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

Інститут електроенергетики
(інститут)
факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

# ПОЯСНЮВАЛЬНА ЗАПИСКА
## кваліфікаційної роботи ступеня магістра
(бакалавра, спеціаліста, магістра)

**Студента** Шипоші Артема Андрійовича
(ПІБ)
**академічної групи** 126-22м
(шифр)
**спеціальності** 126 « Інформаційні системи та технології»
(код і назва спеціальності)
**за освітньо-професійною програмою** «Інформаційні системи та технології»
(офіційна назва)
**на тему** Розробка інформаційної технології пошуку ЗВО для навчання на основі платформи Firebase
(назва за наказом ректора)

| Керівники | Прізвище, ініціали | Оцінка за шкалою | | Підпис |
|---|---|---|---|---|
| | | рейтинговою | інституційною | |
| кваліфікаційної роботи | проф. Коротенко Г.М. | | | |
| розділів: | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

| | | | | |
|---|---|---|---|---|
| **Рецензент** | доц. Галушко О.М. | | | |

| | | | | |
|---|---|---|---|---|
| **Нормоконтролер** | проф. Коротенко Г.М.. | | | |

**Дніпро**
**2023**

**ЗАТВЕРДЖЕНО:**
завідувач кафедри

___інформаційних технологій___
___та комп'ютерної інженерії___
(повна назва)

_____ __Гнатушенко В.В.__
(підпис) (прізвище, ініціали)

«_____»_____ 2023_року

**ЗАВДАННЯ**
**на кваліфікаційну роботу**
**ступеня ____магістр_____**
(бакалавра, магістра)

**студенту** _____Шипоші А. А._____ **академічної групи** ____126-22м___
(прізвище та ініціали) (шифр)

**спеціальності** ___126 « Інформаційні системи та технології »_____

**за освітньою-професійною програмою** _____

_____«Інформаційні системи та технології»_____

**на тему** ____Розробка інформаційної технології пошуку ЗВО для навчання на основі

_____платформи Firebase _____

затверджену наказом ректора НТУ «Дніпровська політехніка» від 9.10.2023 р. № 1227-с

| Розділ | Зміст | Термін виконання |
|---|---|---|
| Розділ 1. | На основі матеріалів із зібраних джерел проаналізувати стан області рішення задачі | 9.10.2023 – 24.10.2023 |
| Розділ 2. | Визначити основні підходи до вибору інструментів для створення мобільного застосунку | 24.10.2023 – 11.11.2023 |
| Розділ 3. | Користуючись матеріалами науково-технічних джерел обрати необхідні методи та засоби й реалізувати проєктні рішення | 12.11.2023 – 1.12.2023 |

**Завдання видано** _____ _____Коротенко Г.М._____
(підпис керівника) (прізвище, ініціали)

**Дата видачі** _____01.02.2023 р._____

**Дата подання до екзаменаційної комісії** _____18.12.2023 р._____

**Прийнято до виконання** _____ _____Шипоша А.А._____
(підпис студента) (прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка: 122 сторінки, 35 рисунків, 4 додатки, 23 джерела.

Об'єкт розробки: інформаційна технологія пошуку ЗВО для навчання на основі платформи Firebase.

Мета кваліфікаційної роботи: створення застосунку для пошуку ЗВО для навчання на основі платформи Firebase, що відповідає стандартам інформаційної технології.

У вступі подано стан проблеми, проведено пошук аналогів, визначена мета роботи та план її виконання. Також аргументовано актуальність створення подібного застосунку та визначено задачі, які вимагають рішення.

В першому розділі пояснювальної записки визначено поняття інформаційної технології, проведено огляд роботи створеного в роботі бакалавра додатку, обґрунтовано необхідність вдосконалення існуючої системи та визначено необхідні зміни для відповідності стандартам інформаційної технології.

У другому розділі розглянуто основні підходи до розробки додатку. Розглянуто платформу Firebase та проведено його порівняння з конкурентами. Було визначено інструмент для роботи з серверною частиною та функціонал, що повинна містити панель адміністратора. Проведено порівняння інструментів для створення дизайну додатків та обґрунтовано вибір.

Третій розділ демонструє процес створення дизайну та розробки додатків користувача та адміністратора.

Практичне значення розробки полягає у тому що, створений додаток дозволяє спростити доступ абітурієнтам ЗВО до пошуку необхідної інформації.

МОБІЛЬНИЙ ДОДАТОК, ЗАСТОСУНОК, FLUTTER, FIREBASE, CLOUD FIRESTORE, FIGMA, USER FLOW, DESIGN, ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, ВИБІР ЗАКЛАДУ ВИЩОЇ ОСВІТИ

## ABSTRACT

Explanatory note: 92 pages, 29 figures, 4 appendices, 30 sources.

Object of development: information technology for university search based on the Firebase platform.

The purpose of the qualification work: creation of an application for university search based on the Firebase platform.

The introduction outlines the problem statement, conducts a search for analogs, defines the work's objective, and outlines its execution plan. The relevance of creating such an application is justified, and tasks requiring resolution are identified.

In the first section of the explanatory note, the concept of information technology is defined, and an overview of the bachelor's application is provided. The need for improving the existing system is justified, and necessary changes are outlined to align with information technology standards.

The second section explores the main approaches to application development. Firebase platform is examined, and a comparison with competitors is made. The tool for working with the server-side and the functionality for the administrator panel are identified. Tools for designing applications are compared, and the choice is justified. The practical significance of the development lies in simplifying access for university applicants to the necessary information.

The third section illustrates the process of designing and developing user and administrator applications.

The practical significance of the development is that the created application allows to facilitate the access of university entrants to the necessary information.

MOBILE APPLICATION, APP, FLUTTER, FIREBASE, CLOUD FIRESTORE, FIGMA, USER FLOW, DESIGN, INFORMATION TECHNOLOGY, HIGHER EDUCATION INSTITUTION SELECTION

**ЗМІСТ**

# ВСТУП

В сучасному цифровому епохальному віці, технології впливають на всі аспекти нашого життя, включаючи освіту. З моменту появи мобільних застосунків та веб-додатків виникла потреба в постійному вдосконаленні та розвитку програмних продуктів для забезпечення актуальності та ефективності.

Темою кваліфікаційної роботи є “Розробка інформаційної технології пошуку ЗВО для навчання на основі платформи Firebase”.

В роботі бакалавра було створено додаток, який допомагає абітурієнтам обрати вищий навчальний заклад для навчання. Створений продукт пропонує стабільну роботу, інтуїтивний інтерфейс та непоганий функціонал.

Але створення додатків не обмежується лише інтерфейсом та функціоналом. Постійна необхідність адаптації та реагування на змінні умови вимагає додавання найновіших технологій та методів розробки. Особливо важливим є управління даними, забезпечення безпеки та надійності інформації.

У даному контексті важливим є також удосконалення системи управління даними та забезпечення надійного бекенду. З метою покращення ефективності та розширення можливостей додатку для огляду вищих навчальних закладів України, планується перенесення даних на платформу *Firebase Firestore*.

Зокрема, планується розробка адміністративної панелі, яка надасть можливість зручно та ефективно додавати та редагувати інформацію про університети. Ця функціональність відкриє нові перспективи управління контентом та дозволить підтримувати актуальність інформації у реальному часі.

Додатковими аспектами вдосконалення є розширення критеріїв пошуку навчальних закладів та впровадження можливості зберігати обрані виші до свого списку.

У світлі зазначених вдосконалень, магістерська робота спрямована на створення не лише зручного та інформативного інструменту для вибору вищих навчальних закладів, але й на реалізацію технології, яка буде стійкою до майбутніх технологічних викликів та вимог сучасного користувача.

Актуальність даної роботи обумовлена тим, що в Україні бракує ефективних додатків для огляду та вибору закладу вищої освіту зі зручним пошуком інформації та постійними оновленнями даних.

Задачі кваліфікаційної роботи:

- аналіз необхідних покращень для додатку;
- пошук альтернатив серверній частині;
- аналіз рішень для покращення пошукової системи додатку;
- розробка інформаційної технології;
- опис всіх задіяних в додатку програмних компонентів;
- опис кінцевого варіанту реалізації проекту.

Об'єктом дослідження є оптимізація користувацького досвіду мобільного додатку для пошуку та огляду вищих навчальних закладів України.

Предметом дослідження є необхідність забезпечення інформаційних систем серверною частиною.

# РОЗДІЛ 1.
# АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ

## 1.1. Поняття інформаційної технології

Інформаційна технологія [1] (IT) – це широкий термін, який визначає використання комп'ютерних систем, програмного забезпечення та засобів зв'язку для створення, управління та оптимізації процесів обробки інформації в рамках конкретного застосунку або сервісу. Це включає в себе використання технічних рішень для збору, зберігання, обробки та передачі даних, спрямованих на вдосконалення функціональності та взаємодії користувача з додатком.

У сучасному програмуванні та розробці програмного забезпечення, IT включає в себе весь спектр технологій, від вибору мов програмування та використання баз даних до впровадження хмарних сервісів та мобільних розробок. Застосовуючи IT у контексті додатка, розробники використовують сучасні технічні можливості для створення продукту, який ефективно вирішує завдання та задовольняє потреби користувачів.

У розробці додатків інформаційна технологія охоплює декілька ключових аспектів. По-перше, це включає в себе вибір оптимальних мов програмування та інструментів розробки для досягнення поставлених цілей. Використання сучасних мов програмування, таких як *Flutter*, *Python* чи *Swift*, дозволяє створювати ефективний та швидкий код.

IT також включає в себе роботу з даними. Розробники використовують бази даних для зберігання та організації інформації, забезпечуючи швидкий доступ та ефективну обробку даних. Вибір відповідної системи управління базами даних, такої як *MySQL, PostgreSQL* чи *MongoDB*, визначає надійність та продуктивність додатка.

Безпека є критичним аспектом інформаційної технології в розробці додатка. Розробники повинні використовувати шифрування даних, механізми

аутентифікації та інші засоби для захисту від несанкціонованого доступу та атак. Вирішення питань безпеки є необхідним етапом у процесі створення додатка для забезпечення конфіденційності та цілісності даних користувачів.

Інформаційні технології також охоплюють використання хмарних та мобільних технологій. Використання хмарних сервісів, таких як *AWS* чи *Google Cloud*, дозволяє зберігати та обробляти дані віддалено. Мобільні технології включають розробку додатків для платформ *Android* та *iOS*, забезпечуючи мобільну доступність та зручний інтерфейс для користувачів.

### 1.2. Огляд роботи існуючого додатку

Робота бакалавра полягала в розробці фрагменту застосунку на платформі ОС Андроїд для обрання вищого навчального закладу за допомогою фреймворку *Flutter*. Пропоную переглянути, як додаток працює та які має функції.

#### 1.2.1. Сплеш скрін

Додаток зустрічає користувача вітальним екраном (рис 1.1), який триває 3 секунди і служить буфером між запуском додатку і його основними екранами.

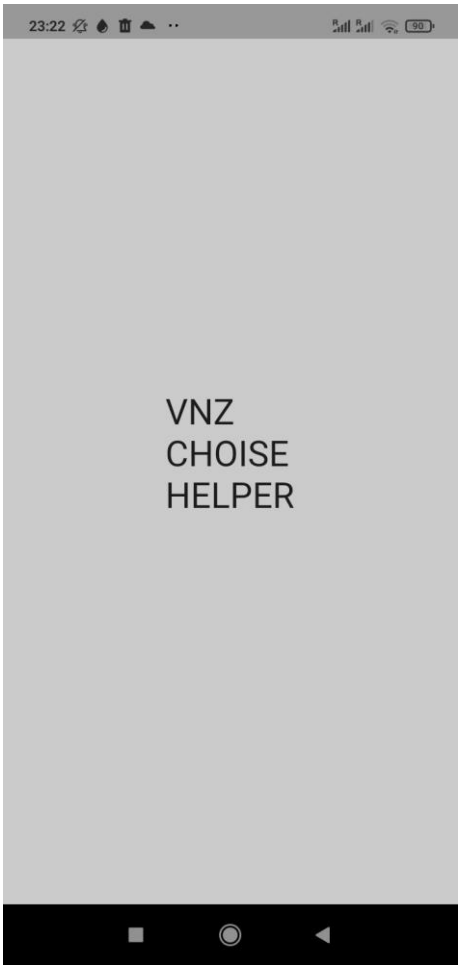


Рис 1.1. Вигляд сплеш скріну додатку

Сплеш-екрани [2] (також відомі як екрани завантаження) у додатках використовуються для передачі короткого, але значущого візуального враження користувачам під час завантаження додатка. Ці екрани мають декілька функцій та важливість в контексті користувацького досвіду. Сплеш-екрани можуть містити логотипи, кольорові схеми та інші елементи, які відображають бренд додатка. Вони використовуються для визначення бренду та створення консистентності. Сплеш-екрани дозволяють створити позитивне перше враження про додаток. Вони можуть викликати емоційну реакцію та створити атмосферу для подальшого використання. Створюючи цей додаток, на меті також було поставлено передати суть додатку, який сфокусований на одній задачі та не містить нічого зайвого.

Також, такі екрани можуть вказати користувачеві, що додаток активно завантажується. Вони служать індикатором того, що додаток відпрацьовує

запуск і невдовзі буде готовий до використання. Сплеш-екрани можуть використовуватися для завантаження ключового контенту або ресурсів, необхідних для подальшого коректного функціонування додатка. В нашому випадку на цій сторінці відбувається ініціалізація списку університетів, від роботи з якими і залежить створений продукт.

### 1.2.2. Головний екран

Одразу після екрану завантаження користувач потрапляє на головний екран (рис 1.2).

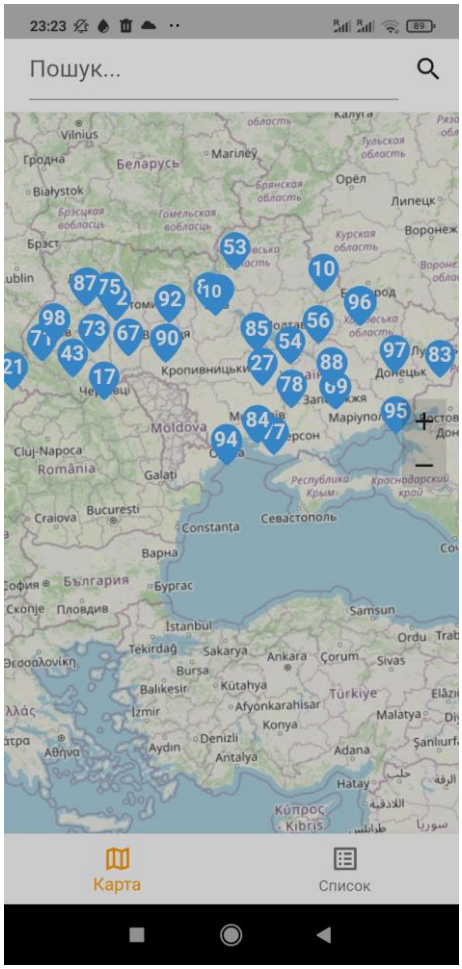


Рис 1.2. Вигляд головного екрану додатку

Головний екран додатка [3] — це центральна точка взаємодії користувача з програмою. Розробка головного екрану відбувається

відповідно до ряду стандартів та кращих практик для забезпечення зручності користувача та ефективності використання додатка.

Ці практики включають:

- Забезпечення зручного доступу до основних функцій через логічну та інтуїтивно зрозумілу навігацію.
- Використання стандартів меню та підменю для організації функцій.
- Використання лейаутів, які забезпечують чітке групування контенту та функцій.
- Розміщення важливих елементів ближче до центру або у верхній частині екрану.
- Використання однорідної колірної схеми, яка відповідає бренду додатка.
- Забезпечення чіткої видимості та контрасту для кращої читабельності.
- Використання чітких та підписаних іконок для швидкого впізнавання функцій.
- Забезпечення адаптації до різних розмірів екранів, щоб додаток виглядав природно на різних пристроях.
- Додавання анімацій та інших інтерактивних ефектів для зрозумілості та задоволення користувача.
- Забезпечення відповідей на дії користувача без зайвого затримання.
- Підтримка альбомної та портретної орієнтацій екрану.

Хочеться відзначити, що головний екран створеного додатку відповідає більшості даних стандартів. В застосунку інтуїтивно розташовані елементи, іконки підібрані для швидкого впізнавання функцій, використана однорідна колірна схема та інше.

Головний екран пропонує нижню навігаційну панель з двома функціями: перегляд мапи та перегляд списку університетів. На екрані з

мапою користувач може зумити мапу за допомогою жестів або елементів управління справа у вигляді символів “+” та “-”.

У верхній панелі розташоване поле пошуку, яке орієнтується на назву університету. При натисканні на нього та введені символів випадає вікно з результатами (рис 1.3).

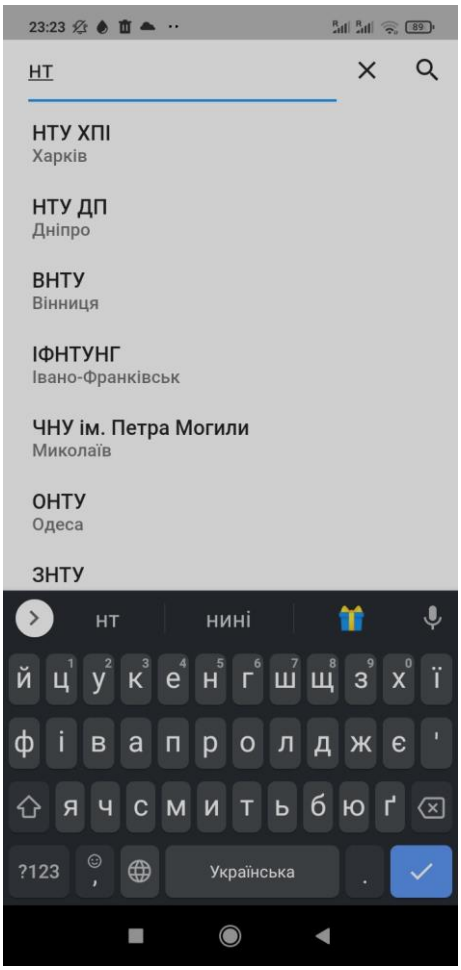


Рис 1.3. Вигляд панелі пошуку

Якщо натиснути на маркер на карті, користувачу відкриється вікно з короткою інформацією про університет (рис 1.4).

На цьому вікні користувач може виділити для себе основну інформацію щодо університету, а саме: позиція в рейтингу, повна назва університету, рейтинг в *Google Maps*, адреса та веб-сайт університету. Для отримання більш детальної інформації, внизу є кнопка “детальніше”.

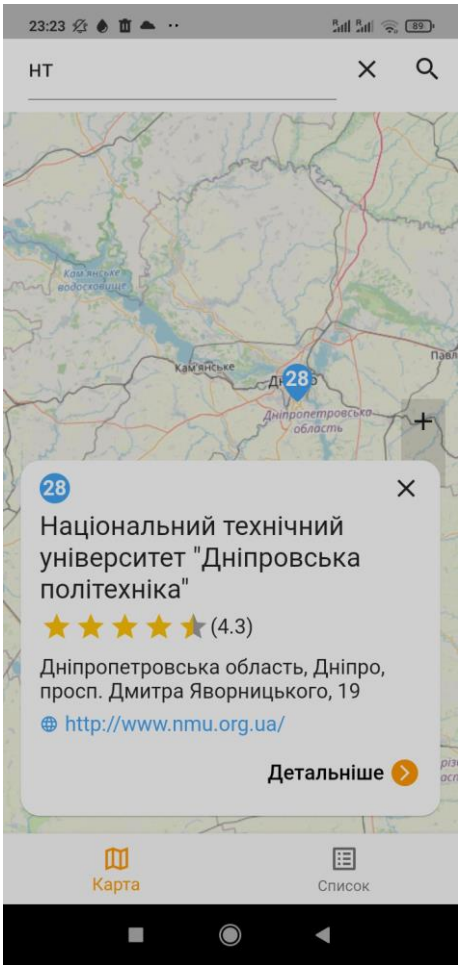


Рис 1.4. Вікно з короткою інформацією про університет

При натисканні на кнопку “список” в нижній навігаційній панелі користувач міняє вкладку на екран з представленням даних по вишам у вигляді списку (рис 1.5).


Рис 1.5. Вигляд екрану списку університетів

На цьому екрані користувач також має змогу користуватись пошуковою панеллю або натиснути на університет, щоб подивитись деталі.

### 1.2.3. Екран детальної інформації про університет

Якщо на екрані списку університетів (рис 1.5) натиснути на елемент зі списку або на вікні з короткою інформацією по вишу (рис 1.4) натиснути на кнопку “детальніше”, користувач потрапить на екран з детальною інформацією по університету (рис 1.6-1.7).

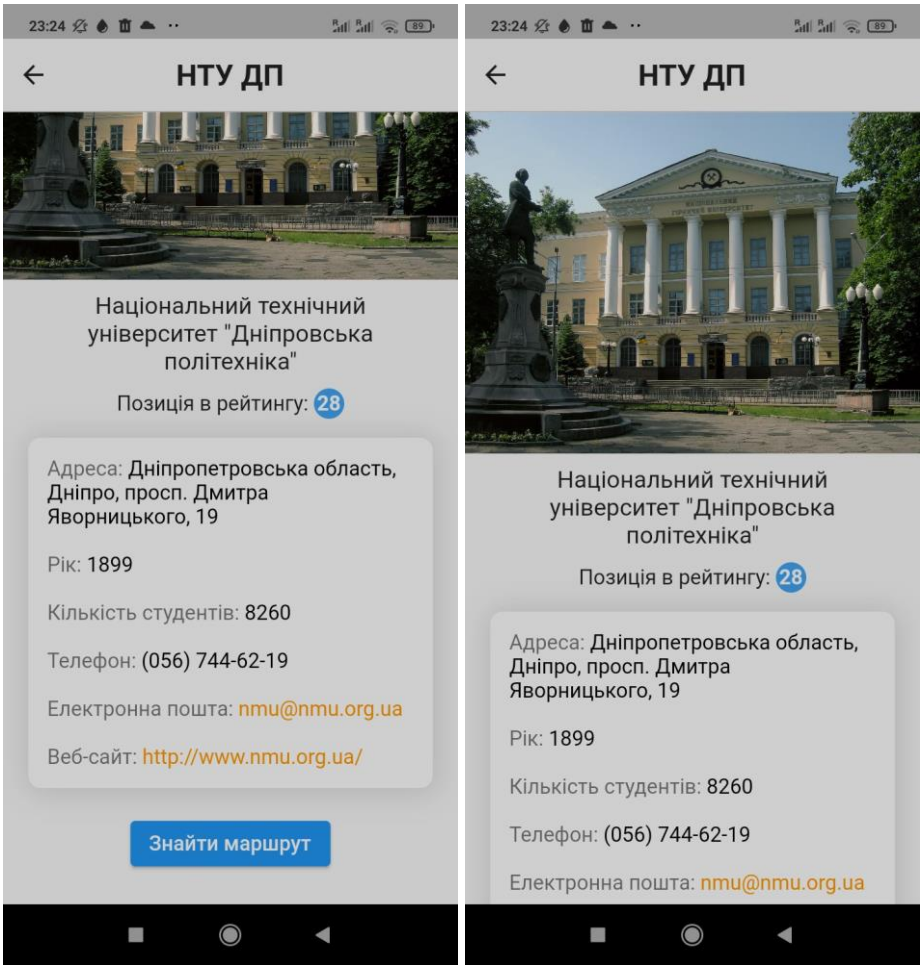


Рис 1.6-1.7. Екран списку університетів

Екран демонструє юзеру інформацію про назву університету, його фотографію, рейтингову позицію, адресу, рік заснування, кількість студентів, телефон, електронну пошту та веб сайт.

Електронна пошта та веб-сайт реагують на дотик, що веде до взаємодії з, відповідно, додатком для пошти та браузером.

Також, внизу екрану є кнопка “знайти маршрут”, яка, як зрозуміло з назви, пропонує юзеру побудувати маршрут до даного вишу. При натисканні на кнопку користувачу запропонує відкрити додаток *Google Maps*, в якому користувач і зможе знайти дорогу до маркеру, що вказує на розташування вишу.

### 1.3. Підґрунтя для необхідності у вдосконаленнях

Темою даної кваліфікаційної роботи є створення інформаційної технології. Як вже було сказано, інформаційна технологія має на меті використання технічних рішень для збору, зберігання, обробки та передачі даних, спрямованих на вдосконалення функціональності та взаємодії користувача з додатком. Оглянувши роботу додатка, можна прийти до висновку що існуюча система не відповідає вимогам інформаційної технології.

Щоб досягти поставленої мети, додатку необхідно змінити роботу з даними, розширити свій функціонал та запровадити систему збору даних про користувацький досвід, щоб в подальшому мати змогу змінювати та покращувати систему на основі отриманої інформації.

### 1.4. Аналіз необхідних змін додатку для відповідності стандартам інформаційної технології

Робота з даними проводиться безпосередньо всередині додатку, тобто додаток залежний від постійних оновлень для підтримання актуальної інформації. Щоб зробити з додатку інформаційну технологію, він повинен залежати від інформації, що надходить з серверу, щоб забезпечити користувачів свіжими даними. Для полегшення досвіду оновлення даних, потрібно зробити адміністративну панель.

Серед однозначних недоліків пошукової системи можна виділити те, що пошук проводиться лише по назві вищого навчального закладу. Таким чином, додаток однозначно потребує розширити можливості пошуку. Для цього можуть бути використані: пошук по місту чи області, по місцю в рейтингу вишів та додавання ключових слів, що описували б університет.

Екран списку університетів однозначно потребує функції сортування за рейтингом у списку, за роком заснування університету, кількістю студентів або оцінкою в *Google Maps*.

Створеному додатку однозначно не вистачає функції “збережене”, аби надати користувачам можливість зберігати виші, щоб в подальшому мати спосіб швидко повернутись до обраного університету.

Для постійного аналізу користувацького досвіду, в застосунку повинен бути присутнім інструмент для збору аналітики. За допомогою зібраних даних про продуктивність, поведінку користувачів і стан системи можна ефективно виявляти та вирішувати проблеми, підвищувати ефективність, оптимізувати витрати ресурсів та забезпечувати безпеку. Цей підхід також сприяє стратегічному плануванню та прийняттю рішень, забезпечуючи підстави для подальшого розвитку та вдосконалення інформаційно-технічного середовища.

# РОЗДІЛ 2. ОСНОВНІ ПІДХОДИ ДО ВИБОРУ ІНСТРУМЕНТІВ ДЛЯ СТВОРЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

## 2.1. Платформа Firebase

У світі стрімкого розвитку інформаційних технологій, створення та підтримка ефективних додатків вимагає не тільки творчості та інновацій, але й відмінної інфраструктури для забезпечення надійності та продуктивності. В умовах постійного зростання обсягів даних та вимог до взаємодії з користувачем, вибір правильної платформи стає стратегічним вирішенням для розробників та підприємств, особливо коли на меті стоїть створення інформаційної системи.

Виникає необхідність в зручному та функціональному інструментарії, що сприятиме швидкому впровадженню нового функціоналу, забезпечить безпеку даних та дозволить ефективно взаємодіяти з аудиторією. В цьому контексті, вибір платформи *Firebase* стає стратегічним кроком для розробників, що прагнуть поєднати потужність інструментів розробки зі спрощеною архітектурою та готовою інфраструктурою в хмарі.

*Firebase* [4, 5] *(рис 2.1)* – це повний стек розробки мобільних та веб-застосунків, який надає широкий спектр інструментів та сервісів для спрощення розробки, впровадження та управління додатками. Створена компанією *Google*, *Firebase* відома своєю простотою використання та інтеграцією з різноманітними платформами.

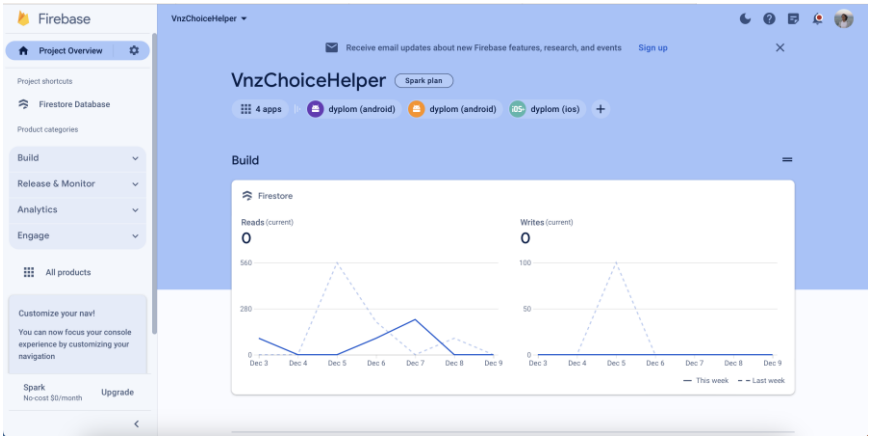


Рис 2.1. Інтерфейс платформи Firebase

Платформа *Firebase* пропонує до використання велику кількість інструментів, призваних полегшити роботу з впровадженням повідомлень, аналітики, відстежування крашів, автентифікації, хмарного сховища, а/б тестингу, реклами, хостингу та багато чого іншого.

*Firebase* користується популярністю та визнанням серед глобальної спільноти розробників завдяки своїй потужності та гнучкості, що відзначається не лише початківцями, але і досвідченими професіоналами. Ця платформа широко використовується у розробці різноманітних додатків, починаючи від мобільних ігор до корпоративних веб-застосунків.

На приклад, додаток "*Instacart*"[6], який відомий своєю популярністю в галузі електронної комерції, використовує *Firebase* для забезпечення реального часу оновлюваної інформації про замовлення та стану товарів. Також, "*Shazam*", додаток для розпізнавання музики, використовує *Firebase* для аналітики та збереження користувацьких налаштувань.

Широкий спектр використання *Firebase* включає в себе також додатки, спрямовані на соціальну взаємодію, освіту, медіа та багато інших галузей. Його інтеграція з різноманітними сервісами дозволяє розробникам швидко створювати функціональні та ефективні додатки, сприяючи тим самим подальшому розширенню індустрії мобільного та веб-розробки.

Дана платформа не лише приводить до зниження витрат часу та ресурсів на створення складної інфраструктури, але й відкриває двері для

реалізації широкого спектру функціоналу: від ефективного управління базою даних до впровадження аналітики та залучення користувачів. У цьому контексті, важливо розглядати *Firebase* як інтегровану та готову до використання платформу, яка стає основою для успішного створення та розвитку повноцінних інформаційних технологій.

## 2.2. Серверна частина додатку

У сучасному цифровому ландшафті, де швидкість та доступність є ключовими складовими успішної роботи програмних застосунків, серверна частина виявляється необхідною опорою для ефективного функціонування та взаємодії з клієнтськими додатками. Сервер — це не лише технічна інфраструктура, але й стратегічний компонент, який визначає продуктивність, безпеку та масштабованість інформаційного середовища.

Створений в роботі бакалавра додаток по суті своїй є мінімально життєздатним продуктом (*MVP*) [7]. Він є стратегічним першим кроком у вивченні ринку та визначенні його прийняття серед користувачів. *MVP*-підходи вимагають невеликої витрати часу та ресурсів для швидкого випробування ідеї та збору відгуків користувачів. У такому контексті слід розглянути серверні рішення, які не потребували б великого часу на створення та імплементацію в додаток.

### 2.2.1. *Firebase Firestore*

*Firebase Firestore* [8] - це розподілена база даних, розроблена компанією *Google*, яка надає інтегровану хмарну інфраструктуру для зберігання та синхронізації даних у реальному часі для веб- і мобільних додатків. *Firestore* використовує гнучку структуру даних у вигляді колекцій та документів, де кожен документ містить поле та значення. Це надає розробникам можливість зручно моделювати та організовувати дані. Однак важливість *Firestore* виявляється не лише в його структурі, а й у можливості

роботи в реальному часі, де зміни в базі даних автоматично відображаються на всіх підключених пристроях без додаткових запитів до сервера. Це робить *Firestore* ідеальним вибором для додатків, де актуальність даних та миттєва синхронізація мають критичне значення. Крім того, *Firestore* легко інтегрується з іншими сервісами *Firebase*, надаючи розробникам готовий набір інструментів для реалізації різноманітних функціональностей, таких як аутентифікація користувачів і зберігання файлів. Його висока масштабованість, безпека та можливість роботи в офлайн-режимі роблять Firestore потужним інструментом для ефективної розробки та управління базами даних у хмарному середовищі.

Такі серверні рішення, як *Firebase Firestore*, виявляються найкращим вибором. Їх легкість використання, швидкість розгортання та гнучка модель даних ідеально підходять для *MVP*-продуктів, де швидкий цикл розробки та можливість масштабування стають важливими критеріями успіху.

*Firebase Firestore* відзначається не лише своєю функціональністю та ефективністю, але і вигідністю з точки зору вартості утримання. У порівнянні з традиційними серверними рішеннями, які часто вимагають великих капіталовкладень у закупівлю та утримання обладнання, Firebase *Firestore* пропонує гнучку модель тарифікації. Спрощена система визначення цін *Firebase Firestore* враховує використання ресурсів, що дозволяє оптимізувати витрати.

Окрім цього, *Firebase Firestore* використовує підхід "*pay-as-you-go*" (рис 2.2-2.3), де користувач сплачує лише за те, що використовує. Це особливо вигідно для початкових етапів розвитку додатків, коли обсяги даних та навантаження можуть бути невеликими. Середньостатистична ціна на утримання серверів для *Firebase Firestore* може бути значно нижчою порівняно із традиційними серверними рішеннями, забезпечуючи при цьому високий рівень ефективності та доступності.

| Free tier | Quota |
|---|---|
| Stored data | 1 GiB |
| Document reads | 50,000 per day |
| Document writes | 20,000 per day |
| Document deletes | 20,000 per day |
| Network egress | 10 GiB per month |

| | Free quota per day | Price beyond the free quota (per unit) | Price unit |
|---|---|---|---|
| **Document Reads** | 50,000 | $0.06 | per 100,000 documents |
| **Document Writes** | 20,000 | $0.18 | per 100,000 documents |
| **Document Deletes** | 20,000 | $0.02 | per 100,000 documents |
| **TTL Deletes** | Not supported | $0.02 | per 100,000 documents |
| **Stored Data** | 1 GiB storage | $0.18 | GiB/Month |
| **PITR data** | Not supported | $0.18 | GiB/Month |
| **Backup data** | Not supported | $0.03 | GiB/Month |
| **Restore operation** | Not supported | $0.4 | GiB |

Рис 2.2-2.3. Ціни за використання *Firebase Firestore*

Здійснюючи огляд розширених можливостей *Firebase Firestore*, важливо також розглянути альтернативні серверні рішення, які пропонують схожі функціональність та можливості. Перелік конкурентів *Firebase Firestore* дозволить зрозуміти вибір серед інших потенційних рішень для визначених потреб у розробці та управлінні базою даних.

### 2.2.2. *Amazon DynamoDB (AWS)*

*Amazon DynamoDB* [9] - це повністю керована облачна база даних від *Amazon Web Services (AWS)*, яка вирізняється високою масштабованістю, низькою затримкою та високою доступністю. Вона використовує модель ключ-значення, де кожен елемент бази даних представлений унікальним ключем. *DynamoDB* є *NoSQL* рішенням, що дозволяє зберігати та отримувати дані швидко та ефективно, а також розміщувати їх у різних регіонах для оптимізації швидкодії в різних частинах світу. Завдяки автоматичній

масштабованості та високій пропускній здатності, *DynamoDB* стає відмінним вибором для великих та розподілених систем, де важливо підтримувати високий рівень продуктивності та доступності.

Запущений у 2012 році, *DynamoDB* є ключовим сервісом *Amazon Web Services (AWS)* для опрацювання значних обсягів даних. Його популярність поширена серед підприємств різних розмірів, завдяки вражаючій швидкості та масштабованості.

Порівняно з *Firebase Firestore*, *Amazon DynamoDB* може бути сприйнятий як більш складний у використанні через більші можливості та налаштування, які він пропонує. Відсутність функціональності в реальному часі в базі даних *DynamoDB* вимагає додаткових зусиль для розробки схеми синхронізації даних, яка дозволить управляти змінами в реальному часі, що може бути менш прямолінійним у порівнянні з вбудованим підходом *Firebase Firestore*. Однак *DynamoDB*, завдяки широким можливостям та конфігураційним опціям, є більш гнучким рішенням для великих та складних проектів, де важливо забезпечити точне налаштування бази даних під конкретні потреби.

Для розрахування вартості використання є спеціальний калькулятор, який дозволить визначити вартість, яку доведеться сплачувати, але у *DynamoDB* також є безкоштовний план.

### 2.2.3. *MongoDB Atlas*

*MongoDB Atlas* [10] - це облачний сервіс управління базами даних, який надається компанією *MongoDB*. Цей сервіс побудований на основі популярної *NoSQL* бази даних *MongoDB* і спрямований на спрощення управління та розгортанням MongoDB в хмарному середовищі. *MongoDB Atlas* пропонує високу масштабованість, гнучкість та надійність для збереження та опрацювання даних.

У порівнянні з *Firebase Firestore*, *MongoDB Atlas* може вважатися більш гнучким і розширеним рішенням, оскільки *MongoDB* використовує

гнучку схему документів у форматі *BSON*. Це означає, що дані можуть мати різні поля, а сама схема може змінюватися з часом. *MongoDB Atlas* також пропонує широкий набір опцій щодо реплікації та шардування, що полегшує обробку великих обсягів даних та підтримку високого рівня доступності.

Однак, в порівнянні з *Firebase Firestore*, *MongoDB Atlas* може вимагати більше конфігурування та управління самостійно, оскільки він не має вбудованих інструментів для роботи в реальному часі та автоматичного масштабування, як у *Firestore*. Також важливо відзначити, що *MongoDB Atlas* може бути особливо ефективним для великих проєктів, де важливо мати повний контроль над базою даних та використовувати багатофункціональні можливості *MongoDB*. Існуючий проект не можна назвати великим, тому *MongoDB* залишається мало сенсу віддавати перевагу.

### 2.2.4. *Microsoft Azure Cosmos DB*

*Microsoft Azure Cosmos DB* [11] - це хмарна база даних від компанії Microsoft, яка надає глобальну реплікацію та розподіленим шардуванням даних. Це розширюване *NoSQL* рішення, призначене для швидкого та масштабованого зберігання даних в різних моделях даних, таких як документи, ключ-значення, колонкові та графові дані. *Azure Cosmos DB* вирізняється гнучкістю та можливістю підтримувати кілька API для взаємодії з різними моделями даних.

У порівнянні з *Firebase Firestore*, *Microsoft Azure Cosmos DB* може вважатися складнішим за використанням через більшу кількість налаштувань та опцій, доступних для користувача. Але в той же час, ця гнучкість дозволяє розробникам точно налаштувати базу даних під конкретні вимоги проєкту. *Azure Cosmos DB* має вбудовану можливість глобальної реплікації, що робить його ідеальним вибором для додатків, які мають глобальних користувачів та потребують низької затримки.

Однак важливо враховувати, що розгортання та управління *Azure Cosmos DB* може вимагати більше часу та експертності в порівнянні з

*Firebase Firestore*, де часто пропонується більше автоматизації та готових інструментів для спрощення розробки та управління базою даних.

### 2.3. Проектування панелі адміністратора

В даній кваліфікаційній роботі планується створити окремий додаток-панель адміністратора, що дозволив би полегшити управління інформацією на серверній частині без необхідності відкривати сам *Firebase*.

Панель адміністратора [12] є дуже зручним, а іноді необхідним елементом будь-якого додатка, надаючи адміністраторам широкий спектр інструментів для ефективного управління та контролю. Використання панелі адміністратора є невід'ємним елементом для успішної реалізації цілей атестаційної роботи, спрямованої на створення інформаційної технології для пошуку закладів вищої освіти.

Суть панелі адміністратора буде заключатись в копіюванні можливостей існуючого додатку, але з додатковими функціями для адмінів. Головний екран повинен містити інтерактивний елемент для переводу на нову сторінку, функція якої буде в заповненні полей моделі університету, який планується додати. На сторінці списку університетів планується також додати кнопку до кожного елементу списку, що дозволила би редагувати інформацію існуючого вишу. При натисканні на цю кнопку, адміністратора повинно перенести на ту саму сторінку з заповненням полей університету, але інформація про вибраний університет вже буде заповнена.

Панель адміністратора дозволить забезпечити зручний та безпечний доступ до додавання та редагування інформації про університети. Інтерфейс панелі адміністратора буде спроектований таким чином, щоб запровадити інтуїтивно зрозуміле користування та максимальну продуктивність для адміністративного персоналу.

#### 2.3.1. Вибір мови програмування для панелі адміністратора

Існуючий додаток, на основі якого буде створено панель адміністратора, вже використовує фреймворк *Flutter*, що в свою чергу використовує мову програмування Dart.

*Flutter* [13, 14] - це відкритий фреймворк розробки користувацьких інтерфейсів (UI) створений компанією *Google*. Однією з ключових особливостей *Flutter* є можливість розробки крос-платформених додатків, які працюють як на *iOS*, так і на *Android*, а також на вебі та інших платформах. Це досягається завдяки використанню власного движка рендерингу *Dart*, який дозволяє створювати відмінно оптимізовані та високопродуктивні додатки.

Однією з переваг *Flutter* є його швидкість розробки та гнучкість. Він пропонує легке створення гарного інтерфейсу користувача за допомогою власного набору віджетів та можливості створення власних. Також, *Flutter* активно оновлюється та має активну спільноту, що робить його сучасним та підтримуємим рішенням для розробки мобільних додатків.

Завдяки *Flutter*, розробникам не потрібно переписувати весь код для кожної платформи, що робить процес розробки ефективнішим та економічнішим. Код, написаний на *Dart*, може бути використаний для створення якісних та продуктивних додатків, які легко адаптуються для різних мобільних платформ. Це спрощує випуск додатку в *App Store* та інші магазини без додаткових труднощів, що є значущим плюсом при розгляді стратегій розробки та планування майбутнього розвитку додатку.

Зазвичай адміністративні панелі створюють використовуючи мови для програмування веб-сторінок для відображення її у баузері, але в нашому випадку буде створено додаток на *Flutter*, що може бути набагато зручнішим за браузерну версію, оскільки при змінах в базі даних адміністратор одразу зможе побачити їх у тому ж додатку. Але в перспективі може бути запущена веб-версія додатку з адміністративною панеллю, оскільки серед платформ, що підтримує флаттер, присутній Web.

### 2.4. Дизайн продукту

Створення дизайну додатка є критично важливим етапом, який визначає зовнішній вигляд і функціональність продукту, а також впливає на загальний враження користувача. Перший крок у цьому процесі - визначення стилістики та концепції дизайну. Важливо враховувати аудиторію додатка, її уподобання та потреби, щоб забезпечити вдалу взаємодію та позитивні враження користувача.

Важливим елементом дизайну є створення єдинообразного та консистентного дизайну інтерфейсу. Це включає в себе вибір кольорової палітри, шрифтів, стилів кнопок та інших елементів. Кожен компонент повинен гармонійно вписуватися в загальний стиль додатка, створюючи єдиний та логічний образ.

### 2.4.1. Вибір інструменту для створення дизайну

Існує безліч інструментів для створення дизайну додатків, які відрізняються за функціональністю, простотою використання та можливостями співпраці. Ось перелік трьох найпопулярніших інструментів:

1) *Sketch*

*Sketch* [15] (рис 2.4) - це інструмент для дизайну векторної графіки, призначений основним чином для створення інтерфейсів та веб-дизайну.

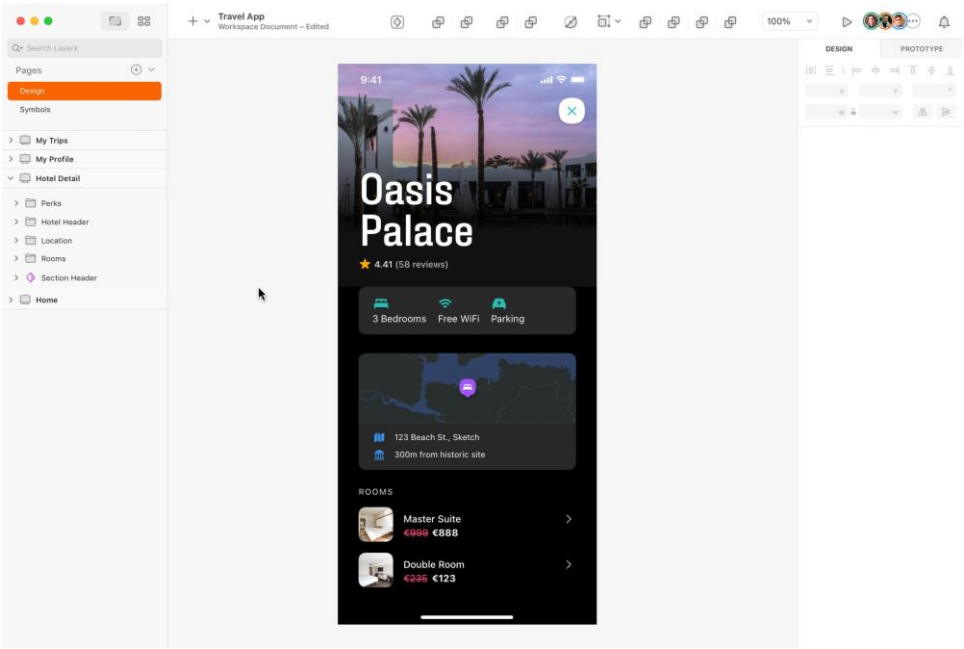


Рис 2.4. Інтерфейс платформи *Sketch*

Його найбільшою перевагою є простота використання та швидкість роботи, що робить його вибором для багатьох дизайнерів. Інтуїтивний інтерфейс дозволяє швидко оволодіти основами програми, а широка гама інструментів для векторного малювання дозволяє створювати складні інтерфейсні елементи.

Окрім того, *Sketch* добре інтегрується з іншими інструментами та ресурсами для дизайну, що дозволяє легко імпортувати та експортувати файли, співпрацювати з растровою графікою та використовувати внутрішню бібліотеку для зберігання стандартних компонентів. Це дозволяє швидко створювати консистентний та професійний дизайн.

Однак, однією з обмежень *Sketch* є його доступність лише для користувачів *macOS*, що може створювати труднощі для команд, які використовують різні операційні системи. Також відсутня можливість спільної роботи в режимі реального часу, що робить його менш придатним для командної роботи в порівнянні з хмарними альтернативами.

*2) Figma*

*Figma* [16] (рис 2.5) виступає як передовий інструмент для дизайну та прототипування, що надає користувачам беззмінну можливість спільної роботи в реальному часі, що робить його винятково потужним для командної співпраці. Здатність бачити зміни, вносимі учасниками команди в реальному часі, сприяє ефективній взаємодії та швидкому узгодженню дизайну.

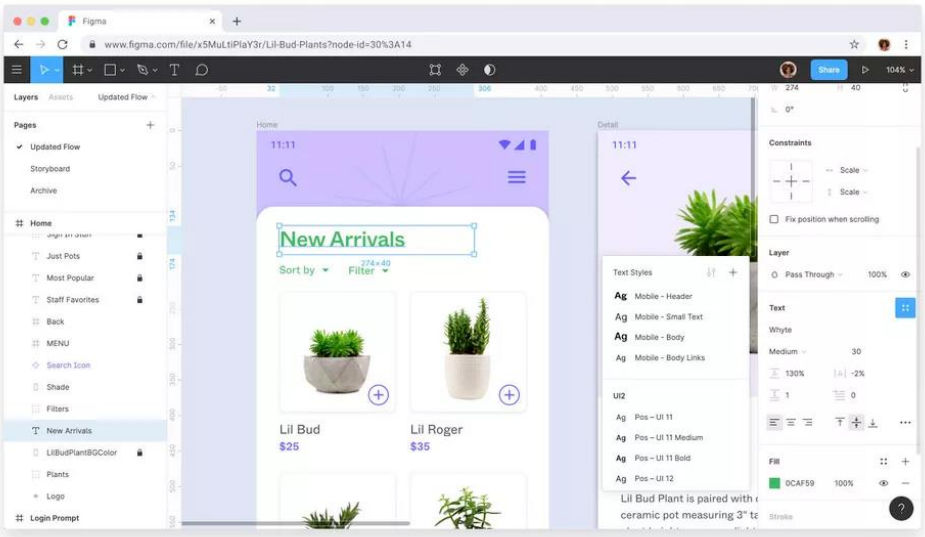


Рис 2.5. Інтерфейс *Figma*

*Figma* є вибором, що враховує сучасні потреби розробників та дизайнерів, оскільки він не вимагає установки та працює з будь-якого пристрою через веб-браузер. Його хмарна інфраструктура гарантує доступність проектів з будь-якого місця та прискорює процеси командної роботи. *Figma* також вражає своєю широкою функціональністю векторного дизайну, підтримкою прототипування та інтеграцією з іншими інструментами.

Навіть у тих випадках, коли доступ до Інтернету тимчасово відсутній, *Figma* дозволяє працювати офлайн, зберігаючи локальні копії проектів. Це робить *Figma* високопродуктивним інструментом для дизайну та розробки, зокрема, для великих та розподілених команд.

*3) Adobe XD*

*Adobe XD* [17] (рис 2.6) видається надзвичайно потужним інструментом для дизайну та прототипування, вирізняючись своєю інтеграцією в екосистему *Adobe Creative Cloud*. Зручний інтерфейс та відмінні інструменти для векторного дизайну дозволяють користувачам легко створювати інтерфейси та прототипи.

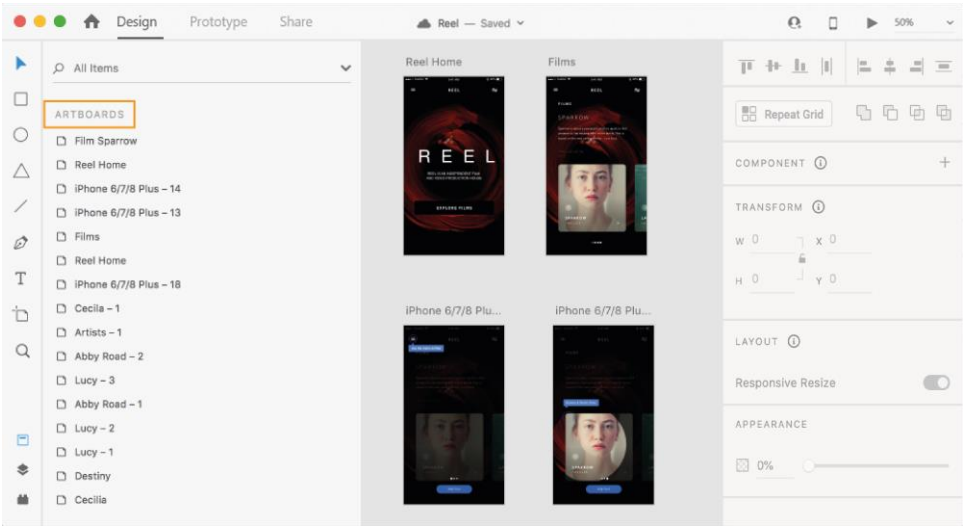


Рис 2.6. Інтерфейс *Adobe XD*

Однією з вагомих переваг *Adobe XD* є його ефективна інтеграція з іншими продуктами *Creative Cloud*, такими як *Photoshop* та *Illustrator*. Це спрощує процес обробки графічних елементів та імпорту ресурсів. Великий вибір шаблонів та ресурсів також полегшує старт проектів.

*Adobe XD* володіє потужними інструментами для створення інтерактивних прототипів та переходів між екранами. Його можливості автозаповнення та переходів дозволяють швидко втілювати ідеї в життя. Однак, підписка на *Adobe Creative Cloud* може бути дорога для окремих користувачів та менших команд.

У порівнянні з іншими інструментами, *Adobe XD* може виглядати більш традиційною альтернативою, але його впроваджені можливості та розширена інтеграція в сімейство *Adobe* роблять його привабливим для тих, хто вже використовує інші продукти *Adobe*.

### 2.4.2. Обґрунтування вибору інструменту Figma для дизайну

*Figma* виходить за межі звичайного інструменту для дизайну, надаючи неперевершені можливості для спільної роботи та розробки інтерфейсів. Його гнучкість та доступність, навіть у веб-браузері, роблять його ідеальним вибором для команд, які працюють над проектами з будь-якого місця світу. Інтерфейс *Figma* інтуїтивно зрозумілий та дружній до користувача, що полегшує як початківцям, так і досвідченим дизайнерам швидко освоїти всі його можливості.

Найбільш значущими перевагами *Figma* є безкоштовність та можливість спільної роботи в режимі реального часу. В перспективі, створений додаток може потребувати змін в дизайні за допомогою команди кваліфікованих дизайнерів. Здійснюючи зміни або вносячи коментарі, команди можуть співпрацювати онлайн, без необхідності відправки файлів чи використання додаткових інструментів. Це дозволяє покращити комунікацію, виправити недорозуміння та зменшити час, витрачений на обговорення дизайну.

Забезпечуючи величезну бібліотеку готових елементів, *Figma* дозволяє швидко створювати та тестувати дизайн-концепції. Його система компонентів і стилізації сприяє однорідності та ефективності у розробці. Ще однією перевагою є можливість перегляду прототипів прямо в браузері без додаткових програм, що спрощує зворотний зв'язок від клієнтів або команди.

Враховуючи зазначені переваги, *Figma* видається найбільш оптимальним вибором для дизайну в сучасному професійному середовищі, надаючи надійність, продуктивність та зручність в роботі над проектами будь-якого масштабу.

# РОЗДІЛ 3. ПРОЕКТНІ РІШЕННЯ

# ОПИС ПРОЦЕСУ РОЗРОБКИ ФРАГМЕНТУ ЗАСТОСУНКУ

## 3.1. ТЕХНІЧНЕ ЗАВДАННЯ.

### 3.1.1. НАЙМЕНУВАННЯ І ОБЛАСТЬ ЗАСТОСУВАННЯ

Інформаційна технологія пошуку ЗВО для навчання на основі платформи Firebase.

### 3.1.2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Завдання на виконання кваліфікаційної роботи за наказом ректора НТУ «Дніпровська політехніка» від від 9.10.2023р. № 1227-с.

### 3.1.3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Інформаційна технологія, призначена для спрощення вибору вищого навчального закладу.

### 3.1.4. ВИМОГИ ДО ФУНКЦІОНАЛЬНИХ ХАРАКТЕРИСТИК

Функціональні характеристики застосунку користувача та адміністратора повинні забезпечувати:

• Карта з відображенням закладів вищої освіти у виді маркерів

• Інтерактивність карти та маркерів

• Пошук вишів на головному екрані за назвою, адресою, містом та областю

• Попередній перегляд обраного навчального закладу

• Екран списку вишів

• Пошук вишів на екрані списку вишів за назвою, адресою, містом та областю

• Можливість перегляду детальної інформації про навчальний заклад

• Можливість відкрити обраний заклад в GoogleMaps з додатка

• Екран з обраними університетами

• Можливість сортувати університети за рейтингом, оцінкою, кількістю студентів та роком заснування за спаданням та зростанням на екранах зі списком університетів

• Можливість додавати чи прибирати університет до списку обраних

• Можливість редагувати інформацію про університет чи додавати новий у список в застосунку адміністратора

### 3.1.5. ВИМОГИ ДО НАДІЙНОСТІ

Надійність і безпека застосунку залежить від:

• Справної роботи застосунку

• Справної роботи операційної системи

• Наявності вільного місця в сховищі смартфона

• Стабільного зв’язку з інтернетом

### 3.1.6. ВИМОГИ ДО СКЛАДУ І ПАРАМЕТРІВ ТЕХНІЧНИХ ЗАСОБІВ

Для функціонування додатку потрібно:

• процесор з частотою 1 ГГц і вище;

• не менше 1 Гб оперативної пам’яті;

• мінімум 30 МБ вільного місця в сховищі смартфона

### 3.1.7. ВИМОГИ ДО ІНФОРМАЦІЙНОЇ ТА ПРОГРАМНОЇ СУМІСНОСТІ

Застосунок призначений для роботи на операційній системі Android. Може також бути запущений на операційній системі iOS.

### 3.1.8. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Для розробленого застосунку повинна бути представлена така технічна документація:

• перелік використаних плагінів;

• програмний код;

• опис проекту.

### 3.2. Створення дизайну

#### 3.2.1. Дизайн додатку клієнта

Створення дизайну додатку починається зі створення *user flow*. *User flow* [18], або потік користувача, представляє модель, яка відображає послідовність етапів, які користувач пройде при взаємодії з веб-сайтом або мобільним додатком.

Створений в роботі бакалавра додаток має потік користувача, зображений на рис 3.1.


Рис 3.1. User flow існуючого додатка

Перш ніж приступати до змін дизайну, потрібно переписати *user flow* під майбутні зміни. Так, використовуючи плагін *Arrow Auto* [19] (рис 3.2) в платформі *Figma* та його інструменти, було створено модель (рис 3.3), що відображає послідовність екранів та їх функціонал.

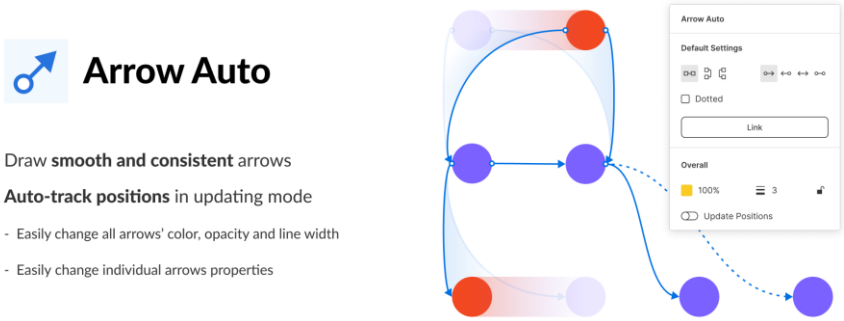


Рис 3.2. Плагін *Arrow Auto*

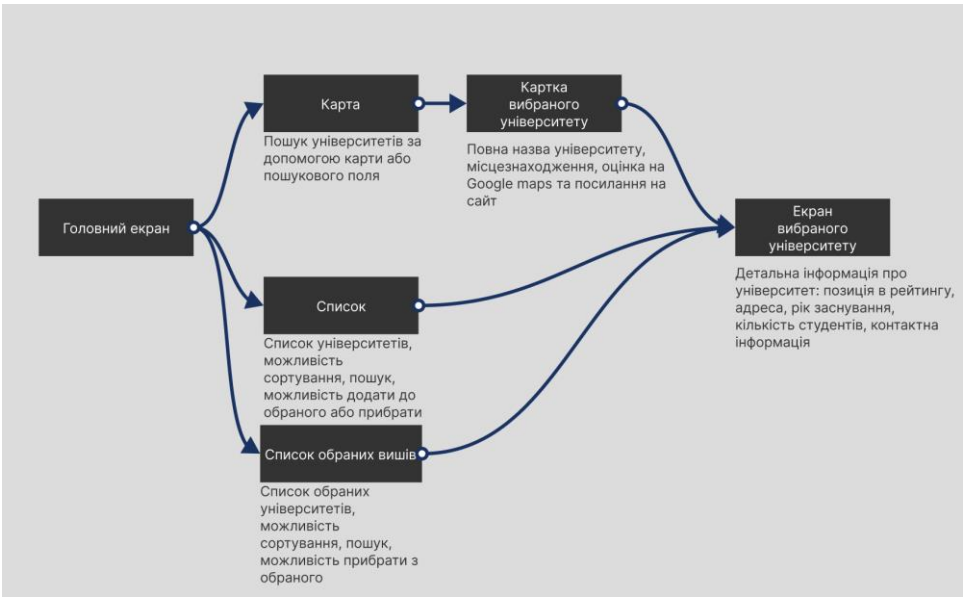


Рис 3.3. Змінений *user flow*

Після створення оновленого *user flow*, можна приступати до створення дизайну додатку.

Для інтерактивного елементу для додавання навчального закладу до списку обраних, було знайдено *svg*-картинку (рис 3.4) на сайті *Google Fonts*, який пропонує широкий вибір іконок в стилі *Material* [20]. В подальшому цю іконку було додано до дизайну головної сторінки додатка (рис 3.5).

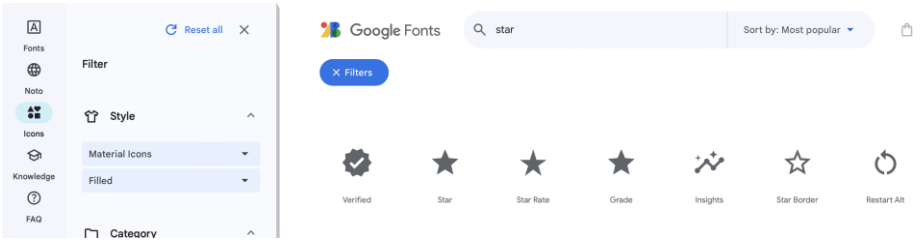


Рис 3.4. Пошук іконки для обраних університетів на сайті *Google Fonts*

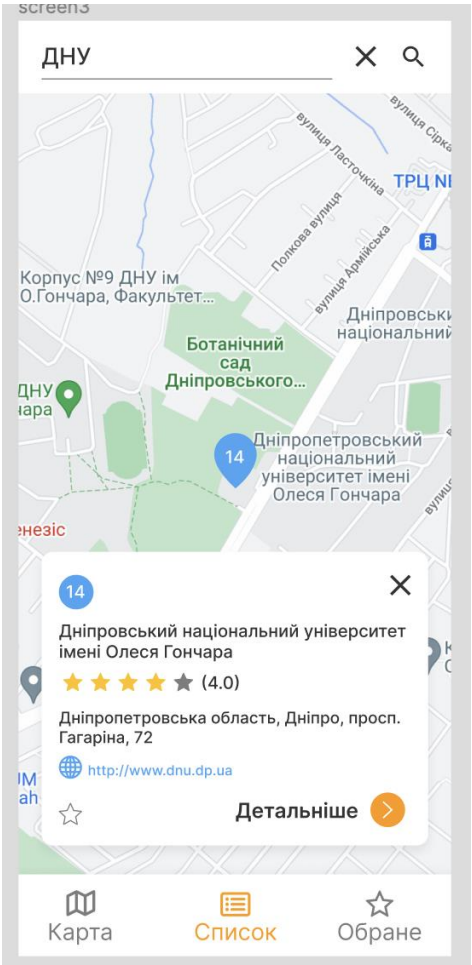


Рис 3.5. Оновлений дизайн головної сторінки

Цю іконку було додано також на екран списку університетів та на екран детальної інформації.

Далі, було додано панель сортування списку університетів (рис 3.6). При створенні дизайну цієї панелі, орієнтація також стояла на збереження стилю *Material*.

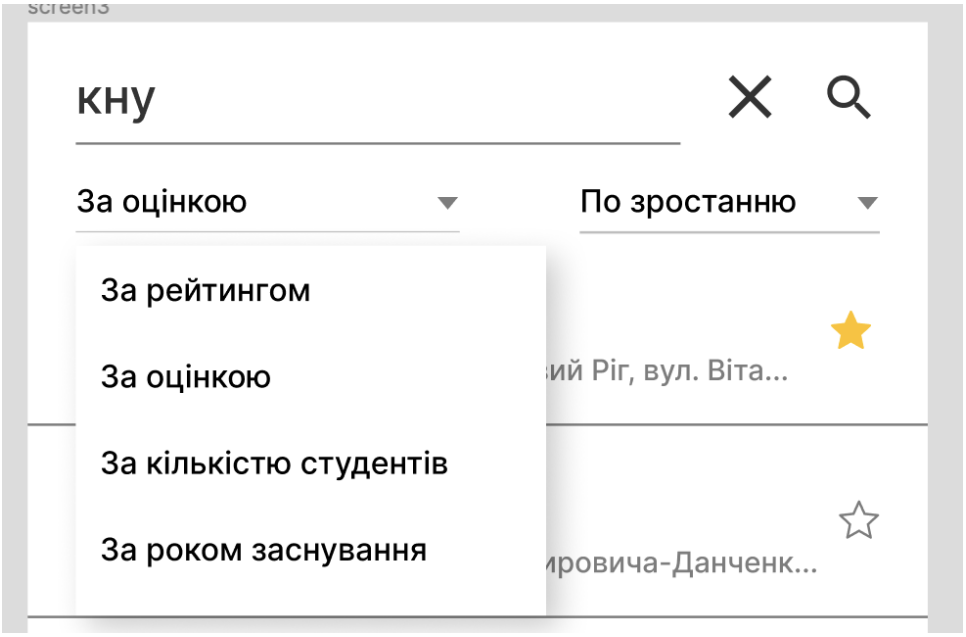


Рис 3.6. Вигляд панелі сортування на екрані списку університетів

Оскільки дизайн сторінки обраних університетів буде дублювати список університетів, то створювати його окреме вікно не було сенсу.

### 3.2.2. Дизайн додатку адміністратора

Слідуючим етапом було перейдено до створення екрану адміністративної панелі. Визначено, що екран повинен не відволікати адміністратора від головної цілі, тому панель (рис 3.7) було створено в мінімалістичному дизайні, використовуючи *Material* елементи. Внизу екрану було розташовано кнопку “зберегти”, щоб внести зміни.

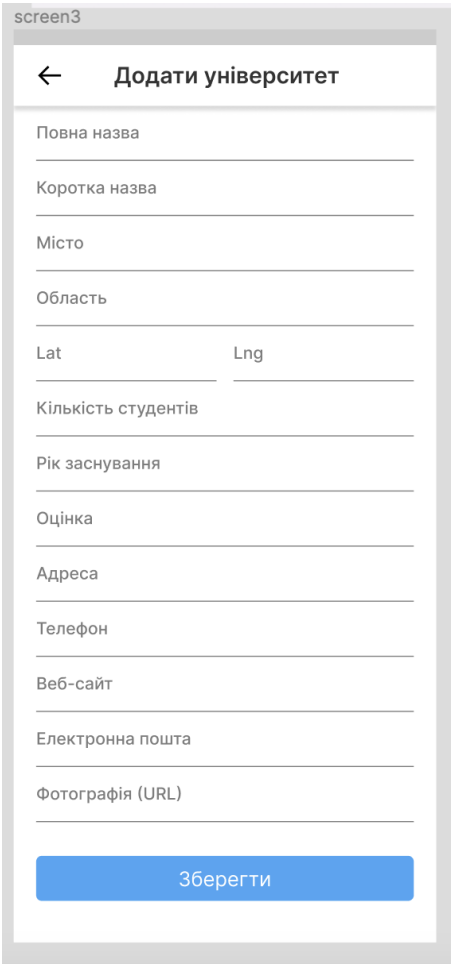


Рис 3.7. Дизайн панелі адміністратора

Щоб потрапити на екран адміністративної панелі, було додано кнопку на домашньому екрані (рис 3.8) та інтерактивний елемент на кожен елемент списку університетів.

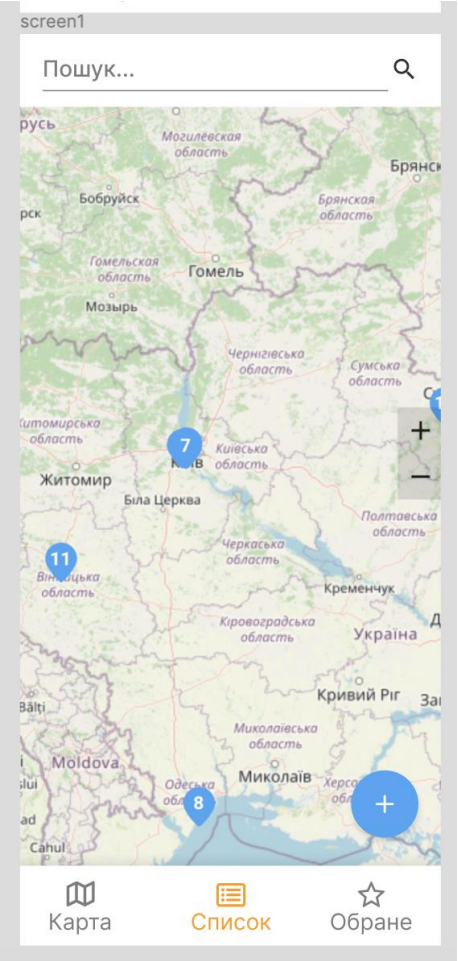


Рис 3.8. Кнопка на головному екрані для потрапляння на адміністративну панель

### 3.3. Розробка інформаційної технології

#### 3.3.1. Розробка додатку клієнта

Першим ділом стоїть задача додати логіку стягування інформації з *Firebase Firestore*. Звернувшись до документації на сайті платформи *Firebase*, було отримано інструкцію по імплементації та налаштуванню їх інструментів в проекті.

Для імплементації залежності від *Firebase Firestore*, потрібно звернутись до сайту *pub.dev* [21], який є основним сайтом для зберігання та пошуку бібліотек для фреймворку *Flutter* та мови програмування *Dart*. На даному сайті було знайдено бібліотеки *firebase_core* та *cloud_firestore* з інформаціїєю про останню версію пакету та інструкціями до використання.

Перелічені пакети разом з номером версії було додано до файлу pubspec.yaml, що визначає залежності проекту.

Далі, звернувшись до терміналу, було встановлено *FlutterFire* [22] та викликано рядок *flutterfire configure* для того, аби створити проект в файербейзі та зв'язати його з нашим *Flutter*-проектом. Після закінченя даної процедури було створено файл *firebase_options.dart*, роль якого полягає у зберіганні налаштувань для зв'язку з файербейзом. Цей файл в подальшому буде використаний для ініціалізації *Firebase* в проекті за допомогою виклику функції *Firebase.initializeApp()* в *main*-функції додатку, передаючи в неї клас *DefaultFirebaseOptions* з того самого файлу.

Щоб додати логіку стягування університетів з *Firestore,* було створено клас *FirestoreRepo*, в якому імпортовано бібліотеку *cloud_firestore* та прописано функцію звернення до бази даних (рис 3.9), заздалегідь створеною у консолі даної платформи (рис 3.10). База даних була заповнена моделями університетів зі збереженням назв полів, щоб при відповіді з сервера не виникло проблем з парсингом. Сам парсинг відбувається за допомогою функції *fromJson(Map<String, dynamic> json)*, всередині якої для моделі вишу встановлюються значення відповідно до ключів.

```
class FirestoreRepo {
  FirebaseFirestore firestore = FirebaseFirestore.instance;

  Future<List<University>> fetchAllUniversities() async {
    final col = firestore.collection('universities');
    final result = await col.get();
    final docs = result.docs;
    List<University> universities = [];
    for (final doc in docs) {
      universities.add(University.fromJson(doc.data()));
    }
    universities
        .sort((a, b) => a.rankingPosition!.compareTo(b.rankingPosition!));
    return universities;
  }
}
```

Рис 3.9. Створена функція по стягуванню університетів

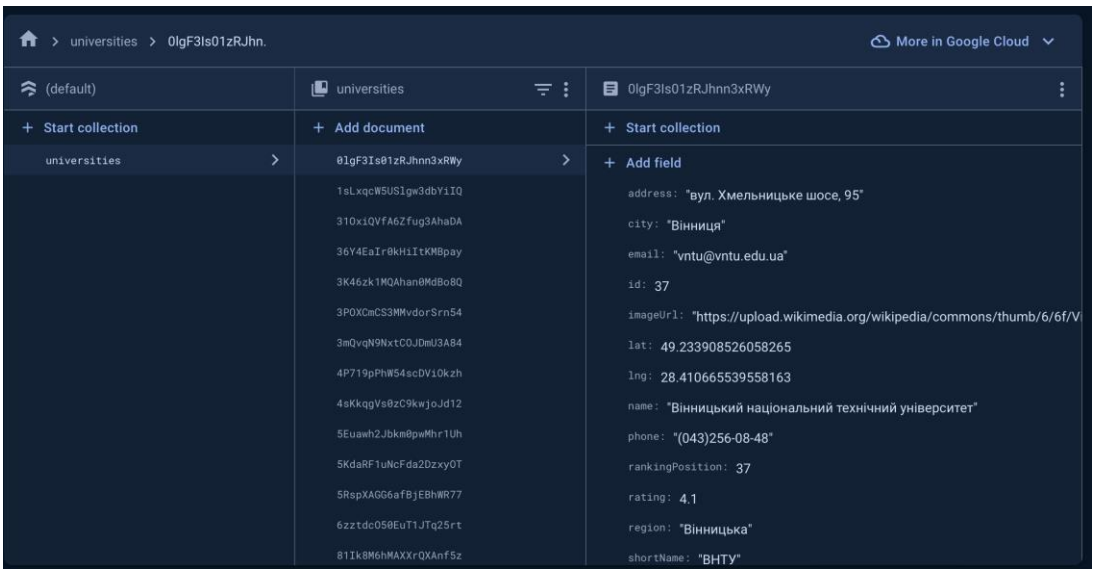


Рис 3.10. База даних у *Firebase Firestore*

Другим кроком почнемо зміни в додатку з панелі пошуку. Як вже було визначено, система пошуку потребує доповнень. На даний момент пошук ведеться по назві університету (рис 3.11). Планується розширити пошук по співпадінням з адрессою, містом та областю.

```
static bool isUniversityContainsValue(University university, String val) {
  var allUniverSearchFields =
      '${university.name} ${university.shortName}'
          .toLowerCase();
  return allUniverSearchFields.contains(val.toLowerCase());
}
```

Рис 3.11. Функція по пошуку університетів за співпадіннями до змін

Для вдосконалення роботи цієї функції було додано поля *region*, *address*, *city* (рис 3.12).

```
static bool isUniversityContainsValue(University university, String val) {
  var allUniverSearchFields =
      '${university.address} ${university.city} ${university.name} ${university.region} ${university.shortName}'
          .toLowerCase();
  return allUniverSearchFields.contains(val.toLowerCase());
}
```

Рис 3.12. Функція по пошуку університетів за співпадіннями змін

Слідуючим кроком є створення функції додавання університетів до списку обраних. Оскільки в додатку відсутня система створення аккаунту користувача, а додавати її, спираючись на існуючий функціонал, поки нема сенсу, було вирішено зробити збереження обраних вишів до локального сховища. Для цього було обрано бібліотеку *Hive*, що являє собою *NoSQL* базу даних.

Найпопулярнішими інструментами для роботи з локальним сховищем у фреймворку *Flutter* є *Shared Preferences*, *Hive* та *Sqflite*. *NoSQL* було обрано в першу чергу через його простоту у використанні та швидкість роботи. Відповідно до статті [23], написної *Bacancy Technology*, та їх графіку (рис 3.13), *Hive* виграє по швидкості записів та зчитувань з величезним відривом.

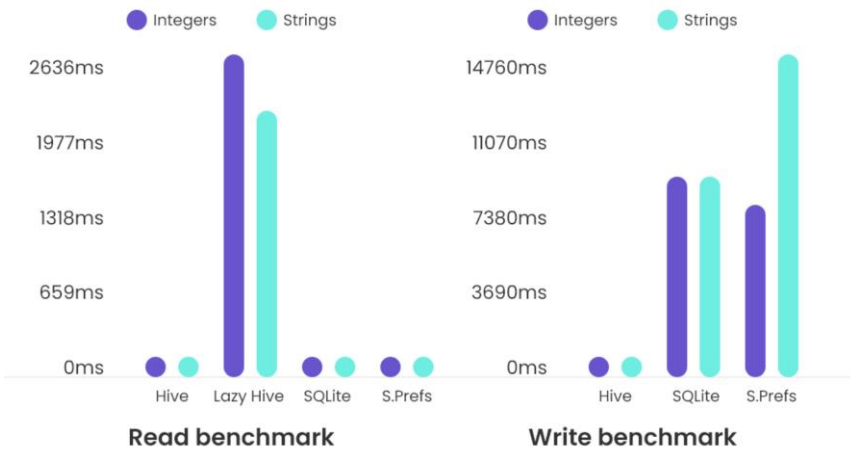


Рис 3.13. Заміри швидкості роботи бібліотек для роботи з локальним сховищем

Робота з функцією по збереженню улюблених вишів почалась з імплементації бібліотеки *Hive* у файл *pubspec.yaml*. Далі, *Hive* було проініціалійовано в *main*-функції додатку за допомогою *Hive.initFlutter()*.

Після цього було створено клас *HiveUtil* для структуризації та відокремлення логіки з локальним сховищем. В цьому класі було додано константний ключ до сховища і прописані функції (рис 3.14) по стягуванню, збереженню та видаленню улюблених університетів. Запис вишів відбувається по ідентифікатору виша. Таким чином, при стягувані списку

вишів, ми отримаємо список ідентифікаторів, по яким далі будуть шукатись співпадіння.

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

class HiveUtil {
  static const _favUniversitiesKey = 'favUniversities';

  static Future<void> setFavouriteUniversity(int id) async {
    final box = await Hive.openBox<int>(_favUniversitiesKey);
    await box.add(id);
  }

  static Future<void> removeFavouriteUniversity(int id) async {
    final box = await Hive.openBox<int>(_favUniversitiesKey);
    final universities = await getFavouriteUniversities();
    universities.removeWhere((element) => element == id);
    await box.clear();
    await box.addAll(universities);
  }

  static Future<List<int>> getFavouriteUniversities() async {
    final box = await Hive.openBox<int>(_favUniversitiesKey);
    return box.values.toList();
  }
}
```

Рис 3.14. Клас *HiveUtil* з функціями роботи з обраними вишами

Після створення даних функцій, було додано відповідні інтерактивні елементи (рис 3.15-3.17) на екранах списку вишів, мапі та екрані детальної інформації про університет. До інтерактивних елементів була підв’язана логіка з роботою по збереженню та видаленню.


Рис 3.15. Новий інтерактивний елемент на екрані детальної інформації про університет

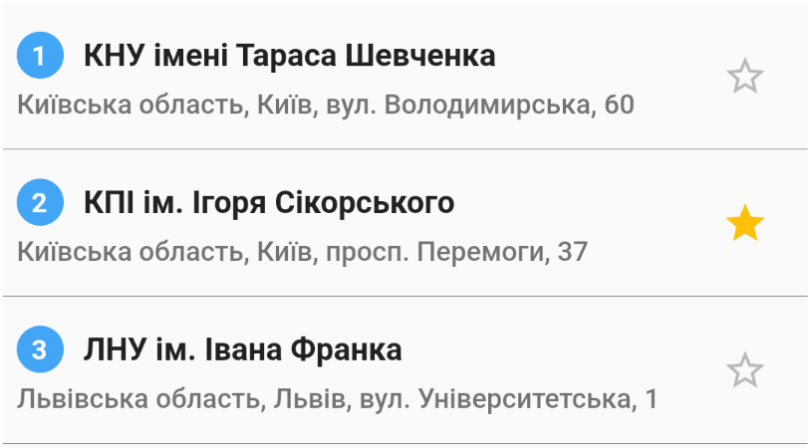


Рис 3.16. Новий інтерактивний елемент на екрані списку університетів

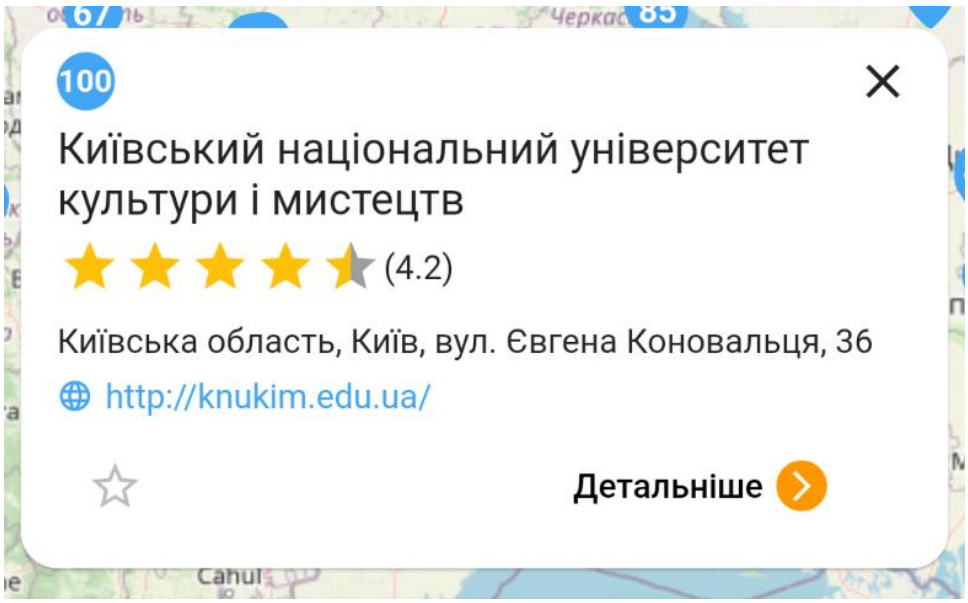


Рис 3.17. Новий інтерактивний елемент на екрані мапи університетів

Слідуючим етапом по роботі зі списком улюблених вишів було створено сам екран улюблених вишів. Цей екран було створено на базі списку університетів, але в ньому була змінена логіка стягування самого списку, орієнтуючись тільки на ті навчальні заклади, що були збережені до обраного.

Для цього, в конструктор класу *UniversitiesListPage* було додано змінну *favourites*. При значені true, даний клас починає себе вести як список обраних вишів (рис 3.18).

```dart
Future<void> _initList() async {
  final favIdList = await HiveUtil.getFavouriteUniversities();
  if (widget.favourites) {
    List<University> allUniversities = List.from(widget.universities);
    universities = [];
    for (var fav in favIdList) {
      final uni = allUniversities.firstWhereOrNull((e) => e.id == fav);
      uni?.favourite = true;
      if (uni != null) universities.add(uni);
    }
    setState(() {});
  } else {
    universities = List.from(widget.universities);
    for (var fav in favIdList) {
      final uni = universities.firstWhereOrNull((e) => e.id == fav);
      uni?.favourite = true;
    }
    setState(() {});
  }
}
```

Рис 3.18. Функція по ініціалізації списку університетів класу *UniversitiesListPage*

Нижню навігаційну панель додатку було розширено, додавши до нього ще один клас *UniversitiesListPage*, але встановивши до *favourites* значення *true*. В результаті було створено нову сторінку з відповідним функціоналом.

Передостаннью частиною роботи з додатком для користувачів залишилась імплементація роботи по сортуванню університетів за рейтингом, роком заснування, кількістю університетів та оцінкою в *Google Maps*. Для цього було додано два випадаючих списки. Перший орієнтується на сортування за рейтингом, роком заснування, кількістю університетів та оцінкою. Другий сортує за зростанням чи спаданням обраних значень. При змінах значення будь-якого випадаючого списку, викликається функція *onSortValuesChanged* (рис 3.19), в якій за допомогою switch-case відбувається сортування відповідно до обраних значень. Функція сортування базується на методі *sort()* інтерфейсу *List*.

```
void _onSortValuesChanged() {
  switch (_sortValue) {
    case byGmapsRate:
      universities.sort((a, b) => (a.rating ?? 0).compareTo(b.rating ?? 0));
      break;
    case byRanking:
      universities.sort((a, b) =>
          (a.rankingPosition ?? 0).compareTo(b.rankingPosition ?? 0));
      break;
    case byStudents:
      universities.sort(
          (a, b) => (a.studentsCount ?? 0).compareTo(b.studentsCount ?? 0));
      break;
    case byYear:
      universities.sort((a, b) => (a.year ?? 0).compareTo(b.year ?? 0));
      break;
    default:
      break;
  }
  if (_orderValue == descending) {
    universities = List.from(universities.reversed);
  }
  setState(() {});
}
```

Рис 3.19. Функція сортування університетів

Створена функція одразу буде працювати і на екрані списку обраних навчальних закладів.

Останнім етапом є впровадження в додаток інструменту для відстежування фатальних помилок та поведінки користувача. Для цього відповідно було використано *Firebase Crashlytics* та *Firebase Analytics*. До *pubspec.yaml* додано бібліотеки *firebase_crashlytics* та *firebase_analytics*, а в main функції додано частину коду (рис 3.20), що проводить ініціалізацію відстежування крашів.

```
  FlutterError.onError = (errorDetails) {
    FirebaseCrashlytics.instance.recordFlutterError(errorDetails);
  };
  PlatformDispatcher.instance.onError = (error, stack) {
    FirebaseCrashlytics.instance.recordError(error, stack);
    return true;
  };
```

Рис 3.20. Функція відстежування крашів в додатку

### 3.3.2. Розробка додатку адміністратора

Як вже було зазначено, для додатку адміністратора буде використано ту саму кодову базу, що і в додатку для користувача, але з новою сторінкою для додавання навчальних закладів у список та їх редагування.

Для цього було створено клас *AddUniversityPage*, що буде приймати в конструкторі модель університету, що може бути нульовою. При нульовому значенні університета сторінка буде відпрацьовувати логіку додавання нового елементу у список, а при ненульовому – редагування.

Відповідно до дизайну, екран складається з набору редагуємих текстових полей. Для взаємодії з ними було прописано список контролерів *TextEditingController* для кожного поля. Даний контролер дозволяє витягувати значення поля, встановлювати нове та реагувати на зміни в полі.

Внизу екрану розташована кнопка “Зберегти”, при натисканні на яку викликається функція, що першочергово проходиться по контролерам в пошуку незаповнених полів, аби запобігти помилок при вставленні документа в базу даних. Якщо функція знаходить незаповнені поля, адміністратору видасть діалогове вікно з повідомленням, що не всі поля заповнені (рис 3.21).

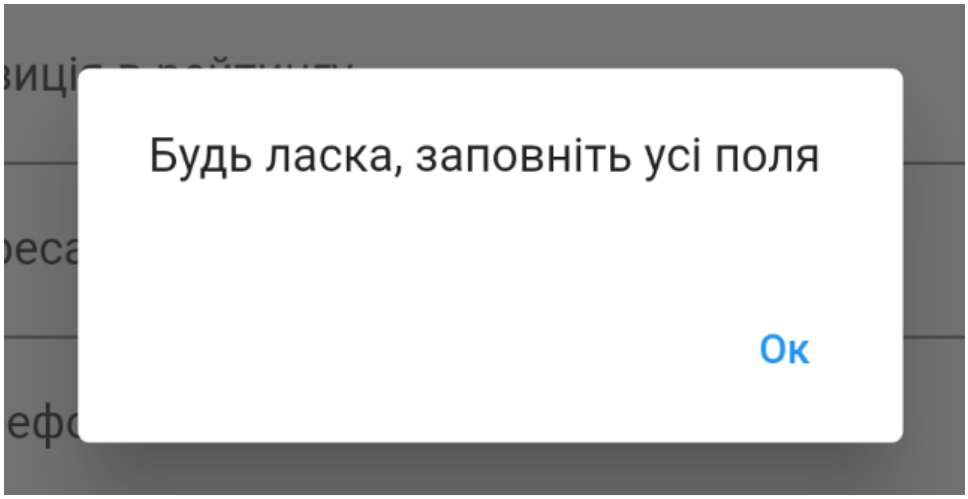


Рис 3.21. Вигляд повідомлення про незаповнені поля

При успішно заповнених полях, додаток звертається до файербейзу (рис 3.22), передаючи створену або змінену модель університету, викликаючи відповідні функції *add()* або *set()*, що на вхід приймають модель ключ-значення, тобто *Map*.

Щоб додати можливість потрапити на сторінку додавання інформації про навчальний заклад, на домашньому екрані було створено плаваючу кнопку *FloatingActionButton*, яка веде на створення університету, а на екрані списку вишів було додано інтерактивний елемент до кожного елементу, що веде на екран редагування інформації з вже з заповненими полями.

```
FirebaseFirestore firestore = FirebaseFirestore.instance;

try {
  final col = firestore.collection('universities');

  if (widget.university != null) {
    await col.doc(university.docId).set(university.toJson());
    if (mounted) Navigator.pop(context);

    return;
  }
  await col.add(university.toJson());
} catch (e) {
  if (!mounted) return;
  showDialog(
      context: context,
      builder: (c) => AlertDialog(
            content: const Text('Щось пішло не так'),
            actions: [
              TextButton(
                  onPressed: () {
                    Navigator.pop(context);
                  },
                  child: const Text('Ок')) // TextButton
            ],
          )); // AlertDialog
  return;
}
if (!mounted) return;
Navigator.pop(context);
```

Рис 3.22. Частина коду по роботі з оновленням списку університетів

Як видно з коду на рисунку, роботу з додаванням вишу до списку через файербейз було обгорнуто в *try-catch*. Зроблено це було з ціллю зловити

критичні помилки та запобігти неправильному перебігу роботи додатка. При виявленні таких помилок, адміністратору покаже діалогове вікно з текстом, що щось пішло не так.

### 3.3.3. Доцільність та ефективність створеної інформаційної технології

Розроблений додаток спрощує процес пошуку вищих навчальних закладів для абітурієнтів, надаючи швидкий доступ до необхідної інформації всього за кілька кліків. Замість перегляду списків університетів в Інтернеті, користувачеві достатньо відкрити додаток. Оскільки в Україні відсутні подібні застосунки, розроблений додаток є першим у своєму роді, що спростить процес вибору вищого навчального закладу, об'єднавши необхідний функціонал в одному місці.

Створена система відповідає стандартам інформаційної технології, оскільки використовує різноманітні технічні рішення для збору, зберігання, обробки та передачі даних, спрямованих на вдосконалення функціональності та взаємодії користувача з додатком. При розширенні чи зміних бази даних додаток не залежний від оновлень, а адміністративна панель запроваджує легкий і швидкий доступ до редагування інформації.

# ВИСНОВОК

Мета кваліфікаційної роботи полягала у створенні застосунку для пошуку ЗВО для навчання на основі платформи Firebase, що відповідає стандартам інформаційної технології.

У ході виконання даної атестаційної роботи було проведено глибокий аналіз інформаційної технології та розглянуто існуючий додаток для огляду вищих навчальних закладів. Аналіз сприяв виявленню необхідності у вдосконаленнях, зокрема у переході на серверні рішення та розширенні функціоналу.

При виконанні кваліфікаційної роботи, було зроблено слідуючі кроки:

- проаналізовано роботу існуючого додатку;
- проведено огляд платформи Firebase та його конкурентів;
- визначенно необхідні покращення пошукової системи додатку;
- розглянуто інструменти для створення дизайну;
- спроектовано панель адміністратора;
- створено хмарну базу даних;
- оновлено додаток користувача;
- розроблено та протестовано програмний код додатку адміністратора.

В кваліфікаційній роботі визначено важливість створення стійкої інформаційної технології в умовах сучасного світу.

Проектування панелі адміністратора та дизайн продукту виконано з дотриманням вимог ергономіки та естетики. Вибір інструменту для дизайну, зокрема Figma, також обґрунтовано в контексті зручності та спільної роботи.

Дана атестаційна робота успішно вирішує поставлені завдання та вносить вагомий внесок у розвиток інформаційних технологій, створюючи додаток, який спростить процес отримання інформації про вищі навчальні заклади в Україні. Застосунок написаний за допомогою фреймворку Flutter,

що дозволяє випустити його одразу на ОС Андроїд та iOS. Завдяки перенесенню роботи з даними на сервер та створенню адміністративного додатку, створена система більше не залежить від оновлень додатку та відповідає стандартам інформаційної технології.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Що таке інформаційні технології [Електронний ресурс] – Доступ до ресурсу: http://apeps.kpi.ua/shcho-take-informatsiini-technologii/en.
2. Перший смартфон в історії мобільних телефонів [Електронний ресурс] – Доступ до ресурсу: https://futurenow.com.ua/pershyj-smartfon-u-sviti-telefon-simon-na-15-rokiv-starshyj-za-iphone/.
3. Найкращі практики для створення домашнього екрану [Електронний ресурс] – Доступ до ресурсу: https://support.jmango360.com/portal/en/kb/articles/best-practices-to-design-home-screen.
4. Історія Firebase [Електронний ресурс] – Доступ до ресурсу: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0.
5. Сайт Firebase [Електронний ресурс] – Доступ до ресурсу: https://firebase.google.com/.
6. Компанії, що використовують Firebase [Електронний ресурс] – Доступ до ресурсу: https://blog.back4app.com/which-companies-use-firebase/.
7. Що таке MVP [Електронний ресурс] – Доступ до ресурсу: https://www.productplan.com/glossary/minimum-viable-product/.
8. Що таке Firebase firestore [Електронний ресурс] – Доступ до ресурсу: https://cloud.google.com/firestore/docs#:~:text=Firestore%20is%20a%20NoSQL%20serverless,business%20logic%20and%20user%20experience.
9. Сайт Amazon Dynamo DB [Електронний ресурс] – Доступ до ресурсу: https://aws.amazon.com/dynamodb/
10. Що таке MongoDB Atlas [Електронний ресурс] – Доступ до ресурсу: https://www.mongodb.com/basics/mongodb-atlas-tutorial#:~:text=MongoDB%20Atlas%20is%20a%20fully,scale%20MongoDB%20in%20the%20cloud.

11. Що таке Microsoft Azure Cosmos DB [Електронний ресурс] – Доступ до ресурсу:https://azure.microsoft.com/en-us/products/cosmos-db.

12. Проектування панелі адміністратора [Електронний ресурс] – Доступ до ресурсу: https://avada-media.ua/ua/services/ux-admin/

13. Про Flutter [Електронний ресурс] – Доступ до ресурсу: https://flutter.dev/

14. Про Dart [Електронний ресурс] – Доступ до ресурсу: https://dart.dev/

15. Що таке Sketch [Електронний ресурс] – Доступ до ресурсу: https://aws.amazon.com/dynamodb/

16. Що таке Figma [Електронний ресурс] – Доступ до ресурсу: https://wezom.academy/ua/chto-takoe-figma-funktsii-instrumenty-ipreimuschestva/

17. Чим Adobe XD приваблює дизайнерів [Електронний ресурс] – Доступ до ресурсу: https://web4u.in.ua/blog/chim-adobe-xd-tak-privablyu-dizayner-v-33

18. Що таке user flow [Електронний ресурс] – Доступ до ресурсу: https://www.productplan.com/glossary/user-flow/#:~:text=User%20flows%20create%20a%20visual,page%20belongs%20where%20it%20is.

19. Плагін Arrow Auto для Figma [Електронний ресурс] – Доступ до ресурсу: https://www.figma.com/community/plugin/751007211632768205

20. Що таке Material Design [Електронний ресурс] – Доступ до ресурсу: https://en.wikipedia.org/wiki/Material_Design

21. Що таке pub.dev [Електронний ресурс] – Доступ до ресурсу: https://stackshare.io/pub-dev#:~:text=It%20is%20the%20package%20manager,AngularDart%2C%20and%20general%20Dart%20programs.

22. Документація FlutterFire [Електронний ресурс] – Доступ до ресурсу: https://firebase.flutter.dev/

23. Порівняння Hive з його конкурентами [Електронний ресурс] – Доступ до ресурсу: https://www.bacancytechnology.com/blog/handle-offline-data-storage-with-flutter-hive