КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»

(шифр і назва спеціальності)

«Прикладне програмування»

(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Програмна система навчання та тестування знань»

Виконав _____
(Підпис)
Зорін Артем Олександрович
(прізвище, ім'я, по батькові)

Керівник _Плескач Валентина Леонідівна
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____
_
(Висновок: "До захисту в екзаменаційній комісії ")
Завідувач кафедри _____ Плескач В.Л.
(Підпис) (Прізвище, ініціали)
(Дата)

Київ – 2021

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	заява
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	05.03.2021	
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	09.04.2021	
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	07.05.2021	
9.	Подання роботи у першому варіанті	11.05.2021	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	24.05.2021	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	
14.	Захист кваліфікаційної роботи бакалавра	25.06.2021	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Завдання до дипломної роботи	1
Календарний план дипломної роботи	1
Відомість дипломної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Зміст	2
Перелік скорочень, умовних позначень, термінів	1
Вступ	2
1	8
2	7
3	28
Висновки	2
Перелік використаних джерел	2
Додатки	4

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.	Зорін А.О.			Відомість дипломної роботи	Лист	Листів
Керівн.	Плескач В.Л.					
Н/контр.	Макаренко С.А.					
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ (РЕФЕРАТ)

Дипломна робота: 61 с., 37 рис., 2 таб., 17 джерел, 1 дод.

Метою дипломної роботи є ефективне проведення навчання та тестування знань за допомогою використання створеної програмної системи. Для досягнення мети роботи потрібно вирішити такі завдання:

– Дослідити теоретичні основи побудови програмних систем навчання та тестування знань.

– Проаналізувати програмно-технічні рішення систем навчання та тестування знань.

– Спроектувати, розробити та впровадити програмну систему навчання та тестування знань.

Об’єкт дослідження.

Процеси організації та проведення навчання та тестування знань.

Предмет дослідження.

Програмно-технічні, організаційні засади, принципи, концептуальні підходи до побудови програмної системи навчання та тестування знань.

Методи дослідження.

Теорія систем, системний аналіз, форми та методи застосування клієнт-серверної моделі архітектурного рішення досліджуваної системи, теорія розподілених систем та порівняльний аналіз ефективності використання сучасних сервісів для навчання та тестування знань. Результати дипломної роботи можна використати для розміщення на хостингу, що дасть змогу користувачам отримати доступ до навчання та тестування знань. Для зберігання даних користувача була використана реляційна база даних (БД) PostgresSql. Програмна система була розроблена мовою програмування Java з використанням платформи JavaScript та допоміжних фреймворків для реалізації клієнтської частини системи.

Ключові слова: веб-застосунок, навчання, тестування, сервер, браузер, фреймворк, технологія.

ANOTATION (ABSTRACT)

Thesis: 61 p., 37 pictures, 2 tables, 17 sources, 1 annex.

The purpose of the thesis is to effectively conduct training and testing of knowledge through the use of software system.

To achieve the goal of the work you need to solve the following tasks:

- Investigate the theoretical foundations of building software systems for learning and testing knowledge;
- Analyze software and hardware solutions for training and knowledge testing systems;
- Design, develop and implement a software system for training and testing of knowledge.

Object of study:

Processes of organizing and conducting training and testing of knowledge.

Subject of study:

Software and technical, organizational principles, principles, conceptual approaches to building a software system for learning and testing knowledge.

Research methods:

System analysis, systems theory and application of client-server model of architectural solution of the researched system, theory of distributed systems and comparative analysis of efficiency of use of modern services for testing of users. The results of the thesis can be used for hosting, which will allow users to access training and testing. The PostgreSQL relational database was used to store user data. The software system was developed in the Java programming language using the JavaScript platform and auxiliary frameworks to implement the client side of the site.

Keywords: web application, training, testing, server, browser, framework, technology.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API (Application Programming Interface) – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

CSS (Cascading Style Sheets) – спеціальна мова стилю сторінок , що використовується для опису їхнього зовнішнього вигляду.

HTML (HyperText Markup Language) – мова тегів, якою пишуться гіпертекстові документи для мережі Інтернет.

HTTP (Hyper Text Transfer Protocol) – протокол передачі даних, що використовується в комп'ютерних мережах.

REST – архітектурний стиль взаємодії компонентів розподіленого застосунку в мережі.

IDE (Integrated development environment) – інтегроване середовище розробки.

JSON (JavaScript Object Notation) – текстовий формат обміну даними між комп'ютерами.

PostgreSQL – вільна об'єктно-реляційна система управління базами даних.

DAO (Data Access Objects) – технологія для доступу до даних компанії Microsoft.

RPC (Remote procedure call) – віддалений виклик процедур.

XML (Extensible Markup Language) – розширювана мова розмітки.

SQL (Structured query language) – мова структурованих запитів

URL (Uniform Resource Locator) – уніфікований покажчик ресурсу.

ОС – операційна система.

БД – база даних.

ІКТ – інформаційно-комунікаційні технології.

ПЗ – програмне забезпечення.

МП – мова програмування.

СУБД – система управління базами даних.

ВСТУП

Нині, проведення навчання та тестування за допомогою програмних систем є поширеним явищем, оскільки це дозволяє швидко визначити певний рівень знань користувача, прискорити обробку отриманої інформації та легко охопити великі за обсягом масиви матеріалу. Аналіз зрізу знань, отриманий після навчання та тестування, дозволяє оцінити рівень знань людини, та може знайти застосування незалежно від призначення та сфери діяльності.

Програмні системи дають можливість створювати навчальний контент, додавати відео- і аудіоматеріали, графічні зображення, фото, додавати тести та ставити задачі, що робить процес навчання більш інтерактивним. Тестові опитування застосовують під час навчання, адже вони дозволяють автоматично конструювати статистичні дані з успішності за підсумками здійсненого аналізу пройденого навчання.

Залежно від призначення використання програмної системи, доцільно використовувати різні види навчального контенту та тестових завдань. Різноманітність видів навчального контенту та методів проведення зрізу знань робить такі програмні системи більш привабливими та доцільними для будь-якої сфери, де є необхідність в проведенні навчання та тестування знань.

Тому створення програмної системи ефективної організації навчання та тестування знань в умовах цифрового суспільства є **актуальною науково-прикладною задачею.**

Метою дипломної роботи є ефективна організація навчання та проведення тестування знань за допомогою використання програмної системи.

Для досягнення мети роботи потрібно вирішити такі **завдання**:

– Дослідити теоретичні основи побудови програмних систем навчання та тестування знань.

– Проаналізувати програмно-технічні рішення систем навчання та тестування знань.

– Спроектувати, розробити та впровадити програмну систему навчання та тестування знань.

Об’єктом дослідження дипломної роботи є процеси організації та проведення навчання та тестування знань.

Предметом дослідження є програмно-технічні, організаційні засади, принципи, концептуальні підходи до побудови програмної системи для навчання та тестування знань.

Методами дослідження є системний аналіз і синтез, теорія систем та застосування клієнт-серверної моделі архітектурного рішення досліджуваної системи, теорія розподілених систем та порівняльний аналіз ефективності використання сучасних сервісів для тестування користувачів. Результати дипломної роботи можна використати для розміщення на хостингу, що дасть змогу користувачам отримати доступ до функціоналу системи. Для зберігання даних користувача (профіль, навчальний контент, тестові завдання, результати тестування) була використана система управління реляційними БД PostgreSQL. Веб-сервіс був розроблений мовою програмування Java на фреймворці Spring Boot, та JavaScript з використанням фреймворку React для відображення клієнтської частини.

1 ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ ПРОГРАМНОЇ СИСТЕМИ НАВЧАННЯ ТА ТЕСТУВАННЯ ЗНАНЬ

1.1 Теоретичні основи навчання та тестування знань на базі програмної системи

Нині стрімко зростає тенденція в впровадженні навчання та тестування знань на базі програмних систем та веб-застосунків. Веб-застосунки для навчання та тестування знань значно полегшують та прискорюють процес обробки великого масиву інформації, автоматизуючи перевірку результатів контролю знань, надаючи можливість розміщувати навчальний контент в мережі Інтернет та повністю контролювати всі етапи навчання онлайн.

Навчання онлайн допомагає отримувати знання і набувати нових навичок за допомогою комп'ютера, мобільного телефону або іншого пристрою з підтримкою можливості підключення до мережі. Цей формат навчання ще називають e-learning або "електронне навчання". Завдяки зручності, онлайн навчання стає все більш популярною формою отримання та перевірки знань.

Під час онлайн-навчання користувач отримує навчальні матеріали у форматі відеозапису, прямої трансляції, проходить тестування, обмінюється файлами з викладачем, вирішує задачі тощо. Таке навчання дозволяє повністю зануритися в освітнє середовище і підвищувати рівень кваліфікаційних без потреби фізичної присутності. Незалежно від сфери застосування, отримання знань в мережі Інтернет за допомогою програмних систем навчання та тестування знань, забезпечує легке налагодження цього процесу, та надає ефективні показники в швидкості обробки зрізу знань учнів.

Переваги онлайн-навчання та тестування знань за допомогою програмних систем:

- Свобода доступу, мобільність. Навчатися можна будь-якому місці та в будь-який час, використовуючи пристрої з підтримкою можливості підключення до мережі Інтернет.
- Зниження витрат на навчання.

– Гнучкість навчання. Тривалість і послідовність вивчення матеріалів, повністю адаптується під процес навчання, можливості і потреби.
– Своєчасне, легке та швидке оновлювання навчальних матеріалів. Оновлення або зміна навчального плану, не призводить до великої трудомісткої роботи по редагуванню навчальних матеріалів.
– Можливість чітко визначати критерії оцінки знань. В онлайн-навчанні є можливість виставляти чіткі критерії, за якими оцінюються знання, отримані користувачем в процесі навчання.

Недоліки онлайн-навчання та тестування знань за допомогою програмних систем:

– Відсутній розвиток комунікабельності. При онлайн-навчанні, спілкування між користувачами зводиться до мінімальних показників, що погано впливає на розвиток впевненості та навичок роботи в команді.
– Недолік отримання практичних знань. Під час навчання за спеціальностями, які передбачають велику кількість практичних завдань та їх перевірки, робота програмної системи має погану ефективність.
– Проблема ідентифікації користувача. Важко ідентифікувати особу, що навчається або проходить тестування, спосіб простежити за тим, чи чесно і самостійно користувач здавав тестування – це відео-контроль, що не завжди зручно.
– Недостатня комп'ютерна грамотність. У багатьох країнах особлива потреба в онлайн-навчанні виникає у віддалених районах.

1.2 Поняття, переваги та недоліки веб-застосунку в глобальній мережі Інтернет

Веб-застосунок – це клієнт-серверна програмна система, в якій клієнт взаємодіє з веб-сервером за допомогою браузера.

Веб-застосунок – особливий тип програмної системи, побудований на архітектурі "клієнт-сервер"(рис. 1.1). Їх особливість полягає в тому, що сам веб-застосунок знаходиться і виконується на сервері – клієнт при цьому отримує тільки результати роботи. Робота веб-застосунку грунтується на отриманні запитів від користувача (клієнту), їх обробці і видачі результатів, а сам процес передачі запитів та їх обробки відбувається через мережу Інтернет.

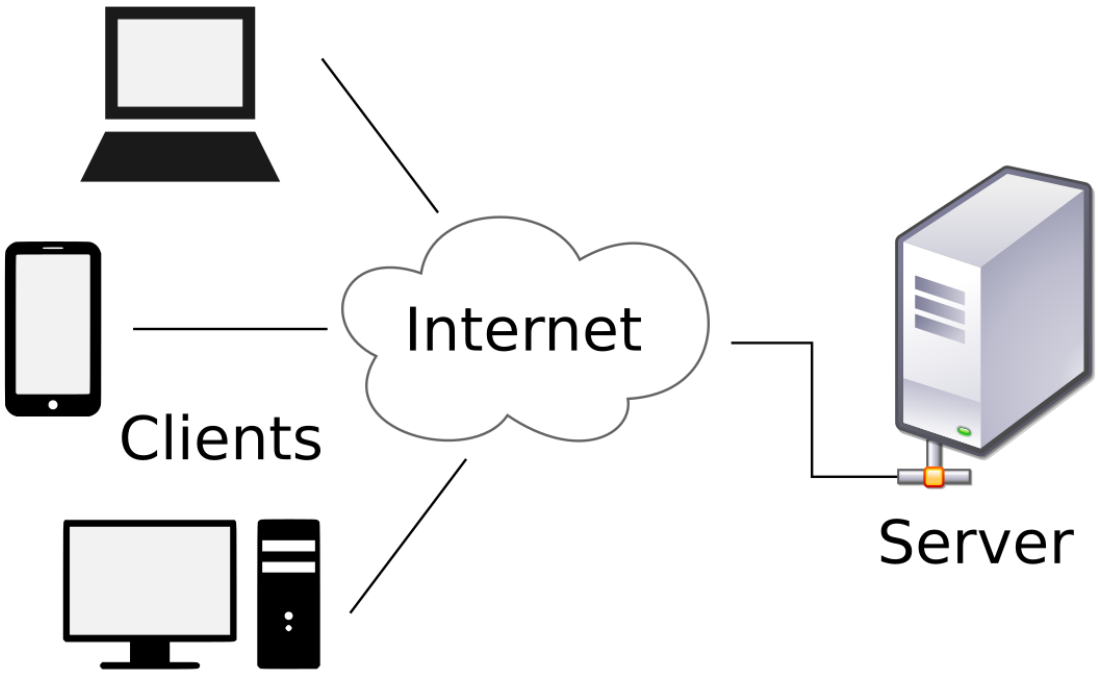

Рисунок 1.1 – Клієнт-серверна архітектура веб-застосунку

Правила взаємодії між сервером та клієнтом називають протоколом обміну (або протоколом взаємодії).

Модель клієнт-серверної архітектури роботи програмної системи визначається розподілом ролей між клієнтською частиною та сервером. Можна відокремити три рівня операцій:

– рівень візуального представлення даних, який являє собою інтерфейс користувача і відповідає за представлення даних користувачу або надання від нього керуючих запитів;
– прикладний рівень, що реалізує логіку серверної частини веб-застосунку і на якому здійснюється необхідна обробка вхідних даних користувача;

– рівень управління даними, який забезпечує можливість зберігання, додавання та видалення даних.

Дволанкова клієнт-серверна архітектура передбачає взаємодію двох програмних модулів клієнтського та серверного.

В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка застосунку та управління даними зосереджена на серверній стороні (клієнтська програма забезпечує тільки візуальне представлення даних);
– модель товстого клієнта, в якій сервер керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта (товстими клієнтами часто також називають пристрої з обмеженою потужністю).

Відображенням результатів запитів, а також прийомом даних від клієнта і їх передачою на сервер зазвичай займається спеціальний застосунок – браузер (Google Chrome, Internet Explorer, Mozilla, Opera тощо). Як відомо, однією з функцій браузера є відображення даних, отриманих з Інтернету, у вигляді сторінки, описаної на мові HTML, отже, результат, який передається сервером клієнтові, повинен бути представлений на цій мові. На стороні сервера веб-застосунок виконується спеціальним програмним забезпеченням (веб-сервером), який і приймає запити клієнтів, обробляє їх, формує відповідь і передає його клієнту.

Для веб-застосунків на стороні сервера можна застосовувати різні технології і будь-які мови програмування. Для клієнта-браузера не важливо, яка ОС налаштована у користувача, в цьому плані веб-застосунки можна вважати універсальними багато-платформовими сервісами.

Веб-застосунки у порівнянні з локальними програмними системами мають низку переваг:

- Простота доступу до застосунку – будь-яка людина, що має пристрій, підключений до мережі Інтернет, може використовувати веб-застосунок у власних цілях;
- Простота розгортання (установки) - на відміну від локальних застосунків веб-застосунки після завершення розробки не вимагають установки на пристроях користувачів. Досить тільки повідомити їм URL-адресу програми. При зміні застосунку все користувачі відразу починають працювати з новою версією;
- Наявність великої кількості навчених користувачів;
- Високий рівень розвитку і надійності мережевих з'єднань та веб-технологій.

Однак, веб-застосунки мають і свої недоліки:

- Слабозв’язна архітектура - відсутність підтримки стану сеансу роботи і затримка при перезавантаженні кожної сторінки. Кожна перевантаження (або оновлення сторінки) викликає помітну затримку, викликану необхідністю встановити HTTP- з'єднання, обробити запит на сервері, передати по мережі у відповідь HTTP-повідомлення і перезавантажити сторінку браузером. Це створює скачки та періодичною очисткою роботи користувача.
- Обмежений набір елементів управління для проектування форм застосунків. Поточна версія мови HTML підтримує тільки обмежений набір елементів управління (текстові елементи, прапорці, списки, що випадають і командні кнопки).

Сучасні веб-застосунки мають важливі архітектурні властивості, серед яких:

- можливість масштабованості веб-системи;
- можливість інтеграції з сторонніми програмними системами;
- розмежування прав доступу до функціоналу;
- зручне розгортання та підтримка системи.

Наявність таких властивостей у веб-застосунку дозволяє використовувати систему максимально ефективно і легко. Наприклад, завдяки властивості масштабованості можна без внесення глобальних змін розширювати веб-систему для роботи з постійно зростаючим числом користувачів, додавати до неї новий функціонал. У свою чергу, завдяки зручному розгортанню процес переходу роботи з тим чи іншим веб-застосунком займає мінімум часу, а в умовах великого завантаження це може бути важливо.

1.3 Інформаційно-комунікаційні технології

Інформатизація суспільства – це перспективний шлях до економічного, соціального та освітнього розвитку суспільства в цілому. Інформатизація освіти спрямовується на формування та розвиток інтелектуального потенціалу нації, удосконалення форм і змісту навчального процесу, впровадження комп'ютерних методів навчання та тестування знань, що надає можливість вирішувати проблеми освіти на більш вищому технічному рівні з урахуванням світових вимог.

Виникнення та розвиток інформаційного суспільства припускає широке застосування інформаційно-комунікаційних технологій в освіті, що визначається багатьма чинниками.

Впровадження інформаційно-комунікаційних технологій у сучасну освіту суттєво прискорює та спрощує процес передачі знань і накопичення технологічного та соціального досвіду людства не тільки від покоління до покоління, а й від однієї людини до іншої. Підвищуючи якість навчання й освіти, ІКТ дають змогу людині успішніше й швидше адаптуватися до навколишнього середовища, до соціальних змін. Це дає кожній людині можливість одержувати необхідні знання як сьогодні, так і в постіндустріальному суспільстві. Активне й ефективне впровадження цих технологій в освіту є важливим чинником створення нової системи освіти, що

відповідає вимогам інформаційного суспільства і процесу модернізації традиційної системи освіти.

Важливість і необхідність впровадження ІКТ у навчання обґрунтовується міжнародними експертами і вченими. Інформаційно-комунікаційні технології торкаються всіх сфер діяльності людини, але, мабуть, найбільш сильний позитивний вплив вони мають на освіту, оскільки відкривають можливості впровадження абсолютно нових методів викладання і навчання.

Новим етапом глобальної технологізації передових країн світу, стала поява сучасних телекомунікаційних мереж та їх інтеграція з інформаційними технологіями, тобто поява ІКТ. Вони стали основою для створення нового інформаційного простору, оскільки об'єднання комп'ютерних систем і глобальних телекомунікаційних мереж зробило можливим створення і розвиток планетарної інфраструктури, що зв'язує нині все людство.

Прикладом успішної реалізації інформаційно-комунікаційних технологій стала поява Інтернету – глобальної комп'ютерної мережі з її практично необмеженими можливостями збирання та збереження інформації, передавання її індивідуально кожному користувачеві.

Розглядаючи елементи складної системи інформаційних технологій навчання, слід наголосити, що важливою умовою успішної інтеграції технологій є професійна підготовка викладачів і фахівців, які здійснюють експлуатацію систем і засобів нової інтегрованої технології навчання. Кожний учасник навчання має володіти необхідною інформаційною грамотністю і розумінням у використанні технологій. У деяких країнах для цього необхідно навіть мати відповідний сертифікат. Наприклад, така вимога є у Великобританії. Введення сертифікатів для учасників процесу навчання дає змогу спростити впровадження інформаційних технологій навчання і підвищити показник ефективності технологій.

Отже, застосування комп'ютерів в освіті привело до появи нового покоління інформаційних освітніх технологій, що дали змогу підвищити

якість навчання. На думку багатьох фахівців, нові інформаційні освітні технології на основі комп'ютерних засобів дають можливість значно підвищити ефективність навчання.

Висновки до розділу

В цьому розділі було проведено аналіз сучасних підходів до навчання за допомогою інформаційно-комунікаційних технологій та значущість інформатизації навчання в суспільстві.

Веб-застосунки працюють на стандартних технологіях та “клієнт-серверній” архітектурі, що робить їх незалежними від програмно-технічних характеристик та можуть використовуватися в будь-яких операційних системах, браузерах та на різних пристроях які матимуть підключення до мережі Інтернет.

Стратегічна цінність веб-застосунків полягає у легкій масштабованості програмної системи, надійному збереженню інформації на сервері та в момент передачі запитів, доступність на будь-яких пристроях, де є підключення до мережі Інтернет.

2 АНАЛІЗ ПРОГРАМНО-ТЕХНІЧНИХ РІШЕНЬ СИСТЕМ НАВЧАННЯ ТА ТЕСТУВАННЯ ЗНАНЬ

2.1 Аналіз наявних рішень програмних систем навчання та тестування знань

В сучасних умовах зростає актуальність онлайнових платформ і сервісів, які допомагають проводити навчання та тестування знань в мережі Інтернет, та в зручний спосіб обробляти та проводити аналіз великого набору даних. Ця задача вирішується за допомогою можливості отримати сучасну платформу, де можна незалежно від сфери діяльності додати навчальний контент, тести, та провести процес тестування знань.

Для проведення аналізу якісних наявних систем були досліджені наступні сервіси для навчання та тестування знань:

– AcademyOcean;

– Docedo;

– Trainual;

– Kahoot.

AcademyOcean. Інтерактивний багатофункціональний сервіс для проведення навчального процесу і тестування через мережу Інтернет. Зручний конструктор для навчальних матеріалів і типів завдань, структурування їх по модулям. Є можливість створювати команду та додавати туди учнів, для подальшого запрошення до навчального модулю.

Модуль містить вбудований конструктор тестів з налаштуваннями типів питань, статистичних звітів і стилізації інтерактивних завдань. Формат тестових питань включає різні варіанти: один або кілька правильних відповідей, відповідь у вільній формі, встановлення послідовності і відповідностей тощо.

Пройти модуль можна запросивши співробітника по пошті, або опублікувати навчальний курс в мережу Інтернет для публічного доступу.

До недоліків можна віднести високу вартість підписки; складність у користуванні інтерфейсом; погана адаптація під різні пристрої; інтерфейс тільки англійською мовою; відсутність пробного періоду користування системою.

Docedo. Функціональний сервіс для створення навчального контенту в інтерактивній формі. Є можливість додавання назначати онлайн зустріч для учнів в календарі та опублікувати новини на головній сторінці навчального модуля.

Сервіс надає можливість створювати різного роду навчальний контент та тести, а також бачити активність учнів, та прогрес їх навчання.

У сервісі можна отримувати мінімальну аналітику щодо навчання учнів, оцінювати тестові завдання та виставляти оцінювальні рейтинги. Є можливість отримати доступ до сервісу на пробний період – 14 днів.

В Docedo складний для користування інтерфейс та наявність непрацюючих розділів в кабінеті адміністратора. Відсутність інформативної аналітики щодо проходження навчання. Можливість додавання учня для подальшого навчання тільки через консоль адміністратора. Відсутність

Trainual. Англомовний сервіс для швидкого конструювання навчальних сторінок та тестових завдань і опитувань з широким форматом відповідей. Менеджер з навчання може створювати і редагувати свої навчальні сторінки і тестові завдання, зберігаючи їх у віртуальній бібліотеці. Менеджер з навчання додає учнів, яких потім має можливість запросити на тестування по електронній адресі. Сервіс має реалізований чат, де учні під час проходження навчання можуть задавати питання.

Сервіс містить шаблони для швидкого створення навчального модулю та надає інформативну аналітику про проходження навчання співробітниками.

Також серед недоліків: тільки англійська мова інтерфейсу, висока ціна, немає можливості вивантажити результати тесту, невеликий вибір доступних типів тестових завдань.

Kahoot. Навчальна платформа, яка дозволяє проводити зрізи знань, тестування, опитування, незалежно від направлення дисципліни, а також виклад нового матеріалу в ігровій формі.

Сайт містить багато готових тестів на різноманітні тематики. Викладач може створювати власні завдання за допомогою готових шаблонів в форматі тесту. До питань можна додавати фото, малюнки, графічні зображення і відео. Платформа дозволяє організовувати опитування або контрольний зріз знань у вигляді інтерактивного змагання. Для цього сайт має режим бонусів для студентів за швидкість надання відповідей.

Платний тариф надає розширене використання функціональних можливостей сервісу.

До недоліків слід віднести обмежені типи питань; обмежені можливості опитування; дуже обмежені параметри налаштування; немає підтримки для запитань і відповідей; немає місця де можна поставити запитання в процесі тестування; заплутана панель інструментів і інтерфейс; переважна більшість функціоналу відсутня у безкоштовному пакеті, а складний процес ціноутворення робить сервіс не зовсім привабливим.

2.2 Використання REST архітектури при розробці веб-застосунків

Передача репрезентативного стану (REST) – це стиль архітектури програмного забезпечення, який використовує підмножину передачі даних HTTP. Він зазвичай використовується для створення інтерактивних застосунків, що використовують веб-застосунки. Веб-застосунок, який дотримується цих рекомендацій, називається RESTful. Такий веб-застосунок повинен надавати свої веб-ресурси в текстовому поданні і дозволяти їх читати і змінювати за допомогою протоколу без збереження стану і визначеного набору операцій. Такий підхід забезпечує взаємодію між комп'ютерними системами в мережі Інтернет. REST - це альтернатива, наприклад, SOAP як спосіб доступу до веб-застосунку. REST архітектура, схематично зображена на рисунку 2.1.

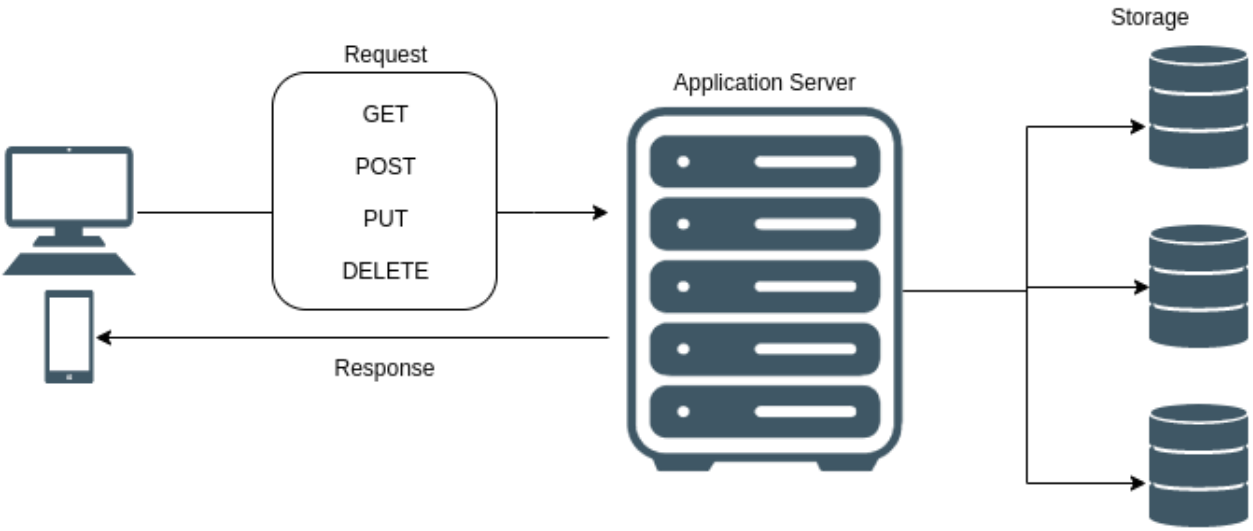

Рисунок 2.1 – Схематичне представлення REST архітектури

Рой Філдінг визначив REST в своїй докторській дисертації 2000 «Архітектурні стилі і проектування мережевих архітектур програмного забезпечення» в Каліфорнійському університеті в Ірвіні. Він розробив архітектурний стиль REST паралельно з HTTP 1.1 в 1996-1999 роках, грунтуючись на існуючому дизайні HTTP 1.0 1996 року.

Рой Філдінг описує вплив REST на масштабованість наступним чином: “Поділ завдань клієнт-сервер в REST спрощує реалізацію компонентів, знижує складність семантики коннектора, підвищує ефективність настройки продуктивності і збільшує масштабованість чистих серверних компонентів. Обмеження багаторівневої системи дозволяють проксі-посередникам, шлюзам і між-мережевим екранам бути представленим на різних етапах комунікації без зміни інтерфейсів між компонентами, що дозволяє їм допомагати в трансляції комунікацій або підвищувати продуктивність за рахунок великомасштабного загального кешування. REST дозволяє проміжну обробку, обмежуючи повідомлення, щоб вони були інформативними: взаємодія між запитами не має стану, стандартні методи і типи мультимедіа використовуються для позначення семантики і обміну інформацією”.

«Веб-ресурси» вперше були визначені в мережі Інтернет як документи, файли які ідентифіковані по їх URL-адресами. Сьогодні це визначення є набагато більш загальним і абстрактним і включає в себе кожну річ, сутність

або дію, які можуть бути ідентифіковані, названі, адресовані, оброблені або виконані будь-яким чином в мережі. У веб-службі RESTful запити до URI ресурсу викликають відповідь з даними, відформатовані в HTML, XML, JSON або іншому форматі. Наприклад, відповідь може підтвердити, що стан ресурсу було змінено. Відповідь також може містити гіпертекст, посилання на пов'язані ресурси тощо. Найбільш поширеним протоколом для цих запитів і відповідей є HTTP. Використовуючи протокол без збереження стану і стандартні операції, системи RESTful прагнуть до високої продуктивності, надійності і можливості зростання за рахунок багаторазового використання компонентів, якими можна керувати і оновлювати, не зачіпаючи систему в цілому, навіть під час її роботи.

Метою REST є підвищення продуктивності, масштабованості, простоти, модифікованості, видимості, переносимості та надійності. Це досягається за рахунок дотримання таких принципів REST, як архітектура клієнт-сервер, відсутність стану, кешування даних, використання багаторівневої системи, підтримка коду за запитом і використання уніфікованого інтерфейсу. Такі принципи повинні дотримуватися, щоб система була класифікована як REST.

Обмеження архітектурного стилю REST впливають на наступні архітектурні властивості:

- продуктивність при взаємодії компонентів, яка може бути домінуючим фактором сприймання користувачем продуктивності і ефективності мережі;
- масштабованість, що дозволяє підтримувати велику кількість компонентів і взаємодії між компонентами;
- простота єдиного інтерфейсу;
- можливість модифікації компонентів для задоволення мінливих потреб (навіть під час роботи програми);
- видимість зв'язку між компонентами сервісними агентами;
- переносимість компонентів за рахунок переміщення програмного коду разом з даними;

– надійність в стійкості до збоїв на системному рівні при наявності збоїв в компонентах або даних.

Переваги архітектурного стилю REST:

– Відсутність додаткових внутрішніх прошарків, що означає передачу даних в тому ж вигляді, що і самі дані. Тобто дані не обертаються в XML, як це робить SOAP і XML-RPC, не використовується AMF, як це робить, наприклад Flash.
– Кожна одиниця інформації (ресурс) однозначно визначається URL посиланням – це значить, що URL по суті є первинним ключем для одиниці даних. Причому абсолютно не має значення, в якому форматі знаходяться дані за адресою - це може бути і HTML, картинка, документ тощо.
– Як відбувається управління інформацією ресурсу – це цілком і повністю ґрунтується на протоколі передачі даних. Використання найбільш поширеного протоколу передачі даних – HTTP. Для HTTP запитів, дії над даними виконуються за допомогою методів: GET (отримати дані), PUT (додати або замінити дані), POST (додати, змінити або видалити дані), DELETE (видалити дані). Таким чином, CRUD операції (Create-Read-Update-Delete) можуть виконуватися як з усіма 4-ма методами, так і тільки за допомогою GET і POST.

Обмеження архітектури REST, які формують ряд недоліків такого підходу:

– Надмірна вибірка даних. Найчастіше кінцева точка API надає на запит зайву інформацію, яка не потрібна клієнтові для здійснення дій. Наприклад, ви хочете отримати тільки ім'я користувача, але метод отримує вам ще його вік, телефон та електронну пошту, що робить процес отримання даних більш трудомістким та таким, що не приносить користі.
– Недостатня вибірка. Можлива і зворотна ситуація, коли клієнту доводиться робити декілька запитів, щоб отримати всю інформацію,

яка йому потрібна. Наприклад, з однієї кінцевої точки API ви отримуєте особисті дані користувача, а з іншого - інформацію про зроблені ним замовленнях.

Висновки до розділу

В цьому розділі було проведено аналіз наявних аналогових програмних систем навчання та тестування знань та визначено їх ключові переваги та недоліки. Було проведено аналіз підходу до проектування веб-застосунків на основі REST архітектури, визначено недоліки та переваги.

Веб-застосунки для навчання та тестування знань все активніше захоплюють різноманітні направлення діяльності суспільства, де необхідна організація навчання та перевірки знань, представляючи собою більш легку альтернативу традиційним підходам до процесу навчання та тестування знань.

3 ПРОЕКТУВАННЯ, РОЗРОБЛЕННЯ, ВПРОВАДЖЕННЯ ПРОГРАМНОЇ СИСТЕМИ НАВЧАННЯ ТА ТЕСТУВАННЯ ЗНАНЬ

3.1 Вибір середовища розробки та інженерія вимог до програмної системи навчання та тестування знань

В якості інструментарію розробки була використана IntelliJ IDEA та WebStorm - продукти компанії JetBrains, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Інтегроване середовище розробки (IDE) являє собою комплекс програмних засобів, які можна використовувати для різних аспектів розробки програмного забезпечення. Компанія JetBrains надає перелік програмних рішень, які можна легко інтегрувати та використовувати при розробці програмного забезпечення, що дозволяє писати код в зручному інтерфейсі редактора і включає в себе компілятори, засоби автоматичного завершення виконання коду, автоматичний запуск тестів, графічні конструктори і багато інших функціональних можливостей для спрощення процесу розробки.

Для написання серверної частини програмної системи, було використано IntelliJ IDEA, оскільки вона має велику кількість вбудованих допоміжних інструментів та інтеграцій з іншими системами, а також включає в себе глибокий аналіз коду, який шукає зв’язок між всіма символами у всіх файлах, на всіх мовах програмування, які використовуються в проекті, а на підставі цього надає підказки при розробці, що допомагає сконцентруватися при написанні коду. З допомгою технології GoLand, IntelliJ IDEA надає змогу з редактора працювати з базами даних SQL, швидко налаштовувати підключення, виконувати запити, переглядати або експортувати дані та керувати графічними схемами бази даних.

IntelliJ IDEA дозволяє створювати і підключати сторонні застосунки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як,

наприклад, Git, SVN, Mercurial, CVS, Perforce та TFS), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення.

Для командних проектів IntelliJ IDEA пропонує підтримку групової роботи, дозволяючи виконувати спільне редагування і налагодження будь-якої частини коду в реальному часі, з можливістю.

Для написання клієнтської частини програмної системи, було використано WebStorm, що є розширенням інтегрованого середовища IntelliJ IDEA, яке дозволяє компілювати JavaScript, HTML, CSS тощо. WebStorm весь описаний функціонал IntelliJ IDEA, та дозволяє зручно створювати, відлагоджувати та компілювати програмний код на JavaScript.

WebStorm дає змогу використовувати функціонал JavaScript з всіма сучасними фреймворками(такими як, Angular, React, Vue.js і Meteor), а також надає можливість реалізації програмних систем для мобільних телефонів(з використанням React Native, PhoneGap, Cordova або Ionic).

Також JetBrains надає комплекс інструментів для автоматизації тестування застосунків в частині перевірки роботи інтерфейсів, модульного і навантажувального тестування.

Створення і впровадження веб-застосунку для навчання та тестування знань, впровадження повного функціоналу, внесення корегувань у функціонал веб-застосунку та оптимальних існуючих рішень, його вдосконалення, а також додавання нових функціональних можливостей для покращення роботи програмної системи − є основним завданням дипломної роботи.

Перевагою веб-застосунку на відміну від схожих існуючих рішень є легкий та інтуїтивно зрозумілий інтерфейс, завдяки чому будь-який інтернет-користувач зможе використовувати даний веб-сервіс для швидкого створення навчальних матеріалів та проходження тестів, легкого та надійного зберігання даних та інтерактивної аналітики проходження навчальних модулів, що дозволить ефективно проводити зріз знань.

Для виконання визначеного завдання необхідно вирішити наступні задачі:

– провести аналіз наявних веб-застосунків, які реалізують функціонал навчання та тестування знань;

– виявити переваги в наявних веб-застосунках та відтворити їх в майбутньому функціоналі власної програмної системи;

– знайти недоліки в наявних веб-сервісах та вирішити їх у власному;

– створення дизайну веб-застосунку зі зручним та зрозумілим інтерфейсом користувача;

– адаптувати відображення інтерфейсу для мобільних телефонів, планшетів та комп'ютерів;

– реалізувати крос-браузерну адаптацію для найбільш використовуваних браузерів на всіх пристроях;

– додати визначений функціонал в застосунок;

– перевірити коректність вхідної інформації та коригування вхідних даних при помилковому введенні;

– забезпечити максимальну безпеку інформації в БД;

– виконати тестування функціоналу та адаптивності веб-застосунку.

Необхідно визначити коло задач, для розв’язання яких розроблюється програмний продукт. Для того, щоб цього досягти, з’ясуємо відомості, необхідні для розробки ефективної структури даних.

Розглянемо етапи даного процесу:

– визначення цілей створення програмної системи;

– визначення обсягів і типів даних;

– визначення способів використання даних.

На першому етапі, а саме визначення цілей системи, слід мати на увазі, що робота веб-застосунку базується на процесах обміну даними. При реалізації проекту будемо використовувати БД, яка повинна бути легко масштабованою, для зберігання та накопичення навчальних матеріалів,

тестових завдань, результатів проходження тестів, інтерактивної аналітики, а також дані безпосередньо про користувача веб-сервісу.

Таким чином, цілями створення даного продукту є:

– забезпечення можливості ефективного і надійного зберігання та управління даними;

– створення можливості для оновлення та розширення інформації в БД;

– забезпечення можливості додавання навчального контенту;

– створення системи проходження тестів.

Другим етапом створення ефективної структури є визначення типів даних та зв’язків сутностей, від яких безпосередньо залежить продуктивність БД.

Внаслідок постійного розширення обсягів навчальної інформації, розмір завантажених даних зростатиме високими темпами, що призводитиме до значного збільшення об’єму сховища та навантаження на нього. Саме тому слід передбачити таку властивість, як розширюваність БД.

Третім етапом створення ефективної структури проекту є визначення рівнів доступу до інформації. Адміністратор має можливість додавати навчальний контент, створювати тести та запрошувати на них співробітників, переглядати аналітику проходження навчального модуля співробітниками, додавати інтеграції для швидкого сповіщення результатів тестування у популярні месенджери. Співробітник має можливість переглядати інформацію про завершені навчальні модулі, повторно переглядати навчальний матеріал, проходити нові доступні навчальні модулі.

Для досягнення високої ефективності роботи системи, а також зведення до мінімуму можливості виникнення нештатних ситуацій, необхідним є налагодження системи безпеки.

Для будь-якого проектованого програмного продукту необхідно визначити вимоги до його візуального оформлення.

Інтерфейс користувача – це сукупність доступних інструментальних засобів, за допомогою яких користувач може взаємодіяти з системою, програмним забезпеченням тощо.

Система повинна мати зручний та зрозумілий для користувача інтерфейс. При розробці інтерфейсу необхідно пам’ятати, що інтерфейс – одна з важливіших частин будь-якої системи з точки зору безпосереднього користувача системи. Тому важливо з’ясувати, який саме тип інтерфейсу можна розробити за допомогою використовуваних інструментальних засобів.

Основні властивості, які повинен мати інтерфейс застосунку:

– адаптованість – інтерфейс повинен бути: сумісним з потребами користувача; простим в переході від виконання однієї функції до іншої; забезпечувати користувача вказівками стосовно його можливих дій; забезпечувати користувача формами представлення результатів в залежності від типу запиту або від характеру отриманого рішення;

– зрозумілість – це простота використання інтерфейсу, при цьому забезпечуючи функціональні запити користувача при розв’язанні певного класу задач;

– гнучкість – це можливість адаптування інтерфейсу до розв’язання різного роду задач.

Таким чином, користувацький інтерфейс має забезпечити можливості:

– авторизації існуючого користувачів з різними ролями;
– додавання, редагування, перегляд та видалення записів в базі даних;
– виконання запитів до сховища і вивід результатів на екран браузеру в зручному для користувача вигляді;
– перегляд проміжних результатів роботи.

В результаті розробки програмного продукту інтерфейс користувача можна відокремити на дві частини – інтерфейс користувача та адміністратора. Інтерфейс учня має містити наступні сторінки:

– сторінка авторизації;
– головна сторінка;

– сторінка налаштування профілю користувача;
– сторінка доступних навчальних курсів;
– сторінка проходження навчального курсу;
– сторінка проходження тестування.

Інтерфейс адміністратора, повинен містити наступні сторінки:

– сторінка авторизації;
– головна сторінка;
– сторінка створення навчального курсу;
– сторінка перегляду звітів навчання;
– сторінка перегляду успішності учнів;
– сторінка підключення інтеграцій з іншими системами;
– сторінка налаштування користувацького інтерфейсу навчальних курсів.

Для вирішення завдань проектованої системи були проаналізовані можливості веб-застосунку для навчання та тестування знань та виокремлено функціонал, який найкраще підходить для розроблювального веб-застосунку, при цьому максимально полегшуючи роботу користувачеві та не зменшуючи продуктивність веб-застосунку в цілому (таблиця 3.1).

Таблиця 3.1 – Основні вимоги до функціональних можливостей системи

Назва функціоналу	Опис функціоналу
Авторизація	Для ідентифікація користувача в системі, надання рівню доступу до перегляду сторінок та використання відповідного функціоналу веб-застосунку необхідно пройти авторизацію. Дані авторизації зберігаються для поточної та наступних сесій за допомогою кешу браузера.

Продовження таблиці 3.1

Створення навчального курсу	Дана функція дозволяє переглядати наявні навчальні курси, та надає можливість створення нових.
Інтеграція з іншими системами	Функція дозволяє налаштувати інтеграцію з популярними месенджарами, для отримання сповіщень про прогрес навчання.
Перегляд аналітики	Функція призначена для інтерактивного перегляду аналітики.
Перегляд звітів	Функція дозволяє переглядати звіт про проходження учнями тестових завдань.
Перегляд учнів	Функція дозволяє отримати інформацію по учням та переглянути їх прогрес навчання.
Кастомізація навчання	Функція дозволяє налаштувати кастомізацію для навчального контенту.
Конструктор навчального курсу	Функція надає можливість створення та розміщення в довільному порядку навчального контенту, створення та налаштування тестів.
Запрошення на навчальний курс	Функція дозволяє запрошувати учнів до проходження навчального курсу по пошті.
Проходження навчального курсу	Функція дозволяє учню проходити навчання та виконувати тести.
Вихід з системи	Функція дозволяє вийти з системи.

В таблиці наведено перелік основних вимог до фунціоналу, який буде реалізовано в рамках дипломної роботи. Ці функції визначають фундаментальний набір необхідних можливостей програмної системи для навчання та тестування знань.

3.2 Проектування та розроблення програмної системи навчання та тестування знань

Для того щоб краще зрозуміти потоки даних та правильно спроектувати функціонал передачі та обробки інформації у веб-застосунку використовується діаграма потоків даних, що дає змогу на етапі аналізу та проектування програмної системи отримати наочне уявлення про обмін даними та їх потоки в програмній системі (рис. 3.1).

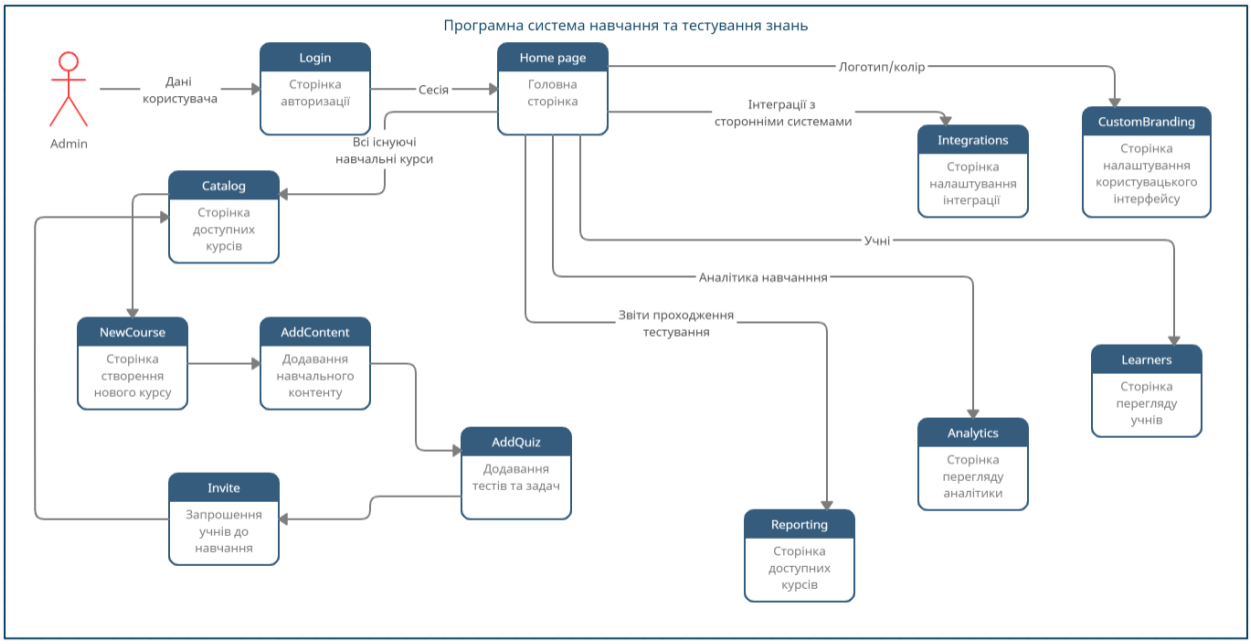

Рисунок 3.1 – Діаграма потоків даних програмної системи навчання та тестування знань

Оскільки веб-застосунок постійно обробляє, та постійно накопичує велику кількість інформації, постає необхідність проектування та реалізації масштабованої бази даних. Заздалегідь можна визначити мінімальні вимоги до бази даних – наявність таблиць для зберігання навчального контенту, тестів, аналітики та відомостей про користувачів. Схема бази даних графічно відображена на рисунку 3.2.

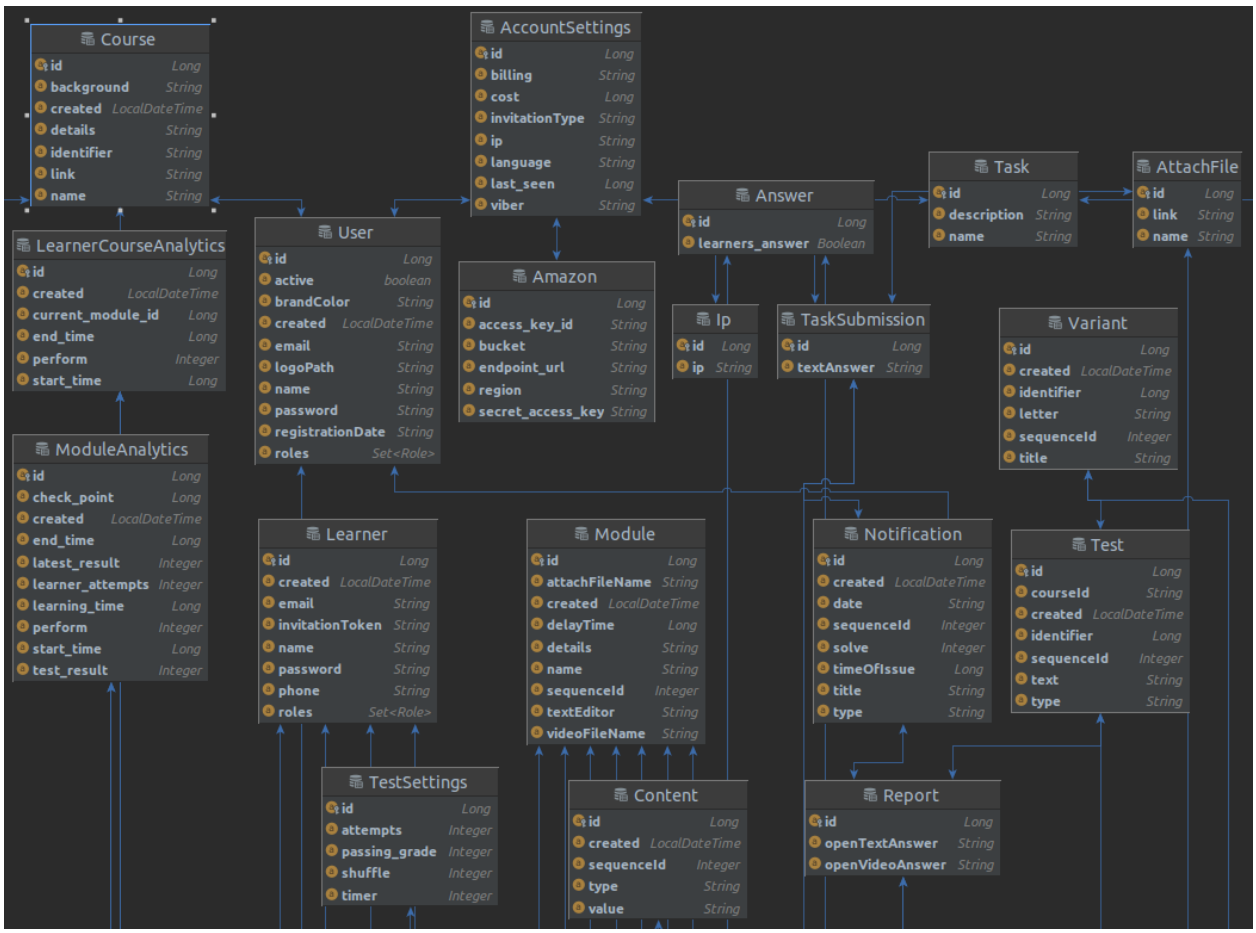

Рисунок 3.2 – Графічна схема бази даних програмної системи навчання та тестування знань

Основними системними елементами в програмній системі є Java контролери, представлені файлами класів серверної частини проектованої системи, та React компоненти, для відображення клієнтської частини.

Робота з даними та базою даних застосунку керується головним чином Java контролерами, але безпосередньо користувачеві застосунок доступний у вигляді представлення, яке і формує зовнішній вигляд веб-застосунку. Для цього використовуються компоненти React, які дозволяють розділяти користувацький інтерфейс на незалежні частини для можливого повторного використання в проекті. Сторінки React зазвичай підкріплюються відповідним файлом типу .jsx, який представляє модель для візуального відображення сторінки та попередньої обробки інформації, перед її візуалізацією. Це дозволяє впроваджувати функціональні можливості мови програмування JavaScript всередині HTML розмітки для реалізації функціональних

можливостей програмної системи та інтерактивним інтерфейсом. Перелік головних компонентів клієнтської частини подано в таблиці 3.2.

Таблиця 3.2 – Головні компоненти клієнтської частини веб-застосунку

Компонент	Опис
Login	Компонент, який реалізує авторизацію користувача в системі за допомогою форми введення даних логіну та пароля.
Account	Компонент, що реалізує головне вікно меню системи.
Reporting	Компонент для відображення результатів тестування знань, та можливістю перегляду пройдених тестів.
Analytics	Компонент для представлення сторінки з інтерактивним відображенням аналітики прогресу навчального процесу та результатів проведення зрізу знань.
Learners	Компонент для представлення сторінки з переліком учнів та можливістю їх видалення.
Integrations	Компонент для реалізації підключення системи оповіщення з популярними системами обміну повідомленнями.
CustomBranding	Компонент для реалізації налаштування користувацького логотипу системи та кольору.

Продовження таблиці 3.2

CreateCourse	Компонент для реалізації створення навчального курсу, створення структури, додавання навчального контенту до модулів, створення тестів. Також сторінка реалізує функціонал запрошення учнів до навчального курсу.
LearnCourse	Компонент для проходження навчання з можливістю складання тестування та задач.

Розроблена програмна система реалізована на основі архітектури веб-застосунку, тому вона визначає мінімальні вимоги до програмного забезпечення користувачів для коректного користування системою. Достатньо буде мати сучасний мобільний пристрій, планшет або комп’ютер з налаштованим підключення до мережі Інтернет та встановлений на ньому сучасний браузер, за допомогою якого буде надано доступ до веб-застосунку.

3.3 Програмне забезпечення веб-застосунку для навчання та тестування знань

Для реалізації проектної системи, було використано технології створення веб-сайтів: Java, Spring Boot, JavaScript, HTML, CSS, SCSS, React, Redux та систему управління реляційними базами даних (СУБД) PostgreSQL. Дані технології дозволяють виконати поставленні вимоги технічного завдання. Завдяки основним відомостям про використані технології буде визначено з якою метою вони були залучені.

Java – строго-типізована об’єктно-орієнтована мова програмування загального призначення, що виконується віртуальною машиною Java, яка в свою чергу оброблює байтовий код і передає інструкції обладнанню як інтерпретатор.

Нині Java є однією з найбільш поширених мов програмування в світі. Перша версія з'явилася ще в 1996 році, створена компанією Sun Microsystems, яка згодом перейшла до Oracle. Java замислювався як універсальна мова програмування, яку можна застосовувати для різного роду завдань. До теперішнього часу мова програмування Java пройшла великий шлях удосконалення функціональних можливостей, було видано безліч різних версій. Поточною версією є Java 16, яка вийшла в березні 2021 року, а Java перетворилася в велику екосистему і платформу, яка об'єднує різні технології, які використовуються для цілого ряду завдань: від створення десктопних застосунків до написання великих веб-порталів і сервісів.

Spring Boot – універсальний фреймворк з відкритим вихідним кодом для Java-платформи. Spring Boot забезпечує вирішення багатьох задач, з якими стикаються Java-розробники, які хочуть створити інформаційну систему, на базі машини Java. Фреймворк більш відомий як постачальник розширень, необхідних для ефективної розробки складних програмних моделей, та пропонує почергову модель розробки і робить її доступною до більшості типів застосунків, які вже створені на платформі Java.

Через громіздкої конфігурації залежностей, налаштування Spring Boot для великих і масштабованих програмних систем перетворилося в досить тяжке завдання, яке нерідко призводить до помилок. Особливо це відноситься до застосунків, які використовують також кілька сторонніх бібліотек.

Кожен раз, створюючи черговий корпоративний Java-застосунок на основі фреймворку Spring, вам необхідно повторювати одні й ті ж рутинні кроки по його налаштуванню:

- Залежно від типу створюваного застосунку (Spring MVC, Spring JDBC, Spring ORM тощо) імпортувати необхідні Spring-модулі.
- Імпортувати бібліотеку веб-контейнерів (в разі веб-застосунків).
- Імпортувати необхідні сторонні бібліотеки (наприклад, Hibernate, Jackson тощо), при цьому ви повинні шукати версії, сумісні з вказаною версією Spring.

– Конфігурувати компоненти DAO, такі, як: джерела даних, управління транзакціями тощо.
– Конфігурувати компоненти веб-шару, такі, як: диспетчер ресурсів, view resolver.
– Визначити клас, який завантажить всі необхідні конфігурації.

JavaScript – мульти-парадигмова мова програмування, яка підтримує об’єктно-орієнтований, імперативний та функціональний стилі програмування. При розробці веб-сервісу для навчання та тестування співробітників було використано різноманітні бібліотеки та сучасні підходи для роботи з мовою програмування JavaScript.

Сьогоднішній світ веб-застосунків важко уявити без мови програмування JavaScript. JavaScript – це те, що робить інтерактивними веб-сторінки, які ми кожен день переглядаємо в своєму веб-браузері.

JavaScript був створений в 1995 році в компанії Netscape в якості мови сценаріїв в браузері Netscape Navigator 2. З самого початку мова називалася LiveScript, але на хвилі популярності в той момент іншої мови Java, LiveScript був перейменований в JavaScript, абсолютно дві різні мови, які пов'язані тільки назвою.

З самого початку існувало небагато веб-браузерів (Netscape, Internet Explorer тощо), які надавали різні реалізації мови програмування JavaScript. І щоб звести різні реалізації до загального і стандартизувати мову під керівництвом організації ECMA був розроблений стандарт ECMAScript. В принципі самі терміни JavaScript і ECMAScript є багато в чому взаємозамінними і відносяться до однієї мови програмування.

До теперішнього часу ECMA було розроблено декілька стандартів мови програмування, які відображають її розвиток. Останнім прийнятим нині стандартом є ECMAScript 2015 (ES6). Але треба сказати, що реалізація цього стандарту в поширених веб-браузерах дуже далека до завершення, і, можливо, на його повне впровадження піде кілька років.

React – це сучасний підхід до реалізації веб-застосунків, що використовує JavaScript XML та підтримує використання різних бібліотек. Особливістю роботи React є можливість створення та оперування функціональними компонентами, компонентами на основі класів, використання вбудованих методів життєвого циклу та робота з віртуальним DOM.

React був створений компанією Facebook, а перший реліз бібліотеки побачив світ у березні 2013 року. Поточною версій нині є версія React v17.0.

Спочатку React призначався для веб-ресурсів, для створення веб-застосунків, проте пізніше з'явилася платформа React Native, яка вже призначалася для мобільних пристроїв.

React є інструментом для створення масштабованих веб-застосунків (в цьому випадку йдеться про клієнтську частину), особливо в тих ситуаціях, коли це застосунок SPA (односторінковий застосунок). React відносно простий у вивченні, має зрозумілий та лаконічний синтаксис, що робить процес розробки програмної системи більш швидким та структурованим.

React вирішує тільки задачі рендерингу сторінок в DOM, тому потребує використання додаткових бібліотек для управління станом сторінки і маршрутизації в системі. Такою бібліотекою в проекті слугує Redux.

Redux – це JavaScript-бібліотека з відкритим початковим кодом, що допомагає керувати станом застосунку. Вона допомагає створювати веб-застосунки, які мають передбачувану поведінку, які працюють на оточеннях клієнту та серверу, та які легко тестувати.

Ця бібліотека ідеально співпрацює з React, тому саме вона була обрана для вирішення поставленої задачі.

Для зберігання та маніпулювання даними використовувалась СУБД PostgreSQL, що нині є однією з сучасних популярних систем управління базами даних.

Реляційна база даних – це набір даних з визначеними зв'язками між ними. Дані організовані у вигляді набору таблиць, що складаються із стовпців

і рядків, а таблиці в свою чергу зберігають інформацію про об'єкти, які представлені в базі даних. У кожному стовпці таблиці зберігається певний тип даних, в кожному полі таблиці – значення атрибута. Кожне поле таблиці являє собою набір взаємопов'язаних значень, що відносяться до однієї сутності. Кожен рядок в таблиці може бути позначеним унікальним ідентифікатором, званим первинним ключем(або “primary key”), а рядки з декількох таблиць можуть бути пов'язані з допомогою зовнішніх ключів. Отримати доступ до даних можна різними способами, при цьому не змінюючи структуру організації збереження таблиць.

Для взаємодії з базою даних використовується декларована мова програмування SQL (Structured Query Language), для керування даними в реляційній БД. Клієнт (наприклад, стороння програмна система) надсилає запит мовою SQL за допомогою спеціального API. Система управління базою даних відповідним чином інтерпретує і переходить до виконання запиту, а потім надсилає клієнту результат виконання.

PostgreSQL – це об'єктно-реляційна система управління базами даних з відкритим вихідним кодом. PostgreSQL заснована на мові програмування SQL і підтримує велику кількість можливостей проведення маніпуляцій з даними. PostgreSQL ідеально може бути використана для різних за розмірами проектів: від маленьких веб-застосунків до великих високонавантажених програмних систем.

PostgreSQL була розроблена на основі некомерційної СУБД Postgres, розробленої як проект з відкритим вихідним кодом в Каліфорнійському університеті.

Переваги СУБД PostgreSQL:

– підтримка роботи бази даних необмеженого розміру;
– ефективні механізми реплікацій і транзакцій;
– спадкування сутностей;
– легка розширюваність та масштабованість.

Недоліки СУБД PostgreSQL:

– відсутність обмеження максимального розміру бази даних;
– відсутність обмеження кількості записів в таблиці;
– відсутність обмеження кількості в таблиці;
– обмеження максимального розміру таблиць;
– обмеження максимального розміру записів;
– обмеження максимального розміру полів;
– обмеження максимальної кількості полів – 250-1600 (в залежності від типів полів).

Більшість популярних фреймворків для розробки підтримують коректну та легку інтеграцію з PostgreSQL при розробці програмних систем.

3.4 Інструкція користувача

Після переходу на сайт, користувач потрапляє на сторінку авторизації, на якій йому доступна авторизації за логіном та паролем (рис. 3.3). При введенні некоректних даних облікового запису або іншої помилки на формі виводиться відповідне повідомлення.

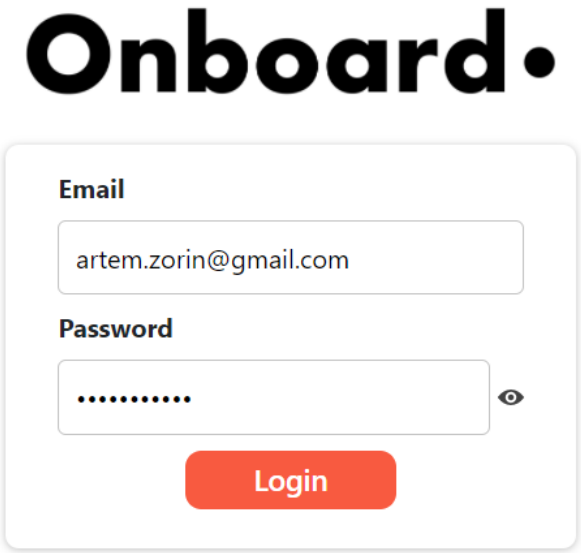

Рисунок 3.3 – Сторінка авторизації користувача

Після того як користувач успішно авторизувався, буде відображена головна сторінка адміністратора, на якій відображається головне меню, та список існуючих навчальних курсів (рис. 3.4).

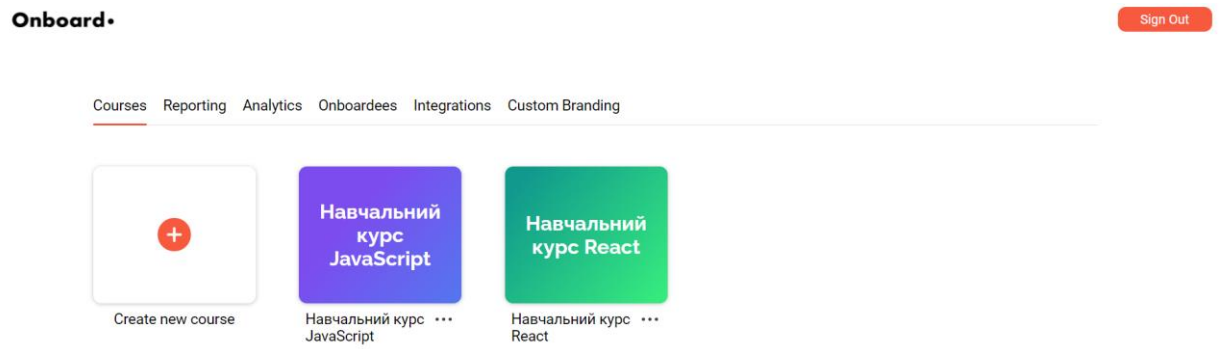

Рисунок 3.4 – Головна сторінка

На головній сторінці, є можливість створення нового навчального курсу. При створенні нового навчального курсу, відкривається вікно де треба вказати його назву (рис. 3.5). Після створення нового курсу, буде відкрито сторінку для створення структури курсу(додавання навчальних модулів), та додавання мети цього курсу(рис. 3.6).

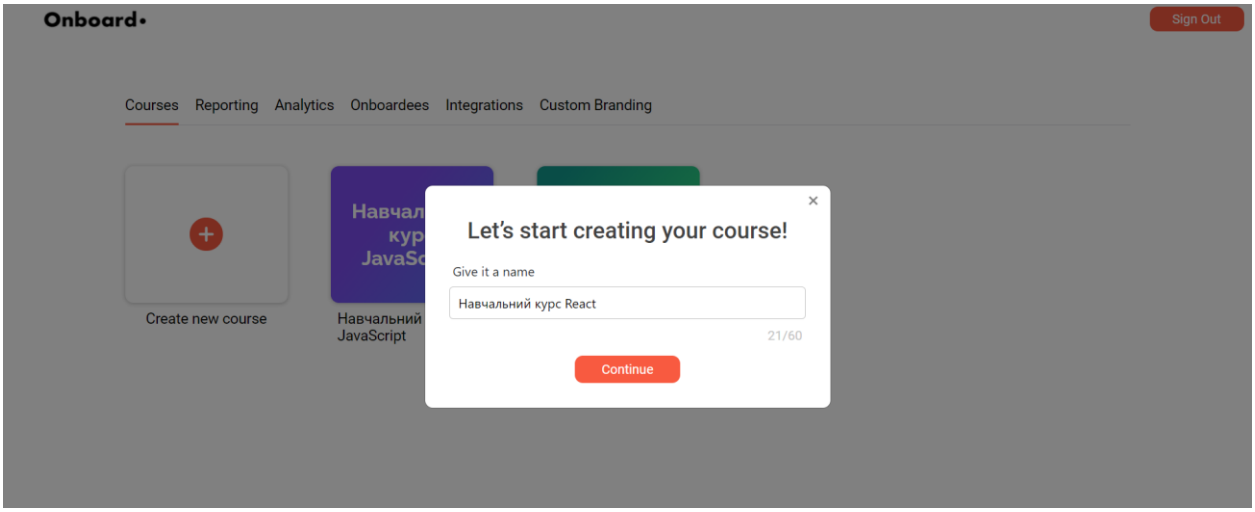

Рисунок 3.4 – Вікно створення назви нового курсу

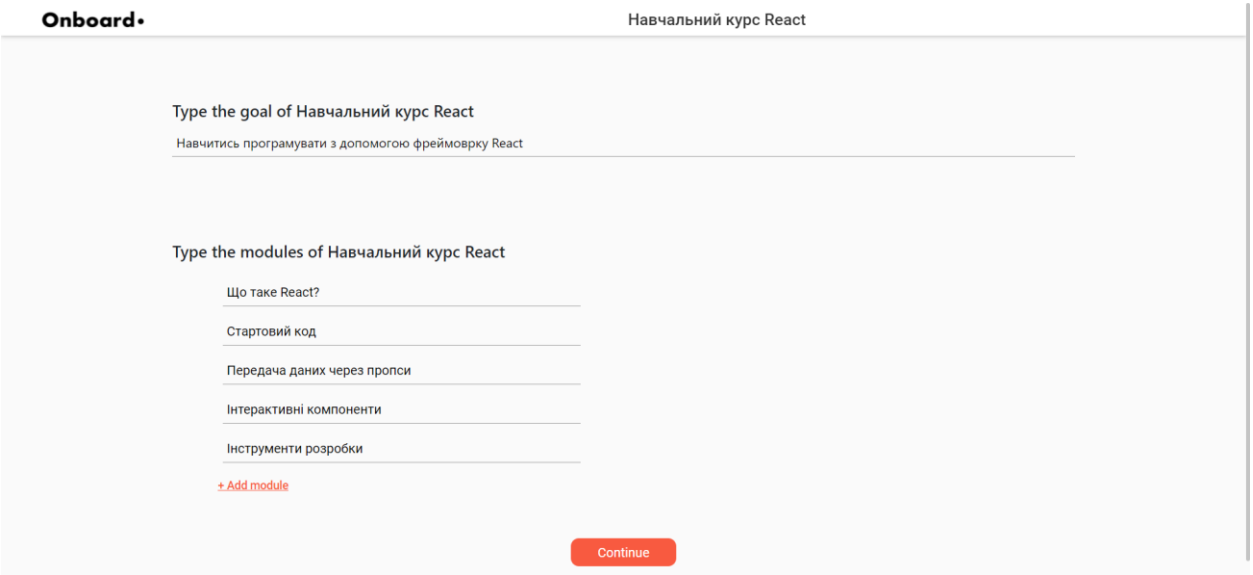

Рисунок 3.5 – Вікно проектування структури нового курсу

Після створення нового навчального курсу, буде відображена головна сторінка конструктора навчального курсу(рис. 3.6), де можна додавати навчальний контент за допомогою панелі інструментів.

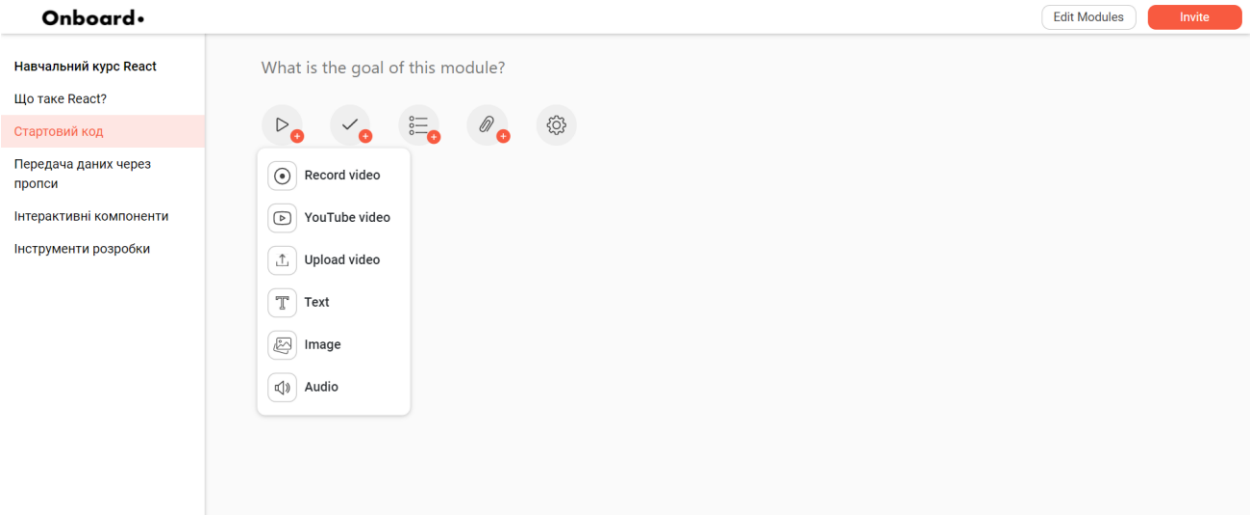

Рисунок 3.6 – Головне вікно конструктора навчального курсу

Також на сторінці конструктора навчального курсу, є можливість додавання тестового завдання(рис. 3.7) та задачі(рис. 3.8) у вікні. При створенні тестового завдання, є можливість обрати тип питання(з декількома варіантами вибору або відкрите питання) та вказати правильну відповідь.

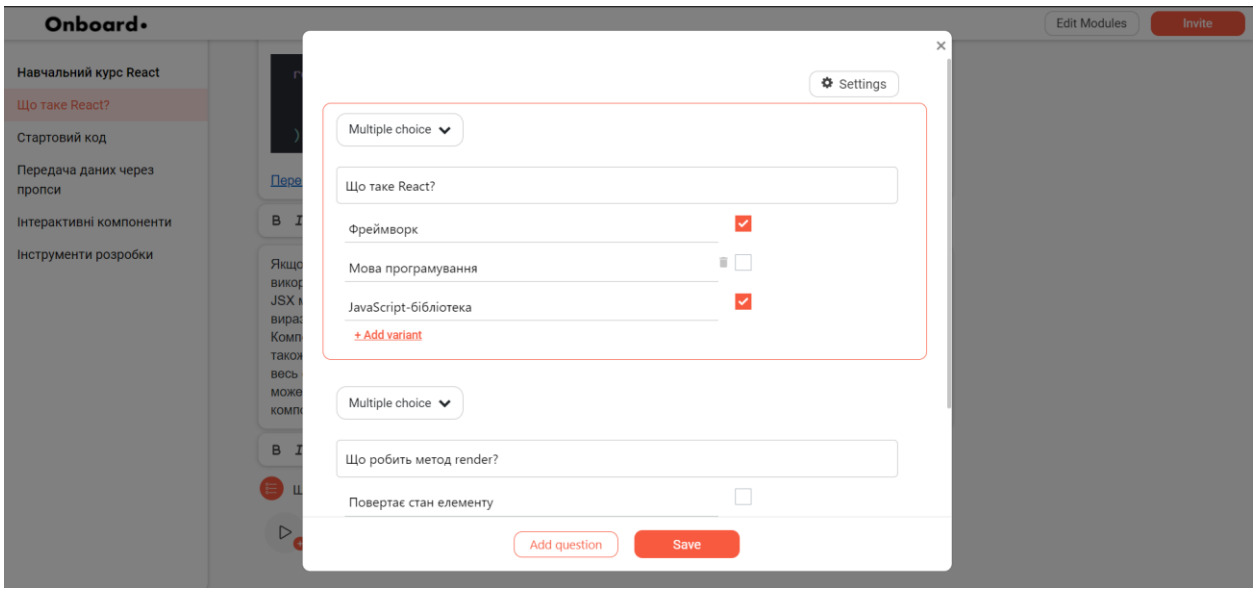

Рисунок 3.7 – Головне вікно створення тестового завдання

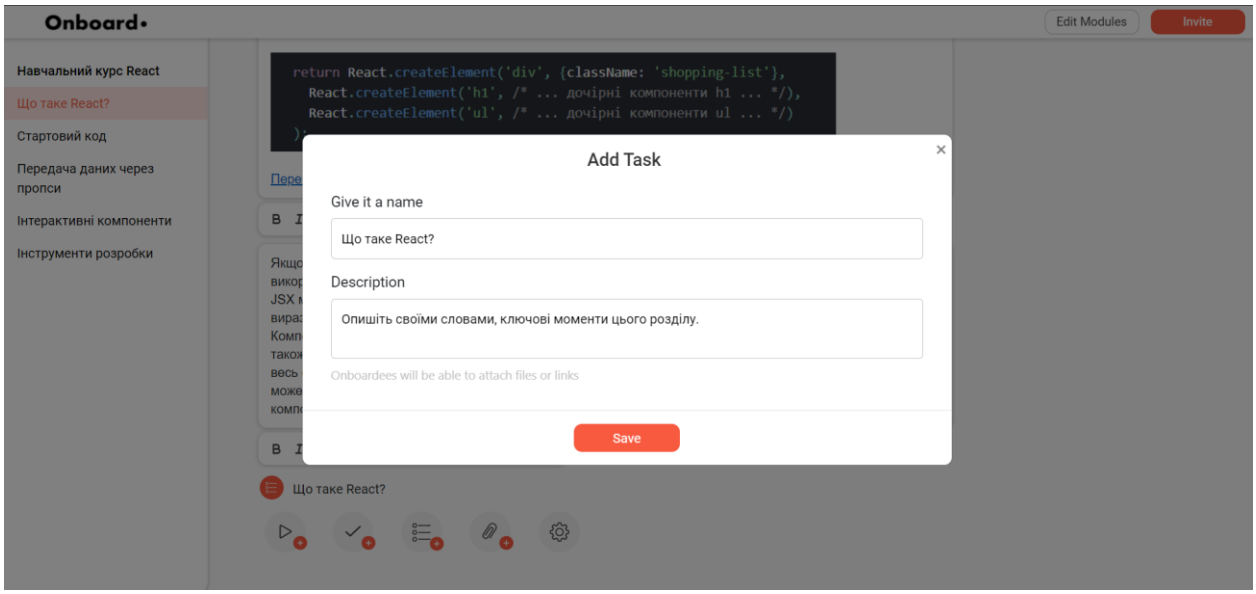

Рисунок 3.8 – Головне вікно створення задачі

Після завершення додавання навчального контенту та наповнювання курсу матеріалами для навчання, є можливість запросити до проходження курсу учнів, або зробити посилання на курс публічним(рис. 3.9).

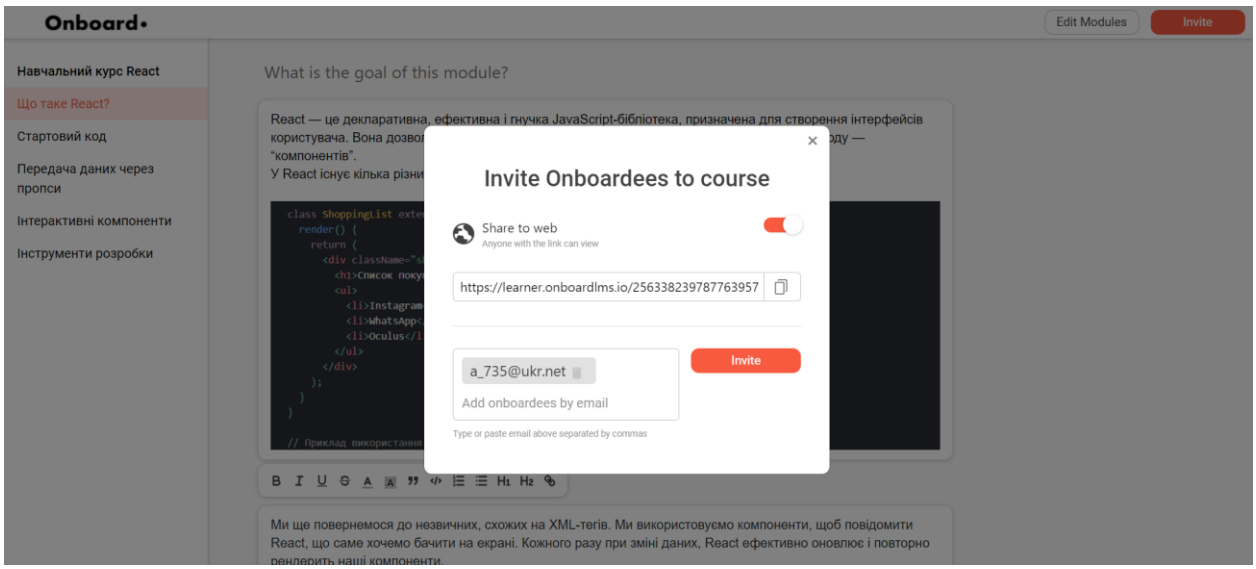

Рисунок 3.9 – Вікно запрошення учнів на курс

Користувач переходить за посиланням яке було створене, та потрапляє на сторінку реєстрації(рис. 3.10), де можна створити акаунт в системі, якщо такий відсутній. Якщо користувач має акаунт, то потрібно увійти в систему на сторінці авторизації(рис. 3.11).

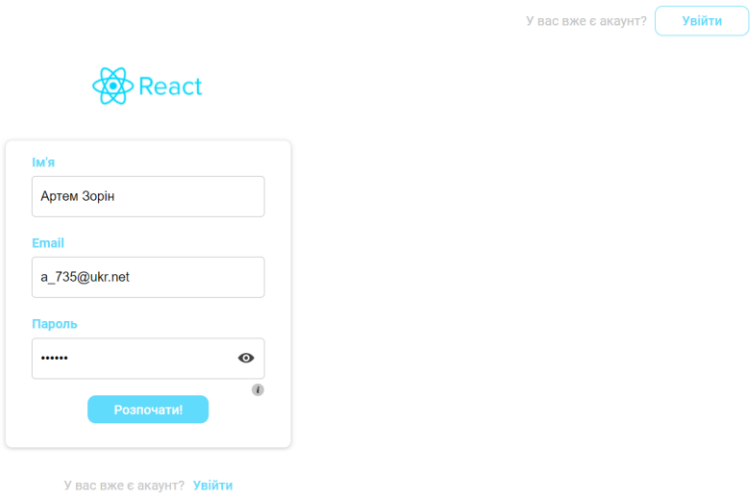

Рисунок 3.10 – Сторінка реєстрації учня в системі

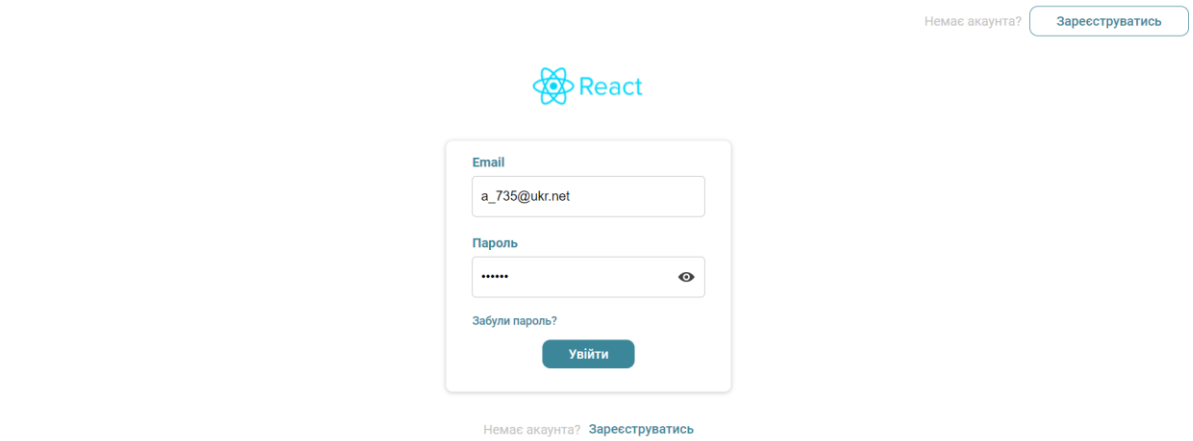

Рисунок 3.11 – Сторінка авторизації учня в системі

Після авторизації учня в системі, відкривається вікно, де можна переглянути доступні курси та прогрес по ним(рис. 3.12).

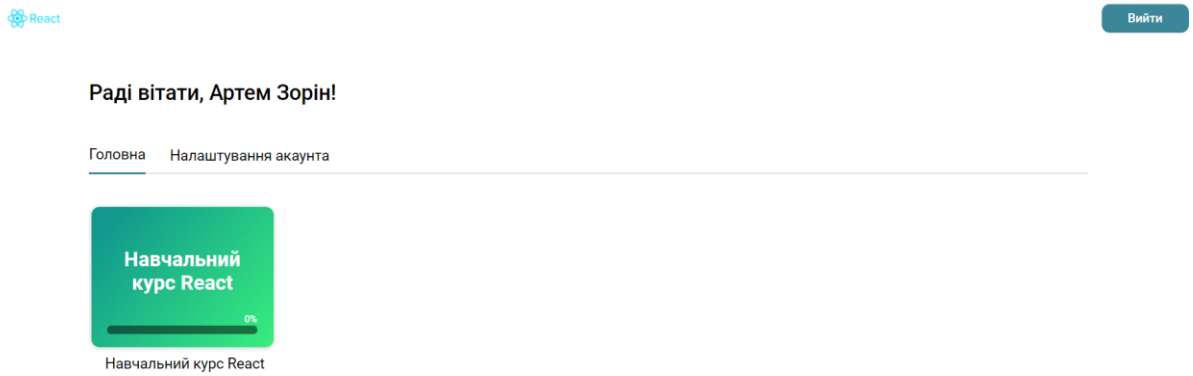

Рисунок 3.12 – Головна сторінка слухача

Також є можливість переходу на сторінку налаштування особистого кабінету, де є можливість зміни мови інтерфейсу(рис. 3.13).

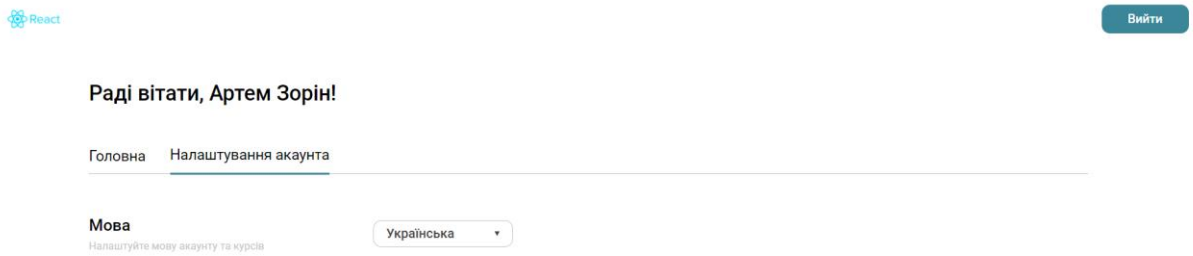

Рисунок 3.13 – Сторінка налаштування особистого кабінету слухача

При натисканні на курс, користувач потрапляє на сторінку, де може пройти навчальний курс(рис. 3.14).

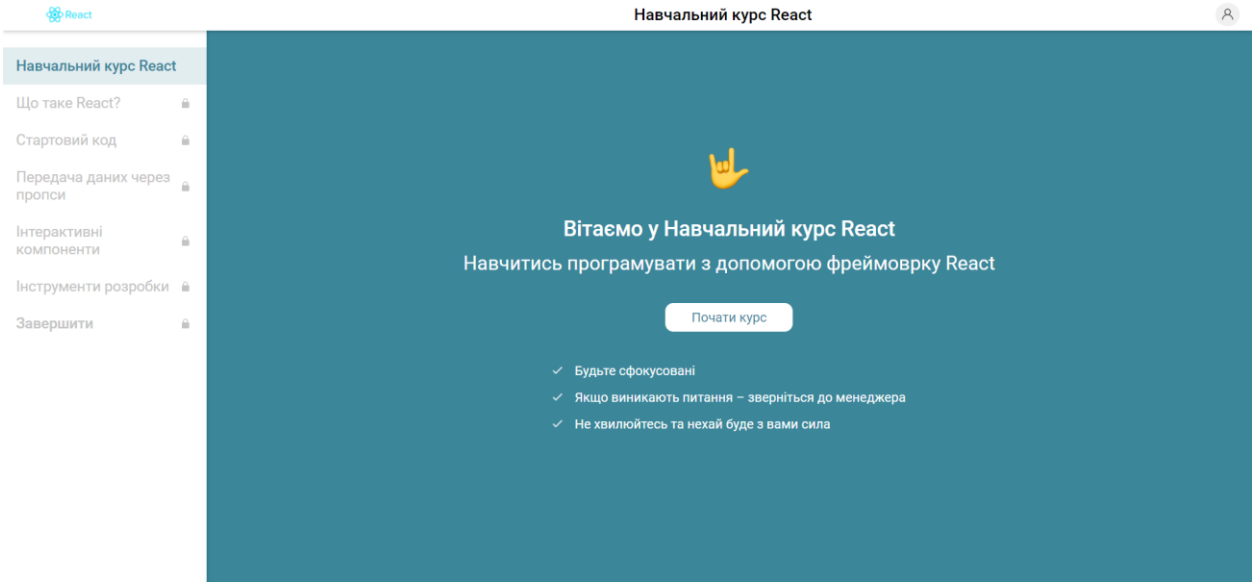

Рисунок 3.14 – Головна сторінка проходження навчального курсу

При проходженні курсу, користувач має можливість переглядати навчальний контент(рис. 3.15) та перейти до складання тестів(рис. 3.16) та виконання задачі(рис. 3.17) в кінці кожного модуля.

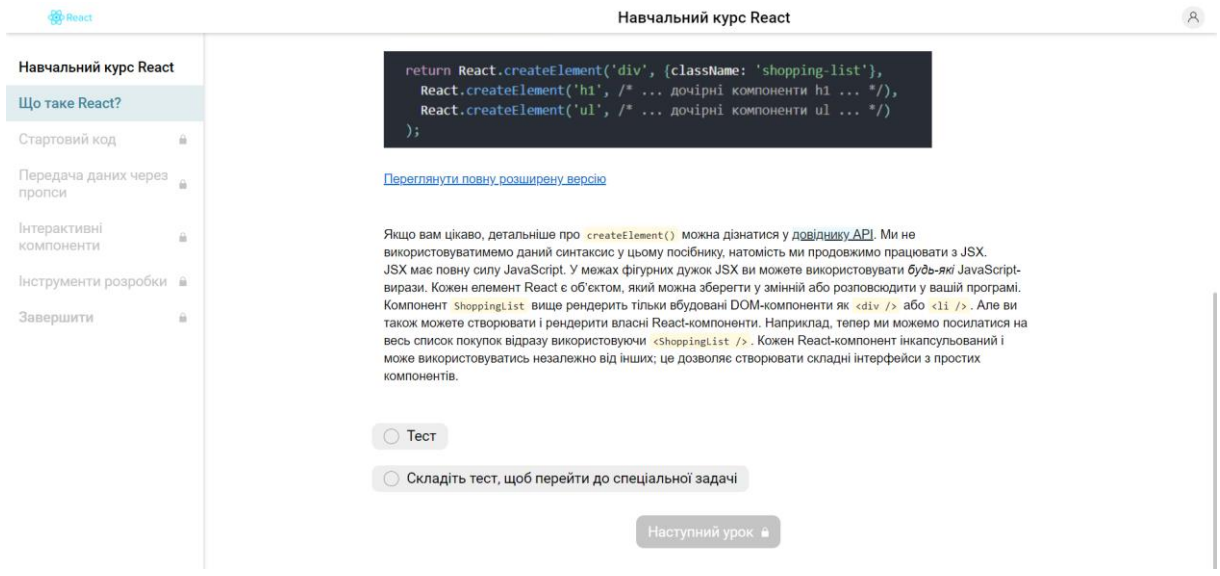

Рисунок 3.15 – Головна сторінка перегляду навчального контенту

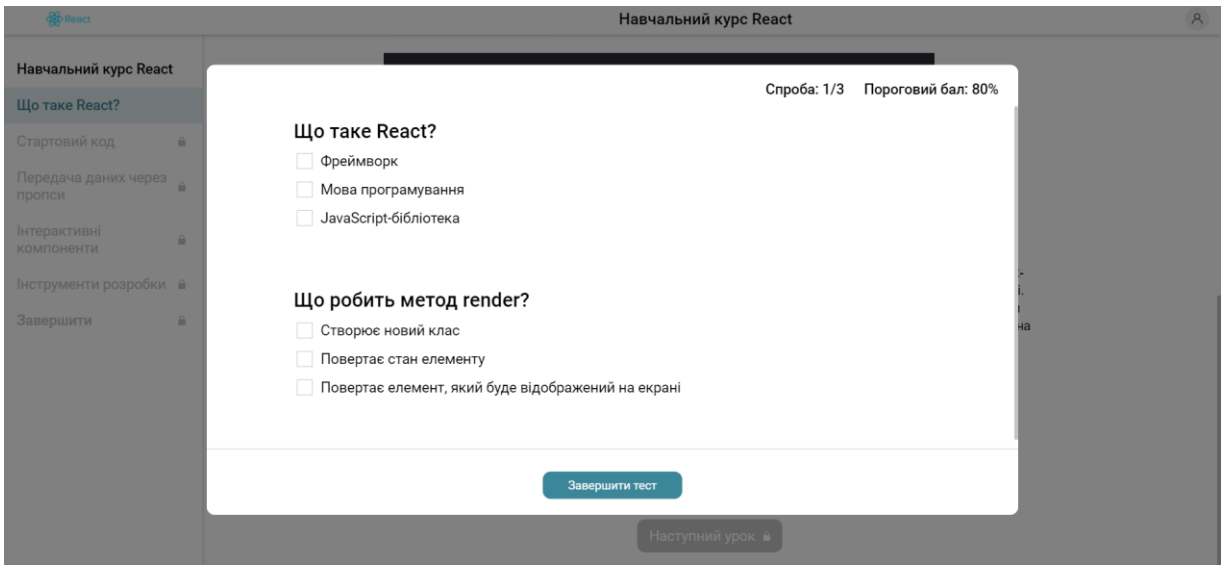

Рисунок 3.16 – Вікно складання тестового завдання

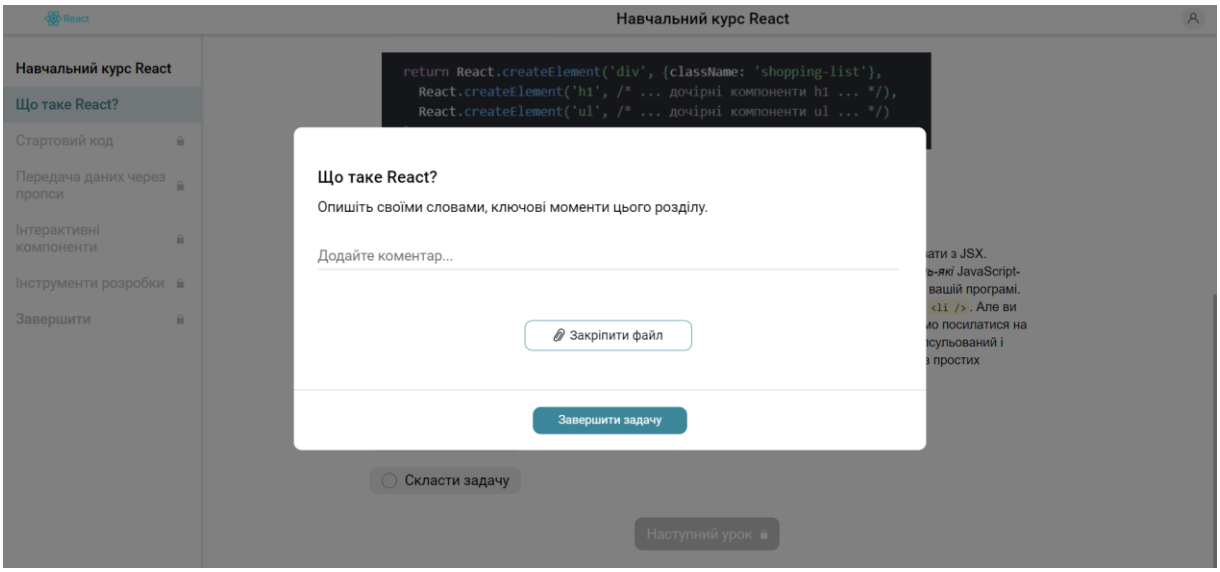

Рисунок 3.17 – Вікно складання задачі

Після складання тестів, буде відображено результат (рис. 3.18). Після проходження тестування а складання задачі, буде відкрито доступ до наступного модуля(рис. 3.19).

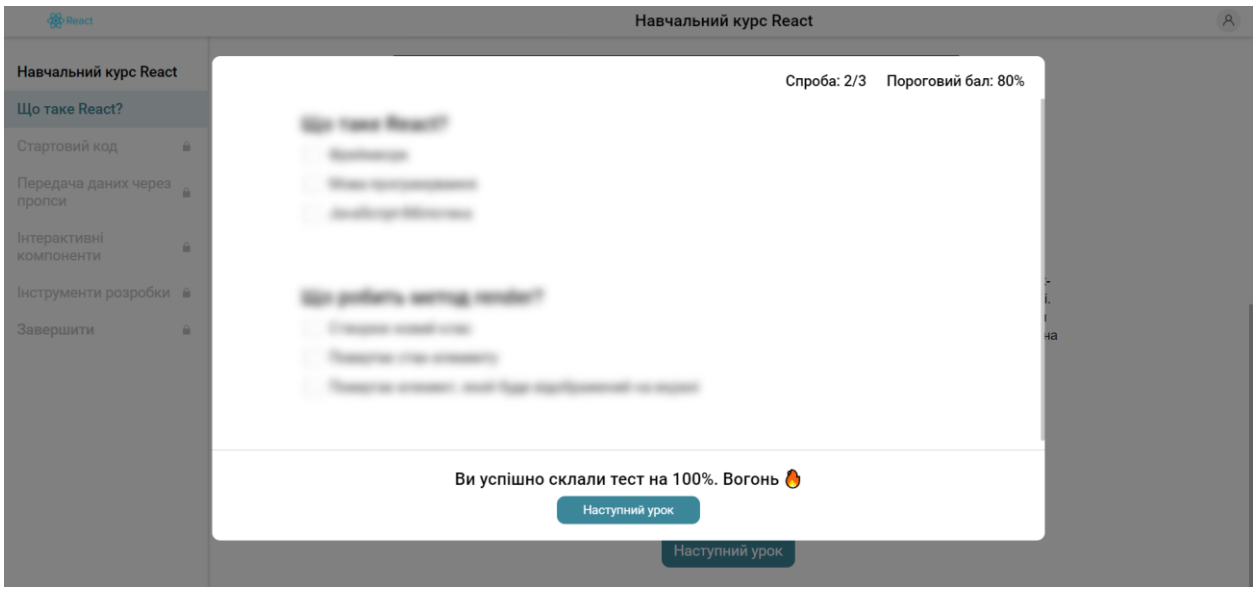

Рисунок 3.18 – Результат тестування

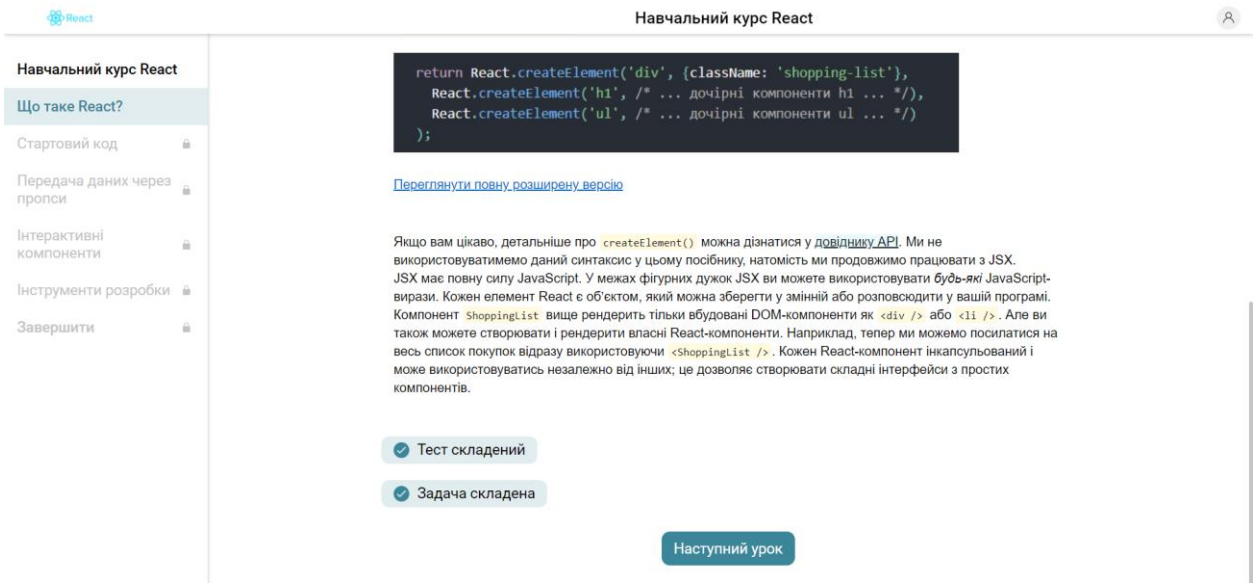

Рисунок 3.19 – Сторінка проходження курсу, після проходження тестування знань

Після завершення навчального курсу(рис. 3.20), можна повернутись в кабінет, для проходження наступних.

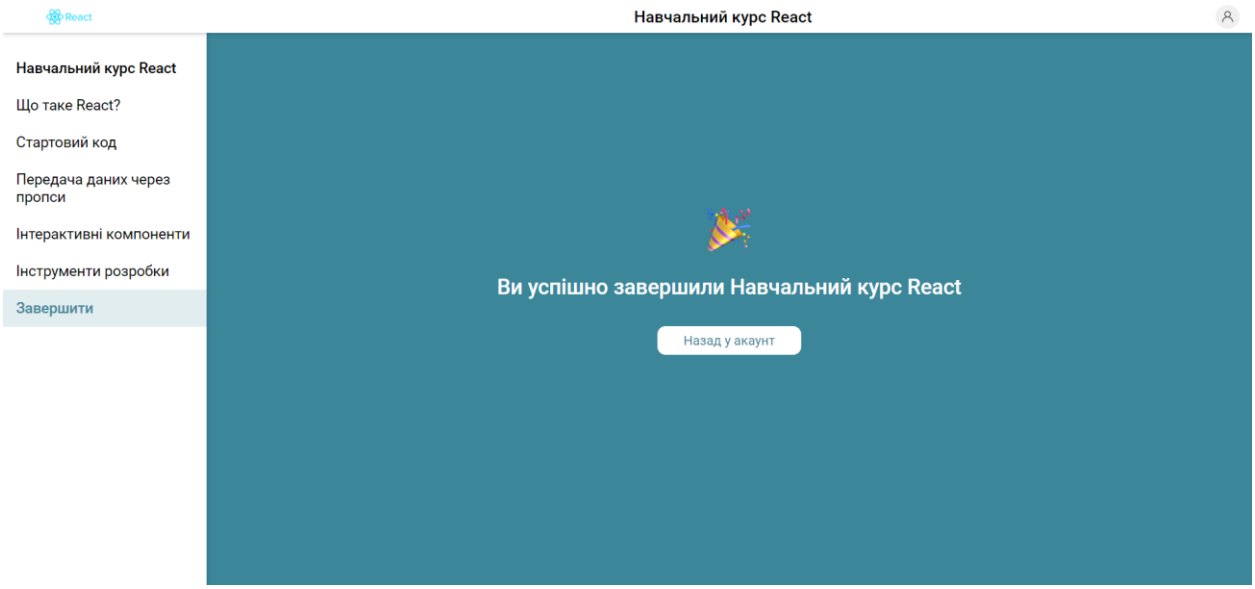

Рисунок 3.20 – Сторінка завершення проходження курсу

В кабінеті вчителя, можна переглянути звіти про виконання тестових завдань та задач учнями(рис. 3.21), а також скористатись фільтрами, для вибірки потрібних звітів. Також передбачена можливість перегляду відповідей учня в окремому вікні(рис. 3.22).

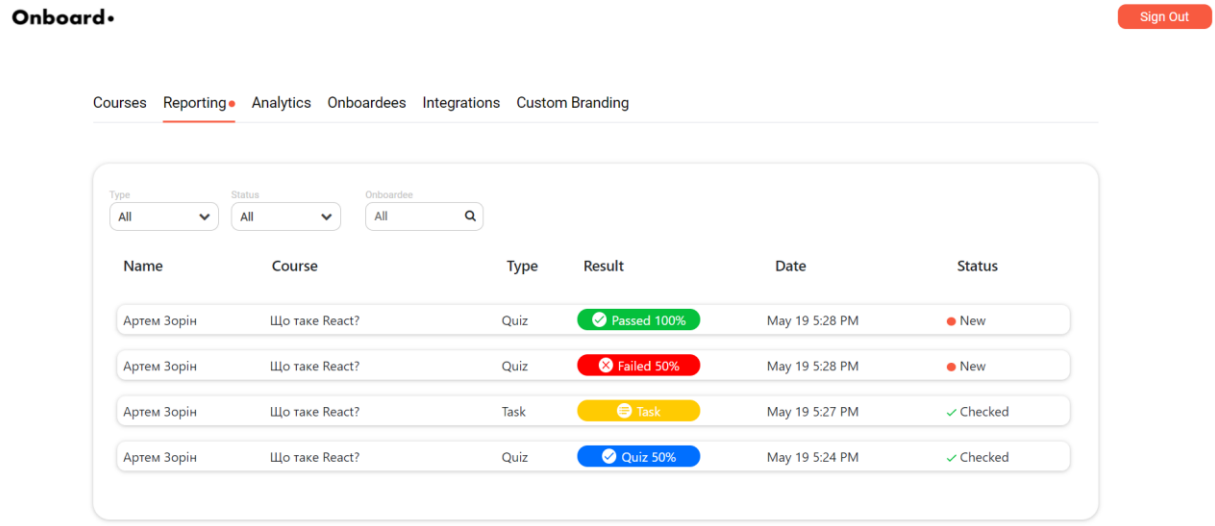

Рисунок 3.21 – Сторінка завершення проходження курсу

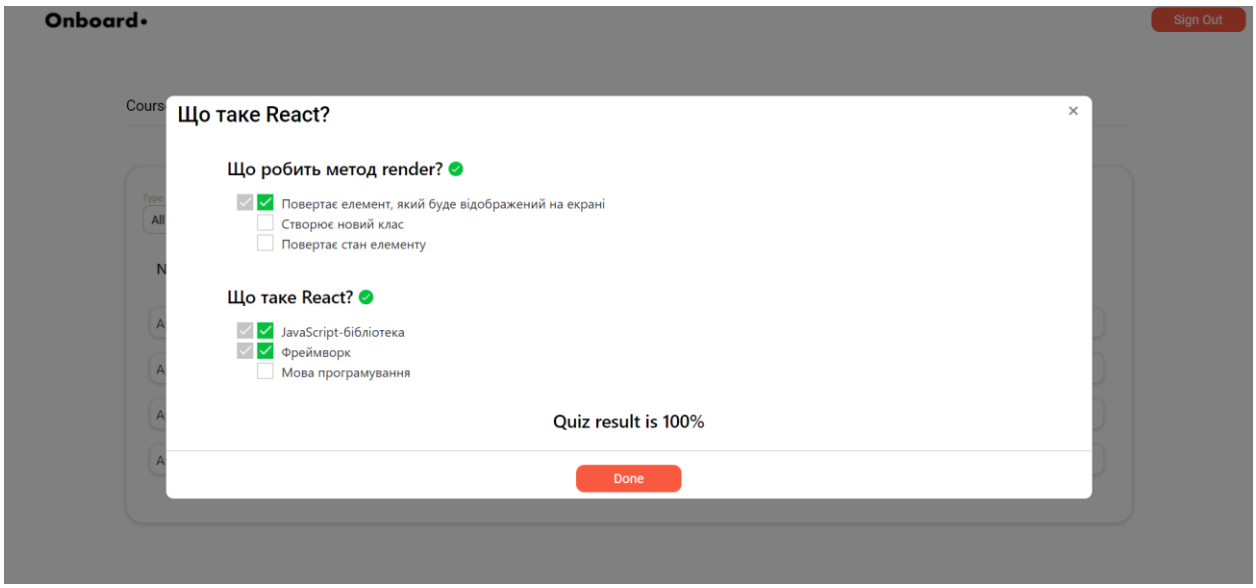

Рисунок 3.21 – Вікно перегляду відповідей учня

Також в кабінеті адміністратора є сторінка з аналітикою, де є можливість переглядати прогрес навчання всіх учнів (рис. 3.22).

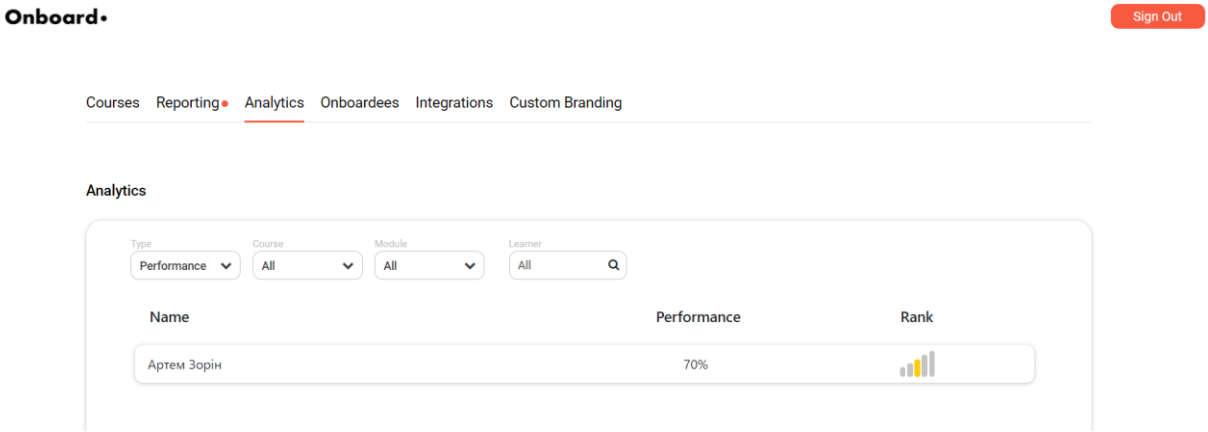

Рисунок 3.22 – Аналітика навчання учнів

Також реалізована сторінка перегляду всіх учнів, та прогресу їх навчання на кожному з курсів(рис. 3.23).

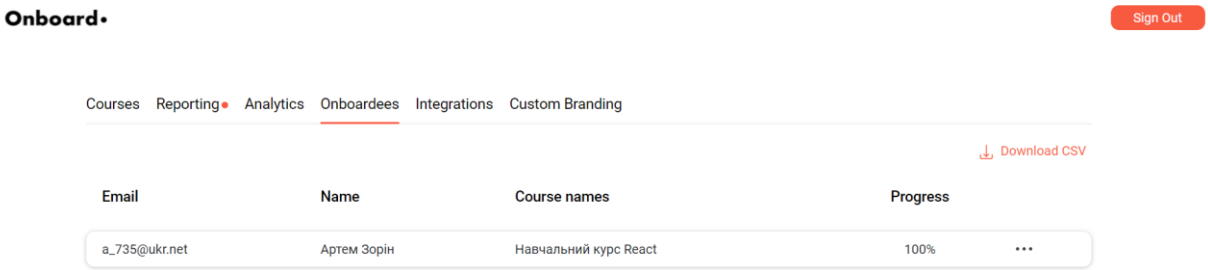

Рисунок 3.23 – Сторінка перегляду учнів

На сторінці інтеграції з сторонніми системами є можливість підключення ботів в популярні месенджери(рис. 3.24), в які будуть приходити

повідомлення про проходження навчальних модулів та прогрес тестування учнів.

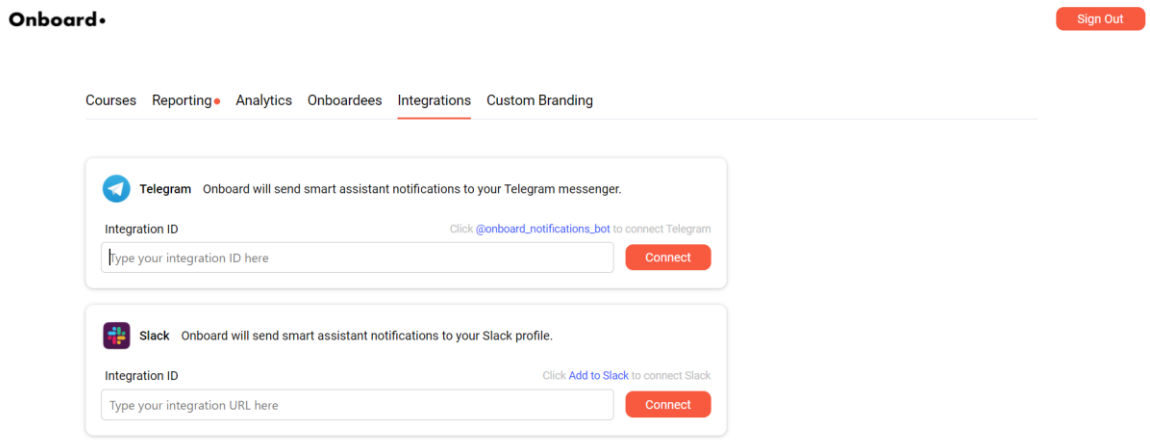

Рисунок 3.24 – Сторінка інтеграції зі сторонніми програмними системами

На сторінці користувацького налаштування інтерфейсу(рис. 2.25), реалізована можливість встановлення користувацького логотипу та кольору, які будуть відображенні при проходженні навчання учнями

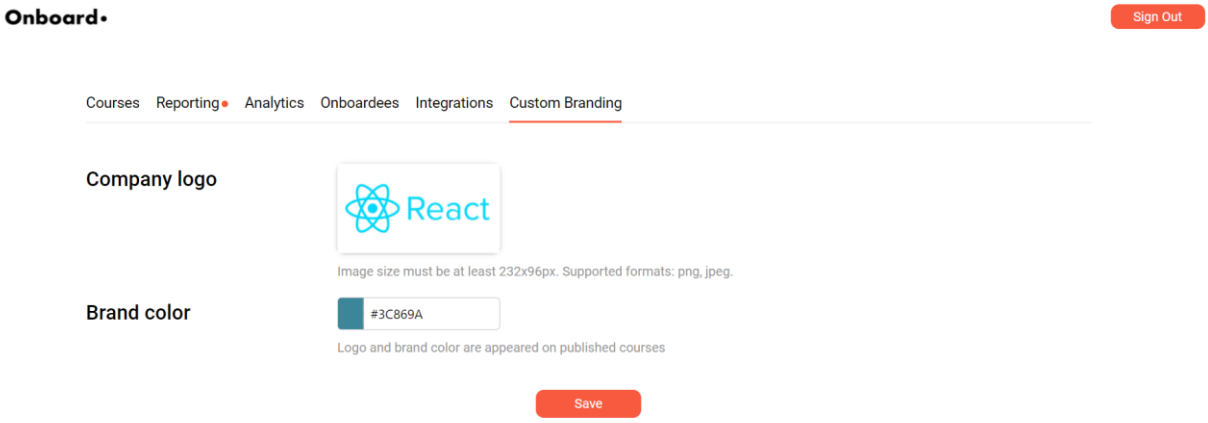

Рисунок 3.25 – Сторінка користувацького налаштування інтерфейсу

Висновки до розділу

В цьому розділі було проаналізовано архітектурні рішення проектування веб-застосунків та програмні засоби, за допомогою яких можна розробити програмний продукт. Було визначено основу задачу, критерії до розробки та оформлення програмної системи, визначено вимоги до програмного забезпечення, візуального оформлення та вимоги відповідно

поставленому завданню, було проведено повний аналіз об’єкту програмування.

У цьому розділі було проведено аналіз розробки програмної системи, наведено схему потоків даних в системі та графічну схему бази даних, надано опис функціональних можливостей веб-застосунку, надано інструкцію користувача та продемонстровано роботу програмного продукту. Відповідно поставленим вимогам, було реалізовано веб-застосунок для навчання та тестування знань. Робота була виконана у середовищі IntelliJ IDEA, WebStorm, за допомогою мови програмування Java(для серверної частини веб-застосунку), JavaScript(для клієнтської частини веб-застосунку) та суміжних технологій HTML, CSS, React тощо.

ВИСНОВОК

Результатом виконання дипломної роботи є створення програмної системи для навчання та тестування знань. Для вирішення завдання дипломної роботи було обрано мови програмування Java та JavaScript, в спільній роботі з популярними фреймворками, які надають ряд функціональних переваг при розробці веб-застосунків та підтримують велику кількість бібліотек для ефективного вирішення задач проектування серверної частини веб-застосунку та інтерфейсу клієнтської частини.

Під час виконання дипломної роботи:

– дослідив теоретичні основи побудови програмної системи навчання та тестування знань;

– проаналізував програмно-технічні рішення і види забезпечень типових веб-застосунків для навчання та тестування знань;

– спроектував і впровадив програмну систему для навчання та тестування знань.

Було проведено аналіз існуючих аналогових програмних рішень, виявлено їх функціональні переваги та недоліки, було проведено аналіз об’єкту розробки та визначено вимоги до інформаційного та програмно-технічного забезпечення, була досліджена загальна концепція принципів роботи веб-застосунку. Також була надана інструкція користувача щодо використання програмної системи навчання та тестування знань.

Основними перевагами розробленого програмного продукту є: зрозумілість та простота у користуванні; мінімальні вимоги до технічних характеристик пристрою за допомогою якого буде надано доступ до веб-застосунку(смартфон, планшет або комп’ютер); впровадження функціоналу на основі проведеного аналізу переваг та недоліків існуючих рішень.

Архітектура розробленого веб-застосунку є може бути масштабованою, що передбачає можливість додавання та впровадження нових функціональних можливостей системи та розширення існуючих в майбутньому.

Розроблений веб-застосунок в повній мірі відповідає основним поставленим вимогам технічного завдання та може конкурувати серед популярних аналогів, завдяки реалізованим функціональним перевагам й простоті у використанні.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розробка веб-застосунків. 2018. URL: https://webstudio2u.net/ua/site-develop/641-razrabotka-veb-prilozheniy.html.

2. Черемних В. Технологія Клієнт-Сервер. 2018. URL: https://it-black.ru/tekhnologiya-kliyent-server/.

3. Змерзлий І. Клієнт-серверна архітектура та ролі серверів. 2017. URL: https://medium.com/@IvanZmerzlyi.

4. Richardson L. RESTful Web APIs: Services for a Changing World. O'Reilly Media, 2013. 406 с.

5. Девід Фленаган JavaScript. Повне керівництво. 7-е видання. Діалектика, 2021. 720 с.

6. Дакетт Д. Javascript і jQuery. Інтерактивна веб-розробка. Ексмо, 2018. 640 с.

7. Макфарланд Д. Нова велика книга CSS. Питер Пресс, 2020. 720 с.

8. Структура і принципи WEB. WEB-сервери та принципи їх роботи. 2015. URL: https://dl.sumdu.edu.ua/textbooks/86975/413008/index.html.

9. Архітектура REST. 2008. URL: https://habr.com/ru/post/38730/.

10. Введення в React Що таке React. 2020. URL: https://metanit.com/web/react/1.1.php.

11. ReactDOM. 2020. URL: https://ru.reactjs.org/docs/react-dom.html.

12. React. JavaScript-бібліотека для створення користувацьких інтерфейсів. 2021. URL: https://ru.reactjs.org/.

13. Опис продукту Intellij IDE. 2020. URL: https://itpro.ua/product/visual-studio-2019-enterprise/?tab=description.

14. Робота з WebStorm. 2020. URL: https://www.jetbrains.com/ru-ru/webstorm/.

15. Демищенко Олена. Онлайн-тестування в освітній та корпоративної сферах. 2015. URL: https://etutorium.ru/blog/onlajn-testirovanie-v-obrazovatelnoj-i-korporativnoj-sferakh.

16. Храмкін Павло. Системи онлайн тестування. 2018. URL: https://www.ispring.ru/elearning-insights/sistema-testirovaniya.

17. Олексій В. Програмування мовою Java. Навчальна книга – Богдан, 2020. 696 с.

ДОДАТКИ

Додаток А

```
const uploadTestsResult = async (isTimeOut) => {
    const isNotEmptyEssayInQuiz = testsAnswersArray.find((item) => item.openTextAnswer !== null);
    setIsEmptyEssayInQuiz(!isNotEmptyEssayInQuiz);

    if((isOnlyEssayInQuiz && isNotEmptyEssayInQuiz) || !isOnlyEssayInQuiz || isTimeOut) {
        setIsVisibleInnerLoader( value: true);
        setIsStartTimer( value: false);
        setTimeOut(isTimeOut);
        setIsStartTimer( value: false);
        const testsResult = await learnerService.uploadTestsResult( testResult: isTimeOut ? null : testsAnswersArray, id);
        const learnerAnalytics = await learnerService.getLearnerAnalytics(id);
        const {quizState} = learnerAnalytics?.data || {};
        const {isAvailableQuiz, isCompletedQuiz} = quizState || {};
        dispatch(setCurrentLearnerAnalytics(learnerAnalytics?.data));

        if(testsResult) {
            setLearnerTestResult(testsResult.data);
            setIsVisibleResultContainer( value: true);
            setIsVisibleFinishQuizButton( value: false);

            if(isAvailableQuiz && !isCompletedQuiz) {
                setIsVisibleRestartQuizButton( value: true);
            }
            if(isCompletedQuiz && isTaskExist && !isTaskCompleted) {
                setIsVisibleDoTheTaskButton( value: true);
            }
            if(isCompletedQuiz && (!isTaskExist || (isTaskExist && isTaskCompleted))){
                setIsVisibleNextModuleButton( value: true);
            }
```

Рисунок А1 – Приклад JavaScript коду сторінки проходження тесту

```
<Modal>
    <PopupFullPage
        classNameBackground={styles.HiddenBackground}
        closePopup={closePopup}
        isSupportOutsidePopupClick={isVisibleResultContainer}
    >
        <div className={styles.PopupContainer}>
            {testSettings &&
                <TestSettingsHeaderComponent
                    learnerAttempts={learner_attempts}
                    totalAttempts={attempts}
                    quizSettings={testSettings}
                    runOutTime={uploadTestsResult}
                    startTimer={isStartTimer}
                />
            }
            <div className={styles.PseudoMainContainer}>
                <div className={`${styles.MainContainer} ${isVisibleResultContainer && styles.BlurMainContainer}`}>
                    {tests?.map((item, index) => {
                        const typeQuestion = item.type;
                        const isEssay = typeQuestion === ESSAY_TYPE;
                        const headerText = item.text;
                        const variants = item.variants;
                        const testIdentifier = item.id;
                        testsAnswersArray.push({testId: testIdentifier,
```

Рисунок А2 – Приклад JSX коду сторінки проходження тесту

```
import axios from 'axios';
import {INVALID_ACCESS_TOKEN_MESSAGE} from '../../consts/error/errorTypesMessage';
import authService from './auth/authService';
import localStorageService from '../localStorage/localStorageService';

const get = (url, requestConfig) => requestApi( method: 'get', url, requestConfig);
const post = (url, requestConfig :{} = {}) => requestApi( method: 'post', url, requestConfig);

const requestApi = async (method, url, requestConfig :{} = {}) => {
    try {
        const accessToken = localStorageService.getAccessToken();
        let config = {method: method, url: url};

        if(!!accessToken) {
            config = {...config, headers: {'Authorization': 'Bearer ' + accessToken}};
        }

        config = {...config, ...requestConfig};

        return await request(config);
    } catch (e) {
        console.log(`requestApi: url: ${url}; requestConfig: ${JSON.stringify(requestConfig)},
        method: ${method}; error: ${e}; response data: ${JSON.stringify(e.response?.data)}`);

        throw e;
    }
};

const request = (requestConfig) => axios.request(requestConfig);
```

Рисунок А3 – Приклад реалізації сервісу для запитів на сервер

```
export const SET_LAYOUT = 'app/global/SET_LAYOUT';
export const SET_IS_APP_LOADER = 'app/global/SET_IS_APP_LOADER';
export const SET_CUSTOM_BRAND_LOGO = 'app/global/SET_CUSTOM_LOGO';
export const SET_CUSTOM_BRAND_COLOR = 'app/global/SET_CUSTOM_BRAND_COLOR';
export const SET_IS_NETWORK_ONLINE = 'app/global/SET_IS_NETWORK_ONLINE';
export const SET_IS_ACTIVE_HEART_BEAT = 'app/global/SET_IS_ACTIVE_HEART_BEAT';

const initialState = {
    isAppLoader: true,
    customBrandLogo: undefined,
    customBrandColor: undefined,
    isActiveHeartBeat: false,
};

export default function reducer(state :{…} = initialState, action) {
    switch (action.type) {
        case SET_LAYOUT: {
            return {...state, layout: action.payload};
        }
        case SET_IS_APP_LOADER: {
            return {...state, isAppLoader: action.payload};
        }
        case SET_CUSTOM_BRAND_LOGO: {
            return {...state, customBrandLogo: action.payload};
        }
        case SET_CUSTOM_BRAND_COLOR: {
            return {...state, customBrandColor: action.payload};
        }
```

Рисунок А4 – Приклад реалізації сховища даних на клієнтській стороні

```
const LoginLearnerPage = (props) => {
    const {customBrandLogo, layout} = props;
    const currentLogo = customBrandLogo ? customBrandLogo : defaultLogo;
    const isDesktopLayout = layoutService.isDesktopLayout(layout);
    const localizationStrings = localizationService.getLocalizedStrings();
    const [loginErrorType, setLoginErrorType] = useState( initialState: false);
    const isPhoneAuthentication = localStorage.getItem( key: 'isPhoneAuthentication');

    const onToSignUpButtonClick = () => {
        appRouterService.forwardToLearnerSignUpPage();
    };

    const onLoginButtonClick = async (email, phone, password) => {
        if ((email || phone) && password) {
            try {
                localStorage.setItem('learnerEmail', (email || phone));
                await authService.loginLearner( learnerEmail: email || `+${phone}`, password);
            } catch (e) {
                if(e.response.data === INVALID_LOGIN_WRONG_IP) {
                    routerService.selfLinkOpen(config.data.wrongIpPagePath);
                }
                setLoginErrorType(e.response.data);
            }
        } else {
            setLoginErrorType( value: true);
        }
    };
```

Рисунок А5 – Приклад реалізації сховища даних на клієнтській стороні

```
import {setIsNetworkOnline} from '../../../state/ducks/global';
import {dispatch} from '../../../state/store';
import NoInternetConnectionPopup from '../../../components/PopupFullPage/NoInternetConnectionPopup/NoInternetConnectionPopup';

const OfflineDetector = () => {
    const changesInternetConnection = (isOnline) => {
        if (isOnline) {
            dispatch(setIsNetworkOnline( value: true));

            console.log(`OfflineDetector: Internet connection is online`);
        } else {
            dispatch(setIsNetworkOnline( value: false));

            console.log(`OfflineDetector: Internet connection is offline`);
        }
    };

    return (
        <>
            <Offline
                polling={{
                    interval: 5000,
                }}
                onChange={changesInternetConnection}
            />
            <NoInternetConnectionPopup />
        </>
    );
};
```

Рисунок А6 – Приклад реалізації функціоналу перевірки стабільного підключення користувача до мережі Інтернет

```
spring.datasource.url=jdbc:postgresql://localhost:5432/mvp_2
spring.datasource.username=postgres
spring.datasource.password=123
spring.jpa.generate-ddl=false

spring.freemarker.expose-request-attributes=true


spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.temp.use_jdbc_metadata_defaults = false

spring.session.jdbc.initialize-schema=always
spring.session.jdbc.table-name=SPRING_SESSION

upload.path=E:/Ideaprojects1/mvp/uploads
```

Рисунок А7 – Приклад реалізації функціоналу підключення серверу до БД

```
/** Sets title for a Notification for Reports with MULTIPLE_CHOICE questions withOUT passing grade in TestSettings */
public void setNotificationTitleFormat(Notification notification,
                                       Learner learner,
                                       Module module,
                                       Double testScore) {
    String moduleName  = module.getName();
    String courseName  = module.getCourse().getName();
    String learnerName = learner.getName();

    String notificationTitle, notificationType;

    notificationTitle = String.format("Check %s quiz report on %s in %s with %.1f%%.", learnerName, moduleName, courseName, testScore);
    notificationType  = PASSED.name();

    notification.setTitle(notificationTitle);
    notification.setType(notificationType);
}


/** Sets title for a Notification for Reports with ESSAY questions only */
public void setNotificationTitleFormat(Notification notification,
                                       Learner learner,
                                       Module module) {
    String moduleName  = module.getName();
    String courseName  = module.getCourse().getName();
    String learnerName = learner.getName();

    String notificationType  = ESSAY.name();
    String notificationTitle = String.format("Check %s quiz report on %s in %s with open answers", learnerName, moduleName, courseName);

    notification.setTitle(notificationTitle);
    notification.setType(notificationType);
}
```

Рисунок А8 – Приклад реалізації функціоналу створення звітів навчання