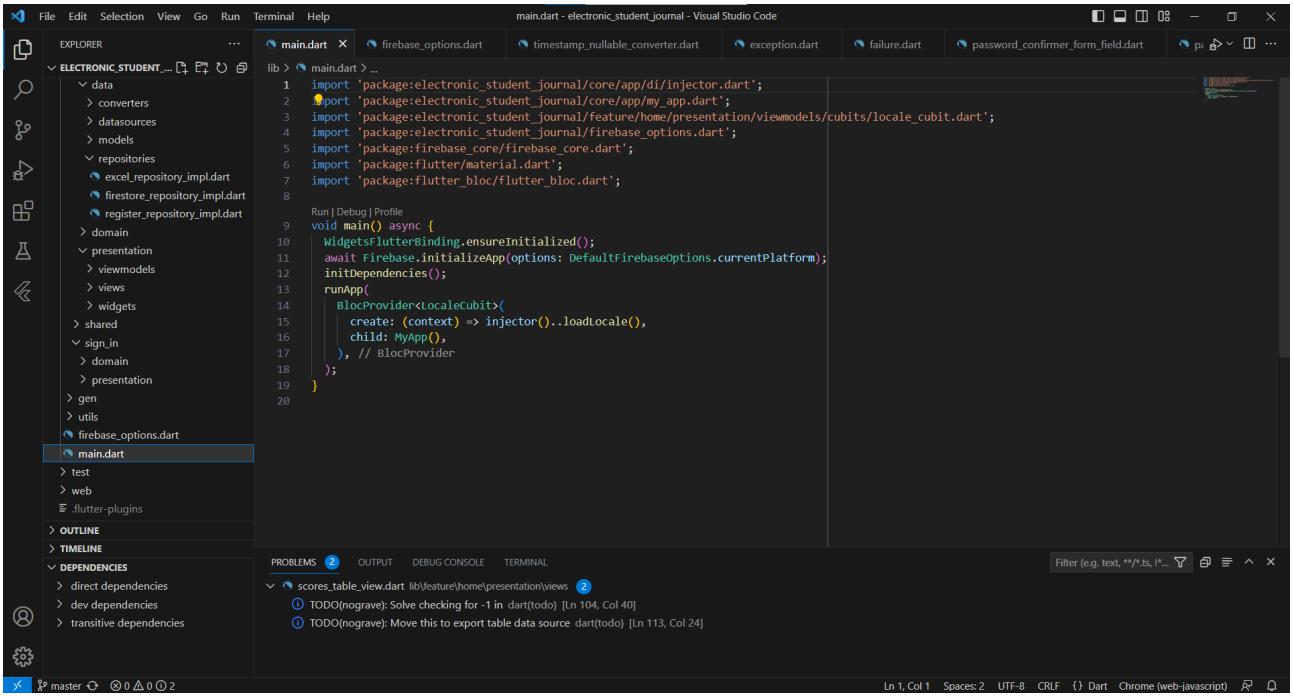

Рис. 2.9. Інтерфейс програми Microsoft Visual Studio Code

Однією з основних переваг Visual Studio Code є його розширюваність. Завдяки великому співтовариству розробників, доступних розширень для VS Code є надзвичайно багато. Розширення дозволяють розширити функціональність редактора, додати нові можливості, інтеграції з різними інструментами та мовами програмування. Наприклад, є розширення для роботи з Git, вбудований термінал, розширення для роботи з Python, JavaScript, HTML та багатьма іншими мовами програмування.

VS Code має інтуїтивний і дружній інтерфейс, що полегшує його використання. Редактор надає потужні можливості для редагування тексту, зокрема автодоповнення, підсвічування синтаксису, перемикання між файлами

та швидкий пошук і заміна тексту. Також, VS Code підтримує роботу з проектами, що дозволяє зручно організовувати та керувати робочими середовищами.

Ще однією з переваг VS Code є його вбудована інтеграція з різними системами контролю версій, зокрема з Git. Редактор надає можливість переглядати, комітити, пушити та тягнути зміни в репозиторії Git безпосередньо з інтерфейсу. Це спрощує роботу з версійним контролем та сприяє ефективній співпраці в команді розробників.

Крім того, VS Code має багато інших корисних функцій, таких як відладка коду, вбудована підтримка терміналу, можливість встановлення тем оформлення та налаштування розмітки редактора під власні потреби. Редактор також підтримує роботу з контейнеризованими середовищами, що дозволяє легко налаштовувати і використовувати різні середовища для розробки.

2.6.3. Виклик та завантаження програми

Для того, щоб розпочати використання проєкту, необхідно спочатку завантажити архів з програмою з Інтернету, а потім розпакувати його за допомогою будь-якого архіватора. Після розпакування, вам просто потрібно запустити виконавчий файл, який знаходиться в папці програми - ніяких додаткових інсталяцій не потрібно. На рис. 2.10 зображений приблизний вигляд архіву після розпакування.

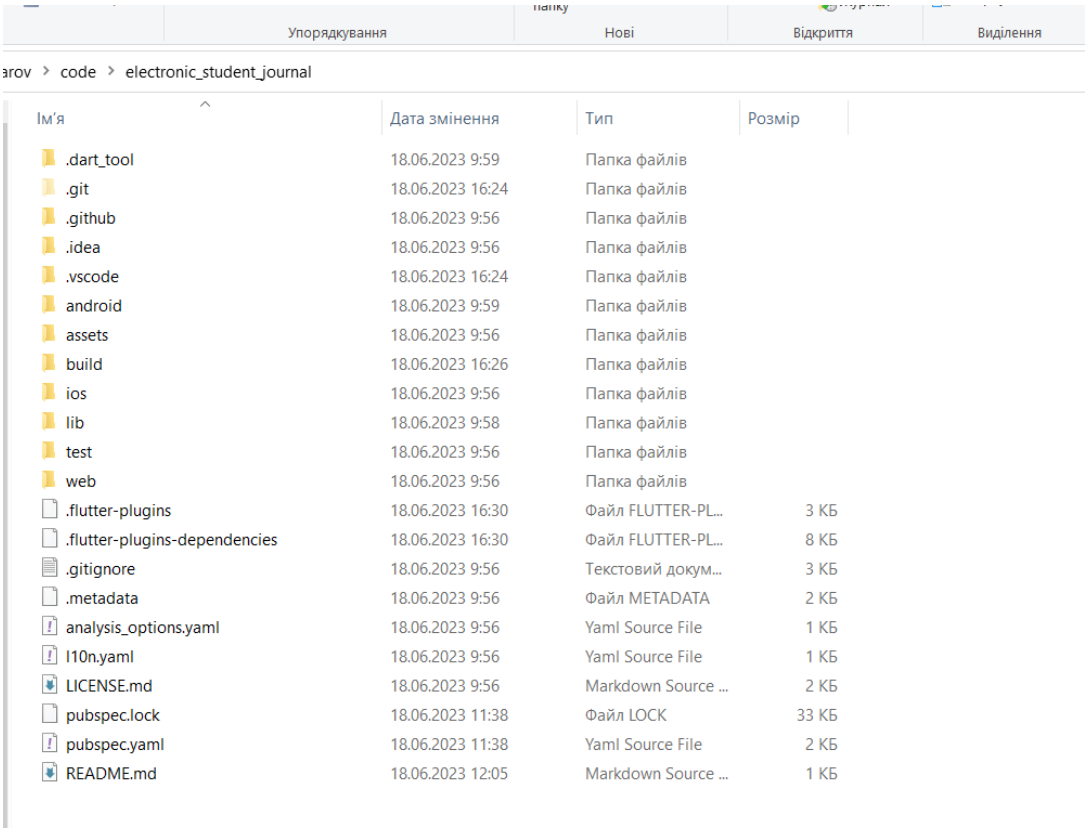

Рис. 2.10. Приклад папки з програмою після розпакування архіву

2.6.4. Опис інтерфейсу користувача

При першому запусці програми після показу сплеш скріну (екрану загрузки) з логотипом проекту нас повинна зустріти сторінка з формою авторизації (рис. 2.11).

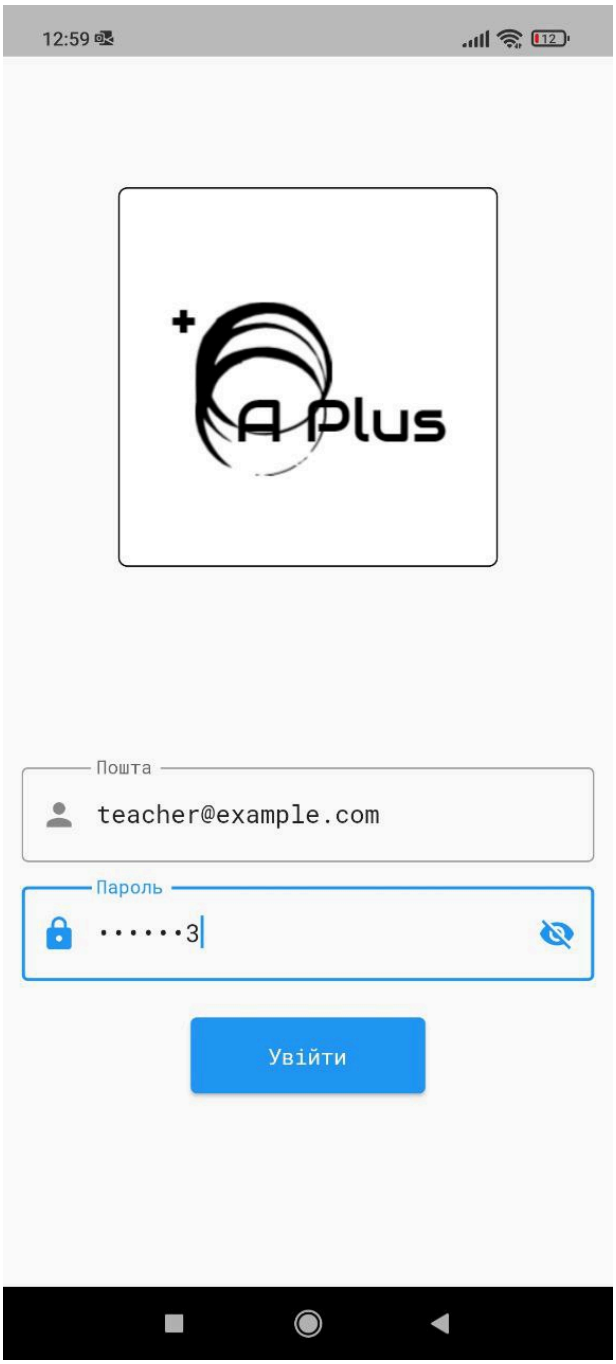

Рис 2.11. Сторінка авторизації

При заповненні форми, з правильними поштою та паролем, в залежності від того яку роль має користувач (роль та інші дані користувача попередньо завантажуються з серверу) буде показані різні головні сторінки. Якщо роль користувача – студент або вчитель буде показана сторінка на рис. 2.12.

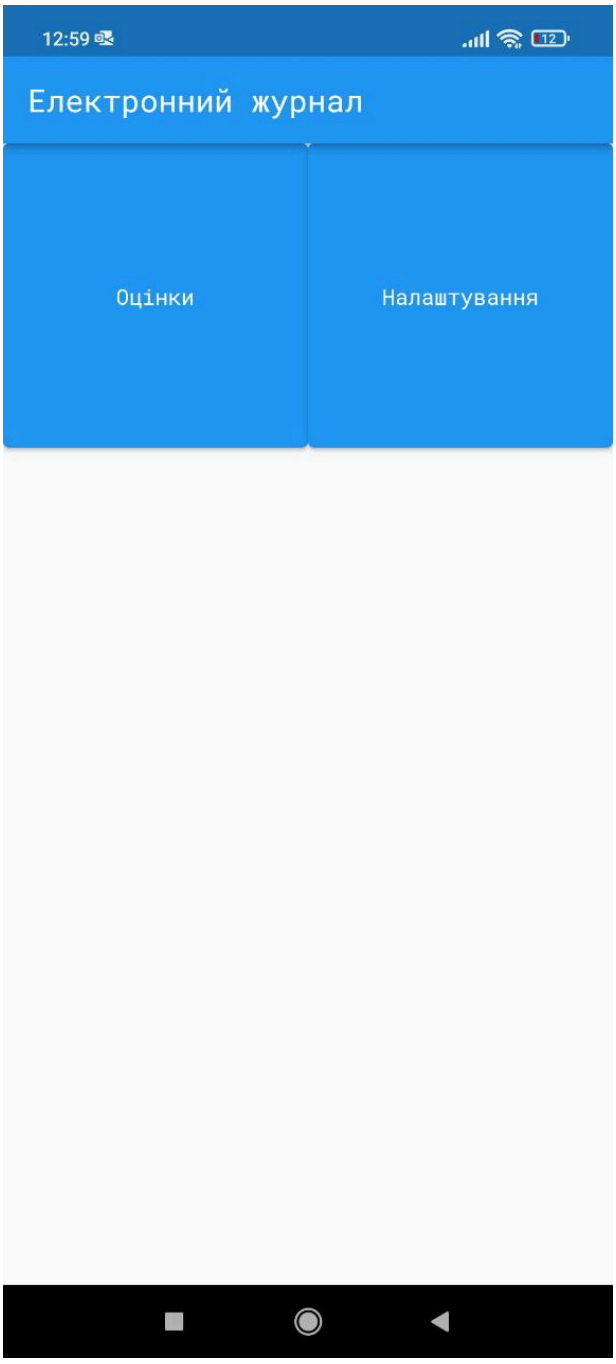

Рис. 2.12. Головна сторінка програми для користувачів
з роллю студент або вчитель

При натисканні на кнопку "Оцінки" користувача буде перенесено на сторінку з завантаженими з серверу назвами існуючих таблиць. Для вчителя будуть показані створені їм таблиці та кнопка для створення оцінки, для студентів – таблиці де є їх оцінки (рис. 2.13). Також задля зручності біля назви

сторінки є стрілка влево при натисканні на яку можна повернутись на попередню сторінку.

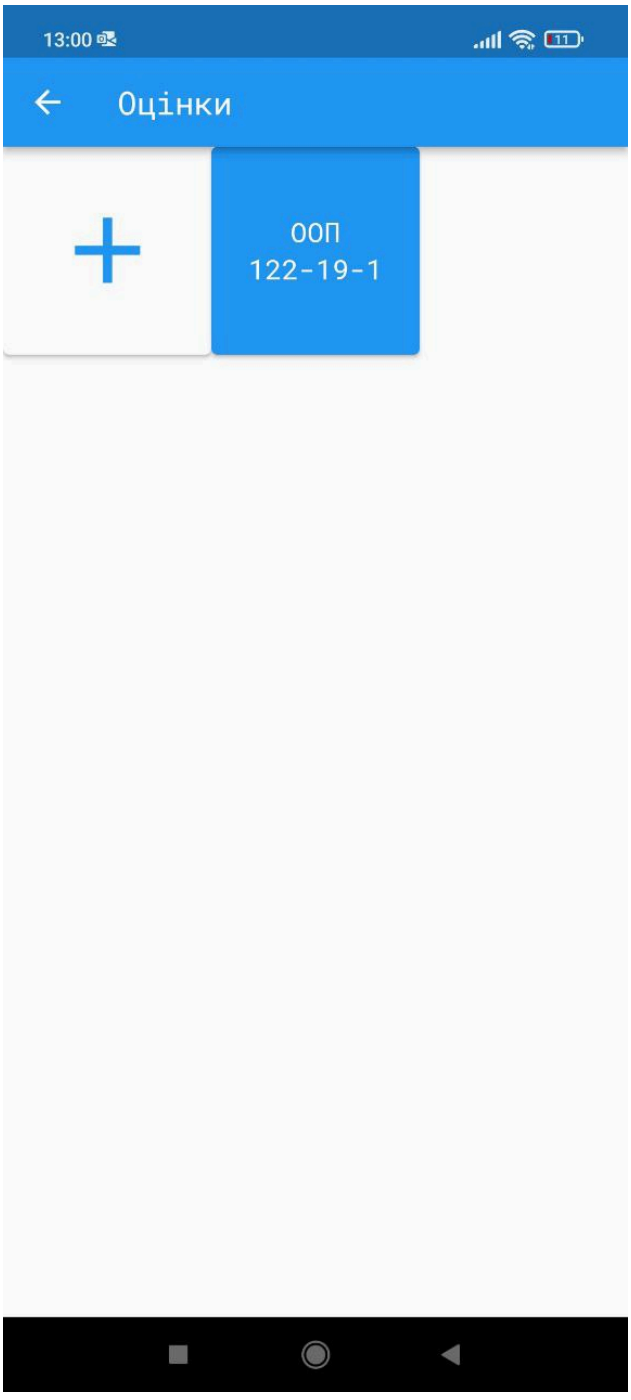

Рис. 2.13. Сторінка з назвами таблиць

При натисканні на кнопку з назвою таблиці, буде виведена сторінка з відповідною таблицею оцінок(рис. 2.14). При натисканні на кнопку "+" буде виведена сторінка редагування таблиці(рис. 2.15) з пустими полями.

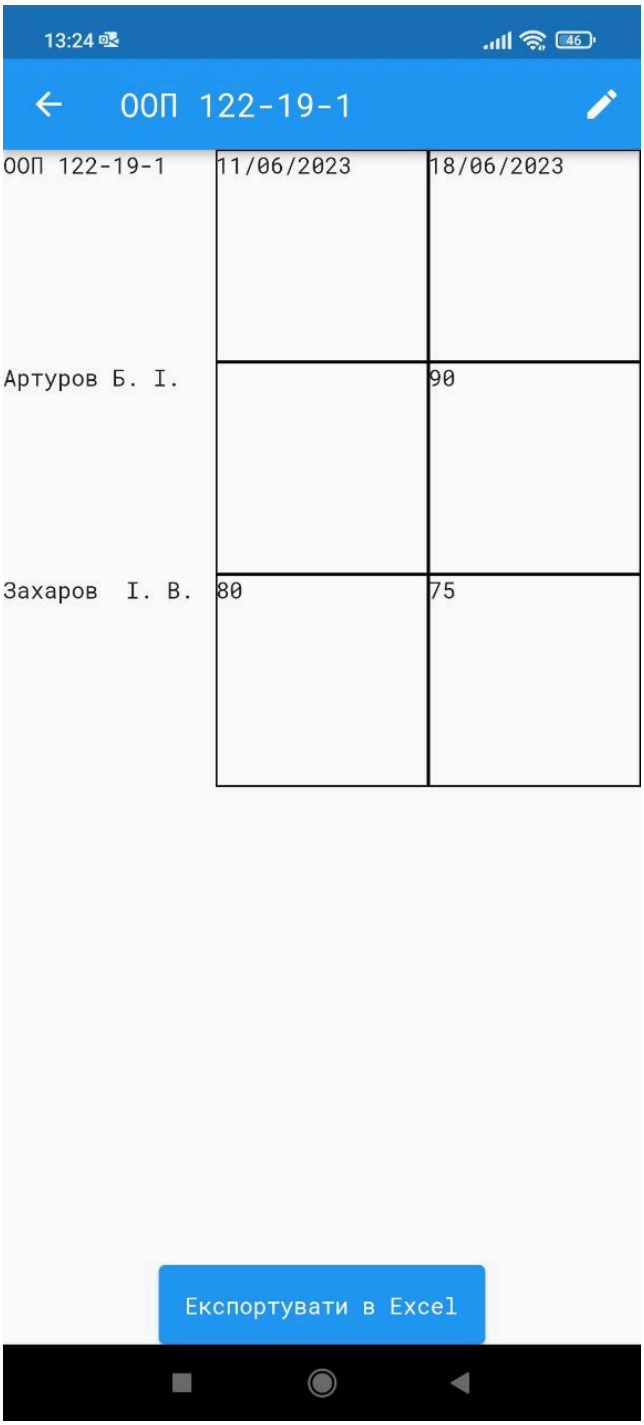

Рис. 2.14. Сторінка з таблицею оцінок

На рис. 2.14 ми бачимо таблицю при ролі вчителя, для якого дозволено редагувати таблицю. При ролі студента буде відображена така ж сторінка але без можливості редагування. При натисканні на кнопку "Експортувати в Excel" в локальному сховищі користувача буде створено файл з розширенням .xlxs з бажаною таблицею. Всі оцінки які ми побачили на сторінці будуть ідентично

виглядати і в Excel файлі. При натисканні на кнопку зверху справа (карандаш) вчитель побачить сторінку редагування таблиці.

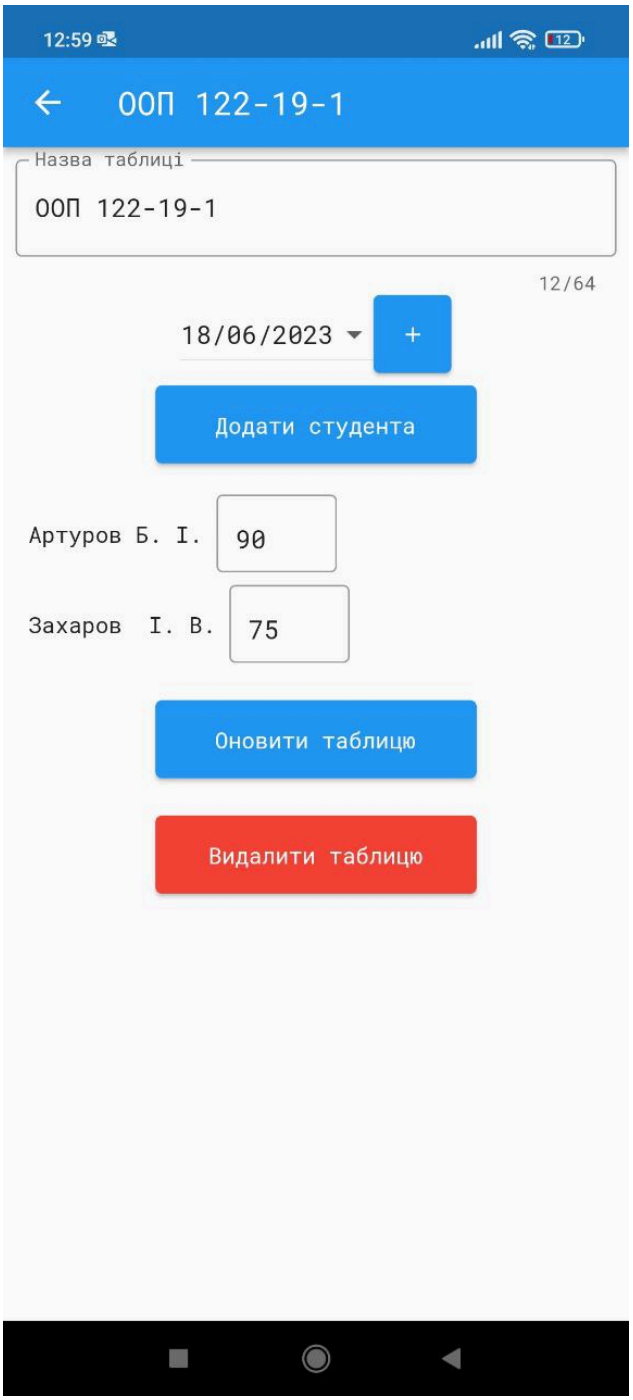

Рис. 2.15. Сторінка редагування таблиці

На рис. 2.15 ми можемо побачити поля та кнопки редагування оцінок та назви таблиці. При натисканні на кнопку "Додати студента" на екрані з’явиться

поле, при вводі тексту в який можна знайти необхідного студента. Знайдені будуть тільки ті студенти назва навчального закладу яких співпадає з навчальним закладом вчителя. При натисканні на кнопку "Оновити таблицю" редагована таблиця буде збережена в базі даних, при натисканні на "Видалити таблицю" – видалена. При натисканні будь-якої з цих кнопок користувача буде перенесено на головну сторінку програми.

Якщо спочатку ми авторизувались з роллю адміна нас буде направлено на наступну сторінку(рис. 2.16).

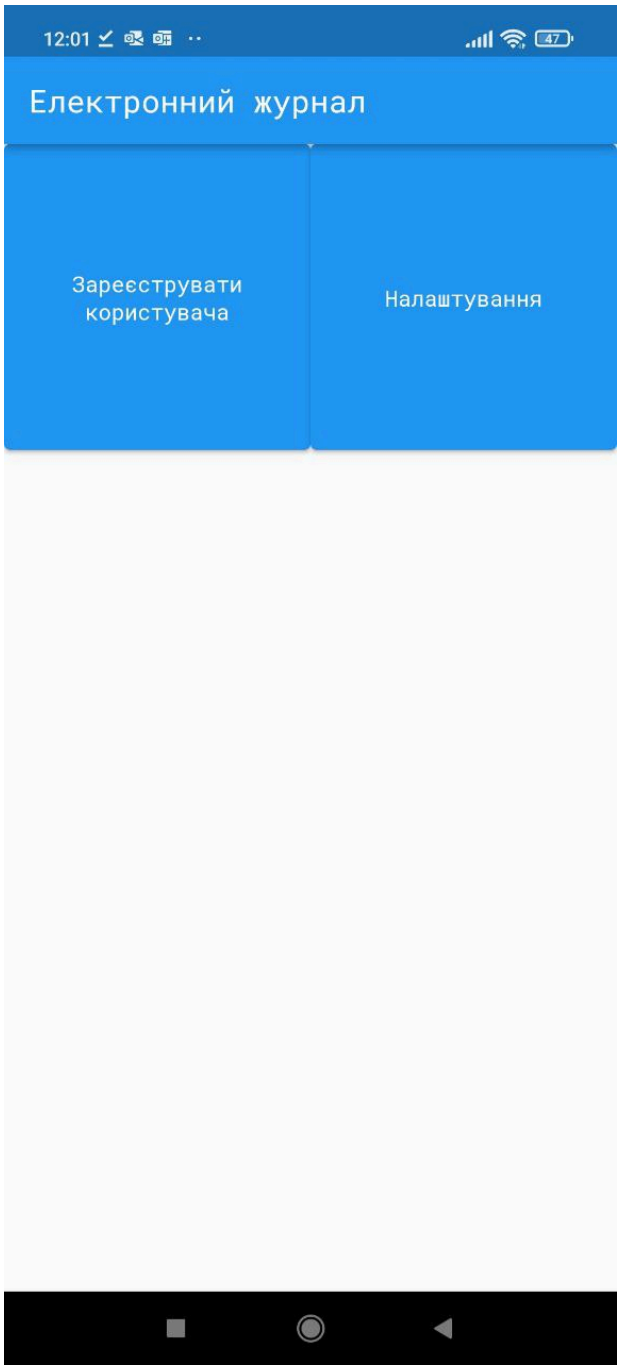

Рис. 2.16. Головна сторінка програми для користувачів з роллю студент або вчитель

При натисканні на кнопку "Зареєструвати користувача" користувача буде перенесено на сторінку реєстрації користувача(рис 2.17).

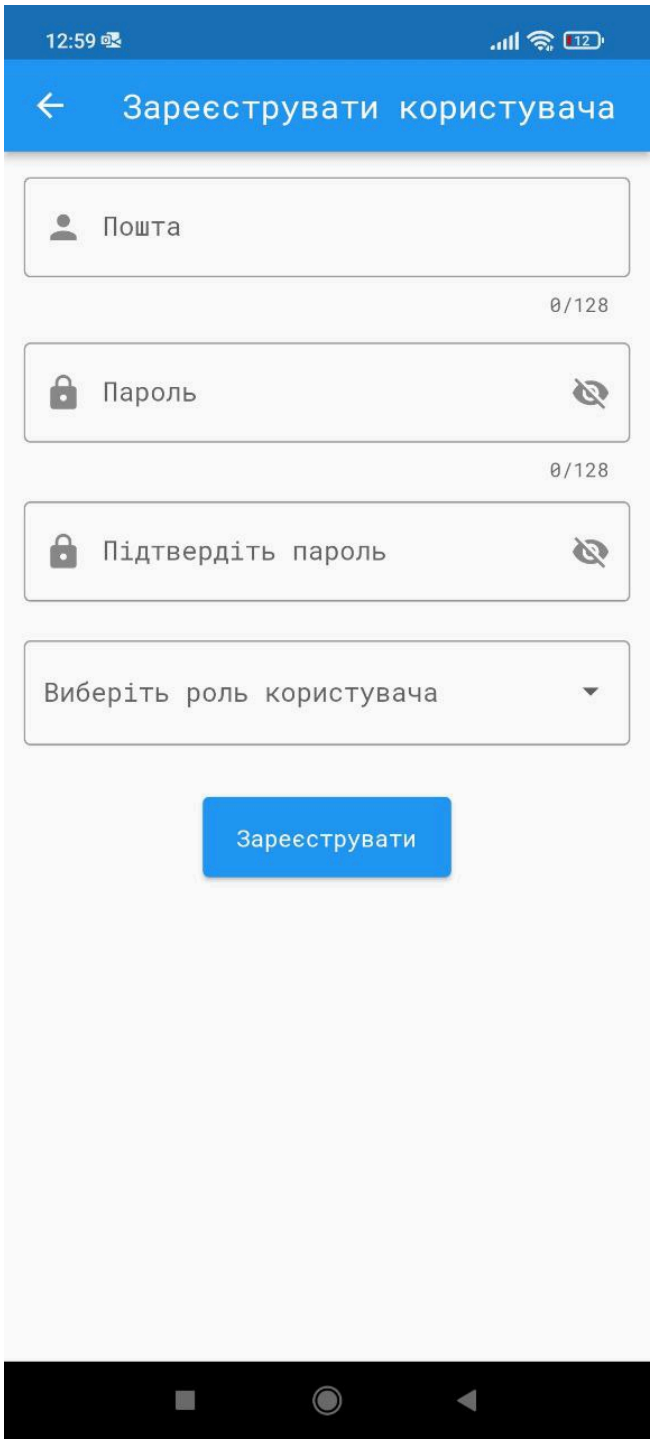

Рис. 2.17 Сторінка реєстрації користувача

Для того, щоб створити нового користувача на сторінці реєстрації (2.17) необхідно заповнити всі поля та натиснути на кнопку "Зареєструвати". На відміну від студентів, у вчителя не з’явиться поле "Група". Всі інші поля будуть ідентичними.

Для всіх ролей на головній сторінці програми присутня кнопка "Налаштування". При натисканні на неї користувача перенесе на сторінку налаштувань (рис. 2.18).

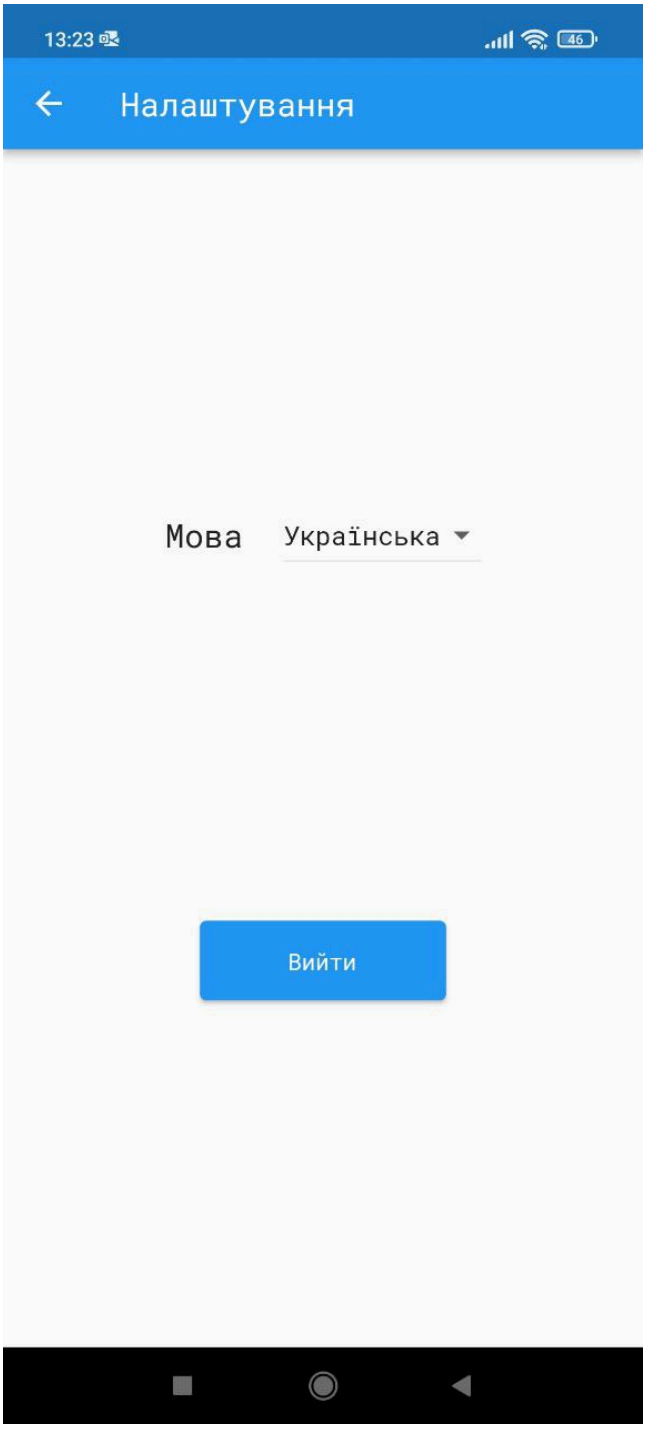

Рис. 2.18. Сторінка налаштувань

На сторінці налаштувань(рис 2.18) можна змінити налаштування мови. В даній версії програми доступно 2 мови - англійська. При першому запуску програми локалізація налаштується виходячи з локалізації смартфону на якому запускається програма. При натисканні на кнопку "Вийти" буде наданий запит до Firebase Authentication щодо виходу з аккаунту та користувача перенесе на сторінку авторизації (рис. 2.11).

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1 Розрахунок трудомісткості та вартості розробки програмного продукту

Ускладнення процесу нормування праці під час створення програмного забезпечення пояснюється творчим характером роботи програміста. Внаслідок цього, для оцінки трудомісткості розробки ПЗ може бути використана система моделей з різним рівнем точності.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C(1 + p), \text{ де} \quad (3.2)$$

q– передбачуване число операторів;

C– коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 2740 * 1,3(1 + 0.15) = 4\ 096,3;$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85)K}, \text{ людино-годин,} \tag{3.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, В=1,3;

K – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності становить 1,1;

$$t_u = (4\ 096,3 * 1.3) / (80 * 1.1) = 60.51, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)K} \tag{3.4}$$

$$t_a = 4\ 096,3 \ / 20 * 1.1 = 186.2, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \tag{3.5}$$

$$t_n = 4\ 096,3 \ / 22 * 1.1 = 169.27 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K} \tag{3.6}$$

$$t_{отл} = 4\ 096,3 / 5 * 1.1 = 744.78 \text{ людино-годин,}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1,2 \cdot t_{отл} \tag{3.7}$$

$t^{к}_{отл}$ = 1.2 * 744.78 = 893.74 людино-годин

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до} \quad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$t_{др} = \frac{Q}{(15...20)K} \quad (3.9)$$

$t_{др}$ = 4 096,3 / 15 * 1.1 = 300.4, людино-годин.

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др} \quad (3.10)$$

$t_{до}$ = 0.75 * 300.4 = 225.3, людино-годин.

$t_д$ = 300.4 + 225.3 = 525.7, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 + 60.51 + 186.2 + 169.27 + 744.78 + 525.7 = 1 736.46, людино-годин.

У результаті ми розрахували, що в загальній складності необхідно = 1 736.46 людино-годин для розробки даного програмного забезпечення.

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ $K_{по}$ включають витрати на заробітну плату виконавця програми $З_{зп}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн}, \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн}, \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$C_{ПР}$ – середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 1\ 736.46 * 125.83 = 218\ 498.76 \text{ грн.}$$

$З_{МВ}$ – Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot C_{М}, \text{ грн}, \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$C_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 1\ 736.46 * 20 = 34\ 729.2 \text{ грн.}$$

$$K_{ПО} = 218\ 498.76 + 34\ 729.2 = 253\ 227.96 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.} \quad (3.14)$$

де B_k - число виконавців;

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = 1\ 736.46 / 1 * 176 = 9{,}87 \text{ міс.}$$

Висновок: Час розробки даного програмного забезпечення складає 1 736.46 людино-годин. Таким чином, очікувана тривалість розробки складе 9,87 місяця при 40 годинному робочому тижні (місячний фонд робочого часу 176 годин), а витрати на створення програмного забезпечення складатимуть 253 227.96 грн.

ВИСНОВКИ

В процесі виконання завдання ми докладно вивчили предметну галузь, що дозволило нам краще розуміти вимоги та потреби користувачів електронного журналу оцінок студентів. Це було важливим кроком для ефективного розроблення додатку.

Планування та розробка користувацького інтерфейсу (UI) були виконані з врахуванням принципів дизайну та навігації, що забезпечило зручність взаємодії користувача з додатком. Користувальницький досвід був максимально оптимізований для досягнення мети додатку.

Написання коду програми відбувалося з дотриманням принципів SOLID та використанням Clean Architecture. Це дозволило нам створити модульну, легко розширювану та тестовану архітектуру додатку. Чітка роздільність між шаровими компонентами забезпечує легкість управління кодом та збільшує його перевикористовуваність.

Написання yml-файлів та використання CI/CD за допомогою GitHub Actions дозволили автоматизувати процеси збірки, тестування та розгортання додатку. Це забезпечило стабільність та швидкість розробки, а також забезпечило якість коду та його відповідність вимогам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дослідження кросплатформного фреймворку flutter. Чи означатиме розквіт цієї технології зникнення нативної розробки на Android та iOS? Укладач Олександр Синельніков. Харківський національний університет імені В.Н. Каразіна. – 2019. – 6с.
2. Beginning Git and GitHub: A Comprehensive Guide to Version Control, Project Management, and Teamwork for the New Developer. Mariot Tsitoara, 2019. 308 p.
3. The SOLID Principles of Object-Oriented Programming Explained in Plain English. URL: https://www.freecodecamp.org/news/solid-principles-explained-in-plain-english/ (дата звернення: 13.06.2023).
4. The Clean Architecture—Beginner's Guide. URL: https://betterprogramming.pub/the-clean-architecture-beginners-guide-e4b7058c1165 (дата звернення: 14.06.2023).
5. Firebase Cloud Storage in Flutter | Flutter a Firebase Tutorial. URL: https://medium.com/flutter-community/firebase-cloud-storage-in-flutter-flutter-an-firebase-tutorial-c5de7835c6cd (дата звернення: 14.06.2023).
6. What is Firebase Auth? URL: https://www.cbtnuggets.com/blog/technology/programming/what-is-firebase-authentication (дата звернення 14.06.2023).
7. The difference between stateless and stateful widgets in Flutter. URL: https://blog.logrocket.com/difference-between-stateless-stateful-widgets-flutter/ (дата звернення: 14.06.2023).
8. Getting Started With the BLoC Pattern. URL: https://www.kodeco.com/31973428-getting-started-with-the-bloc-pattern (дата звернення 15.06.2023).
9. How to Parse JSON in Dart/Flutter with Code Generation using Freezed. URL: https://codewithandrea.com/articles/parse-json-dart-codegen-freezed/ (дата звернення: 15.06.2023).

10. Flutter: Pragmatic State Management Using Provider. URL: https://medium.com/flutter-community/flutter-pragmatic-state-management-using-provider-5c1129f9b5bb (дата звернення: 15.06.2023).
11. Stream and RxDart in Flutter. URL: https://levelup.gitconnected.com/streams-and-rxdart-in-flutter-9135ef64912e (дата звернення: 15.06.2023).
12. Using SharedPreferences in Flutter to store data locally. URL: https://blog.logrocket.com/using-sharedpreferences-in-flutter-to-store-data-locally/ (дата звернення: 16.06.2023).
13. CI/CD For flutter using Github actions and Fastlane. URL: https://levelup.gitconnected.com/ci-cd-for-flutter-using-github-actions-and-fastlane-6dfc9431ee9a (дата звернення: 16.06.2023).
14. Internationalization for Flutter apps. URL: https://blog.logrocket.com/internationalization-flutter-apps/ (дата звернення: 16.06.2023).
15. Creating an adaptive app with Flutter. URL: https://blog.logrocket.com/creating-adaptive-app-flutter/ (дата звернення: 14.06.2023).
16. Принципи побудови адаптивного інтерфейсу мультимедійних додатків. Укладач О.Б. Бережна. Харківський національний економічний університет ім. С. Кузнеця. – 2022. – 8с.
17. Handling Permissions in Flutter App| Permission Handlers | The right way to check for permissions. URL: https://medium.com/vijay-r/handling-permissions-in-flutter-app-permission-handlers-the-right-way-to-check-for-permissions-b9ab24ebcc30. (дата звернення: 16.06.2023).
18. Firebase for Flutter Developers: Authentication, Database and Storage Mastery. Rahul Agarwal, 2023. 98 p.
19. Inherited Widget In Flutter. URL: https://medium.flutterdevs.com/inherited-widget-in-flutter-604b0f009297 (дата звернення: 14.06.2023).
20. Flutter Navigator 2.0: Using go_router. URL: https://www.kodeco.com/28987851-flutter-navigator-2-0-using-go_router (дата звернення: 14.06.2023).
21. Flutter Succinctly. Ed Freitas, 2019. 129 p.

ДОДАТОК А

КОД ПРОГРАМИ

lib\core\app\di\injector.dart

```
import 'package:electronic_student_journal/feature/home/data/datasources/excel_local_data_source.dart';
import 'package:electronic_student_journal/feature/home/data/datasources/excel_local_data_source_impl.dart';
import 'package:electronic_student_journal/feature/home/data/datasources/firestore_remote_data_source.dart';
import 'package:electronic_student_journal/feature/home/data/datasources/firestore_remote_data_source_impl.dart';
import 'package:electronic_student_journal/feature/home/data/datasources/register_remote_data_source.dart';
import 'package:electronic_student_journal/feature/home/data/datasources/register_remote_data_source_impl.dart';
import 'package:electronic_student_journal/feature/home/data/repositories/excel_repository_impl.dart';
import 'package:electronic_student_journal/feature/home/data/repositories/firestore_repository_impl.dart';
import 'package:electronic_student_journal/feature/home/data/repositories/register_repository_impl.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/excel_repository.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/firestore_repository.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/locale_repository.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/locale_repository_impl.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/register_repository.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/create_table_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/delete_table_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/export_table_to_excel_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/find_students_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/get_locale_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/get_scores_tables_list_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/get_scores_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/get_user_changes_stream_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/get_user_data_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/get_users_data_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/register_user_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/set_locale_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/sign_out_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/update_access_time_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/update_table_usecase.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/blocs/user_changes_bloc.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/create_table_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/delete_table_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/export_to_excel_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/find_students_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_scores_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_scores_tables_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_user_data_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_users_data_cubit_dart_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/locale_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/register_user_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/sign_out_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/update_table_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/group_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/name_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/password_confirmer_hinter.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/password_confirmer_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/patronymic_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/role_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/score_name_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/scores_table_name_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/show_confirm_score_button.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/show_student_search_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/surname_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/university_provider.dart';
import 'package:electronic_student_journal/feature/shared/data/datasources/firebase_auth_remote_data_source.dart';
import 'package:electronic_student_journal/feature/shared/data/datasources/firebase_auth_remote_data_source_impl.dart';
import 'package:electronic_student_journal/feature/shared/data/repositories/auth_repository_impl.dart';
import 'package:electronic_student_journal/feature/shared/domain/repositories/auth_repository.dart';
import 'package:electronic_student_journal/feature/sign_in/domain/usecases/sign_in_usecase.dart';
```

```
import 'package:electronic_student_journal/feature/sign_in/presentation/viewmodels/email_provider.dart';
import 'package:electronic_student_journal/feature/sign_in/presentation/viewmodels/password_hinter.dart';
import 'package:electronic_student_journal/feature/sign_in/presentation/viewmodels/password_provider.dart';
import 'package:electronic_student_journal/feature/sign_in/presentation/viewmodels/sign_in_cubit.dart';
import 'package:get_it/get_it.dart';
import 'package:logger/logger.dart';

final injector = GetIt.instance;

void initDependencies() {
  // Data sources
  injector
    ..registerLazySingleton<FirebaseAuthRemoteDataSource>(
      FirebaseAuthRemoteDataSourceImpl.new,
    )
    ..registerLazySingleton<RegisterRemoteDataSource>(
      RegisterRemoteDataSourceImpl.new,
    )
    ..registerLazySingleton<FirestoreRemoteDataSource>(
      FirestoreRemoteDataSourceImpl.new,
    )
    ..registerLazySingleton<ExcelLocalDataSource>(ExcelLocalDataSourceImpl.new)

    // Repositories
    ..registerLazySingleton<AuthRepository>(
      () => AuthRepositoryImpl(firebaseRemoteDataSource: injector()),
    )
    ..registerLazySingleton<RegisterRepository>(
      () => RegisterRepositoryImpl(registerRemoteDataSource: injector()),
    )
    ..registerLazySingleton<FirestoreRepository>(
      () => FirestoreRepositoryImpl(firestoreRemoteDataSource: injector()),
    )
    ..registerLazySingleton<ExcelRepository>(
      () => ExcelRepositoryImpl(excelLocalDataSource: injector()),
    )
    ..registerLazySingleton<LocaleRepository>(LocaleRepositoryImpl.new)

    // Usecases
    ..registerLazySingleton(() => SignInUseCase(authRepository: injector()))
    ..registerLazySingleton(() => SignOutUseCase(authRepository: injector()))
    ..registerLazySingleton(
      () => GetUserChangesStreamUseCase(authRepository: injector()),
    )
    ..registerLazySingleton(
      () => GetUserDataUsecase(firestoreRepository: injector()),
    )
    ..registerLazySingleton(
      () => RegisterUserUsecase(registerRepository: injector()),
    )
    ..registerLazySingleton(
      () => UpdateAccessTimeUseCase(firestoreRepository: injector()),
    )
    ..registerLazySingleton(
      () => GetScoresTablesUseCase(firestoreRepository: injector()),
    )
    ..registerLazySingleton(
      () => GetScoresUseCase(firestoreRepository: injector()),
    )
    ..registerLazySingleton(
      () => ExportTableToExcelUseCase(excelRepository: injector()),
    )
    ..registerLazySingleton(
      () => GetUsersDataUseCase(firestoreRepository: injector()),
    )
    ..registerLazySingleton(
      () => GetLocaleUseCase(localeRepository: injector()),
    )
    ..registerLazySingleton(
```

```
    () => SetLocaleUseCase(localeRepository: injector()),
  )
  ..registerLazySingleton(
    () => DeleteTableUseCase(firestoreRepository: injector()),
  )
  ..registerLazySingleton(
    () => CreateTableUseCase(firestoreRepository: injector()),
  )
  ..registerLazySingleton(
    () => UpdateTableUseCase(firestoreRepository: injector()),
  )
  ..registerLazySingleton(
    () => FindStudentsUseCase(firestoreRepository: injector()),
  )

  // Cubits
  ..registerFactory(() => SignInCubit(signInUseCase: injector()))
  ..registerFactory(() => SignOutCubit(signOutUseCase: injector()))
  ..registerFactory(() => GetUserDataCubit(getUserDataUsecase: injector()))
  ..registerFactory(() => RegisterUserCubit(registerUserUsecase: injector()))
  ..registerFactory(
    () => GetScoresTablesCubit(getScoresTablesUseCase: injector()),
  )
  ..registerFactory(() => GetScoresCubit(getScoresUseCase: injector()))
  ..registerFactory(
    () => ExportToExcelCubit(exportTableToExcelUseCase: injector()),
  )
  ..registerFactory(() => GetUsersDataCubit(getUsersDataUseCase: injector()))
  ..registerFactory(
    () => LocaleCubit(
      getLocaleUseCase: injector(),
      setLocaleUseCase: injector(),
    ),
  )
  ..registerFactory(() => DeleteTableCubit(deleteTableUseCase: injector()))
  ..registerFactory(() => CreateTableCubit(createTableUseCase: injector()))
  ..registerFactory(() => UpdateTableCubit(updateTableUseCase: injector()))
  ..registerFactory(() => FindStudentsCubit(findStudentsUseCase: injector()))

  // BLoCs
  ..registerFactory(
    () => UserChangesBloc(
      getUserChangesStreamUseCase: injector(),
      updateAccessTimeUseCase: injector(),
    ),
  )

  // Providers
  ..registerFactory(EmailProvider.new)
  ..registerFactory(PasswordProvider.new)
  ..registerFactory(PasswordHinter.new)
  ..registerFactory(PasswordConfirmerProvider.new)
  ..registerFactory(PasswordConfirmerHinter.new)
  ..registerFactory(RoleProvider.new)
  ..registerFactory(SurnameProvider.new)
  ..registerFactory(NameProvider.new)
  ..registerFactory(PatronymicProvider.new)
  ..registerFactory(UniversityProvider.new)
  ..registerFactory(GroupProvider.new)
  ..registerFactory(ScoresTableNameProvider.new)
  ..registerFactory(ScoreNameProvider.new)
  ..registerFactory(ShowStudentSearchProvider.new)
  ..registerFactory(ShowConfirmScoreButton.new)

  // Logger
  ..registerLazySingleton(Logger.new);
}
```

lib\core\app\my_app.dart

```
import 'package:electronic_student_journal/core/app/di/injector.dart';
import 'package:electronic_student_journal/core/app/router/app_router.dart';
import 'package:electronic_student_journal/core/theme/light_theme.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/locale_cubit.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:flutter_localizations/flutter_localizations.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:logger/logger.dart';
import 'package:overlay_support/overlay_support.dart';

class MyApp extends StatelessWidget {
 MyApp({super.key});

 final _logger = injector<Logger>();

 @override
 Widget build(BuildContext context) {
  return OverlaySupport.global(
   child: ScreenUtilInit(
    designSize: const Size(390, 844),
    minTextAdapt: true,
    splitScreenMode: true,
    builder: (context, child) {
     return BlocBuilder<LocaleCubit, LocaleState>(
      builder: (context, state) {
       Locale? locale;

       state.whenOrNull(
        loaded: (loadedLocale) => locale = Locale(loadedLocale),
        saved: (savedLocale) {
         locale = Locale(savedLocale);
         _logger.i(locale);
        },
        failure: (message) {
         _logger.w(message);
        },
       );

       return MaterialApp.router(
        title: 'Electronic student journal',
        theme: lightTheme,
        locale: locale,
        localizationsDelegates: const [
         AppLocalizations.delegate,
         GlobalMaterialLocalizations.delegate,
         GlobalWidgetsLocalizations.delegate,
         GlobalCupertinoLocalizations.delegate,
        ],
        supportedLocales: AppLocalizations.supportedLocales,
        routerConfig: appRouter,
        builder: (context, child) {
         return MediaQuery(
          data: MediaQuery.of(context),
          child: child!,
         );
        },
       );
      },
     );
    },
   ),
  );
 }
}
```

lib\main.dart

```
import 'package:electronic_student_journal/core/app/di/injector.dart';
import 'package:electronic_student_journal/core/app/my_app.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/locale_cubit.dart';
import 'package:electronic_student_journal/firebase_options.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);
  initDependencies();
  runApp(
    BlocProvider<LocaleCubit>(
      create: (context) => injector()..loadLocale(),
      child: MyApp(),
    ),
  );
}
```

lib\firebase_options.dart

```
// File generated by FlutterFire CLI.
// ignore_for_file: lines_longer_than_80_chars, avoid_classes_with_only_static_members
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      return web;
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        return ios;
      case TargetPlatform.macOS:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for macos - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      case TargetPlatform.windows:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for windows - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      case TargetPlatform.linux:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for linux - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      default:
        throw UnsupportedError(
          'DefaultFirebaseOptions are not supported for this platform.',
        );
    }
```

```
  }

  static const FirebaseOptions web = FirebaseOptions(
    apiKey: 'AIzaSyAV45CKIDZ14IyRPlODmlCm2FyIbx5n7E0',
    appId: '1:340213947966:web:87433d7e3a635ed8d93b6f',
    messagingSenderId: '340213947966',
    projectId: 'electronic-student-journal',
    authDomain: 'electronic-student-journal.firebaseapp.com',
    storageBucket: 'electronic-student-journal.appspot.com',
    measurementId: 'G-89294D3MR8',
  );

  static const FirebaseOptions android = FirebaseOptions(
    apiKey: 'AIzaSyCEe1aU_lRiF0Zb8LRtNLtml0bz2LH8uNc',
    appId: '1:340213947966:android:1c7ceea506685203d93b6f',
    messagingSenderId: '340213947966',
    projectId: 'electronic-student-journal',
    storageBucket: 'electronic-student-journal.appspot.com',
  );

  static const FirebaseOptions ios = FirebaseOptions(
    apiKey: 'AIzaSyARFEHkIUWjFWnzKFLQkTVT8NeXxKJwzDU',
    appId: '1:340213947966:ios:4d15301fb46a84ebd93b6f',
    messagingSenderId: '340213947966',
    projectId: 'electronic-student-journal',
    storageBucket: 'electronic-student-journal.appspot.com',
    iosClientId:
        '340213947966-ho26n86siigpp8r3iglcu4j5j1qnvdap.apps.googleusercontent.com',
    iosBundleId: 'com.example.electronicStudentJournalSimple',
  );
}
```

lib\core\error\exception.dart

```
class ServerException implements Exception {}

class AdminSelectedException implements Exception {}

class EmptyDataException implements Exception {}

class EmptyLocaleException implements Exception {}
```

lib\core\error\failure.dart

```
import 'package:equatable/equatable.dart';

class Failure extends Equatable {
  const Failure(this.message);

  final String message;

  @override
  List<Object> get props => [message];
}

class ServerFailure extends Failure {
  const ServerFailure(super.message);
}

class ConnectionFailure extends Failure {
  const ConnectionFailure(super.message);
}

class EmptyDataFailure extends Failure {
  const EmptyDataFailure(super.message);
}

class AdminSelectedFailure extends Failure {
  const AdminSelectedFailure(super.message);
```

```
}

class EmptyLocaleFailure extends Failure {
  const EmptyLocaleFailure(super.message);
}

class SomeFailure extends Failure {
  const SomeFailure(super.message);
}
```

lib\core\l10n\app_en.arb

```
{
  "@@locale": "en",

  "emailLabelText": "Email",
  "emailHintText": "Enter your email",

  "passwordLabelText": "Password",
  "passwordHintText": "Enter your password",

  "confirmPasswordLabelText": "Confirm Password",
  "confirmPasswordHintText": "Enter password again",

  "invalidEmail": "Invalid Email Address",
  "invalidPassword": "Password must be at least 6 characters",
  "invalidConfirmPassword": "Passwords don't match",

  "signInButtonText": "Sign in",
  "signOutButtonText": "Sign out",

  "appNameTitle": "Electronic Journal",
  "signUpUser": "Sign up user",
  "settings": "Settings",

  "selectUserRoleHintText": "Select user role",
  "invalidUserRole": "Please select user role",

  "signUpUserButtonText": "Sign up",

  "scores": "Scores",

  "student": "student",
  "teacher": "teacher",
  "admin": "admin",

  "patronymicLabelText": "Patronymic",
  "patronymicErrorText": "Enter patronymic",
  "nameLabelText": "Name",
  "nameErrorText": "Enter name",
  "surnameLabelText": "Surname",
  "surnameErrorText": "Enter surname",
  "groupLabelText": "Group",
  "groupErrorText": "Enter group",
  "universityLabelText": "University",
  "universityErrorText": "Enter university's name",

  "create": "Create",
  "tableTitle": "Table",

  "tableNameLabelText": "Table name",
  "tableNameErrorText": "Enter table name",

  "language": "Language",
  "enLang": "English",
  "uaLang": "Ukrainian",

  "exportToExcel": "Export to Excel",
  "updateTable": "Update table",
```

```
  "deleteTable": "Delete table",
  "addStudent": "Add student"
}
```

lib\core\l10n\app_uk.arb

```
{
  "@@locale": "uk",

  "emailLabelText": "Пошта",
  "emailHintText": "Введіть адресу електронної пошти",

  "passwordLabelText": "Пароль",
  "passwordHintText": "Введіть пароль",

  "confirmPasswordLabelText": "Підтвердіть пароль",
  "confirmPasswordHintText": "Повторно введіть пароль",

  "invalidEmail": "Невірна адреса пошти",
  "invalidPassword": "Пароль повинен мати довжину не менше 6 символів",
  "invalidConfirmPassword": "Паролі не співпадають",

  "signInButtonText": "Увійти",
  "signOutButtonText": "Вийти",

  "appNameTitle": "Електронний журнал",
  "signUpUser": "Зареєструвати користувача",
  "settings": "Налаштування",

  "selectUserRoleHintText": "Виберіть роль користувача",
  "invalidUserRole": "Будь ласка, оберіть роль користувача",

  "signUpUserButtonText": "Зареєструвати",

  "scores": "Оцінки",

  "student": "студент",
  "teacher": "викладач",
  "admin": "адмін",

  "patronymicLabelText": "По-батькові",
  "patronymicErrorText": "Введіть по-батькові",
  "nameLabelText": "Ім'я",
  "nameErrorText": "Введіть ім'я",
  "surnameLabelText": "Прізвище",
  "surnameErrorText": "Введіть прізвище",
  "groupLabelText": "Група",
  "groupErrorText": "Введіть групу",
  "universityLabelText": "Університет",
  "universityErrorText": "Введіть назву університету",

  "create": "Створити",
  "tableTitle": "Таблиця",

  "tableNameLabelText": "Назва таблиці",
  "tableNameErrorText": "Введіть назву таблиці",

  "language": "Мова",
  "enLang": "Англійська",
  "uaLang": "Українська",

  "exportToExcel": "Експортувати в Excel",
  "updateTable": "Оновити таблицю",
  "deleteTable": "Видалити таблицю",
  "addStudent": "Додати студента"
}
```

lib\core\theme\light_theme.dart

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

final lightTheme = ThemeData(
  textTheme: GoogleFonts.robotoMonoTextTheme(),
  inputDecorationTheme: const InputDecorationTheme(
    border: OutlineInputBorder(borderSide: BorderSide(width: 3)),
  ),
);
```

lib\core\usecase\usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:equatable/equatable.dart';

abstract class AsyncUseCase<Type, Params> {
  Future<Either<Failure, Type>> call(Params params);
}

abstract class SyncUseCase<Type, Params> {
  Either<Failure, Type> call(Params params);
}

class NoParams extends Equatable {
  @override
  List<Object?> get props => [];
}
```

lib\feature\home\data\converters\timestamp_converter.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:json_annotation/json_annotation.dart';

class TimestampConverter implements JsonConverter<DateTime, Timestamp> {
  const TimestampConverter();

  @override
  DateTime fromJson(Timestamp json) => json.toDate();

  @override
  Timestamp toJson(DateTime object) => Timestamp.fromDate(object);
}
```

lib\feature\home\data\converters\timestamp_nullable_converter.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

class TimestampNullableConverter
    implements JsonConverter<DateTime?, Timestamp?> {
  const TimestampNullableConverter();

  @override
  DateTime? fromJson(Timestamp? json) => json?.toDate();

  @override
  Timestamp? toJson(DateTime? object) =>
      object == null ? null : Timestamp.fromDate(object);
}
```

lib\feature\home\data\datasources\excel_local_data_source_impl.dart

```
import 'dart:io';
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/data/datasources/excel_local_data_source.dart';
import 'package:electronic_student_journal/feature/home/domain/params/exporting_table_params.dart';
import 'package:excel/excel.dart';
```

```
import 'package:path_provider/path_provider.dart';

class ExcelLocalDataSourceImpl implements ExcelLocalDataSource {
  @override
  Future<Either<Failure, void>> exportToExcel(
    ExportingTableParams params,
  ) async {
    try {
      final excel = Excel.createExcel();
      final sheet = excel.sheets[excel.getDefaultSheet()];

      for (var i = 0; i < params.rows; i++) {
        for (var j = 0; j < params.cols; j++) {
          sheet!
              .cell(CellIndex.indexByColumnRow(columnIndex: j, rowIndex: i))
              .value = params.content[i * params.cols + j];
        }
      }

      final fileBytes = excel.save(fileName: '${params.tableName}.xlsx');

      final String downloadPath;
      if (Platform.isAndroid) {
        downloadPath = '/storage/emulated/0/Download/';
      } else {
        final directory = await getDownloadsDirectory();
        downloadPath = directory!.path;
      }

      File('$downloadPath/${params.tableName}.xlsx')
        ..createSync(recursive: true)
        ..writeAsBytesSync(fileBytes!);

      return const Right(null);
    } catch (e) {
      return Left(SomeFailure(e.toString()));
    }
  }
}
```

lib\feature\home\data\datasources\excel_local_data_source.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/domain/params/exporting_table_params.dart';

abstract interface class ExcelLocalDataSource {
  Future<Either<Failure, void>> exportToExcel(ExportingTableParams params);
}
```

lib\feature\home\data\datasources\firestore_remote_data_source_impl.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/exception.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/data/datasources/firestore_remote_data_source.dart';
import 'package:electronic_student_journal/feature/home/data/models/score_model.dart';
import 'package:electronic_student_journal/feature/home/data/models/scores_table_model.dart';
import 'package:electronic_student_journal/feature/home/data/models/user_model.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/edit_table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/find_students_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_model_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/users_params.dart';

class FirestoreRemoteDataSourceImpl implements FirestoreRemoteDataSource {
```

```
final _firebaseFirestore = FirebaseFirestore.instance;

@override
Future<Either<Failure, UserModel>> getUserData(String uid) async {
  try {
    final doc = await _firebaseFirestore.collection('users').doc(uid).get();
    if (doc.data() != null) {
      return Right(UserModel.fromJson(doc.data()!));
    }
    return Left(EmptyDataFailure('Empty data: ${doc.data()}'));
  } catch (e) {
    return Left(SomeFailure(e.toString()));
  }
}

@override
Future<Either<Failure, void>> updateAccessTime(UserParams params) async {
  try {
    await _firebaseFirestore
        .collection('users')
        .doc(params.uid)
        .update({'lastAccessed': DateTime.now()});
    return const Right(null);
  } catch (e) {
    return Left(SomeFailure(e.toString()));
  }
}

@override
Future<Either<Failure, List<ScoresTableModel>>> getScoresTables(
  UserModelParams params,
) async {
  try {
    if (params.user.role == UserRole.student) {
      final tablesSnapshot =
          await _firebaseFirestore.collection('scores_tables').get();

      final tables = <ScoresTableModel>[];

      for (final doc in tablesSnapshot.docs) {
        final scoresSnapshot = await _firebaseFirestore
            .doc(doc.reference.path)
            .collection('scores')
            .where('studentUid', isEqualTo: params.user.uid)
            .get();
        if (scoresSnapshot.docs.isNotEmpty) {
          tables.add(ScoresTableModel.fromJson(doc.data()));
        }
      }

      return Right(tables);
    } else if (params.user.role == UserRole.teacher) {
      final tablesSnapshot = await _firebaseFirestore
          .collection('scores_tables')
          .where('ownerUid', isEqualTo: params.user.uid)
          .get();
      final tables = tablesSnapshot.docs
          .map((doc) => ScoresTableModel.fromJson(doc.data()))
          .toList();

      return Right(tables);
    }
    throw AdminSelectedException();
  } on AdminSelectedException catch (e) {
    return Left(AdminSelectedFailure(e.toString()));
  } catch (e) {
    return Left(SomeFailure(e.toString()));
  }
}
```

```
@override
Future<Either<Failure, List<ScoreModel>>> getScores(
  TableParams params,
) async {
  final tableSnapshot = await _firebaseFirestore
      .collection('scores_tables')
      .where('uid', isEqualTo: params.uid)
      .get();

  final scoresSnapshot = await _firebaseFirestore
      .doc(tableSnapshot.docs.first.reference.path)
      .collection('scores')
      .orderBy('date')
      .get();

  final scores = scoresSnapshot.docs
      .map((doc) => ScoreModel.fromJson(doc.data()))
      .toList();

  return Right(scores);
}

@override
Future<Either<Failure, List<UserModel>>> getUsersData(
  UsersParams params,
) async {
  try {
    final usersModels = <UserModel>[];

    final uids = params.uids;
    for (final uid in uids) {
      final doc = await _firebaseFirestore.collection('users').doc(uid).get();
      if (doc.data() != null) {
        usersModels.add(UserModel.fromJson(doc.data()!));
      } else {
        throw EmptyDataException();
      }
    }

    return Right(usersModels);
  } on EmptyDataException catch (e) {
    return Left(EmptyDataFailure(e.toString()));
  } catch (e) {
    return Left(SomeFailure(e.toString()));
  }
}

@override
Future<Either<Failure, void>> createTable(EditTableParams params) async {
  try {
    final tableReference = await _firebaseFirestore
        .collection('scores_tables')
        .add(params.table.toJson());
    await tableReference.update({'uid': tableReference.id});

    final scoresReference = tableReference.collection('scores');
    for (final score in params.scores) {
      await scoresReference.add(score.toJson());
    }

    return const Right(null);
  } catch (e) {
    return Left(SomeFailure(e.toString()));
  }
}

@override
Future<Either<Failure, void>> deleteTable(TableParams params) async {
```

```
  try {
    final tableReference =
        _firebaseFirestore.collection('scores_tables').doc(params.uid);

    await _firebaseFirestore.runTransaction<Transaction>(
      (transaction) async => transaction.delete(tableReference),
    );

    return const Right(null);
  } catch (e) {
    return Left(SomeFailure(e.toString()));
  }
}

@override
Future<Either<Failure, void>> updateTable(EditTableParams params) async {
  try {
    // final tableReference =
    //     _firebaseFirestore.collection('scores_tables').doc(params.table.name);
    // await tableReference.set(params.table.toJson());

    // final scoresReference = tableReference.collection('scores');
    // final scoresSnapshot = await scoresReference.get();
    // for (final scoreSnapshot in scoresSnapshot.docs) {
    //   await _firebaseFirestore.runTransaction<Transaction>(
    //     (transaction) async => transaction.delete(scoreSnapshot.reference),
    //   );
    // }

    // for (final score in params.scores) {
    //   await scoresReference.add(score.toJson());
    // }

    await deleteTable(TableParams(uid: params.table.uid));
    await createTable(params);

    return const Right(null);
  } catch (e) {
    return Left(SomeFailure(e.toString()));
  }
}

@override
Future<Either<Failure, List<UserModel>>> findStudents(
  FindStudentsParams params,
) async {
  try {
    if (params.studentNameQuery.isEmpty) {
      return const Right([]);
    }

    final response = await _firebaseFirestore
        .collection('users')
        .where('role', isEqualTo: 'student')
        .where('university', isEqualTo: params.teacherUniversity)
        .get();

    final studentsSnapshot = response.docs;

    final students = studentsSnapshot
        .map((studentSnapshot) => UserModel.fromJson(studentSnapshot.data()))
        .where(
          (student) => student.fullName!
              .toLowerCase()
              .contains(params.studentNameQuery.toLowerCase()),
        )
        .toList();

    return Right(students);
```

```
    } catch (e) {
      return Left(SomeFailure(e.toString()));
    }
  }
}
```

lib\feature\home\data\datasources\firestore_remote_data_source.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/data/models/score_model.dart';
import 'package:electronic_student_journal/feature/home/data/models/scores_table_model.dart';
import 'package:electronic_student_journal/feature/home/data/models/user_model.dart';
import 'package:electronic_student_journal/feature/home/domain/params/edit_table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/find_students_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_model_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/users_params.dart';

abstract interface class FirestoreRemoteDataSource {
  Future<Either<Failure, UserModel>> getUserData(String uid);
  Future<Either<Failure, List<UserModel>>> getUsersData(
    UsersParams params,
  );
  Future<Either<Failure, void>> updateAccessTime(UserParams params);
  Future<Either<Failure, List<ScoresTableModel>>> getScoresTables(
    UserModelParams params,
  );
  Future<Either<Failure, List<ScoreModel>>> getScores(TableParams params);
  Future<Either<Failure, void>> createTable(EditTableParams params);
  Future<Either<Failure, void>> updateTable(EditTableParams params);
  Future<Either<Failure, void>> deleteTable(TableParams params);
  Future<Either<Failure, List<UserModel>>> findStudents(
    FindStudentsParams params,
  );
}
```

lib\feature\home\data\datasources\register_remote_data_source_impl.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/data/datasources/register_remote_data_source.dart';
import 'package:electronic_student_journal/feature/home/domain/params/register_params.dart';
import 'package:firebase_auth/firebase_auth.dart';

class RegisterRemoteDataSourceImpl implements RegisterRemoteDataSource {
  final _firebaseAuth = FirebaseAuth.instance;
  final _firebaseFirestore = FirebaseFirestore.instance;

  @override
  Future<Either<Failure, void>> registerUser(
    RegisterParams params,
  ) async {
    try {
      final userCredentials =
          await _firebaseAuth.createUserWithEmailAndPassword(
        email: params.user.email,
        password: params.password,
      );

      final userUid = userCredentials.user!.uid;
      final userData = params.user.copyWith(uid: userUid);

      await _firebaseFirestore
          .collection('users')
          .doc(userUid)
          .set(userData.toJson());
```

```
      return const Right(null);
    } on FirebaseAuthException catch (e) {
      return Left(ServerFailure(e.message ?? "Can't create this user"));
    } catch (e) {
      return Left(SomeFailure(e.toString()));
    }
  }
}
```

lib\feature\home\data\datasources\register_remote_data_source.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/domain/params/register_params.dart';

abstract interface class RegisterRemoteDataSource {
  Future<Either<Failure, void>> registerUser(RegisterParams params);
}
```

lib\feature\home\data\models\score_model.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:electronic_student_journal/feature/home/data/converters/timestamp_converter.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/score_entity.dart';
import 'package:flutter/foundation.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'score_model.freezed.dart';
part 'score_model.g.dart';

@freezed
class ScoreModel with _$ScoreModel {
  const factory ScoreModel({
    required String studentUid,
    required int score,
    @TimestampConverter() required DateTime date,
  }) = _ScoreModel;

  const ScoreModel._();

  factory ScoreModel.fromJson(Map<String, Object?> json) =>
      _$ScoreModelFromJson(json);

  ScoreEntity toEntity() => ScoreEntity(
        studentUid: studentUid,
        score: score,
        date: date,
      );
}
```

lib\feature\home\data\models\scores_table_model.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:electronic_student_journal/feature/home/data/converters/timestamp_converter.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/scores_table_entity.dart';
import 'package:flutter/foundation.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'scores_table_model.freezed.dart';
part 'scores_table_model.g.dart';

@freezed
class ScoresTableModel with _$ScoresTableModel {
  const factory ScoresTableModel({
    required String name,
    @TimestampConverter() required DateTime createdAt,
    required String ownerUid,
    required String uid,
  }) = _ScoresTableModel;
```

```
  const ScoresTableModel._();

  factory ScoresTableModel.fromJson(Map<String, Object?> json) =>
      _$ScoresTableModelFromJson(json);

  ScoresTableEntity toEntity() => ScoresTableEntity(
        name: name,
        createdAt: createdAt,
        ownerUid: ownerUid,
        uid: uid,
      );
}
```

lib\feature\home\data\models\user_model.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:electronic_student_journal/feature/home/data/converters/timestamp_converter.dart';
import 'package:electronic_student_journal/feature/home/data/converters/timestamp_nullable_converter.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'user_model.freezed.dart';
part 'user_model.g.dart';

@freezed
class UserModel with _$UserModel {
  const factory UserModel({
    required String uid,
    required String email,
    required String role,
    @TimestampConverter() required DateTime registeredAt,
    @TimestampNullableConverter() DateTime? lastAccessed,
    String? surname,
    String? name,
    String? patronymic,
    String? university,
    String? group,
    String? fullName,
  }) = _UserModel;

  const UserModel._();

  factory UserModel.fromJson(Map<String, dynamic> json) =>
      _$UserModelFromJson(json);

  UserEntity toEntity() => UserEntity(
        uid: uid,
        email: email,
        role: role,
        registeredAt: registeredAt,
        lastAccessed: lastAccessed,
        surname: surname,
        name: name,
        patronymic: patronymic,
        university: university,
        group: group,
        fullName: fullName,
      );
}
```

lib\feature\home\data\repositories\excel_repository_impl.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/data/datasources/excel_local_data_source.dart';
import 'package:electronic_student_journal/feature/home/domain/params/exporting_table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/excel_repository.dart';
```

```
class ExcelRepositoryImpl implements ExcelRepository {
  ExcelRepositoryImpl({required ExcelLocalDataSource excelLocalDataSource})
      : _excelLocalDataSource = excelLocalDataSource;

  final ExcelLocalDataSource _excelLocalDataSource;

  @override
  Future<Either<Failure, void>> exportToExcel(
    ExportingTableParams params,
  ) async {
    final response = await _excelLocalDataSource.exportToExcel(params);

    return response;
  }
}
```

lib\feature\home\data\repositories\firestore_repository_impl.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/data/datasources/firestore_remote_data_source.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/score_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/scores_table_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/edit_table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/find_students_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_model_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/users_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/firestore_repository.dart';

class FirestoreRepositoryImpl implements FirestoreRepository {
  FirestoreRepositoryImpl({
    required FirestoreRemoteDataSource firestoreRemoteDataSource,
  }) : _firestoreRemoteDataSource = firestoreRemoteDataSource;

  final FirestoreRemoteDataSource _firestoreRemoteDataSource;

  @override
  Future<Either<Failure, UserEntity>> getUserData(String uid) async {
    final response = await _firestoreRemoteDataSource.getUserData(uid);

    return response.fold(
      Left.new,
      (userModel) => Right(userModel.toEntity()),
    );
  }

  @override
  Future<Either<Failure, void>> updateAccessTime(UserParams params) async {
    final response = await _firestoreRemoteDataSource.updateAccessTime(params);

    return response;
  }

  @override
  Future<Either<Failure, List<ScoresTableEntity>>> getScoresTables(
    UserModelParams params,
  ) async {
    final response = await _firestoreRemoteDataSource.getScoresTables(params);

    return response.fold(
      Left.new,
      (scoresTables) =>
          Right(scoresTables.map((table) => table.toEntity()).toList()),
    );
  }
```

```
  @override
  Future<Either<Failure, List<ScoreEntity>>> getScores(
    TableParams params,
  ) async {
    final response = await _firestoreRemoteDataSource.getScores(params);

    return response.fold(
      Left.new,
      (scores) => Right(scores.map((score) => score.toEntity()).toList()),
    );
  }

  @override
  Future<Either<Failure, List<UserEntity>>> getUsersData(
    UsersParams usersParams,
  ) async {
    final response = await _firestoreRemoteDataSource.getUsersData(usersParams);

    return response.fold(
      Left.new,
      (usersModels) => Right(
        usersModels.map((userModel) => userModel.toEntity()).toList(),
      ),
    );
  }

  @override
  Future<Either<Failure, void>> deleteTable(TableParams params) async {
    final response = await _firestoreRemoteDataSource.deleteTable(params);

    return response;
  }

  @override
  Future<Either<Failure, void>> createTable(EditTableParams params) async {
    final response = await _firestoreRemoteDataSource.createTable(params);

    return response;
  }

  @override
  Future<Either<Failure, void>> updateTable(EditTableParams params) async {
    final response = await _firestoreRemoteDataSource.updateTable(params);

    return response;
  }

  @override
  Future<Either<Failure, List<UserEntity>>> findStudents(
    FindStudentsParams params,
  ) async {
    final response = await _firestoreRemoteDataSource.findStudents(params);

    return response.fold(
      Left.new,
      (students) =>
          Right(students.map((student) => student.toEntity()).toList()),
    );
  }
}
```

lib\feature\home\data\repositories\register_repository_impl.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/data/datasources/register_remote_data_source.dart';
import 'package:electronic_student_journal/feature/home/domain/params/register_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/register_repository.dart';
```

```
class RegisterRepositoryImpl implements RegisterRepository {
  RegisterRepositoryImpl({
    required RegisterRemoteDataSource registerRemoteDataSource,
  }) : _registerRemoteDataSource = registerRemoteDataSource;

  final RegisterRemoteDataSource _registerRemoteDataSource;

  @override
  Future<Either<Failure, void>> registerUser(
    RegisterParams params,
  ) async {
    final response = await _registerRemoteDataSource.registerUser(params);

    return response;
  }
}
```

lib\feature\home\domain\entities\score_entity.dart

```
import 'package:electronic_student_journal/feature/home/data/models/score_model.dart';
import 'package:equatable/equatable.dart';

class ScoreEntity extends Equatable {
  const ScoreEntity({
    required this.studentUid,
    required this.score,
    required this.date,
  });

  final String studentUid;
  final int score;
  final DateTime date;

  ScoreModel toModel() => ScoreModel(
        studentUid: studentUid,
        score: score,
        date: date,
      );

  @override
  List<Object?> get props => [
        studentUid,
        score,
        date,
      ];
}
```

lib\feature\home\domain\entities\scores_table_entity.dart

```
import 'package:equatable/equatable.dart';

class ScoresTableEntity extends Equatable {
  const ScoresTableEntity({
    required this.name,
    required this.createdAt,
    required this.ownerUid,
    required this.uid,
  });

  final String name;
  final DateTime createdAt;
  final String ownerUid;
  final String uid;

  @override
  List<Object?> get props => [name, createdAt, ownerUid];
}
```

lib\feature\home\domain\entities\user_entity.dart

```
import 'package:electronic_student_journal/feature/home/data/models/user_model.dart';
import 'package:equatable/equatable.dart';

class UserRole {
  UserRole._();

  static const String student = 'student';
  static const String teacher = 'teacher';
  static const String admin = 'admin';
}

class UserEntity extends Equatable {
  const UserEntity({
    required this.uid,
    required this.email,
    required this.role,
    required this.registeredAt,
    this.lastAccessed,
    this.surname,
    this.name,
    this.patronymic,
    this.university,
    this.group,
    this.fullName,
  });

  final String uid;

  final String email;

  final String role;

  final DateTime registeredAt;

  final DateTime? lastAccessed;

  final String? surname;

  final String? name;

  final String? patronymic;

  final String? university;

  final String? group;

  final String? fullName;

  UserModel toModel() => UserModel(
        uid: uid,
        email: email,
        role: role,
        registeredAt: registeredAt,
        lastAccessed: lastAccessed,
        surname: surname,
        name: name,
        patronymic: patronymic,
        university: university,
        group: group,
        fullName: fullName,
      );

  String get fullNameWithInitials => '$surname ${name![0]}. ${patronymic![0]}.';

  @override
  List<Object?> get props => [
        uid,
        email,
```

```
      role,
      registeredAt,
      lastAccessed,
      surname,
      name,
      patronymic,
      university,
      group,
      fullName,
    ];
}
```

lib\feature\home\domain\params\edit_table_params.dart

```
import 'package:electronic_student_journal/feature/home/data/models/score_model.dart';
import 'package:electronic_student_journal/feature/home/data/models/scores_table_model.dart';
import 'package:flutter/foundation.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'edit_table_params.freezed.dart';

@freezed
class EditTableParams with _$EditTableParams {
  const factory EditTableParams({
    required ScoresTableModel table,
    required List<ScoreModel> scores,
  }) = _EditTableParams;
}
```

lib\feature\home\domain\params\exporting_table_params.dart

```
import 'package:flutter/foundation.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'exporting_table_params.freezed.dart';

@freezed
class ExportingTableParams with _$ExportingTableParams {
  const factory ExportingTableParams({
    required List<dynamic> content,
    required int rows,
    required int cols,
    required String tableName,
  }) = _ExportingTableParams;
}
```

lib\feature\home\domain\params\find_students_params.dart

```
import 'package:flutter/foundation.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'find_students_params.freezed.dart';

@freezed
class FindStudentsParams with _$FindStudentsParams {
  const factory FindStudentsParams({
    required String studentNameQuery,
    required String teacherUniversity,
  }) = _FindStudentsParams;
}
```

lib\feature\home\domain\params\locale_params.dart

```
import 'package:flutter/foundation.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'locale_params.freezed.dart';

@freezed
```

```
class LocaleParams with _$LocaleParams {
  const factory LocaleParams({
    required String locale,
  }) = _LocaleParams;
}
```

lib\feature\home\domain\params\register_params.dart

```
import 'package:electronic_student_journal/feature/home/data/models/user_model.dart';
import 'package:flutter/foundation.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'register_params.freezed.dart';

@freezed
class RegisterParams with _$RegisterParams {
  const factory RegisterParams({
    required UserModel user,
    required String password,
  }) = _RegisterParams;
}
```

lib\feature\home\domain\params\table_params.dart

```
import 'package:flutter/foundation.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'table_params.freezed.dart';

@freezed
class TableParams with _$TableParams {
  const factory TableParams({
    required String uid,
  }) = _TableParams;
}
```

lib\feature\home\domain\params\user_model_params.dart

```
import 'package:electronic_student_journal/feature/home/data/models/user_model.dart';
import 'package:flutter/foundation.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'user_model_params.freezed.dart';

@freezed
class UserModelParams with _$UserModelParams {
  const factory UserModelParams({
    required UserModel user,
  }) = _UserModelParams;
}
```

lib\feature\home\domain\params\user_params.dart

```
import 'package:flutter/foundation.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'user_params.freezed.dart';

@freezed
class UserParams with _$UserParams {
  const factory UserParams({
    required String uid,
  }) = _UserParams;
}
```

lib\feature\home\domain\params\users_params.dart

```
import 'package:flutter/foundation.dart';
import 'package:freezed_annotation/freezed_annotation.dart';
```

```
part 'users_params.freezed.dart';

@freezed
class UsersParams with _$UsersParams {
  const factory UsersParams({
    required List<String> uids,
  }) = _UsersParams;
}
```

lib\feature\home\domain\repositories\excel_repository.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/domain/params/exporting_table_params.dart';

abstract interface class ExcelRepository {
  Future<Either<Failure, void>> exportToExcel(ExportingTableParams params);
}
```

lib\feature\home\domain\repositories\firestore_repository.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/score_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/scores_table_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/edit_table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/find_students_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_model_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_params.dart';
import 'package:electronic_student_journal/feature/home/domain/params/users_params.dart';

abstract interface class FirestoreRepository {
  Future<Either<Failure, UserEntity>> getUserData(String uid);
  Future<Either<Failure, void>> updateAccessTime(UserParams params);
  Future<Either<Failure, List<ScoresTableEntity>>> getScoresTables(
    UserModelParams params,
  );
  Future<Either<Failure, List<ScoreEntity>>> getScores(TableParams params);
  Future<Either<Failure, List<UserEntity>>> getUsersData(
    UsersParams usersParams,
  );
  Future<Either<Failure, void>> deleteTable(TableParams params);
  Future<Either<Failure, void>> createTable(EditTableParams params);
  Future<Either<Failure, void>> updateTable(EditTableParams params);
  Future<Either<Failure, List<UserEntity>>> findStudents(
    FindStudentsParams params,
  );
}
```

lib\feature\home\domain\repositories\locale_repository_impl.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/exception.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/domain/params/locale_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/locale_repository.dart';
import 'package:shared_preferences/shared_preferences.dart';

class LocaleRepositoryImpl implements LocaleRepository {
  final Future<SharedPreferences> _prefs = SharedPreferences.getInstance();

  @override
  Future<Either<Failure, String>> getLocale() async {
    try {
      final prefs = await _prefs;
      final locale = prefs.getString('locale');
      if (locale == null) {
```

```
        throw EmptyLocaleException();
      }
      return Right(locale);
    } on EmptyLocaleException {
      return const Left(EmptyLocaleFailure('No locale found.'));
    } catch (e) {
      return Left(SomeFailure(e.toString()));
    }
  }

  @override
  Future<Either<Failure, String>> setLocale(LocaleParams params) async {
    try {
      final prefs = await _prefs;
      await prefs.setString('locale', params.locale);
      return Right(params.locale);
    } catch (e) {
      return Left(SomeFailure(e.toString()));
    }
  }
}
```

lib\feature\home\domain\repositories\locale_repository.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/domain/params/locale_params.dart';

abstract interface class LocaleRepository {
  Future<Either<Failure, String>> setLocale(LocaleParams params);
  Future<Either<Failure, String>> getLocale();
}
```

lib\feature\home\domain\repositories\register_repository.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/home/domain/params/register_params.dart';

abstract interface class RegisterRepository {
  Future<Either<Failure, void>> registerUser(RegisterParams params);
}
```

lib\feature\home\domain\usecases\create_table_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/params/edit_table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/firestore_repository.dart';

class CreateTableUseCase extends AsyncUseCase<void, EditTableParams> {
  CreateTableUseCase({required FirestoreRepository firestoreRepository})
      : _firestoreRepository = firestoreRepository;

  final FirestoreRepository _firestoreRepository;

  @override
  Future<Either<Failure, void>> call(EditTableParams params) {
    final response = _firestoreRepository.createTable(params);

    return response;
  }
}
```

lib\feature\home\domain\usecases\delete_table_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
```

```
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/params/table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/firestore_repository.dart';

class DeleteTableUseCase extends AsyncUseCase<void, TableParams> {
  DeleteTableUseCase({required FirestoreRepository firestoreRepository})
      : _firestoreRepository = firestoreRepository;

  final FirestoreRepository _firestoreRepository;

  @override
  Future<Either<Failure, void>> call(TableParams params) {
    final response = _firestoreRepository.deleteTable(params);

    return response;
  }
}
```

lib\feature\home\domain\usecases\export_table_to_excel_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/params/exporting_table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/excel_repository.dart';

class ExportTableToExcelUseCase
    extends AsyncUseCase<void, ExportingTableParams> {
  ExportTableToExcelUseCase({required ExcelRepository excelRepository})
      : _excelRepository = excelRepository;

  final ExcelRepository _excelRepository;

  @override
  Future<Either<Failure, void>> call(ExportingTableParams params) {
    final response = _excelRepository.exportToExcel(params);

    return response;
  }
}
```

lib\feature\home\domain\usecases\find_students_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/find_students_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/firestore_repository.dart';

class FindStudentsUseCase
    extends AsyncUseCase<List<UserEntity>, FindStudentsParams> {
  FindStudentsUseCase({required FirestoreRepository firestoreRepository})
      : _firestoreRepository = firestoreRepository;

  final FirestoreRepository _firestoreRepository;

  @override
  Future<Either<Failure, List<UserEntity>>> call(
    FindStudentsParams params,
  ) async {
    final response = await _firestoreRepository.findStudents(params);

    return response;
  }
}
```

lib\feature\home\domain\usecases\get_locale_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/locale_repository.dart';

class GetLocaleUseCase extends AsyncUseCase<String, NoParams> {
  GetLocaleUseCase({required LocaleRepository localeRepository})
      : _localeRepository = localeRepository;

  final LocaleRepository _localeRepository;

  @override
  Future<Either<Failure, String>> call(NoParams params) async {
    final response = await _localeRepository.getLocale();

    return response;
  }
}
```

lib\feature\home\domain\usecases\get_scores_tables_list_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/scores_table_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_model_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/firestore_repository.dart';

class GetScoresTablesUseCase
    extends AsyncUseCase<List<ScoresTableEntity>, UserModelParams> {
  GetScoresTablesUseCase({required FirestoreRepository firestoreRepository})
      : _firestoreRepository = firestoreRepository;

  final FirestoreRepository _firestoreRepository;

  @override
  Future<Either<Failure, List<ScoresTableEntity>>> call(
    UserModelParams params,
  ) async {
    final response = await _firestoreRepository.getScoresTables(params);

    return response;
  }
}
```

lib\feature\home\domain\usecases\get_scores_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/score_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/firestore_repository.dart';

class GetScoresUseCase extends AsyncUseCase<List<ScoreEntity>, TableParams> {
  GetScoresUseCase({required FirestoreRepository firestoreRepository})
      : _firestoreRepository = firestoreRepository;

  final FirestoreRepository _firestoreRepository;

  @override
  Future<Either<Failure, List<ScoreEntity>>> call(TableParams params) async {
    final response = await _firestoreRepository.getScores(params);

    return response;
  }
}
```

lib\feature\home\domain\usecases\get_user_changes_stream_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/shared/domain/repositories/auth_repository.dart';
import 'package:firebase_auth/firebase_auth.dart';

class GetUserChangesStreamUseCase extends SyncUseCase<Stream<User?>, NoParams> {
  GetUserChangesStreamUseCase({required AuthRepository authRepository})
      : _authRepository = authRepository;

  final AuthRepository _authRepository;

  @override
  Either<Failure, Stream<User?>> call(NoParams params) {
    return _authRepository.getUserChangesStream();
  }
}
```

lib\feature\home\domain\usecases\get_user_data_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/firestore_repository.dart';

class GetUserDataUsecase extends AsyncUseCase<UserEntity, UserParams> {
  GetUserDataUsecase({required FirestoreRepository firestoreRepository})
      : _firestoreRepository = firestoreRepository;

  final FirestoreRepository _firestoreRepository;

  @override
  Future<Either<Failure, UserEntity>> call(UserParams params) async {
    final response = await _firestoreRepository.getUserData(params.uid);

    return response;
  }
}
```

lib\feature\home\domain\usecases\get_users_data_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/users_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/firestore_repository.dart';

class GetUsersDataUseCase extends AsyncUseCase<List<UserEntity>, UsersParams> {
  GetUsersDataUseCase({required FirestoreRepository firestoreRepository})
      : _firestoreRepository = firestoreRepository;

  final FirestoreRepository _firestoreRepository;

  @override
  Future<Either<Failure, List<UserEntity>>> call(
    UsersParams params,
  ) async {
    final response = await _firestoreRepository.getUsersData(params);

    return response;
  }
}
```

lib\feature\home\domain\usecases\register_user_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/params/register_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/register_repository.dart';

class RegisterUserUsecase extends AsyncUseCase<void, RegisterParams> {
  RegisterUserUsecase({required RegisterRepository registerRepository})
      : _registerRepository = registerRepository;

  final RegisterRepository _registerRepository;

  @override
  Future<Either<Failure, void>> call(RegisterParams params) {
    final response = _registerRepository.registerUser(params);

    return response;
  }
}
```

lib\feature\home\domain\usecases\set_locale_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/params/locale_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/locale_repository.dart';

class SetLocaleUseCase extends AsyncUseCase<void, LocaleParams> {
  SetLocaleUseCase({required LocaleRepository localeRepository})
      : _localeRepository = localeRepository;

  final LocaleRepository _localeRepository;

  @override
  Future<Either<Failure, String>> call(LocaleParams params) async {
    final response = await _localeRepository.setLocale(params);

    return response;
  }
}
```

lib\feature\home\domain\usecases\sign_out_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/shared/domain/repositories/auth_repository.dart';

class SignOutUseCase extends AsyncUseCase<void, NoParams> {
  SignOutUseCase({required AuthRepository authRepository})
      : _authRepository = authRepository;

  final AuthRepository _authRepository;

  @override
  Future<Either<Failure, void>> call(NoParams params) async =>
      _authRepository.signOut();
}
```

lib\feature\home\domain\usecases\update_access_time_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/firestore_repository.dart';

class UpdateAccessTimeUseCase extends AsyncUseCase<void, UserParams> {
```

```
  UpdateAccessTimeUseCase({
    required FirestoreRepository firestoreRepository,
  }) : _firestoreRepository = firestoreRepository;

  final FirestoreRepository _firestoreRepository;

  @override
  Future<Either<Failure, void>> call(UserParams params) async {
    final response = await _firestoreRepository.updateAccessTime(params);

    return response;
  }
}
```

lib\feature\home\domain\usecases\update_table_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/params/edit_table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/repositories/firestore_repository.dart';

class UpdateTableUseCase extends AsyncUseCase<void, EditTableParams> {
  UpdateTableUseCase({required FirestoreRepository firestoreRepository})
      : _firestoreRepository = firestoreRepository;

  final FirestoreRepository _firestoreRepository;

  @override
  Future<Either<Failure, void>> call(EditTableParams params) async {
    final response = await _firestoreRepository.updateTable(params);

    return response;
  }
}
```

lib\feature\home\presentation\viewmodels\blocs\user_changes_bloc.dart

```
import 'dart:async';

import 'package:electronic_student_journal/core/app/di/injector.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_params.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/get_user_changes_stream_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/update_access_time_usecase.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';
import 'package:logger/logger.dart';

part 'user_changes_bloc.freezed.dart';
part 'user_changes_event.dart';
part 'user_changes_state.dart';

class UserChangesBloc extends Bloc<UserChangesEvent, UserChangesState> {
  UserChangesBloc({
    required GetUserChangesStreamUseCase getUserChangesStreamUseCase,
    required UpdateAccessTimeUseCase updateAccessTimeUseCase,
  })  : _updateAccessTimeUseCase = updateAccessTimeUseCase,
        super(const _Initial()) {
    getUserChangesStreamUseCase.call(NoParams()).fold(
          (failure) => logger.e(failure.message),
          (stream) => _stream = stream,
        );

    on<_Started>(_started);
    on<_Observe>(_observe);
    on<_PauseObserve>(_pauseObserve);
    on<_ResumeObserve>(_resumeObserve);
```

```
    on<_UserStateChange>(_userStateChange);

    on<UserChangesEvent>(_logEvent);

    add(const _Started());
  }

  final UpdateAccessTimeUseCase _updateAccessTimeUseCase;

  late Stream<User?> _stream;
  StreamSubscription<User?>? _subscription;
  final logger = injector<Logger>();

  void _logEvent(UserChangesEvent event, Emitter<UserChangesState> emit) {
    logger.i(event);
  }

  void _started(_Started event, Emitter<UserChangesState> emit) {}

  void _observe(_Observe event, Emitter<UserChangesState> emit) {
    logger.i(event);
    _subscription ??= _stream.listen((event) {
      add(_UserStateChange(event));
    });
  }

  void _pauseObserve(_PauseObserve event, Emitter<UserChangesState> emit) {
    if (_subscription != null) {
      _subscription!.pause();
    }
  }

  void _resumeObserve(_ResumeObserve event, Emitter<UserChangesState> emit) {
    if (_subscription != null) {
      _subscription!.resume();
    }
  }

  void _userStateChange(
    _UserStateChange event,
    Emitter<UserChangesState> emit,
  ) {
    if (event.user != null) {
      _updateAccessTimeUseCase.call(UserParams(uid: event.user!.uid));
      emit(_UserSignsIn(event.user!));
    } else {
      emit(const _UserSignsOut());
    }
  }

  @override
  Future<void> close() {
    if (_subscription != null) {
      _subscription!.cancel();
    }
    return super.close();
  }
}
```

lib\feature\home\presentation\viewmodels\blocs\user_changes_event.dart

```
part of 'user_changes_bloc.dart';

@freezed
class UserChangesEvent with _$UserChangesEvent {
  const factory UserChangesEvent.started() = _Started;
  const factory UserChangesEvent.observe() = _Observe;
  const factory UserChangesEvent.pauseObserve() = _PauseObserve;
  const factory UserChangesEvent.resumeObserve() = _ResumeObserve;
```

```
  const factory UserChangesEvent.userStateChange(User? user) = _UserStateChange;
}
```

lib\feature\home\presentation\viewmodels\blocs\user_changes_state.dart

```
part of 'user_changes_bloc.dart';

@freezed
class UserChangesState with _$UserChangesState {
  const factory UserChangesState.initial() = _Initial;
  const factory UserChangesState.userSignsIn(User user) = _UserSignsIn;
  const factory UserChangesState.userSingsOut() = _UserSignsOut;
}
```

lib\feature\home\presentation\viewmodels\cubits\create_table_cubit.dart

```
import 'package:electronic_student_journal/feature/home/domain/params/edit_table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/create_table_usecase.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'create_table_state.dart';
part 'create_table_cubit.freezed.dart';

class CreateTableCubit extends Cubit<CreateTableState> {
  CreateTableCubit({required CreateTableUseCase createTableUseCase})
      : _createTableUseCase = createTableUseCase,
        super(const _Initial());

  final CreateTableUseCase _createTableUseCase;

  Future<void> createTable(EditTableParams params) async {
    emit(const _Loading());
    final response = await _createTableUseCase.call(params);

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (_) => emit(const _Success()),
    );
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\create_table_state.dart

```
part of 'create_table_cubit.dart';

@freezed
class CreateTableState with _$CreateTableState {
  const factory CreateTableState.initial() = _Initial;
  const factory CreateTableState.loading() = _Loading;
  const factory CreateTableState.success() = _Success;
  const factory CreateTableState.failure(String message) = _Failure;
}
```

lib\feature\home\presentation\viewmodels\cubits\delete_table_cubit.dart

```
import 'package:electronic_student_journal/feature/home/domain/params/table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/delete_table_usecase.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'delete_table_state.dart';
part 'delete_table_cubit.freezed.dart';

class DeleteTableCubit extends Cubit<DeleteTableState> {
  DeleteTableCubit({required DeleteTableUseCase deleteTableUseCase})
      : _deleteTableUseCase = deleteTableUseCase,
        super(const _Initial());
```

```
  final DeleteTableUseCase _deleteTableUseCase;

  Future<void> deleteTable(TableParams params) async {
    emit(const _Loading());
    final response = await _deleteTableUseCase.call(params);

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (_) => emit(const _Success()),
    );
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\delete_table_state.dart

```
part of 'delete_table_cubit.dart';

@freezed
class DeleteTableState with _$DeleteTableState {
  const factory DeleteTableState.initial() = _Initial;
  const factory DeleteTableState.loading() = _Loading;
  const factory DeleteTableState.success() = _Success;
  const factory DeleteTableState.failure(String message) = _Failure;
}
```

lib\feature\home\presentation\viewmodels\cubits\export_to_excel_cubit.dart

```
import 'package:electronic_student_journal/feature/home/domain/params/exporting_table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/export_table_to_excel_usecase.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'export_to_excel_state.dart';
part 'export_to_excel_cubit.freezed.dart';

class ExportToExcelCubit extends Cubit<ExportToExcelState> {
  ExportToExcelCubit({
    required ExportTableToExcelUseCase exportTableToExcelUseCase,
  }) : _exportTableToExcelUseCase = exportTableToExcelUseCase,
       super(const _Initial());

  final ExportTableToExcelUseCase _exportTableToExcelUseCase;

  Future<void> exportToExcel(ExportingTableParams params) async {
    emit(const _Loading());
    final response = await _exportTableToExcelUseCase.call(params);

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (_) => emit(const _Success()),
    );
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\export_to_excel_state.dart

```
part of 'export_to_excel_cubit.dart';

@freezed
class ExportToExcelState with _$ExportToExcelState {
  const factory ExportToExcelState.initial() = _Initial;
  const factory ExportToExcelState.loading() = _Loading;
  const factory ExportToExcelState.success() = _Success;
  const factory ExportToExcelState.failure(String message) = _Failure;
}
```

lib\feature\home\presentation\viewmodels\cubits\find_students_cubit.dart

```
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
```

```
import 'package:electronic_student_journal/feature/home/domain/params/find_students_params.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/find_students_usecase.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'find_students_state.dart';
part 'find_students_cubit.freezed.dart';

class FindStudentsCubit extends Cubit<FindStudentsState> {
  FindStudentsCubit({required FindStudentsUseCase findStudentsUseCase})
      : _findStudentsUseCase = findStudentsUseCase,
        super(const _Initial());

  final FindStudentsUseCase _findStudentsUseCase;

  Future<void> findStudents(FindStudentsParams params) async {
    emit(const _Loading());
    final response = await _findStudentsUseCase.call(params);

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (students) => emit(_Success(students)),
    );
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\find_students_state.dart

```
part of 'find_students_cubit.dart';

@freezed
class FindStudentsState with _$FindStudentsState {
  const factory FindStudentsState.initial() = _Initial;
  const factory FindStudentsState.loading() = _Loading;
  const factory FindStudentsState.success(List<UserEntity> students) = _Success;
  const factory FindStudentsState.failure(String message) = _Failure;
}
```

lib\feature\home\presentation\viewmodels\cubits\get_scores_cubit.dart

```
import 'package:electronic_student_journal/feature/home/domain/entities/score_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/get_scores_usecase.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'get_scores_state.dart';
part 'get_scores_cubit.freezed.dart';

class GetScoresCubit extends Cubit<GetScoresState> {
  GetScoresCubit({required GetScoresUseCase getScoresUseCase})
      : _getScoresUseCase = getScoresUseCase,
        super(const _Initial());

  final GetScoresUseCase _getScoresUseCase;

  Future<void> getScores(String tableUid) async {
    emit(const _Loading());
    final response = await _getScoresUseCase.call(TableParams(uid: tableUid));

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (scores) => emit(_Success(scores)),
    );
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\get_scores_state.dart

```
part of 'get_scores_cubit.dart';

@freezed
class GetScoresState with _$GetScoresState {
  const factory GetScoresState.initial() = _Initial;
  const factory GetScoresState.loading() = _Loading;
  const factory GetScoresState.success(List<ScoreEntity> scores) = _Success;
  const factory GetScoresState.failure(String message) = _Failure;
}
```

lib\feature\home\presentation\viewmodels\cubits\get_scores_tables_cubit.dart

```
import 'package:electronic_student_journal/feature/home/domain/entities/scores_table_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_model_params.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/get_scores_tables_list_usecase.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'get_scores_tables_state.dart';
part 'get_scores_tables_cubit.freezed.dart';

class GetScoresTablesCubit extends Cubit<GetScoresTablesState> {
  GetScoresTablesCubit({required GetScoresTablesUseCase getScoresTablesUseCase})
      : _getScoresTablesUseCase = getScoresTablesUseCase,
        super(const _Initial());

  final GetScoresTablesUseCase _getScoresTablesUseCase;

  Future<void> getScoresTables(UserEntity user) async {
    emit(const _Loading());
    final response = await _getScoresTablesUseCase
        .call(UserModelParams(user: user.toModel()));

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (tables) => emit(_Success(tables)),
    );
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\get_scores_tables_state.dart

```
part of 'get_scores_tables_cubit.dart';

@freezed
class GetScoresTablesState with _$GetScoresTablesState {
  const factory GetScoresTablesState.initial() = _Initial;
  const factory GetScoresTablesState.loading() = _Loading;
  const factory GetScoresTablesState.success(List<ScoresTableEntity> tables) =
      _Success;
  const factory GetScoresTablesState.failure(String message) = _Failure;
}
```

lib\feature\home\presentation\viewmodels\cubits\get_user_data_cubit.dart

```
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/user_params.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/get_user_data_usecase.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'get_user_data_state.dart';
part 'get_user_data_cubit.freezed.dart';

class GetUserDataCubit extends Cubit<GetUserDataState> {
  GetUserDataCubit({required GetUserDataUsecase getUserDataUsecase})
      : _getUserDataUsecase = getUserDataUsecase,
        super(const _Initial());
```

```
  final GetUserDataUsecase _getUserDataUsecase;

  Future<void> getUserData(String uid) async {
    emit(const _Loading());
    final response = await _getUserDataUsecase.call(UserParams(uid: uid));

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (userEntity) => emit(_Success(userEntity)),
    );
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\get_user_data_state.dart

```
part of 'get_user_data_cubit.dart';

@freezed
class GetUserDataState with _$GetUserDataState {
  const factory GetUserDataState.initial() = _Initial;
  const factory GetUserDataState.loading() = _Loading;
  const factory GetUserDataState.success(UserEntity userEntity) = _Success;
  const factory GetUserDataState.failure(String message) = _Failure;
}
```

lib\feature\home\presentation\viewmodels\cubits\get_users_data_cubit_dart_cubit.dart

```
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/users_params.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/get_users_data_usecase.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'get_users_data_cubit_dart_cubit.freezed.dart';
part 'get_users_data_cubit_dart_state.dart';

class GetUsersDataCubit extends Cubit<GetUsersDataState> {
  GetUsersDataCubit({required GetUsersDataUseCase getUsersDataUseCase})
      : _getUsersDataUseCase = getUsersDataUseCase,
        super(const _Initial());

  final GetUsersDataUseCase _getUsersDataUseCase;

  Future<void> getUsersData(List<String> uids) async {
    emit(const _Loading());
    final response = await _getUsersDataUseCase.call(UsersParams(uids: uids));

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (userEntities) => emit(_Success(userEntities)),
    );
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\get_users_data_cubit_dart_state.dart

```
part of 'get_users_data_cubit_dart_cubit.dart';

@freezed
class GetUsersDataState with _$GetUsersDataState {
  const factory GetUsersDataState.initial() = _Initial;
  const factory GetUsersDataState.loading() = _Loading;
  const factory GetUsersDataState.success(List<UserEntity> userEntities) =
      _Success;
  const factory GetUsersDataState.failure(String message) = _Failure;
}
```

lib\feature\home\presentation\viewmodels\cubits\locale_cubit.dart

```
import 'package:electronic_student_journal/core/app/di/injector.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/params/locale_params.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/get_locale_usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/set_locale_usecase.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';
import 'package:logger/logger.dart';

part 'locale_state.dart';
part 'locale_cubit.freezed.dart';

class LocaleCubit extends Cubit<LocaleState> {
  LocaleCubit({
    required GetLocaleUseCase getLocaleUseCase,
    required SetLocaleUseCase setLocaleUseCase,
  })  : _getLocaleUseCase = getLocaleUseCase,
        _setLocaleUseCase = setLocaleUseCase,
        super(const _Initial());

  final GetLocaleUseCase _getLocaleUseCase;
  final SetLocaleUseCase _setLocaleUseCase;
  final _logger = injector<Logger>();

  Future<void> loadLocale() async {
    emit(const _Loading());
    final response = await _getLocaleUseCase.call(NoParams());
    _logger.w(response);

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (locale) => emit(_Loaded(locale)),
    );
  }

  Future<void> saveLocale(LocaleParams params) async {
    emit(const _Loading());
    final response = await _setLocaleUseCase.call(params);
    _logger.w(response);

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (locale) => emit(_Saved(locale)),
    );
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\locale_state.dart

```
part of 'locale_cubit.dart';

@freezed
class LocaleState with _$LocaleState {
  const factory LocaleState.initial() = _Initial;
  const factory LocaleState.loading() = _Loading;
  const factory LocaleState.loaded(String locale) = _Loaded;
  const factory LocaleState.saved(String locale) = _Saved;
  const factory LocaleState.failure(String message) = _Failure;
}
```

lib\feature\home\presentation\viewmodels\cubits\register_user_cubit.dart

```
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/register_params.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/register_user_usecase.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';
```

```
part 'register_user_state.dart';
part 'register_user_cubit.freezed.dart';

class RegisterUserCubit extends Cubit<RegisterUserState> {
  RegisterUserCubit({required RegisterUserUsecase registerUserUsecase})
      : _registerUserUsecase = registerUserUsecase,
        super(const _Initial());

  final RegisterUserUsecase _registerUserUsecase;

  Future<void> registerUser(UserEntity user, String password) async {
    emit(const _Loading());
    final response = await _registerUserUsecase
        .call(RegisterParams(user: user.toModel(), password: password));

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (success) => emit(const _Success()),
    );
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\register_user_state.dart

```
part of 'register_user_cubit.dart';

@freezed
class RegisterUserState with _$RegisterUserState {
  const factory RegisterUserState.initial() = _Initial;
  const factory RegisterUserState.loading() = _Loading;
  const factory RegisterUserState.success() = _Success;
  const factory RegisterUserState.failure(String message) = _Failure;
}
```

lib\feature\home\presentation\viewmodels\cubits\scores_cubit.dart

```
import 'package:electronic_student_journal/feature/home/domain/entities/score_entity.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'scores_state.dart';
part 'scores_cubit.freezed.dart';

class ScoresCubit extends Cubit<ScoresState> {
  ScoresCubit({List<ScoreEntity>? scores}) : super(_Initial(scores ?? []));

  void addScore(ScoreEntity score) {
    final scores = List.of(state.scores)..add(score);

    emit(_ScoreAdded(scores));
  }

  void removeScore(ScoreEntity score) {
    final scores = List.of(state.scores)..remove(score);

    emit(_ScoreDeleted(scores));
  }

  void changeScore({
    required ScoreEntity initialScore,
    required ScoreEntity score,
  }) {
    final scores = List.of(state.scores);
    scores[scores.indexOf(initialScore)] = score;

    emit(_ScoreChanged(scores));
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\scores_names_cubit.dart

```
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'scores_names_state.dart';
part 'scores_names_cubit.freezed.dart';

class ScoresNamesCubit extends Cubit<ScoresNamesState> {
  ScoresNamesCubit({List<DateTime>? scoresNames})
      : super(ScoresNamesState.initial(scoresNames ?? []));

  void addScoreName(DateTime scoreName) {
    final scoresNames = List.of(state.scoresNames)..add(scoreName);
    emit(_ScoreNameAdded(scoresNames));
  }

  void removeScoreName(DateTime scoreName) {
    final scoresNames = List.of(state.scoresNames)..remove(scoreName);

    emit(_ScoreNameRemoved(scoresNames));
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\scores_names_state.dart

```
part of 'scores_names_cubit.dart';

@freezed
class ScoresNamesState with _$ScoresNamesState {
  const factory ScoresNamesState.initial(List<DateTime> scoresNames) = _Initial;
  const factory ScoresNamesState.scoreNameAdded(List<DateTime> scoresNames) =
      _ScoreNameAdded;
  const factory ScoresNamesState.scoreNameRemoved(List<DateTime> scoresNames) =
      _ScoreNameRemoved;
}
```

lib\feature\home\presentation\viewmodels\cubits\scores_state.dart

```
part of 'scores_cubit.dart';

@freezed
class ScoresState with _$ScoresState {
  const factory ScoresState.initial(List<ScoreEntity> scores) = _Initial;
  const factory ScoresState.scoreAdded(List<ScoreEntity> scores) = _ScoreAdded;
  const factory ScoresState.scoreDeleted(List<ScoreEntity> scores) =
      _ScoreDeleted;
  const factory ScoresState.scoreChanged(List<ScoreEntity> scores) =
      _ScoreChanged;
}
```

lib\feature\home\presentation\viewmodels\cubits\sign_out_cubit.dart

```
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/sign_out_usecase.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'sign_out_state.dart';
part 'sign_out_cubit.freezed.dart';

class SignOutCubit extends Cubit<SignOutState> {
  SignOutCubit({required SignOutUseCase signOutUseCase})
      : _signOutUseCase = signOutUseCase,
        super(const _Initial());

  final SignOutUseCase _signOutUseCase;
```

```
  Future<void> signOut() async {
    final response = await _signOutUseCase.call(NoParams());

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (success) => emit(const _Success()),
    );
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\sign_out_state.dart

```
part of 'sign_out_cubit.dart';

@freezed
class SignOutState with _$SignOutState {
  const factory SignOutState.initial() = _Initial;
  const factory SignOutState.loading() = _Loading;
  const factory SignOutState.success() = _Success;
  const factory SignOutState.failure(String message) = _Failure;
}
```

lib\feature\home\presentation\viewmodels\cubits\students_cubit.dart

```
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'students_state.dart';
part 'students_cubit.freezed.dart';

class StudentsCubit extends Cubit<StudentsState> {
  StudentsCubit({List<UserEntity>? students}) : super(_Initial(students ?? []));

  void addStudent(UserEntity student) {
    final students = List.of(state.students)..add(student);

    emit(_StudentAdded(students));
  }

  void removeStudent(UserEntity student) {
    final students = List.of(state.students)..remove(student);

    emit(_StudentRemoved(students));
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\students_state.dart

```
part of 'students_cubit.dart';

@freezed
class StudentsState with _$StudentsState {
  const factory StudentsState.initial(List<UserEntity> students) = _Initial;
  const factory StudentsState.studentAdded(List<UserEntity> students) =
      _StudentAdded;
  const factory StudentsState.studentRemoved(List<UserEntity> students) =
      _StudentRemoved;
}
```

lib\feature\home\presentation\viewmodels\cubits\update_table_cubit.dart

```
import 'package:electronic_student_journal/feature/home/domain/params/edit_table_params.dart';
import 'package:electronic_student_journal/feature/home/domain/usecases/update_table_usecase.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'update_table_state.dart';
part 'update_table_cubit.freezed.dart';
```

```
class UpdateTableCubit extends Cubit<UpdateTableState> {
  UpdateTableCubit({required UpdateTableUseCase updateTableUseCase})
      : _updateTableUseCase = updateTableUseCase,
        super(const _Initial());

  final UpdateTableUseCase _updateTableUseCase;

  Future<void> updateTable(EditTableParams params) async {
    emit(const _Loading());
    final response = await _updateTableUseCase.call(params);

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (_) => emit(const _Success()),
    );
  }
}
```

lib\feature\home\presentation\viewmodels\cubits\update_table_state.dart

```
part of 'update_table_cubit.dart';

@freezed
class UpdateTableState with _$UpdateTableState {
  const factory UpdateTableState.initial() = _Initial;
  const factory UpdateTableState.loading() = _Loading;
  const factory UpdateTableState.success() = _Success;
  const factory UpdateTableState.failure(String message) = _Failure;
}
```

lib\feature\home\presentation\viewmodels\inherited\group_controller.dart

```
import 'package:flutter/material.dart';

class GroupController extends InheritedWidget {
  GroupController({required super.child, super.key});

  final TextEditingController controller = TextEditingController();

  static GroupController? maybeOf(BuildContext context) {
    return context.dependOnInheritedWidgetOfExactType<GroupController>();
  }

  static GroupController of(BuildContext context) {
    final result = maybeOf(context);
    assert(result != null, 'No GroupController found in context');
    return result!;
  }

  @override
  bool updateShouldNotify(GroupController oldWidget) =>
      controller != oldWidget.controller;
}
```

lib\feature\home\presentation\viewmodels\inherited\password_controller.dart

```
import 'package:flutter/material.dart';

class PasswordController extends InheritedWidget {
  PasswordController({required super.child, super.key});

  final TextEditingController controller = TextEditingController();

  static PasswordController? maybeOf(BuildContext context) {
    return context.dependOnInheritedWidgetOfExactType<PasswordController>();
  }

  static PasswordController of(BuildContext context) {
```

```
    final result = maybeOf(context);
    assert(result != null, 'No PasswordController found in context');
    return result!;
  }

  @override
  bool updateShouldNotify(PasswordController oldWidget) =>
      controller != oldWidget.controller;
}
```

lib\feature\home\presentation\viewmodels\providers\group_provider.dart

```
import 'package:flutter/material.dart';

class GroupProvider extends ChangeNotifier {
  GroupProvider({String? group}) : _group = group;

  String? _group;

  String? get group => _group;

  void changeGroup(String? group) {
    _group = group;
    notifyListeners();
  }
}
```

lib\feature\home\presentation\viewmodels\providers\name_provider.dart

```
import 'package:flutter/material.dart';

class NameProvider extends ChangeNotifier {
  NameProvider({String? name}) : _name = name;

  String? _name;

  String? get name => _name;

  void changeName(String name) {
    _name = name;
    notifyListeners();
  }
}
```

lib\feature\home\presentation\viewmodels\providers\password_confirmer_hinter.dart

```
import 'package:flutter/foundation.dart';

class PasswordConfirmerHinter extends ChangeNotifier {
  bool _isPasswordHinted = true;

  bool get isPasswordHinted => _isPasswordHinted;

  void toggleVisibility() {
    _isPasswordHinted = !_isPasswordHinted;
    notifyListeners();
  }
}
```

lib\feature\home\presentation\viewmodels\providers\password_confirmer_provider.dart

```
import 'package:flutter/foundation.dart';

class PasswordConfirmerProvider extends ChangeNotifier {
  PasswordConfirmerProvider({String? confirmedPassword})
      : _confirmedPassword = confirmedPassword;

  String? _confirmedPassword;
```

```
  String? get confirmedPassword => _confirmedPassword;

  void changeConfirmedPassword(String password) {
    _confirmedPassword = password;
    notifyListeners();
  }
}
```

lib\feature\home\presentation\viewmodels\providers\patronymic_provider.dart

```
import 'package:flutter/material.dart';

class PatronymicProvider extends ChangeNotifier {
  PatronymicProvider({String? patronymic}) : _patronymic = patronymic;

  String? _patronymic;

  String? get patronymic => _patronymic;

  void changePatronymic(String patronymic) {
    _patronymic = patronymic;
    notifyListeners();
  }
}
```

lib\feature\home\presentation\viewmodels\providers\role_provider.dart

```
import 'package:flutter/material.dart';

class RoleProvider extends ChangeNotifier {
  RoleProvider({String? role}) : _role = role;

  String? _role;

  String? get role => _role;

  void changeRole(String role) {
    _role = role;
    notifyListeners();
  }
}
```

lib\feature\home\presentation\viewmodels\providers\score_name_provider.dart

```
import 'package:flutter/material.dart';

class ScoreNameProvider extends ChangeNotifier {
  ScoreNameProvider({String? scoreName}) : _scoreName = scoreName;

  String? _scoreName;

  String? get scoreName => _scoreName;

  void changeScoreName(String scoreName) {
    _scoreName = scoreName;
    notifyListeners();
  }
}
```

lib\feature\home\presentation\viewmodels\providers\scores_table_name_provider.dart

```
import 'package:flutter/material.dart';

class ScoresTableNameProvider extends ChangeNotifier {
  ScoresTableNameProvider({String? tableName}) : _tableName = tableName;

  String? _tableName;

  String? get tableName => _tableName;
```

```
  void changeScoresTableName(String tableName) {
    _tableName = tableName;
    notifyListeners();
  }
}
```

lib\feature\home\presentation\viewmodels\providers\show_confirm_score_button.dart

```
import 'package:flutter/material.dart';

class ShowConfirmScoreButton extends ChangeNotifier {
  bool _isButtonShown = false;

  bool get isButtonShown => _isButtonShown;

  void showButton() {
    if (!_isButtonShown) {
      _isButtonShown = true;
      notifyListeners();
    }
  }

  void hideButton() {
    _isButtonShown = false;
    notifyListeners();
  }
}
```

lib\feature\home\presentation\viewmodels\providers\show_student_search_provider.dart

```
import 'package:flutter/foundation.dart';

class ShowStudentSearchProvider extends ChangeNotifier {
  ShowStudentSearchProvider({bool showStudentSearch = false})
      : _showStudentSearch = showStudentSearch;

  bool _showStudentSearch;

  bool get showStudentSearch => _showStudentSearch;

  void toggleVisibility() {
    _showStudentSearch = !_showStudentSearch;
    notifyListeners();
  }
}
```

lib\feature\home\presentation\viewmodels\providers\surname_provider.dart

```
import 'package:flutter/material.dart';

class SurnameProvider extends ChangeNotifier {
  SurnameProvider({String? surname}) : _surname = surname;

  String? _surname;

  String? get surname => _surname;

  void changeSurname(String surname) {
    _surname = surname;
    notifyListeners();
  }
}
```

lib\feature\home\presentation\viewmodels\providers\university_provider.dart

```
import 'package:flutter/material.dart';

class UniversityProvider extends ChangeNotifier {
```

```
  UniversityProvider({String? university}) : _university = university;

  String? _university;

  String? get university => _university;

  void changeUniversity(String university) {
    _university = university;
    notifyListeners();
  }
}
```

lib\feature\home\presentation\views\edit_scores_table_view.dart

```
import 'package:electronic_student_journal/feature/home/domain/entities/scores_table_entity.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/edit_scores_table_form.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';

class EditScoresTableView extends StatelessWidget {
  const EditScoresTableView({
    this.userRole,
    this.table,
    super.key,
  });

  final String? userRole;
  final ScoresTableEntity? table;

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return Scaffold(
      appBar: AppBar(
        title: table != null ? Text(table!.name) : Text(l10n.tableTitle),
      ),
      body: SingleChildScrollView(
        child: EditScoresTableForm(table: table),
      ),
    );
  }
}
```

lib\feature\home\presentation\views\home_view.dart

```
import 'package:electronic_student_journal/core/app/router/app_router.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/blocs/user_changes_bloc.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_user_data_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/sign_out_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/scores_button.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:go_router/go_router.dart';

class HomeView extends StatelessWidget {
  const HomeView({super.key});

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return BlocListener<UserChangesBloc, UserChangesState>(
      listener: (context, state) {
        state.whenOrNull(
          userSignsIn: (user) =>
              context.read<GetUserDataCubit>().getUserData(user.uid),
          userSingsOut: () => appRouter.go(Routes.signIn.path),
        );
```

```
    },
    child: Scaffold(
      appBar: AppBar(
        title: Text(l10n.appNameTitle),
      ),
      body: BlocConsumer<GetUserDataCubit, GetUserDataState>(
        listener: (context, state) {
          state.whenOrNull(
            failure: (_) => context.read<SignOutCubit>().signOut(),
          );
        },
        builder: (context, state) {
          return state.maybeWhen(
            success: (userEntity) {
              final buttons = <Widget>[];

              switch (userEntity.role) {
                case UserRole.admin:
                  buttons.add(
                    ElevatedButton(
                      onPressed: () => context.goNamed(
                        Routes.signUp.name,
                        queryParameters: {'userRole': userEntity.role},
                        extra: context.read<UserChangesBloc>(),
                      ),
                      child: Text(
                        l10n.signUpUser,
                        textAlign: TextAlign.center,
                      ),
                    ),
                  );
                  break;
                case UserRole.student:
                  buttons.add(const ScoresButton());
                  break;
                case UserRole.teacher:
                  buttons.add(const ScoresButton());
                  break;
              }

              buttons.add(
                ElevatedButton(
                  onPressed: () => context.go(
                    Routes.settings.path,
                    extra: context.read<UserChangesBloc>(),
                  ),
                  child: Text(l10n.settings),
                ),
              );

              return GridView.count(
                crossAxisCount: 2,
                children: buttons,
              );
            },
            orElse: () => const Center(child: CircularProgressIndicator()),
          );
        },
      ),
    ),
  );
 }
}
```

lib\feature\home\presentation\views\scores_table_view.dart

```
import 'package:electronic_student_journal/core/app/router/app_router.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/scores_table_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
```

```
import 'package:electronic_student_journal/feature/home/domain/params/exporting_table_params.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/blocs/user_changes_bloc.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/export_to_excel_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_scores_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_users_data_cubit_dart_cubit.dart';
import 'package:electronic_student_journal/utils/ext/score_date.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:go_router/go_router.dart';
import 'package:overlay_support/overlay_support.dart';
import 'package:permission_handler/permission_handler.dart';

class ScoresTableView extends StatelessWidget {
  const ScoresTableView({
    required this.userRole,
    required this.table,
    super.key,
  });

  final String userRole;
  final ScoresTableEntity table;

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return Scaffold(
      appBar: AppBar(
        title: Text(table.name),
        actions: [
          if (userRole == UserRole.teacher)
            IconButton(
              onPressed: () => context.goNamed(
                Routes.editScoresTable.name,
                extra: (context.read<UserChangesBloc>(), table),
                queryParameters: {'userRole': userRole},
              ),
              icon: const Icon(Icons.edit),
            ),
        ],
      ),
      body: BlocBuilder<GetScoresCubit, GetScoresState>(
        builder: (context, state) {
          return state.maybeWhen(
            success: (scores) {
              final tableName = table.name;
              final tableNameText = Text(tableName);

              final scoresTitles = scores
                  .map(
                    (score) => score.date.scoreDateFormat(),
                  )
                  .toSet()
                  .toList();

              final scoresTitlesTexts = scoresTitles.map((scoreTitle) {
                return DecoratedBox(
                  decoration: BoxDecoration(border: Border.all()),
                  child: Text(scoreTitle),
                );
              });

              final studentsUids =
                  scores.map((score) => score.studentUid).toSet().toList();

              context.read<GetUsersDataCubit>().getUsersData(studentsUids);

              return BlocBuilder<GetUsersDataCubit, GetUsersDataState>(
```

```
builder: (context, state) => state.maybeWhen(
  success: (userEntities) {
    final scoresMap = <String, List<int>>{};
    final studentsFullNames = <String>[];

    for (var i = 0; i < userEntities.length; i++) {
      studentsFullNames.add('${userEntities[i].surname} '
          '${userEntities[i].name![0]}. '
          '${userEntities[i].patronymic![0]}.');

      final scoresValues = scores
          .where(
            (score) => score.studentUid == userEntities[i].uid,
          )
          .map((score) => score.score)
          .toList();

      scoresMap.addAll({studentsFullNames[i]: scoresValues});
    }

    final studentsFullNamesTexts =
        studentsFullNames.map(Text.new).toList();

    final scoresMapWidgets = scoresMap.map(
      (studentFullName, scoresValues) => MapEntry(
        studentFullName,
        scoresValues
            .map(
              (score) => DecoratedBox(
                decoration: BoxDecoration(border: Border.all()),
                child:
                    // TODO(nograve): Solve checking for -1 in
                    // scores only once
                    Text(score == -1 ? '' : score.toString()),
              ),
            )
            .toList(),
      ),
    );

    // TODO(nograve): Move this to export table data source
    final resultScores = <dynamic>[];
    for (var i = 0; i < studentsFullNames.length; i++) {
      final studentScores = <dynamic>[];
      for (var j = 0;
          j < scoresMap.values.toList()[i].length;
          j++) {
        if (scoresMap.values.toList()[i][j] == -1) {
          studentScores.add('');
        } else {
          studentScores.add(scoresMap.values.toList()[i][j]);
        }
      }

      resultScores.addAll([
        studentsFullNames[i],
        ...studentScores,
      ]);
    }

    final resultScoresWidgets = <Widget>[];
    for (var i = 0; i < studentsFullNamesTexts.length; i++) {
      resultScoresWidgets.addAll([
        studentsFullNamesTexts[i],
        ...scoresMapWidgets.values.toList()[i],
      ]);
    }

    final resultTable = <Widget>[
```

```
                tableNameText,
                ...scoresTitlesTexts,
                ...resultScoresWidgets,
              ];

              final crossAxisCount = scoresTitlesTexts.length + 1;
              final mainAxisCount = studentsFullNamesTexts.length + 1;
              final content = <dynamic>[
                tableName,
                ...scoresTitles,
                ...resultScores,
              ];

              return Column(
                children: [
                  Expanded(
                    child: GridView.count(
                      crossAxisCount: crossAxisCount,
                      children: resultTable,
                    ),
                  ),
                  BlocListener<ExportToExcelCubit, ExportToExcelState>(
                    listener: (context, state) => state.whenOrNull(
                      success: () => toast('Saved!'),
                      failure: toast,
                    ),
                    child: SizedBox(
                      height: 50.h,
                      child: ElevatedButton(
                        onPressed: () async {
                          final exportToExcelCubit =
                              context.read<ExportToExcelCubit>();

                          if (await Permission.manageExternalStorage
                              .request()
                              .isGranted) {
                            await exportToExcelCubit.exportToExcel(
                              ExportingTableParams(
                                cols: crossAxisCount,
                                rows: mainAxisCount,
                                content: content,
                                tableName: table.name,
                              ),
                            );
                          }
                        },
                        child: Text(l10n.exportToExcel),
                      ),
                    ),
                  ),
                ],
              );
            },
            orElse: () => const Center(
              child: CircularProgressIndicator(),
            ),
          ),
        );
      },
      orElse: () => const Center(
        child: CircularProgressIndicator(),
      ),
    );
  }
}
```

lib\feature\home\presentation\views\scores_view.dart

```
import 'package:electronic_student_journal/core/app/router/app_router.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/scores_table_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/blocs/user_changes_bloc.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_scores_tables_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_user_data_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/sign_out_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/add_scores_table_button.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

class ScoresView extends StatelessWidget {
  const ScoresView({super.key});

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return BlocListener<UserChangesBloc, UserChangesState>(
      listener: (_, state) {
        state.whenOrNull(
          userSignsIn: (user) =>
              context.read<GetUserDataCubit>().getUserData(user.uid),
          userSingsOut: () => appRouter.go(Routes.signIn.path),
        );
      },
      child: Scaffold(
        appBar: AppBar(
          title: Text(l10n.scores),
        ),
        body: BlocBuilder<UserChangesBloc, UserChangesState>(
          builder: (context, userChangesState) {
            userChangesState.whenOrNull(
              userSignsIn: (user) =>
                  context.read<GetUserDataCubit>().getUserData(user.uid),
            );
            return BlocConsumer<GetUserDataCubit, GetUserDataState>(
              listener: (context, state) {
                state.whenOrNull(
                  failure: (_) => context.read<SignOutCubit>().signOut(),
                );
              },
              builder: (context, state) {
                return state.maybeWhen(
                  success: (userEntity) {
                    context
                        .read<GetScoresTablesCubit>()
                        .getScoresTables(userEntity);

                    return BlocBuilder<GetScoresTablesCubit,
                        GetScoresTablesState>(
                      builder: (context, state) {
                        final buttons = <Widget>[];

                        final scoreTables = state.maybeWhen(
                          success: (tables) => tables,
                          orElse: () => <ScoresTableEntity>[],
                        );

                        final scoreTablesButtons = scoreTables.map(
                          (table) => ElevatedButton(
                            onPressed: () => appRouter.goNamed(
                              Routes.scoresTable.name,
                              extra: (context.read<UserChangesBloc>(), table),
                              queryParameters: {'userRole': userEntity.role},
                            ),
```

```
                    child: Text(
                      table.name,
                      textAlign: TextAlign.center,
                      style: TextStyle(
                        fontSize: 18.sp,
                      ),
                      overflow: TextOverflow.ellipsis,
                      maxLines: 3,
                    ),
                  ),
                );

                if (userEntity.role == UserRole.teacher) {
                  buttons.add(const AddScoresTableButton());
                }

                buttons.addAll(scoreTablesButtons);

                return GridView.count(
                  crossAxisCount: 3,
                  children: buttons,
                );
              },
            );
          },
          orElse: () =>
              const Center(child: CircularProgressIndicator()),
        );
      },
    );
  },
),
),
);
}
}
```

lib\feature\home\presentation\views\settings_view.dart

```
import 'package:electronic_student_journal/core/app/router/app_router.dart';
import 'package:electronic_student_journal/feature/home/domain/params/locale_params.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/blocs/user_changes_bloc.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/locale_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/sign_out_cubit.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:overlay_support/overlay_support.dart';

class SettingsView extends StatelessWidget {
  const SettingsView({super.key});

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return BlocListener<UserChangesBloc, UserChangesState>(
      listener: (_, state) {
        state.whenOrNull(
          userSingsOut: () => appRouter.go(Routes.signIn.path),
        );
      },
      child: Scaffold(
        appBar: AppBar(
          title: Text(l10n.settings),
        ),
        body: Column(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [
```

```
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Text(
      l10n.language,
      style: const TextStyle(
        fontSize: 20,
        fontWeight: FontWeight.w500,
      ),
    ),
    Padding(
      padding: EdgeInsets.only(left: 24.w),
      child: BlocBuilder<LocaleCubit, LocaleState>(
        builder: (context, state) {
          final locale = state.maybeWhen(
            loaded: Locale.new,
            saved: Locale.new,
            orElse: () => Localizations.localeOf(context),
          );

          final String? localeValue;
          if (locale.languageCode == 'uk') {
            localeValue = l10n.uaLang;
          } else if (locale.languageCode == 'en') {
            localeValue = l10n.enLang;
          } else {
            localeValue = null;
          }

          return DropdownButton<String>(
            value: localeValue,
            items: <String>[l10n.uaLang, l10n.enLang]
                .map(
                  (value) => DropdownMenuItem(
                    value: value,
                    child: Text(value),
                  ),
                )
                .toList(),
            onChanged: (newLocaleValue) {
              final String? localeLang;
              if (newLocaleValue == l10n.uaLang) {
                localeLang = 'uk';
              } else if (newLocaleValue == l10n.enLang) {
                localeLang = 'en';
              } else {
                localeLang = null;
              }

              if (localeLang != null) {
                context
                    .read<LocaleCubit>()
                    .saveLocale(LocaleParams(locale: localeLang));
              }
            },
          );
        },
      ),
    )
  ],
),
SizedBox(
  width: 150.w,
  height: 50.h,
  child: ElevatedButton(
    onPressed: () {
      toast('Signed out');
      context.read<SignOutCubit>().signOut();
    },
```

```
            child: Text(l10n.signOutButtonText),
          ),
        ),
      ],
    ),
  ),
 );
 }
}
```

lib\feature\home\presentation\views\sign_up_view.dart

```
import 'package:electronic_student_journal/core/app/router/app_router.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/blocs/user_changes_bloc.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/sign_up_form.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

class SignUpView extends StatelessWidget {
 const SignUpView({
  this.userRole,
  super.key,
 });

 final String? userRole;

 @override
 Widget build(BuildContext context) {
  final l10n = AppLocalizations.of(context)!;
  return BlocListener<UserChangesBloc, UserChangesState>(
   listener: (_, state) {
    state.whenOrNull(
     userSingsOut: () => appRouter.go(Routes.signIn.path),
    );
   },
   child: Scaffold(
    appBar: AppBar(
     title: Text(l10n.signUpUser),
    ),
    body: SingleChildScrollView(
     reverse: true,
     padding: EdgeInsets.only(bottom: 8.h),
     child: userRole == UserRole.admin
       ? const SignUpForm()
       : const Center(
         child: CircularProgressIndicator(),
        ),
    ),
   ),
  );
 }
}
```

lib\feature\home\presentation\widgets\add_score_date_button.dart

```
import 'package:electronic_student_journal/feature/home/domain/entities/score_entity.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/scores_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/scores_names_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/students_cubit.dart';
import 'package:electronic_student_journal/utils/ext/score_date.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:overlay_support/overlay_support.dart';

class AddScoreDateButton extends StatelessWidget {
```

```
 const AddScoreDateButton({super.key});

 @override
 Widget build(BuildContext context) {
  return BlocBuilder<ScoresNamesCubit, ScoresNamesState>(
    builder: (context, scoresNameState) =>
        BlocBuilder<StudentsCubit, StudentsState>(
      builder: (context, studentsState) {
        return ElevatedButton(
          onPressed: () async {
            final scoresNamesCubit = context.read<ScoresNamesCubit>();
            final scoresCubit = context.read<ScoresCubit>();

            final pickedDate = await showDatePicker(
              context: context,
              initialDate: DateTime.now(),
              firstDate: DateTime(1900),
              lastDate: DateTime(2101),
            );

            if (pickedDate != null) {
              if (!scoresNameState.scoresNames.contains(pickedDate)) {
                scoresNamesCubit.addScoreName(pickedDate);

                final students = studentsState.students;
                for (final student in students) {
                  scoresCubit.addScore(
                    ScoreEntity(
                      studentUid: student.uid,
                      score: -1,
                      date: pickedDate,
                    ),
                  );
                }
              } else {
                toast(
                  '${pickedDate.scoreDateFormat()} is already in the list!',
                );
              }
            }
          },
          child: Icon(
            Icons.add,
            size: 16.r,
          ),
        );
      },
    ),
  );
 }
}
```

lib\feature\home\presentation\widgets\add_scores_table_button.dart

```
import 'package:electronic_student_journal/core/app/router/app_router.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/blocs/user_changes_bloc.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:go_router/go_router.dart';

class AddScoresTableButton extends StatelessWidget {
 const AddScoresTableButton({super.key});

 @override
 Widget build(BuildContext context) {
  return ElevatedButton(
    onPressed: () => context.go(
      Routes.createScoresTable.path,
```

```
        extra: (context.read<UserChangesBloc>(), null),
      ),
      style: ElevatedButton.styleFrom(
        backgroundColor: Theme.of(context).scaffoldBackgroundColor,
        foregroundColor: Theme.of(context).primaryColor,
      ),
      child: Icon(
        Icons.add,
        size: 72.r,
      ),
    );
  }
}
```

lib\feature\home\presentation\widgets\add_student_button.dart

```
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/show_student_search_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:provider/provider.dart';

class AddStudentButton extends StatelessWidget {
  const AddStudentButton({super.key});

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return Consumer<ShowStudentSearchProvider>(
      builder: (_, showStudentSearchProvider, __) => ElevatedButton(
        onPressed: () => showStudentSearchProvider.toggleVisibility(),
        child: Text(l10n.addStudent),
      ),
    );
  }
}
```

lib\feature\home\presentation\widgets\add_student_form.dart

```
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/find_students_params.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/find_students_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/score_name_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/show_student_search_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/add_student_button.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/score_item_widget.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:provider/provider.dart';

class AddStudentForm extends StatelessWidget {
  const AddStudentForm({
    required this.teacher,
    super.key,
  });

  final UserEntity teacher;

  @override
  Widget build(BuildContext context) {
    return Consumer<ScoreNameProvider>(
      builder: (context, scoreNameProvider, child) {
        if (scoreNameProvider.scoreName != null) {
          return Consumer<ShowStudentSearchProvider>(
            builder: (_, showStudentSearchProvider, __) => Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                SizedBox(
                  width: 200.w,
```

```
                height: 50.h,
                child: const AddStudentButton(),
              ),
              if (showStudentSearchProvider.showStudentSearch)
                Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    Padding(
                      padding: EdgeInsets.symmetric(vertical: 8.h),
                      child: SizedBox(
                        width: 250.w,
                        height: 50.h,
                        child: TextField(
                          onChanged: (query) {
                            context.read<FindStudentsCubit>().findStudents(
                                  FindStudentsParams(
                                    studentNameQuery: query,
                                    teacherUniversity: teacher.university!,
                                  ),
                                );
                          },
                        ),
                      ),
                    ),
                  ],
                )
              else
                const SizedBox.shrink(),
              BlocBuilder<FindStudentsCubit, FindStudentsState>(
                builder: (context, state) {
                  return state.maybeWhen(
                    success: (students) {
                      if (students.isNotEmpty) {
                        return Column(
                          children: [
                            ...students.map(
                              (student) => ScoreItemWidget(student: student),
                            ),
                          ],
                        );
                      } else {
                        return const SizedBox.shrink();
                      }
                    },
                    loading: () => const Center(
                      child: CircularProgressIndicator(),
                    ),
                    orElse: SizedBox.shrink,
                  );
                },
              )
            ],
          ),
        );
      } else {
        return const SizedBox.shrink();
      }
    },
  );
 }
}
```

lib\feature\home\presentation\widgets\create_table_button.dart

```
import 'package:electronic_student_journal/core/app/di/injector.dart';
import 'package:electronic_student_journal/core/app/router/app_router.dart';
import 'package:electronic_student_journal/feature/home/data/models/scores_table_model.dart';
import 'package:electronic_student_journal/feature/home/domain/params/edit_table_params.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/create_table_cubit.dart';
```

```
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_user_data_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/scores_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/scores_table_name_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:provider/provider.dart';

class CreateTableButton extends StatelessWidget {
  const CreateTableButton({
    required GlobalKey<FormState> formKey,
    super.key,
  }) : _formKey = formKey;

  final GlobalKey<FormState> _formKey;

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return BlocProvider<CreateTableCubit>(
      create: (context) => injector(),
      child: Consumer<ScoresTableNameProvider>(
        builder: (_, scoresTableNameProvider, __) =>
            BlocBuilder<GetUserDataCubit, GetUserDataState>(
          builder: (context, getUserDataState) {
            return SizedBox(
              width: 150.w,
              height: 50.h,
              child: BlocBuilder<ScoresCubit, ScoresState>(
                builder: (context, scoresState) {
                  return ElevatedButton(
                    onPressed: () {
                      if (_formKey.currentState!.validate()) {
                        _formKey.currentState!.save();

                        final createdTable = ScoresTableModel(
                          name: scoresTableNameProvider.tableName!,
                          createdAt: DateTime.now(),
                          ownerUid: getUserDataState.whenOrNull(
                                success: (userEntity) => userEntity.uid,
                              ) ??
                              '',
                          uid: '',
                        );

                        context.read<CreateTableCubit>().createTable(
                              EditTableParams(
                                table: createdTable,
                                scores: scoresState.scores
                                    .map((score) => score.toModel())
                                    .toList(),
                              ),
                            );

                        appRouter.go(Routes.home.path);
                      }
                    },
                    child: Text(l10n.create),
                  );
                },
              ),
            );
          },
        ),
      ),
    );
  }
}
```

lib\feature\home\presentation\widgets\delete_table_button.dart

```
import 'package:electronic_student_journal/core/app/router/app_router.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/scores_table_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/table_params.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/delete_table_cubit.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';

class DeleteTableButton extends StatelessWidget {
  const DeleteTableButton({
    required this.table,
    super.key,
  });

  final ScoresTableEntity table;

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return ElevatedButton(
      onPressed: () {
        context.read<DeleteTableCubit>().deleteTable(
              TableParams(uid: table.uid),
            );
        appRouter.go(Routes.home.path);
      },
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.red,
      ),
      child: Text(l10n.deleteTable),
    );
  }
}
```

lib\feature\home\presentation\widgets\edit_scores_table_form.dart

```
import 'package:electronic_student_journal/core/app/di/injector.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/scores_table_entity.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_scores_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_user_data_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_users_data_cubit_dart_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/scores_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/scores_names_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/students_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/add_score_date_button.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/add_student_form.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/create_table_button.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/delete_table_button.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/score_name_field.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/scores_list_widget.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/scores_table_name_field.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/update_table_button.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

class EditScoresTableForm extends StatelessWidget {
  const EditScoresTableForm({
    this.table,
    super.key,
  });

  final ScoresTableEntity? table;
  static final _formKey = GlobalKey<FormState>();

  @override
```

```
Widget build(BuildContext context) {
 return Form(
  key: _formKey,
  child: Padding(
   padding: EdgeInsets.all(8.h),
   child: table != null
     // Edit existing table
     ? BlocBuilder<GetUserDataCubit, GetUserDataState>(
       builder: (context, state) {
        return state.maybeWhen(
         success: (teacher) {
          return BlocBuilder<GetScoresCubit, GetScoresState>(
           builder: (context, state) {
            return state.maybeWhen(
             success: (scores) {
              final scoresNames = scores
                .map((score) => score.date)
                .toSet()
                .toList();

              final studentsUids = scores
                .map((score) => score.studentUid)
                .toSet()
                .toList();

              return BlocProvider<GetUsersDataCubit>(
               create: (context) =>
                 injector()..getUsersData(studentsUids),
               child: BlocProvider<ScoresNamesCubit>(
                create: (context) => ScoresNamesCubit(
                 scoresNames: scoresNames,
                ),
                child: BlocBuilder<GetUsersDataCubit,
                  GetUsersDataState>(
                 builder: (context, state) {
                  return state.maybeWhen(
                   success: (students) =>
                     BlocProvider<StudentsCubit>(
                    create: (context) =>
                      StudentsCubit(students: students),
                    child: BlocProvider<ScoresCubit>(
                     create: (context) =>
                       ScoresCubit(scores: scores),
                     child: Column(
                      mainAxisAlignment:
                        MainAxisAlignment.center,
                      children: [
                       const ScoresTableNameField(),
                       Row(
                        mainAxisAlignment:
                          MainAxisAlignment.center,
                        children: [
                         BlocBuilder<
                           ScoresNamesCubit,
                           ScoresNamesState>(
                          builder:
                            (context, state) {
                           return ScoreNameField(
                            scoresNames:
                              state.scoresNames,
                           );
                          },
                         ),
                         SizedBox(
                          width: 50.r,
                          height: 50.r,
                          child:
                            const AddScoreDateButton(),
                         ),
```

```
                                ],
                              ),
                              Padding(
                                padding: EdgeInsets.only(
                                  top: 8.h,
                                  bottom: 12.h,
                                ),
                                child: AddStudentForm(
                                  teacher: teacher,
                                ),
                              ),
                              const ScoresListWidget(),
                              Padding(
                                padding: EdgeInsets.only(
                                  top: 24.h,
                                ),
                                child: SizedBox(
                                  width: 200.w,
                                  height: 50.h,
                                  child: UpdateTableButton(
                                    formKey: _formKey,
                                    table: table!,
                                  ),
                                ),
                              ),
                              Padding(
                                padding: EdgeInsets.only(
                                  top: 24.h,
                                ),
                                child: SizedBox(
                                  width: 200.w,
                                  height: 50.h,
                                  child: DeleteTableButton(
                                    table: table!,
                                  ),
                                ),
                              ),
                            ],
                          ),
                        ),
                      ),
                      orElse: () => const Center(
                        child: CircularProgressIndicator(),
                      ),
                    );
                  },
                ),
              ),
            );
          },
          orElse: () => const Center(
            child: CircularProgressIndicator(),
          ),
        );
      },
      orElse: () => const Center(
        child: CircularProgressIndicator(),
      ),
    );
  },
 )
// Create new table
: BlocBuilder<GetUserDataCubit, GetUserDataState>(
    builder: (context, state) {
      return state.maybeWhen(
        success: (userEntity) {
          final teacher = userEntity;
```

```
                return BlocProvider<ScoresNamesCubit>(
                  create: (context) => ScoresNamesCubit(),
                  child: BlocProvider<StudentsCubit>(
                    create: (context) => StudentsCubit(),
                    child: BlocProvider<ScoresCubit>(
                      create: (context) => ScoresCubit(),
                      child: Column(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: [
                          const ScoresTableNameField(),
                          Row(
                            mainAxisAlignment: MainAxisAlignment.center,
                            children: [
                              BlocBuilder<ScoresNamesCubit,
                                  ScoresNamesState>(
                                builder: (context, state) {
                                  return ScoreNameField(
                                    scoresNames: state.scoresNames,
                                  );
                                },
                              ),
                              SizedBox(
                                width: 50.r,
                                height: 50.r,
                                child: const AddScoreDateButton(),
                              ),
                            ],
                          ),
                          Padding(
                            padding:
                                EdgeInsets.only(top: 8.h, bottom: 12.h),
                            child: AddStudentForm(teacher: teacher),
                          ),
                          const ScoresListWidget(),
                          Padding(
                            padding: EdgeInsets.only(top: 50.h),
                            child: CreateTableButton(formKey: _formKey),
                          ),
                        ],
                      ),
                    ),
                  ),
                );
              },
              orElse: () => const Center(
                child: CircularProgressIndicator(),
              ),
            );
          },
        ),
    ),
  );
 }
}
```

lib\feature\home\presentation\widgets\group_form_field.dart

```
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/inherited/group_controller.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/group_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:provider/provider.dart';

class GroupFormField extends StatelessWidget {
  const GroupFormField({super.key});

  @override
  Widget build(BuildContext context) {
```

```
    final l10n = AppLocalizations.of(context)!;
    return Consumer<GroupProvider>(
      builder: (_, groupProvider, __) => TextFormField(
        controller: GroupController.maybeOf(context)?.controller,
        validator: (group) {
          if (group != null && group.isNotEmpty) {
            return null;
          }
          return l10n.groupErrorText;
        },
        onSaved: (newGroup) => groupProvider.changeGroup(newGroup),
        decoration: InputDecoration(
          labelText: l10n.groupLabelText,
          hintText: l10n.groupErrorText,
        ),
        maxLength: 32,
      ),
    );
  }
}
```

lib\feature\home\presentation\widgets\name_form_field.dart

```
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/name_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:provider/provider.dart';

class NameFormField extends StatelessWidget {
  const NameFormField({super.key});

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return Consumer<NameProvider>(
      builder: (_, nameProvider, __) => TextFormField(
        initialValue: nameProvider.name,
        validator: (name) {
          if (name != null && name.isNotEmpty) {
            return null;
          }
          return l10n.nameErrorText;
        },
        onSaved: (newName) => nameProvider.changeName(newName!),
        decoration: InputDecoration(
          labelText: l10n.nameLabelText,
          hintText: l10n.nameErrorText,
        ),
        maxLength: 64,
      ),
    );
  }
}
```

lib\feature\home\presentation\widgets\password_confirmer_form_field.dart

```
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/inherited/password_controller.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/password_confirmer_hinter.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/password_confirmer_provider.dart';
import 'package:electronic_student_journal/feature/sign_in/presentation/viewmodels/password_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:provider/provider.dart';

class PasswordConfirmerFormField extends StatelessWidget {
  const PasswordConfirmerFormField({super.key});

  @override
  Widget build(BuildContext context) {
```

```
    final l10n = AppLocalizations.of(context)!;
    return Consumer3<PasswordConfirmerProvider, PasswordProvider,
        PasswordConfirmerHinter>(
      builder: (
        _,
        passwordConfirmerProvider,
        passwordProvider,
        passwordConfirmerHinter,
        __,
      ) =>
          TextFormField(
        obscureText: passwordConfirmerHinter.isPasswordHinted,
        enableSuggestions: false,
        autocorrect: false,
        validator: (password) =>
            password != PasswordController.of(context).controller.text
                ? l10n.invalidConfirmPassword
                : null,
        onSaved: (newPassword) =>
            passwordConfirmerProvider.changeConfirmedPassword(newPassword!),
        decoration: InputDecoration(
          prefixIcon: const Icon(Icons.lock),
          labelText: l10n.confirmPasswordLabelText,
          hintText: l10n.confirmPasswordHintText,
          suffixIcon: IconButton(
            onPressed: () => passwordConfirmerHinter.toggleVisibility(),
            icon: passwordConfirmerHinter.isPasswordHinted
                ? const Icon(Icons.visibility_off)
                : const Icon(Icons.visibility),
          ),
        ),
      ),
    );
  }
}
```

lib\feature\home\presentation\widgets\patronymic_form_field.dart

```
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/patronymic_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:provider/provider.dart';

class PatronymicFormField extends StatelessWidget {
  const PatronymicFormField({super.key});

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return Consumer<PatronymicProvider>(
      builder: (_, patronymicProvider, __) => TextFormField(
        initialValue: patronymicProvider.patronymic,
        validator: (patronymic) {
          if (patronymic != null && patronymic.isNotEmpty) {
            return null;
          }
          return l10n.patronymicErrorText;
        },
        onSaved: (newPatronymic) =>
            patronymicProvider.changePatronymic(newPatronymic!),
        decoration: InputDecoration(
          labelText: l10n.patronymicLabelText,
          hintText: l10n.patronymicErrorText,
        ),
        maxLength: 64,
      ),
    );
  }
}
```

lib\feature\home\presentation\widgets\role_form_field.dart

```
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/role_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:provider/provider.dart';

class RoleFormField extends StatelessWidget {
  const RoleFormField({super.key});

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return Consumer<RoleProvider>(
      builder: (_, roleProvider, __) => DropdownButtonFormField<String>(
        items: <String>[l10n.student, l10n.teacher]
            .map(
              (value) => DropdownMenuItem(
                value: value,
                child: Text(value),
              ),
            )
            .toList(),
        hint: Text(l10n.selectUserRoleHintText),
        validator: (role) => role == null ? l10n.invalidUserRole : null,
        onChanged: (role) {
          if (role == l10n.student) {
            role = UserRole.student;
          } else if (role == l10n.teacher) {
            role = UserRole.teacher;
          }

          roleProvider.changeRole(role!);
        },
      ),
    );
  }
}
```

lib\feature\home\presentation\widgets\score_item_widget.dart

```
import 'package:electronic_student_journal/feature/home/domain/entities/score_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/find_students_params.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/find_students_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/scores_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/scores_names_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/students_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/show_student_search_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:overlay_support/overlay_support.dart';

class ScoreItemWidget extends StatelessWidget {
  const ScoreItemWidget({
    required this.student,
    super.key,
  });

  final UserEntity student;

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        SizedBox(
```

```
        height: 12.h,
      ),
      BlocBuilder<ScoresNamesCubit, ScoresNamesState>(
        builder: (context, state) {
          final scoresNames = state.scoresNames;

          return BlocBuilder<StudentsCubit, StudentsState>(
            builder: (context, state) {
              final students = state.students;

              return InkWell(
                onTap: () {
                  if (!students.contains(student)) {
                    context.read<StudentsCubit>().addStudent(student);

                    for (final scoreName in scoresNames) {
                      context.read<ScoresCubit>().addScore(
                            ScoreEntity(
                              studentUid: student.uid,
                              score: -1,
                              date: scoreName,
                            ),
                          );
                    }

                    context
                        .read<ShowStudentSearchProvider>()
                        .toggleVisibility();

                    context.read<FindStudentsCubit>().findStudents(
                          const FindStudentsParams(
                            studentNameQuery: '',
                            teacherUniversity: '',
                          ),
                        );
                  } else {
                    toast('Student is already in the list.');
                  }
                },
                child: Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    Text(
                      student.group!,
                      style: TextStyle(
                        fontSize: 18.sp,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                    Padding(
                      padding: EdgeInsets.only(left: 8.w),
                      child: Text(
                        student.fullNameWithInitials,
                        style: TextStyle(
                          fontSize: 24.sp,
                        ),
                      ),
                    )
                  ],
                ),
              );
            },
          );
        },
      ),
    ],
  );
 }
}
```

lib\feature\home\presentation\widgets\score_name_field.dart

```
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/score_name_provider.dart';
import 'package:electronic_student_journal/utils/ext/score_date.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

class ScoreNameField extends StatelessWidget {
  const ScoreNameField({required this.scoresNames, super.key});

  final List<DateTime> scoresNames;

  @override
  Widget build(BuildContext context) {
    return Consumer<ScoreNameProvider>(
      builder: (_, scoreNameProvider, __) => DropdownButton<String>(
        value: scoreNameProvider.scoreName,
        items: scoresNames
            .map(
              (scoreName) => DropdownMenuItem(
                value: scoreName.scoreDateFormat(),
                child: Text(scoreName.scoreDateFormat()),
              ),
            )
            .toList(),
        onChanged: (newScoreName) {
          scoreNameProvider.changeScoreName(newScoreName!);
        },
      ),
    );
  }
}
```

lib\feature\home\presentation\widgets\scores_button.dart

```
import 'package:electronic_student_journal/core/app/router/app_router.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/blocs/user_changes_bloc.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:go_router/go_router.dart';

class ScoresButton extends StatelessWidget {
  const ScoresButton({super.key});

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return ElevatedButton(
      onPressed: () => context.goNamed(
        Routes.scores.name,
        extra: (context.read<UserChangesBloc>(), null),
      ),
      child: Text(l10n.scores),
    );
  }
}
```

lib\feature\home\presentation\widgets\scores_list_widget.dart

```
import 'package:electronic_student_journal/core/app/di/injector.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_user_data_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/scores_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/score_name_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/show_confirm_score_button.dart';
import 'package:electronic_student_journal/utils/ext/score_date.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
```

```
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:provider/provider.dart';

class ScoresListWidget extends StatelessWidget {
  const ScoresListWidget({super.key});

  @override
  Widget build(BuildContext context) {
    return Consumer<ScoreNameProvider>(
      builder: (_, scoreNameProvider, __) =>
          BlocBuilder<ScoresCubit, ScoresState>(
        builder: (context, state) {
          final scores = state.scores
              .where(
                (score) =>
                    score.date.scoreDateFormat() == scoreNameProvider.scoreName,
              )
              .toList();

          return Column(
            children: [
              ...scores.map(
                (score) => ChangeNotifierProvider<ShowConfirmScoreButton>(
                  create: (context) => injector(),
                  child: BlocProvider<GetUserDataCubit>(
                    create: (context) => injector()
                      ..getUserData(
                        score.studentUid,
                      ),
                    child: BlocBuilder<GetUserDataCubit, GetUserDataState>(
                      builder: (context, state) {
                        return state.maybeWhen(
                          success: (student) {
                            final scoreController = TextEditingController(
                              text: !score.score.isNegative
                                  ? score.score.toString()
                                  : '',
                            );

                            final initialScore = score;

                            return Padding(
                              padding: EdgeInsets.only(
                                left: 8.w,
                                top: 8.h,
                              ),
                              child: Row(
                                children: [
                                  Text(
                                    student.fullNameWithInitials,
                                  ),
                                  Padding(
                                    padding: EdgeInsets.only(
                                      left: 8.w,
                                    ),
                                    child: SizedBox(
                                      width: 75.w,
                                      height: 50.h,
                                      child: TextField(
                                        keyboardType: TextInputType.number,
                                        inputFormatters: [
                                          LengthLimitingTextInputFormatter(
                                            3,
                                          )
                                        ],
                                        controller: scoreController,
                                        onChanged: (_) => context
                                            .read<ShowConfirmScoreButton>()
```

```
                                .showButton(),
                          ),
                        ),
                      ),
                      Consumer<ShowConfirmScoreButton>(
                        builder: (_, showConfirmScoreButton, __) =>
                            showConfirmScoreButton.isButtonShown
                                ? Padding(
                                    padding:
                                        EdgeInsets.only(left: 8.w),
                                    child: SizedBox(
                                      width: 50.w,
                                      height: 50.h,
                                      child: ElevatedButton(
                                        onPressed: () {
                                          context
                                              .read<
                                                  ShowConfirmScoreButton>()
                                              .hideButton();
                                          context
                                              .read<ScoresCubit>()
                                              .changeScore(
                                                initialScore:
                                                    initialScore,
                                                score: initialScore
                                                    .toModel()
                                                    .copyWith(
                                                      score:
                                                          int.tryParse(
                                                              scoreController.text,
                                                            ) ??
                                                            -1,
                                                    )
                                                    .toEntity(),
                                              );
                                        },
                                        child: Icon(
                                          Icons.done,
                                          size: 18.sp,
                                        ),
                                      ),
                                    ),
                                  )
                                : const SizedBox.shrink(),
                          )
                        ],
                      ),
                    );
                  },
                  loading: () => const Center(
                    child: CircularProgressIndicator(),
                  ),
                  orElse: SizedBox.shrink,
                );
              },
            ),
          ),
        ),
      )
    ],
  );
 },
 ),
 );
 }
}
```

lib\feature\home\presentation\widgets\scores_table_name_field.dart

```
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/scores_table_name_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:provider/provider.dart';

class ScoresTableNameField extends StatelessWidget {
  const ScoresTableNameField({super.key});

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return Consumer<ScoresTableNameProvider>(
      builder: (_, scoresTableNameProvider, __) => TextFormField(
        initialValue: scoresTableNameProvider.tableName,
        validator: (tableName) {
          if (tableName != null && tableName.isNotEmpty) {
            return null;
          }
          return l10n.tableNameErrorText;
        },
        onSaved: (newTableName) =>
            scoresTableNameProvider.changeScoresTableName(newTableName!),
        decoration: InputDecoration(
          labelText: l10n.tableNameLabelText,
          hintText: l10n.tableNameErrorText,
        ),
        maxLength: 64,
      ),
    );
  }
}
```

lib\feature\home\presentation\widgets\sign_up_form.dart

```
import 'package:electronic_student_journal/core/app/router/app_router.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/user_entity.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/register_user_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/inherited/group_controller.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/inherited/password_controller.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/group_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/name_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/patronymic_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/role_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/surname_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/university_provider.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/group_form_field.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/name_form_field.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/password_confirmer_form_field.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/patronymic_form_field.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/role_form_field.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/surname_form_field.dart';
import 'package:electronic_student_journal/feature/home/presentation/widgets/university_form_field.dart';
import 'package:electronic_student_journal/feature/shared/presentation/widgets/email_form_field.dart';
import 'package:electronic_student_journal/feature/shared/presentation/widgets/password_form_field.dart';
import 'package:electronic_student_journal/feature/sign_in/presentation/viewmodels/email_provider.dart';
import 'package:electronic_student_journal/feature/sign_in/presentation/viewmodels/password_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:overlay_support/overlay_support.dart';
import 'package:provider/provider.dart';

class SignUpForm extends StatelessWidget {
  const SignUpForm({super.key});

  static final _formkey = GlobalKey<FormState>();

  @override
```

```
Widget build(BuildContext context) {
  final l10n = AppLocalizations.of(context)!;
  return Form(
    key: _formkey,
    child: PasswordController(
      child: Padding(
        padding: EdgeInsets.symmetric(horizontal: 12.w),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Padding(
              padding: EdgeInsets.only(
                top: 16.h,
              ),
              child: const EmailFormField(),
            ),
            Padding(
              padding: EdgeInsets.only(
                top: 16.h,
              ),
              child: const PasswordFormField(),
            ),
            Padding(
              padding: EdgeInsets.only(
                top: 12.h,
              ),
              child: const PasswordConfirmerFormField(),
            ),
            Padding(
              padding: EdgeInsets.only(
                top: 24.h,
              ),
              child: const RoleFormField(),
            ),
            GroupController(
              child: Consumer<RoleProvider>(
                builder: (_, roleProvider, __) {
                  if (roleProvider.role == null) {
                    return const SizedBox.shrink();
                  } else {
                    if (roleProvider.role == UserRole.student) {
                      return Column(
                        children: [
                          Padding(
                            padding: EdgeInsets.only(
                              top: 24.h,
                            ),
                            child: const SurnameFormField(),
                          ),
                          Padding(
                            padding: EdgeInsets.only(
                              top: 12.h,
                            ),
                            child: const NameFormField(),
                          ),
                          Padding(
                            padding: EdgeInsets.only(
                              top: 12.h,
                            ),
                            child: const PatronymicFormField(),
                          ),
                          Padding(
                            padding: EdgeInsets.only(
                              top: 12.h,
                            ),
                            child: const UniversityFormField(),
                          ),
                          Padding(
                            padding: EdgeInsets.only(
```

```
                top: 12.h,
              ),
              child: const GroupFormField(),
            ),
          ],
        );
      } else if (roleProvider.role == UserRole.teacher) {
        return Column(
          children: [
            Padding(
              padding: EdgeInsets.only(
                top: 24.h,
              ),
              child: const SurnameFormField(),
            ),
            Padding(
              padding: EdgeInsets.only(
                top: 12.h,
              ),
              child: const NameFormField(),
            ),
            Padding(
              padding: EdgeInsets.only(
                top: 12.h,
              ),
              child: const PatronymicFormField(),
            ),
            Padding(
              padding: EdgeInsets.only(
                top: 12.h,
              ),
              child: const UniversityFormField(),
            ),
          ],
        );
      } else {
        return const SizedBox.shrink();
      }
    }
  },
 ),
),
Padding(
 padding: EdgeInsets.only(top: 32.h),
 child: BlocListener<RegisterUserCubit, RegisterUserState>(
  listener: (_, state) {
   state.whenOrNull(
    success: () => appRouter.go(Routes.home.path),
    failure: (message) =>
      showSimpleNotification(Text(message)),
   );
  },
  child: Consumer6<
    EmailProvider,
    PasswordProvider,
    RoleProvider,
    NameProvider,
    SurnameProvider,
    UniversityProvider>(
   builder: (
    _,
    emailProvider,
    passwordProvider,
    roleProvider,
    nameProvider,
    surnameProvider,
    universityProvider,
    __,
   ) =>
```

```
                  Consumer2<GroupProvider, PatronymicProvider>(
                builder: (_, groupProvider, patronymicProvider, __) =>
                  SizedBox(
                width: 150.w,
                height: 50.h,
                child: ElevatedButton(
                  onPressed: () {
                    if (_formkey.currentState!.validate()) {
                      _formkey.currentState!.save();

                      context.read<RegisterUserCubit>().registerUser(
                          UserEntity(
                            // Uid will be reassigned in register
                            // remote data source
                            uid: '',
                            email: emailProvider.email!,
                            role: roleProvider.role!,
                            registeredAt: DateTime.now(),
                            surname: surnameProvider.surname,
                            name: nameProvider.name,
                            patronymic: patronymicProvider.patronymic,
                            university: universityProvider.university,
                            group: groupProvider.group,
                            fullName: '${surnameProvider.surname} '
                                '${nameProvider.name} '
                                '${patronymicProvider.patronymic}',
                          ),
                          passwordProvider.password!,
                        );
                    }
                  },
                  child: Text(l10n.signUpUserButtonText),
                ),
              ),
            ),
          ),
        ),
      ),
    ],
  ),
),
),
),
);
}
}
```

lib\feature\home\presentation\widgets\surname_form_field.dart

```
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/surname_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:provider/provider.dart';

class SurnameFormField extends StatelessWidget {
  const SurnameFormField({super.key});

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return Consumer<SurnameProvider>(
      builder: (_, surnameProvider, __) => TextFormField(
        initialValue: surnameProvider.surname,
        validator: (surname) {
          if (surname != null && surname.isNotEmpty) {
            return null;
          }
          return l10n.surnameErrorText;
        },
        onSaved: (newSurname) => surnameProvider.changeSurname(newSurname!),
```

```
        decoration: InputDecoration(
          labelText: l10n.surnameLabelText,
          hintText: l10n.surnameErrorText,
        ),
        maxLength: 64,
      ),
    );
  }
}
```

lib\feature\home\presentation\widgets\university_form_field.dart

```
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/university_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:provider/provider.dart';

class UniversityFormField extends StatelessWidget {
  const UniversityFormField({super.key});

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return Consumer<UniversityProvider>(
      builder: (_, universityProvider, __) => TextFormField(
        initialValue: universityProvider.university,
        validator: (university) {
          if (university != null && university.isNotEmpty) {
            return null;
          }
          return l10n.universityErrorText;
        },
        onSaved: (newUniversity) =>
            universityProvider.changeUniversity(newUniversity!),
        decoration: InputDecoration(
          labelText: l10n.universityLabelText,
          hintText: l10n.universityErrorText,
        ),
        maxLength: 128,
      ),
    );
  }
}
```

lib\feature\home\presentation\widgets\update_table_button.dart

```
import 'package:electronic_student_journal/core/app/di/injector.dart';
import 'package:electronic_student_journal/core/app/router/app_router.dart';
import 'package:electronic_student_journal/feature/home/data/models/scores_table_model.dart';
import 'package:electronic_student_journal/feature/home/domain/entities/scores_table_entity.dart';
import 'package:electronic_student_journal/feature/home/domain/params/edit_table_params.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/get_user_data_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/scores_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/cubits/update_table_cubit.dart';
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/providers/scores_table_name_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:provider/provider.dart';

class UpdateTableButton extends StatelessWidget {
  const UpdateTableButton({
    required GlobalKey<FormState> formKey,
    required this.table,
    super.key,
  }) : _formKey = formKey;

  final GlobalKey<FormState> _formKey;
```

```
 final ScoresTableEntity table;

 @override
 Widget build(BuildContext context) {
  final l10n = AppLocalizations.of(context)!;
  return BlocProvider<UpdateTableCubit>(
   create: (context) => injector(),
   child: Consumer<ScoresTableNameProvider>(
    builder: (_, scoresTableNameProvider, __) =>
      BlocBuilder<GetUserDataCubit, GetUserDataState>(
     builder: (context, getUserDataState) {
      return SizedBox(
       width: 150.w,
       height: 50.h,
       child: BlocBuilder<ScoresCubit, ScoresState>(
        builder: (context, scoresState) {
         return ElevatedButton(
          onPressed: () {
           if (_formKey.currentState!.validate()) {
            _formKey.currentState!.save();

            final createdTable = ScoresTableModel(
             name: scoresTableNameProvider.tableName!,
             createdAt: table.createdAt,
             ownerUid: getUserDataState.whenOrNull(
                  success: (userEntity) => userEntity.uid,
                 ) ??
                 '',
             uid: table.uid,
            );

            context.read<UpdateTableCubit>().updateTable(
                 EditTableParams(
                  table: createdTable,
                  scores: scoresState.scores
                    .map((score) => score.toModel())
                    .toList(),
                 ),
                );

            appRouter.go(Routes.home.path);
           }
          },
          child: Text(l10n.updateTable),
         );
        },
       ),
      );
     },
    ),
   ),
  );
 }
}
```

lib\feature\shared\data\datasources\firebase_auth_remote_data_source_impl.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/shared/data/datasources/firebase_auth_remote_data_source.dart';
import 'package:electronic_student_journal/feature/shared/domain/params/user_credentials_params.dart';
import 'package:firebase_auth/firebase_auth.dart';

class FirebaseAuthRemoteDataSourceImpl implements FirebaseAuthRemoteDataSource {
 final _firebaseAuth = FirebaseAuth.instance;

 @override
 Future<Either<Failure, User>> signIn(
  UserCredentialsParams params,
```

```
  ) async {
    try {
      final userCredential = await _firebaseAuth.signInWithEmailAndPassword(
        email: params.email,
        password: params.password,
      );

      return Right(userCredential.user!);
      // return const Left(EmptyDataFailure('No data'));
    } on FirebaseAuthException catch (e) {
      return Left(ServerFailure(e.message ?? 'Server failure.'));
    } catch (e) {
      return Left(SomeFailure(e.toString()));
    }
  }

  @override
  Future<Either<Failure, void>> signOut() async {
    try {
      await _firebaseAuth.signOut();
      return const Right(null);
    } catch (e) {
      return Left(SomeFailure(e.toString()));
    }
  }

  @override
  Either<Failure, Stream<User?>> getUserChangesStream() {
    try {
      return Right(_firebaseAuth.authStateChanges());
    } catch (e) {
      return Left(Failure(e.toString()));
    }
  }
}
```

lib\feature\shared\data\datasources\firebase_auth_remote_data_source.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/shared/domain/params/user_credentials_params.dart';
import 'package:firebase_auth/firebase_auth.dart';

abstract interface class FirebaseAuthRemoteDataSource {
  Future<Either<Failure, User>> signIn(UserCredentialsParams signInParams);

  Future<Either<Failure, void>> signOut();

  Either<Failure, Stream<User?>> getUserChangesStream();
}
```

lib\feature\shared\data\repositories\auth_repository_impl.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/shared/data/datasources/firebase_auth_remote_data_source.dart';
import 'package:electronic_student_journal/feature/shared/domain/params/user_credentials_params.dart';
import 'package:electronic_student_journal/feature/shared/domain/repositories/auth_repository.dart';
import 'package:firebase_auth/firebase_auth.dart';

class AuthRepositoryImpl implements AuthRepository {
  AuthRepositoryImpl({
    required FirebaseAuthRemoteDataSource firebaseRemoteDataSource,
  }) : _firebaseRemoteDataSource = firebaseRemoteDataSource;

  final FirebaseAuthRemoteDataSource _firebaseRemoteDataSource;

  @override
  Future<Either<Failure, User>> signIn(UserCredentialsParams params) async {
```

```
    final response = await _firebaseRemoteDataSource.signIn(params);
    return response;
  }

  @override
  Future<Either<Failure, void>> signOut() async {
    final response = await _firebaseRemoteDataSource.signOut();
    return response;
  }

  @override
  Either<Failure, Stream<User?>> getUserChangesStream() {
    return _firebaseRemoteDataSource.getUserChangesStream();
  }
}
```

lib\feature\shared\domain\params\user_credentials_params.dart

```
import 'package:freezed_annotation/freezed_annotation.dart';

part 'user_credentials_params.freezed.dart';

@freezed
class UserCredentialsParams with _$UserCredentialsParams {
  const factory UserCredentialsParams({
    required String email,
    required String password,
  }) = _UserCredentialsParams;
}
```

lib\feature\shared\domain\repositories\auth_repository.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/feature/shared/domain/params/user_credentials_params.dart';
import 'package:firebase_auth/firebase_auth.dart';

abstract interface class AuthRepository {
  Future<Either<Failure, User>> signIn(UserCredentialsParams signInParams);

  Future<Either<Failure, void>> signOut();

  Either<Failure, Stream<User?>> getUserChangesStream();
}
```

lib\feature\shared\presentation\widgets\email_form_field.dart

```
import 'package:electronic_student_journal/feature/sign_in/presentation/viewmodels/email_provider.dart';
import 'package:electronic_student_journal/utils/ext/auth_string.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:provider/provider.dart';

class EmailFormField extends StatelessWidget {
  const EmailFormField({
    this.maxLength = 128,
    super.key,
  });

  final int? maxLength;

  @override
  Widget build(BuildContext context) {
    final l10n = AppLocalizations.of(context)!;
    return Consumer<EmailProvider>(
      builder: (_, emailProvider, __) => TextFormField(
        validator: (email) => email != null
            ? (!email.isValidEmail() ? l10n.invalidEmail : null)
            : null,
```

```
      onSaved: (newEmail) => emailProvider.changeEmail(newEmail!),
      decoration: InputDecoration(
        prefixIcon: const Icon(Icons.person),
        labelText: l10n.emailLabelText,
        hintText: l10n.emailHintText,
      ),
      keyboardType: TextInputType.emailAddress,
      maxLength: maxLength,
    ),
  );
 }
}
```

lib\feature\shared\presentation\widgets\password_form_field.dart

```
import 'package:electronic_student_journal/feature/home/presentation/viewmodels/inherited/password_controller.dart';
import 'package:electronic_student_journal/feature/sign_in/presentation/viewmodels/password_hinter.dart';
import 'package:electronic_student_journal/feature/sign_in/presentation/viewmodels/password_provider.dart';
import 'package:electronic_student_journal/utils/ext/auth_string.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:provider/provider.dart';

class PasswordFormField extends StatelessWidget {
 const PasswordFormField({
  this.maxLength = 128,
  super.key,
 });

 final int? maxLength;

 @override
 Widget build(BuildContext context) {
  final l10n = AppLocalizations.of(context)!;
  return Consumer2<PasswordProvider, PasswordHinter>(
   builder: (_, passwordProvider, passwordHinter, __) => TextFormField(
    controller: PasswordController.maybeOf(context)?.controller,
    obscureText: passwordHinter.isPasswordHinted,
    enableSuggestions: false,
    autocorrect: false,
    validator: (password) => password != null
      ? (!password.isValidPassword() ? l10n.invalidPassword : null)
      : null,
    onSaved: (newPassword) => passwordProvider.changePassword(newPassword!),
    decoration: InputDecoration(
     prefixIcon: const Icon(Icons.lock),
     labelText: l10n.passwordLabelText,
     hintText: l10n.passwordHintText,
     errorMaxLines: 2,
     suffixIcon: IconButton(
      onPressed: () => passwordHinter.toggleVisibility(),
      icon: passwordHinter.isPasswordHinted
        ? const Icon(Icons.visibility_off)
        : const Icon(Icons.visibility),
     ),
    ),
    maxLength: maxLength,
   ),
  );
 }
}
```

lib\feature\sign_in\domain\usecases\sign_in_usecase.dart

```
import 'package:dartz/dartz.dart';
import 'package:electronic_student_journal/core/error/failure.dart';
import 'package:electronic_student_journal/core/usecase/usecase.dart';
import 'package:electronic_student_journal/feature/shared/domain/params/user_credentials_params.dart';
import 'package:electronic_student_journal/feature/shared/domain/repositories/auth_repository.dart';
```

```
import 'package:firebase_auth/firebase_auth.dart';

class SignInUseCase extends AsyncUseCase<User, UserCredentialsParams> {
  SignInUseCase({required AuthRepository authRepository})
      : _authRepository = authRepository;

  final AuthRepository _authRepository;

  @override
  Future<Either<Failure, User>> call(UserCredentialsParams params) async =>
      _authRepository.signIn(params);
}
```

lib\feature\sign_in\presentation\viewmodels\email_provider.dart

```
import 'package:flutter/foundation.dart';

class EmailProvider extends ChangeNotifier {
  EmailProvider({String? email}) : _email = email;

  String? _email;

  String? get email => _email;

  void changeEmail(String email) {
    _email = email;
    notifyListeners();
  }
}
```

lib\feature\sign_in\presentation\viewmodels\password_hinter.dart

```
import 'package:flutter/material.dart';

class PasswordHinter extends ChangeNotifier {
  bool _isPasswordHinted = true;

  bool get isPasswordHinted => _isPasswordHinted;

  void toggleVisibility() {
    _isPasswordHinted = !_isPasswordHinted;
    notifyListeners();
  }
}
```

lib\feature\sign_in\presentation\viewmodels\password_provider.dart

```
import 'package:flutter/foundation.dart';

class PasswordProvider extends ChangeNotifier {
  PasswordProvider({String? password}) : _password = password;

  String? _password;

  String? get password => _password;

  void changePassword(String password) {
    _password = password;
    notifyListeners();
  }
}
```

lib\feature\sign_in\presentation\viewmodels\sign_in_cubit.dart

```
import 'package:electronic_student_journal/core/app/di/injector.dart';
import 'package:electronic_student_journal/feature/shared/domain/params/user_credentials_params.dart';
import 'package:electronic_student_journal/feature/sign_in/domain/usecases/sign_in_usecase.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
```

```
import 'package:freezed_annotation/freezed_annotation.dart';
import 'package:logger/logger.dart';

part 'sign_in_cubit.freezed.dart';
part 'sign_in_state.dart';

class SignInCubit extends Cubit<SignInState> {
  SignInCubit({
    required SignInUseCase signInUseCase,
  }) : _signInUseCase = signInUseCase,
       super(const _Loading());

  final SignInUseCase _signInUseCase;

  Future<void> signIn(UserCredentialsParams params) async {
    emit(const _Loading());
    final response = await _signInUseCase.call(params);

    response.fold(
      (failure) => emit(_Failure(failure.message)),
      (user) {
        injector<Logger>().i(user.toString());
        emit(_Success(user));
      },
    );
  }
}
```

lib\feature\sign_in\presentation\viewmodels\sign_in_state.dart

```
part of 'sign_in_cubit.dart';

@freezed
class SignInState with _$SignInState {
  const factory SignInState.initial() = _Initial;

  const factory SignInState.loading() = _Loading;

  const factory SignInState.success(User user) = _Success;

  const factory SignInState.failure(String message) = _Failure;
}
```

lib\feature\sign_in\presentation\views\sign_in_view.dart

```
import 'package:electronic_student_journal/feature/sign_in/presentation/widgets/sign_in_form.dart';
import 'package:electronic_student_journal/gen/assets.gen.dart';
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

class SignInView extends StatelessWidget {
  const SignInView({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        reverse: true,
        child: Column(
          children: [
            Padding(
              padding: EdgeInsets.only(
                top: 120.h,
                bottom: 100.h,
              ),
              child: Container(
                width: 250.h,
                height: 250.h,
                decoration: BoxDecoration(
```

```
                border: const Border.fromBorderSide(BorderSide()),
                borderRadius: const BorderRadius.all(Radius.circular(6)),
                image: DecorationImage(
                  image: Assets.images.logo.image().image,
                ),
              ),
            ),
          ),
          SignInForm(),
        ],
      ),
    ),
  );
 }
}
```

lib\feature\sign_in\presentation\widgets\sign_in_form.dart

```
import 'package:electronic_student_journal/core/app/router/app_router.dart';
import 'package:electronic_student_journal/feature/shared/domain/params/user_credentials_params.dart';
import 'package:electronic_student_journal/feature/shared/presentation/widgets/email_form_field.dart';
import 'package:electronic_student_journal/feature/shared/presentation/widgets/password_form_field.dart';
import 'package:electronic_student_journal/feature/sign_in/presentation/viewmodels/email_provider.dart';
import 'package:electronic_student_journal/feature/sign_in/presentation/viewmodels/password_provider.dart';
import 'package:electronic_student_journal/feature/sign_in/presentation/viewmodels/sign_in_cubit.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:overlay_support/overlay_support.dart';
import 'package:provider/provider.dart';

class SignInForm extends StatelessWidget {
 SignInForm({super.key});

 final _formkey = GlobalKey<FormState>();

 @override
 Widget build(BuildContext context) {
  final l10n = AppLocalizations.of(context)!;
  return Form(
   key: _formkey,
   child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
     Padding(
      padding: EdgeInsets.fromLTRB(12.w, 32.h, 12.w, 16.h),
      child: const EmailFormField(
       maxLength: null,
      ),
     ),
     Padding(
      padding: EdgeInsets.fromLTRB(12.w, 0, 12.w, 24.h),
      child: const PasswordFormField(
       maxLength: null,
      ),
     ),
     BlocListener<SignInCubit, SignInState>(
      listener: (context, state) {
       state.whenOrNull(
        success: (_) {
         appRouter.go(Routes.home.path);
        },
        failure: (message) => showSimpleNotification(Text(message)),
       );
      },
      child: Consumer2<EmailProvider, PasswordProvider>(
       builder: (_, emailProvider, passwordProvider, __) => SizedBox(
        width: 150.w,
```

```
              height: 50.h,
              child: ElevatedButton(
                onPressed: () async {
                  if (_formkey.currentState!.validate()) {
                    _formkey.currentState!.save();
                    await context.read<SignInCubit>().signIn(
                          UserCredentialsParams(
                            email: emailProvider.email!,
                            password: passwordProvider.password!,
                          ),
                        );
                  }
                },
                child: Text(l10n.signInButtonText),
              ),
            ),
          ),
        ),
      ],
    ),
  );
 }
}
```

lib\utils\ext\auth_string.dart

```
extension AuthString on String {
  bool isValidEmail() => contains(
        RegExp(
          r'^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$',
        ),
      );

  bool isValidPassword() => length > 6;
}
```

lib\utils\ext\score_date.dart

```
import 'package:intl/intl.dart';

extension ScoreDate on DateTime {
  String scoreDateFormat() => DateFormat('dd/MM/yyyy').format(this);
}
```

ДОДАТОК Б

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна_робота_Захаров.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна_робота_Захаров.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Диплом_Захаров.zip	Архів. Містить коди програми і скомпільовану програму
Презентація	
Презентація_Захаров.pptx	Презентація кваліфікаційної роботи