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)

студентки *Гордієнко Анастасії Юріївни*
(ПІБ)

академічної групи *121-17-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Платформа для генерації і проходження навчальних тестів онлайн. Розробка клієнтської частини.*

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

Дніпро
2020

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

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

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2020 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студентки *121-17-1* *Гордієнко Анастасії Юріївни*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Платформа для генерації та проходження навчальних тестів онлайн. Розробка клієнтської частини.*

затверджена наказом ректора НТУ «ДП» від №

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	15.05.2021
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПО й тривалості його розробки***	28.05.2021

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

Завдання прийняв до виконання _____ *Гордієнко А.Ю.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: .

Термін подання кваліфікаційної роботи до ЕК: .

РЕФЕРАТ

Кваліфікаційна робота включає у себе 76 с., 49 рисунків, 10 таблиць, 3 додатка і 32 літературних джерела.

Об'єктом розробки є клієнтська частина платформи для генерації і проходження навчальних тестів онлайн.

Мета кваліфікаційної роботи: є оптимізація навчання шляхом обробки, аналізу та зберігання результатів, отриманих під час тестування певної групи осіб.

У першому розділі приведені загальні відомості з предметної галузі, призначення розробки та галузь застосування, підстави для розробки, огляд існуючих рішень, аналіз та усунення їх недоліків, постановка завдання, обґрунтування вибору мови програмування, вимоги до інформаційної системи, до функціональних характеристик, інформаційної безпеки, складу параметрів технічних засобів, інформаційної та програмної сумісності.

У другому розділі подано інформацію щодо опису структури системи та алгоритмів її функціонування, схеми і опису бази даних, опису інтерфейсу системи, збереження даних, мов програмування, ілюстрацій, поділу доступу, ергономіки та технічної естетики, інформації для користувачів, подання головної сторінки сайту, графічної оболонки внутрішніх сторінок.

В економічному розділі визначено трудомісткість розробленої системи, проведений підрахунок вартості роботи по створенню системи та розраховано час на її створення.

Практичне значення полягає у створенні інформаційної системи, що надає можливість автоматизувати процес скаладння, обробку та аналіз результатів проходження тестів з метою покращення якості освіти. Актуальність продукту, що розробляється зумовлена відсутністю аналогічних рішень і великими витратами людських ресурсів для досягнення потрібної цілі.

Перелік ключових слів: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, JAVASCRIPT, ОСВІТА, REACT.

ABSTRACT

Qualification work includes 79 pages, 49 figures, 10 tables, 3 appendices and 32 references.

The object of development is the client part of the platform for generating and passing online training tests.

The purpose of the qualification work: is to optimize learning by processing, analyzing and storing the results obtained during testing of a certain group of people.

The first section provides general information on the subject area, purpose of development and scope, grounds for development, review of existing solutions, analysis and elimination of their shortcomings, problem statement, justification of the choice of programming language, information system requirements, functional characteristics, information security, composition of parameters of technical means, information and software compatibility.

The second section provides information on the description of the system structure and algorithms of its operation, scheme and description of the database, description of the system interface, data storage, programming languages, illustrations, access sharing, ergonomics and technical aesthetics, information for users, shell of internal pages.

In the economic section, the complexity of the developed system is determined, the cost of work on the creation of the system is calculated and the time for its creation is calculated.

The practical significance lies in the creation of an information system that provides the ability to automate the process of composing, processing and analysis of test results in order to improve the quality of education. The relevance of the developed product is due to the lack of similar solutions and high cost of human resources to achieve the desired goal.

List of keywords: SOFTWARE, JAVASCRIPT, EDUCATION, REACT.

ЗМІСТ

ВСТУП

З розвитком комп'ютерних технологій з'явилися нові методики організації та проведення тестувань. Так, для проходження та створення тестів можна використовувати онлайн платформи в Інтернеті. Інформаційні технології забезпечують ефективну обробку даних, отриманих в результаті наукових досліджень або на практичних занятях. З цією метою широкого поширення набули електронні таблиці, реляційні бази даних, пакети програм математичної і статистичної обробки та аналізу даних. Для цього можна застосовувати технологію електронної пошти для розсилки завдань.

Слід також вважати, що значне використання паперу і часу роботи з ним також виправдовує використання комп’ютерних технологій, - аби зберегти час, сили і ресурси. Одним із таких проблемних питань є використання , у даному випадку для покращення якості освіти вищих навчальних закладів у вигляді оналйн тестувань.

Необхідністю створення вебдодатку зумовлена автоматизацією процесу тестування та обробки результатів навчання. Значне використання паперу і часу роботи виправдовує використання розробляємого додатку – збереження часу, сил і ресурсів.

Ціль розробки - вебдодаток для генерації та проходження навчальних онлайн тестів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальні відомості з предметної галузі

Об'єктом розробки є навчальний вебдодаток для генерації та проходження онлайн тестів.

Методи:

— аналіз предметної галузі;

— виділення інформаційних артефактів;

— нормалізація даних;

— аналіз атрибутів якості ПЗ.

Аналізуючи обрану область було виявлено, що більшість наявних платформ надають можливість створення онлайн тестів та поверхневий розгляд отриманих результатів. Однак не фокусуються на оптимізації навчального процесу, шляхом надання структурованого та повного аналізу відповідей (рис. 1.1, рис 1.2, рис. 1.3).

Розглянемо приклади наявних платформ для створення онлайн тестів:

Приклад 1: Google Forms

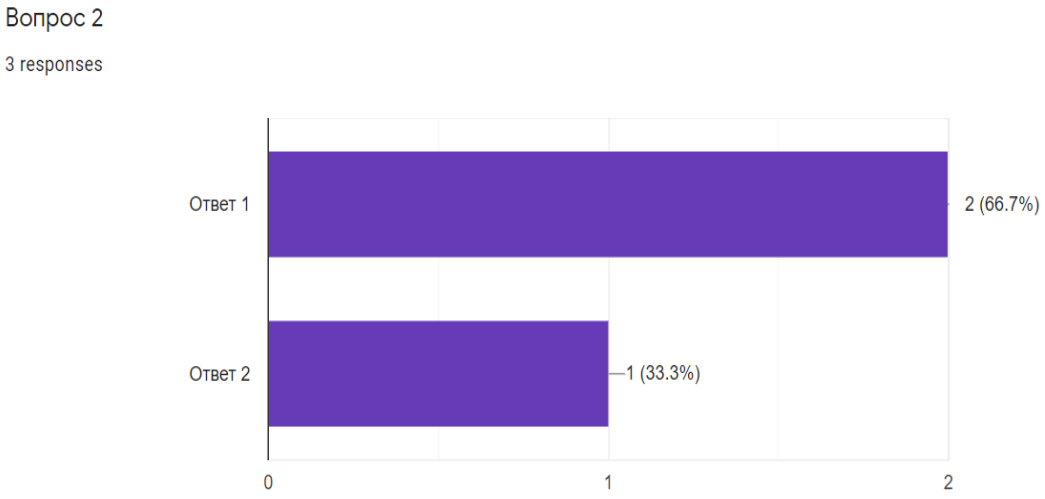

Рис. 1.1. Фрагмент аналізу Google Forms

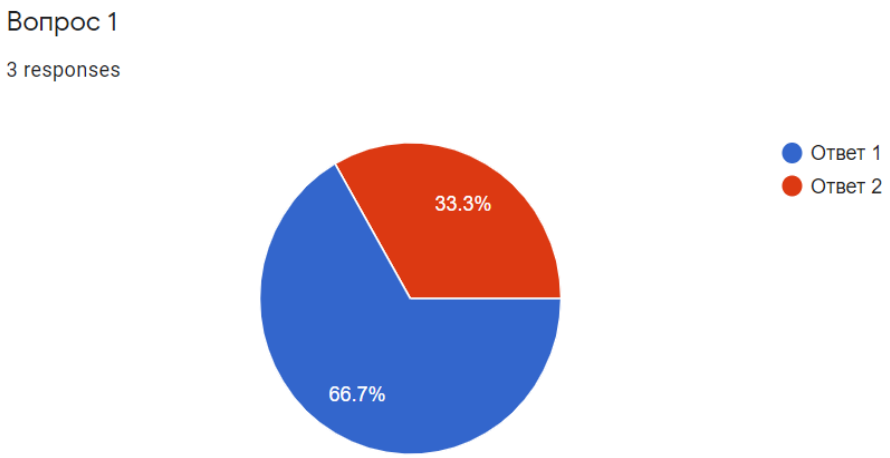

Рис. 1.2. Фрагмент аналізу Google Forms

Недоліки:

— Відсутня можливість групувати тести;

— Відсутня можливість фільтрувати результати;

— Відсутня можливість сортувати відповіді;

— Відсутня можливість формувати та зберігати групу респондентів для повторного використання;

— Аналіз даних обмежується двома діаграмами.

Приклад 2: Survey Monkey

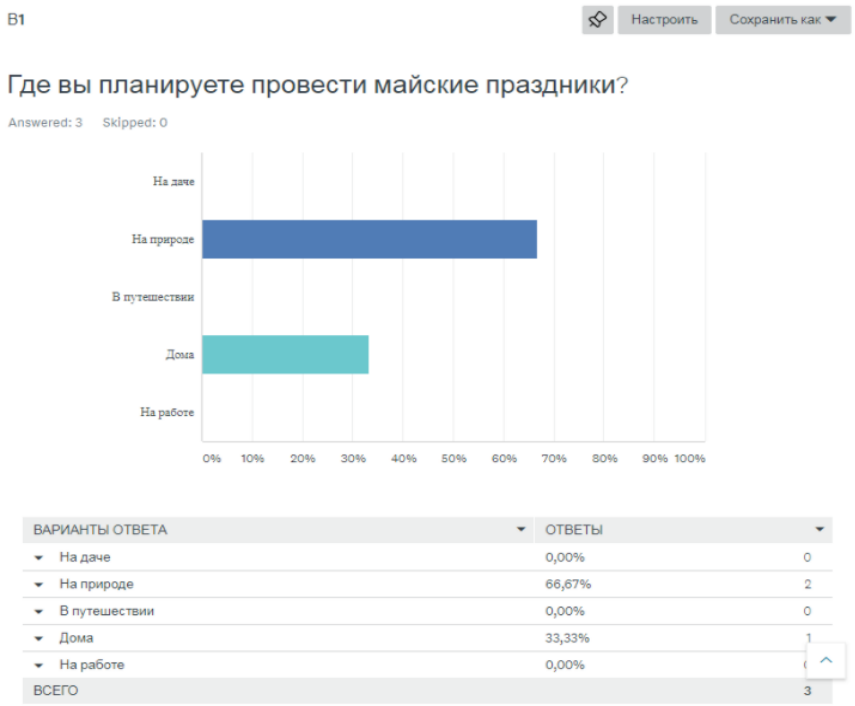

Рис. 1.3. Фрагмент аналізу Survey Monkey

Недоліки:

— Немає безкоштовної версії;

— Аналіз базується лише на відповідях;

— Відсутня можливість групувати тести;

— Відсутня можливість фільтрувати результати;

— Відсутня можливість сортувати відповіді;

— Відсутня можливість формувати та зберігати групу респондентів для повторного використання;

— Аналіз даних однією діаграмою та таблицею.

Висновок: більшість розглянутих платформ для генерації та проходження тестів спрямовані на узагальнений аналіз відповідей та тимчасове зберігання, що впливає на якість процесу оцінювання та оптимізацію навчального процесу. Не було знайдено вебдодатків для покращення якості освіти у відкритому доступі, який би містив:

— Можливість фільтрувати результати;

— Можливість сортувати відповіді (дата, тема);

— Можливість формувати та зберігати групу респондентів для повторного використання;

— Аналіз даних за різними показниками.

Підсумувавши все викладене вище, актуальність і унікальність розробленої системи була підтверджена.

1.2 Призначення розробки та галузь застосування

Ціль розробки - навчальний вебдодаток для генерації та проходження онлайн тестів.

Необхідністю створення вебдодатку зумовлена автоматизацією процесу тестування та обробки результатів навчання. Значне використання паперу і часу роботи виправдовує використання розробляємого додатку – збереження часу, сил і ресурсів.

Призначенням для розробки є оптимізація навчання шляхом обробки, аналізу та зберігання результатів, отриманих під час тестування певної групи осіб.

1.3 Підстава для розробки

Відповідно до освітньої програми, згідно з навчальним планом та графіком навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу. Тема роботи узгоджується з керівником роботи, випускаючою кафедрою, та затверджується наказом ректора.

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

а) освітня програма спеціальності 121 «Інженерія програмного забезпечення»;

б) навчальний план та графік навчального процесу;

в) наказ ректора Національного технічного університету «Дніпровська політехніка» № 275-с від 26.05.2021 р.;

г) завдання на кваліфікаційну роботу на тему: «Розробка навчального вебдодатоку для генерації та проходження онлайн тестів».

1.4 Постановка завдання

Метою кваліфікаційної роботи є розробка інформаційної системи, за допомогою якої можна забезпечити тестування осіб для подальшого аналізу та обробки результатів.

Розроблений додаток може використовуватись будь-яким закладом освіти.

Для досягнення поставленої мети, необхідно розробити механізм створення онлайн тестів, зберігання та обробка результатів, розробити інтерфейс системи. Для цього необхідно провести аналіз стану проблеми та розв'язувати наступні задачі:

– формування основних ідей і принципів оцінювання та створення завдань;

– формування архітектури додатка;

– проведення випробувань і оцінка ефективності розробленої структури.

Для практичної реалізації задачі необхідно створити інформаційну структуру з наступними функціональними можливостями:

1. Інформаційна система повинна створювати тести, забезпечувати доступ для їх проходження.

2. Система повинна мати зручний та інтуїтивно зрозумілий інтерфейс для користувача.

3. Результати аналізу повинні виводитися у інтерфейсі. Результатом є статистичні дані, представлені у вигляді таблиць, які доступні лише власникам тестів.

4. Вхідна інформація, яка передається користувачем в обробку повинна мати обмежену довжину (виходячи з обмеження на кількість символів в базі

даних, а також типу відповідного поля), контролювати раціональність та логіку значень, з якими буде оперувати система та запобігати її подвоєнню.

Передбачається, що на основі отриманих статистичних даних вчителі та зацікавлені особи матимуть змогу швидше аналізувати та обробляти отримані результати, що покращить та пришвидшить освітній процес.

1.5 Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Розробляючи інтерфейс системи, необхідно враховувати, що ним можуть користуватися люди, які мають різний досвід роботи з технологіями, тому платформа для онлайн тестів повинна бути надзвичайно простою та наглядною.

Інтерфейс повинен бути пристосований до основних норм створення веб-додатків, мати адаптивну верстку. Дані вводяться шляхом заповнення HTML – форм. У разі помилок зі сторони користувача, відповідні помилки виводяться у інтерфейсі.

Має бути реалізовано:

а) вікно логіна;

б) вікно реєстрації;

в) поділ користувачів:

1. Викладач
2. Студент
3. Всі користувачі

г) для кожної групи користувачів власний варіант інтерфейсу особистої сторінки;

д) можливість пройти та скласти тест;

е) сторінка редагування тесту;

ж) списки тестів;

з) сторінка власної інформації;

и) фільтрація даних за датою;

к) пошук за назвою тесту.

1.5.2 Вимоги до інформаційної безпеки

Щоб уникнути некоректної роботи платформи необхідно реалізувати:

– контроль вхідних даних;збереження даних та їх цілісності;

– обробку виключних ситуацій;

– вивід повідомлень про помилки.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для нормального функціонування системи необхідно, щоб обчислювальна машина, на якій буде функціонувати система, відповідала наступним вимогам:

— процесор з тактовою частотою не нижче 1,8 ГГц. Рекомендується використовувати як мінімум двоядерний процесор;

— 2 ГБ оперативної пам'яті; рекомендується 8 ГБ оперативної пам'яті (мінімум 2,5 ГБ при виконанні на віртуальній машині);

— місце на жорсткому диску: до 210 ГБ (мінімум 800 МБ) вільного місця в залежності від встановлених компонентів;

— швидкість жорсткого диска: для підвищення продуктивності встановіть Windows і Visual Studio на твердотільний накопичувач (SSD);

— відеоадаптер з мінімальним дозволом 720p (1280 на 720 пікселів);

— для оптимальної роботи рекомендується дозвіл WXGA (1366 на 768 пікселів) або вище.

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, додаток буде функціонувати, проте не гарантується надійність, швидкість обробки даних і безпека.

1.5.4 Вимоги до інформаційної та програмної сумісності

Дана інформаційна система повинна бути розроблена з використанням наступних програмно-апаратних засобів:

а) операційна система Microsoft Windows 7/8/10, Linux, macOS;

б) мови програмування JavaScript, HTML, CSS;

в) бібліотеки React, Redux;

Функціонування системи повинно бути забезпечено на наступних операційних системах (рекомендується 64-розрядної версія):

— Windows 10 версії 1703 і вище: Домашня, Pro, для освітніх установ і Корпоративна (випуски LTSC і S не підтримуються);

— Windows Server 2016/2019: Standard і Datacenter;

— Windows 8.1 (з оновленням 2919355): Core, Професійна і Enterprise;

— Windows Server 2012 R2 (з оновленням 2919355): Essentials, Standard, Datacenter;

— Windows 7 з пакетом оновлень 1 (SP1) (з останніми оновленнями Windows): Home Premium, Professional, Корпоративна, Максимальна;

— Mac OS X 10.9+;

— Ubuntu 16.04 LTS/16.10;

— Fedora 24/25;

— Oracle Enterprise Linux 7.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Функціональне призначення програми

Функціональним призначенням програми є надання можливості створення тестів, їх проходження та надання результатів, виходячі від ролі користувача: Вчитель (створення, надання та перегляд результатів тесту), Студент (перегляд результатів та проходження тесту), звичайний користувач (створення та проходження тестів, отримання результату).

Експлуатаційне призначення – автоматизація та спрощення ручної праці людини. Розроблена система дозволяє звести використання паперу до мінімуму, значно зменшує витрати часу та людських ресурсів для аналізу отриманих даних, їх упорядкування, збереження, а також сортування, редагування, пошуку та додавання.

2.2 Опис застосованих математичних методів

Жодних математичних методів при розробці даної платворми використано не було.

2.3 Опис використаних технологій та мов програмування

1. HTML - це мова розмітки, який ми використовуємо для візуального і смислового структурування нашого web контента, наприклад, визначаємо параграфи, заголовки, таблиці даних, або вставляємо зображення і відео на сторінку.

Мова розмітки HyperText - це комп’ютерна мова, яка полегшує створення веб-сайтів. Мов, яка має кодові слова та синтаксис, як і будь-яка інша мова,

порівняно легка для сприйняття і, з часом, стає все потужнішою у тому, що дозволяє комусь творити. HTML продовжує розвиватися, щоб задовольнити запити та вимоги Інтернету під виглядом Консорціуму всесвітньої павутини, організації, яка розробляє та підтримує мову; наприклад, з переходом на Web 2.0.

HyperText - метод, за допомогою якого користувачі Інтернету здійснюють навігацію в Інтернеті. Натискаючи на спеціальний текст, який називається гіперпосиланням, користувачі переходять на нові сторінки. Використання гіпер означає, що воно нелінійне, тому користувачі можуть перейти в будь-яку точку Інтернету, просто натиснувши на доступні посилання. Розмітка - це теги HTML, які роблять з текстом усередині них; вони позначають це як певний тип тексту. Наприклад, текст розмітки може мати жирний шрифт або курсив, щоб привернути увагу до слова чи фрази [11].

2. CSS - це мова стилів за допомогою якого ми надаємо стиль відображення нашого HTML контенту, наприклад надаємо колір фону (background) і шрифту, надаємо контенту Багатоколоночних вид.

Каскадні таблиці стилів, які залюбки називають CSS, - це проста мова дизайну, призначена для спрощення процесу зробити веб-сторінки презентабельними.

CSS обробляє зовнішній вигляд частини веб-сторінки. За допомогою CSS ви можете керувати кольором тексту, стилем шрифтів, інтервалом між абзацами, розміром та розміщенням стовпців, які фонові зображення або кольори використовуються, дизайном макета, варіаціями відображення для різних пристроїв та розмірами екрану а також безліч інших ефектів.

CSS легко вивчити і зрозуміти, але він забезпечує потужний контроль над поданням HTML-документа. Найчастіше CSS поєднується з мовами розмітки HTML або XHTML [22].

3. JavaScript – мова програмування, яка дозволяє вам створити динамічно оновлюваний контент, управляє мультимедіа, анімує зображення, втім, робить все, що завгодно.

Сторінки HTML чудово підходять для відображення статичного вмісту, напр. просте зображення або текст. Однак сьогодні більшість сторінок рідко бувають статичними. На багатьох сучасних сторінках є меню, форми, слайд-шоу та навіть зображення, що забезпечують взаємодію користувачів. Javascript - це мова, яку веб-розробники використовують для забезпечення такої взаємодії. Оскільки JavaScript працює зі сторінками HTML, розробник повинен знати HTML, щоб використати весь потенціал цієї мови сценаріїв. Хоча існують інші мови, які можна використовувати для створення сценаріїв в Інтернеті, на практиці це фактично всі Javascript [8].

Існує два способи використання JavaScript у файлі HTML. Перший передбачає вбудовування всього коду JavaScript у HTML-код, тоді як другий метод використовує окремий файл JavaScript, який викликається з елемента Script, тобто, укладеного тегами Script. Файли JavaScript ідентифікуються за розширенням .js. Хоча JavaScript здебільшого використовується для взаємодії з HTML-об'єктами, він також може бути створений для взаємодії з іншими не-HTML-об'єктами, такими як плагіни браузера, властивості CSS (каскадні таблиці стилів), поточна дата або сам браузер.

4. Node або Node.js - програмна платформа, заснована на движку V8 (здійснює трансляцію JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованого мови в мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виведення через свій API, написаний на C ++, підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з JavaScript-коду [15].

Node.js схожий за дизайном на та під впливом таких систем, як Ruby's Event Machine та Python's Twisted. Node.js піднімає модель події трохи далі. Він представляє цикл подій як конструкцію середовища виконання, а не як бібліотеку. В інших системах завжди є виклик блокування для запуску циклу подій.

Як правило, поведінка визначається через зворотні виклики на початку сценарію, а в кінці сервер запускається через блокуючий виклик, як EventMachine :: run (). У Node.js немає такого виклику циклу старту події. Node.js просто вводить цикл подій після виконання вхідного сценарію. Node.js виходить із циклу подій, коли більше немає зворотних викликів. Ця поведінка схожа на JavaScript браузера - цикл подій прихований від користувача.цикл обробки подій - єдиний потік, виконуючий функції з стека викликів і черги подій; (стек викликів - LIFO-чергу виконуваних функцій коду; черга подій - колбеки подій, що відбулися.);

– установка;

Платформа Node.js може бути встановлена кількома способами:

– за допомогою офіційних файлів для різних ОС;
– менеджером пакетів операційної системи;
– nvm;

Пакетний менеджер Node.js бере на себе всю роботу з установки цього зовнішніх залежностей проекту. Всі файли при цьому завантажуються в папку node_modules.

5. React JS - це бібліотека JavaScript, яка використовується у веб-розробці для створення інтерактивних елементів на веб-сайтах. Але якщо ви не знайомі з JavaScript або бібліотеками JavaScript, це не корисне визначення. Тож давайте зробимо крок назад і спершу розберемося з цими умовами.

React розробляється і підтримується Facebook, Instagram і співтовариством окремих розробників і корпорацій [18].

React може використовуватися для розробки односторінкових і мобільних додатків. Його мета - надати високу швидкість, простоту і масштабованість. Як бібліотеки для розробки призначених для користувача інтерфейсів React часто використовується з іншими бібліотеками, такими як MobX, Redux і GraphQL.

Віртуальний DOM (VDOM) - це концепція програмування, в якій ідеальне або «віртуальне» уявлення призначеного для користувача інтерфейсу

зберігається в пам'яті і синхронізується з «справжнім» DOM за допомогою бібліотеки, такий як ReactDOM (рис. 2.1.). Цей процес називається узгодженням.

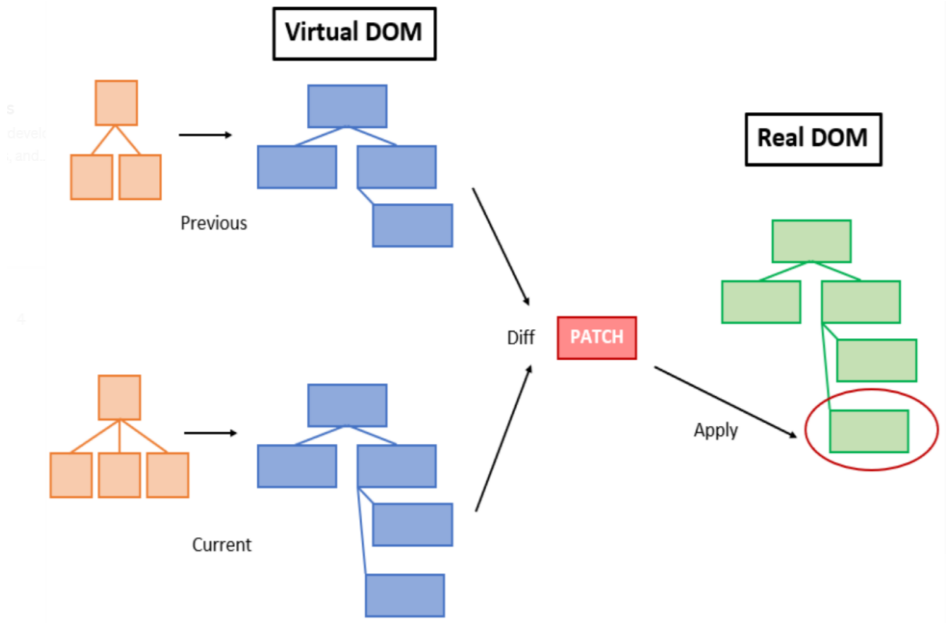

Рис. 2.1. Структура VDOM

Такий підхід і робить API React декларативним: ви вказуєте, в якому стані повинен перебувати призначений для користувача інтерфейс, а React домагається, щоб DOM відповідав цьому стану. Це відволікає маніпуляції з атрибутами, обробку подій і ручне оновлення DOM, які в іншому випадку довелося б використовувати при розробці програми.

Оскільки «віртуальний DOM» - це скоріше патерн, ніж конкретна технологія, цим терміном інколи позначають різні поняття. У світі React «віртуальний DOM» зазвичай асоціюється з React-елементами, оскільки вони є об'єктами, що представляють призначений для користувача інтерфейс. Проте, React також використовує внутрішні об'єкти, звані «волокнами» (fibers), щоб зберігати додаткову інформацію про дерево компонентів. Їх також можна вважати частиною реалізації «віртуального DOM» в React.

6. Redux - бібліотека з простим API, передбачуване сховище стану додатків. Вона працює за тим же принципом, що і функція reduce, один з концептів функціонального програмування. Її творці надихалися функціональним мовою програмування Elm (рис. 2.2.).

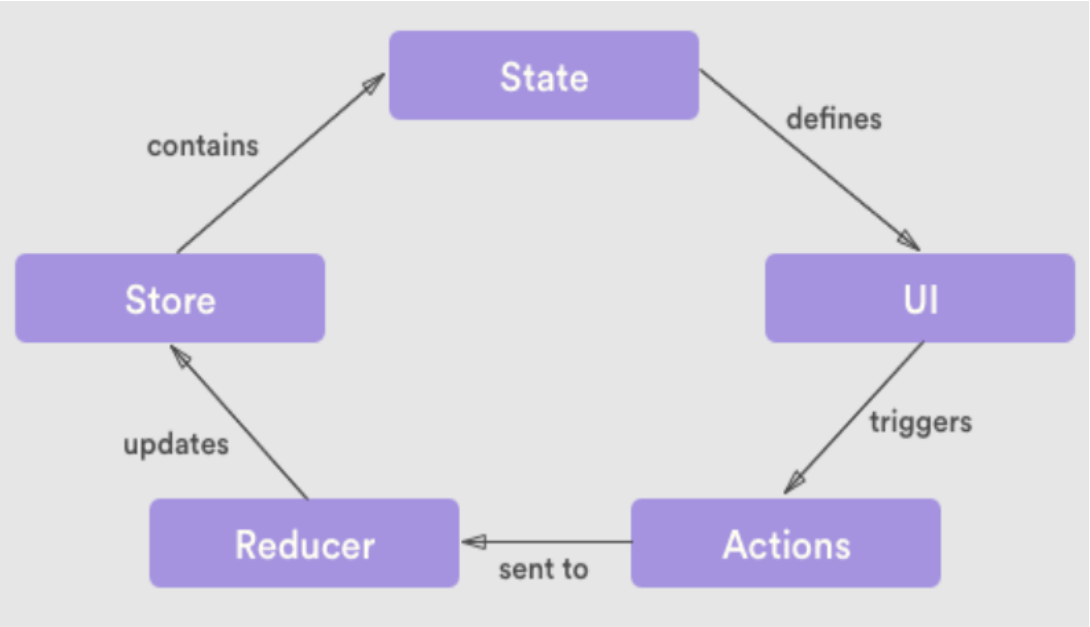

Рис. 2.2 Структура Redux

2.4 Опис структури системи та алгоритмів її функціонування

Файлова структура складається із 5 папок: core, node_modules, public, src (рис. 2.3).

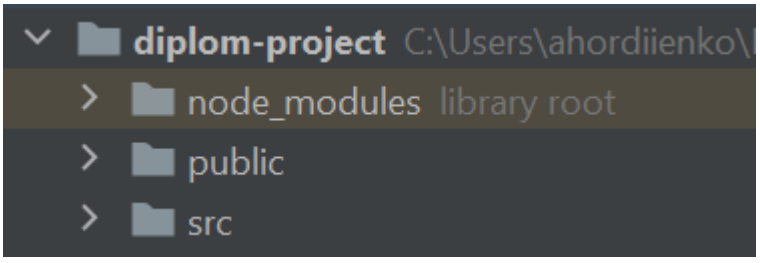

Рис. 2.3. Файлова структура кваліфікаційної роботи

1. node_modules – каталог, автоматично створений Node.js, який містить встановлені залежності та бібліотеки для використання.
2. public – каталог, який містить файл входу front-end`у - index.js (рис. 2.4):

Рис. 2.4. Вміст каталогу «public»

3. src – каталог, що містить функціональну складову платформи (рис. 2.5):

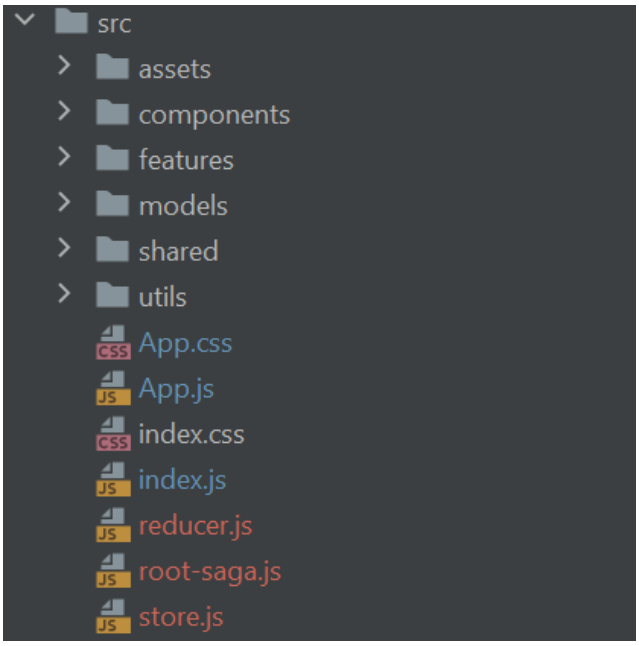

Рис. 2.5. Вміст каталогу «src»

a) assets – деректива, що містить медіа файли;

b) components – деректива, що містить перевикористовуюмі компоненти (кнопки, форми, таблиці) (рис. 2.6);

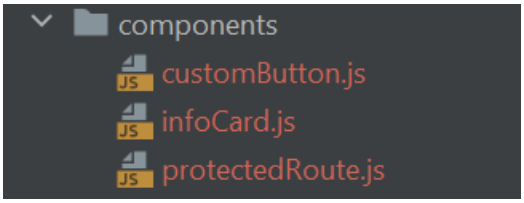

Рис. 2.6. Деректива – components

c) features – деректива, що містить усі сторінки платформи (рис 2.7);

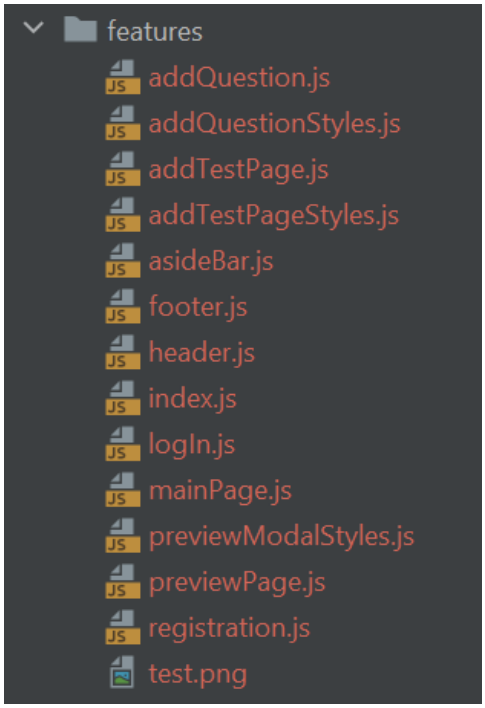

Рис. 2.7. Деректива – features

d) models – файли, що слугують для з‘єднання з базою даних (рис 2.8);

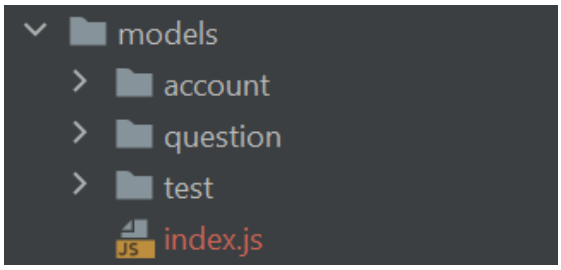

Рис. 2.8. Деректива – models

e) shared – основні файли для задання логіки даних (рис. 2.9);

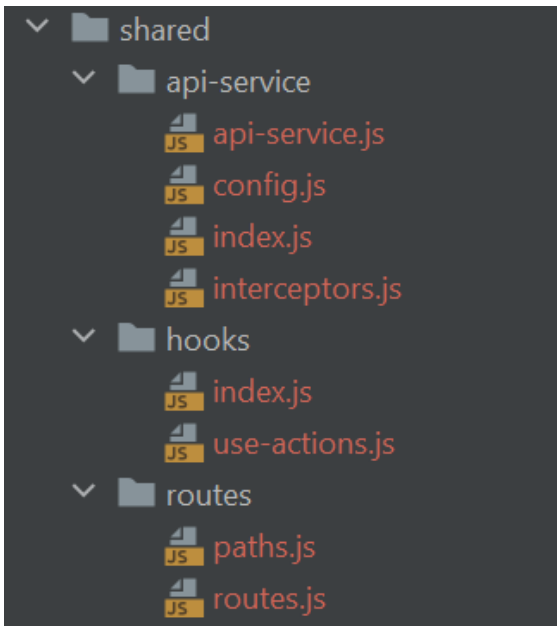

Рис. 2.9. Деректива – shared

f) utils – валідатори (рис. 2.10).

Рис. 2.10. Деректива – utils

Наповнення сайту, функціонування підтримується однією інсталяцією системи, зберігається під управлінням єдиної СУБД (рис. 2.11).

Спроектована база даних містить 10 пов’язаних між собою таблиць у першій її версії. Нижче можна побачити фізичну модель створюваної бази даних.

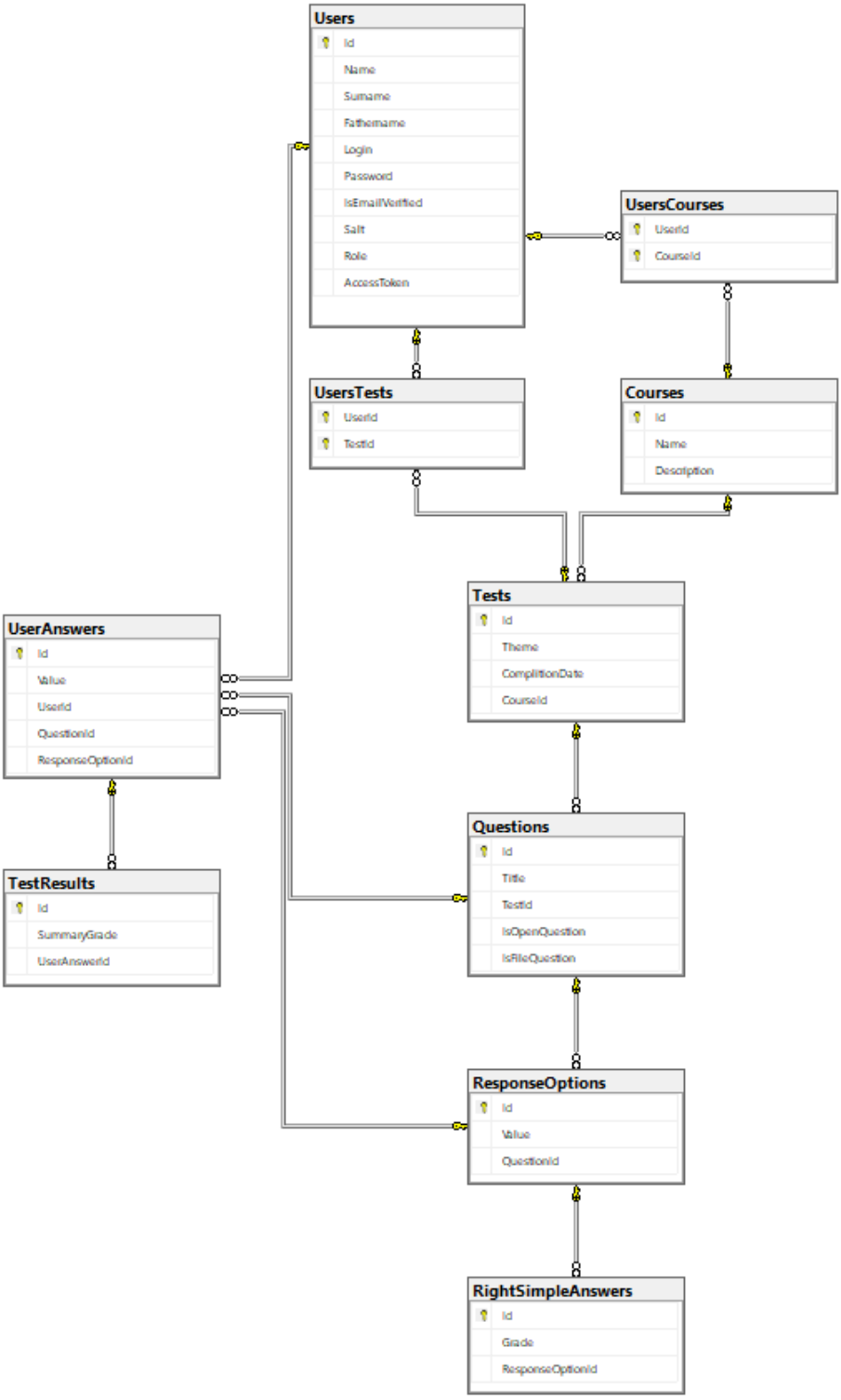

Рис. 2.11. – Фізична модель створюваної БД

Розглянемо таблиці у вищезазначеній базі даних.

Таблиця 2.1

Користувач

Назва поля	Призначення	Тип даних	Ключ
Id	ID користувача	int	PK
Name	Ім'я	Varchar(50)	
Surname	Прізвище	Varchar(50)	
Fathername	По-батькові	Varchar(50)	
Login	Логін (електронна пошта)	Varchar(50)	
Password	Захешований пароль	Varchar(50)	
IsEmailVerified	Флаг верифікації електронної пошти	Bit	
Salt	Сіль для хешування паролю	Varchar(50)	
Role	Роль	int	
AccessToken	Токен доступу	Varchar(50)	

Таблиця 2.2

Курс

Назва поля	Призначення	Тип даних	Ключ
Id	ID курсу	int	PK
Name	Назва	Varchar(50)	
Description	Детальний опис курсу	Varchar(50)	

Таблиця 2.3

Тест

Назва поля	Призначення	Тип даних	Ключ
Id	ID теста	int	PK
Theme	Тема теста	Varchar(50)	
ComplitionDate	Дата проходження теста	Date (UTC)	
PassTime	Час проходження теста (у хвилинах)	int	
CourseId	ID курсу, до якого прикріплений тест	int	FK

Таблиця 2.4

Питання

Назва поля	Призначення	Тип даних	Ключ
Id	ID питання	int	PK
Title	Заголовок	Varchar(50)	
TestId	ID теста, якому належить питання	int	FK
IsOpenQuestion	Флаг відкритого питання	Bit	
IsFileQuestion	Флаг файлового питання	Bit	

Таблиця 2.5

Варіант відповіді

Назва поля	Призначення	Тип даних	Ключ
Id	ID варіанту відповіді	int	PK
Title	Заголовок	Varchar(50)	
Value	Зміст варіанту відповіді	Varchar(50)	
QuestionId	ID питання, за яким закріплений варіант відповіді	int	FK

Таблиця 2.6

Правильні відповіді

Назва поля	Призначення	Тип даних	Ключ
Id	ID правильної відповіді	int	PK
Grade	Оцінка за обраний варіант відповіді	Decimal	
ResponseOptionId	ID варіанта відповіді, яка є вірною	int	FK

Таблиця 2.7

Відповідь користувача

Назва поля	Призначення	Тип даних	Ключ
Id	ID відповіді користувача	int	PK
UserId	ID користувача	int	FK
QuestionId	ID питання, якому надав відповідь користувач	int	FK
ResponseOptionId	ID варіанта відповіді, обраної користувачем	int	FK
Value	Шлях до файлу, якщо користувач надав файлову відповідь на питання, або відкрита відповідь на питання	Varchar(500)	

Також в базі наявні сполучні таблиці для забезпечення зв’язку багато до багатьох між сутностями.

1. Модель для створення акаунта (рис. 2.12)

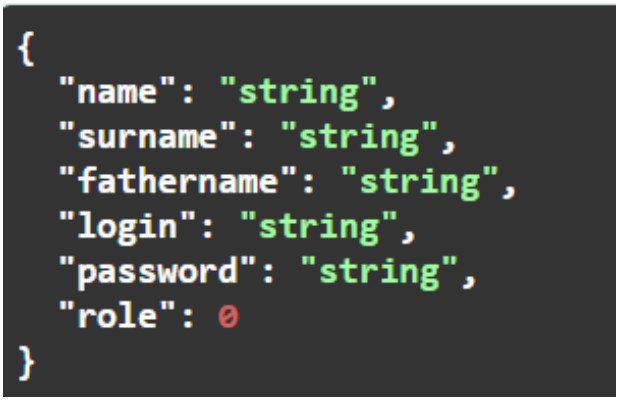

Рис. 2.12. Моделя створення акаунта

2. Модель входу в акаунт (рис. 2.13).

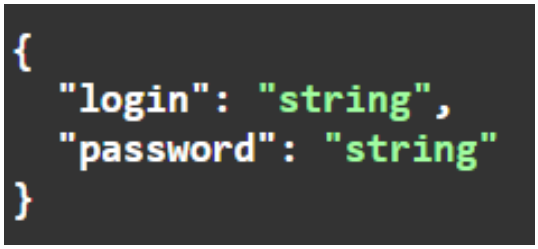

Рис. 2.13. Модель входу до акаунта

3. Модель виходу з системи (рис. 2.14).

Рис. 2.14. Модель виходу з системи

4. Модель оновлення токена (рис. 2.15).

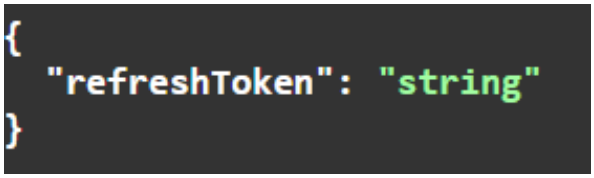

Рис. 2.15. Модель оновлення токена

5. Модель створення курсу (рис. 2.16).

Рис. 2.16. Модель створення курсу

6. Модель створення теста (рис. 2.17).

Рис. 2.17. Модель створення тесту

2.5 Обґрунтування та організація вхідних та вихідних даних програми

Вхідні дані надаються системі шляхом заповнення форм користувачами, валідація для кожного з полей, аби запобігти непередбачуваним помилкам. Згідно валідації дані впорядковані за логікою та мають обмежену довжину виходячи з обмежень бази даних. Інформаційна система складається із форм: registration, login, add-test, add-question.

1. Форма «registration» – це форма (рис. 2.18), яка призначена для реєстрації користувача у системі.

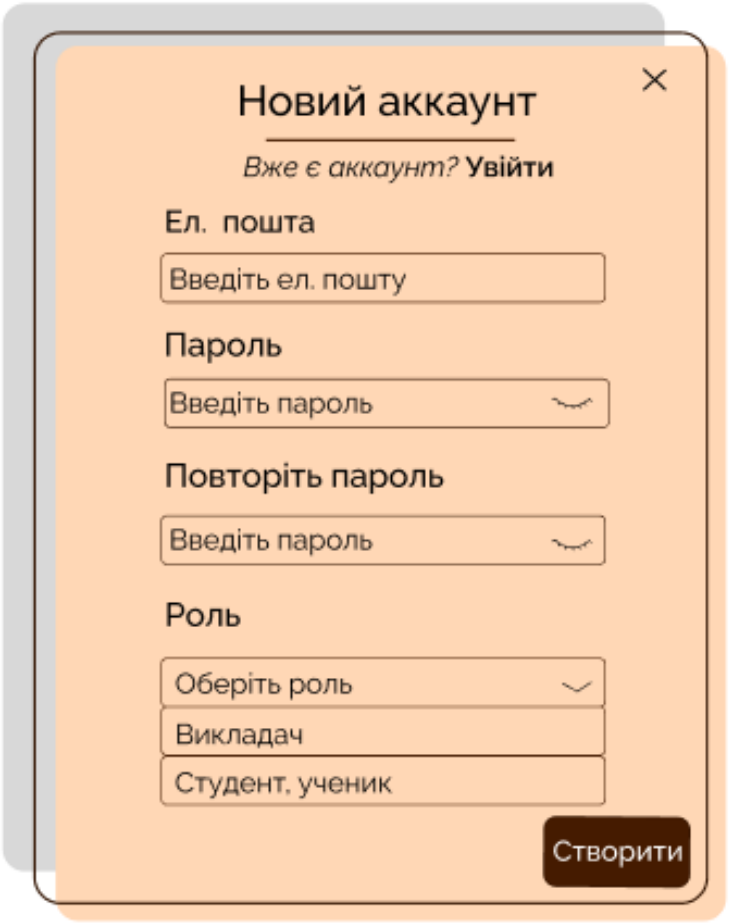

Рис. 2.18. Зовнішній вигляд форми «registration»

Містить у собі наступні поля (формат запису – html-елемент (ім’я форми)):

a) Input (ел.пошта) – використовується для задання логіну користувача, за яким він може бути авторизований:

– обов’язковий для заповнення;

– мінімальна довжина: 6;

– максимальна довжина: 12;

– може включати у себе: англійські літери, нижні підкреслювання (першими повинні обов’язково бути літери, два нижніх підкреслювання не можуть бути один за одним);

b) Input (пароль) – використовується для задання паролю користувача, за яким він може бути авторизований:

– обов’язковий для заповнення;

– мінімальна довжина: 6;

– максимальна довжина: 20;

– повинно включати у себе: хоча б одну англійську літеру, одну цифру;

c) Input (повторити пароль) – використовується для перевірки введеного вище паролю в цілях безпеки:

– обов’язковий для заповнення;

– мінімальна довжина: 6;

– максимальна довжина: 20;

– повинно включати у себе: хоча б одну англійську літеру, одну цифру;

– ідентичний до першого введеного паролю;

d) Select (роль) – використовується для вибору категорії користувача (Студент, Викладач), за якою потім буде розрахована подальша робота профілю користувача (особисті дані, питання тощо):

– обов’язковий для заповнення;

e) Input (submit) – призначений для відправки форми на перевірку.

2. Форма «login» – це форма (рис. 2.19), яка призначена для авторизації користувача у системі.

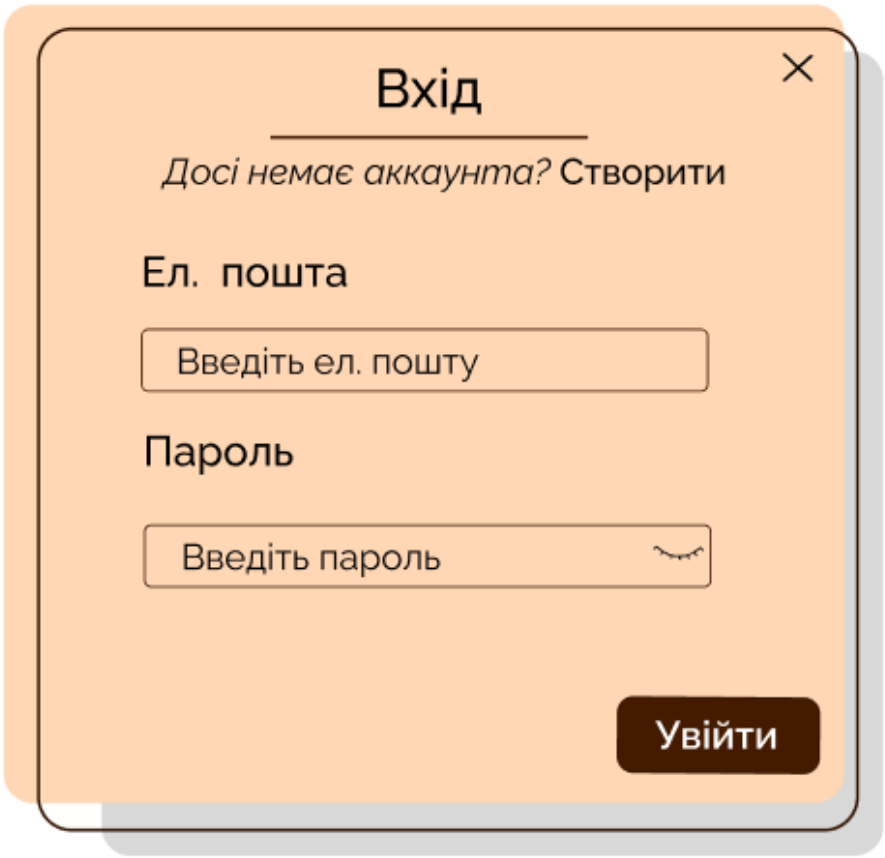

Рис. 2.19. Зовнішній вигляд форми «login»

Містить у собі наступні поля:

a) Input (ел.пошта) – використовується для задання логіну користувача, за яким він може бути авторизований:

– обов’язковий для заповнення;

b) Input (пароль) – використовується для задання паролю користувача, за яким він може бути авторизований:

– обов’язковий для заповнення;

c) Input (створити) – призначений для відправки форми на перевірку.

3. Форма «add-test» – це форма, яка призначена для занесення у базу даних інформації про новий тест (рис. 2.20) .

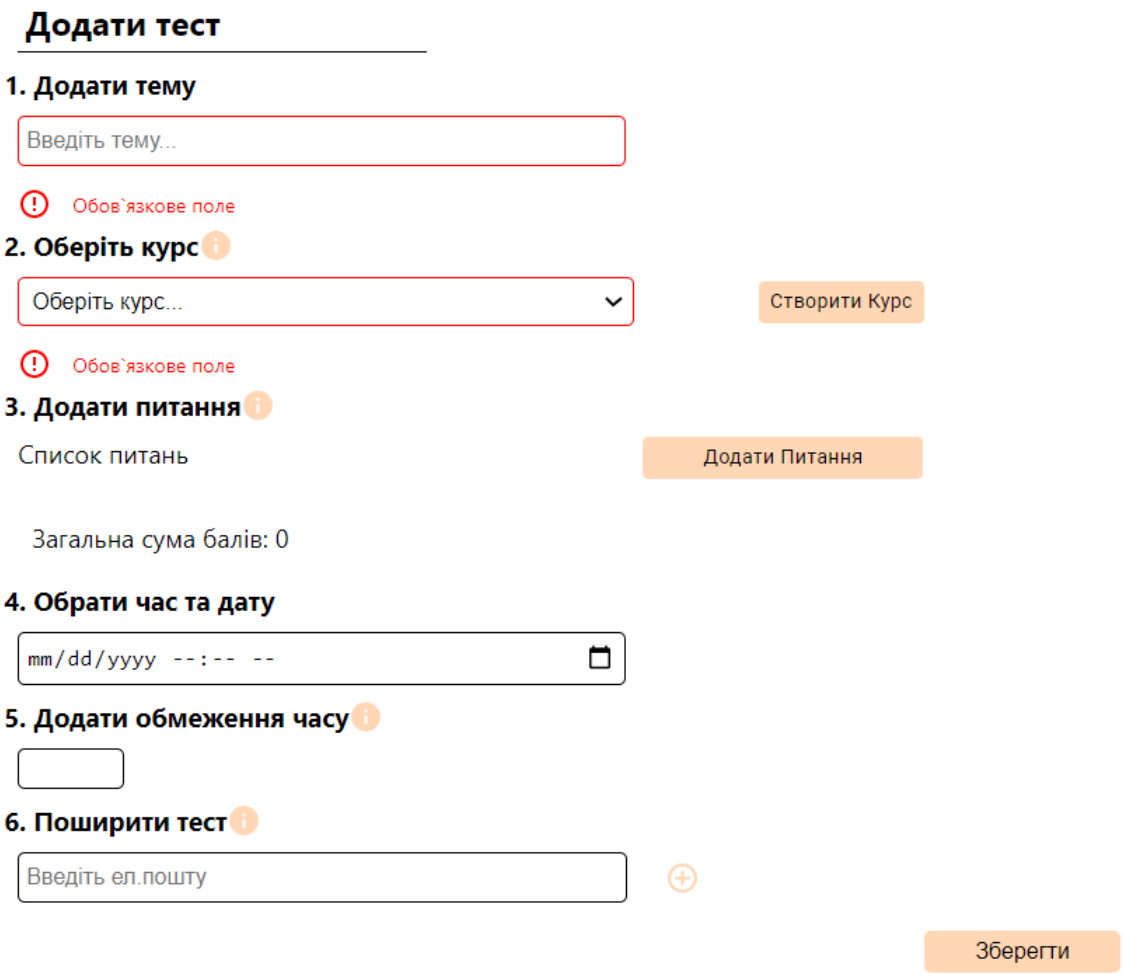

Рис. 2.20. Зовнішній вигляд форми «add-test»

a) Input (додати тему) – поле для введення теми тесту.
   - Обов’язкове для введення;
b) Select (оберіть курс) – поле для вибору курсу.
   - Обов’язкове для введення;
c) Select (час та дата) – поле для вибору часу та дати проведення тесту.
   - Обов’язкове для введення;
d) Input (обмеження часу) – поле для введення часу обмеження тесту.
e) Input (респонденти тесту) – поле для введення ел. пошт.

4. Форма «add-question» – це форма, яка призначена для занесення у базу даних інформації про новий тест (рис. 2.21).

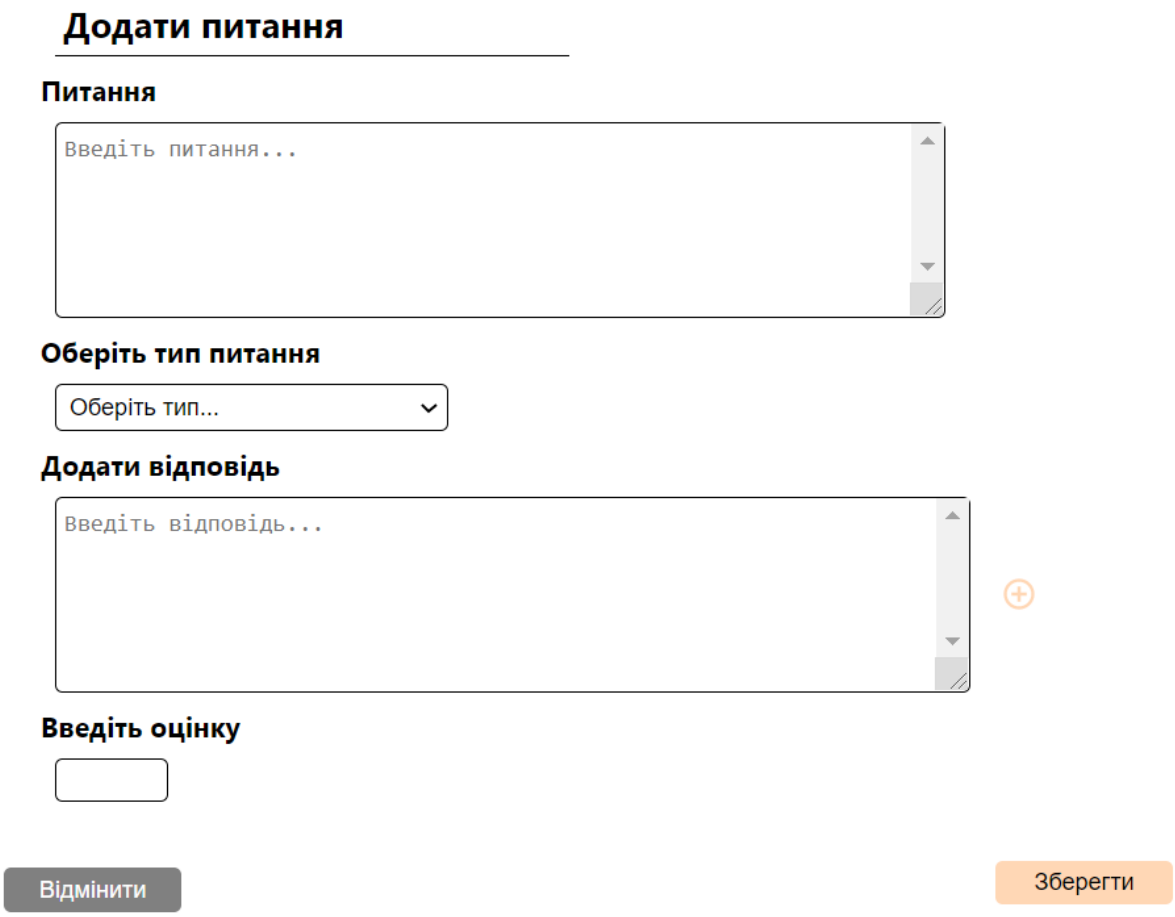

Рис. 2.21. Зовнішній вигляд форми «add-question»

a) Input (питання) – поле для введення питання.
   - Обов’язкове для введення;
b) Input (тип) – поле для введення типу питання.
   - Обов’язкове для введення;
   - Має опції (одна відповіль, декілька відповідей, відкрита відповідь, документ)
c) Input (оцінка) – поле для введення оцінки.
   - Обов’язкове для введення;

2.6 Опис розробленої системи

2.6.1. Використані технічні засоби

В процесі тестування та розробки були використані наступні технічні засоби:

– оперативна пам’ять обсягом 32 гб;

– процесор AMD Ryzen 5 1600 Six-Core Processor 3.20 GHz;

– накопичувач на жорсткому диску обсягом 250 гб;

– відеоадаптер NVIDIA GeForce GTX 1050 Ti;

– клавіатура та маніпулятор типу «миш».

Вказані технічні засоби не мають бути обов’язково ідентичних характеристик для безперешкодної роботи системи.

2.6.2. Використані програмні засоби

В процесі тестування та розробки були використані наступні програмні засоби:

– операційна система Windows (64-розрядна);

– Visual Studio Code (редактор програмного коду з можливістю взаємодії з програмою через консоль, в тому числі для запуску, коригування помилок програми та її перезапуску за необхідністю);

– GitHub (сервіс, який дозволяє зберігати прогрес розробки проекту в цілому або окремих його файлів з можливістю відновлення за необхідністю);

Згадані програмні засоби використовувалися при розробці та не мають використовуватися користувачем (виключення – операційна система).

2.6.3. Виклик та завантаження програми

Для роботи з розробленим веб-додатком потрібно лише звернутися за адресою https://safe-brushlands-00491.herokuapp.com (підтримується хмарним сервісом Heroku [35,36,37] та не потребує додаткових дій або ПЗ, за виключенням засобу звернення до інтернету – браузера, вбудованих сервісів Windows тощо).

2.6.4. Опис інтерфейсу користувача

1. Головна сторінка.

Головна сторінка сайту містить графічну частину, навігаційне меню сайту, а також контентну область для того, щоб відвідувач сайту з першої сторінки міг отримати ввідну інформацію про сайт. Авторизувавшись, користувач зможе ознайомитись з можливостями платформи “Test On”. Тематична область першої сторінки поділяється на наступні розділи:

а) вступ «навігаційне меню» («Test On» - перехід до головної сторінки сайту, якщо користувач не авторизований, або інакше – на домашню сторінку, «Курси» - список існуючих курсів, «Тести» - список існуючих тестів, (рис. 2.22);

Рис. 2.22. Зовнішній вигляд навігаційного меню

б) банер (основна думка сайту (рис. 2.23));

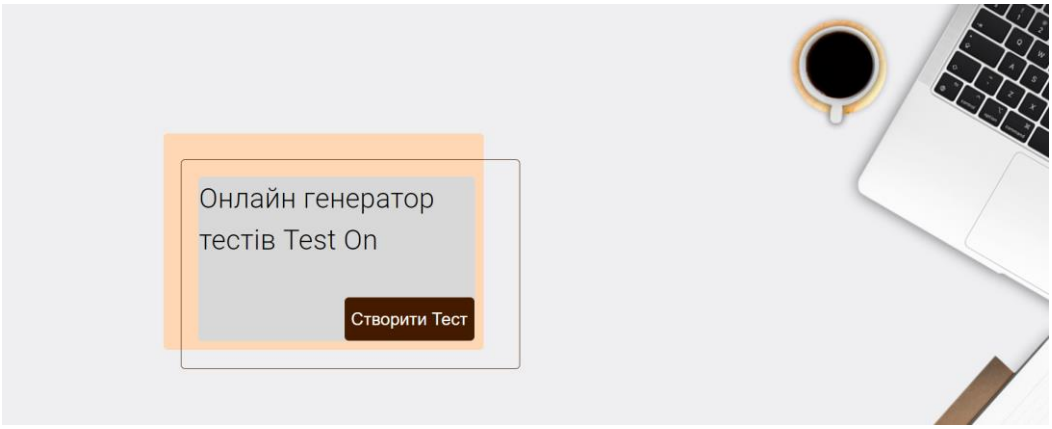

Рис. 2.23. Зовнішній вигляд банеру

в) інформативна частина (рис. 2.24);

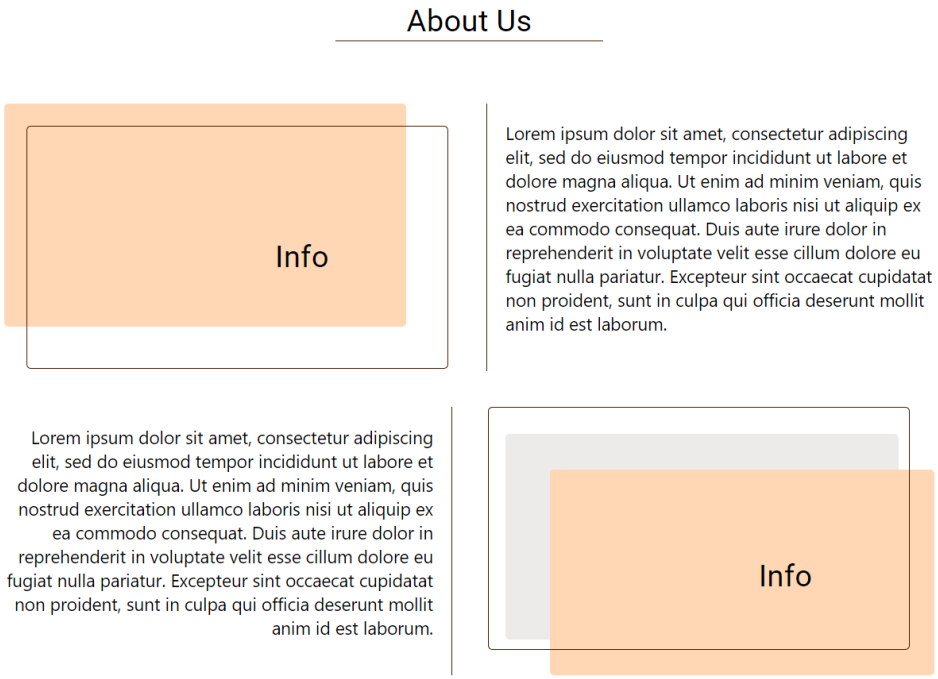

Рис. 2.24. Зовнішній інформативної частини

г) заключна частина (контактна інформація (Facebook, Twitter, Instagram), інформація про сайт, посилання з навігаційного меню (за виключенням мобільних пристроїв)) (рис. 2.25);

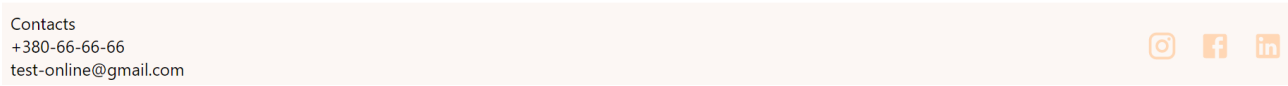

Рис. 2.25. Зовнішній вигляд заключної частини

2. Список тестів (рис. 2.26).

– Містить фільтрацію за курсом, статусом, датою;

– Містить пошук за темою тестів;

– Можливий перехід до сторінки редагування;

– Можливість перегляду аналізу тестів;

– Можливість переходу на сторінку створення тесту;

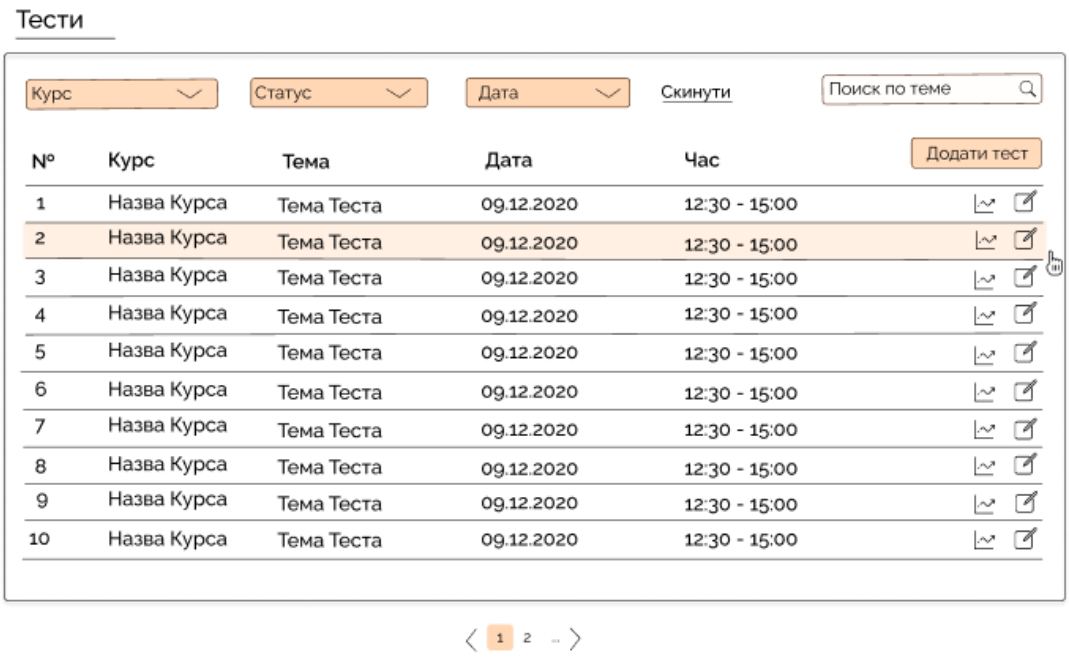

Рис. 2.26. Список тестів

3. Список курсів (рис. 2.27).

– Містить пошук за назвою курсу;

– Можливий перехід до сторінки редагування;

– Можливість перегляду респондентів;

– Можливість переходу на сторінку створення курсу;

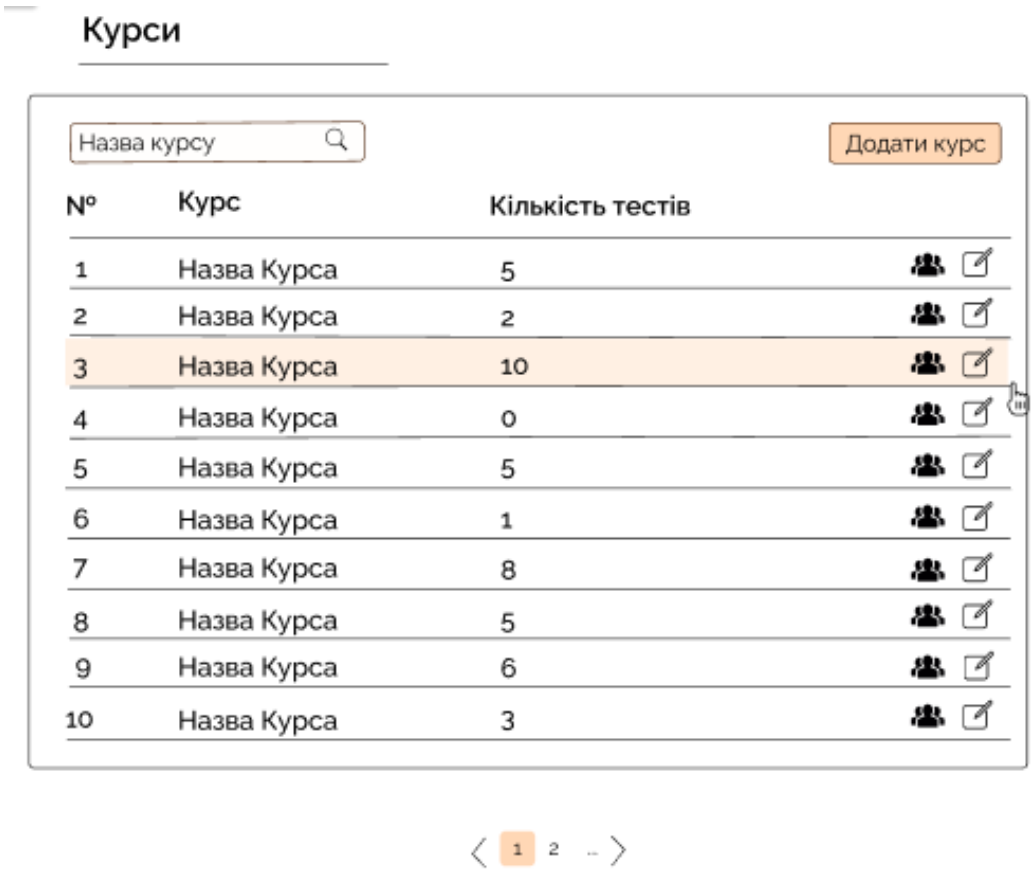

Рис. 2.27. Список курсів

4. Список робіт для оцінювання відкритих завдань (рис.2.28).

– Містить фільтрацію за курсом, датою;

– Містить пошук за темою тестів;

– Можливий перехід до сторінки редагування;

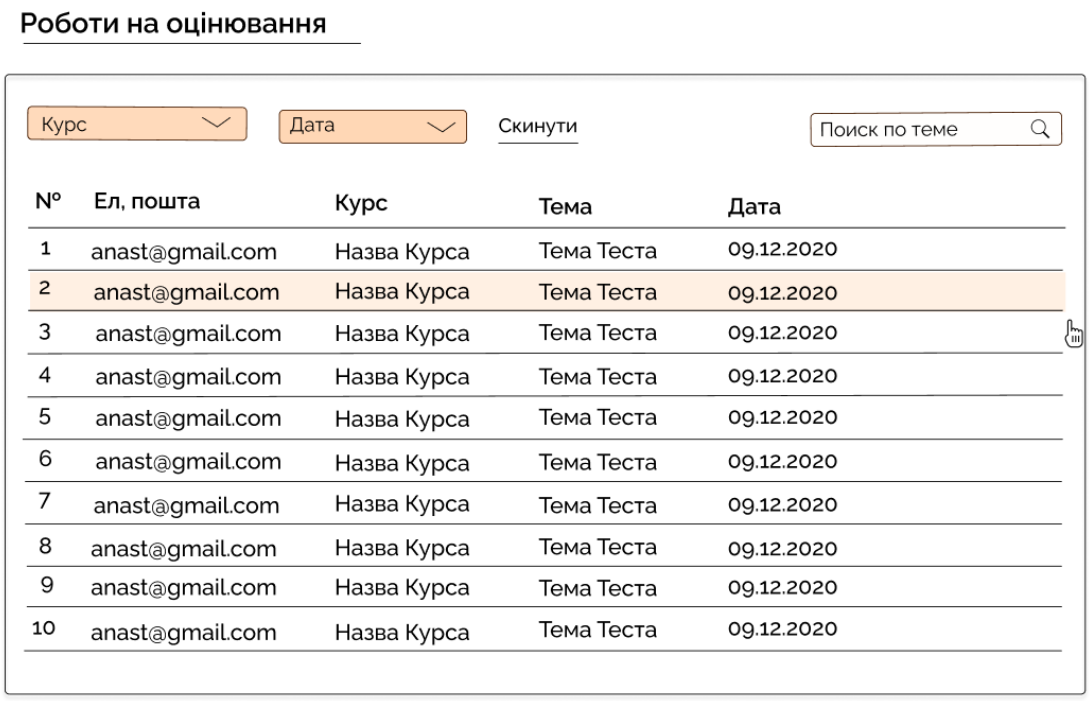

Рис. 2.28. Роботи на оцінювання

5. Список робіт для оцінювання відкритих завдань (рис.2.29).

– Містить ім’я респондента;

– Містить номер варіанту;

– Містить тему тесту;

– Містить дату надсилання завдання;

– Містить формулювання завдання;

– Містить відповідь;

– Містить поле для введення оцінки;

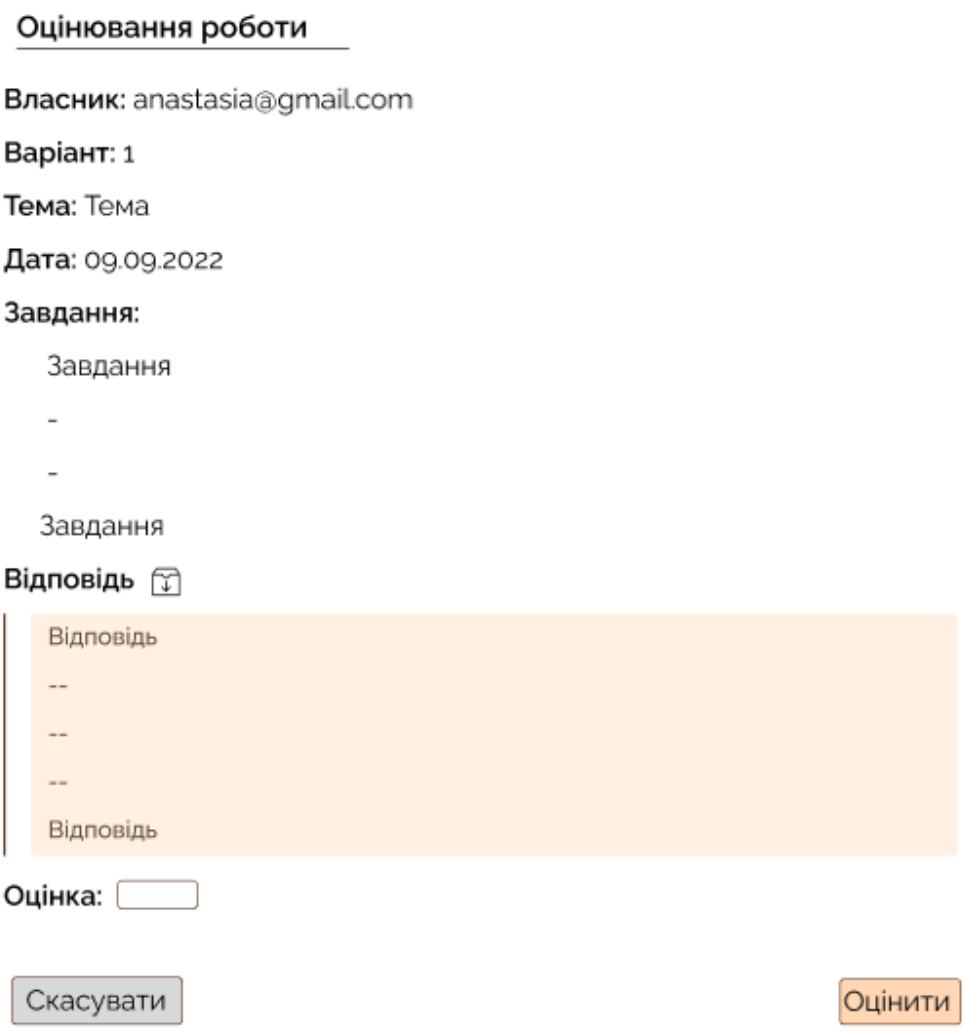

Рис. 2.29. Робота на оцінювання

6. Сорінка створення курсу (рис. 2.30).

a) Input (назва курсу) – введення назви створюваного курсу.

b) Input (підписники курсу) – додавання підписників курсу.

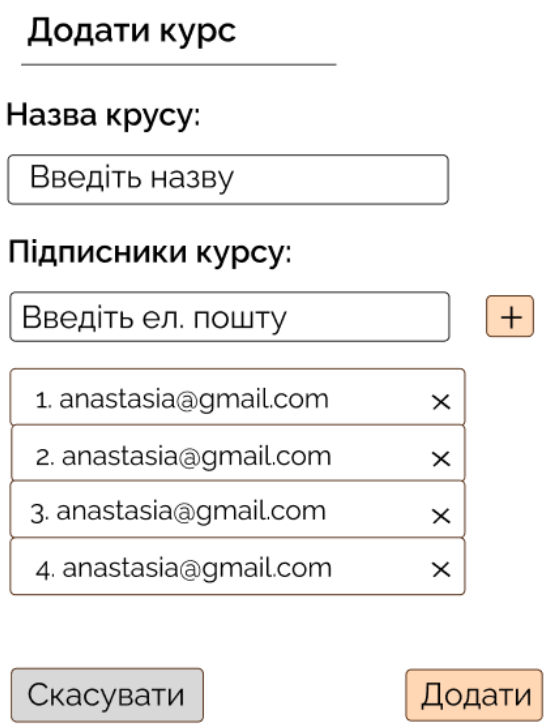

Рис. 2.30. Сторінка створення курсу

Сайт оптимізований для перегляду на всіх девайсах без горизонтальної смуги прокрутки (за винятком звітів) і без порожніх (білих) полів для основних типів дозволу. На кожній сторінці відображається контактна інформація.

Інтерфейс модулів виконаний в єдиному стилі з інтерфейсом ядра системи і забезпечує можливість прозорого переміщення між модулями системи і використання однакових процедур управління і навігаційних елементів для виконання однотипних операцій.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

Автоматизуючи суспільні процеси за допомогою ПЗ важливо визначити трудомісткість розробки програмного забезпечення і розрахувати витрати на створення інформаційної системи .

3.1. Розрахунок трудомісткості та вартості розробки інформаційної системи

Початкові дані:

1. передбачуване число операторів – 1100;

2. коефіцієнт складності програми – 1.2;

3. коефіцієнт корекції програми в ході її розробки – 0.05;

4. годинна заробітна плата програміста, грн/год – 72;

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;

6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1.2;

7. вартість машино-години, грн/год – 11.

Нормування праці в процесі створення програмного забезпечення істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки програмного забезпечення може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки програмного забезпечення можна розрахувати за формулою:

$$t = t_0 + t_u + t_a + t_n + t_{отл} + t_д, людино - годин, \quad (3.1)$$

де t_0 - витрати праці на підготовку й опис поставленої задачі (приймається 50);

$t_и$ - витрати праці на дослідження алгоритму рішення задачі;

$t_а$ - витрати праці на розробку блок-схеми алгоритму;

$t_п$ - витрати праці на програмування за готовою блок-схемою;

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

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється. Умовне число операторів (тегів):

$$Q = q * C * (1 + p), \quad (3.2)$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт корекції програми в ході її розробки.

Звідси умовне число операторів в програмі:

$$Q = 1100 * 1{,}2 * (1 + 0{,}05) = 1386. \quad (3.3)$$

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

$$t_u = \frac{Q*B}{(75...85)*k}, людино - годин, \quad (3.4)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1,2). З урахуванням коефіцієнта кваліфікації k = 1,2, отримуємо витрати праці на вивчення опису завдання:

$$t_u = \frac{1386*1,2}{85*1,2} = 16, \text{людино} - \text{годин}. \quad (3.5)$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20...25)*k}, \text{людино} - \text{годин} \quad (3.6)$$

$$t_a = \frac{1386}{25*1,2} = 46, \text{людино} - \text{годин}. \quad (3.7)$$

Витрати на складання програми за готовою блок-схемою:

$$t_n = \frac{Q}{(20...25)*k}, \text{людино} - \text{годин}, \quad (3.8)$$

$$t_n = \frac{1386}{25*1,2} = 46, \text{людино} - \text{годин}. \quad (3.9)$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)*k}, \text{людино} - \text{годин}. \quad (3.10)$$

$$t_{отл} = \frac{1386}{5*1,2} = 231, \text{людино} - \text{годин}. \quad (3.11)$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{k} = 1{,}5 * t_{отл}, людино - годин. \quad (3.12)$$

$$t_{отл}^{k} = 1{,}5 * 231 = 277, людино - годин. \quad (3.13)$$

Витрати праці на підготовку документації:

$$t_{д} = t_{др} + t_{до}, людино - годин, \quad (3.14)$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{15...20*k}, людино - годин. \quad (3.15)$$

$$t_{др} = \frac{1386}{20*1{,}2} = 58, людино - годин, \quad (3.16)$$

де $t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 * t_{др}, людино - годин. \quad (3.17)$$

$$t_{до} = 0{,}75 * 58 = 44, людино - годин. \quad (3.18)$$

$$t_{д} = 58 + 44 = 102, людино - годин. \quad (3.19)$$

Тепер розрахуємо трудомісткість ПЗ:

$$t = 50 + 16 + 46 + 46 + 231 + 102 = 491, людино - годин. \quad (3.20)$$

3.2. Розрахунок витрат на створення інформаційної системи

Витрати на створення програмного забезпечення Витрати на створення даного продукту $К_{по}$ включають витрати на заробітну плату виконавця програми $З_{зп}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{по} = З_{зп} + З_{мв}, грн. \quad (3.21)$$

Заробітна плата виконавців визначається за формулою:

$$З_{зп} = t * С_{пр}, грн, \quad (3.22)$$

де t - загальна трудомісткість, людино-годин;

$C_{пр}$ - середня годинна заробітна плата програміста, грн/година.

$$З_{зп} = 491 * 72 = 35,352 грн. \quad (3.23)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{мв} = t_{отл} * С_{мч}, грн, \quad (3.24)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

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

$$З_{мв} = 231 * 11 = 2,541 грн. \quad (3.25)$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП:

$$К_{по} = 35,352 + 2,541 = 37,893 \text{ грн.} \quad (3.26)$$

Очікуваний період створення ПЗ:

$$Т = \frac{t}{B_к * F_р}, \text{міс}, \quad (3.27)$$

де B_k - число виконавців (1);

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

$$Т = \frac{491}{1*176} = 3 \text{ міс.} \quad (3.28)$$

Висновок:

У кваліфікаційній роботі був створений експериментальний веб-додаток для створення онлайн тестів. В третьому (економічному) розділі були визначені витрати на створення ПЗ, зазначеного у технічному завданні кваліфікаційної роботи – 37,893 грн. і час створення програмного забезпечення – 3 місяці.

ВИСНОВКИ

Результатом розробки навчальний вебдодаток для генерації та проходження онлайн тестів, призначений для автоматизації та оптимізації використання людських ресурсів (часу, здоров'я, паперу тощо).

Необхідністю створення вебдодатку зумовлена автоматизацією процесу тестування та обробки результатів навчання. Значне використання паперу і часу роботи виправдовує використання розробляємого додатку – збереження часу, сил і ресурсів.

Призначенням для розробки є оптимізація навчання шляхом обробки, аналізу та зберігання результатів, отриманих під час тестування певної групи осіб.

Створений додаток має наступні функціональні можливості:

1. Інформаційна система повинна створювати тести, забезпечувати досутп для їх проходження.
2. Система повинна мати зручний та інтуїтивно зрозумілий інтерфейс для користувача.
3. Результати аналізу повинні виводитися у інтерфейсі. Результатом є статистичні дані, представлені у вигляді таблиць та графіуів, які доступні лише власникам тестів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алиев Ш.М. О совершенствовании современной парадигмы образования // СГЗ. – 2011. - №3. – С. 150-156.

2. Артюхина М.Г. Ценности и приоритеты в современной парадигме образования // СГЗ. – 2012. - № 1. – С. 320-326.

3. Вербицкий А.А. Становление новой образовательной парадигмы в российском образовании // Образование и наука. Известия УральскогоРАО. – 2012. - №6. – С. 5-19.

4. Воронов М.В. Профессиональное обучение студентов на основе интегрированных курсов // Инновации в образовании. – 2011. - № 9. – С. 4- 15.

5. Газалиев А.М. Значение профессиональной мобильности в процессе становления специалиста технической направленности // Высшее образование сегодня. – 2011. - № 10. – С. 6-10.

6. Голованова Н.Ф. Проблема воспитания студентов в контексте модернизации высшего образования // Высшее образование в России. – 2012. - №7. – С. 29-35.

7. Гришнова Е.Е. Модернизация учебного процесса: проблемы и тенденции // Высшее образование в России. – 2011. - № 8-9. – С. 41-46.

8. Гуськова М.В. Эволюция процесса и результатов внедрения федеральных государственных образовательных стандартов в практику высшего профессионального образования // Высшее образование сегодня. – 2011. - №9. – С. 10-14.

9. Данилов М.Б. Компетентностный подход в образовании как новая форма подготовки специалистов пищевых отраслей // Пищевая промышленность. – 2011. - № 12. – С. 8-10.

10. Егорова И.П. Потенциальные возможности проблемных методов обучения в профессиональной подготовке студентов технических специальностей вуза // Инновации в образовании. – 2011. - № 10. – С. 25-37.

11. Ершова Н.Ю. Формирование профессиональных компетенций специалистов в области информационных технологий // Высшее образование сегодня. – 2012. - №3. – С. 24-28.

12. Жохов А.П. О культуре профессионала как главном ориентире модернизации современного образования // Образование и наука. Известия Уральского РАО. – 2011. - №9. – С. 42-52.

13. Загвязинский В.И. Стратегические ориентиры развития отечественного образования и пути их реализации // Образование и наука. Известия Уральского РАО. – 2012. - № 4. – С. 3-16.

14. Захарова Г.П. Основная образовательная программа бакалавра: опыт и оценка перспектив // Педагогика. – 2012. - № 3. - С. 70-76.

15. Зеер Э.Ф. Компетентностный подход как фактор реализации инновационного образования // Образование и наука. Известия Уральского РАО. – 2011. - № 8. – С. 3-15.

16. Игошев Б.М. Современное образование: проблемы и решения // Альма Матер. – 2011. - № 10. – С. 6-11.

17. Красинская Л.Ф. Реформирование высшего технического образования и новые требования к психолого-педагогической компетентности преподавателя // Пед.образование и наука. – 2012. - №5. 0 С. 28-32.

18. Лазарев В.С. Новое понимание методов проектов в образовании // Педагогика. – 2011. - № 10. – С. 3-12.

19. Лапчик М.П. ИКТ-компетентность магистров образования // Информатика и образование. – 2012. - № 5. – С. 24-31.

20. Підготовка спеціалістів дошкільної освіти у вищих освітніх закладах Турції / Дегирменджи Мюджахит, ЮНПУ, 2015 / URL: http://dspace.pdpu.edu.ua/bitstream/123456789/1594/1/%D0%94%D0%95%D0%93%D0%98%D0%A0%D0%9C%D0%95%D0%9D%D0%94%D0%96%D0%98%20%D0%9C%D1%8E%D0%B4%D0%B6%D0%B0%D1%85%D0%B8%D1%82%20%281%29.pdf. Дата звернення: 03.01.2020.

21. Якісна освіта в Україні: тенденції, проблеми, перспективи / Ігор Михайлович Зварич / ДВНЗ ЧНУ, 2017 / URL: http://quaere.fmi.org.ua/docs/conference_theses.pdf. Дата звернення: 07.10.2019.

22. Концепція забезпечення якості вищої освіти України / команда TRUST, 2015 / URL: http://dovira.eu/images/QA_concept_Final.pdf. Дата звернення: 08.10.2019.

23. NODE системные требования / URL: https://studbooks.net/2263675/informatika/harakteristiki_node. Дата звернення: 16.11.2019.

24. Расчёт затрат на создание программного обеспечения / 2016 / URL: https://studfile.net/preview/6308970/. Дата звернення: 25.04.2020.

25. Розробка програм з відкритим програмним кодом: плюси і мінуси / 2020 / URL: https://techrocks.ru/2020/01/17/open-source-software-development/. Дата звернення: 18.05.2020.

26. Цикл подій Node.js / 2017 / URL: https://medium.com/devschacht/event-loop-timers-and-nexttick-18579cd122e0. Дата звернення: 19.05.2020.

27. Переваги та недоліки Open Source Software / 2015 / URL: https://spsystems.lv/blog/eto-interesno/preimushhestva-i-nedostatki-open-source-software.html. Дата звернення: 19.05.2020.

28. Керівництво за Node.js, частина 6: цикл подій, стек викликів, таймери / 2018 / URL: https://habr.com/ru/company/ruvds/blog/424553/. Дата звернення: 20.05.2020.

29. Getting started on Heroku with Node.js / 2020 / URL: https://devcenter.heroku.com/articles/getting-started-with-nodejs?singlepage=true. Дата звернення: 10.02.2020.

30. Connecting mysql database with laravel app in heroku / 2018 / URL: https://medium.com/@ajshantudas/connecting-mysql-database-with-laravel-app-in-heroku-5d593b60847a. Дата звернення: 3.03.2020.

31. Quick Start Guide: ClearDB for Managed MySQL on Bluemix / 201 / URL: https://w2.cleardb.net/quick-start-guide-cleardb-for-managed-mysql-on-bluemix/. Дата звернення: 3.03.2020.

32. Bootstrap forms / 2020 / URL: https://getbootstrap.com/docs/4.0/components/forms/. Дата звернення: 17.04.2020.

ДОДАТОК А

КОД ПРОГРАМИ

mainPage.js

```
import React from 'react';
import { useHistory } from 'react-router-dom';
import {Button, Container, makeStyles, Typography} from "@material-ui/core";
import BannerImage from '../assets/images/test.png'
import {ArrowUpward} from "@material-ui/icons";
import {InfoCard} from "../components/infoCard";
import {CustomButton} from "../components/customButton";
import {paths} from "../shared/routes/paths";
import {AsideBar} from "./asideBar";

const useStyles = makeStyles({
   mainPage:{
      display:"flex",
      flexDirection:"column",
      width:"100%"
   },
   banner:{
      width: '100%',
      height:'500px',
      backgroundImage: `url(${BannerImage})`,
      backgroundPosition: 'top',
      backgroundSize: 'cover',
      backgroundRepeat: 'no-repeat'
   },
   bannerCardWrapper:{
      position:'absolute',
      height: '300px',
      width: '450px',
      margin: '150px 0 0 200px',
      //border: '1px solid black'
   },
   bannerCardSecondary:{
      width: '370px',
      height: '250px',
      backgroundColor: '#FFD7B5',
      borderRadius: '5px'
   },
   bannerCardThirdly:{
      position:'absolute',
      margin: '50px 0 0 40px',
      width: '320px',
      height: '190px',
      backgroundColor: '#D8D8D8',
      borderRadius: '5px'
   },
   mainCard:{
      position: "relative",
      margin: '-220px 0 0 20px',
      padding:'20px',
      width: '350px',
      height: '200px',
      border: '1px solid #451B00',
      borderRadius: '5px'
   },
```

```
bannerCardText:{
    fontSize:'32px',
    fontWeight:'350'
},
aboutUs: {
    paddingTop:"20px"
},
title: {
    width:'300px',
    margin:"0 auto",
    paddingTop:'20px',
    textAlign: "center",
    borderBottom: "1px solid #451B00",
    transform:'translateY(-20px)'
},
cardsContainer:{
    margin:'30px 0 30px 0',
    display:'flex',
    flexFlow:'row wrap',
    alignItems:'center',
    justifyContent:'center'
},
infoWrapper:{
    margin:'20px',
    paddingLeft:'20px',
    paddingTop:'20px',
    width:'480px',
    height:'280px',
    //border:'1px solid red',
    //borderRadius:'5px',
    fontSize:'20px',
    borderLeft:'1px solid #451B00'
},
infoWrapperReverse:{
    textAlign:"right",
    margin:'20px',
    paddingRight:'20px',
    paddingTop:'20px',
    width:'480px',
    height:'280px',
    //border:'1px solid red',
    //borderRadius:'5px',
    fontSize:'20px',
    borderRight:'1px solid #451B00'
},
buttonUp:{
    zIndex:500,
    position:'absolute',
    width:'60px',
    height:'60px',
    backgroundColor:'#FFD7B5',
    borderRadius:'50%',
    fontSize:'30px',
    textAlign: 'center',
    marginLeft:'90%',
    marginTop:'130%'
},
iconUp:{
    color:'#FCF7F4',
    fontSize: 30,
    marginTop:'20%',
    '&:hover': {
        opacity:'0.7',
```

```
            color: '#451B00',
            cursor:'pointer'
        },
    },
    btnDark:{
        width:"150px",
        marginLeft:'48%',
        textTransform:'capitalize',
        fontWeight:'100',
        fontSize: '20px',
        backgroundColor: '#451B00',
        color: "white",
        '&:hover':{
            opacity: 0.8,
            backgroundColor: '#451B00',
            cursor:"pointer"
        },
        border:"none",
        borderRadius:"5px",
        height:"50px"
    }
})

export const MainPage = () => {
    const styles = useStyles();

    const history = useHistory();

    const handleAddTestClick = () => {
        history.push(paths.ADD_TEST);
    }

    return(
            <div className={styles.mainPage}>
                <div className={styles.banner} >
                    <a name="banner"> </a>
                    <div className={styles.bannerCardWrapper}>
                        <div className={styles.bannerCardThirdly}>
                        </div>
                        <div className={styles.bannerCardSecondary}>
                        </div>
                        <div className={styles.mainCard}>
                            <Typography className={styles.bannerCardText} >
                                Онлайн генератор тестів Test On
                            </Typography>
                            <br/>
                            <br/>
                            <button className={styles.btnDark} onClick={handleAddTestClick}>створити тест</button>
                        </div>
                    </div>
                </div>
                <Container>
                    <div className={styles.aboutUs}>
                        <div className={styles.title}>
                            <Typography variant="h4" >
                                About Us
                            </Typography>
                        </div>
                        <div className={styles.cardsContainer}>

                            <InfoCard variant={'first'}/>

                            <div className={styles.infoWrapper}>
```

```
                    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
                </div>

                <div className={styles.infoWrapperReverse}>
                    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
                </div>

                <InfoCard variant={'second'}/>

                <InfoCard variant={'third'}/>

                <div className={styles.infoWrapper}>
                    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
                </div>

                <div className={styles.infoWrapperReverse}>
                    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
                </div>

                <InfoCard variant={'fourth'}/>

                <InfoCard variant={'fifth'}/>

                <div className={styles.infoWrapper}>
                    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
                </div>

                <p className={styles.buttonUp}>
                    <a href="#banner">
                        <ArrowUpward className={styles.iconUp}/>
                    </a>
                </p>
            </div>
        </div>
    </Container>
</div>
  )
}
```

addQuestion.js

```
import React, {useState} from 'react';
import HighlightOffOutlinedIcon from '@material-ui/icons/HighlightOffOutlined';
import AddCircleOutlineOutlinedIcon from '@material-ui/icons/AddCircleOutlineOutlined';
import ErrorOutlineIcon from '@material-ui/icons/ErrorOutline';
import {AsideBar} from "./asideBar";
import {addQuestionValidation, addTestValidation, authValidation} from "../utils/helpers/validation-heplers";
```

```
import {Field, Form, Formik} from "formik";
import {useStylesModal} from "./addQuestionStyles";
import {createQuestion} from "../models/question";
import {useActions} from "../shared/hooks";

export const AddQuestionModal = ({handleSubmitAddQuestion}) => {
   const styles = useStylesModal();

   const [answerType, setType] = useState('one');
   const [answerList, setAnswerList] = useState([]);

   const [addQuestion] = useActions([createQuestion]);

   const handleSubmit = (values) =>{

      const resultValues = {...values, answers: answerList};
      console.log('answers', resultValues.answers);
      switch(values.type){
         case 'one':{
            const responseOptions = resultValues.answers.map(answer => {
               return {
                  responseOption:answer,
                  isValid:resultValues.checked.includes(answer),
               }
            });
            const questionObject = {
               title:resultValues.question,
               responseOptions,
               isOpenQuestion: false,
               isFileQuestion: false,
               grade:resultValues.mark
            }
            addQuestion(questionObject);
            handleSubmitAddQuestion(resultValues);
         }
            break;
         case 'open':{
            const questionObject = {
               title:resultValues.question,
               responseOptions:null,
               isOpenQuestion: true,
               isFileQuestion: false,
               grade:resultValues.mark
            }
            addQuestion(questionObject);
            handleSubmitAddQuestion(resultValues);
         }
            break;
         case 'other':{
            const questionObject = {
               title:resultValues.question,
               responseOptions:null,
               isOpenQuestion: false,
               isFileQuestion: true,
               grade:resultValues.mark
            }
            addQuestion(questionObject);
            handleSubmitAddQuestion(resultValues);
         }
            break;
      }
   }
```

```
const handleCancel = () =>{
   handleSubmitAddQuestion();
}

const handleChangeAnswerType = () =>{
   const value = document.getElementById('type').value;
   setType(value);
}

const handleAddAnswer = () =>{
   let inputContent = document.querySelector("#answer").value;
   const editedArr = [...answerList];
   const checkValue = editedArr.includes(inputContent)
   if(!checkValue && inputContent && inputContent !==''){
      editedArr.push(inputContent);
      setAnswerList(editedArr);
      document.querySelector("#answer").value = null;
   }
}

const handleRemoveAnswer = (id) =>{
   const editedArr = [...answerList];
   const result = editedArr.filter((el, index)=> index !== id);
   setAnswerList(result);
}

return(
   <div className={styles.modal}>
      <div className={styles.modalChild}>
         <Formik
            initialValues={{
               question: '',
               type: '',
               answers: '',
               mark: '',
               checked:[]
            }}
            onSubmit={handleSubmit}
            validationSchema={addQuestionValidation}
         >
            {({
                errors,
                touched,
             }) => (
               <Form className={styles.form} noValidate>
                  <div className={styles.stepsContainer}>
                     <a href="#" name="add-topic"> </a>
                     <h2 className={styles.containerTitle}>Додати питання</h2>
                     <h3 className={styles.stepTitle}>Питання</h3>
                     <Field
                        className={errors.question ? styles.textAreaError : styles.textArea}
                        type="textarea"
                        name="question"
                        id="question"
                        placeholder="Введіть питання..."
                        as="textarea"
                        rows="5" cols="45"
                     />
                     {
                        errors.question ?
                           <div style={{display:"flex", flexDirection:"row", alignItems:"center"}}>
                              <ErrorOutlineIcon  className={styles.errorIcon} />
                              <p className={styles.errorText}>{errors.question}</p>
```

```
                        </div>
                        : null
                    }

                    <h3 className={styles.stepTitle}>Оберіть тип питання</h3>
                    <div className={styles.selectContainer}>
                      <Field
                        as="select"
                        className={errors.type ? styles.selectError : styles.textInput}
                        name="type"
                        id="type"
                        placeholder="Оберіть тип питання..."
                        onClick={handleChangeAnswerType}
                      >
                        <option value="type" >Оберіть тип...</option>
                        <option value="one" >Одна відповідь</option>
                        <option value="open">Відкрита відповідь</option>
                        <option value="other">Документ, фото</option>
                      </Field>
                    </div>
                    {
                      errors.type ?
                        <div style={{display:"flex", flexDirection:"row", alignItems:"center"}}>
                          <ErrorOutlineIcon  className={styles.errorIcon} />
                          <p className={styles.errorText}>{errors.type}</p>
                        </div>
                        : null
                    }

                    {
                      answerType === 'one' ?
                        <>
                          <div className={styles.stepTitleContainer}>
                            <h3 className={styles.stepTitle}>Додати відповідь</h3>
                          </div>
                          <div style={{width:"80%", display:"flex", flexDirection:"row", justifyContent:"space-
between",alignItems:"center"}}>
                            <Field
                              className={errors.answer ? styles.textAreaError : styles.textArea}
                              type="textarea"
                              name="answer"
                              id="answer"
                              placeholder="Введіть відповідь..."
                              as="textarea"
                              rows="5" cols="45"
                              style={{width:"90%"}}
                            />
                            <AddCircleOutlineOutlinedIcon
                              id="add-button"
                              className={styles.icon}
                              onClick={(e) => {handleAddAnswer(e)}}
                            />
                          </div>
                          {
                            errors.answer ?
                              <div style={{display:"flex", flexDirection:"row", alignItems:"center"}}>
                                <ErrorOutlineIcon  className={styles.errorIcon} />
                                <p className={styles.errorText}>{errors.answer}</p>
                              </div>
                              : null
                          }

                          <div
```

```
                                role="group"
                                aria-labelledby="checkbox-group">
                                { answerList && answerList.map((answer, index) => (
                                   <div
                                      key={index}
                                      className={styles.questCard}
                                      style={{width:"80%", height:"auto", margin:"5px 10px"}}
                                   >
                                      <div
                                         className={styles.questHeader}
                                      >
                                         <p className={styles.questionTitle}>{answer}</p>
                                         <div style={{display:"flex", justifyContent:"center"}}>
                                            <Field type="checkbox" name="checked" value={answer} />
                                            <HighlightOffOutlinedIcon className={styles.icon} onClick={() =>
{handleRemoveAnswer(index)}}/>
                                         </div>
                                      </div>
                                   </div>
                                ))}
                             </div>
                          </>
                          : null
                       }

                       <h3 className={styles.stepTitle}>Введіть оцінку</h3>
                       <Field
                          className={errors.mark ? styles.markError : styles.markArea}
                          type="number"
                          name="mark"
                          id="mark"
                          max="100"
                          min="0"
                       />
                       {
                          errors.mark ?
                             <div style={{display:"flex", flexDirection:"row", alignItems:"center"}}>
                                <ErrorOutlineIcon  className={styles.errorIcon} />
                                <p className={styles.errorText}>{errors.mark}</p>
                             </div>
                             : null
                       }

                    </div>

                    <div className={styles.buttonContainer}>
                       <button className={styles.cancelButton} onClick={handleCancel}>Відмінити</button>
                       <button className={styles.submitButton} type="submit">Зберегти</button>
                    </div>
                 </Form>
              )}
           </Formik>
        </div>
     </div>
  )
}
```

addTestPage.js

```
import React, {useEffect, useState} from 'react';
import {Button, Container, makeStyles, Typography} from "@material-ui/core";
import InfoIcon from '@material-ui/icons/Info';
import {CustomButton} from "../components/customButton";
import DoneOutlinedIcon from '@material-ui/icons/DoneOutlined';
import EditOutlinedIcon from '@material-ui/icons/EditOutlined';
import HighlightOffOutlinedIcon from '@material-ui/icons/HighlightOffOutlined';
import AddCircleOutlineOutlinedIcon from '@material-ui/icons/AddCircleOutlineOutlined';
import ErrorOutlineIcon from '@material-ui/icons/ErrorOutline';
import {AsideBar} from "./asideBar";
import {addTestValidation, authValidation} from "../utils/helpers/validation-heplers";
import {Field, Form, Formik} from "formik";
import {useStylesAddTest} from "./addTestPageStyles";
import {AddQuestionModal} from "./addQuestion";
import {clearQuestions, createdQuestionsSelector, removeQuestion} from "../models/question";
import {shallowEqual, useSelector} from "react-redux";
import {useActions} from "../shared/hooks";
import {PreviewPageModal} from "./previewPage";

export const AddTestPage = () =>{
    const styles = useStylesAddTest();

    const initialInfoState = [false, false, false, false, false];
    const [stepInfo, setInfoArr] = useState(initialInfoState);

    const [questionListState, setQuestionList] = useState([]);
    const [expandList, setExpandList] = useState([]);

    const [emailList, setEmailList] = useState([]);

    const [markSum, setMarkSum] = useState(0);

    const [isOpen, setModalState] = useState(false);
    const [isOpenPreview, setModalPreviewState] = useState(false);
    const [submitValue, setSubmitValue] = useState(null);

    let questions = [];

    const { data } = useSelector(createdQuestionsSelector, shallowEqual);

    const [removeQuestionDispatch, clearAllQuestions] = useActions([removeQuestion, clearQuestions]);

    useEffect(() => {
        setQuestionList(data);
    }, []);

    useEffect(() => {
        setQuestionList(data);
    }, [data, questionListState]);

    const handleSubmit = (values) =>{
        const applicants = emailList.map(el =>{
            return { email: el}
        });

        const date = values.dateTime.slice(0, values.dateTime.indexOf('T'));
        const resultDate = date.split('-').reverse().join('.');

        const dateTime = `${resultDate} ${values.dateTime.slice(values.dateTime.indexOf('T')+1,
values.dateTime.length)}:00`;

        const testObject = {
            theme: values.topic,
```

```
        courseId:1,
        questions:data,
        applicants,
        dateTime,
        expireTime: values.timeLimit
    }
    setSubmitValue(testObject);
    setModalPreviewState(prevState => !prevState);
}

const handleSubmitAddQuestion = (values) =>{
    if(values) {
        const resultAnswerList = [...questionListState];
        resultAnswerList.push(values);
        setQuestionList(resultAnswerList);

        const resultExpand = [...expandList];
        resultExpand.push(false)
        setExpandList(resultExpand);

        const mark = markSum + values.mark;
        setMarkSum(mark);
    }
    handleModalDialog();
}

const handleModalDialog = () =>{
    setModalState(prevState => !prevState);
}
const handleModalPreviewDialog = () =>{
    setModalPreviewState(prevState => !prevState);
}

const handleStepState = (stepNumber) =>{
    const editedArr = [...stepInfo];
    editedArr[stepNumber - 1] = !editedArr[stepNumber - 1];
    setInfoArr(editedArr);
}

const handleAnswerClick = (answerNumber) =>{
    const editedArr = [...expandList];
    editedArr[answerNumber] = !editedArr[answerNumber];
    setExpandList(editedArr);
}

const handleAddEmail = () =>{
    let inputContent = document.querySelector("#email").value;
    const editedArr = [...emailList];
    const checkValue = editedArr.includes(inputContent)
    if(!checkValue && inputContent && inputContent !==""){
        editedArr.push(inputContent);
        setEmailList(editedArr);
        document.querySelector("#email").value = null;
    }
}

const handleAddEmailKey = (e) => {
    if (e.keyCode === 13) {
        e.preventDefault();
        handleAddEmail();
    }
}
```

```
const handleRemoveEmail = (id) =>{
      const editedArr = [...emailList];
      const result = editedArr.filter((el, index)=> index !== id);
      setEmailList(result);
}

const handleQuestionRemove = (id) =>{
   removeQuestionDispatch(id);
}

return(
   <div className={styles.testPage}>
     <>
        {
          isOpen ?
                <AddQuestionModal handleSubmitAddQuestion={handleSubmitAddQuestion}/>
            : null
        }
        {
          isOpenPreview ?
             <PreviewPageModal data={submitValue} handlePreviewCancel={handleModalPreviewDialog}/>
             : null
        }
        <AsideBar />
        <Formik
          initialValues={{
             email: '',
             topic: '',
             timeLimit: '',
             dateTime: 0,
             course:''
          }}
          onSubmit={handleSubmit}
          validationSchema={addTestValidation}
        >
          {({
              errors,
              touched,
           }) => (
             <Form className={styles.form} noValidate>
               <div className={styles.stepsContainer}>
                  <a href="#" name="add-topic"> </a>
                  <h2 className={styles.containerTitle}>Додати тест</h2>
                  <h3 className={styles.stepTitle}>1. Додати тему</h3>
                  <Field
                     className={errors.topic ? styles.fieldError : styles.textInput}
                     type="topic"
                     name="topic"
                     id="topic"
                     placeholder="Введіть тему..."
                  />
                  {
                     errors.topic ?
                        <div style={{display:"flex", flexDirection:"row", alignItems:"center"}}>
                           <ErrorOutlineIcon  className={styles.errorIcon} />
                           <p className={styles.errorText}>{errors.topic}</p>
                        </div>
                        : null
                  }

                  <a href="#" name="course"> </a>
                  <div className={styles.stepTitleContainer}>
                     <h3 className={styles.stepTitle}>2. Оберіть курс</h3>
```

```
   <InfoIcon className={styles.icon}
         onClick={()=>{handleStepState(2)}}/>
</div>
{
   stepInfo[1] ?
      <p className={styles.stepInfo}>
         Необхідно додати курс, обравши його з переліку існуючих курсів.
         <br />Ви також можете створити курс, натиснувши кнопку “Додати курс”.
         <br />Ви можете пропустити цей крок, якщо не бажаєте прив’язувати тест до крусу.
      </p>
      : null
}
<div className={styles.selectContainer}>
   <Field
      as="select"
      className={errors.topic ? styles.selectError : styles.textInput}
      name="course"
   >
      <option value="course" >Оберіть курс...</option>
      <option value="course1" >Course Name 1</option>
      <option value="course2">Course Name 2</option>
   </Field>
   <CustomButton textContent={'Створити курс'} variant='light'/>
</div>
{
   errors.course ?
      <div style={{display:"flex", flexDirection:"row", alignItems:"center"}}>
         <ErrorOutlineIcon  className={styles.errorIcon} />
         <p className={styles.errorText}>{errors.course}</p>
      </div>
      : null
}

<a href="#" name="add-question"> </a>
<div className={styles.stepTitleContainer}>
   <h3 className={styles.stepTitle}>3. Додати питання</h3>
   <InfoIcon className={styles.icon}
         onClick={()=>{handleStepState(3)}}
   />
</div>
{
   stepInfo[2] ?
      <p className={styles.stepInfo}>
         Натисніть на кнопку “Додати питання”.
         <br/>Оберіть варіант.
         <br/>Введіть питання.
         <br/>Оберіть тип відповіді.
         <br/>Додайте необхідну кількість відповідей.
         <br/>Оцініть питання.
         <br/>Перевірте, що ваше питання додалося у список.
      </p>
      : null
}
<div style={{width:"71%", display:"flex", flexDirection:"row", justifyContent:"space-between"}}>
   <p className={styles.questionTitle}>Список питань</p>
   <Button
      style={{width:"30%"}}
      className={styles.buttonLight}
      onClick={handleModalDialog}>Додати питання</Button>
</div>
<div className={styles.questContainer}>
   <p className={styles.questionTitle}>Загальна сума балів: {markSum}</p>
   { data ?
```

```
                        data.map((el, index) => (
                            <>
                                <div className={styles.questCard} key={index}>
                                    <div
                                        style={expandList[index] ? {
                                            background: 'rgba(255, 215, 181, 0.37)',
                                            borderRadius: '5px'
                                        } : null}
                                        className={styles.questHeader}
                                        onClick={() => handleAnswerClick(index)}>
                                        <p className={styles.questionTitle} style={{width:"90%"}}>{el.title.length >
100 ? `${index+1}. ${el.title.slice(0,100)}...` : `${index+1}. ${el.title}`}</p>
                                        <div style={{width:"5%"}}>
                                            <HighlightOffOutlinedIcon onClick={() => { handleQuestionRemove(index)}
} className={styles.answerIcon}/>
                                            <EditOutlinedIcon className={styles.answerIcon}/>
                                        </div>
                                    </div>
                                    {
                                        expandList[index] ?
                                            <>
                                                <p className={styles.answerText}>Питання: {el.title}</p>
                                                {
                                                    el.isOpenQuestion?
                                                        <p className={styles.answerText}>Фото відповідь</p>
                                                        : el.isFileQuestion ?
                                                        <p className={styles.answerText}>Файлова відповідь</p>
                                                        :<>
                                                            <p className={styles.answerText}>Варіанти відповідей</p>
                                                            <div>
                                                                {
                                                                    el.responseOptions ?
                                                                        el.responseOptions.map((option, index) =>(
                                                                            <div key={index} className={styles.answerCard}>
                                                                                <p
className={styles.answerTitle}>{option.responseOption}</p>
                                                                                {option.isValid ? <DoneOutlinedIcon
className={styles.answerIcon}/> : null}
                                                                            </div>
                                                                        ))
                                                                        :
                                                                        null
                                                                }
                                                            </div>
                                                        </>

                                                }
                                                <p className={styles.answerText}>Оцінка: {el.grade}</p>
                                            </>
                                            :
                                            null
                                    }
                                </div>
                            </>

                        )
                    )
                    : null
                }
            </div>

            <a href="#" name="date-time"> </a>
            <h3 className={styles.stepTitle}>4. Обрати час та дату</h3>
```

```
                    <Field
                      className= { errors.dateTime ? styles.fieldError : styles.textInput }
                      type="datetime-local"
                      id="dateTime"
                      name="dateTime"
                    />
                    {
                      errors.dateTime ?
                        <div style={{display:"flex", flexDirection:"row", alignItems:"center"}}>
                          <ErrorOutlineIcon  className={styles.errorIcon} />
                          <p className={styles.errorText}>{errors.dateTime}</p>
                        </div>
                        : null
                    }

                    <a href="#" name="time-limit"> </a>
                    <div className={styles.stepTitleContainer}>
                      <h3 className={styles.stepTitle}>5. Додати обмеження часу</h3>
                      <InfoIcon className={styles.icon}
                                onClick={()=>{handleStepState(5)}}/>
                    </div>
                    {
                      stepInfo[4] ?
                        <p className={styles.stepInfo}>
                          Функція надає можливість обмеження часу проведення тесту та виведення таймера
під час тестування.
                        </p>
                        : null
                    }
                    <Field
                      className={errors.timeLimit ? styles.errorTimeLimit : styles.timeLimit}
                      type="number"
                      name="timeLimit"
                      id="timeLimit"
                      min="0"
                      max="200"
                    />
                    {
                      errors.timeLimit ?
                        <div style={{display:"flex", flexDirection:"row", alignItems:"center"}}>
                          <ErrorOutlineIcon  className={styles.errorIcon} />
                          <p className={styles.errorText}>{errors.timeLimit}</p>
                        </div>
                        : null
                    }

                    <a href="#" name="test-subscribe"> </a>
                    <div className={styles.stepTitleContainer}>
                      <h3 className={styles.stepTitle}>6. Поширити тест</h3>
                      <InfoIcon className={styles.icon}
                                onClick={()=>{handleStepState(6)}}/>
                    </div>
                    {
                      stepInfo[5] ?
                        <p className={styles.stepInfo}>
                          Додайте список ел. пошт людей, яким необхідно відкрити доступ до даного тесту.
                          <br/>Лист з часом, темою і датою проведення тесту надійде їм за вказаною поштою.
                        </p>
                        : null
                    }
                    <div style={{width:"53%", display:"flex", flexDirection:"row", justifyContent:"space-
between",alignItems:"center"}}>
                      <Field
```

```
                        id="email"
                        name="email"
                        type="email"
                        className={errors.email ? styles.fieldError : styles.textInput}
                        style={{width:"85%"}}
                        placeholder="Введіть ел.пошту"
                        onKeyUp={(e)=>{handleAddEmailKey(e)}}
                      />
                      <AddCircleOutlineOutlinedIcon
                        id="add-button"
                        className={styles.icon}
                        onClick={(e) => {handleAddEmail(e)}}
                      />
                    </div>
                    {
                      errors.email ?
                        <div style={{display:"flex", flexDirection:"row", alignItems:"center"}}>
                          <ErrorOutlineIcon  className={styles.errorIcon} />
                          <p className={styles.errorText}>{errors.email}</p>
                        </div>
                        : null
                    }
                    {
                      emailList && emailList.map((email, index) => (
                        <div
                          key={index}
                          className={styles.questCard}
                          style={{width:"80%", height:"40px", margin:"5px 10px"}}
                        >
                          <div
                            className={styles.questHeader}
                          >
                            <p className={styles.questionTitle}>{email}</p>
                            <div>
                              <EditOutlinedIcon className={styles.answerIcon}/>
                              <HighlightOffOutlinedIcon className={styles.answerIcon} onClick={() =>
{handleRemoveEmail(index)}}/>
                            </div>
                          </div>
                        </div>
                      ))
                    }
                    <button className={styles.submitButton} type="submit">Зберегти</button>
                  </div>
                </Form>
              )}
            </Formik>
          </>
        </div>
      )
}
```

logIn.js

```
import React, {useState} from 'react';
import { Formik, Field, Form } from 'formik';
import { authValidation } from '../utils/helpers/validation-heplers';

import {Button, Container, makeStyles, Typography} from "@material-ui/core";
import VisibilityIcon from '@material-ui/icons/Visibility';
import {VisibilityOff} from "@material-ui/icons";

const useStyles = makeStyles({
```

```
beigeRect:{
    height:'57%',
    width:'32%',
    position:'absolute',
    marginTop:'4%',
    marginRight:'3%',
    zIndex:'-100',
    backgroundColor:'#FFD7B5',
    borderRadius:'5px',
},
greyRect:{
    height:'55%',
    width:'32%',
    position:'absolute',
    marginTop:'7%',
    marginLeft:'2%',
    zIndex:'-200',
    backgroundColor:'#EDEAEA',
    borderRadius:'5px',
    boxShadow: '0 4px 8px 0 rgba(0, 0, 0, 0.5)',
},
formAuth:{
    width:'30%',
    height:'auto',
    margin:'6%',
    padding:'10px',
    display:'flex',
    flexDirection:'column',
    border:'1px solid #331502',
    borderRadius:'5px'
},
formHeader:{
    margin:0,
    padding: '5px 0px',
    display:'flex',
    flexDirection:'column',
    alignItems:'center',
},
formHeaderTitle:{
    margin:0,
    padding: '10px 0px',
    fontSize:'25px',
    fontWeight:'400'
},
formHeaderBody:{
    display:'flex',
    flexDirection:'row',
    alignItems: 'center',
    borderTop:'1px solid #331502'
},
formHeaderBodyInfo:{
    margin:0,
    padding: '10px 5px',
    fontSize:'18px',
    fontWeight:'300'
},
regLink:{
    textDecoration:'none',
    fontSize:'18px',
    fontWeight:'300'
},
inputContainer:{
    height:'20%',
    marginLeft:'10%',
```

```
    marginTop:'15px',
    fontSize:'18px',
    fontWeight:'400',
    display: 'flex',
    flexDirection:'column',
    alignItems:'flex-start',
},
fieldForm:{
    width:'80%',
    marginTop:'10px',
    paddingLeft:'10px',
    backgroundColor:'#FFD7B5',
    color: 'black',
    lineHeight:'28px',
    fontSize:'16px',
    border: '1px solid #451B00',
    borderRadius: '5px'
},
fieldError:{
    width:'80%',
    marginTop:'10px',
    paddingLeft:'10px',
    backgroundColor:'#FFD7B5',
    color: 'black',
    lineHeight:'28px',
    fontSize:'16px',
    border: '1px solid red',
    borderRadius: '5px',
    '&:focus':{
        borderColor: 'red',
    }
},
errorText:{
    padding:0,
    paddingLeft:'5px',
    margin:0,
    fontSize:'12px',
    color:'red'
},
submitButton:{
    marginLeft:'72%',
    marginTop:'15px',
    width:'100px',
    height:'35px',
    textTransform:'capitalize',
    fontWeight:'100',
    fontSize: '18px',
    backgroundColor: '#451B00',
    color: "white",
    '&:hover':{
        opacity: 0.8,
        backgroundColor: '#451B00',
    }
},
icon:{
    zIndex:500,
    position:'absolute',
    width:'20px',
    height:'60px',
    marginTop: '1.7%',
    marginLeft: '20%',
    fontSize:'30px',
    textAlign: 'center',
    '&:hover':{
```

```
            cursor:'pointer'
        }
    }
});

export const LogIn = () => {
    const styles = useStyles();

    const [visible, setVisibility] = useState(false);

    const handleSubmit = ({email, password}) =>{
        console.log(`Successful LogIn
        password -> ${password}
        email -> ${email}
        `)
    }
    return(
        <>
            <div className={styles.beigeRect}> </div>
            <div className={styles.greyRect}> </div>
            <div className={styles.formAuth}>
                <Formik
                    initialValues={{
                        email: '',
                        password: '',
                    }}
                    onSubmit={handleSubmit}
                    validationSchema={authValidation}
                >
                    {({
                          errors,
                          touched,
                      }) => (
                        <Form noValidate>
                            <div className={styles.formHeader}>
                                <p className={styles.formHeaderTitle}>Вхід</p>
                                <div className={styles.formHeaderBody}>
                                    <p
className={styles.formHeaderBodyInfo}><i>Досі немає аккаунта? </i></p>
                                    <a className={styles.regLink}
href="#">Створити</a>
                                </div>
                            </div>
                            <div className={styles.inputContainer}>
                                <label htmlFor="email">Ел.пошта</label>
                                <Field
                                    type="email"
                                    className={errors.email ? styles.fieldError
: styles.fieldForm}
                                    name="email"
                                    placeholder="Введіть ел.пошту"
                                    id="email"
                                />
                                <p
className={styles.errorText}>{errors.email}</p>
                            </div>
                            <div className={styles.inputContainer}>
                                <label htmlFor="password">Пароль</label>
                                <Field
                                    type={ visible ? 'text' : 'password'}
                                    className={errors.password ?
styles.fieldError : styles.fieldForm}
                                    name="password"
                                    placeholder="Введіть пароль"
```

```
                                    id="password"
                                    autoComplete="on"
                                />
                                {
                                    visible ?
                                    <VisibilityIcon onClick={() =>
setVisibility(prevState => !prevState)} className={styles.icon}/>
                                    : <VisibilityOff onClick={() =>
setVisibility(prevState => !prevState)} className={styles.icon} />
                                }
                                <p
className={styles.errorText}>{errors.password}</p>
                            </div>
                            <div className="d-flex justify-content-center">
                                <Button type="submit"
className={styles.submitButton}>Увійти</Button>
                            </div>
                        </Form>
                    )}
                </Formik>
            </div>
        </>
    )
}
```

registration.js

```
import React, {useState} from 'react';
import { Formik, Field, Form } from 'formik';
import {authValidation, registrationValidation} from '../utils/helpers/validation-heplers';

import {Button, Container, makeStyles, Typography} from "@material-ui/core";
import VisibilityIcon from '@material-ui/icons/Visibility';
import {VisibilityOff} from "@material-ui/icons";

const useStyles = makeStyles({
   beigeRect:{
      height:'70%',
      width:'32%',
      position:'absolute',
      marginTop:'4%',
      marginRight:'3%',
      zIndex:'-200',
      backgroundColor:'#EDEAEA',
      borderRadius:'5px',
   },
   greyRect:{
      height:'70%',
      width:'32%',
      position:'absolute',
      marginTop:'6%',
      marginLeft:'2%',
      zIndex:'-100',
      backgroundColor:'#FFD7B5',
      borderRadius:'5px',
      boxShadow: '0 4px 8px 0 rgba(0, 0, 0, 0.5)',
   },
   formAuth:{
      width:'30%',
      height:'410px',
      margin:'5%',
      padding:'10px',
```

```
    display:'flex',
    flexDirection:'column',
    border:'1px solid #331502',
    borderRadius:'5px'
},
formHeader:{
    margin:0,
    padding: '5px 0px',
    display:'flex',
    flexDirection:'column',
    alignItems:'center',
},
formHeaderTitle:{
    margin:0,
    padding: '5px 0px',
    fontSize:'25px',
    fontWeight:'400'
},
formHeaderBody:{
    display:'flex',
    flexDirection:'row',
    alignItems: 'center',
    borderTop:'1px solid #331502'
},
formHeaderBodyInfo:{
    margin:0,
    padding: '10px 5px',
    fontSize:'18px',
    fontWeight:'300'
},
regLink:{
    textDecoration:'none',
    fontSize:'18px',
    fontWeight:'300'
},
inputContainer:{
    height:'18%',
    marginLeft:'10%',
    marginTop:'10px',
    fontSize:'18px',
    fontWeight:'400',
    display: 'flex',
    flexDirection:'column',
    alignItems:'flex-start',
},
fieldForm:{
    width:'80%',
    marginTop:'10px',
    paddingLeft:'10px',
    backgroundColor:'#FFD7B5',
    color: 'black',
    lineHeight:'28px',
    fontSize:'16px',
    border: '1px solid #451B00',
    borderRadius: '5px'
},
fieldError:{
    width:'80%',
    marginTop:'10px',
    paddingLeft:'10px',
    backgroundColor:'#FFD7B5',
    color: 'black',
    lineHeight:'28px',
```

```
        fontSize:'16px',
        border: '1px solid red',
        borderRadius: '5px',
        '&:focus':{
            borderColor: 'red',
        }
    },
    errorText:{
        padding:'5px',
        margin:'0px',
        fontSize:'12px',
        color:'red'
    },
    submitButton:{
        marginLeft:'72%',
        marginTop:'10px',
        width:'100px',
        height:'35px',
        textTransform:'capitalize',
        fontWeight:'100',
        fontSize: '18px',
        backgroundColor: '#451B00',
        color: "white",
        '&:hover':{
            opacity: 0.8,
            backgroundColor: '#451B00',
        }
    },
    disabledButton:{
        marginLeft:'72%',
        marginTop:'10px',
        width:'100px',
        height:'35px',
        textTransform:'capitalize',
        fontWeight:'200',
        fontSize: '18px',
        backgroundColor: 'darkgrey',
        color: "grey",
    },
    icon:{
        zIndex:500,
        position:'absolute',
        width:'20px',
        height:'60px',
        marginTop: '1.6%',
        marginLeft: '20%',
        fontSize:'30px',
        textAlign: 'center',
        '&:hover':{
            cursor:'pointer'
        }
    }
});

export const Registration = () => {
    const styles = useStyles();

    const [visiblePass, setVisibilityPass] = useState(false);
    const [visibleConfirmPass, setVisibilityConfirmPass] = useState(false);

    const handleSubmit = ({email, password, confirmPassword}) =>{
        console.log(`Successful Registration
        password -> ${password}
```

```
    confirmPassword -> ${confirmPassword}
    email -> ${email}
    `)
  }
  return(
    <>
      <div className={styles.beigeRect}> </div>
      <div className={styles.greyRect}> </div>
      <div className={styles.formAuth}>
        <Formik
          initialValues={{
            email: '',
            password: '',
            confirmPassword:''
          }}
          onSubmit={handleSubmit}
          validationSchema={registrationValidation}
        >
          {({
              errors,
              touched,
            }) => (
            <Form noValidate>
              <div className={styles.formHeader}>
                <p className={styles.formHeaderTitle}>Реєстрація</p>
                <div className={styles.formHeaderBody}>
                  <p className={styles.formHeaderBodyInfo}><i>Вже є аккаунт? </i></p>
                  <a className={styles.regLink} href="#">Увійти</a>
                </div>
              </div>
              <div className={styles.inputContainer}>
                <label htmlFor="email">Ел.пошта</label>
                <Field
                  type="email"
                  className={errors.email ? styles.fieldError : styles.fieldForm}
                  name="email"
                  placeholder="Введіть ел.пошту"
                  id="email"
                />
                <p className={styles.errorText}>{errors.email}</p>
              </div>
              <div className={styles.inputContainer}>
                <label htmlFor="password">Пароль</label>
                <Field
                  type={ visiblePass ? 'text' : 'password'}
                  className={errors.password ? styles.fieldError : styles.fieldForm}
                  name="password"
                  placeholder="Введіть пароль"
                  id="password"
                  autoComplete="on"
                />
                {
                  visiblePass ?
                    <VisibilityIcon onClick={() => setVisibilityPass(prevState => !prevState)}
className={styles.icon}/>
                    : <VisibilityOff onClick={() => setVisibilityPass(prevState => !prevState)}
className={styles.icon} />
                }
                <p className={styles.errorText}>{errors.password}</p>
              </div>
              <div className={styles.inputContainer}>
                <label htmlFor="confirmPassword">Повторіть пароль</label>
                <Field
```

```
                        type={ visibleConfirmPass ? 'text' : 'password'}
                        className={errors.confirmPassword ? styles.fieldError : styles.fieldForm}
                        name="confirmPassword"
                        placeholder="Повторіть пароль"
                        id="confirmPassword"
                    />
                    {
                        visibleConfirmPass ?
                            <VisibilityIcon onClick={() => setVisibilityConfirmPass(prevState => !prevState)}
classNameName={styles.icon}/>
                            : <VisibilityOff onClick={() => setVisibilityConfirmPass(prevState => !prevState)}
classNameName={styles.icon} />
                    }
                    <p className={styles.errorText}>{errors.confirmPassword}</p>
                </div>
                <div>
                    <Button
                        type="submit"
                        className={!!(errors.password || errors.email || errors.confirmPassword) ? styles.disabledButton
: styles.submitButton}
                        disabled={!!(errors.password || errors.email || errors.confirmPassword)}
                    >
                        Створити
                    </Button>
                </div>
            </Form>
        )}
    </Formik>
</div>
</>
)
}
```

actions.js

```
import {
  put, call, takeLatest, all, fork, takeEvery,
} from 'redux-saga/effects';

import * as actionTypes from './types.js';
import { ApiService } from '../../shared/api-service';
import {Cookie} from "../../utils/helpers";

export const login = (currentUser) => ({
  type: actionTypes.LOGIN_REQUESTING,
  payload: {
    currentUser,
  },
});

export const registration = (newUser) => ({
  type: actionTypes.REGIST_REQUSTING,
  payload: {
    newUser,
  },
});

export const clearRegistration = () => ({
  type: actionTypes.CLEAR_LOADED,
});

export const logOut = () => ({
```

```
  type: actionTypes.LOGOUT,
});


function* loginWorker({ payload }) {
  try {
    yield put({ type: actionTypes.LOGIN_STARTED });
    const logUser = yield call(ApiService.create, '/accounts/auth', payload.currentUser);
    logUser.email = payload.currentUser.email;
    yield call(Cookie.set, 'user', JSON.stringify(logUser), 1);
    yield put({ type: actionTypes.LOGIN_SUCCESS, payload: { logUser } });
  } catch (error) {
    yield put({ type: actionTypes.LOGIN_ERROR, payload: { error } });
  }
}

function* logOutWorker() {
  yield call(Cookie.del, 'user');
  yield call(Cookie.del, 'jwt');
  yield put({ type: actionTypes.CLEAR_USER });
}

function* registrationWorker(data) {
  try {
    yield put({ type: actionTypes.REGIST_STARTED });
    const regUser = yield call(ApiService.create, '/accounts/reg', data.payload.newUser);
    yield put({ type: actionTypes.REGIST_SUCCESS, payload: { regUser } });
    // yield put({type: actionTypes.CLEAR_LOADED});
  } catch (error) {
    yield put({ type: actionTypes.REGIST_ERROR, payload: { error } });
  }
}



function* loginWatcher() {
  yield takeLatest(actionTypes.LOGIN_REQUESTING, loginWorker);
}

function* logOutWatcher() {
  yield takeEvery(actionTypes.LOGOUT, logOutWorker);
}

function* registrationWatcher() {
  yield takeLatest(actionTypes.REGIST_REQUSTING, registrationWorker);
}

export function* authWatcher() {
  yield all([
    fork(loginWatcher),
    fork(logOutWatcher),
    fork(registrationWatcher),
  ]);
}
```

validations-helpers.js

```
import * as Yup from 'yup';

export const authValidation = Yup.object().shape({
```

```
  email: Yup.string()
    .required('Обов`язкове поле'),
  password: Yup.string()
    .required('Обов`язкове поле'),
});

export const registrationValidation = Yup.object().shape({
  email: Yup.string()
    .email('Невірна ел.пошта')
    .required('Обов`язкове поле'),
  password: Yup.string()
    .min(8, 'Мінімум 8 символів')
    .matches(
      /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{8,}$/,
      'Мінімум одна велика літера,мала літера та цифра',
    )
    .required('Обов`язкове поле'),
  confirmPassword: Yup.string()
    .oneOf([Yup.ref('password'), null], 'Паролі не збігаються')
    .required('Обов`язкове поле'),
});

export const addTestValidation = Yup.object().shape({
  email: Yup.string()
    .email('Невірна ел.пошта'),
  topic: Yup.string()
    .min(2, 'Мінімум 2 символи')
    .matches('^([A-Za-zA-Яа-яёЁ0-9.#/]([ &_-]|, |: )?)+[A-Za-zA-Яа-яёЁ0-9]+$', 'Invalid course name')
    .max(50, 'Максимум 50 символів')
    .required('Обов`язкове поле'),
  course: Yup.string()
    .required('Обов`язкове поле'),
  dateTime: Yup.string()
    .required('Обов`язкове поле'),
  timeLimit:Yup.number()
    .min(0, 'Мінімум 0 символи')
    .max(200, 'Максимум 200 символів')
});

export const addQuestionValidation = Yup.object().shape({
  question: Yup.string()
    .min(2, 'Мінімум 2 символи')
    .max(500, 'Максимум 500 символів')
    .required('Обов`язкове поле'),
  type: Yup.string()
    .required('Обов`язкове поле'),
  answer: Yup.string()
    .min(2, 'Мінімум 2 символи')
    .max(500, 'Максимум 500 символів'),
  mark: Yup.number()
    .required('Обов`язкове поле'),
});
```

ptotectedRoute.js

```
import React from 'react';
import { Route, Redirect } from 'react-router-dom';
import { useSelector, shallowEqual } from 'react-redux';
import {paths} from "../shared/routes/paths";
import {Cookie} from "../utils/helpers";
```

```
export const ProtectedRoute = ({ roles, ...otherProps }) => {
   const { currentUser } = useSelector(currentUserSelector, shallowEqual);
   const jwt = Cookie.get('jwt');

   if (!jwt) {
      return <Redirect to={paths.AUTH} />;
   }

   if (!roles.includes(currentUser.role)) {
      return <Redirect to={paths.NOT_FOUND} />;
   }
   // eslint-disable-next-line react/jsx-props-no-spreading
   return <Route {...otherProps} />;
};
```

index.js

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0,
minimum-scale=1.0">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap" />
   <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons" />
   <title>WHAT</title>
</head>
<body>
  <div id="root"></div>
</body>
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ВІДГУК
на кваліфікаційну роботу бакалавра
на тему:
«Платформа для генерації і проходження навчальних тестів онлайн. Розробка клієнтської частини.»
студентки групи 121-17-1 Гордієнко А.Ю.

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_Гордієнко.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_Гордієнко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму.
Презентація	
Презентація_Гордієнко.ppt	Презентація ІС.

ВІДГУК

на кваліфікаційну роботу бакалавра
на тему:
«Платформа для генерації і проходження навчальних тестів онлайн. Розробка клієнтської частини.»
студентки групи 121-17-1 Гордієнко А.Ю.

Керівник кваліфікаційної роботи
асистент каф. ПЗКС **С.Д. Приходченко**

РЕЦЕНЗІЯ
на кваліфікаційну роботу бакалавра
на тему:
«Платформа для генерації і проходження навчальних тестів онлайн. Розробка клієнтської частини.»
студентки групи 121-17-1 Гордієнко А.Ю.

Рецензент кваліфікаційної роботи: