НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою “Комп’ютерні системи та мережі”
спеціальності 123 “Комп’ютерна інженерія”

на тему: Веб-застосунок для створення інтерактивних навчальних квестів

Виконав : студент __4__ курсу, групи _ІО-92_
(шифр групи)

_____Іванов Родіон Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____доцент, к.т.н. Ткаченко В. В._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) ст. вик., Виноградов Ю. М._____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент професор кафедри ІСТ, д. т. н., Корнієнко Б. Я._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2023 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“ Комп’ютерні системи та мережі ”

спеціальності 123 “ Комп’ютерна інженерія ”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2023 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Іванова Родіона Олександровича

1. Тема проєкту *Веб-застосунок для створення інтерактивних навчальних квестів.*
керівник проєкту Ткаченко Валентина Василівна, доцент, к.т.н.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 8 червня 2023 року №2101-с

2. Термін здачі студентом закінченого проєкту *8 червня 2023 р.*

3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Огляд існуючих рішень реалізації.

Розділ 2. Огляд технологій для проектування веб-застосунку.

Розділ 3. Моделювання та конструювання програмного забезпечення.

Розділ 4. Візуалізація результатів розробленого веб-застосунку.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) діаграма прецедентів (структурна схема), схема бази даних (функціональна схема), алгоритм дій програмного забезпечення (принципова схема).

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Ст. викладач Виноградов Ю. М.		

7. Дата видачі завдання «15» березня 2023 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*09.03.2023-15.03.2023*	
2.	*Вивчення та аналіз завдання*	*16.03.2023-23.03.2023*	
3.	*Розробка архітектури та загальної структури системи*	*23.03.2023-07.04.2023*	
4.	*Розробка структур окремих підсистем*	*07.04.2023-17.04.2023*	
5.	*Програмна реалізація системи*	*17.04.2023-24.04.2023*	
6.	*Оформлення пояснювальної записки*	*28.04.2023-15.05.2023*	
7.	*Захист програмного продукту*	*25.05.2023*	
8.	*Передзахист*	*05.06.2023*	
9.	*Захист*	*19.06.2023*	

Студент-дипломник _____ Родіон ІВАНОВ
(підпис)

Керівник проєкту _____ Валентина ТКАЧЕНКО
(підпис)

АНОТАЦІЯ

У даній роботі було детально розглянуто навчальні веб-застосунку для проведення квестів та опитувань. Були проаналізовані їхні слабкі та сильні сторони. На основі аналізу було акцентовано увагу на їх особливості, які лягли в основу вирішення задачі проектування веб-застосунку для створення інтерактивних навчальних квестів. В результаті роботи було розроблено веб-застосунок, який вирішує задачу створення квесту та його проведення, досліджено ефективність його роботи. Розроблена програма дає можливість викладачу створювати та проводити змістовні інтерактивні квести та отримувати результати проходження у вигляді ретйингів.

Основна частина документу викладена у пояснювальній записці, виконаній на 105 сторінках, та містить 53 рисунки і 34 таблиці. Також до його змісту входить 4 додатки. Серверне програмне забезпечення написане мовою програмування високого рівня JavaScript та реалізоване за допомогою фреймворку Express.js. Графічний інтерфейс розроблений за допомогою React і Redux.

Ключові слова: інтерактивні квести, створення квестів, навчання, React, JavaScript.

ANNOTATION

In this work, a detailed review of web-based learning applications for quests and surveys was made. Their weaknesses and strengths were analyzed. Based on the analysis, attention was focused on their features, which formed the basis for solving the problem of designing a web application for creating interactive learning quests. As a result of the work, a web application was developed that solves the problem of creating a quest and conducting it, and the effectiveness of its work was investigated. The developed program allows the teacher to create and conduct meaningful interactive quests and receive the results of passing in the form of ratings.

The main part of the document is presented in the explanatory note, which is made on 105 pages and contains 53 figures and 34 tables. It also includes 4 supplements. The server software is written in the high-level programming language JavaScript and implemented using the Express.js framework. The graphical interface is developed using React and Redux.

Keywords: interactive quests, creating quests, learning, React, JavaScript.

справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпля	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467200.002 ТЗ***	Веб-застосунок для	3		
			створення інтерактивних			
			навчальних квестів			
			Технічне завдання			
	А4	***ІАЛЦ.467200.003 ПЗ***	Веб-застосунок для	105		
			створення інтерактивних			
			навчальних квестів			
			Пояснювальна записка			
	А4	***ІАЛЦ.467200.004 Д1***	Веб-застосунок для	1		
			створення інтерактивних			
			навчальних квестів			
			Діаграма прецедентів			
			(структурна схема)			
	А4	***ІАЛЦ.4672008.005 Д2***	Веб-застосунок для	1		
			створення інтерактивних			
			навчальних квестів			
			Схема бази даних			
			(функціональна схема)			
	А4	***ІАЛЦ.467200.006 Д3***	Веб-застосунок для	1		
			створення інтерактивних			
			навчальних квестів			
			Алгоритм дій програмного			
			забезпечення (принципова			
			схема)			
	А4	***ІАЛЦ.467200.007 Д4***	Веб-застосунок для	20		
			створення інтерактивних			
			навчальних квестів			
			Текст програмного коду			

					ІАЛЦ.467200.001 ОА		
Зм	***Лист***	***№ докум.***	***Підп***	***Дата***			
Розроб		Іванов Р. О.			*Веб-застосунок для створення інтерактивних навчальних квестів.* Опис альбому	Літ.	Аркуш 1 / Аркушів 1
Перев		Ткачено В. В.					
						КПІ ім. Ігоря Сікорського, ФІОТ, ІО-92	

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Веб-застосунок для створення інтерактивних навчальних квестів*.»

Київ – 2023

ЗМІСТ

					ІАЛЦ.467200.002 ТЗ		
		№ докум.	Підпис	Дата			
Розробив		Іванов Р. О.			*Веб-застосунок для створення інтерактивних навчальних квестів.* Технічне завдання	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Ткаченко В.В.					
Н. Контр.		Виноградов Ю. М.				***КПІ ім. Ігоря Сікорського, ФІОТ ІО-92***	
Затвердив							

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку веб-застосунку для створення інтерактивних навчальних квестів, а також на подальшу підтримку та вдосконалення розробленого продукту.

Область застосування веб-застосунку для створення інтерактивних навчальних квестів охоплює освітні заклади будь-якого рівня, різні предметні галузі, включаючи науку, мови, математику, літературу, історію, онлайн-курси та дистанційне навчання, тренінги та семінари, самостійне навчання, підготовку до іспитів та оцінювань, корпоративне навчання та навчання працівників.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп’ютерної інженерії», який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного університету України «Київський Політехнічний інститут імені Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки веб-застосунку для створення інтерактивних квестів є забезпечення зручного та ефективного інструменту для викладачів, щоб створювати цікаві та змістовні навчальні квести для учнів чи студентів.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є офіційні документації, публікації та статті в мережі Інтернет на дану тему, науково-технічна література.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена веб-застосунку має виконувати такі вимоги:

- Простий і інтуїтивно-зрозумілий інтерфейс веб-застосунку.
- Забезпечення викладачів функціональною варіативністю для розробки квестів, додаючи мультимедійне наповнення опитувань, налаштування питань, їх типів та визначення балів за складністю.
- Надати можливість користувачам проходити квести та отримувати результати проходження у вигляді рейтингів та оцінки.
- Надати вичерпну та зрозумілу документацію для розробленого продукту.

5.2. Вимоги до програмного забезпечення

- ОС Windows, Mac чи Linux.
- Visual Studio 2017 IDE версії або вище.

5.3. Вимоги до апаратної частини

- ЦП не менше ніж Intel® Core (TM) i3-2100T.
- ROM не менше ніж 64 ГБ.
- RAM не менше ніж 4 ГБ.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	08.03.2023-14.03.2023
Вивчення та аналіз завдання	14.03.2023-23.03.2023
Розробка архітектури та загальної структури системи	23.03.2023-07.04.2023
Розробка структур окремих частин системи	07.04.2023-17.04.2023
Програмна реалізація системи	17.04.2023-24.04.2023
Виправлення помилок	24.04.2023-28.04.2023
Оформлення пояснювальної записки	28.04.2023-15.05.2023

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Веб-застосунок для створення інтерактивних навчальних квестів.*»

Київ – 2023

ПЕРЕЛІК СКОРОЧЕНЬ

HTTP	(HyperText Transfer Protocol) Протокол передачі даних
DOM	(Document Object Model) Специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами
HTML	(HyperText Markup Language) Мова для створення веб-сторінок
CSS	(Cascading Style Sheets) Каскадні таблиці стилів
API	(Application Programming Interface) Програмний інтерфейс
LMS	(Learning Management System) Система управління навчанням
UI	(User Interface) Користувацький інтерфейс
UX	(User Experience) Користувацький досвід
SSR	(Server-Side Rendering) Технологія конвертування HTML-файлів на сервері в повністю відрендерену HTML-сторінку для клієнта
JSON	(JavaScript Object Notation) Текстовий формат обміну даними
MVT	(Model View Template) Шаблон проектування, який розділяє програму на 3 окремі частини: model, view і template
MVC	(Model View Controller) Шаблон проектування, який розділяє програму на 3 окремі частини: model, view і controller
ORM	(Object-Relational Mapping) Створення віртуальних баз даних
JWT	(JSON Web Token) Відкритий стандарт для створення токенів доступу, на основі JSON
SPA	(Single-Page Application) Веб-додаток, який взаємодіє з користувачем, динамічно переписуючи поточну веб-сторінку новими даними з веб-сервера
CSRF	(Cross-Site Request Forgery) Атака, яка обманом змушує жертву надіслати шкідливий запит.
MERN	(MongoDB, Express, React, Node.JS) Стек технологій

ODM	(Object Document Mapping) Об'єкт, пов'язаний з базою даних NoSQL
IDE	(Integrated Development Environment) Інтегроване середовище розробки
SQL	(Structured Query Language) Мова для взаємодії з базами даних

ВСТУП

На сьогоднішній день у освітньому середовищі викладачі та студенти зіштовхнулись з численними викликами, які необхідно вирішувати для забезпечення якісного навчання. Зміна потреб та очікувань учнів, швидкі технологічні зміни та необхідність адаптації до дистанційного навчання — це лише деякі з проблем, з якими зіштовхуються освітяни. Теперішні учні та студенти мають доступ до різноманітної інформації та технологій, і традиційні методи навчання вже не викликають особливо великий інтерес та мотивацію.

У контексті дистанційного навчання та цифрової освіти реалізація веб-застосунку для створення інтерактивних навчальних квестів виявляється дуже актуальною. Розроблений веб-застосунок дозволить викладачам створювати змістовні квести для учнів та студентів, задля активного залучення їх до процесу навчання, при отриманні нових умінь та досягненні результатів. Увага інтерактивних навчальних квестів зосереджена на розв'язанні цікавих завдань, що сприяє запам’ятовуванню та поглибленню знань. Такий веб-застосунок відкриває нові можливості, що спонукають учнів досліджувати, міркувати критично та розвивати свої креативні та аналітичні навички.

У ході роботи будуть виокремлені основні вимоги та особливості веб-застосунків такого типу, досліджені потреби та очікувань учнів, технологічні зміни та особливості дистанційного навчання. Буде проведено аналіз та огляд існуючих рішень та засобів реалізації. Дипломний проєкт має на меті забезпечення ефективного навчання та покращення взаємодію між викладачами та студентами. Спроектований веб-застосунок має бути оснащений спектром можливостей, що допоможуть вчителям і учням досягати поставлених освітніх цілей та поліпшувати якість навчання.

РОЗДІЛ 1.
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ РЕАЛІЗАЦІЇ

1.1 Актуальність та доцільність розробки

Перед початком огляду існуючих рішень реалізації, слід визначити сутність та актуальність веб-застосунку, а також виділити його основні переваги й недоліки в рамках освітнього процесу. Ці критерії стануть в нагоді при детальному аналізі існуючих варіантів та оцінюванні важливості розробки такого застосунку для освітян.

1.1.1 Поняття веб-застосунку

Веб-застосунки – застосунки, в яких програмними модулями визначаються клієнт та сервер, які не вимагають окремого встановлення на операційну систему. Їх запуск здійснюється у веб-середовищі , а взаємодія з користувачем відбувається за допомогою інтерфейсу, створеного клієнтською частиною та доступного через мережу (рис. 1.1).

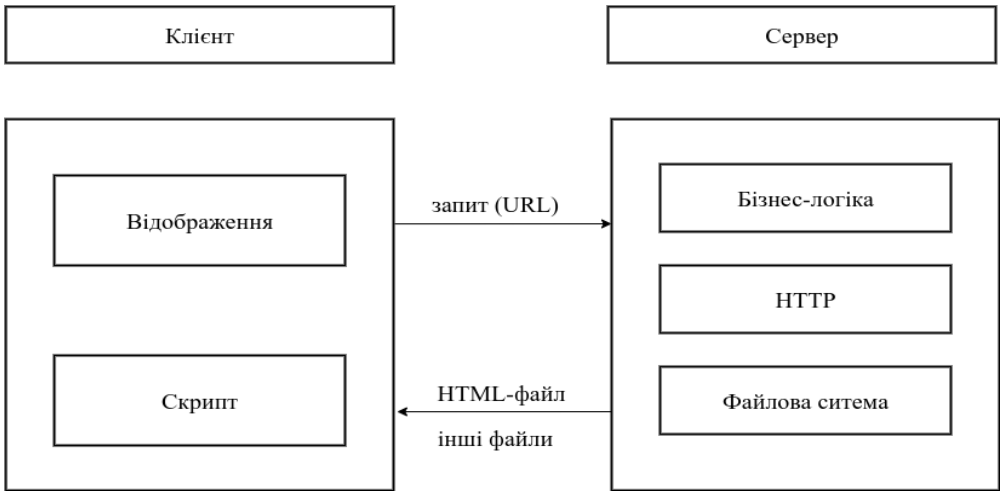


Рисунок 1.1 – Клієнт-серверна архітектура

Браузер в клієнт-серверній архітектурі веб-застосунків є клієнтом , а сервером виступає веб-сервер, що обробляє запити від клієнта через протокол HTTP. Після ініціювання запиту та обчислень веб-сервер дає відповідь на запитувану інформацію та клієнт відображає веб-сторінку [1].

1.1.2 Переваги й недоліки веб-застосунків

Характерною особливістю веб-застосунків є їх кросплатформеність. Як згадано вище, вони можуть бути запущені на різних операційних системах і пристроях та не залежати від їх конфігурацій чи програмного забезпечення. З одного боку це можна вважати перевагою, оскільки створюється можливість для отримання доступу до застосунків будь-де, підключившись до мережі, і використання їх на різних пристроях. Проте це також створює певні проблеми, через те що відображення застосунків здійснюється через веб-браузер, їх інтерфейс може бути обмеженим. Браузери можуть по-різному інтерпретувати вміст веб-сторінок, виходячи з їх специфікації, таких як DOM, HTML, CSS та інших, а також стандартів, налаштувань користувача. Розробникам слід ретельно підходити до цього, вирішувати проблеми сумісності та створювати альтернативні рішення реалізації деякого функціоналу.

Неодмінно суттєвою перевагою веб-застосунків є їх простота оновлення та розгортання. Внесення змін та покращення веб-застосунків може бути проведено на сервері, що дозволить користувачам мати доступ до останніх версій на всіх пристроях без додаткового інсталювання. З цього випливають низькі витрати як на розробку, так і на підтримку веб-застосунків в порівнянні з десктопними додатками, в яких не вийде обмежитись загальновідомими та доступними технологіями та стандартами.

Використовуючи веб-застосунки, користувацькі дані не зберігаються в системі, за рахунок чого, у разі виходу з ладу пристрою, вони не будуть

втрачені. Однак, це супроводжується проблемами з безпекою даних, оскільки відповідальність за користувацькі дані у веб-застосунках несуть розробники та власник веб-застосунку або постачальник хостингу чи хмарних послуг. Тому існує ризик оприлюднення чи ураження персональних даних атаками з використанням міжсайтового скриптування, перехоплення сеансу та іншими атаками у випадку несумлінності відповідальних.

Веб-застосунки забезпечують багатофункціональність, покриваючи різні потреби користувачів. При збільшенні функціоналу з провадженням складних вартісних обчислень, операційна система користувача не перевантажується, за умови, що веб-сервер реалізує більшість з них. Веб-застосунки можливо інтегрувати з іншими сервісами, з такими як сторонні API, соціальні мережі чи платіжні системи.

1.1.3 Актуальність навчальних веб-застосунків

На сьогодні спостерігається зростання впливу веб-технологій на всі сфери людського життя, галузь освіти не є виключенням. Наразі, в умовах дистанційного чи змішаного навчальних процесів, особливу увагу привертають електронні освітні джерела та ресурси. Саме вони надають нових можливостей у створенні та реорганізації інноваційних та ефективних підходів до навчання чи викладання. Розробка веб-застосунків з підтримкою інтерактивних навчальних квестів є рішенням проблем з якими зіштовхуються здобувачі освіти, а саме питання мотивації.

Підставами зацікавленості у освітян впровадженням веб технологій є зручність та простота використання новітніх інструментів для проведення навчальних заходів. Використовуючи даний тип веб-застосунків, можливо істотно підвищити ефективність навчального процесу, активізуючи пізнавальну

та самостійну діяльність учнів, розвинути їх критичне та візуальне мислення, грамотність та сприйняття.

Для підвищення в учнів інтересу до навчання, формуванню навичок пошуку, аналізу та обробки інформації сучасні педагоги здебільшого використовують веб-застосунки для створення та проведення інтерактивних навчальних квестів. Завдяки використанню цих засобів створюється можливість більш акцентовано та цікавіше пояснювати новий матеріал, проводити нестандартні заняття, а також перевіряти знання учнів. Додатково, такі інтерактивні уроки надають перспективи як для індивідуального навчання, так і для командних проєктів, де школярі чи студенти створюють власні квести з тестами та запитаннями, обмінюючись знанням із пройдених матеріалів. Вчителі мають змогу управляти процесом засвоєння матеріалу в режимі реального часу, відстежуючи поточну успішність та прогрес учня протягом курсу, аналізуючи досягнення з метою виявлення слабких місць. Функціонал додавання медіа-файлів, відео, аудіо та зображень до запитань та відповідей квестів не тільки розвантажує викладача, але й пришвидшує процес опанування знань та оцінювання. Викладач виконує функції модератора в індивідуальній освітній траєкторії студента.

Підсумовуючи, існує широке розмаїття функцій, які реалізуються в контексті веб-застосунку для створення інтерактивних навчальних квестів. При використанні таких підходів в асинхронному режимі зросте кількість мотивованих студентів, що готові до самостійної навчальної діяльності, дотримуючись свого власного, гнучкого графіка занять у будь-якому місці з доступом до інтернету, з будь-якого пристрою.

1.2 Огляд існуючих веб-застосунків

Розробка веб-застосунку для створення інтерактивних навчальних квестів реалізується після детального аналізу вимог і функцій існуючих аналогів з метою вибору та запозичення найкращих ідей, рішень. Нижче розглянуто декілька прикладів таких застосунків, описані особливості, дана харакатеристика, визначено їх можливості, а також технології за допомогою яких вони створені.

1.2.1 Kahoot

Kahoot – це ігрова навчальна платформа, що дає можливість легко створювати інтерактивні заняття, поширювати та грати в навчальні квести та вікторини, перевіряючи знання студентів чи учнів в онлайн режимі. Інтерфейс веб-застосунку продемонстровано на рис. 1.2. Застосунок має навчальні ігри, що називаються ‘‘кахутами’’, а формат і кількість запитань в них залежить від уподобань користувача [2].

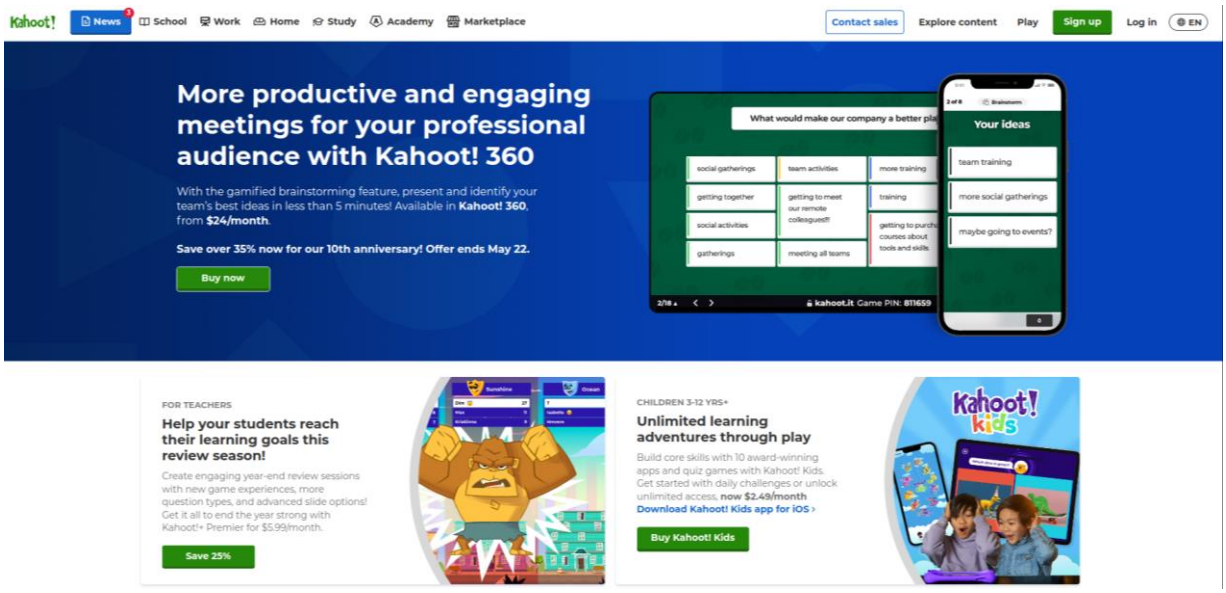


Рисунок 1.2 – Веб-застосунок Kahoot

Після вдалої реєстрації на платформі з вибором типу облікового запису та робочого місця стають доступними власні та публічні опитування. При створенні власного квесту обирається тип запитання, виконується введення тексту запитання і варіантів відповідей з вибором правильного, за необхідності завантажуються медіа дані для зручного та інтерактивного використання. Також вказується проміжок часу, протягом якого учню слід дати відповідь на питання і кількісне оцінювання за правильну відповідь. При проведенні створеного квесту вчителю бажано мати великий екран для транслювання запитань. Учасники приєднуються за унікальним PIN-кодом квесту та проходять опитування на власних пристроях (рис. 1.3).

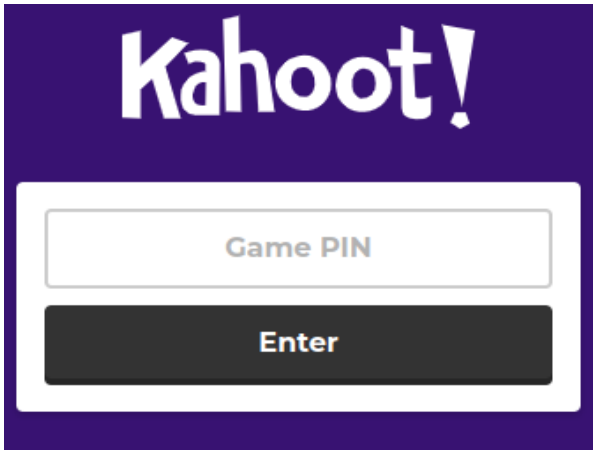


Рисунок 1.3 – Приєднання до квесту у Kahoot

Проте існує можливість асинхронного проведення квестів, що передбачає надсилання квест-викликів, які виконуються в індивільному темпі студента.

Можна виділити наступні особливості:

– велика вибірка типів запитань при створенні квестів, такі як ‘multiple choice’, ‘true/false’ відповіді, відповіді на відкриті питання та інші. Користувач вільний у своєму виборі, тож створення квестів займатиме доволі мало часу;

– за рахунок великої глобальної спільноти вчителів та учнів скорочується час підготовки до заняття, оскільки Kahoot має систему готових ігор, розроблених іншими користувачами даної платформи. Викладачі надають перевагу цьому ресурсу за реалізацію співпраці з колегами по всьому світу, де вони можуть ділитись створеними напрацюваннями та матеріалами;

– вагомою перевагою веб-застосунку Kahoot є аналітика і звітність. Користувачі мають змогу відслідковувати прогрес, переглядати як власну, так і статистику інших користувачів, виявляти слабкі місця, адаптуючи навчальний план;

– реалізація практичної функції Ghost, що дозволяє студентам проходити квести, конкуруючи зі своїми попередніми результатами, перетворюючи покращення зрозумілості матеріалу в захоплююче проведення часу;

Але, попри велику популярність даного веб-застосунку, Kahoot має свої недоліки. По-перше, це обмежені можливості редагування, чим користується автор під час проведення квестів. Відсутність потрібних налаштувань, не реалізован функціонал зміни шрифту та кольору тексту, оформлення кнопок або фону запитань. Служба підтримки також обмежена, не реалізовано онлайн-чат з консультантом, в наявності є лише контакти електронної пошти [3]. Також мінусом Kahoot є перебої в роботі застосунку, що спричинені хакерськими атаками, тож розробники постійно імплементують нові покращення із забезпеченням безпеки, збереженням даних та оптимізації продуктивності роботи. Але найголовнішим недоліком даного веб-застосунку є його висока вартість планів, що обмежує роботу із застосунком, оскільки сильно зменшує кількість доступних функцій та інструментів.

1.2.2 Quizizz

Quizizz - популярний веб-застосунок, що робить дистанційне навчання зручнішим та ефективнішим, надаючи учням та викладачам функції створення контрольних перевірок та домашніх завдань у формі тестів і вікторин. Має широкий вибір існуючих квестів з кожної теми створених користувачами.

Quizizz можливо використовувати як індивідуально, задля вивчення певних тем та набуття кваліфікації, так і в навчальних закладах для аудиторії, однак у веб-застосунку не реалізовано повноцінну інтеграцію з системами управління навчанням (LMS), такими як Google Classroom або Moodle. Інтерфейс веб-застосунку продемонстровано на рис. 1.4.

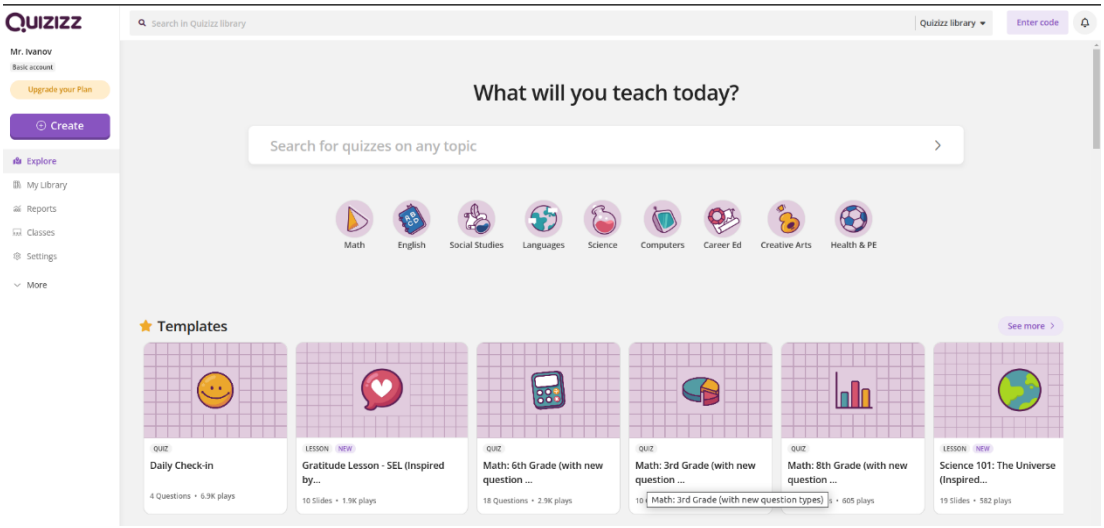


Рисунок 1.4 – Веб-застосунок Quizizz

Як і в Kahoot, квест створює викладач на своєму комп'ютері, використовуючи безліч наданих шаблонів, що надані веб-застосунком (рис. 1.5). Студенти чи школярі мають змогу брати участь в ньому дистанційно за допомогою власних гаджетів.

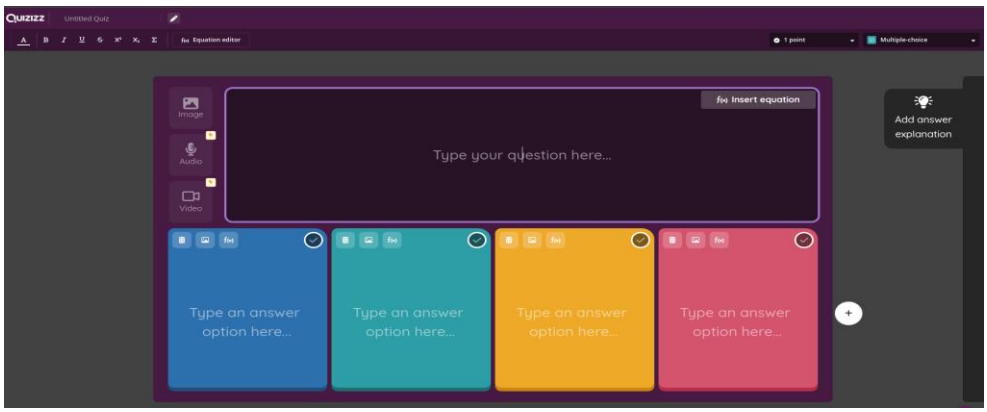


Рисунок 1.5 – Створення квесту у веб-застосунку Quizizz

Виділимо наступні особливості:

– у Quizizz впроваджено детальну систему звітів, що визначає, чи мають учні певні труднощі з опануванням певної теми чи предметної області. Таким чином, викладач може створювати квести з питаннями, на які учасники відповіли незадовільно;

– функція копіювання та вставки окремого питання з квесту іншого користувача у власний з повним збереженням налаштувань квесту дозволяє створювати завдання за лічені хвилини [4];

– перевага даного веб-застосунку полягає у продуманій реалізації оцінювання з можливістю вибору типу. Викладачі можуть заохочувати студентів до швидкого виконання завдання підвищуючи оцінку не тільки за правильність відповіді, але й за спритність, тим самим вносячи зацікавленості та конкуренції до процесу уроку чи лекції;

– швидка система пошуку серед безлічі квестів чи шаблонів на різні теми;

– зручна навігація у веб-застосунку зі зрозумілим та барвистим інтерфейсом сприяє залученню студентів до проходження квестів.

Щодо недоліків, то Quizizz надає користувачам обмежену кількість типів запитань, що присвячений опитуванням, враховуючи питання з кількома правильними відповідями та типами відповідей. Веб-застосунок також не

підтримує вставку в текст запитання медіа-файлів, таких як математичні чи хімічні формули, код. Це приносить великі труднощі у користування для вчителів з цих галузей, що обирають дану платформу для створення квестів. Звичайним недоліком таких застосунків є відсутність онлайн-підтримки. Проте, в порівнянні із Kahoot, Quizizz має цілодобове консультування у Twitter.

1.2.3 Mentimeter

Mentimeter - веб-застосунок для створення інтерактивних презентацій, що підтримують функцію отримання і збереження відповідей учасників конференції в режимі реального часу. Mentimeter дозволяє створювати опитування чи вікторини, що влучно інтегруються в навчальний процес під час лекцій чи контрольних робіт. Окрім вікторин, користувачі також можуть створювати хмари слів чи опитування, що не обмежуються лише однією інтерактивною функцією, але є чудовим інструментом привернути увагу студентів та значно підвищити зацікавленість [5]. Mentimeter - унікальне поєднання традиційних елементів Kahoot-подібних веб-застосунків з інформативними аспектами, що присутні в багатофункціональних презентаціях.

Конструктор презентацій слугує для створення чи імпорту існуючих презентацій з таких застосунків, як Google Slides, PowerPoint чи інші. Створюючи інтерактивні презентації веб-застосунок надає функції додавання різних елементів. Серед інтерактивних елементів що містить Mentimeter можливо виділити реакції, опитування чи хмари слів. Саме за допомогою цього учні зможуть брати участь в інтерактивних заходах під час лекції викладача, висловлюючи свої думки та доносячи цінну інформацію на кожному занятті чи семінарі.

Вибравши тип квесту, редагується його наповнення та налаштування фону квесту, шрифтів, виконується конструювання його макету та інших деталей (рис. 1.6). Доєднання до події здійснюється за рахунок коду, що надсилається аудиторії. Всі відгуки, реакції, опитування зберігаються після закінчення презентації, що в майбутньому можуть покращити досягнення у навчанні у будь-якій освітній та робочій ситуації.

Рисунок 1.6 – Створення квесту у веб-застосунку Mentimeter

Особливості застосунку:

– широкий вибір візуальних інтерактивних ефектів для оформлення презентацій і вікторин. Mentimeter надає право самостійно підключати обрані стилі та анімації, що допомагають створювати зручний та привабливий дизайн;

– впорядкованість та чітка структуризація папок з презентаціями, що відфільтровані за змістом чи правом власності. Є можливість обміну квестами та презентаціями між користувачами;

– насиченість веб-застосунку різноманітністю користувацьких налаштувань. Наприклад, функція фільтрації нецензурної лексики у відповідях

чи реакціях студентів, всі вони будуть виключені і не матимуть змогу попасти на загальну презентацію;

– підтримка інтеграції з платформою Zoom, що забезпечує взаємодію між викладачами та учнями за допомогою відеочату, значно полегшуючи проведення дистанційних занять.

Головний недолік Mentimeter полягає у обмеженості доступного функціоналу при підключенні до безкоштовного плану. Безкоштовна версія зменшує кількість запитань до двох, що обумовлює неможливість повноцінного використання цього веб-застосунку. Інша проблема - це налаштування слайд-шоу Mentimeter, функція зміни порядку слайдів погано реалізована, через що виникають складнощі взаємодії між PowerPoint і Mentimeter при плавному переході до наступного питання.

1.2.4 Майстер-Тест

Майстер-Тест - простий, безкоштовний веб-застосунок, призначений для створення та проведення інтерактивних тестів з метою поліпшення навчального процесу. Має велику базу квестів, що можуть бути запозичені викладачами та пройдені студентами. Проте, зважаючи на безкоштовність даного веб-застосунку, його складно віднести до застосунків з яскравим інтерфейсом та багатим функціоналом. В ньому реалізовані базові функції, що дозволяють проходити інтерактивні опитування.

Майстер-Тест надає можливість вибору ролі при реєстрації, зареєструвавшись як студент, матимете змогу проходити квести, вчителю ж доступні функції додавання груп студентів та створення тестів. Відсутність у веб-застосунку різноманітності користувацьких налаштувань та типів питань стає вагомою проблемою для викладачів.

Під час активізації тесту, необхідно студентам заздалегідь бути зареєстрованими та отримати код-запрошення від викладача. Проходження квестів можна відкладати, вказавши проміжок часу, коли він буде доступний (рис. 1.7). Доступна функція завантаження вмісту квестів [6].

Майстер-тест має наступні особливості:

– веб-застосунок підтримує лише базовий набір типів питань з однією чи кількома правильними відповідями , чисельні та текстові відповіді. В порівнянні з іншими навчальними платформами має обмежену функціональність і не підтримує більш складні та інтерактивні квести;

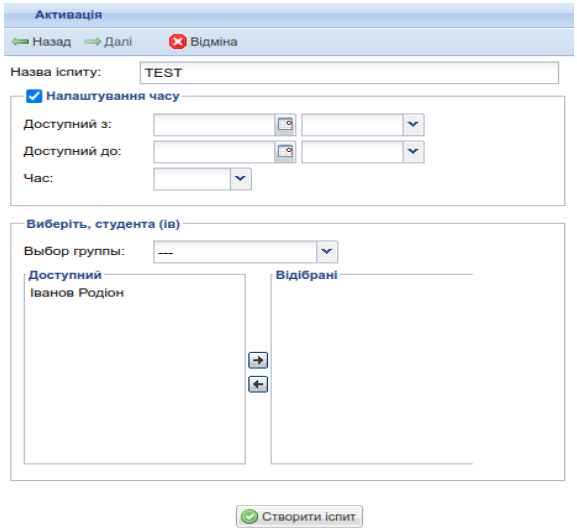


Рисунок 1.7 – Проведення квесту у веб-застосунку Майстер-Тест

– майстер-Тест виконує просту обробку та аналіз відповідей, надаючи викладачу лише оцінку пройдених квестів, без рейтингу студентів та звітності. Таким чином, це унеможливлює автоматизацію обробки результатів та створює додаткове навантаження на користувача.

Але, безкоштовний веб-застосунок Майстер-Тест не є сучасним додатком, тому має застарілий візуальний інтерфейс. Оскільки дизайн веб-застосунку відіграє ключову роль у залученні учнів та студентів до проходження квестів, це може значно зменшити їх зацікавленість та мотивацію

у цьому. Він є менш продуктивними та досить повільним, оскільки не використовує сучасні інструменти та технології для розробки. В Майстер-Тест простежується довготривале завантаження та отримання відповідей на запити користувачів. Більше того, в користувачів відсутня можливість налаштування персоналізованого відображення контенту, пошук квестів з різних сфер навчання займає багато часу і створює проблеми для учнів. Відображення питань під час квестів проводиться строго по порядку, що заданий викладачем заздалегідь, відсутня динамічна зміна питань чи управління логікою тестів.

1.2.5 LearningApps

LearningApps є безкоштовним веб-застосунком, що грунтується на роботі з готовими шаблонами-заготовками для заповнення, призначений для створення інтерактивних завдань різного рівня складності, тематики та формату [7]. LearningApps пропонує користувачам широкий вибір тематик створення завдань, від послідовностей та розподілу, до заповнення пропусків і тестів на знаходження пари. Веб-застосунок доступний на декількох мовах, впровадження підказок та роз'яснень робить користування додатком легким та зручним. LearningApps - комфортний додаток для розробки інтерактивно-дидактичних вправ, що вдало поєднуються при роботі з інтерактивною дошкою. Інтерфейс веб-застосунку продемонстровано на рис. 1.8.

Для створення та проведення квестів викладачу пропонується вибрати серед запропонованих LearningApps шаблонів і завдань для змісту та наповнення квесту. При побудові квесту можна використовувати вправи інших користувачів або створювати власні завдання. Веб-застосунок містить функцію супроводу опитувань за допомогою графічних, звукових або комбінованих ефектів. Підкріплення запитань різноманітними підказками пришвидшує

Рисунок 1.8 – Веб-застосунок LearningApps

процес проходження квестів, реалізована система зворотнього зв'язку з викладачем.

Поширення квесту відбувається за рахунок надсилання QR-коду, який створює веб-застосунок автоматично для кожного квесту, чи вбудови посилання на власному сайті.

LearningApps має наступні особливості:

– LearningApps містить широкий вибір типів вправ, які можна створювати, включаючи пазли, перетягування, заповнення пропусків, сортування, відповіді на зображення і багато інших. Це дозволяє використовувати різноманітні методи навчання та оцінювання;

– зручний конструктор веб-застосунку LearningApps має інтуїтивно зрозумілий графічний інтерфейс, який дозволяє створювати вправи без необхідності дослідження інструкцій;

– особливістю даного веб-застосунку також є запозичення окремих , вже існуючих завдань інших користувачів для прикладу розроблення

інтерактивної вправи, що підходить під користувацькі потреби. Модифікація готових шаблонів квестів LearningApps імплементує інноваційні та ефективні підходи у навчання чи викладання.

Попри можливість створення безлічі квестів, що наповнені різноманітними завданнями, індивідуально адаптувати LearningApps під власні навчальні потреби не вийде. Веб-застосунок не містить функцій додавання оформлення фонів, шрифтів чи полів для введення, вони є загальними. Відсутність суперництва та комунікацій між учнями залишається проблемою LearningApps, оскільки тут не реалізовано обговорення та коментування відповідей, спілкування в середині груп, відстежування рейтингів, тощо. Застарілий дизайн також є мінусом даного веб-застосунку.

1.3 Порівняльний аналіз розглянутих веб-застосунків

Після огляду існуючих веб-застосунків для створення інтерактивних навчальних квестів слід провести аналіз для визначення найкращих рішень реалізації. Головними критеріями обрано легкість використання та інтуїтивність інтерфейсу, функціональність та різноманітність типів запитань, оцінювання і зворотній зв'язок. Прослідковується залежність між такими характеристиками як вартість та наповненість функціональністю веб-застосунка.

Таблиця 1.1. Вибір найкращих застосунків за критеріями

Критерій/Застосунок	Kahoot	Quizizz	Mentimeter	Майстер-Тест	LearningApps
Дизайн		+	+		
Зручність		+			+
Онлайн-підтримка			+		

Кінець таблиці 1.1

Критерій/Застосунок	Kahoot	Quizizz	Mentimeter	Майстер-Тест	LearningApps
Типи квестів	+		+		+
База квестів	+	+			+
Вартість		+		+	+
Система звітів	+	+			
Оцінювання	+	+			
Аналітика	+	+			
Інтегрованість	+	+	+		

Найкращим серед розглянутих веб-застосунків виявився Quizizz, що надає користувачам можливість повноцінно користуватись його послугами, обравши в налаштування безкоштовний план. Саме перважна більшість його рішень стануть основою для розробки веб застосунку згідно теми дипломного проєкту.

1.4 Проблематика розроблення веб-застосунку

Ціллю веб-застосунку для створення інтерактивних квестів є забезпечення зручного та ефективного інструменту для викладачів, щоб створювати цікаві та змістовні навчальні квести для учнів чи студентів. Він надає можливості для навчання, спілкування, оцінювання та обміну ресурсами між вчителями та учнями. Веб-застосунок може бути спеціалізованим для певного предметного поля або загальним, що охоплює різні аспекти навчання.

Легкість використання має полягати у простому та інтуїтивно зрозумілому інтерфейсі, щоб викладачі з різним рівнем технічної підготовки

могли легко створювати свої квести без особливих труднощів. Користувачі мають швидко опановувати основні функції для створення квестів, їх відображення та налаштування. Дизайн веб-застосунку має бути спроектований таким чином, аби викладач чи студент могли вирішити певну проблему за мінімальну кількість кроків і логічних послідовностей. Реалізація адаптивності дизайну має на меті забезпечити оптимальне використання веб-застосунку на різних пристроях та екранах з різними характеристиками.

Розроблений веб застосунок повинен забезпечувати користувачів функціональною варіативністю для розробки квестів, додаючи мультимедійне наповнення опитувань, налаштування питань, їх типів та визначення балів за складністю. В залежності від обраної ролі, користувачу надаються певні права, студенти мають можливість проходити квести та відслідковувати свій прогрес, викладачі створювати та активізовувати ці квести. Отримання відгуків на квести покращить взаємодію між учнями та вчителями і допоможе удосконалити навчальний процес.

Однією з важливих особливостей веб-застосунку для створення інтерактивних квестів є оцінювання. За допомогою нього, викладачі можуть визначити прогрес студентів у навчанні та досягнення мети інтерактивного квесту. Після проходження тестів, оцінювання повинно відображатись у вигляді підсумкової кількості набраних балів та проценту опанування матеріалу. Для виявлення слабких та сильних місць учнів веб-застосунок має надавати звітність у вигляді переліку запитань, на які були дані неправильні відповіді. Як підсумок, ті, хто навчаються можуть зрозуміти на які аспекти необхідно звернути більшу увагу та які теми потребують повторного проходження. Саме це дозволяє персоналізувати навчання та забезпечити адаптацію навчального процесу в залежності від успіхів конкретного студента.

1.4.1 Характерні особливості та складові

Для оптимізація процесу створення квестів та ефективного використання функціональності слід розуміти основні складові інтерактивних навчальних квестів. Складова квесту - це окремий компонент, що входить до склада структури квесту. Кожна складова має свою функцію, сприяє досягненню навчальних цілей та оптимізації процесу створення квестів.

– контролер. В якості контролеру виступає викладач, що демонструє створений квест на головному єкрані. Він має доступ до конструктора квесту, де додає питання чи завдання, налаштовуючи параметри. Має можливість переглядати оцінки та рейтинг учасників;

– учасники. Студенти та учні що проходять завдання чи опитування на власних пристроях. Учасники можуть працювати окремо або у команді, залежно від концепції квесту;

– навігація. Важливою складовою є інтерактивність квесту, яка дозволяє учасникам взаємодіяти з веб-застосунком. Навігація забезпечує логічний та зручний перехід між питаннями та задачами;

– результат. Результат відображається після кожної окремої задачі чи питання та показує наскільки успішно учасник їх виконав. Учасники отримують відповідні повідомлення в залежності від правильності їх відповідей. З результатів учасників формується рейтинг;

– оцінка. Під час проходження тесту веб-застосунок веде облік успішності участника та формує загальну кількісну оцінку та висновок засвоєння матеріалу в процентному відношенні;

– рейтинг. Рейтинг використовується для порівняння успішності учасників квесту. Демонстрація місця учасника у загальному рейтингу або порівняння з іншими учасниками, що тількино завершили тест;

ВИСНОВКИ ДО РОЗДІЛУ 1

В даному розділі було проведено огляд існуючих рішень реалізації, що мають попит серед освітян. В порівняльному аналізі розглянутих веб-застосунків представлено їх основні особливості та недоліки. Виявлено найкращі технічні рішення, що будуть враховані при розробці веб-застосунку. Обґрунтовано ціль та актуальність розробки навчального веб-застосунку. Надано проблематику реалізації та характерні складові.

Завданням дипломного проєкту буде розробка веб-застосунку з підтримкою створення та проведення інтерактивних навчальних квестів.

Функціональність застосунку зі зручним користувацьким інтерфейсом має на меті вирішення проблем з якими зіштовхуються здобувачі освіти. Необхідно:

– дослідити та обрати технології для проектування інтерфейсів веб-застосунку, серверної та клієнтської частин;

– розробити користувацький інтерфейс веб-застосунку, забезпечуючи зручну навігацію, можливість створення, редагування та видалення квестів з налаштуванням питань, задач та додаткових параметрів;

– розробити алгоритми та структури даних для створення та проведення інтерактивних квестів, включаючи обробку запитів користувача, додавання питань та задач, збереження та відображення результатів;

– розробити документацію проєкту, що містить аналіз вимог, огляд основних функцій та можливостей веб-застосунку, компонентів, архітектуру системи, результати розробки та тестування.

РОЗДІЛ 2.
ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ ПРОЕКТУВАННЯ ВЕБ-ЗАСТОСУНКУ

Перед оглядом технологій та інструментів реалізації проектування веб-застосунку для створення навчальних інтерактивних квестів необхідно визначити частини та етапи розробки. Для таких веб-застосунків притаманним є шаблон архітектури з розділеним стеком, що полягає у розмежуванні фронтенд і бекенд розробки (рис. 2.1). Кожна з цих частин функціонує незалежно та взаємодіють одна з одною через API. Вони мають свої особливості та відповідають за різні аспекти функціонування веб-застосунку. Основними перевагами цього підходу є відсутність обмеження вибору технологій та одночасність розробки кожної з частин.

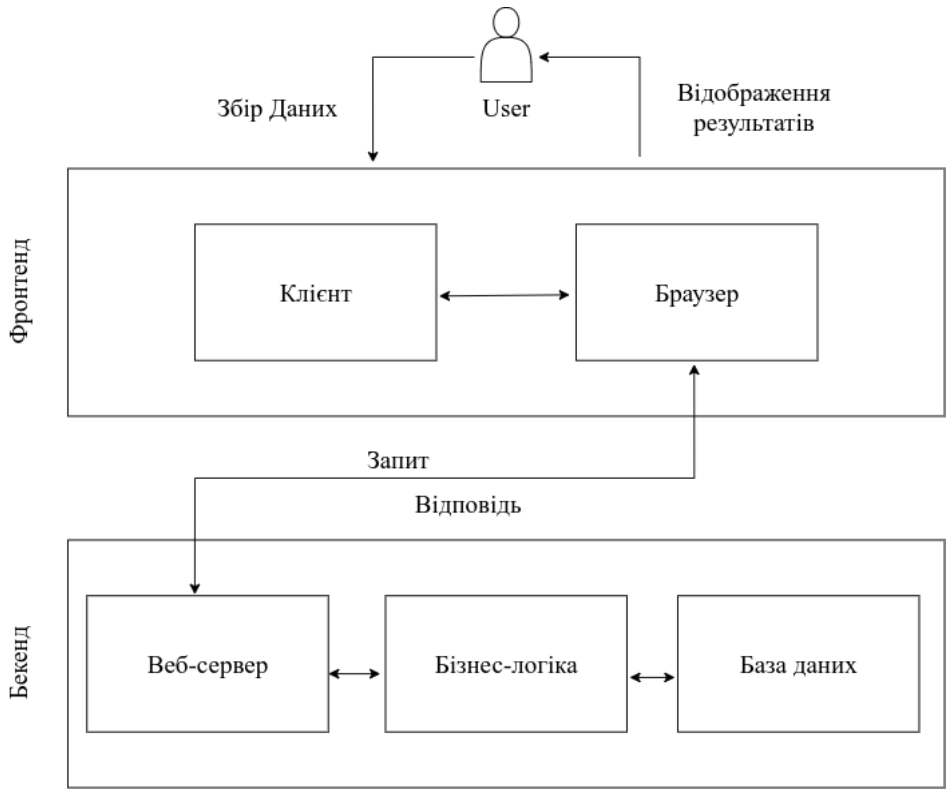


Рисунок 2.1 – Базова структура веб-застосунку

Серверна частина веб-застосунку відповідає за обробку запитів користувача, взаємодію з базою даних та іншими зовнішніми сервісами. Користувачі не мають прямого доступу до внутрішньої реалізації і бізнес-логіки веб-застосунку. Бекенд веб-застосунку може бути фізично розташований в дата-центрі або хоститися на хмарних платформах, що віддаленні від користувачів.

Клієнтська частина взаємодіє з користувачем безпосередньо у браузері та відповідає за структуру, оформлення та інтерактивність інтерфейсу. Фронтенд компонент обробляє запити користувачів, після чого відправляє їх на сервер. Важливою функцією цієї частини є коректне відображення даних, що надсилаються сервером у відповідь. Гарно спроектований та зручний інтерфейс забезпечує залучення нових користувачів та їх позитивний користувацький досвід.

2.1 Дизайн та проектування інтерфейсів веб-застосунку

UI та UX дизайн є двома важливими аспектами проектування веб-застосунків та інших цифрових продуктів. Вони спрямовані на покращення взаємодії та створення ефективного та зручного користування цифровими продуктами.

Інтерфейс користувача (UI) - це система і користувач, які взаємодіють один з одним за допомогою команд або методів для керування системою, введення даних та використання вмісту. Інтерфейси користувача варіюються від систем, таких як комп'ютери та мобільні пристрої до прикладних програм і використання контенту. [8] UI дизайн відповідає за вигляд та візуальний аспект інтерфейсу, що включає розташування елементів, використання кольорів, шрифтів та графічних елементів. Він має на меті розробку привабливого,

легкозапам'ятовуваного та зрозумілого інтерфейсу, що полегшує взаємодію з користувачем.

Користувацький Досвід (UX) відноситься до загального досвіду користувача, пов'язаного зі сприйняттям та реакцією, які він відчуває коли користується продуктом , системою чи контентом. Досвід користувача описує складність під час використання функціоналу конкретного інтерфейсу за яким звернувся користувач. Також він включає в себе розуміння потреб та цілей користувачів, проектування напрямків роботи, навігації та інтерактивності. UX дизайну ставить метою реалізацію програмного продукту, який є логічним, зручним та задовольняє потреби користувачів, покращуючи їх враження.

Загалом, UI/UX - це інтерфейс за допомогою якого людина може взаємодіяти з системою або застосунком у комп'ютерному та комунікаційному середовищі. Ці два процеси є взаємопов'язаними і доповнюють один одного.

При відсутності зв’язків та роботою над обома частинами інтерфейсу, результатом виконання стане зовнішньо гарний, однак незручний інтерфейс, або багатофункціональний та корисний застосунок, що буде візуально застарілим чи не привабливим. Тому, гармонійне поєднання зрозумілого та естетичного вигляду інтерфейсу (UI) зі зручністю використання та задоволенням користувача (UX) сприяє покращенню якості продукту та його успіху на ринку.

2.2 Технології для розробки клієнтської частини веб застосунків

Розробка клієнтської частини веб-застосунку включає створення та налаштування всіх елементів з якими взаємодіє користувач на веб-сторінках.

При реалізації клієнтської частини веб-застосунку використовуються три основні компоненти: HTML, CSS та мова програмування JavaScript:

HTML означає мову гіпертекстової розмітки, що використовує систему розмітки, яка складається з елементів для представлення конкретного змісту. Розмітка означає створення структури та організації вмісту веб-сторінки за допомогою визначених тегів, а не те , як відбуваються ці процеси. HTML не є мовою програмування, оскільки вона , як мова розмітки, не має вбудованих можливостей для виконання логічних операцій, керування потоком виконання і зміненням стану програми. Елементи HTML є основними компонентами, з яких складається структура веб-сторінки. Назви елементів можна розглядати як описові ключові слова для вмісту, який вони містять, наприклад відео, аудіо чи таблиця. Сторінка HTML може складатися з потенційно сотень елементів, які потім читаються та інтерпретуються веб-браузером, перетворюючи розмітку на зрозуміле відображення на екрані [9]. HTML теги можливо розділити на категорії в залежності від їх призначення. Наприклад, структурні теги <html>, <head>, <body> використовуються для визначення логічної структури веб-сторінки, а теги <p>, , , для форматування та представлення тексту.

CSS - це мова опису стилів, що використовується для оформлення веб-сторінок. Вона визначає зовнішній вигляд і форматування елементів HTML документу, додаючи шрифти, розміри, кольори, межі, відступи, фонові зображення та інше. CSS розглядає кожен елемент HTML так, ніби він знаходиться всередині у власному блоці і використовує правила, щоб вказати, як має відображатися вміст визначених елементів. CSS дозволяє створювати правила, які контролюють спосіб представлення кожного окремого блоку та його вмісту [10]. Чіткий розподіл обов'язків між розміткою і стилем може забезпечити численні переваги у продуктивності розробки. CSS правила складаються з селекторів та декларацій . Селектори визначають елементи, до яких застосовується правило, а в деклараціях зазначено принцип відображення цих елементів та їх вигляд. Декларації описуються за допомогою властивості та значення, які розділяються двокрапкою. Використовуючи різні типи селекторів

можливо точно вибрати та стилізувати певні елементи HTML , тим самим забезпечити гнучкість та контроль над стилізацією. Важливою особливістю CSS є каскадування, що дає змогу визначити, яким стилям застосовуватись у разі конфлікту між правилами.

JavaScript - це мова програмування з підтримкою імперативного, функціонального та подієво-орієнтованого стилів. Від початку ідея створення полягала в тому, що основні інтерактивні частини клієнтської частини веб мали бути реалізовані на мові Java, а JavaScript мав стати мовою-клеєм для цих частин, а також зробити HTML трохи більш інтерактивним [11]. Наразі JavaScript використовується як для веб-розробки, так і для серверної частини. Він застосовується для динамічного змінення вмісту сторінок, маніпуляції DOM, взаємодії з користувачем через обробку форм та подій, виконання асинхронних запитів до сервера, тощо. JavaScript код інтерпретується та виконується JavaScript-движком браузера, після того як код HTML і CSS був відображений у веб-сторінку, тобто структура й стилі будуть сформовані до цього моменту. За допомогою платформи Node.js, JavaScript також має можливість виконуватись на серверній стороні, що дозволяє створювати повноцінні веб-сервери, API, мікросервіси та інше.

На сьогодні, технології для розробки клієнтської частини веб-застосунків на цьому не обмежуються. Для спрощення та прискорення процесу розробки використовуються фреймворки та бібліотеки, що надають багато корисних функціональних можливостей та інструментів. Вони включають в себе готові компоненти і шаблони, мають оптимізовані алгоритми та методи для ефективної обробки даних, управління станом, рендерингу та інших операцій.

2.2.1 React

React – JavaScript бібліотека, яка змінила підхід до розробки клієнтської частини веб-застосунків, здобувши популярність та широке використання.

React має відкритий вихідний код, розроблений суспільством самостійних програмістів та компанією Meta. Бібліотека покликана вирішити проблему створення інтерактивних та швидких інтерфейсів користувача, використовуючи концепцію компонентів. React має суттєвий вплив на розробку односторінкових веб-застосунків та допомагає позбутись труднощів, пов'язаних з їх розробкою, за рахунок впровадження технології часткового оновлення компонентів веб-сторінок.

React дотримується принципу декларативності, який полягає у описі різних частин інтерфейсу у кожному стані веб-застосунку, React визначає які компоненти потрібно оновити чи відрендерити при зміні даних. Декларативний підхід відрізняється від імперативного, в якому послідовно вказуються інструкції яким чином оновлювати DOM для досягнення певного стану. React приховує безпосередню маніпуляцію з DOM та дозволяє фокусуватися на описі результату, що робить код більш передбачуваним і обумовлює простоту налагоджування (рис. 2.2).

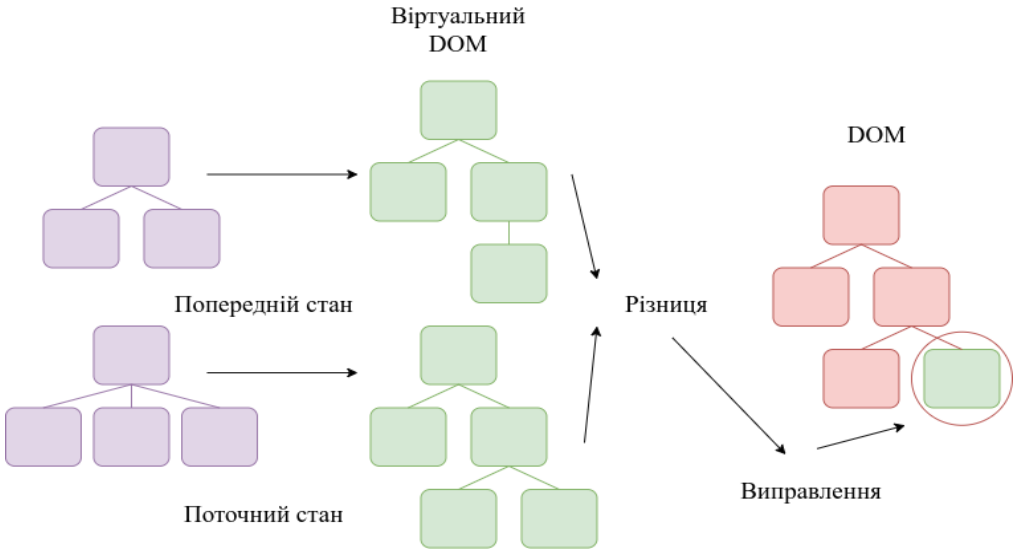


Рисунок 2.2 – Принцип роботи Virtual DOM

Концепція Virtual DOM в React є проміжним шаром між компонентами React і реальним DOM, що забезпечує оптимізацію оновлення та відображення

елементів інтерфейсу. В основу покладено створення віртуального представлення дерева елементів, що відображає поточний стан. Це дерево елементів схоже на реальний DOM, але існує тільки в пам'яті. При зміні стану компонентів, React порівнює новий Virtual DOM з попереднім станом та шукає елементи, що потребують оновлення. Вбудований у React алгоритм порівняння визначає набір змін, необхідних для оновлення реального DOM, щоб відповідати новому стану. Реалізація віртуального DOM у React містить важливі оптимізації продуктивності, які роблять її дуже швидкою. Віртуальний DOM використовує ефективні алгоритми диференціювання для пошуку змін, одночасно пакетно оновлює піддерево DOM [12]. Все це забезпечує зниження навантаження на браузер та оптимізований спосіб створення односторінкових веб-додатків.

React використовує односторонній потік даних, переміщуючи дані лише від батьківських компонентів до дочірніх. Властивості передається вниз через пропси (props). Компонент не має можливості змінювати напряму властивості, але здійснює їх модифікації, передаючи вгору функцію зворотного виклику. При зміні даних, React автоматично оновлює залежні від цих даних компоненти (рис. 2.3).

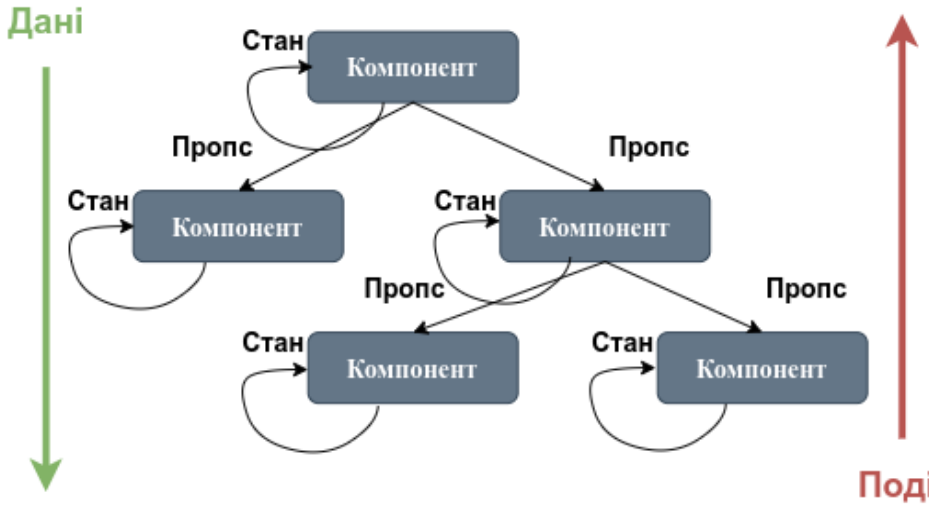


Рисунок 2.3 – Потік даних у React

Однією з головних переваг React є можливість повторного використання компонентів, що дає змогу використовувати компоненти з різними функціональностями у будь-яких частинах веб-застосунку або навіть в інших проєктах. Повторне використання компонентів забезпечує модульність веб-застосунку та дозволяє розбити складний інтерфейс на менші, керовані компоненти. Такий підхід також забезпечує спрощення процесу тестування, оскільки компоненти відокремленні один від одного.

React має широкий вибір інструментів та розширень, які полегшують розробку, налаштування та тестування додатків на React. Для полегшення розробки та розширення можливостей і функціональності використовуються, наприклад, для керування станом додатку - Redux , а для налагодження маршрутизації - React Router. React можливо поєднувати з бібліотеками компонентів Material-UI чи Bootstrap. React також сумісний з різними серверними технологіями, такими як платформа Node.js або фреймворками для створення API.

2.2.2 Vue

Vue.js - це фреймворк для розробки веб-інтерфейсів на JavaScript, який використовує шаблон MVVM для створення інтерактивних інтерфейсів користувача.

Одна з ключових особливостей Vue полягає в його реактивному зв'язуванні даних, що дозволяє з легкістю керувати та оновлювати їх відображення. Коли дані, пов'язані з відображенням, змінюються, Vue автоматично виявляє ці зміни і оновлює всі елементи, які від них залежать. Такий декларативний підхід забезпечує створення динамічних інтерфейсів та ефективне керування станом веб-застосунку за рахунок уникнення непотрібних маніпуляцій з DOM.

Використовуючи синтаксис шаблонів, подібний до HTML, Vue дозволяє декларативно зв'язати дані з DOM елементами веб-сторінки. Важливою перевагою Vue є те, що його шаблони є стандартними HTML-кодом, який можна легко розпізнати HTML парсерами браузеру. Vue займається внутрішньою компіляцією цих шаблонів в функції рендерингу для віртуального DOM, що реалізує ефективне оновлення веб-сторінки при зміні даних (рис. 2.4).

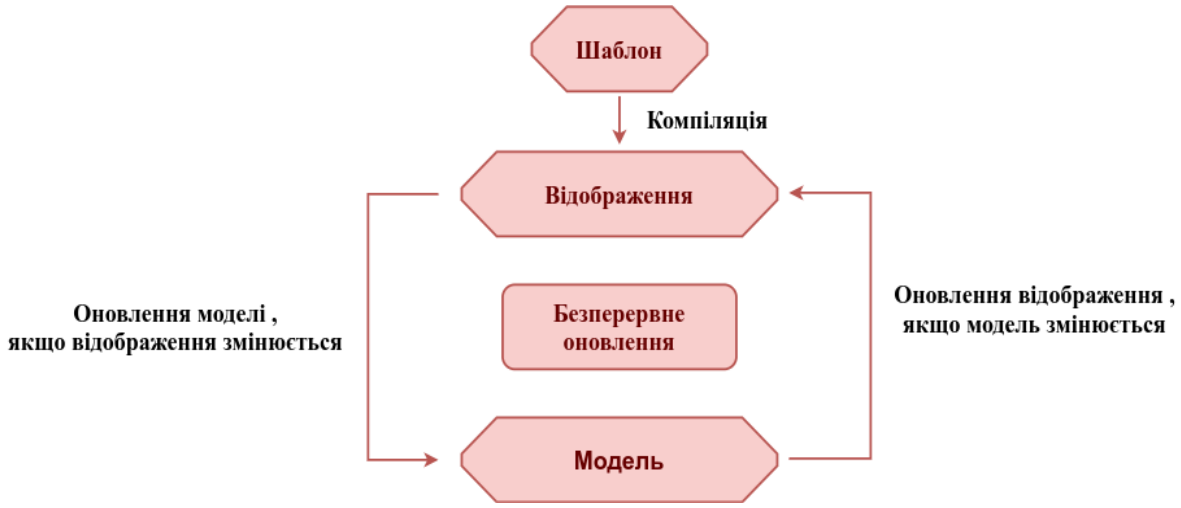


Рисунок 2.4 – Двостороннє зв’язування даних

Vue.js містить двостороннє зв'язування даних - механізм, що автоматично синхронізує дані між моделлю та відображенням. За допомогою цього механізму, зміна значення в моделі спричиняє оновлення відповідного відображення, і навпаки, зміна значення відображення автоматично оновлює модель. Двостороннє зв'язування даних працює автоматично за допомогою директиви v-model, яка дозволяє зв'язувати дані між елементом введення (input) з відповідною властивістю даних в компоненті. Коли користувач змінює значення в елементі введення, відповідне значення властивості даних також оновлюється, і навпаки. Ця технологія надає можливість швидко та просто реагувати на зміни користувача та автоматично оновлювати відображення

даних. Такий підхід спрощує роботу з формами та взаємодію з користувачем, оскільки не потрібно вручну слідкувати за змінами значень та оновлювати їх.

Для веб-застосунків на Vue існує патерн централізованого управління станом. Місце чи сховище, де зберігаються дані додатку, тобто стан, є серцем реалізації Vuex. Цей стан безпосередньо ніколи не може бути змінений , а лише за допомогою виклику мутацій. Всередині компонентів виконуються дії, що відповідають за виклик мутацій (рис. 2.5).

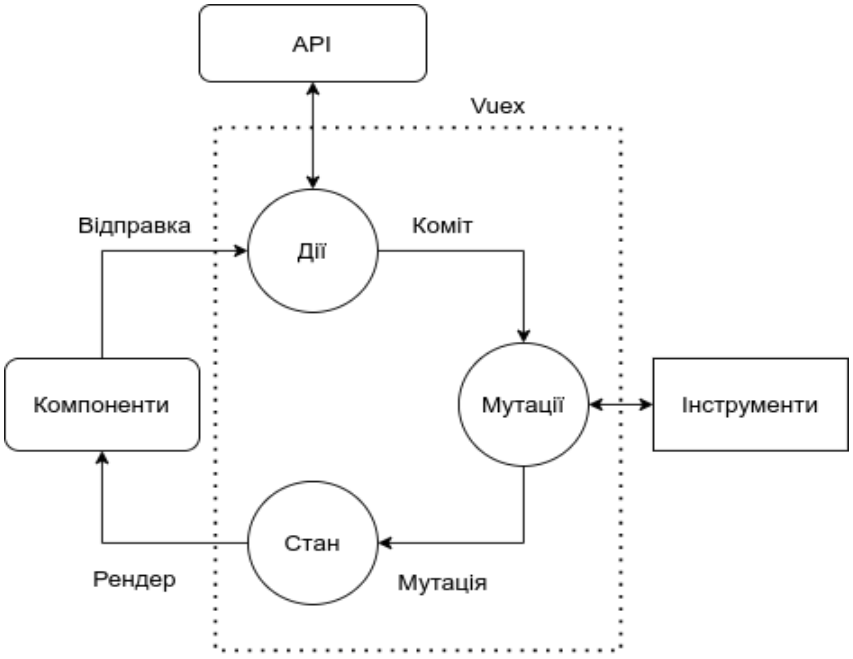


Рисунок 2.5 - Принцип роботи Vuex

Сховище Vuex також дозволяє нам визначати гетери - методи, які передбачають отримання обчислених даних про стан [13]. Vue може бути використаний для розробки як невеликих, так і масштабних веб-додатків, в порівнянні з React або Angular, він може мати обмеження в масштабованості для деяких великих проєктів. Це пов'язано з тим, що багато плагінів та інструментів, розроблених для Vue.js, можуть не мати широкої підтримки та сумісності з іншими фреймворками.

2.2.3 Angular

Angular – фреймворк для розробки веб-додатків, був розроблений компанією Google і вперше випущений як AngularJS. Проте з часом AngularJS був перероблений і вдосконалений, і з'явився Angular - повністю переписаний фреймворк. Будучи написаним на TypeScript, Angular надає переваги сильної типізації, підказок під час розробки. Це дозволяє розробникам писати більш структурований і безпечний код, зменшує кількість помилок і полегшує розробку та підтримку додатків.

В порівнянні з React, структура Angular є значно складнішою. Окрім використання компонентів як основних блоків додатків, він містить багато елементів зі своїм призначенням. Наприклад, класи, що застосовуються як користувацькі атрибути елементів розмітки називаються директивами, а для централізованого зберігання даних Angular надає такі інструменти як сервіси. Для групування зазначених елементів за функціоналом чи сторінками призначені модулі.

Angular підтримує набір інструментів Angular Universal, який дає змогу виконувати рендеринг веб-застосунків на стороні сервера (рис. 2.6). SSR дозволяє генерувати сторінку на сервері і надіслати її до клієнтської частини, що прискорює відображення і покращує продуктивність.

Використовуючи цей різновид рендерингу, браузер отримує відповідь після структуризації шаблонів на клієнтській стороні, позбуваючись додаткових запитів.

Angular має ієрархічну систему впровадження залежностей. Ін'єкція залежностей - це патерн проектування програмного забезпечення, який намагається зменшити кількість тісно пов'язаних компонентів шляхом зміни способу обробки залежностей між компонентами [14]. Це означає, що залежності компонентів вирішуються індивідуально, що сприяє зменшенню зайвого оновлення та перетворення компонентів, що не потребують змін.

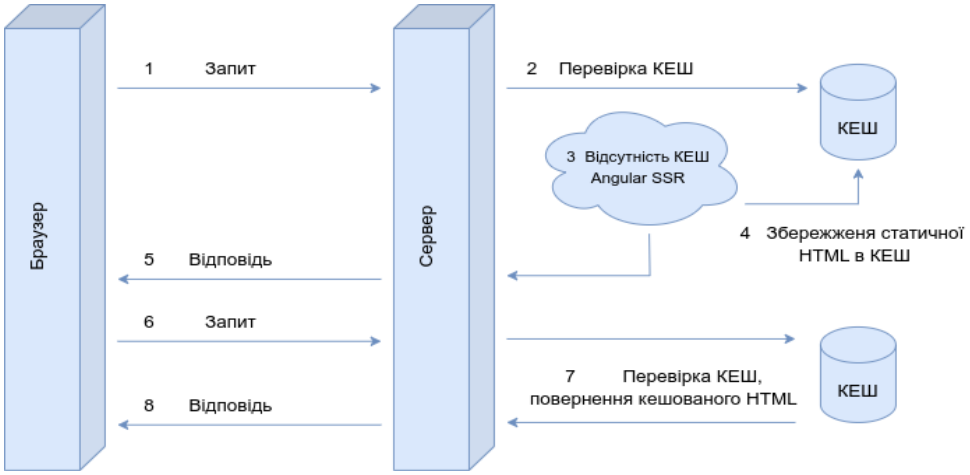


Рисунок 2.6 – Angular SSR

Також підвищити швидкодію допомагає компілятор та рушій шаблонів Ivy. Основна мета полягає в тому, щоб зменшити розмір та поліпшити продуктивність генерованого коду Angular веб-застосунків. Ivy прискорює час компіляції Angular додатків завдяки впровадженню прикладного компілювання. Вона полягає у генерації лише того коду, який потрібно оновити, замість повного перекомпілювання всього додатку.

Підсумовуючи, Angular має складну архітектуру та широкий спектр функціональності , багато концепцій та термінології, таких як модулі, компоненти, сервіси, директиви тощо. Складність обумовлена його спрямованістю на великомасштабні проєкти. Все це може бути викликом для розробників, які мають обмежений досвід з фреймворками. Тому Angular обирають компанії з великою командою розробників, що вже тривалий час використовують TypeScript.

2.3 Технології для розробки серверної частини веб-застосунків

Розробка серверної частини веб-застосунку включає обробку запитів користувачів, взаємодію з базою даних та бізнес-логікою додатку. Існує багато технологій, які можна використовувати для розробки серверної частини веб-застосунків.

2.3.1 Express.js

Express.js є високопродуктивним та мінімалістичним фреймворком на мові програмування JavaScript. Ця технологія, заснована на платформі Node.js та надає розробникам потужний інструментарій для побудови ефективних та масштабованих веб-застосунків та API. Він дозволяє створювати серверну частину веб-додатків, забезпечуючи зручний механізм обробки маршрутів та взаємодію з базою даних.

Express починає свою роботу зі створення об'єкта app, який представляє веб-додаток. Для створення об'єкта app використовується функція express(), яка є твірним методом Express. Цей об'єкт включає в себе набір вбудованих функцій та методів, які використовуються для налаштування та обробки запитів.

Проміжне програмне забезпечення middleware є однією з ключових концепцій в Express.js. Воно є проміжним шаром між сервером і додатком, представляючи собою функції, які обробляють HTTP-запити та відповіді, може бути використане для обробки логіки аутентифікації, перехоплення помилок, перетворення даних, перевірки безпеки та багатьох інших задач (рис. 2.7). У Express.js проміжне програмне забезпечення може бути додане за допомогою методу app.use() або HTTP методів, наприклад, app.get(), app.post(), тощо.

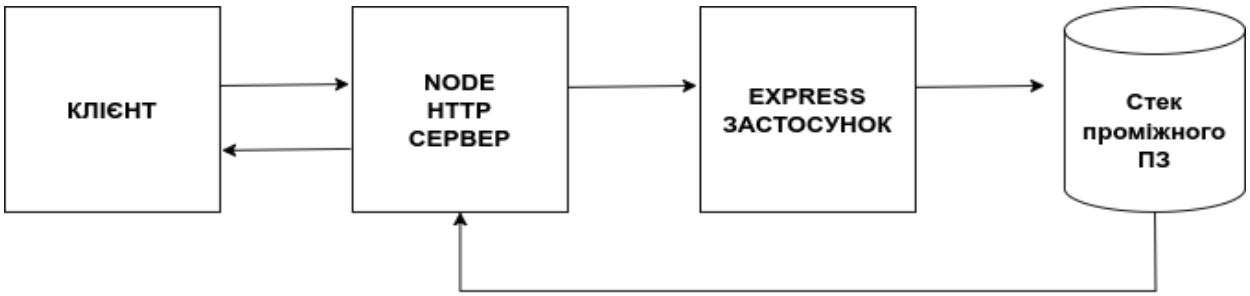


Рисунок 2.7 – Взаємодія Node.js та Express.js

Можна використовувати вбудовані middleware, такі як express.static для обробки статичних файлів або express.json для обробки даних у форматі JSON. Виконання відбувається в порядку додавання запитів, передаючи черговому проміжному програмному забезпеченню об'єкт request (запит), об'єкт response (відповідь) та функцію next, яка вказує на перехід до наступного проміжного програмного забезпечення. Можна використовувати вбудовані middleware, такі як express.static для обробки статичних файлів або express.json для обробки даних у форматі JSON. Цей підхід забезпечує гнучкість та розширюваність веб-додатків, дозволяючи розробникам легко додавати та налаштовувати middleware залежно від потреб проєкту.

Маршрутизація, будучи одним із найважливіших аспектів веб-застосунку, в Express є простою, гнучкою та надійною. Маршрутизація - це механізм, за допомогою якого запити, визначені URL-адресою і HTTP-методом, перенаправляються до коду, який їх обробляє.[15] У Express.js роутинг відбувається за допомогою методів маршрутизатора, які пов'язані з екземпляром додатка. Основні методи маршрутизатора включають GET, POST, PUT та DELETE, що дозволяє розробникам легко взаємодіяти з клієнтом.

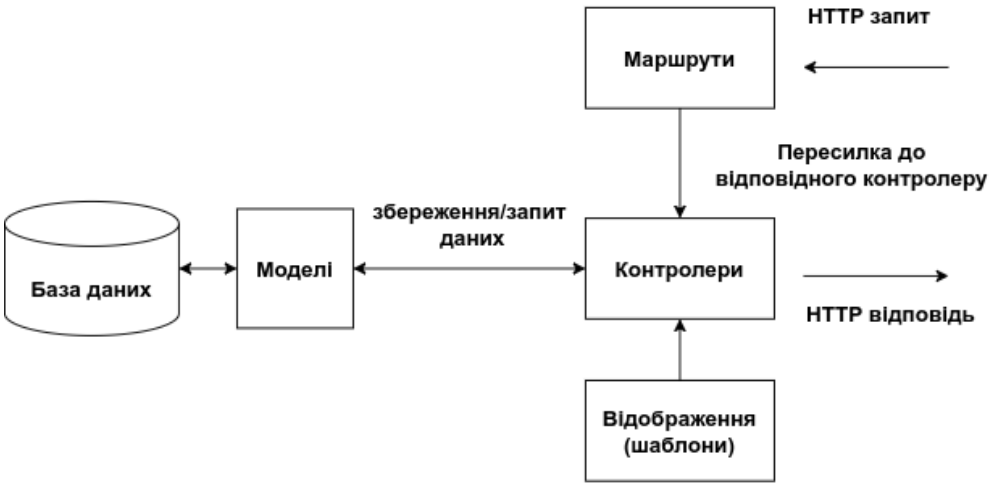


Рисунок 2.8 – Взаємодія компонентів Express веб-застосунків

Express.js не накладає суворих правил та стандартів щодо структури проєкту. Це може ускладнити розуміння та підтримку, оскільки розробка здійснюється враховуючи різні підходи та стилі. Використання Express.js дозволяє легко інтегрувати його з різними фронтенд-технологіями, такими як Angular, React або Vue.js. Це дозволяє створювати повноцінні веб-додатки з розподіленою архітектурою клієнт-сервер.

2.3.2 Django

Django є відкритим фреймворком веб-розробки, розробленим на мові програмування Python. Він надає комплексний набір інструментів і бібліотек, які допомагають швидко розробляти високоякісні веб-додатки і забезпечувати ефективну роботу з базами даних, маршрутизацією URL-адрес, керуванням сеансами, автентифікацією користувачів, тощо. Django заснований на MVT(Model-View-Template) патерні, який є альтернативою MVC , саме це дозволяє легко розділити бізнес-логіку та представлення даних (рис 2.9).

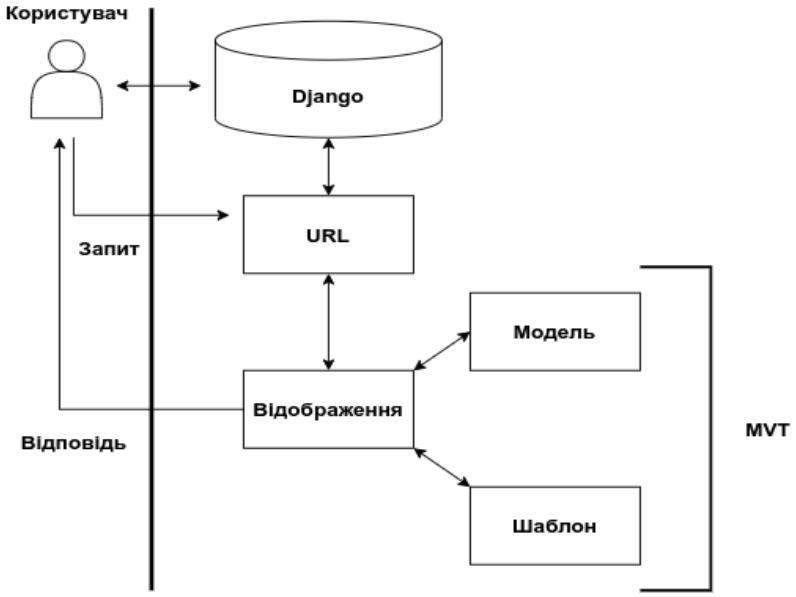


Рисунок 2.9 – Архітектура MVT

Основна відмінність між MVT і MVC полягає у розміщенні логіки обробки даних. У MVC, логіка контролера відокремлена від представлення, тоді як у MVT, бізнес-логіка може бути розміщена в моделі або у представленні.

Моделі в Django використовуються для визначення структури та поведінки даних в базі , зв'язків між ними. Модель представляє окремий об'єкт або концепцію, яка зберігається у базі даних і з якою можна взаємодіяти. Вони використовують ORM (Object-Relational Mapping) , що дозволяє працювати з даними без прямого використання SQL-запитів, що спрощує розробку та підтримку веб-застосунків.

View відповідає за обробку HTTP-запитів та відправку відповідей. Він отримує дані від моделей, виконує потрібні операції та передає дані в шаблони для відображення. У view також можна використовувати шаблони для генерації вихідного HTML-коду. Шаблони дозволяють розділити логіку обробки даних від їх представлення.

Django має вбудовану адміністративну панель, яка надає зручний інтерфейс та побудована на основі моделей, що дає змогу до автоматичного адаптування структури даних в базі даних. Розробники можуть налаштовувати

відображення полів, фільтри, пошукові можливості та інші аспекти адміністративного інтерфейсу за допомогою спеціальних класів моделей та параметрів. Це дозволяє розробникам швидко створювати адміністративні інтерфейси без необхідності вручну розробляти адмін-панель.

Підтримка безпеки є ще однією важливою особливістю Django. Пакет django.contrib.csrf захищає від Cross-Site Request Forgery (CSRF) – "перехоплення сеансу", що є вразливістю в безпеці веб сайтів. CsrfMiddleware змінює вихідні запити, додаючи до всіх POST-форм приховане поле з назвою «csrfmiddlewaretoken» і значення, яке є хешем ідентифікатора сеансу та секретного ключа [16].

Для веб-застосунків з простою функціональністю та невеликим обсягом даних, використання Django може здатися зайвим або надмірним через багату кількість його розширених можливостей. Також, веб-застосунки Django можуть мати обмежену швидкодію порівняно з іншими легковаговими фреймворками, оскільки для ефективної роботи вимагають достатню кількість пам'яті та обчислювальних ресурсів сервера.

2.4 Вибір інструментів і технологій для реалізації веб-застосунку

Після проведення огляду та аналізу інструментів і технологій клієнтської та серверної частин, дослідження їх основних особливостей та недоліків було обрано наступні рішення реалізації веб-застосунку для створення інтерактивних навчальних квестів.

Бібліотека React буде використовуватись для створення Front-end частини. Причиною вибору став її широкий спектр бази рішень та розповсюдженість застосування в реальних задачах та проєктах, порівнюючи з Angular і Vue. React дає більшу свободу в організації коду та більш гнучкі можливості при виборі додаткових бібліотек та інструментів. Бібліотека є

досить швидкою, завдяки реалізації React Virtual DOM і різним оптимізаціям рендерингу, її архітектура дозволяє ефективно управляти станом та оновлювати компоненти за необхідності.

Фреймворк Express.js на платформі Node.js буде використовуватись для створення Back-end частини. Виходячи з того, що Express.js до стеку технологій MERN, це робить розробку простішою, оскільки немає потреби опановувати різні мови програмування, всі технології використовують JavaScript. Express.js є досить швидким фреймворком, оскільки базується на Node.js, який працює на одному потоці та здатний ефективно обробляти велику кількість одночасних з'єднань. Продуктивність роботи розробленого веб-застосунку не буде зменшуватись, оскільки Node.js буде обробляти лише легковагові запити, що не потребують тривалого опрацювання даних та складних обчислень.

2.5 Огляд додаткових інструментів розробки

2.5.1 Redux

Для синхронізації станів у різних частинах веб-застосунку використовується бібліотека управління станом Redux. Вона організовує роботу компонентів зі складною логікою та взаємодією, забезпечує послідовність поведінки та спрощує тестування JavaScript веб-застосунків. Redux надає загальний підхід до управління станом додатків, незалежно від того, який фреймворк або бібліотека використовується для розробки.

Основна мета Redux полягає в зберіганні стану веб-застосунку у централізованому контейнері сторі (store), з яким компоненти можуть взаємодіяти, отримуючи чи оновлюючи його. Flux-архітектура є концептуальною основою для розробки Redux. Він є патерном архітектури, який виник з метою керування потоком даних у додатках з одним напрямком.

Redux використовує цей патерн, але зі значними модифікаціями та поліпшеннями.

Для здійснення змін в стані веб-застосунку створюються дії. Дії є об'єктами, які містять тип дії і необов'язкові дані, які використовуються для оновлення стану. За передачу дій до редукторів відповідає dispatcher . Він є посередником між діями, які створюються в додатку, і редюсерами, які змінюють стан на основі цих дій. На основі отриманих дій та стану, редуктори (reducers) визначають, які зміни мають бути проведені в стані. Після виконання редукторів і отримання нового стану, стор оновлюється з цим новим станом. Для запобігання недоцільного повторного рендеренгу Redux при зміні стану повертає новий, що дає змогу виявити зміни в стані та оновити компоненти, які підписані на нього. Таким чином, компоненти оновлюють свої візуальні представлення відповідно до нового стану (рис.2.10).

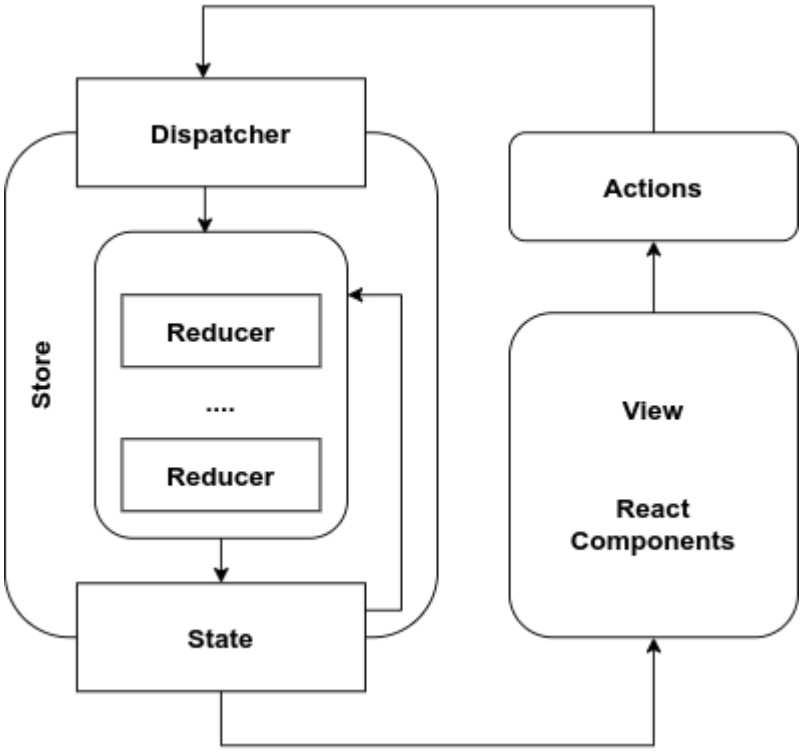


Рисунок 2.10 – Архітектура Redux

Однією з головних переваг Redux є відокремлення управління станом від шарів представлення та логіки. Завдяки такому розподілу обов'язків, дизайн макету стану можна робити окремо від дизайну інтерфейсу користувача і будь-яких складних логічних потоків.[17] Redux зберігає всі дії, які відбуваються в додатку, та їх послідовного відтворення, що дає змогу ефективно аналізувати та розуміти роботу додатка, здійснювати відлагодження, тестування та вдосконалення. Redux має суворі рекомендації щодо оформлення коду, що забезпечує його читабельність та прискорює роботу над проєктом.

2.5.2 Material UI

Material UI - це відкрита бібліотека компонентів, яка базується на "Material Design" - дизайн-системі, що розроблена компанією Google, яка визначає стандарти та принципи розробки інтерфейсів для веб-сайтів та застосунків. Вона надає готові стилізовані компоненти і інструменти для швидкої і простої розробки інтерактивних елементів інтерфейсу.

Material UI надає можливість налаштування теми додатка, що дозволяє змінювати кольори та інші стилізовані параметри компонентів відповідно до вимог проекту. Провайдер теми (ThemeProvider) містить функції налаштування вигляду та поведінки всіх компонентів Material-UI в межах веб-застосунку. Він приймає об'єкт теми, який містить різні значення для кольорів, шрифтів, розмірів та інших стилів. Для застосування провайдеру слід огорнути ваш компонент або його відповідну частину провайдером теми і передати об'єкт теми як його властивість.

Material-UI пропонує компоненти для навігації по сторінкам або розділам продукту, компоненти для розміщення та організації вмісту на екрані, компоненти дій. Бібліотека має компоненти, що дозволяють користувачам вводити дані або здійснювати вибір, компоненти для сповіщення користувачів

про важливу інформацію або повідомлення. Налаштування компонентів здійснюється шляхом передачі пропсів, змінюючи стиль, кольори, розміри та інші властивості. Також Material-UI містить велику кількість векторних іконок, що доступні після імпортування.

Material UI є бібліотекою компонентів, яка надає розширену документацію для встановлення, використання, оновлення та налаштування компонентів. Ця документація містить якісні пояснення та приклади коду, які допомагають розробникам швидко оволодіти бібліотекою і ефективно використовувати її у своїх проєктах.

2.5.3 MongoDB

MongoDB є NoSQL базою даних, одним із ключових принципів якої є документо-орієнтована модель, де дані зберігаються у вигляді JSON-подібних документів. Це дає можливість зберігати структуровані дані без потреби задавати жорстку схему.

Колекції є контейнерами для групування пов'язаних документів в MongoDB. Якщо документ у MongoDB є аналогом рядка в реляційній базі даних, то колекцію можна вважати аналогом таблиці. Колекції не мають схем. Це означає, що документи в межах однієї колекції можуть мати будь-яку кількість різних форм [18].

Особливістю MongoDB є використання запитів та операцій для маніпулювання даними у документах. Наприклад, ви можете додавати нові документи у колекцію, оновлювати існуючі, видаляти документи або виконувати складні запити для пошуку та фільтрації даних. MongoDB надає потужну мову запитів, яка дозволяє здійснювати різноманітні операції над документами та колекціями.

MongoDB дешевше та простіше в обслуговуванні, ніж реляційні бази даних, оскільки має простіший розподіл даних, моделі даних та автоматичне

відновлення. Підтримка горизонтально розширення забезпечує комфортне масштабування, додаючи більше машин до набору ресурсів.

Веб-розробка є однією з основних сфер використання MongoDB. Її гнучкість та зручність роблять її ідеальним вибором для розробки веб-застосунків різної складності. В поєднанні з Express.js утворюють потужну комбінацію для створення надійних та масштабованих серверних API чи веб-застосунків.

2.5.4 Mongoose

Mongoose - це бібліотека об'єктного моделювання даних (ODM) для роботи з MongoDB у середовищі Node.js, що дозволяє зручно та ефективно взаємодіяти з базою даних, використовуючи моделі, схеми та запити. У Mongoose схеми і моделі використовуються для визначення структури та поведінки документів, які будуть зберігатися в MongoDB.

Схема визначає структуру документа MongoDB, включаючи поля, типи даних, валідацію, побудову індексів та інші властивості. Для цього використовується конструктор mongoose.Schema(). Mongoose підтримує визначення різних типів даних, таких як рядки, числа, дати, масиви та об'єкти. У схемі також прописані правила валідації для полів , які будуть перевірятися перед збереженням документа в базі даних. Додатковою можливістю є узгодження взаємозв'язків між документами через вкладені об'єкти чи посилання.

Для взаємодії з базою даних призначені моделі, а саме для таких операцій як створення нових документів, зчитування, оновлення та видалення. Модель представляє конкретний тип документа, який буде зберігатися в MongoDB. Вона побудована на основі схеми, використовуюючи метод mongoose.model() і надає інтерфейс для взаємодії з колекцією документів в базі даних.

2.5.5 Socket.IO

WebSocket є двонаправленим, повнодуплексним протоколом, що протистоїть обмеженням традиційного HTTP протоколу і надає змогу розробникам створювати високоефективні та інтерактивні веб-застосунки, що вимагають частого обміну даними в реальному часі. Перевага WebSockets полягає в тому, що він у режимі реального часу, забезпечує зв'язок між клієнтом і сервером без необхідності постійних HTTP-запитів та відповідей. Протокол має підтримку стану, тому з'єднання між клієнтом і сервером існуватиме до моменту розриву однієї зі сторін. При встановленому та активному з’єднанні, коли відбувається подія, сервер отримує її і надсилає відповідному підключеному клієнту. WebSocket, завдяки зменшенню використання пропускної здатності і навантаження на сервер, підвищує продуктивність та швидкість роботи веб-застосунків.

Socket.IO – JavaScript бібліотека, що надає можливість реалізації веб сокетів та забезпечує двосторонню комунікацію між клієнтом та сервером у режимі реального часу. Socket.IO має бібліотеки для клієнта, яка запускається в браузері, та серверну бібліотеку для node.js, що мають однакові API. Socket.IO створює з'єднання з довгим опитуванням, використовуючи xhr-опитування, яке змінює на найкращий доступний шлях після встановлення з'єднання. У більшості випадків це призводить до створення з'єднання WebSocket. Через створення об'єкту Socket.io та передачі йому серверу Express відбувається ініціалізація Socket.io. Для відправки повідомлень клієнту використовується метод emit(), а для прийому повідомлень - метод on().

Socket.io є потужною та ефективною технологією, яка може виконувати завдання будь-якої складності. Вона знайшла широке застосування у багатьох галузях, зокрема в електронних транзакціях, ігрових платформах, а також у складних бізнес-проєктах. Однією з ключових переваг Socket.io є його простота використання та налаштування. Бібліотека надає зручний API, який дозволяє

користувачам взаємодіяти один з одним в реальному часі, надсилати повідомлення, обмінюватися даними та спілкуватися.

2.6 Середовище розробки веб-застосунку WebStorm

WebStorm - це інтегроване середовище розробки (IDE), яке забезпечує широкі можливості для створення, відлагодження та керування веб-застосунками[19]. WebStorm, розроблений компанією JetBrains, є одним з провідних редакторів коду для веб-розробки. Він відомий своєю потужністю, функціональністю та зручним інтерфейсом, що допомагає розробникам ефективно працювати з веб-технологіями, забезпечує швидкий розвиток проєктів та знижує кількість помилок.

WebStorm підтримує мови програмування JavaScript і TypeScript, технології HTML та CSS, платформу Node.js, а також популярні фреймворки та бібліотеки, такі як React, React Native, Vue, Angular. Він автоматично підказує та доповнює код, розпізнає потенційні помилки та надмірності, і пропонує швидкі варіанти виправлення. Для прискорення кодування вбудовані постфіксне завершення, шаблони, Emmet, підказки параметрів у викликах методів і функцій. WebStorm забезпечує легкість рефакторингу змінних, функції або класів, за рахунок автоматичного пошуку всіх посилань та оновлення в проєкті.

Крім того, у WebStorm інтегровано популярні інструменти розробки, наприклад, система керування версіями Git чи менеджер пакетів npm, системи автоматичної збірки та багато інших. WebStorm оснащений різними лінтерами, такими як Stylelint і ESLint, що дозволяє вам легко контролювати якість коду та дотримуватися кращих практик розробки.

Особливо корисно WebStorm при розгляді великих кодових баз, де швидкий пошук і аналіз можуть значно зекономити час розробників. WebStorm

надає можливість швидко переглядати всі збіги, пов'язані з пошуком, переходити до оголошень змінних чи функцій, показувати їх використання по всьому проєкту, що полегшує навігацію та редагування коду.

WebStorm підтримує модульне тестування, за допомогою популярних фреймворків, таких як Jest, Mocha, Karma, що є важливою складовою розробки програмного забезпечення. WebStorm надає зручний інтерфейс для перегляду результатів тестування у вигляді дерева, що дозволяє швидко знайти проблемні частини коду.

ВИСНОВКИ ДО РОЗДІЛУ 2

В даному розділі було проведено огляд технологій для проектування клієнтської та серверної частин веб-застосунку. Досліджено основні особливості та недоліки існуючих фреймворків та бібліотек для реалізації інтерфейсів. Після аналізу ключових функціональних переваг та доцільності використання методів, було зроблено вибір рішень реалізації. Надано опис додаткових інструментів розробки, що будуть використовуватись при проектуванні веб-застосунку для створення інтерактивних навчальних квестів. Обрано середовище розробки. Як підсумок, було визначено основний технологічний стек необхідний для розробки кінцевого веб-застосунку.

РОЗДІЛ 3.
МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Після дослідження та вибору технології для проектування інтерфейсів веб-застосунку серверної та клієнтської частин слід виокремити функціональні вимоги та перейти до безпосереднього опису розробки програмного забезпечення веб-застосунку для створення інтерактивних навчальних квестів.

3.1 Функціональні вимоги

Таблиця 3.1 – Функціональна вимога REQ001

Назва	Реєстрація користувача
Зміст	Неавторизований користувач зобов'язаний пройти етап реєстрації у веб-застосунку .

Таблиця 3.2 – Функціональна вимога REQ002

Назва	Авторизація користувача
Зміст	Перевірка правильності введення користувачем електронної пошти та паролю, що зберігається зашифрований в базі даних. Надання доступу до перегляду сторінок в разі успішного проходження.

Таблиця 3.3 – Функціональна вимога REQ003

Назва	Перегляд сторінок
Зміст	Перегляд сторінок доступний лише авторизованому користувачу.

Таблиця 3.4 – Функціональна вимога REQ004

Назва	Перегляд власних створених квестів
Зміст	За умови визначення ролі користувача, як викладача, можливе відображення переліку існуючих квестів, які містять основні дані та налаштування.

Таблиця 3.5 – Функціональна вимога REQ005

Назва	Створення квестів
Зміст	За умови визначення ролі користувача, як викладача, можливе створення квестів.

Таблиця 3.6 – Функціональна вимога REQ006

Назва	Налаштування параметрів квесту
Зміст	Викладач має можливість задати параметри квесту для подальшого його відображення в переліку існуючих квестів та використовувати їх під час проведення квесту.

Таблиця 3.7 – Функціональна вимога REQ007

Назва	Додавання задач та питань
Зміст	Викладачу надається можливість додавання задач та питань до заздалегідь створеного квесту.

Таблиця 3.8 – Функціональна вимога REQ008

Назва	Налаштування параметрів задач та питань
Зміст	Викладач має можливість задати параметри задач та питань для подальшого їх відображення в конструкторі квесту та використовувати їх під час проведення квесту.

Таблиця 3.9 – Функціональна вимога REQ009

Назва	Перегляд доданих задач та питань
Зміст	Відображення переліку доданих задач та питань до заздалегідь існуючих квестів користувача, що містить основні дані та налаштування.

Таблиця 3.10 – Функціональна вимога REQ010

Назва	Видалення задач та питань
Зміст	Викладач має можливість видалення задач та питань заздалегідь створеного квесту.

Таблиця 3.11 – Функціональна вимога REQ011

Назва	Редагування задач та питань
Зміст	Викладач має можливість повернутись до доданих задач і питань, редагувати їх зміст та налаштування.

Таблиця 3.12 – Функціональна вимога REQ012

Назва	Видалення квестів
Зміст	Викладач має можливість видалення створених квестів.

Таблиця 3.13 – Функціональна вимога REQ013

Назва	Редагування квестів
Зміст	Викладач має можливість повернутись до створених квестів, редагувати їх задачі, питання та налаштування.

Таблиця 3.14 – Функціональна вимога REQ014

Назва	Перегляд публічних квестів
Зміст	Відображення переліку існуючих квестів користувачів, які містять основні дані та налаштування, за умови визначення у квесті публічного доступу.

Таблиця 3.15 – Функціональна вимога REQ015

Назва	Пошук публічних квестів
Зміст	Відображення публічних квестів, які задовольнили критерій користувацького пошуку, введений у форму.

Таблиця 3.16 – Функціональна вимога REQ016

Назва	Перегляд профілю квесту
Зміст	Користувач має можливість перегляду профілів публічних квестів, де відображаються всі характеристики, питання, задачі та відгуки.

Таблиця 3.17 – Функціональна вимога REQ017

Назва	Відгук на квест
Зміст	Користувач має можливість оцінити та залишити відгук на публічні квести в їх профілі.

Таблиця 3.18 – Функціональна вимога REQ018

Назва	Активізація квесту
Зміст	Викладач має можливість активізації власних, заздалегідь створених квестів для проходження їх студентами в онлайн режимі.

Таблиця 3.19 – Функціональна вимога REQ019

Назва	Проходження квесту
Зміст	Студент має можливість проходити публічні та активовані квести. Для проходження активованого квесту студенту необхідно отримати код доступу від викладача.

Таблиця 3.20 – Функціональна вимога REQ020

Назва	Формування оцінки за квест
Зміст	Автоматична перевірка веб-застосунком правильності наданих відповідей та формування загального результату, спираючись на налаштування задач та питань пройденого квесту.

Таблиця 3.21 – Функціональна вимога REQ021

Назва	Формування рейтингів
Зміст	Порівняння веб-застосунком результатів проходження квесту користувачами, спираючись на їх отримані оцінки.

Таблиця 3.22 – Функціональна вимога REQ022

Назва	Формування звітів
Зміст	Формування веб-застосунком переліку питань та задач, на які були дані неправильні відповіді.

Таким чином, визначено функціональні вимоги для подальшого моделювання ПЗ веб-застосунку, що прискорить його розробку.

3.2 Моделювання програмного забезпечення веб-застосунку

Для розуміння методів проектування та управління веб-застосунком слід виконати моделювання програмного забезпечення, що дозволить встановити та проаналізувати функціональність , спроектувати архітектуру та візуалізувати вимоги перед фактичною реалізацією.

Опис процесу реєстрації користувача:

- перехід на сторінку реєстрації;
- заповнення всіх полів регістрації , в полі Student Type користувачу потрібно обрати роль, що надалі забезпечить надану функціональність;
- за успішної валідації всіх полів реєстрації, користувач має можливість підтвердити свої дії, натиснувши на кнопку реєстрації;
- надсилання запиту на сервер;
- перевірка відправлених даних користувача та створення облікового запису зі збереженням у базі даних;
- виконання автоматичної авторизації та переходу користувача на головну сторінку.

Опис процесу авторизації користувача:

- перехід на сторінку авторизації;
- заповнення всіх полів авторизації;
- при успішній валідації всіх полів авторизації, користувач має можливість підтвердити свої дії, натиснувши на кнопку авторизації;
- перевірка відповідності введених даних зі значеннями , що містяться в базі даних;

- при успішній перевірці виконується авторизація та автоматичний перехід користувача на головну сторінку, при виявленні помилок - користувач мусить повторно пройти авторизацію.

Опис процесу створення квесту:

- перехід на сторінку відображення власних квестів та їх створення,при умові авторизації як викладач;
- заповнення необхідних полів для створення квесту;
- при успішній перевірці на відсутність створюваного квесту в базі даних, користувач натискає на кнопку створення квесту;
- збереження квесту в базі даних та автоматичний перехід на сторінку конструктору квесту;
- заповнення необхідних полів для налаштування квесту та додавання задач;
- при успішній перевірці правильності створюваної задачі, користувач натискає на кнопку додавання задачі, процедура виконується бажану кількість разів при автоматичному оновленні компоненту відображення задач, всі задачі зберігаються у state Redux;
- користувач натискає кнопку збереження квесту , відбувається оновлення квесту в базі даних;
- перехід на сторінку відображення власних квестів та їх створення.

Для моделювання бізнес-процесів ПЗ призначені BPMN діаграми, що дозволяють зрозуміти та проаналізувати їх, візуалізувавши послідовність дій, ролі учасників, потоки даних та різні сценарії взаємодії між процесами та системами. Приклад BPMN діаграми для даного веб-застосунку наведено на рис. 3.1.

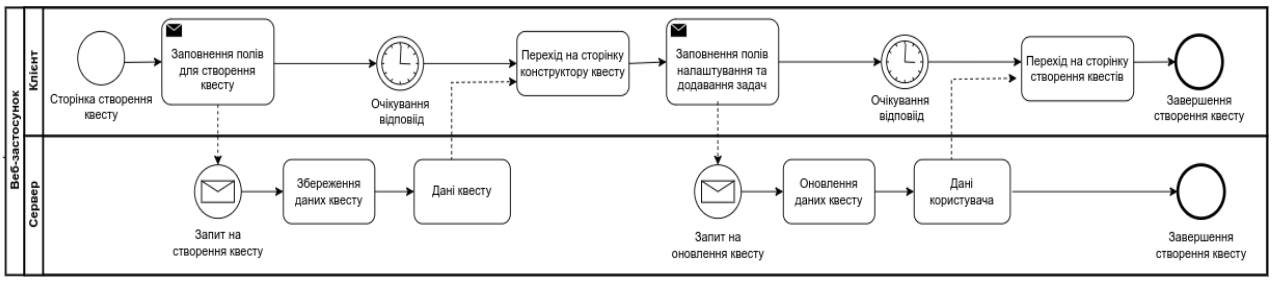


Рисунок 3.1 – BPMN діаграма для процесу створення квесту

Опис процесу пошуку квесту:

- перехід на сторінку публічних квестів;
- заповнення поля для пошуку ключовими словами ;
- користувач натискає на кнопку пошуку;
- виконання пошуку серед публічних квестів за введеним ключовим словом у базі даних;
- формування списку знайдених відповідних квестів.

Опис процесу створення відгуку на квест:

- перехід на сторінку профілю квесту;
- заповнення поля для відгуку та підтвердження дій за допомогою кнопки додавання відгуку;
- збереження відгуку у базі даних ;
- оновлення сторінки профілю квесту з відгуком на квест та кнопкою для його видалення.

Опис процесу проведення квесту:

- перехід на сторінку гра, при умові авторизації як студент, викладач виконує перехід на сторінку відображення власних квестів та їх створення;
- активація квесту викладачем , збереження інформації про гру в базі даних;
- надсилання викладачем пін коду для приєднання , введення пін коду студентом;

- поступове відображення та оновлення сторінки з задачами та відповідями для викладача та тільки відповідями для студента, відповіді студентів зберігаються у state Redux;
- збереження відповідей студентів та їх результатів у базі даних;
- збереження інформації про гру у базі даних;
- відображення рейтингу студентів та їх результатів для викладача;
- відображення оцінки зі звітом, що містить неправильні відповіді для студентів.

3.3 Архітектура програмного забезпечення веб-застосунку

Розробка програмного забезпечення реалізується з використанням клієнт-серверної архітектури стеку MERN. Він складається з чотирьох технологій JavaScript - MongoDB, Express.js, React та Node.js, що обумовлює легкість використання та підтримки при вивченні однієї мови програмування та структури документа JSON. Стек MERN використовує трирівневу архітектуру, що складається з інтерфейсу, серверної частини та бази даних. На рисунку 3.2 проілюстровано схему даної архітектури.

Архітектура стеку MERN забезпечує як інтерфейс користувача з користувацьким досвідом та взаємодією анімованих елементів клієнтської частини, так і бекенду з алгоритмічним та аналітичним наповненням.

У даній архітектурі, клієнтські та серверні компоненти взаємодіють між собою через мережу за допомогою HTTP протоколу. Верхній, клієнтський рівень стеку реалізується за допомогою Javascript, HTML та CSS з використанням ReactJS як фреймворку. Він відповідає за відображення інтерфейсу користувача і доступ до взаємодії з функціоналом веб-застосунку. Клієнтська частина відправляє HTTP-запити на серверну частину з

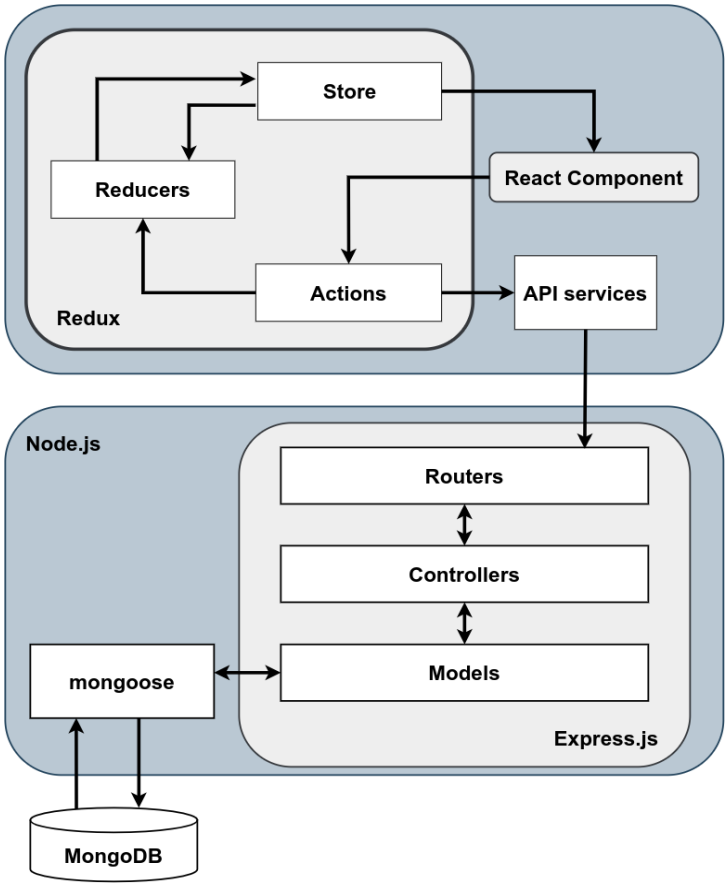


Рисунок 3.2 – Архітектура програмного забезпечення веб-застосунку

використанням Node.js та Express.js після отримання запитів, виконує їх обробку чи збереження та надає HTTP відповідь. Express.js підтримує серверний фреймворк і проміжне програмне забезпечення, що працює всередині сервера Node.js. Серверна частина є сервером додатків, які застосовуються в якості мосту для зв'язку клієнтської частини та рівня бази даних. Зв'язок між клієнтською та серверною частинами в архітектурі MERN веб-застосунку відбувається в результаті виконання запиту від користувача. Розглянемо цей процес послідовно.

При взаємодії з клієнтською частиною веб-застосунку , користувач має можливість виконувати різні операції, наприклад, натискання кнопки підтвердження чи заповнення полів форми. У зв’язку з цим відбувається виклик методу обробника подій у React компоненті, де знаходився відповідний

елемент. Функції обробки подій можуть включати в себе Redux actions. Вони представляють функції, які відправляють запити на зміну стану додатка. Actions містять типи подій та корисне навантаження з необхідними даними для оновлення стану. Далі отриманий об’єкт actions та поточний стан веб-застосунку направляються у reducers, які приймають рішення щодо оновлення стану у Redux store при потребі, відповідно до actions. Кожен reducer відповідає за певну частину стану. Після оновлення стану, React компоненти, що були підписані на відповідний стан, отримують нові дані. Відбувається повторний рендеринг з використанням цих даних, що дозволяє оновити відображення.

В Actions також реалізовано виклики функції, які взаємодіють з серверною частиною через HTTP-запити. API services використовують роутинг для визначення URL-шляхів для обробки запитів. В момент виклику запиту до серверної частини, відбувається відправка запиту з даними до відповідного route. Маршрутизація визначається за допомогою методів об'єкта router Express-додатку. Для того, щоб вказати контролер, як функцію зворотного виклику використовується app.use(). Контролери приймають вхідні дані отриманого запиту та реалізують операції взаємодії з базою даних чи обробки бізнес-логіки веб-застосунку, повертаючи відповідь клієнтській частині. Контролери виконуються до визначених маршрутів і використовуючи необхідні моделі для взаємодії з базою даних. Моделі представлені бібліотекою ODM mongoose та відповідають за структуризацію даних та взаємозв’язки між ними. За рахунок моделей забезпечується доступ до бази даних, попередня валідація даних при операціях з нею.

3.4 Конструювання програмного забезпечення

Процес конструювання програмного забезпечення для веб-застосунку, спрямованого на створення інтерактивних навчальних квестів розподілено на

дві частини, клієнтську та серверну. Слід описати необхідні етапи розробки програмного забезпечення, процес проектування бази даних та взаємодію обміну даними між клієнтською та серверною частинами.

3.4.1 Розробка серверної частини веб-застосунку

Далі розглянуто структуру проекту серверної частини веб-застосунку. Як зазначено раніше, розробка серверної частини веб-застосунку здійснювалась за допомогою Express.js на платформі Node.js.

Розглянемо структуру серверної частини проєкту:

– controllers/ - директорія, що призначена для розміщення файлів контролерів, які відповідають за обробку запитів та взаємодію з базою даних;

– routes/ - директорія , що призначена для розміщення файлів маршрутів, які визначають по якому url-шляху викликається певний контролер;

– models/ - директорія, що призначена для розміщення файлів моделей, які визначають структуру та взаємодію об'єктів, що призначені для збереження та отримання даних з бази даних;

– .gitignore - файл, що включає в себе перелік файлів та директорій проєкту, які повинні бути проігноровані системою контролю версій;

– package-lock.json - файл, що включає в себе інформацію про залежності або пакети, встановлені для проєкту з точними номерами їх версій;

– package.json - файл, що призначений для зберігання важливих метаданих проєкту, його функціональних властивостей;

– server.js - головний файл серверної частини веб-застосунку;

– webSocket.js - файл, що призначений для роботи з веб-сокетами, містить логіку обробки подій, що відбуваються під час з'єднання та взаємодії з веб-сокетами;

– .env - файл, що призначений для зберігання змінних середовища.

Файлом запуску серверної частини є server.js, що містить всі налаштування серверу та підключення різних файлів проєкту. До об’єкту app, що представляє собою застосунок, через метод фреймворку Express додаються middleware, які обробляють запити , що надходять до серверу. Для надання дозволу чи обмеження доступу до серверу з певних доменів використовують cors, вони забезпечують безпечну клієнт-серверну взаємодію. Middleware body-parser розбирає дані, передані в тілі запиту, та розташовує їх у властивостях об'єкта req.body. Передача параметру extended дозволяє обробляти розширені об'єкти JSON, що дає змогу передавати вкладені об'єкти та масиви у формі запиту. Також тут реалізовано підключення до бази даних та роутерів, створення вебсокет серверу. Server.js прослуховує вхідні запити на зазначеному порті та обробляє їх за допомогою відповідних маршрутів та контролерів. Інформацію про опис елементів директорії controllers, а також функції, що в них реалізовано надано у таблицях 3.23 та 3.24.

Таблиця 3.23 - Опис елементів директорії controllers

Назва файлу	Опис
auth.js	Зазначений файл включає реалізацію трьох функцій контролеру для реєстрації та автентифікації , взаємодіє з колекцією User. Містить функції login, register, generateAccessToken.
quest.js	Зазначений файл включає реалізацію функцій контролеру для створення та зміни даних в колекції Quest. Містить createQuest, getQuest, updateQuest, removeQuest, getPersonalQuests, getPublicQuests, getTask addTask, updateTask, removeTask, likeQuest, commentQuest searchQuests.

Кінець таблиці 3.23

Назва файлу	Опис
game.js	Зазначений файл включає реалізацію функцій контролеру для створення та зміни даних в колекції Game. Містить activateGame, getGame, addStudent, updateGame.
studentScore.js	Зазначений файл включає реалізацію функцій контролеру для створення та зміни даних в колекції StudentScore. Містить saveStudentScore, updateStudentScore addAnswer, getAnswers, calculatePoints
ratings.js	Зазначений файл включає реалізацію функцій контролеру для створення та зміни даних в колекції Ratings. Містить saveRatings, getRatings, updateRatings

Таблиця 3.24 - Опис функцій контролерів

Назва функції	Опис
login	Функція перевірки на відповідність даних що прийшли з клієнтської частини при заповненні форми для входу та даних у колекції User. Оновлюється refresh_token, в разі відсутності користувача, повертається помилка – ‘’не дозволено’’.
generateAccessToken	Функція для генерації Access Token. Використовується при виклику у функції контролеру реєстрації.
createQuest	Функція створення нового запису у колекції Quest. Дані надходять з форми для створення квесту. Тільки викладач має право на користування.
getQuest	Функція пошуку запису у колекції Quest за id. Запит даних відбувається при переході на сторінку конструктору квесту. В разі відсутності такого квесту, повертається помилка - ‘’квест не знайдено’’. Тільки викладач має право на користування.

Продовження таблиці 3.24

Назва функції	Опис
updateQuest	Функція пошуку та оновлення запису у колекції Quest за id. Дані надходять при збереженні квесту в конструкторі. В разі відсутності такого квесту, повертається помилка - ‘’квест не знайдено’’. Оновлюється запис квесту. Тільки викладач має право на користування.
removeQuest	Функція видалення запису у колекції Quest за id. При відсутності такого квесту, повертається помилка - ‘’квест не знайдено’’. Запит даних відбувається на сторінці створення та відображення власних квестів. Тільки викладач має право на користування.
getPersonalQuests	Функція пошуку квестів за id користувача у колекції Quest. В разі відсутності таких квестів, повертається пустий об’єкт. Запит даних відбувається на сторінці створення та відображення власних квестів. Тільки викладач має право на користування.
getPublicQuests	Функція пошуку усіх публічних квестів у колекції Quest. Функція повертає 15 найновіших публічних квестів. В разі відсутності таких квестів, повертається пустий об’єкт. Запит даних відбувається на сторінці публічних квестів. Право на користування мають студенти та викладачі.
getTasks	Функція пошуку задачі у записі колекції Quest за id запису квесту. В разі відсутності такого квесту, повертається помилка - “квест не знайдено”. Запит даних відбувається на сторінці конструктору квесту. Тільки викладач має право на користування.
addTask	Функція додавання задачі у запис колекції Quest за id запису квесту. В разі відсутності такого квесту, повертається помилка - “квест не знайдено”. Дані надходять зі сторінки конструктору квесту. Тільки викладач має право на користування.

Продовження таблиці 3.24

Назва функції	Опис
updateTask	Функція оновлення задачі у записі колекції Quest за id запису квесту. В разі відсутності такого квесту, повертається помилка - “квест не знайдено”. Дані надходять зі сторінки конструктору квесту. Тільки викладач має право на користування.
removeTask	Функція видалення задачі у записі колекції Quest за id запису квесту. В разі відсутності такого квесту, повертається помилка - “квест не знайдено”. Запит даних відбувається на сторінці конструктору квесту. Тільки викладач має право на користування.
likeQuest	Функція додавання до поля кількості лайків колекції Quest id користувача. Якщо id користувача перебуває в масиві - видалити. В разі відсутності такого квесту, повертається помилка - “квест не знайдено”. Запит даних відбувається на сторінці профілю квесту. Право на користування мають студенти та викладачі.
commentQuest	Функція оновлення відгуку запису квесту колекції Quest за id квесту. В разі відсутності такого квесту, повертається помилка - “квест не знайдено”. Дані надходять зі сторінки профілю квесту. Право на користування мають студенти та викладачі.
searchQuests	Функція пошуку записів квестів колекції Quest за ключовими словами. В разі відсутності такого квесту, повертається помилка пустий об’єкт. Запит даних відбувається на сторінці публічних квестів. Право на користування мають студенти та викладачі.
activateGame	Функція створення нового запису у колекції Game. Дані надходять зі сторінки створення та відображення власних квестів. Тільки викладач має право на користування.
getGames	Функція пошуку усіх ігор у колекції Game. В разі відсутності ігор, повертається пустий об’єкт. Запит даних відбувається на сторінці проведення квесту. Тільки викладач має право на користування.

Продовження таблиці 3.24

Назва функції	Опис
addStudent	Функція додавання до поля студенти запису колекції Game за id гри. В разі відсутності гри, повертається помилка - “гру не знайдено”. Запит даних відбувається на сторінці проведення квесту.
updateGame	Функція оновлення запису колекції Game за id гри. В разі відсутності гри, повертається помилка - “гру не знайдено”. Запит даних відбувається на сторінці проведення квесту.
saveStudentScore	Функція створення нового запису у колекції StudentScore. Дані надходять зі сторінки проведення квестів. Право на користування мають студенти та викладачі.
updateStudentScore	Функція оновлення запису колекції StudentScore за id. Дані надходять зі сторінки проведення квестів. Право на користування мають студенти та викладачі.
addAnswer	Функція додавання до поля answers запису колекції StudentScore. Запит даних відбувається на сторінці проведення квесту. Право на користування мають тільки студенти.
getAnswers	Функція пошуку поля відповіді у колекції StudentScore за id. В разі відсутності ігор, повертається пустий об’єкт. Запит даних відбувається на сторінці завершення проведення квесту. Право на користування мають тільки студенти.
updateAnswers	Функція оновлення поля answers запису колекції StudentScore за id. Дані надходять зі сторінки проведення квестів. Право на користування мають тільки студенти.
calculatePoints	Функція обчислення загальної набраної кількості балів за квест. Використовується при виклику у функції контролеру addAnswer.
saveRatings	Функція створення нового запису у колекції Ratings. Дані надходять зі сторінки завершення проведення квестів. Тільки викладач має право на користування.

Кінець таблиці 3.24

Назва функції	Опис
getRatings	Функція пошуку запису рейтингу у колекції Game за id. В разі відсутності рейтингу, повертається помилка - “рейтинг не знайдено” . Запит даних відбувається на сторінці завершення проведення квесту. Тільки викладач має право на користування.
updateRatings	Функція оновлення запису колекції Ratings за id рейтингу. Дані надходять зі сторінки проведення квестів. Тільки викладач має право на користування.

Веб-застосунок для інтерактивних навчальних квестів має реалізовані наступні типи головних маршрутів: auth, quest, game, studentScore та ratings. Кожен з цих типів маршрутів має відповідний роутер, який встановлює обробники для кожного маршруту та визначає контролери, які виконуються при зверненні до них. Далі, у таблиці 3.25, зазначено перелік доступних запитів за маршрутами. Роутери відповідають за маршрутизацію запитів та виклик відповідних контролерів. Наприклад, для маршруту /auth/login, роутер authRouter встановлює обробник, який виконує процес аутентифікації користувача. При зверненні до цього маршруту, контролер, пов'язаний з маршрутом /auth/login, отримує дані запиту, перевіряє автентичність користувача та повертає результат авторизації.

Таблиця 3.25 - Опис доступних запитів

Метод	Маршрут	Функція контролеру
POST	api/auth/login	login
POST	api/auth/register	register
POST	api/quests	createQuest
GET	api/quests/${id}	getQuest
PATCH	api/quests/${id}	updateQuest

Кінець таблиці 3.25

Метод	Маршрут	Функція контролеру
DELETE	api/quests/${id}	removeQuest
GET	api/quests	getPersonalQuests
GET	api/quests/public	getPublicQuests
GET	api/quests/${id}	getTasks
POST	api/quests/${id}	addTask
PATCH	api/quests/${id}	updateTask
DELETE	api/quests/${id}	removeTask
PATCH	api/quests/${id}/like	likeQuest
PATCH	api/quests/${id}/comment	commentQuest
GET	api/quests/search/${query}	searchQuests
POST	api/game	activateGame
GET	api/game/${id}	getGame
POST	api/game/${id}/students	addStudent
POST	api/game/${id}	updateGame
PATCH	api/studentScore	saveStudentScore
POST	api/ratings/${id}/studentScore	updateStudentScore
PATCH	api/studentScore/${id}/answers	addAnswer
POST	api/studentScore/${id}/answers	getAnswers
GET	api/studentScore/${id}/answers	updateAnswers
PATCH	api/ratings	saveRatings
POST	api/ratings/${id}	getRatings
GET	api/ratings/${id}	updateRatings

MongoDB зберігає дані у вигляді документів у форматі бінарного представленням JSON-подібних об'єктів BSON. База даних використовує документно орієнтовану модель зберігання даних, кожен документ містить ключі і значення, а його структура може змінюватись в межах однієї колекції.

Зберігання даних об'єкту квесту є найголовнішим в роботі веб-застосунку, оскільки він містить всю необхідну інформацію про квест, його питання, відповіді та правильні варіанти. Для зберігання даних квесту використовується колекція в MongoDB, де кожен документ представляє окремий квест. Слід описати його структуру даних:

- назва квесту надає розуміння основної теми квесту, галузі опитування;
- опис квесту дозволяє зрозуміти підтему та звузити спектр необхідного вивченого матеріалу;
- зображення квесту слугує для привернення уваги студентів;
- id викладача та його ім’я ідентифікують викладача, який створив квест;
- кількість балів за задачу допомагає визначити оцінку студента та здійснювати розрахунки щодо набраних балів;
- кількість задач використовується для статистики та для розрахунку відсотку виконаних задач;
- масив налаштування задач – найважливіша частина даних про квест, тут зберігаються питання та їх типи, метод нарахування балів, перелік відповідей з позначками коректності;
- відмітка про публічність квесту, дозволяє викладачу створювати приватні квести, що не є доступними для інших користувачів;
- дата створення - допомагає відстежувати дату створення квесту та його оновлення.

Опис полів колекції Quest міститься у таблиці 3.26

Таблиця 3.26 – Структура документів колекції Quest

Поле	Тип	Опис
title	String	Назва квесту.
description	String	Опис квесту.
backgroundImage	String	Зображення квесту.
teacherId	ObjectId	ID викладача, що створив квест.
creatorName	String	Вигадане ім’я користувача, що створив квест.
pointsPerTask	Number	Кількість балів, що нараховується при правильному виконанні квесту.
numberOfTasks	Number	Кількість задач, що містить відповідний квест.
isPublic	Boolean	Логічний тип значення, що зазначає приватність чи публічність квесту.
taskList	List	Масив об’єктів, в яких прописані налаштування задач квесту, тип квесту, час проходження , питання, відповіді, тощо.
dateOfCreation	Date	Поточна дата створення квесту.

Також збереження потребують й інші дані, що необхідні для реалізації веб-застосунку. Далі наведено перелік розроблених колекцій:

- колекція , що зберігає зареєстрованих користувачів;
- колекція, що зберігає проведені ігри;
- колекція , що зберігає результати студентів;
- колекція , що зберігає рейтинг студентів;
- колекція, що зберігає статистику пройдених квестів.

Опис полів цих колекцій міститься у таблицях 3.27, 3.28, 3.29, 3.30, 3.31

Таблиця 3.27 – Структура документів колекції User

Поле	Тип	Опис
firstName	String	Прізвище користувача.
lastName	String	Ім’я користувача.
userType	String	Тип користувача, в залежності від вибору Student чи Teacher надається різний функціонал користувачу.
userName	String	Вигадане ім’я користувача, що відображається на головній сторінці.
email	String	Електронна пошта користувача.
password	String	Пароль користувача.
date	Date	Дата авторизації.

Таблиця 3.28 – Структура документів колекції Game

Поле	Тип	Опис
questId	ObjectId	ID квесту, за яким почалась гра.
teacherId	ObjectId	ID викладача, що розпочав гру.
pin	String	Пін-код приєднання , що надсилається викладачем студентам для проходження квесту.
studentList	List of ObjectId	Масив ID студентів, що приєднались до гри.
isLive	Boolean	Логічний тип значення, що зазначає чи проведення квесту відбувається онлайн чи ні.
studentResultList	List of ObjectId	Масив ID результатів студентів, що приєднались до гри.
holdingDate	Date	Поточна дата проведення квесту.

Таблиця 3.29 – Структура документів колекції StudentScore

Поле	Тип	Опис
gameId	ObjectId	ID гри, що було проведено.
studentId	ObjectId	ID студента, що проходив гру.
points	Number	Кількість набраних балів студентом під час проходження квесту.
answers	List	Масив, що зберігає значення відповідей і їх номерів, часу виконання задачі, кількість отриманих балів,

Таблиця 3.30 – Структура документів колекції StudentRatings

Поле	Тип	Опис
gameId	ObjectId	ID гри, що було проведено.
studentScoreList	List of ObjectId	Масив ID результатів проведення квесту студентів, що приєднались до гри.
rank	List	Масив об’єктів ID студентів та їх набраних балів.

Таблиця 3.31 – Структура документів колекції GameStatistic

Поле	Тип	Опис
gameId	ObjectId	ID гри, що було проведено.
studentScoreList	List of ObjectId	Масив ID результатів проведення квесту студентів, що приєднались до гри.
meanQuestionTime	Number	Середнє значення тривалості відповіді на питання.
meanQuestionPoints	Number	Середнє значення отриманої оцінки за питання.
easiestQuestion	Number	Номер найлегшого запитання
hardestQuestion	Number	Номер найскладнішого запитання

Отже, було проведено опис колекцій бази даних, він дозволяє визначити, які типи даних будуть зберігатись і як вони будуть організовані. Це важливо для забезпечення ефективного доступу до інформації та зручності у взаємодії з даними.

3.4.2 Розробка клієнтської частини веб-застосунку

Далі розглянуто структуру проекту клієнтської частини веб-застосунку. Як зазначено раніше, розробка клієнтської частини веб-застосунку здійснювалась за допомогою React з використанням інструментів Redux.

Розглянемо структуру клієнтської частини проєкту:

– actions/ - директорія, що призначена для розміщення файлів які визначають дії (actions) для Redux;

– api/ - директорія, що призначена для розміщення файлів які містять функції для взаємодії з серверним API;

– assets/ - директорія, що призначена для розміщення статистичних файлів, зображення лого, шрифти тощо;

– components/ - директорія, що призначена для розміщення файлів React компонентів , які використовуються в сторінках веб-застосунку;

– constants/ - директорія, щомістить файл констант типів Redux дій.

– pages/ - директорія, що призначена для розміщення файлів React компонентів, які представляють окремі сторінки веб-застосунку;

– reducers/ - директорія, що призначена для розміщення файлів Redux reducers, що управляють станом веб-застосунку;

– routes/ - директорія, що містить файл встановлення приватних маршрутів у веб-застосунк;

– theme/ - директорія, що містить файл, де визначається тему додатку, прописують глобальні стилі веб-застосунку;

– App.js - головний файл додатку, що включає структуру веб-застосунку та його маршрутизацію.;

– index.js - файл, що відповідає за ініціалізацію веб-застосунку та рендеринг React-компонентів в контейнер зі вказаним HTML-елементом, що має ідентифікатор "root".

Розроблений на React веб-застосунок використовує такий архітектурний підхід, що реалізація завантаження всього контенту відбувається під час старту. Браузер завантажує файл index.html з HTML-елементом, який має ідентифікатор "root", що є контейнером для рендерингу React веб-застосунку. Саме за допомогою index.js виконується процес рендерингу головного компонента App.js у кореневий елемент index.html Після завантаження браузером index.js, що містить весь код React веб-застосунку, проходить відображення вмісту початкової сторінки. Маніпулювання елементами веб-застосунку породжує зміну стану та оновлення відображення без потреби перезавантаження сторінки.

App.js виконує функції управління компонентами у всьому застосунку. За допомогою react-router-dom встановлюється роутинг відповідних шляхів до компонентів, які будуть відображатись для заданого маршруту. В таблиці 3.32 наведено опис компонентів сторінок з відповідними їм маршрутами. У веб-застосунку реалізована зміна теми , яка впроваджується через провайдер контексту ColorSwitchMode.Provider для передачі глобальних даних та функцій за рахунок обгортання всіх компонентів. Також в App.js відбувається підключення сокетів через socketIO.connect для встановити з'єднання з сервером через WebSocket.

Таблиця 3.32 – Опис компонентів сторінок

Файл сторінки	Опис компоненту	Маршрут
Home.js	Компонент головної сторінки веб-застосунку, де реалізовано привітання, інформація про користувача та його статистика.	/
Auth.js	Компонент сторінки реєстрації та авторизації у веб-застосунку. При успішній реєстрації чи авторизації відбувається перехід на компонент сторінки Home.	/auth
Play.js	Компонент сторінки приєднання до квесту, що проводиться. Доступний користувачам, що зареєстровані , як Student. При успішному приєднанні відбувається перехід на компонент сторінки StudentScreen.	/play
MyQuests.js	Компонент сторінки створення та перегляду власних квестів. Реалізовано слайдер відображення власних квестів. Містить компоненти Quest та QuestList. При успішному заповненні форми створення квесту здійснюється перехід на компонент сторінки QuestCreate.	/myQuests
PublicQuests.js	Компонент сторінки перегляду та пошуку публічних квестів. Використовує компонент PublicQuest для відображення переліку квестів. При натисканні на відображення публічного квесту здійснюється перехід на компонент сторінки QuestProfile.js. Оновлення компоненту відбувається при оцінюванні та пошуку квесту.	/publicQuests

Кінець таблиці 3.32

Файл сторінки	Опис компоненту	Маршрут
QuestCreate.js	Компонент сторінки конструктору квестів. Складається з таких компонентів:AddTask, AnswerList, TaskDisplay, TaskSettings, TaskList, QuestSettings. На сторінці відбувається налаштування задач та квесту, додавання запитань з вибором правильної відповіді, відображення доданих задач. При успішному наповненні квесту запитаннями відбувається перехід на компонент сторінки MyQuests.	/myQuests/:userName
QuestProfile.js	Компонент сторінки профілю квесту. Використовує компоненти TaskList і CommentSection. Оновлення компоненту відбувається при коментуванні квесту.	/publicQuests/:id
About.js	Компонент сторінки з інформацією про веб-застосунок.	/about
Contacts.js	Компонент сторінки веб-застосунок з контактами.	/contacts
TeacherScreen.js	Компонент сторінки проведення квестів. Можливість переходу на дану сторінку надається лише викладачу. Відбувається постійне оновлення компоненту, здійснюється відображення задач та питань, запуск таймерів, формування рейтингів. Складається з компонентів WaitingRoom, Task.	/play/teacher/:id
StudentScreen.js	Компонент сторінки проведення квестів. Можливість переходу на дану сторінку надається лише студенту. Відбувається постійне оновлення компоненту, здійснюється відображення кнопок відповідей, запуск таймерів, формування звітів та оцінки. Використовує компонент AnswerButton.	/play/student/:id

Окрім зазначених компонентів сторінок, у веб-застосунку використовуються додаткові компоненти Sidebar, Topbar та Layout. Sidebar розташований з лівого боку головного екрану та відповідає за навігаційне меню, що містить посилання на відповідні сторінки. Верхня панель Topbar містить кнопку зміни теми веб-застосунку. Layout визначає загальну структуру сторінки наповненість компонентами, а саме Sidebar та Topbar.

Для керування станом веб-застосунку використовувався Redux. Спочатку було створено необхідні reducers, що зазначені в таблиці 3.33. Вони оновлюють стан даних у store веб-застосунку, отримуючи дію та її тип. Використовуючи функцію createStore, що надає бібліотека redux, створено Redux store, параметрами вказано reducers та middleware thunk, який має функції асинхронності. Для доступу до Redux store з будь-якого компоненту веб-застосунку, головний компонент React, App огорнуто в компонент Provider з бібліотеки react-redux і параметром передано store. Для оновлення стану в React компоненті викликається хук useDispatch з параметром дії. Щоб отримати дані зі store застосовано хук useSelector з бібліотеки react-redux

Таблиця 3.33 – Опис реалізованих Redux reducers

Назва файлу	Опис
authReducer.js	Reducer , що змінює стан веб-застосунку, обробляючи типи дій, що стосуються зміни даних користувача. Аналізує такі типи actions з відповідними їм діями: AUTH, LOGOUT.
questReducer.js	Reducer , що змінює стан веб-застосунку, обробляючи типи дій, що стосуються зміни даних квесту. Аналізує такі типи actions з відповідними їм діями: CREATE_QUEST, GET_QUEST, UPDATE_QUEST, REMOVE_QUEST, GET_PERSONAL_QUESTS, GET_PUBLIC_QUESTS, LIKEQ_UEST COMMENT_QUEST, SEARCH_QUESTS.

Кінець таблиці 3.33

gameReducer.js	Reducer , призначений для зміни стану веб-застосунку, обробляючи типи дій, що стосуються зміни даних проведення квесту. Аналізує такі типи actions з відповідними їм діями: ACTIVATE_GAME, GET_GAMES, ADD_STUDENT, UPDATE_GAME, GENERATE_ACCESS_TOKEN.
studentScoreReducer.js	Reducer , призначений для зміни стану веб-застосунку, обробляючи типи дій, що стосуються зміни даних результатів проходження квесту. Аналізує такі типи actions з відповідними їм діями: SAVE_STUDENT_SCORE, UPDATE_STUDENT_SCORE, ADD_ANSWER GET_ANSWERS.
ratingsReducer.js	Reducer , що відповідає за зміни стану веб-застосунку, обробляючи типи дій, що стосуються зміни даних рейтингів студентів після проведення квесту. Аналізує такі типи actions з відповідними їм діями: SAVE_RATINGS, GET_RATINGS, UPDATE_RATINGS.
reducers.js	Головний reducer, що містить в собі всі reducers та передається параметром при створені store.

Для розробки клієнтської частини веб-застосунку було використано MaterialUI компоненти. Компонент Grid застосовувався для створення гнучкого сіткового макету, що робить можливим розміщення елементів в різних колонках та рядках. Розміщення текстових елементів здійснювалось через компонент Typography, а групування елементів через Box. Окрім цього, використовувались Button, IconButton, Card, FormControl, TextField, RadioGroup.

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі було проведено розробку програмного забезпечення веб-застосунку для створення інтерактивних навчальних квестів. Було описано моделювання та конструювання програмного забезпечення, зазначено всі послідовності процесів, їх особливості та залежності функціоналу. Обрано та проаналізовано архітектуру створеного веб-застосунку. У контексті обраної архітектури були детально розібрані взаємодії елементів, що складають систему.

Розглянуто проектування бази даних, розробку клієнтської та серверної частин і їх взаємодію обміну даними. Створено документацію проєкту, що містить аналіз вимог, огляд головних функцій та можливостей веб-застосунку, використання і структуру компонентів, процеси збереження та оновлення даних.

РОЗДІЛ 4.
ВІЗУАЛІЗАЦІЯ РЕЗУЛЬТАТІВ РОЗРОБЛЕНОГО ВЕБ-ЗАСТОСУНКУ

Після опису розробки програмного забезпечення слід розглянути користувацький інтерфейс та створити послідовність інструкцій користування веб-застосунком для створення інтерактивних навчальних квестів.

4.1 Демонстрація роботи веб-застосунку

В цьому розділі надається опис користувацького інтерфейсу та наведена інструкція щодо використання запропонованого веб-застосунку для створення та проведення інтерактивних навчальних квестів. Розроблене програмне забезпечення має інтуїтивно зрозумілий та добре організований інтерфейс користувача, що робить його зручним у використанні.

Для початку користування веб-застосунком необхідне встановлення всіх залежностей за допомогою установки пакетів, вказаних у файлі package.json.

Виконання команди “npm install” у командному рядку директорій клієнту та серверу автоматично завантажує та інсталює необхідні пакети. Серверний код, що встановлює зв’язок з базою даних можливо запустити за рахунок використання команди “npm run dev”, що забезпечує перезавантаження при внесенні змін у код. Компіляція та запуск клієнтської частини React на локальному сервері для перегляду у браузері відбувається при застосуванні команди “npm start”.

При правильному проведенні зазначених операцій, веб-застосунок стане доступний за локальною адресою і буде працювати на 3000 порті. При переході за даним посиланням виконується перевірка авторизованості користувача, в

разі відсутності даних відбувається перенаправлення на сторінку входу. Сторінка авторизація продемонстрована на рис. 4.1.

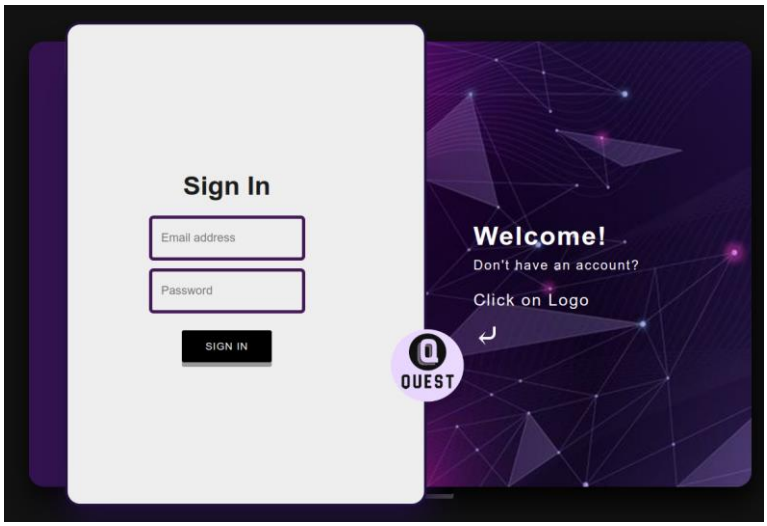


Рисунок 4.1 – Сторінка авторизації

Для переведення користувача на відображення форми реєстрації слугує кнопка з зображенням логотипу веб-застосунку. Користувачу необхідно заповнити вказані поля та натиснути кнопку “SIGN UP” для успішного створення акаунту. На рис. 4.2 наведено сторінку відображення реєстраційної форми.

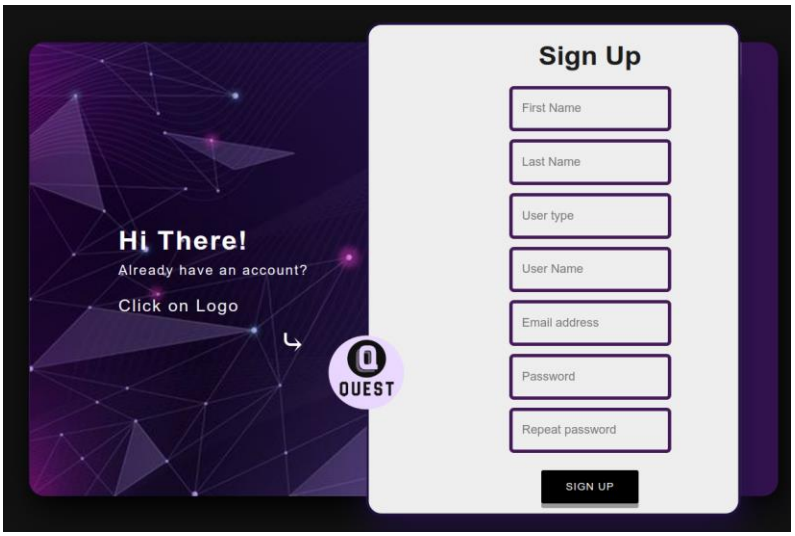


Рисунок 4.2 – Сторінка реєстрації

При невірному заповненні полів реєстраційної форми виникає помилка валідації даних та відображаються повідомлення з підказками щодо правильно формату введених значень. Побачити таку поведінку веб-застосунку можливо на рис. 4.3. Продемонстровані результати показують неправильність заповнення поля “User type”, що не має бути порожнім, а відповідати значенням Student або Teacher. Також помилково введені дані полів “User Name”, “Email” та “Confirmed Password”.

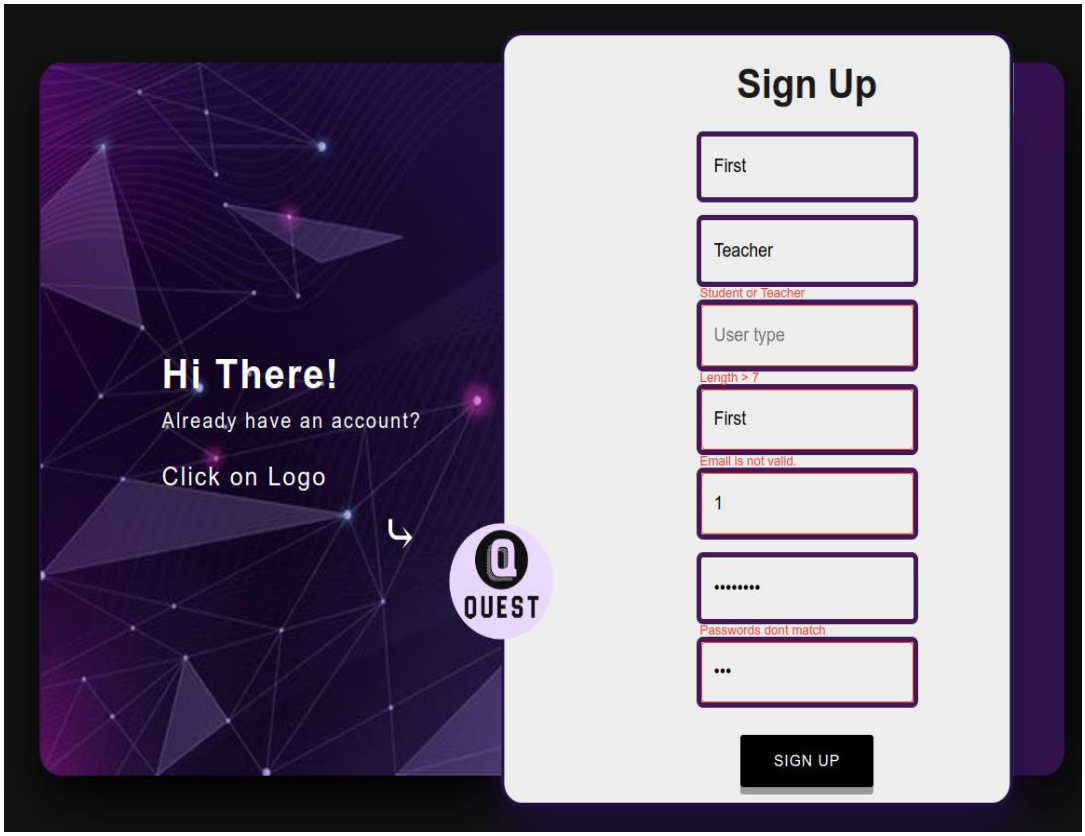


Рисунок 4.3 – Валідація даних під час реєстрації

При вдалому завершенні процесу реєстрації користувача автоматично надсилається до головної сторінки (рис. 4.4).

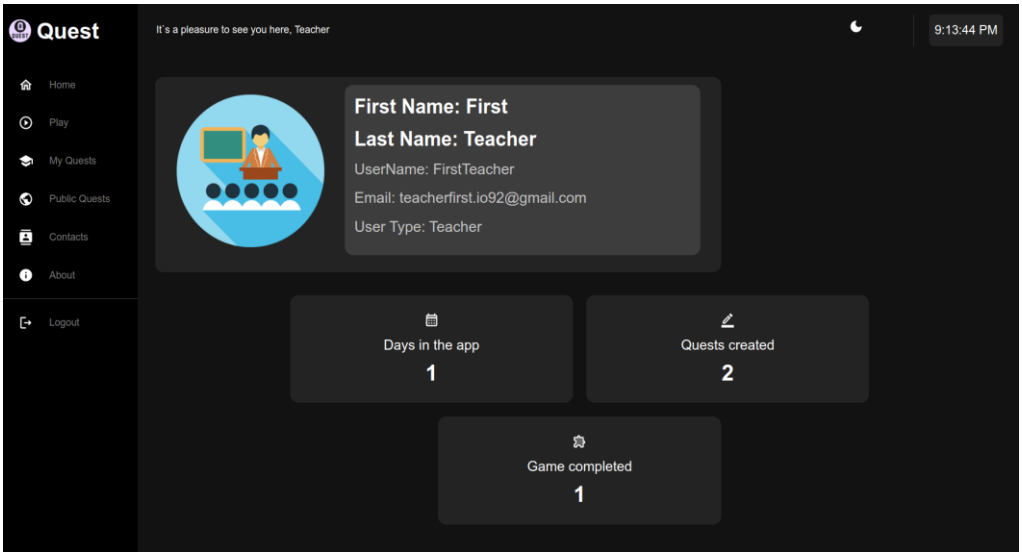


Рисунок 4.4 – Головна сторінка веб-застосунку

Візуальне представлення головної сторінки складається з таких компонентів, як навігаційна бічна панель, профіль користувача, статистика користувача. Верхня панель містить привітання, кнопку зміни теми веб-застосунку та поточний локалізований час. У зв'язку з тим, що веб-застосунок надає різний функціонал для ролей "Teacher" та "Student", навігаційна панель змінює свій зміст (рис. 4.5 - 4.6). Панель навігації надає можливості переходу на сторінки "Home", "Play", "My Quests", "Public Quests", "Contacts" та "About".

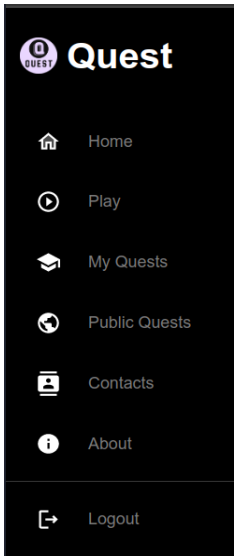


Рисунок 4.5 – Навігаційна бічна панель для студента

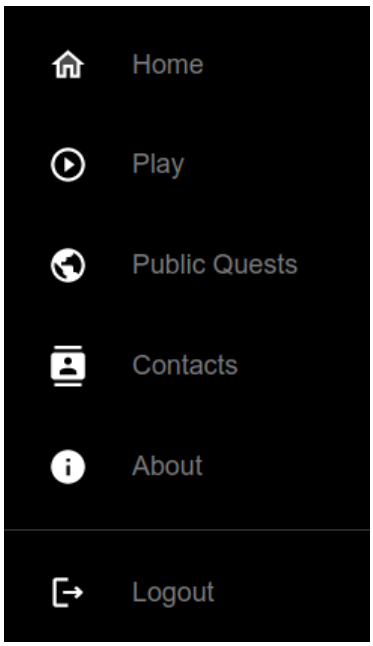


Рисунок 4.6 – Навігаційна бічна панель для викладача

У порівнянні з викладачем, студенту не доступні сторінки перегляду і створення власних квестів. “About” та “Contacts” містять загальну інформацію про веб-застосунок та контакти розробника проєкту. На сторінці “Home” представлені дані профілю користувача (рис. 4.7).

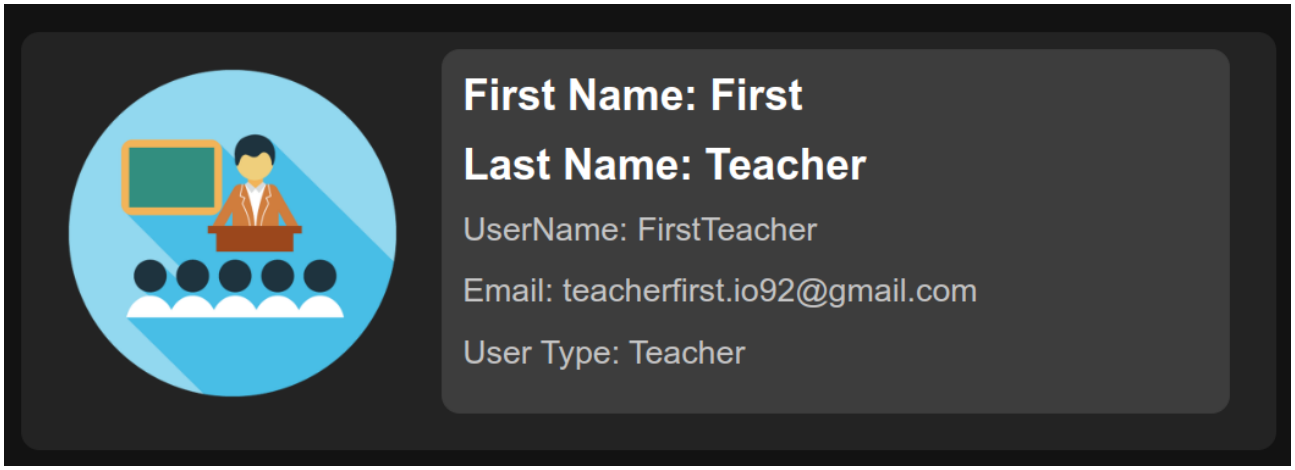


Рисунок 4.7 – Профіль користувача

Нижче ілюструються результати взаємодії користувача із застосунком, а саме кількість днів з моменту реєстрації, кількість створених та проведених

квестів. Дані статистики, в залежності від ролі, також різняться (рис. 4.8 – 4.9). Студенту замість кількості створених квестів показується відсоток його успішності, що обчислюється як середнє значення фактичної кількості зароблених балів під час проходження квестів, поділене на можливу їх кількість.

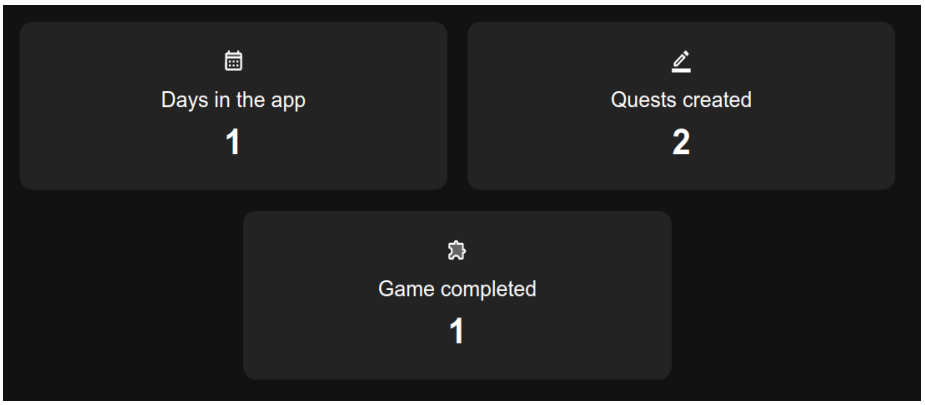


Рисунок 4.8 – Статистика викладача

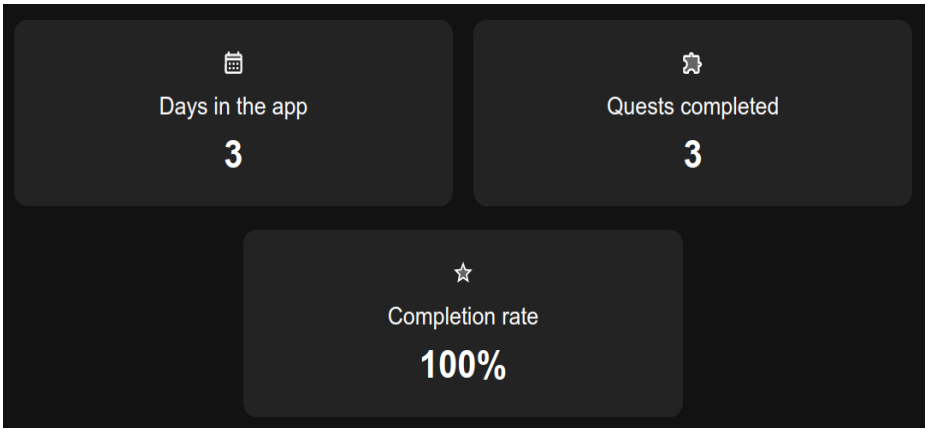


Рисунок 4.9 – Статистика студента

Наступна сторінка навігаційної панелі “My Quests”, зображена на рис. 4.10 призначена для створення та відображення квестів викладача. Вона містить інтерфейс у вигляді послідовних слайдів з картками та форму створення квесту.

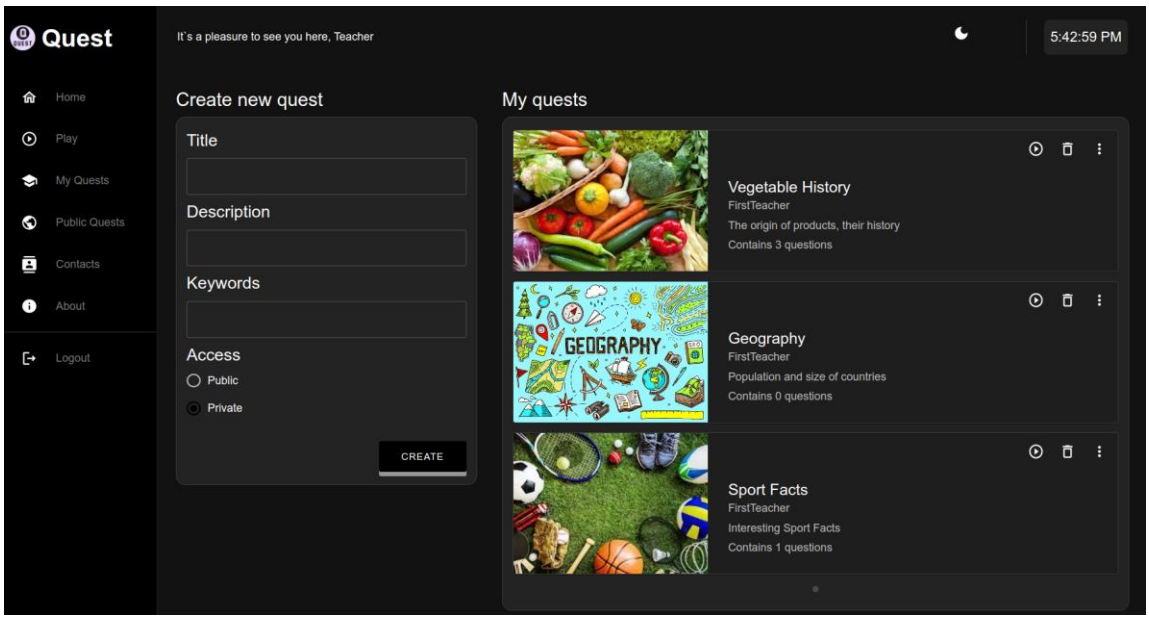


Рисунок 4.10 – Сторінка створення та відображення власних квестів

Картка квесту складається з головного зображення квесту та його інформації. Тут зазначені заголовки квестів, їх опис, ім’я користувача та кількість питань з яких вони складаються (рис. 4.11). На картці присутні три кнопки у вигляді іконок, що відповідають за певні дії:

- кнопка активація квесту супроводжується переходом на сторінку проведення квесту для викладача;
- кнопка видалення квесту прибирає вибраний квест з переліку карток;
- кнопка розширеного перегляду створеного квесту, що дозволяє оновити запитання квесту.

Рисунок 4.11 – Картка створеного квесту

Форма для створення квесту потребує заповнення назви квесту, його опису, ключових слів та встановлення доступу (рис. 4.12). Ключові слова квесту необхідні для пошуку квестів в розділі “Public Quests”. Доступність квестів передбачає можливість вибору публічності квестів, Public квести стануть доступними усім користувачам веб-застосунку. Після заповнення полів форми, користувач натискає кнопку “CREATE”, що відповідає за перенаправлення на сторінку конструктору квесту та збереження даних. Якщо користувач введе назву квесту, що вже існує, буде викликано помилку з відповідними рекомендаціями.

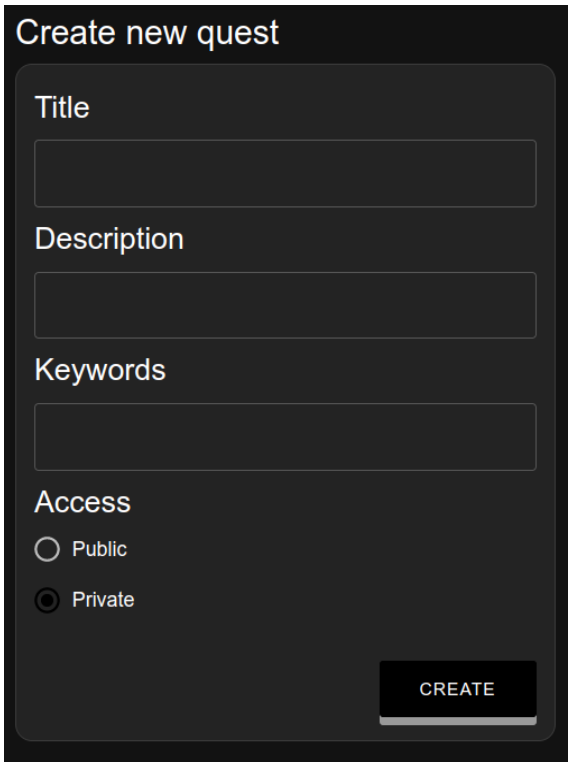


Рисунок 4.12 – Форма для створення квесту

Після створення квесту, користувач автоматично потрапляє на сторінку конструктору квесту. Інтерфейс сторінки конструктору квесту зображений на рис. 4.13. Він складається з трьох вікон, вікно налаштувань, вікно додавання задач та вікно відображення доданих задач до квесту.

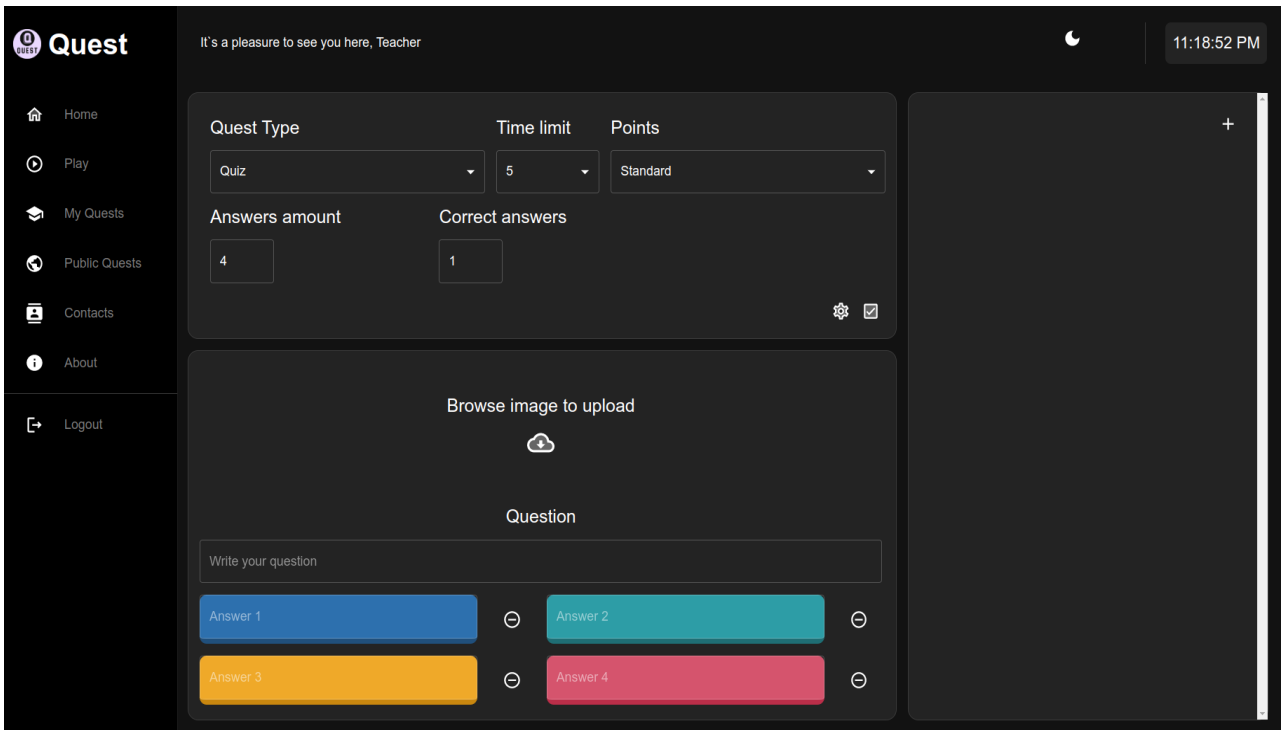


Рисунок 4.13 – Сторінка конструктору квестів

Перше вікно відповідає за налашталштування параметрів задач у квестах (рис. 4.14). Тут визначається тип задачі, користувачу дозволено вибрати з переліку запропонованих - “Quiz”, “True/False”, “OpenQuestion”.

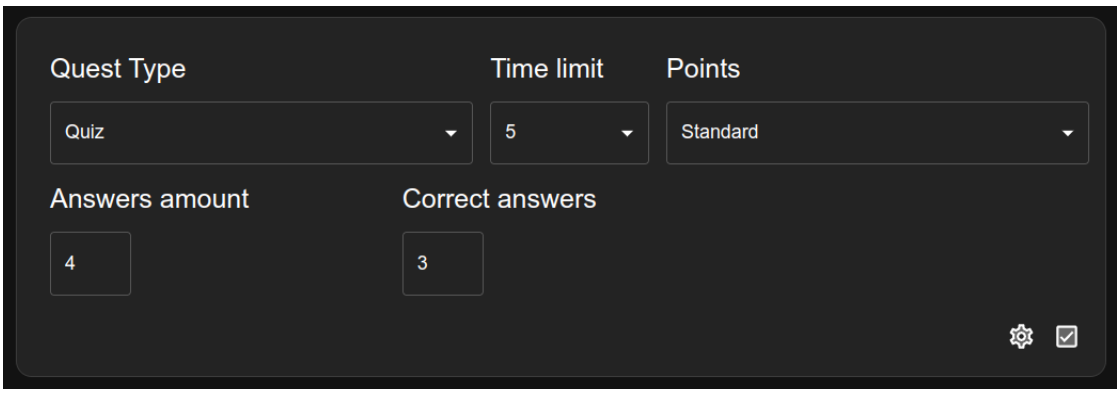


Рисунок 4.14 – Налаштування параметрів задачі

Перелік вибору відведеного часу на проходження квесту варіюється від 5 до 20 секунд. Для типу задачі “Quiz” призначено дві додаткові можливості налаштувань, а саме встановлення кількості відповідей та міток правильності

відповіді. При оцінюванні студента, під час проходження задачі, враховується тип нарахування балів. Викладачу надається можливість встановити тип з урахування часу, що було витрачено на відповідь. Також, тут знаходяться відповідні іконки збереження налаштувань та даних, переключення між вікнами налаштувань квестів і задач (рис. 4.15).

Рисунок 4.15 – Кнопки для збереження налаштувань та переключення між ними

При натисканні кнопки переключення між налаштуваннями, викладачу запропоновано змінити попередні параметри квесту та додати головне зображення.

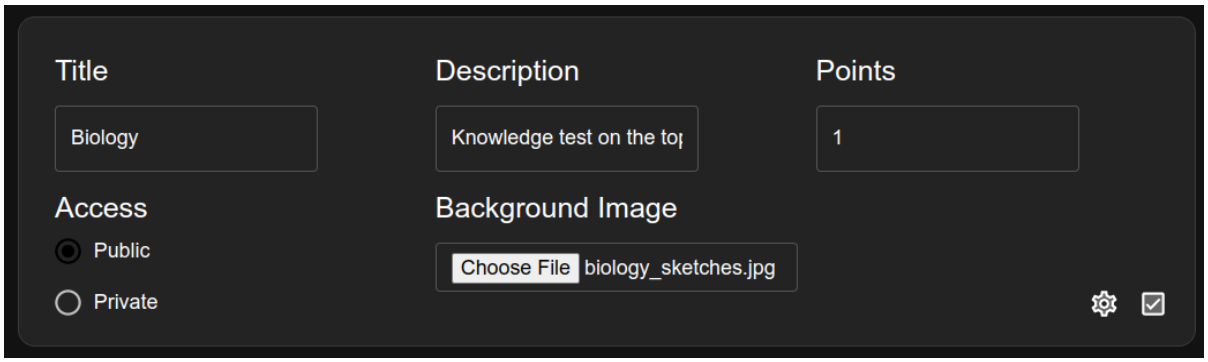


Рисунок 4.16 – Налаштування параметрів квесту

Наступним кроком відбувається додавання задач до квесту. Заповнення всіх необхідних даних продемонстровано на рис. 4.17. Для цього викладачу пропонується додати зображення задачі з можливістю видалення, встановити запитання та вказати правильні відповіді. В разі відсутності правильної відповіді викладач отримує сповіщення про помилку.

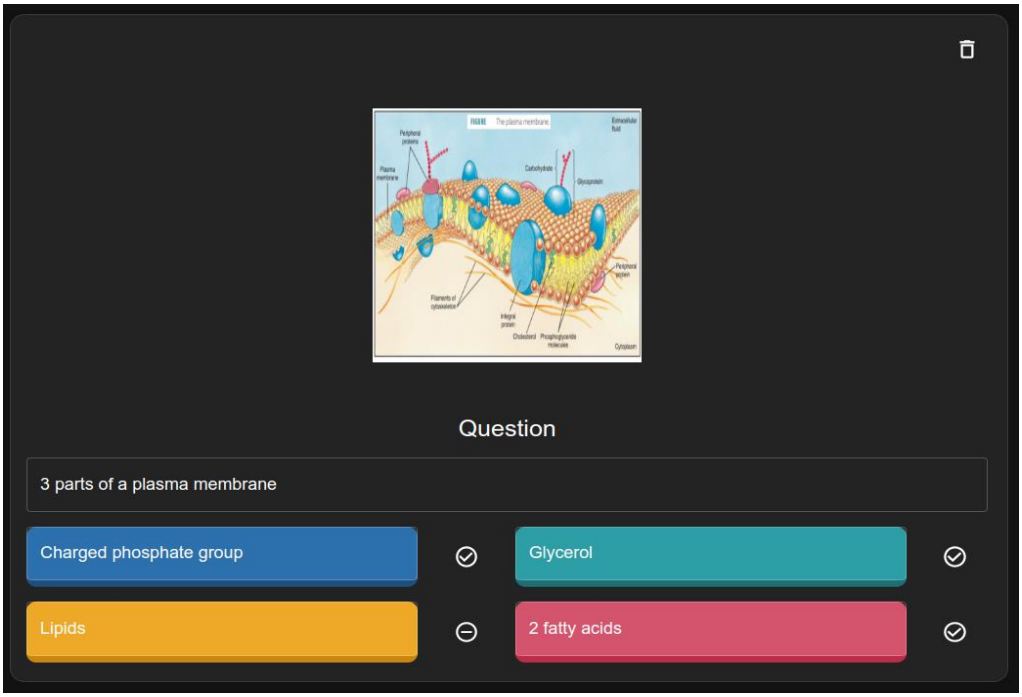


Рисунок 4.17 – Створення задачі

Після заповнення контенту задачі викладач виконує збереження, додана задача відображається у третьому вікні, як показано на рис. 4.18. Таким чином, викладач має змогу слідкувати за процесом заповнення квесту задачами. При натисканні на певну задачу, всі вікна відображають її зміст, викладач може повернутись до задачі та відредагувати. Натискаючи на іконку видалення, задача зникає з переліку доданих.

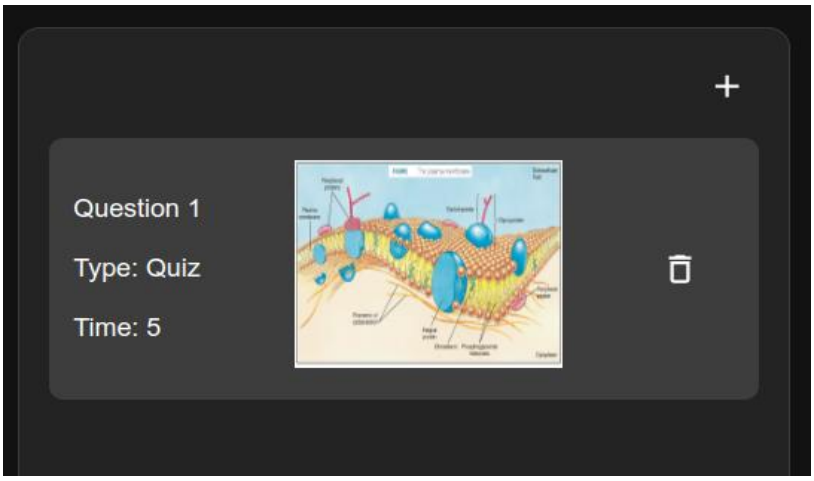


Рисунок 4.18 – Відображення доданих до квесту задач

Додавши необхідну кількість задач, викладач натискає на іконку збереження та автоматично перенаправляється на сторінку створення та відображення власних квестів. На рис. 4.19 показані результати створення квесту.

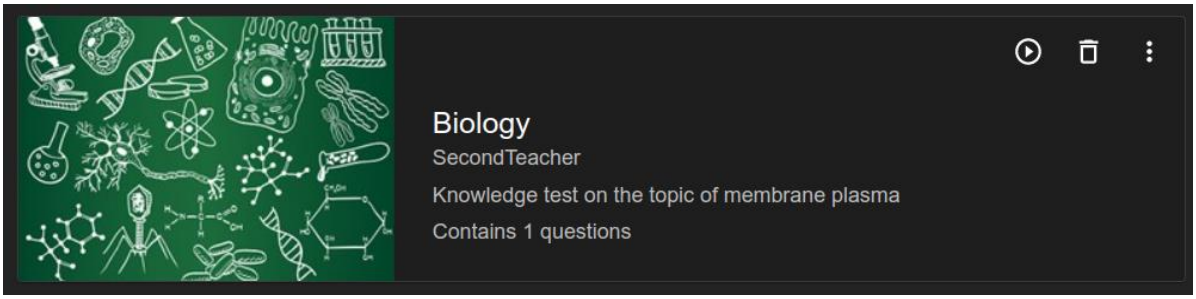


Рисунок 4.19 – Результат створення квесту

Після створення квесту, викладач має право активувати його. Для цього він натискає іконку активації квесту та потрапляє на сторінку проведення. На початковій сторінці проведення квесту користувачу відображається кімната очікування (рис. 4.20). Викладачу надається код для поширення серед користувачів будь-яким способом. Саме за цим кодом студенти мають мають можливість долучитись до гри.

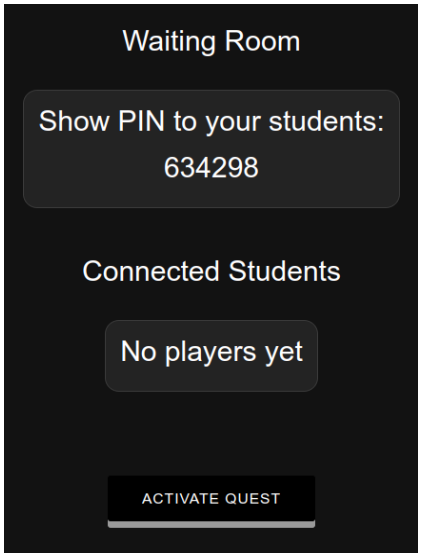


Рисунок 4.20 – Вміст сторінки викладача під час активації квесту

Користувач, отримавши код вводить його у відповідне поле на сторінці для проведення квестів “Play” (рис. 4.21). Після введення коду, користувач натискає кнопку “Join” та в разі успішної перевірки коду зустрічі отримує повідомлення - “You have joined”.

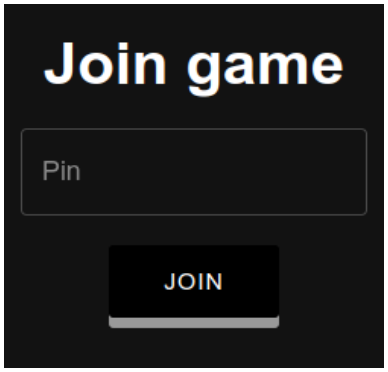


Рисунок 4.21 – Вміст сторінки студента під час приєднання до квесту

Коли користувач успішно додався до кімнати проведення квесту, викладач отримує його ім’я у веб-застосунку, таким чином викладач відслідковує хто доєднався для проходження квесту (рис. 4.22).

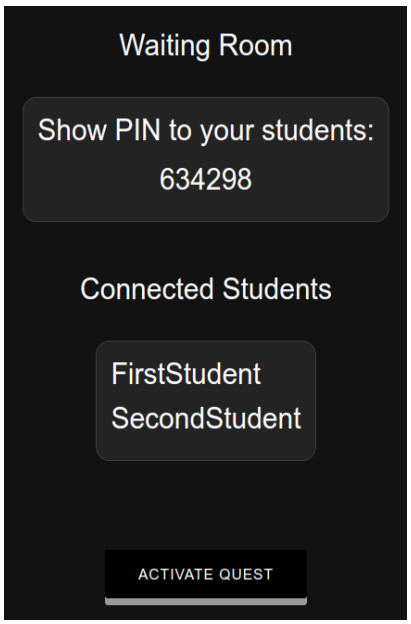


Рисунок 4.22 – Відображення успішного приєднання користувачів до квесту

Після приєднання необхідних викладачу користувачів, він натискає кнопку “ACTIVATE QUEST” та перенаправляється на сторінку старту проведення квесту. З цього моменту відображення екрану користувача та викладача синхронізується. Розпочинається відлік часу до початку квесту, відображаючись у обох учасників проведення квесту. На своєму пристрої викладач транслює зміст питання, зображення та відповіді (рис. 4.23).

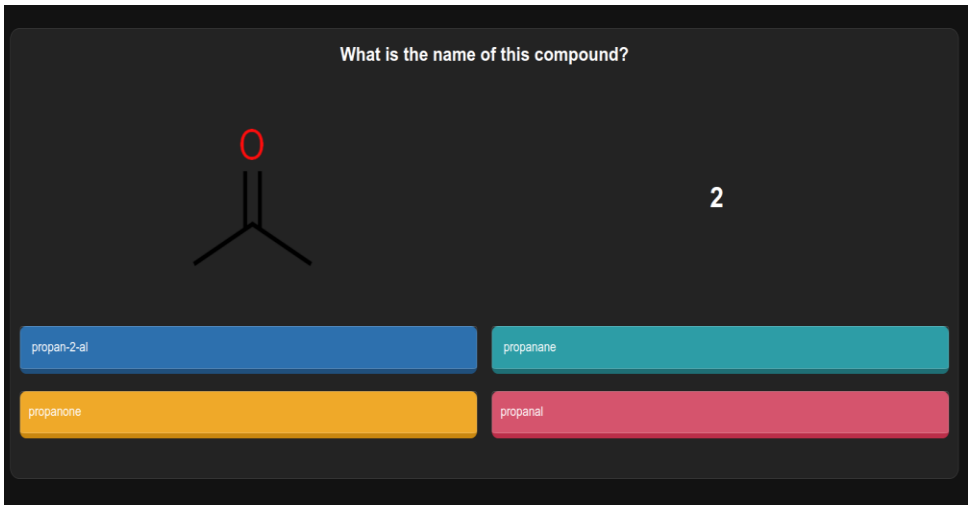


Рисунок 4.23 – Вміст сторінки викладача під час проведення квесту

Залишок часу, виділений на питання, у викладача транслюється справа, а у користувача зліва. Користувач дивиться та аналізує зміст питання та відповіді на головному екрані викладача, після того приймає рішення та натискає на кнопку відповідного кольору. Результати зберігаються задля подальшого отримання рейтингів та оцінки за квест. Вміст сторінки користувача під час проведення квесту наданий на рис. 4.24.

Рисунок 4.24 – Вміст сторінки студента під час проведення квесту

У веб-застосунку реалізована сторінка пошуку та перегляду публічних квестів (рис. 4.25). Користувач має можливість переглянути список з 6 найновіших публічних квестів, дізнатись хто є автором квесту та які теми він охоплює.

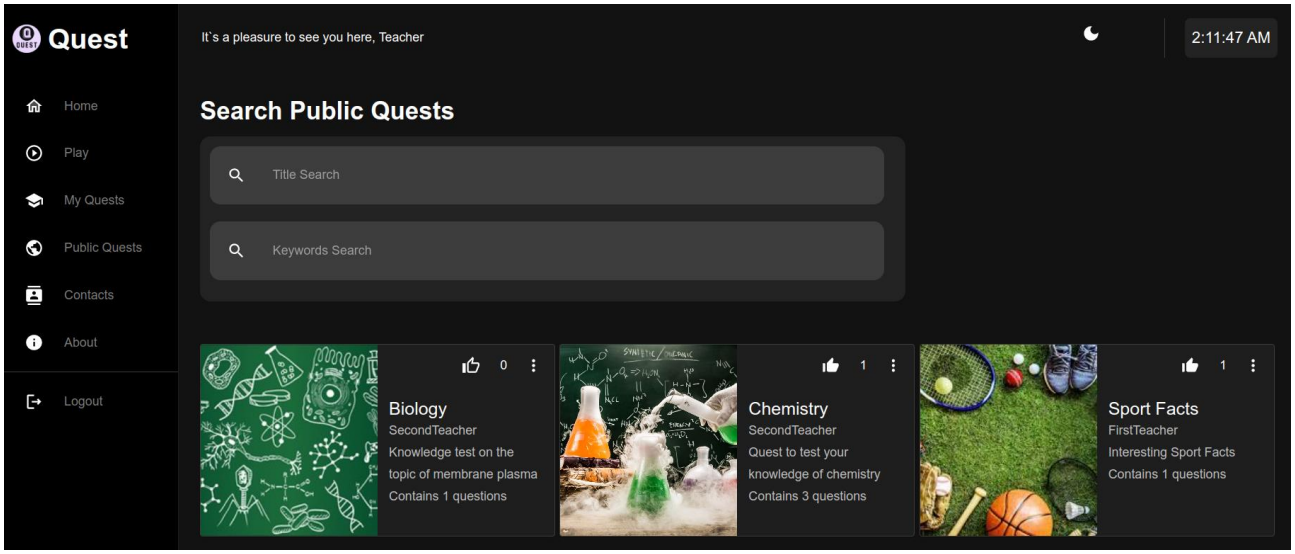


Рисунок 4.25 – Сторінка пошуку та перегляду публічних квестів

Картка публічного квесту містить іконку вподобайки для оцінення квесту. Натискаючи її, змінюється кольорове заповнення іконки та кількість користувачів, що уподобали даний квест (рис. 4.26). Натиснувши ще раз, іконка повернеться до попереднього стану.

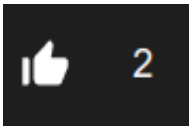


Рисунок 4.26 – Оцінювання публічного квесту

Користувачу надано функціонал пошуку публічного квесту за назвою та ключовими словами. Для пошуку квесту за ключовими словами користувач вводить у поле для пошуку певний запит та натискає “Enter”, таким чином формуються ключові слова для запиту (рис. 4.27). Після отримання результатів, поле для пошуку очищується.

Рисунок 4.27 – Пошук квесту

Після встановлення ключових слів, користувач натискає на іконку для пошуку та отримує відповідь. Рузультат пошуку продемонстровано на рис. 4.28.

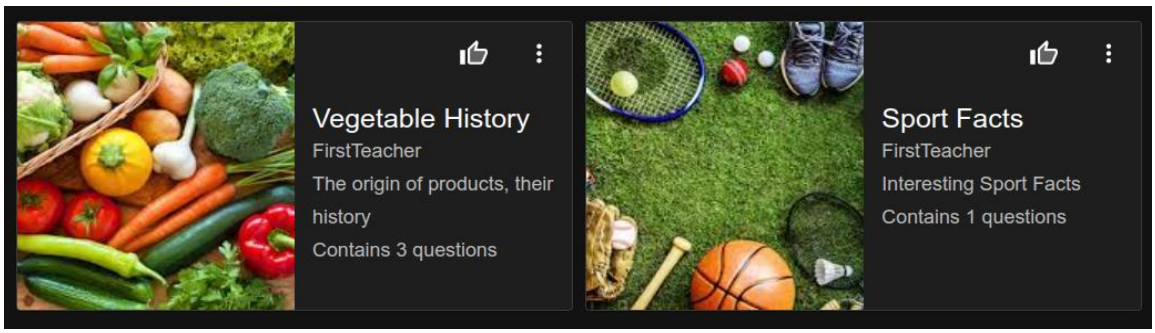


Рисунок 4.28 – Результат пошуку квесту

На картці публічного квесту, у верхньому правому куті, реалізовано іконку, за допомогою якої користувач має змогу перейти на сторінку профілю публічного квесту та дізнатись більш детальну інформацію. Сторінка перегляду публічного квесту складається з чотирьох частин. Опишемо призначення кожної. У самій верхній, лівій частині екрану знаходиться загальна інформація про квест, його опис (рис. 4.29).

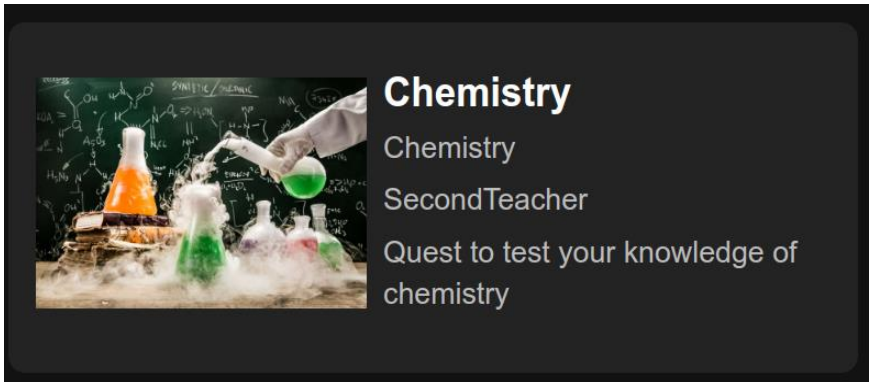


Рисунок 4.29 – Картка на сторінці профілю квесту

Нижче картки розташовується перелік залишених коментарів користувачами та поле для додавання коментарю (рис. 4.30).

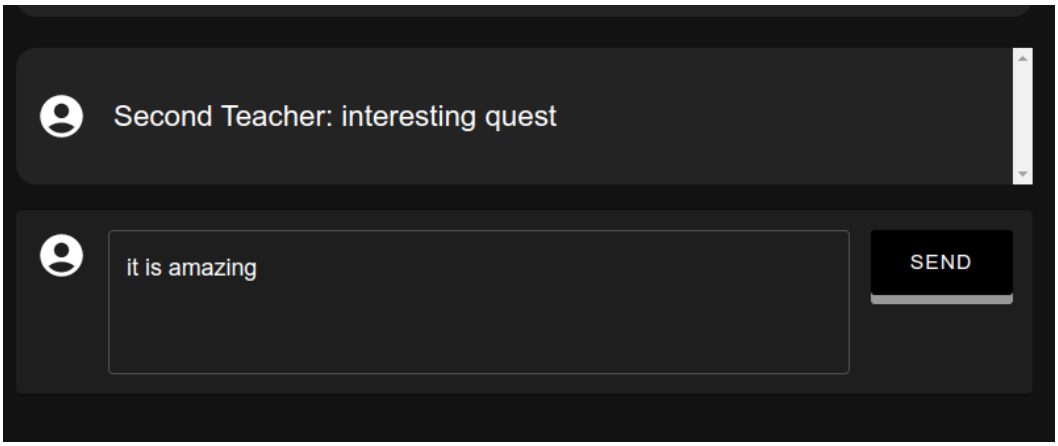


Рисунок 4.30 – Перелік та форма для додавання коментарів

Для залишення коментарю слід ввести бажаний текст у поле для відправки та натуснути кнопку “SEND”. Перелік автоматично оновиться з доданим коментарем. Результати наведені на рис. 4.31.

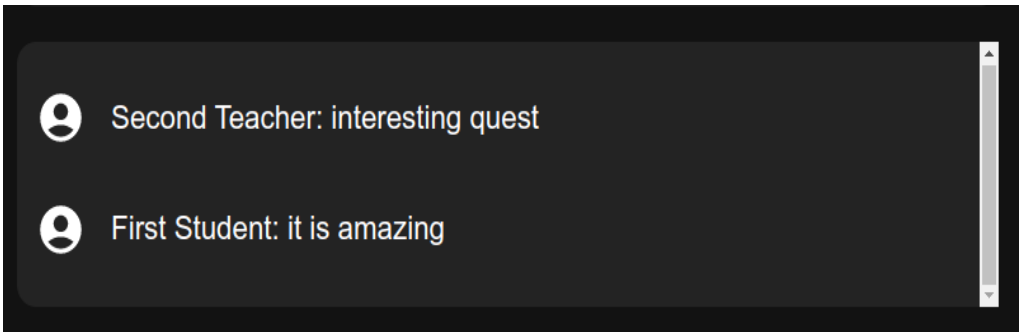


Рисунок 4.31 – Результати додавання коментарю

Після секції роботи з коментуванням слідує перелік задач квесту (рис. 4.32). Тут наведені усі задачі, що відображають запитання, зображення задачі та відповіді.

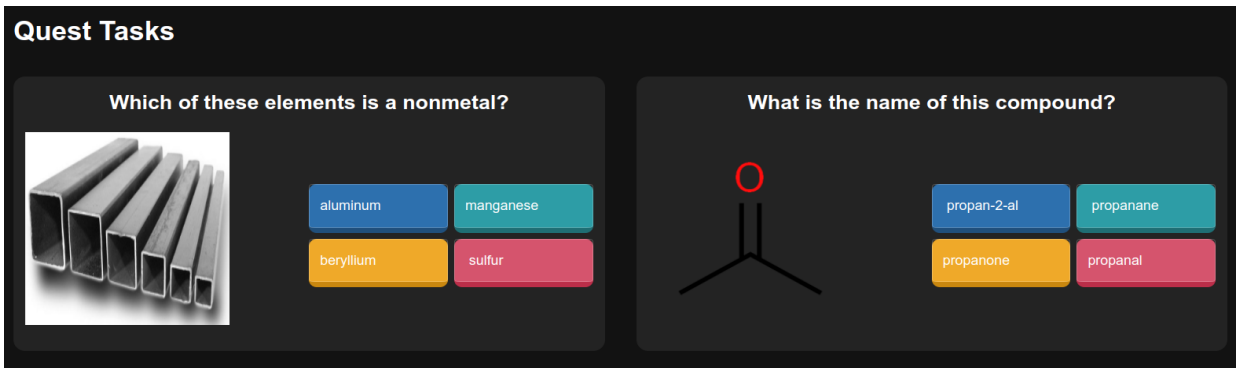


Рисунок 4.32 – Відображення задач квесту

В кінці сторінки розміщено картки запропонованих для перегляду квестів від автора даного квесту (рис. 4.33).

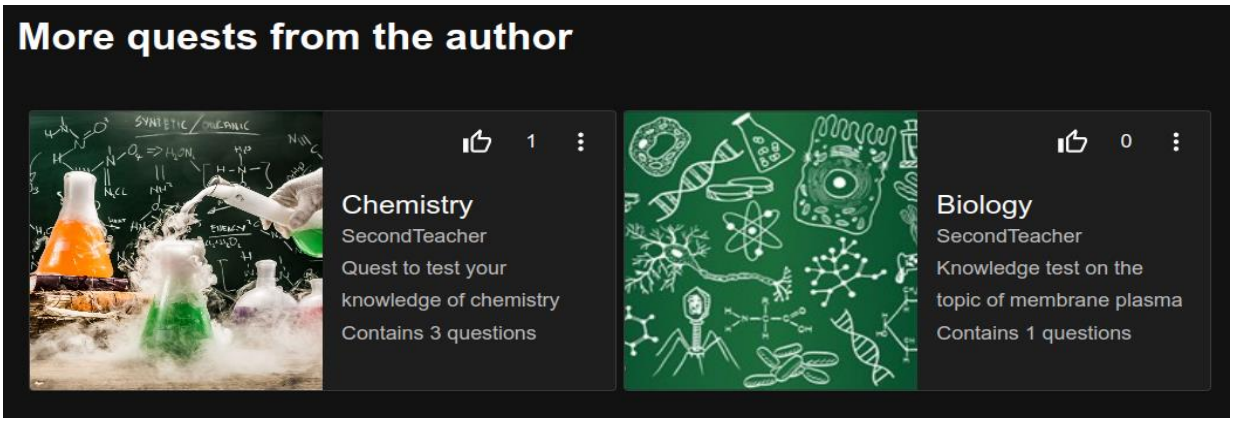


Рисунок 4.33 – Запропоновані квести до перегляду

Таким чином, користувачу надається можливість знайти для себе більше квестів від даного автору, при умові уподобання одного з них.

4.2 Рекомендації з розвитку та покращення

При подальшому розвитку розробленого веб-застосунку рекомендується реалізувати спілкування в режимі реального часу для задавання питань та можливості отримати підтримку, обміну думками щодо реалізації інтерактивних квестів. Також слід додати заплановане викладачем проведення

квесту у заданий час, можливість створювати розклади та надсилати повідомлення з нагадуванням. Неодмінною перевагою такого веб-застосунку стане збереження та відновлення прогресу користувачів при проведенні квесту.

ВИСНОВКИ ДО РОЗДІЛУ 4

У даному розділі представлено веб-застосунок для створення інтерактивних навчальних квестів, що був реалізований протягом всього процесу виконання дипломної роботи. Детально описано користувацький інтерфейс, зокрема розміщення елементів, їх функціональність та взаємодію з користувачем. Ретельно розроблена послідовність інструкцій, щодо операцій користування веб-застосунком для двох типів користувачів. Розглянуто необхідні операції перед використанням веб-застосунку, продемонстровано послідовності створення та проведення квестів. Все це забезпечує легкість створення та проведення квестів з метою поліпшення навчального процесу. Веб-застосунок містить ефективні інструменти для додавання питань, встановлення правильності відповідей та налаштування квестів. Визначено можливості покращення розробленого веб-застосунку та наведено рекомендації щодо розвитку програмного забезпечення та функціоналу у майбутньому.

ВИСНОВКИ

У даній дипломній роботі було реалізовано веб-застосунок для створення інтерактивних навчальних квестів. Зважаючи на результати проведення аналізу інструментів і технологій, дослідження їх основних особливостей та недоліків, в розробці використовувались бібліотека React та фреймворк Express.js на платформі Node.js. Причиною розробки веб-застосунку послугувала зміна потреб та очікувань учнів у сучасному освітньому середовищі. Ціллю веб-застосунку є забезпечення викладачів зручним та ефективним інструментом з можливостями для навчання, спілкування, оцінювання та обміну ресурсами між вчителями та учнями.

В першому розділі було проведено порівняльний аналіз існуючих аналогів веб-застосунку. Завдяки цьому було виявлено найкращі технічні рішення, що були враховані при розробці веб-застосунку. Другий розділ охоплював аналіз ключових функціональних переваг та доцільності використання технологій для проектування клієнтської та серверної частин веб-застосунку. Як підсумок, було визначено основний технологічний стек необхідний для розробки кінцевого веб-застосунку. Третій розділ був призначений для опису розробки програмного забезпечення, його функціональних вимог, тут було зазначено всі послідовності процесів, їх особливості та залежності функціоналу. Четвертий розділ описує послідовність інструкцій, щодо операцій користування веб-застосунком.

Реалізований веб-застосунок для створення інтерактивних навчальних квестів забезпечує користувачів функціональною варіативністю для розробки квестів, додаючи мультимедійне наповнення опитувань, налаштування питань, їх типів та визначення балів за складністю. Таким чином, розроблений продукт відповідає поставленим вимогам, має зручний користувацький інтерфейс та можливість покращення у майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. WHAT IS A WEB APPLICATION? [Електронний ресурс] – Режим доступу до ресурсу: https://www.stackpath.com/edge-academy/what-is-a-web-application/.
2. What is Kahoot!? [Електронний ресурс] – Режим доступу до ресурсу: https://kahoot.com/what-is-kahoot/.
3. Найкращі безкоштовні альтернативи Kahoot для проведення конкурсів [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://guiasdigitales.net/uk/alternativas-gratis-a-kahoot/.
4. 7 найкращих альтернатив Kahoot | Такі ігри, як Kahoot у 2023 році [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://ahaslides.com/uk/blog/similar-alternatives-to-kahoot/.
5. Best Free Kahoot! Alternatives for Any Situation [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.mentimeter.com/blog/stand-out-get-ahead/best-kahoot-alternative/.
6. Створюємо тести онлайн [Електронний ресурс] – Режим доступу до ресурсу: https://uk.soringpcrepair.com/how-to-create-a-test-online/.
7. СЕРВІСИ СТВОРЕННЯ ІНТЕРАКТИВНИХ ЗАВДАНЬ У ІНОЗЕМНИХ МОВАХ (ДЕМОНСТРАЦІЯ СЕРВІСУ НА ПРИКЛАДІ LEARNINGAPPS) [Електронний ресурс] // Київський університет імені Бориса Грінченка, м. Київ. – 2016. – Режим доступу до ресурсу: https://core.ac.uk/reader/78038960/.
8. Heonsik Joo. A Study on Understanding of UI and UX, and Understanding of Design According to User Interface Change [Електронний ресурс] / Heonsik Joo // *Division of Computer and Mechatronics Engineering, Sahmyook University, Hwarang –ro 815, Korea. – 2017. – Режим доступу до ресурсу: https://www.ripublication.com/ijaer17/ijaerv12n20_96.pdf.

9. Beautiful people at Stack Overflow. HTML5 Notes for Professionals / beautiful people at Stack Overflow. – United States of America: Goalkicker.com, 2018. – 124 с.
10. Jon Duckett. HTML and CSS: Design and Build Websites / Jon Duckett. – Ogden, UT, U.S.A: Wiley, 2011. – 512 с.
11. Dr. Axel Rauschmayer. JavaScript for impatient programmers / Dr. Axel Rauschmayer. – Dallas, TX, U.S.A.: Independently published, 2019. – 526 с.
12. Anthony Accomazzo. Fullstack React: The Complete Guide to ReactJS and Friends / Anthony Accomazzo, Nate Murray, Ari Lerner. – Austin, TX, U.S.A.: Fullstack.io, 2017. – 836 с.
13. Hassan Djirdeh. Fullstack Vue The Complete Guide to Vue.js and Friends / Hassan Djirdeh, Nate Murray, Ari Lerner. – San Francisco, California: Fullstack.io, 2018. – 439 с.
14. Joseph D. Booth. Angular Succinctly / Joseph D. Booth. – Morrisville, NC, USA: Syncfusion Inc., 2019. – 124 с.
15. Ethan Brown. Web Development with Node and Express, 2nd Edition / Ethan Brown. – United States of America: O'Reilly Media, Inc., 2019. – 343 с.
16. Adrian Holovaty. The Definitive Guide to Django: Web Development Done Right 2nd ed. Edition / Adrian Holovaty, Jacob Kaplan-Moss. – CA, United States: Apress, 2009. – 572 с.
17. Boris Dinkevich. The Complete Redux Book: Everything you need to build real projects with Redux / Boris Dinkevich, Ilya Gelman. – British Columbia, Canada: Leanpub, 2018. – 211 с.
18. Kristina Chodorow. MongoDB: The Definitive Guide 1st Edition / Kristina Chodorow, Michael Dirolf. – Beijing: O'Reilly Media, Inc., 2010. – 216 с.
19. WebStorm Features [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.jetbrains.com/webstorm/features/.

ДОДАТОК 1

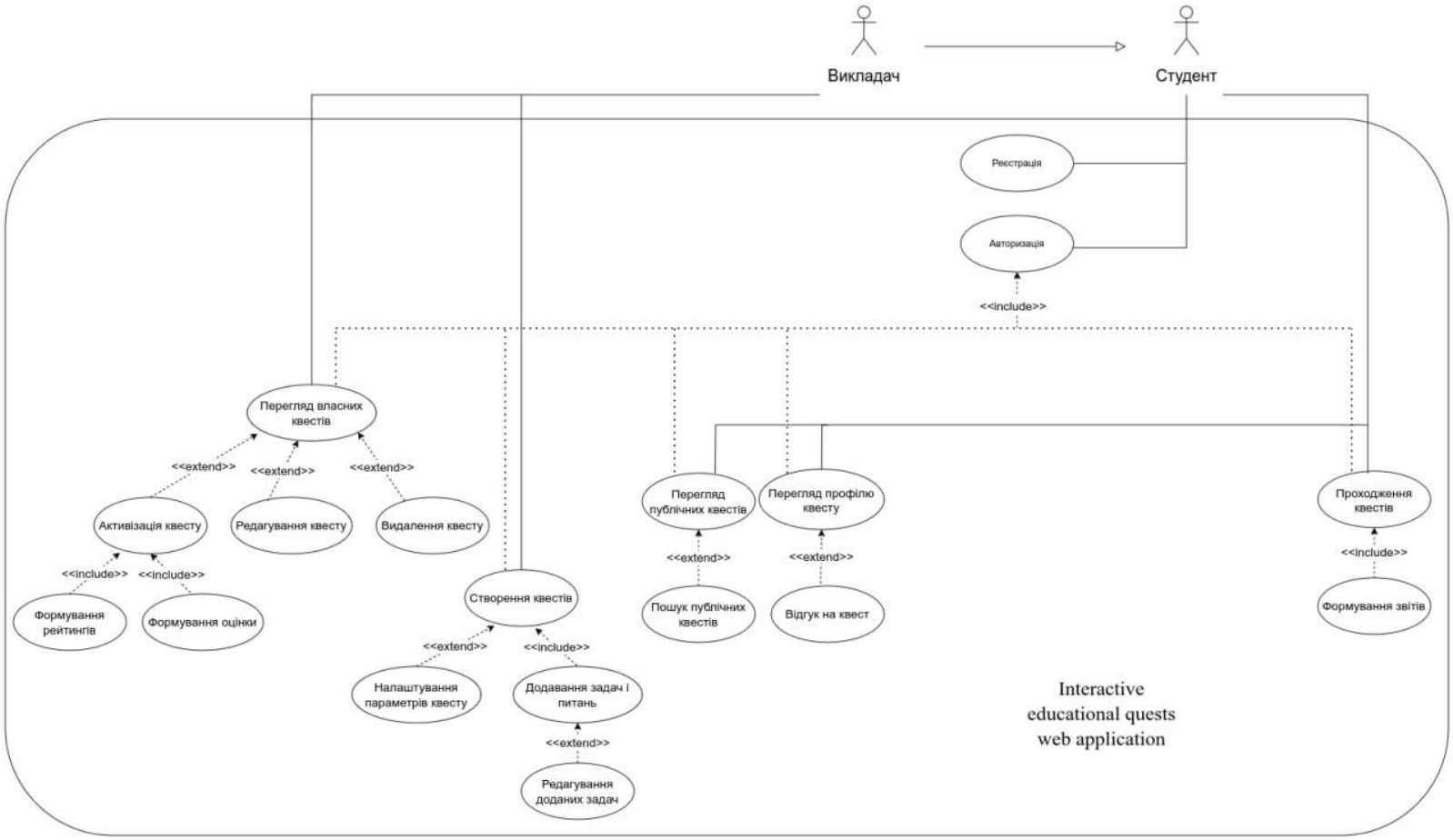
Веб-застосунок для створення інтерактивних навчальних квестів

Діаграма прецедентів (структурна схема)

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2023 р



Зм.	Арк.	Прізвище	Підп.	Дата	ІАЛЦ 467200.004 Д1		
Розроб.		Іванов Р. О.			Веб-застосунок для створення інтерактивних навчальних квестів Діаграма прецедентів (Структурна схема)	Літ.	Арк. 1 / Аркушів 1
Перевірив.		Ткаченко В. В.				КПІ ім. Ігоря Сікорського, ФІОТ, ІО-92	

ДОДАТОК 2

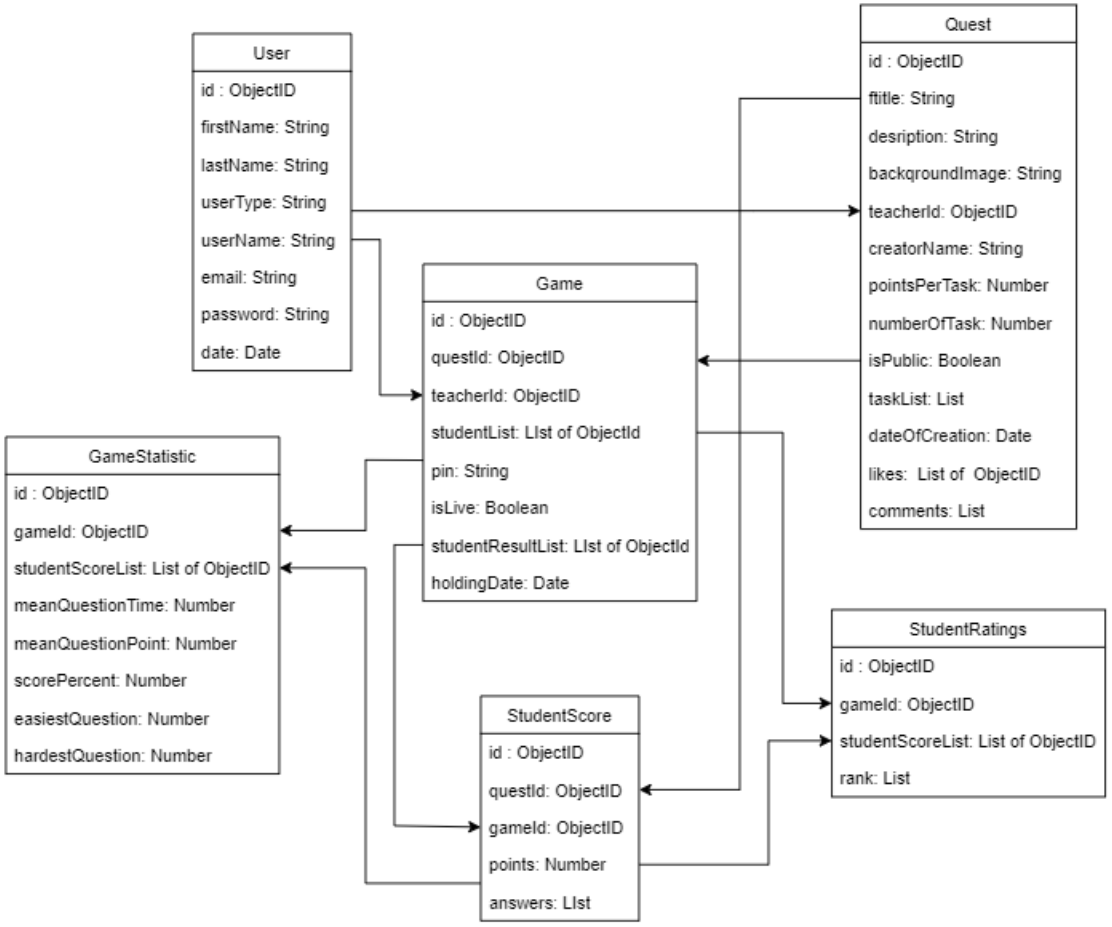
Веб-застосунок для створення інтерактивних навчальних квестів

Схема бази даних (функціональна схема)

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2023 р



Зм.	Арк.	Прізвище	Підп.	Дата	ІАЛЦ 467200.005 Д2		
Розроб.		Іванов Р. О.			Веб-застосунок для створення інтерактивних навчальних квестів Схема бази даних (Функціональна схема)	Літ.	Арк. 1 / Аркушів 1
Перевірив.		Ткаченко В. В.				КПІ ім. Ігоря Сікорського, ФІОТ, ІО-92	

ДОДАТОК 3

Веб-застосунок для створення інтерактивних навчальних квестів

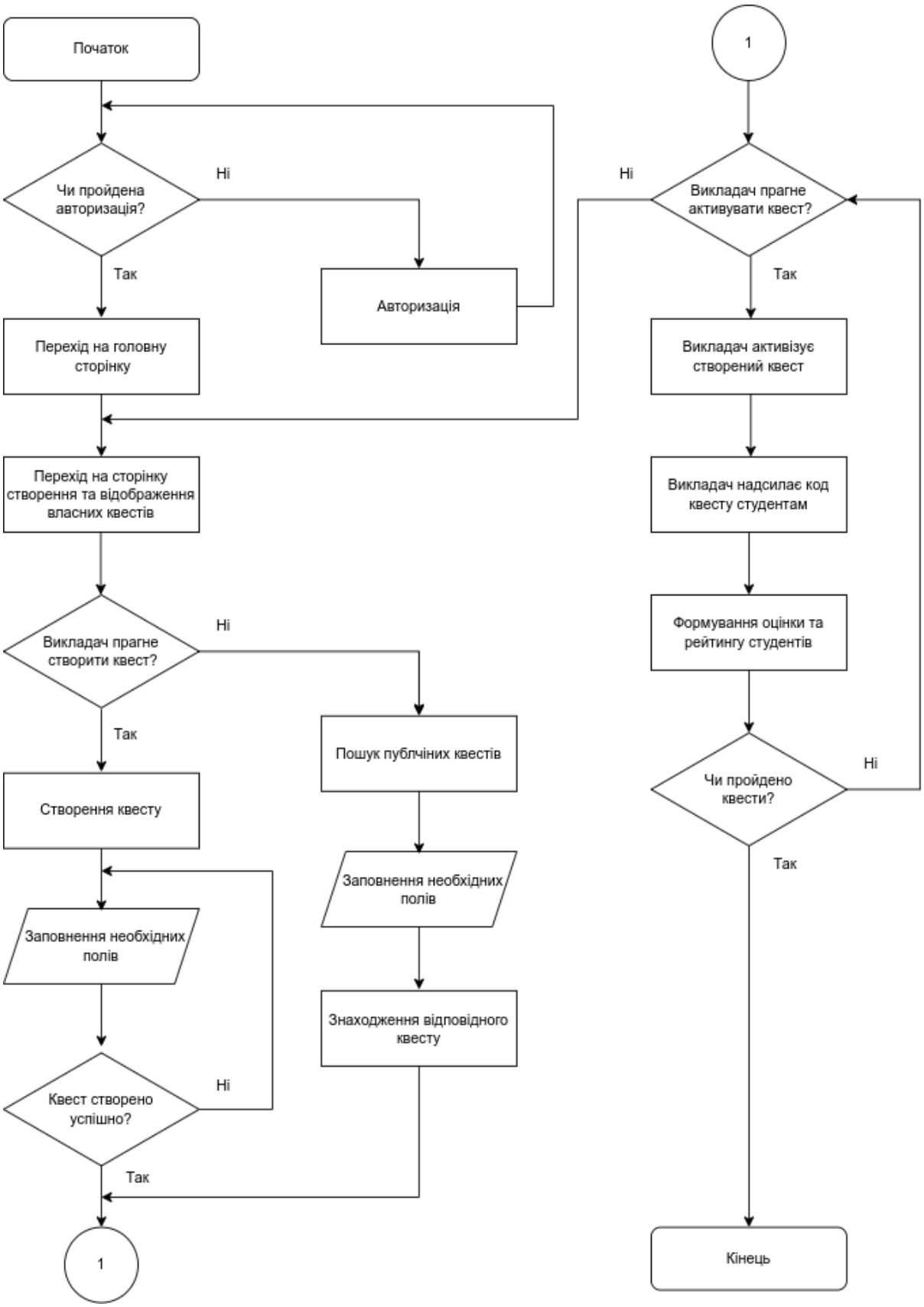
Алгоритм дій програмного забезпечення

(принципова схема)

ІАЛЦ.467200.006 Д3

Аркушів 1

Київ 2023 р.



					ІАЛЦ.467200.006 Д3	
		№ докум.	Підпис	Дата		
Розробив		Іванов Р. О.			**Веб-застосунок для створення інтерактивних навчальних квестів** Алгоритм дій програмного забезпечення (принципова схема)	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Ткаченко В. В.				
Н. Контр.		Виноградов Ю.М.				КПІ ім. Ігоря Сікорського, ФІОТ, ІО-92
Затвердив						

ДОДАТОК 4

Веб-застосунок для створення інтерактивних навчальних квестів

Текст програмного коду

ІАЛЦ.467200.007 Д4

Аркушів 20

Київ 2023 р

QuestCreate.js

```
import { Box, Grid } from '@mui/material'
import React, { useEffect, useState } from 'react'
import { useDispatch, useSelector } from 'react-redux'
import { getQuest, updateQuest } from '../actions/quest'
import { useNavigate, useParams } from 'react-router-dom'
import { useStyles } from '../components/questCreate/styles'
import QuestionSettings from '../components/questCreate/QuestionSettings'
import QuestSettings from '../components/questCreate/QuestSettings'
import AddQuestion from '../components/questCreate/AddQuestion'
import QuestionDisplay from '../components/questCreate/QuestionDisplay'

function QuestCreator() {
  const dispatch = useDispatch()
  const user = JSON.parse(localStorage.getItem('profile'))
  const history = useNavigate()
  const [isQuestOptionsVisible, setIsQuestOptionsVisible] = useState(false)
  const [isQuestionDataSave, setIsQuestionDataSave] = useState(false)
  const [questionType, setQuestionType] = useState('Quiz')
  const [correctAmount, setCorrectAmount] = useState(1)
  const [answerAmount, setAnswerAmount] = useState(4)
  const [isPickCorrectAnswers, setIsPickCorrectAnswers] = useState({
    answersIndexes: [],
    isPick: false,
  })
  const classes = useStyles()
  const questionStructure = {
    questionType: 'Quiz',
    pointType: 'Standard',
    answerTime: 5,
    backgroundImage: '',
    question: '',
    answerList: [
      { answerNumber: '1', answer: '', isCorrect: false },
      { answerNumber: '2', answer: '', isCorrect: false },
      { answerNumber: '3', answer: '', isCorrect: false },
      { answerNumber: '4', answer: '', isCorrect: false },
    ],
    questionNumber: 1,
  }
  const [questionData, setQuestionData] = useState(questionStructure)

  const [questData, setQuestData] = useState({
    title: '',
    creatorName: `${user?.result.firstName} ${user?.result.lastName}`,
    backgroundImage: '',
    description: '',
    keywords: '',
    pointsPerQuestion: 1,
    numberOfQuestions: 0,
    isPublic: true,
    questionList: [],
  })

   const userId = useSelector(state =>
       state.questReducer.quests[state.questReducer.quests.length - 1]._id,
```

```
)
const { userId } = useParams()

const quest = useSelector(state => state.questReducer.quest)

useEffect(() => {
  dispatch(getQuest(userId))
}, [userId, dispatch])

useEffect(() => {
  if (quest) {
    setQuestData(quest)
  }
}, [quest])

const showQuestOptions = () => {
  setIsQuestOptionsVisible(!isQuestOptionsVisible)
}

const verifyAnswers = () => {
  return questionData.answerList.every(answerData => answerData.answer !== '' if )
}

const verifyQuestion = () => {
  return questionData.question !== ''
}
const verifyIsCorrectAnswerPick = () => {
  return isPickCorrectAnswers.isPick
}

const handleQuestionSubmit = () => {
  if (!verifyAnswers()) {
    alert('Please, fill all answers inputs')
  } else if (!verifyIsCorrectAnswerPick()) {
    alert(`Please, pick ${correctAmount} right answers`)
  } else if (!verifyQuestion()) {
    alert('Please, fill question input')
  } else {
    setIsQuestionDataSave(true)

    if (
      questData.questionList.find(
        question => question.questionNumber === questionData.questionNumber,
      )
    ) {
      setQuestData(prevState => ({
        ...prevState,
        questionList: [
          ...prevState.questionList.slice(0, questionData.questionNumber - 1),
          questionData,
          ...prevState.questionList.slice(
            questionData.questionNumber,
            prevState.questionList.length,
          ),
        ],
      }))
    } else {
      setQuestData({
        ...questData,
        questionList: [...questData.questionList, questionData],
      })
    }
  }
}
```

```
const handleQuestSubmit = () => {
  dispatch(updateQuest(quest._id, questData, history))
}

const handleQuestionChange = e => {
  setQuestionData({ ...questionData, [e.target.name]: e.target.value })
}

const handleQuestChange = e => {
  setQuestData({ ...questData, [e.target.name]: e.target.value })
}

return (
  <Box className={classes.root}>
    <Grid container spacing={2}>
      <Grid item xs={8}>
        <Grid item xs container direction="column" spacing={2}>
          <Grid item xs={5}>
            <QuestionSettings
              isQuestOptionsVisible={isQuestOptionsVisible}
              questionData={questionData}
              handleQuestionChange={handleQuestionChange}
              setQuestionType={setQuestionType}
              correctAmount={correctAmount}
              setCorrectAmount={setCorrectAmount}
              answerAmount={answerAmount}
              setAnswerAmount={setAnswerAmount}
              showQuestOptions={showQuestOptions}
              handleQuestionSubmit={handleQuestionSubmit}
              questionType={questionType}
              setQuestionData={setQuestionData}
              questionStructure={questionStructure}
              isPickCorrectAnswers={isPickCorrectAnswers}
              setIsPickCorrectAnswers={setIsPickCorrectAnswers}
            />
            <QuestSettings
              isQuestOptionsVisible={isQuestOptionsVisible}
              questData={questData}
              setQuestData={setQuestData}
              handleQuestChange={handleQuestChange}
              handleQuestSubmit={handleQuestSubmit}
              showQuestOptions={showQuestOptions}
            />
          </Grid>
          <AddQuestion
            questionData={questionData}
            setQuestionData={setQuestionData}
            handleQuestionChange={handleQuestionChange}
            answerAmount={answerAmount}
            correctAmount={correctAmount}
            isPickCorrectAnswers={isPickCorrectAnswers}
            setIsPickCorrectAnswers={setIsPickCorrectAnswers}
          />
        </Grid>
      </Grid>
      <QuestionDisplay
        questData={questData}
        setQuestData={setQuestData}
        setQuestionData={setQuestionData}
        isQuestionDataSave={isQuestionDataSave}
        setIsQuestionDataSave={setIsQuestionDataSave}
        setIsQuestOptionsVisible={setIsQuestOptionsVisible}
      />
    </Grid>
```

```
    </Box>
  )
}

export default QuestCreator
```

QuestionSettings.js

```
import React from 'react'
import {
  Box,
  FormControl,
  Grid,
  IconButton,
  MenuItem,
  Select,
  TextField,
  Typography,
} from '@mui/material'
import SettingsTwoToneIcon from '@mui/icons-material/SettingsTwoTone'
import CheckBoxTwoToneIcon from '@mui/icons-material/CheckBoxTwoTone'
import { useStyles } from './styles'

function QuestionSettings({
  isQuestOptionsVisible,
  questionData,
  handleQuestionChange,
  correctAmount,
  showQuestOptions,
  handleQuestionSubmit,
  questionType,
  setQuestionType,
  answerAmount,
  setAnswerAmount,
  setCorrectAmount,
  setQuestionData,
  questionStructure,
  isPickCorrectAnswers,
  setIsPickCorrectAnswers,
}) {
  const classes = useStyles()

  const changeQuestionType = e => {
    setQuestionType(e.target.value)
    if (e.target.value === 'Quiz') {
      setAnswerAmount(4)
      setCorrectAmount(1)
    } else if (e.target.value === 'OpenQuestion') {
      setAnswerAmount(1)
      setCorrectAmount(1)
    } else if (e.target.value === 'TrueOrFalse') {
      setAnswerAmount(2)
      setCorrectAmount(1)
    }
  }

  const setMaxCorrectAmount = e => {
    setCorrectAmount(e.target.value)
    questionData.answerList.forEach(answer => (answer.isCorrect = false))
    setIsPickCorrectAnswers({
      ...isPickCorrectAnswers,
      answersIndexes: [],
    })
```

```
}

const setMaxAnswerAmount = currentAmount => {
  setAnswerAmount(currentAmount)
  setCorrectAmount(1)

  if (currentAmount < questionData.answerList.length) {
    setQuestionData(prevState => ({
      ...prevState,
      answerList: [
        ...prevState.answerList.filter((_, index) => index < currentAmount),
      ],
    }))
  } else if (currentAmount > questionData.answerList.length) {
    setQuestionData(prevState => ({
      ...prevState,
      answerList: [
        ...prevState.answerList,
        questionStructure.answerList[currentAmount - 1],
      ],
    }))
  }
}

return (
  <Grid
    className={classes.settings}
    container
    spacing={2}
    style={{
      display: isQuestOptionsVisible ? 'none' : 'flex',
    }}>
    <Grid item xs={5}>
      <Box>
        <Typography variant="h4" sx={{ marginBottom: '15px' }}>
          Quest Type
        </Typography>
        <FormControl fullWidth>
          <Select
            value={questionData.questionType}
            name="questionType"
            onChange={e => {
              handleQuestionChange(e)
              changeQuestionType(e)
            }}
            autoWidth>
            <MenuItem value="Quiz">Quiz</MenuItem>
            <MenuItem value="OpenQuestion">Open Question</MenuItem>
            <MenuItem value="TrueOrFalse">True/False</MenuItem>
          </Select>
        </FormControl>
      </Box>
    </Grid>
    <Grid item xs={2}>
      <Box>
        <Typography variant="h4" sx={{ marginBottom: '15px' }}>
          Time limit
        </Typography>
        <FormControl fullWidth>
          <Select
            value={questionData.answerTime}
            name="answerTime"
            onChange={handleQuestionChange}
            autoWidth>
```

```
            <MenuItem value={5}>5</MenuItem>
            <MenuItem value={10}>10</MenuItem>
            <MenuItem value={15}>15</MenuItem>
            <MenuItem value={20}>20</MenuItem>
          </Select>
        </FormControl>
      </Box>
    </Grid>
    <Grid item xs={5}>
      <Box>
        <Typography variant="h4" sx={{ marginBottom: '15px' }}>
          Points
        </Typography>
        <FormControl fullWidth>
          <Select
            value={questionData.pointType}
            name="pointType"
            onChange={handleQuestionChange}
            autoWidth>
            <MenuItem value="Standard">Standard</MenuItem>
            <MenuItem value="BasedOnTime">BasedOnTime</MenuItem>
          </Select>
        </FormControl>
      </Box>
    </Grid>

    {questionType === 'Quiz' && (
      <Grid item xs={4}>
        <Box>
          <Typography variant="h4" sx={{ marginBottom: '15px' }}>
            Answers amount
          </Typography>
          <TextField
            type="number"
            InputProps={{
              inputProps: {
                max: 4,
                min: 2,
              },
            }}
            value={answerAmount}
            onChange={e => setMaxAnswerAmount(e.target.value)}></TextField>
        </Box>
      </Grid>
    )}

    {questionType === 'Quiz' && (
      <Grid item xs={4}>
        <Box>
          <Typography variant="h4" sx={{ marginBottom: '15px' }}>
            Correct answers
          </Typography>
          <TextField
            type="number"
            InputProps={{
              inputProps: {
                max: answerAmount - 1,
                min: 1,

              },
            }}
            value={correctAmount}
            onChange={setMaxCorrectAmount}></TextField>
        </Box>
```

```
        </Grid>
      )}
      <Grid
        item
        xs={12}
        sx={{
          display: 'flex',
          alignItems: 'flex-end',
          justifyContent: 'flex-end',
        }}>
        <Box>
          <IconButton onClick={showQuestOptions}>
            <SettingsTwoToneIcon />
          </IconButton>
        </Box>
        <Box>
          <IconButton onClick={handleQuestionSubmit}>
            <CheckBoxTwoToneIcon />
          </IconButton>
        </Box>
      </Grid>
    </Grid>
  )
}
```

QuestSettings.js

```
import React from 'react'
import {
  Box,
  FormControl,
  FormControlLabel,
  Grid,
  IconButton,
  Radio,
  RadioGroup,
  TextField,
  Typography,
} from '@mui/material'
import SettingsTwoToneIcon from '@mui/icons-material/SettingsTwoTone'
import CheckBoxTwoToneIcon from '@mui/icons-material/CheckBoxTwoTone'
import { useStyles } from './styles'

function QuestSettings({
  isQuestOptionsVisible,
  questData,
  setQuestData,
  handleQuestChange,
  handleQuestSubmit,
  showQuestOptions,
}) {
  const classes = useStyles()
  return (
    <Grid
      className={classes.settings}

      container
      spacing={2}
      style={{
        display: isQuestOptionsVisible ? 'flex' : 'none',
      }}>
      <Grid item xs={4}>
        <Box>
```

```
      <Typography variant="h4" sx={{ marginBottom: '15px' }}>
        Title
      </Typography>
      <TextField
        value={questData.title}
        name="title"
        onChange={handleQuestChange}></TextField>
    </Box>
  </Grid>
  <Grid item xs={4}>
    <Box>
      <Typography variant="h4" sx={{ marginBottom: '15px' }}>
        Description
      </Typography>
      <TextField
        value={questData.description}
        name="description"
        onChange={handleQuestChange}></TextField>
    </Box>
  </Grid>
  <Grid item xs={4}>
    <Box>
      <Typography variant="h4" sx={{ marginBottom: '15px' }}>
        Points
      </Typography>
      <TextField
        type="number"
        min={1}
        value={questData.pointsPerQuestion}
        name="pointsPerQuestion"
        onChange={handleQuestChange}></TextField>
    </Box>
  </Grid>
  <Grid item xs={4}>
    <Box>
      <FormControl>
        <Typography variant="h4">Access</Typography>
        <RadioGroup
          aria-labelledby="demo-radio-buttons-group-label"
          value={questData.isPublic}
          name="isPublic"
          onChange={handleQuestChange}>
          <FormControlLabel
            value="true"
            control={<Radio />}
            label="Public"
          />
          <FormControlLabel
            value="false"
            control={<Radio />}
            label="Private"
          />
        </RadioGroup>
      </FormControl>
    </Box>
  </Grid>
  <Grid item xs={4}>

    <Box>
      <Typography variant="h4" sx={{ marginBottom: '15px' }}>
        Background Image
      </Typography>
      <TextField
        size="small"
```

```
            type="file"
            multiple={false}
            name="backgroundImage"
            onChange={e => {
              const reader = new FileReader()
              reader.readAsDataURL(e.target.files[0])
              reader.onload = () => {
                setQuestData({
                  ...questData,
                  [e.target.name]: reader.result,
                })
              }
            }}
          />
        </Box>
      </Grid>
      <Grid
        item
        xs={4}
        sx={{
          display: 'flex',
          alignItems: 'flex-end',
          justifyContent: 'flex-end',
        }}>
        <Box>
          <IconButton onClick={showQuestOptions}>
            <SettingsTwoToneIcon />
          </IconButton>
        </Box>
        <Box>
          <IconButton onClick={handleQuestSubmit}>
            <CheckBoxTwoToneIcon />
          </IconButton>
        </Box>
      </Grid>
    </Grid>
  )
}

export default QuestSettings
```

AddQuestion.js

```
import React, { useEffect } from 'react'
import { useStyles } from './styles'
import {
  Box,
  FormControl,
  Grid,
  IconButton,
  TextField,
  Typography,
} from '@mui/material'
import DeleteOutlineTwoToneIcon from '@mui/icons-material/DeleteOutlineTwoTone'
import CloudDownloadTwoToneIcon from '@mui/icons-material/CloudDownloadTwoTone'
import AnswersList from './AnswersList'

function AddQuestion({
  questionData,
  setQuestionData,

  handleQuestionChange,
  answerAmount,
```

```
  correctAmount,
  isPickCorrectAnswers,
  setIsPickCorrectAnswers,
}) {
  const classes = useStyles()

  const updateAnswer = (answer, index) => {
    const newAnswerList = questionData.answerList.map(data =>
      data.answerNumber === index ? { ...data, answer: answer } : data,
    )
    setQuestionData({
      ...questionData,
      answerList: newAnswerList,
    })
  }
  const updateCorrectAnswer = index => {
    if (isPickCorrectAnswers.answersIndexes.indexOf(index) !== -1) {
      setIsPickCorrectAnswers(prevState => ({
        answersIndexes: prevState.answersIndexes.filter(
          currIndex => currIndex !== index,
        ),
        isPick: false,
      }))
      const newAnswerList = questionData.answerList.map(data =>
        data.answerNumber === index
          ? { ...data, isCorrect: !data.isCorrect }
          : data,
      )
      setQuestionData({
        ...questionData,
        answerList: newAnswerList,
      })
    } else {
      if (!isPickCorrectAnswers.isPick) {
        setIsPickCorrectAnswers({
          ...isPickCorrectAnswers,
          answersIndexes: [...isPickCorrectAnswers.answersIndexes, index],
        })
        const newAnswerList = questionData.answerList.map(data =>
          data.answerNumber === index
            ? { ...data, isCorrect: !data.isCorrect }
            : data,
        )
        setQuestionData({
          ...questionData,
          answerList: newAnswerList,
        })
      }
    }
  }

  useEffect(() => {
    isPickCorrectAnswers.answersIndexes.length >= correctAmount
      ? setIsPickCorrectAnswers({ ...isPickCorrectAnswers, isPick: true })
      : setIsPickCorrectAnswers({ ...isPickCorrectAnswers, isPick: false })
  }, [correctAmount, isPickCorrectAnswers, setIsPickCorrectAnswers])

  return (

    <Grid item xs={7} className={classes.addQuestion}>
      <Grid item container spacing={2}>
        <Grid item xs={12}>
          <Box
            sx={{
```

```
        display: 'flex',
        alignItems: 'center',
        justifyContent: 'flex-end',
      }}>
      {questionData.backgroundImage === '' || (
        <IconButton
          onClick={() => {
            setQuestionData({
              ...questionData,
              backgroundImage: '',
            })
          }}>
          <DeleteOutlineTwoToneIcon fontSize="medium" />
        </IconButton>
      )}
    </Box>
    <FormControl
      sx={{
        display: 'flex',
        alignItems: 'center',
      }}
      onClick={() => document.querySelector('#textField').click()}>
      <Box
        sx={{
          display: 'flex',
          alignItems: 'center',
          justifyContent: 'flex-end',
        }}>
        <TextField
          sx={{ visibility: 'hidden' }}
          size="small"
          type="file"
          multiple={false}
          name="backgroundImage"
          id="textField"
          onChange={e => {
            const reader = new FileReader()
            reader.readAsDataURL(e.target.files[0])
            reader.onload = () => {
              setQuestionData({
                ...questionData,
                [e.target.name]: reader.result,
              })
            }
          }}
        />
      </Box>

      <Box
        sx={{
          display: 'flex',
          flexDirection: 'column',
          justifyContent: 'center',
          alignItems: 'center',
          marginBottom: '40px',
        }}>
        {questionData.backgroundImage !== '' ? (

          <img
            width={275}
            height={275}
            src={questionData.backgroundImage}
            alt="question background"
          />
```

```
              ) : (
                <>
                  <Box>
                    <Typography variant="h4" sx={{ marginBottom: '15px' }}>
                      Browse image to upload
                    </Typography>
                  </Box>
                  <Box>
                    <CloudDownloadTwoToneIcon fontSize="large" />
                  </Box>
                </>
              )}
            </Box>
          </FormControl>
        </Grid>
        <Grid item xs={12}>
          <Box
            sx={{
              display: 'flex',
              justifyContent: 'center',
              alignItems: 'center',
            }}>
            <Typography variant="h4">Question</Typography>
          </Box>
        </Grid>
        <Grid item xs={12}>
          <Box
            sx={{
              display: 'flex',
              justifyContent: 'center',
              alignItems: 'center',
            }}>
            <TextField
              name="question"
              placeholder="Write your question"
              value={questionData.question}
              onChange={handleQuestionChange}
              fullWidth
            />
          </Box>
        </Grid>
        {questionData.answerList.map(
          (answer, index) =>
            index < answerAmount && (
              <AnswersList
                key={index}
                answer={answer}
                answerNumber={index}
                answerAmount={answerAmount}
                updateAnswer={updateAnswer}
                updateCorrectAnswer={updateCorrectAnswer}
                isPickCorrectAnswers={isPickCorrectAnswers}
              />
            ),
        )}
      </Grid>
    </Grid>
  )
}
export default AddQuestion
```

QuestionDisplay.js

```
import React from 'react'
```

```
import { Grid, IconButton } from '@mui/material'
import { useStyles } from './styles'
import AddIcon from '@mui/icons-material/Add'
import QuestionsList from './QuestionsList'

function QuestionDisplay({
  isQuestionDataSave,
  setQuestionData,
  questData,
  setQuestData,
  setIsQuestionDataSave,
  setIsQuestOptionsVisible,
}) {
  const classes = useStyles()

  const addNewQuestion = () => {
    resetQuestionData()
    setIsQuestionDataSave(false)
    setIsQuestOptionsVisible(false)
  }

  const resetQuestionData = () => {
    setQuestionData({
      questionType: 'Quiz',
      pointType: 'Standard',
      answerTime: 5,
      backgroundImage: '',
      question: '',
      answerList: [
        { answerNumber: '1', answer: '', isCorrect: false },
        { answerNumber: '2', answer: '', isCorrect: false },
        { answerNumber: '3', answer: '', isCorrect: false },
        { answerNumber: '4', answer: '', isCorrect: false },
      ],
      questionNumber: questData.questionList.length + 1,
    })
  }

  const pickQuestion = questionNumber => {
    const question = questData.questionList.find(
      question => question.questionNumber === questionNumber,
    )
    setQuestionData(question)
  }

  const deleteQuestion = questionNumber => {
    setQuestData(prevState => ({
      ...prevState,
      questionList: [
        ...prevState.questionList.slice(0, questionNumber - 1),
        ...prevState.questionList.slice(
          questionNumber,
          prevState.questionList.length,
        ),
      ],
    }))
    questData.questionList.forEach(question => {
      if (question.questionNumber > questionNumber) {
        question.questionNumber -= 1
      }
    })

    if (questData.questionList.length > 1 && questionNumber > 1) {
      pickQuestion(questionNumber - 1)
```

```
    } else if (questData.questionList.length > 1 && questionNumber === 1) {
      pickQuestion(1)
    } else {
      resetQuestionData()
    }
  }

  return (
    <Grid
      container
      gap={2}
      xs={4}
      className={classes.displayQuestion}
      sx={{
        alignContent: 'flex-start',
      }}>
      <Grid
        item
        xs={12}
        sx={{
          display: 'flex',
          alignItems: 'flex-start',
          justifyContent: 'flex-end',
        }}>
        <IconButton
          onClick={() => {
            isQuestionDataSave
              ? addNewQuestion()
              : console.log('Save changes, please')
          }}>
          <AddIcon />
        </IconButton>
      </Grid>
      {questData.questionList.length > 0 &&
        questData.questionList.map(question => (
          <QuestionsList
            key={question.questionNumber}
            onClick={() => pickQuestion(question.questionNumber)}
            deleteQuestion={() => deleteQuestion(question.questionNumber)}
            questionNumber={question.questionNumber}
            questionType={question.questionType}
            answerTime={question.answerTime}
            backgroundImage={question.backgroundImage}
          />
        ))}
    </Grid>
  )
}

export default QuestionDisplay
```

AnswersList.js

```
import React from 'react'
import { Box, Grid, IconButton, TextField } from '@mui/material'
import CheckCircleOutlineIcon from '@mui/icons-material/CheckCircleOutline'
import RemoveCircleOutlineIcon from '@mui/icons-material/RemoveCircleOutline'

function AnswersList({
  answer,
  answerNumber,
  answerAmount,
  updateAnswer,
  updateCorrectAnswer,
  isPickCorrectAnswers,
```

```
}) {
  const colors = [
    '#2d70ae',
    '#204f7b',
    '#2d9da6',
    '#1f6e74',
    '#efa929',
    '#c9870f',
    '#d5546d',
    '#bb2e49',
  ]
  return (
    <>
      <Grid
        item
        xs={
          answerAmount % 2 !== 0 && answerAmount - 1 === answerNumber ? 11 : 5
        }>
        <Box>
          <TextField
            sx={{
              backgroundColor: colors[answerNumber * 2],
              borderRadius: '10px',
              borderBottom: `7px solid ${colors[answerNumber * 2 + 1]}`,
            }}
            fullWidth
            value={answer.answer}
            name="answer"
            placeholder={`Answer ${answerNumber + 1}`}
            onChange={e => {
              updateAnswer(e.target.value, (answerNumber + 1).toString())
            }}
          />
        </Box>
      </Grid>
      <Grid
        item
        xs={1}
        sx={{
          display: 'flex',
          justifyContent: 'center',
          alignItems: 'center',
        }}>
        <IconButton
          onClick={() => {
            updateCorrectAnswer((answerNumber + 1).toString())
          }}>
          {answer.isCorrect ? (
            <CheckCircleOutlineIcon />
          ) : (
            <RemoveCircleOutlineIcon />
          )}
        </IconButton>
      </Grid>
    </>
  )
}

export default AnswersList

import React from 'react'
import { Grid, IconButton } from '@mui/material'
import { useStyles } from './styles'
import DeleteOutlineTwoToneIcon from '@mui/icons-material/DeleteOutlineTwoTone'
```

```
function QuestionsList({
  questionNumber,
  questionType,
  answerTime,
  backgroundImage,
  onClick,
  deleteQuestion,
}) {
  const classes = useStyles()
  return (
    <Grid
      className={classes.displayQuestionList}
      onClick={onClick}
      item
      xs={12}>
      <Grid item xs={4}>
        <Grid item xs container direction="column" spacing={2}>
          <Grid item xs={12}>
            Question {questionNumber}
          </Grid>
          <Grid item xs={12}>
            Type: {questionType}
          </Grid>
          <Grid item xs={12}>
            Time: {answerTime}
          </Grid>
        </Grid>
      </Grid>
      <Grid item xs={6}>
        <img
          width={180}
          height={150}
          src={backgroundImage}
          alt="question background"
        />
      </Grid>
      <Grid
        item
        xs={2}
        sx={{
          display: 'flex',
          justifyContent: 'center',
          alignItems: 'center',
        }}>
        <IconButton onClick={deleteQuestion}>
          <DeleteOutlineTwoToneIcon />
        </IconButton>
      </Grid>
    </Grid>
  )
}

export default QuestionsList
```

MyQuests.js

```
import {
  Box,
  FormControl,
  FormControlLabel,
  Grid,
  Radio,
  RadioGroup,
  TextField,
```

```
  Typography,
} from '@mui/material'
import React, { useEffect, useState } from 'react'
import { useDispatch, useSelector } from 'react-redux'
import { useNavigate } from 'react-router-dom'
import { createQuest, getPersonalQuests } from '../actions/quest'
import QuestList from '../components/myQuests/QuestList'
import { useStyles } from '../components/myQuests/styles'
import Carousel from 'react-material-ui-carousel'
import { StyledButton } from '../components/auth/styles'

function MyQuests() {
  const user = JSON.parse(localStorage.getItem('profile'))
  const dispatch = useDispatch()
  const history = useNavigate()
  const classes = useStyles()
  const [questData, setQuestData] = useState({
    title: '',
    creatorName: `${user?.result.userName}`,
    creatorId: `${user?.result._id}`,
    backgroundImage: '',
    description: '',
    keywords: '',
    pointsPerQuestion: 1,
    isPublic: 'false',
    questionList: [],
  })
  useEffect(() => {
    dispatch(getPersonalQuests(user.result._id))
  }, [])

  const quests = useSelector(state => state.questReducer.quests)
  const handleSubmit = () => {
    dispatch(createQuest(questData, history))
  }
  const handleChange = e => {
    setQuestData({ ...questData, [e.target.name]: e.target.value })
  }
  function chunk(arr, size) {
    const result = []
    for (let i = 0; i < Math.ceil(arr.length / size); i++) {
      result.push(arr.slice(i * size, i * size + size))
    }
    return result
  }
  const questsGroup = chunk(quests, 3)

  return (
    <Box className={classes.root}>
      <Grid container spacing={5}>
        <Grid item xs={4}>
          <Typography sx={{ marginBottom: '10px' }} variant="h3">
            Create new quest
          </Typography>
          <Box className={classes.questSettings}>
            <Box sx={{ marginBottom: '10px' }}>
              <Typography variant="h5" sx={{ marginBottom: '10px' }}>
                Title
              </Typography>
              <TextField
                fullWidth
                name="title"
                onChange={handleChange}></TextField>
            </Box>
            <Box sx={{ marginBottom: '10px' }}>
```

```
              <Typography variant="h5" sx={{ marginBottom: '10px' }}>
                Description
              </Typography>
              <TextField
                fullWidth
                name="description"
                onChange={handleChange}></TextField>
            </Box>
            <Box sx={{ marginBottom: '10px' }}>
              <Typography variant="h5" sx={{ marginBottom: '10px' }}>
                Keywords
              </Typography>
              <TextField
                fullWidth
                name="keywords"
                onChange={handleChange}></TextField>
            </Box>
            <Box sx={{ marginBottom: '10px' }}>
              <FormControl>
                <Typography variant="h5">Access</Typography>
                <RadioGroup
                  aria-labelledby="demo-radio-buttons-group-label"
                  value={questData.isPublic}
                  name="isPublic"
                  onChange={handleChange}>
                  <FormControlLabel
                    value="true"
                    control={<Radio />}
                    label="Public"
                  />
                  <FormControlLabel
                    value="false"
                    control={<Radio />}
                    label="Private"
                  />
                </RadioGroup>
              </FormControl>
            </Box>
            <Box sx={{ display: 'flex', justifyContent: 'flex-end' }}>
              <StyledButton onClick={handleSubmit}>Create</StyledButton>
            </Box>
          </Box>
        </Grid>
        <Grid item xs={8}>
          <Typography sx={{ marginBottom: '10px' }} variant="h3">
            My quests
          </Typography>
          <Box className={classes.questSettings}>
            <Carousel
              navButtonsAlwaysInvisible={true}
              interval={'10000'}
              duration={'800'}>
              {questsGroup.map((questsList, index) => (
                <QuestList key={index} questsList={questsList} />
              ))}
            </Carousel>
          </Box>
        </Grid>
      </Grid>
    </Box>
  )
}

export default MyQuests
```

Quest.js

```
import {
  Box,
  Card,
  CardActions,
  CardContent,
  CardMedia,
  Grid,
  IconButton,
  Typography,
} from '@mui/material'
import React from 'react'
import DeleteOutlineTwoToneIcon from '@mui/icons-material/DeleteOutlineTwoTone'
import PlayCircleOutlineTwoToneIcon from '@mui/icons-material/PlayCircleOutlineTwoTone'
import MoreVertTwoToneIcon from '@mui/icons-material/MoreVertTwoTone'
import { useStyles } from './styles'
import { useNavigate } from 'react-router-dom'
import { useDispatch, useSelector } from 'react-redux'
import { activateGame } from '../../actions/game'
import { removeQuest } from '../../actions/quest'

function Quest({ key, quest }) {
  const classes = useStyles()
  const history = useNavigate()
  const dispatch = useDispatch()

  const socket = useSelector(state => state.gameReducer.socket)

  const openQuestCreator = e => {
    history(`/myQuests/${quest._id}`)
  }
  const startGame = async () => {
    let gameData = {
      questId: quest._id,
      teacherId: quest.teacherId,
      isLive: true,
      pin: String(Math.floor(Math.random() * 900000) + 100000),
    }
    const activatedGame = await dispatch(activateGame(gameData, history))
    socket.emit('init-game', activatedGame)
  }

  return (
    <Card className={classes.card}>
      <Grid container spacing={2}>
        <Grid item xs={4}>
          <CardMedia
            component="img"
            image={quest.backgroundImage}
            height="225"
          />
        </Grid>
        <Grid item xs={8}>
          <Box className={classes.cardActions}>
            <CardActions>
              <IconButton onClick={startGame}>
                <PlayCircleOutlineTwoToneIcon />
              </IconButton>
              <IconButton onClick={() => dispatch(removeQuest(quest._id))}>
                <DeleteOutlineTwoToneIcon />
              </IconButton>
              <IconButton onClick={openQuestCreator}>
                <MoreVertTwoToneIcon />
              </IconButton>
```

```
            </CardActions>
          </Box>
          <CardContent>
            <Typography variant="h4">{quest.title}</Typography>
            <Typography variant="subtitle1" color="text.secondary">
              {quest.creatorName}
            </Typography>
            <Typography variant="subtitle1" color="text.secondary">
              {quest.description}
            </Typography>
            <Typography variant="subtitle1" color="text.secondary">
              Contains {quest.numberOfQuestions} questions
            </Typography>
          </CardContent>
        </Grid>
      </Grid>
    </Card>
  )
}

export default Quest
```