Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно–інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб–застосунку для перевірки знань з ПДР

Виконав: студент IV курсу, групи СН–41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Швець О.Я.
(підпис) (прізвище та ініціали)

Керівник _____ Марценко С.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Литвиненко Я.В
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Микитишин А.Г.
(підпис) (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно–інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Швецю Олександру Ярославовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка веб–застосунку для перевірки знань з ПДР

Керівник роботи Марценко Сергій Володимирович , к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 7 » лютого 2023 року № 4/7–133

2. Термін подання студентом завершеної роботи 20 червня 2023р.

3. Вихідні дані до роботи Літературні та інтернет джерела.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Постановка задачі. Підбір бібліотек та технологій розробки. 1.1 Аналіз предметної області. 1.2 Огляд існуючих рішень. 1.3 Вибір середовища розробки. 1.4 Огляд використаних технологій. 1.5 Формування структури веб–застосунку. 1.6 Висновки до першого розділу. 2.Проєктування та реалізація веб–застосунку для перевірки знань. ПДР 2.1 Розробка серверної частини на базі Node.js. 2.2 Розробка клієнтської частини на базі React.js. 2.3 Опис типової схеми використання веб застосунку. 2.4 Висновки до другого розділу. 3. Безпека життєдіяльності, основи охорони праці. 3.1 Безпека при надзвичайній ситуації в будівлі сервісного центру МВС. 3.2 Протипожежна безпека в сервісному центру МВС. 3.3 Висновок до третього розділу. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульна сторінка. 2. Мета, об'єкт та предмет дослідження. 3. Аналіз існуючих рішень. 4. Підбір середовища розробки. 5. Огляд використаних технологій. 6. Структура веб-застосунка. 7. Розробка серверної частини. 8. Розробка клієнтської частини. 9. Вигляд користувацької частини сайту. 10. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я., доцент кафедри МТ	05.06.2023	08.06.2023

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.01.2023	*Виконано*
2.	Підбір джерел про розробку веб-застосунків для	24.01.2023–26.01.2023	*Виконано*
	перевірки знань ПДР		
3.	Опрацювання джерел по темі кваліфікаційної роботи	27.01.2023–31.01.2023	*Виконано*
4.	Виконання дослідження щодо веб-застосунків для	01.02.2023–07.02.2023	*Виконано*
	перевірки знань ПДР. Підбір технологій для		
	розробки серверної та клієнтської частини застосунка.		
	Розроблення веб-застосунку для перевірки знань ПДР		
5.	Оформлення розділу «Постановка задачі. Підбір	08.02.2023–09.02.2023	*Виконано*
	бібліотек та технологій розробки»		
6.	Оформлення розділу «Проєктування та реалізація	10.02.2023–12.02.2023	*Виконано*
	веб–застосунку для перевірки знань»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	05.06.2023–06.06.2023	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	07.06.2023–08.06.2023	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	09.06.2023–11.06.2023	*Виконано*
10.	Нормоконтроль	12.06.2023–13.06.2023	*Виконано*
11.	Перевірка на плагіат	14.06.2023	*Виконано*
12.	Попередній захист кваліфікаційної роботи	15.06.2023	*Виконано*
13.	Захист кваліфікаційної роботи	20.06.2023	

Студент _____ Швець О.Я.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Марценко С.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка веб–застосунку для перевірки знань з ПДР // Кваліфікаційна робота освітнього рівня «Бакалавр» // Швець Олександр Ярославович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно–інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН–41 // Тернопіль, 2023 // С. – 53 , рис. – 15 , табл. – 1, слайди – 10, додат. – 5, бібліогр. – 30.

Ключові слова: бази даних, веб–застосунок, компоненти, програмування, MongoDB, Node.js, React, Apache.

Кваліфікаційна робота присвячена розробці веб-застосунку для перевірки знань з ПДР.

Метою даної кваліфікаційної роботи є розробка веб-застосунку для тренування учнів водійських шкіл та майбутніх водіїв до теоретичного екзамену із правил дорожнього руху.

В першому розділі кваліфікаційної роботи проведено аналіз предметної області, огляд вже існуючих рішень, підбір середовища розробки проєкту, огляд використаних технологій та сформовано структуру майбутнього веб-застосунку.

В другому розділі кваліфікаційної роботи описано розробку клієнтської та серверної частини веб-застосунку за допомогою технологій Node.js Apollo Server, Express, MongoDB та React.js, Axios відповідно. Та описано типову схему використання веб-застосунку користувачем.

В третьому розділі кваліфікаційної роботи розглянуто питання щодо безпеки при надзвичайній ситуації будівлі сервісного центру МВС та протипожежної безпеки в сервісному центру МВС.

ANNOTATION

Web Application Development for Knowledge Testing on Traffic Rules // Qualification work of the educational level "Bachelor" // Shvets Oleksandr Yaroslavovych // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN–41 // Ternopil, 2023 // P. – 53 , fig. – 15 , tabl. – 1 , chair. – 10 , annexes. – 5 , references – 30 .

Keywords: databases, web applications, components, programming, MongoDB, Node.js, React, Apache.

The qualification work is dedicated to the development of a web application for testing knowledge of traffic rules.

The goal of the work of this qualification work is to develop a web application for training driving school students and future drivers for the theoretical examination on traffic rules.

The first section of the qualification paper includes an analysis of the subject area, an overview of existing solutions, selection of the project development environment, an overview of the technologies used, and the formation of the structure of the future web application.

In the second section of the qualification work describes the development of the client and server parts of the web application using Node.js, Apollo Server, Express, MongoDB, and React.js, as well as Axios. It also describes the typical usage scenario of the web application by the user.

The third section of the qualification work discusses issues related to emergency safety in the building of the Ministry of Internal Affairs service center and fire safety in the MIA service center.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – певний набір структур, класів чи процедур, що використовуються для взаємодії одного компоненту програми з іншими.

CSS(від англ. Cascading Style Sheets) – каскадні таблиці стилів.

DB (від англ. Database) – база даних.

HTTP (від англ. HyperText Transfer Protocol) – протокол прикладного рівня передачі даних.

IDE(від англ. Integrated development environment) – інтегроване середовище розробки.

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

JSON Web Token (JWT) - це стандарт для представлення токенів безпеки у форматі JSON.

URL(від англ Uniform Resource Locator) – унікальний ідентифікатор, який використовується для пошуку ресурсу в Інтернеті.

VS Code – Visual Studio Code.

БД – база даних.

МВС – міністерство внутрішніх справ.

ПДР – правила дорожнього руху.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі дорожня безпека є однією з найважливіших проблем, які потребують уваги та вирішення. Дотримання правил дорожнього руху необхідна умова для запобігання аваріям та збереження життя. Щороку сотні тисяч людей здають обов'язкові іспити з теоретичної та практичної частини аби отримати право водіння транспортним засобом. Тому для досконалої підготовки, кожен майбутній водій шукає сервіси, де пропонуються онлайн тести максимально схожі із справжнім екзаменом, із доступом перегляду помилок та загальними результатами учня.

На основі цього можна зробити висновок що розробка подібного веб–застосунка є актуальним та затребуваним напрямом дослідження . З метою надання доступу до зручного та ефективного засобу навчання правилам дорожнього руху, у даній дипломній роботі пропонується розробка веб–застосунку, який дозволить користувачам перевіряти свої знання та отримувати інформацію про правила дорожнього руху.

Мета і задачі дослідження. Мета цієї кваліфікаційної роботи освітнього рівня «Бакалавр», є розробка веб–застосунку для перевірки знань з правил дорожнього руху Щоб досягнути поставлену мету, потрібно виконати ряд завдань, зокрема:

– проаналізувати вибрану предметну область;

– спроектувати БД, та розробити моделі для зберігання даних;

– провести проектування робочої архітектури веб–застосунку;

– розробити веб–застосунок відповідно до поставлених вимог;

– провести повне тестування усіх функціональних можливостей веб–застосунку.

Практичне значення одержаних результатів. Розроблений у ході виконання веб–застосунок матиме можливість перегляду усіх освітніх матеріалів із офіційних джерел, а також доступ до тестування знань. Після закінчення тесту користувачу буде доступний список з всіх своїх питань. У разі

помилки у завданні відповідь підсвічуватиметься червоним, а правильна зеленим, якщо ж він відповів правильно відповідь світитиметься зеленим. Незалежно від того, чи було питання пропущене, правильні відповіді будуть виділені в загальному списку. Для наочності буде надана статистична діаграма, яка покаже кількість правильних, неправильних та пропущених відповідей.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ. ПІДБІР БІБЛІОТЕК ТА ТЕХНОЛОГІЙ РОЗРОБКИ

1.1 Аналіз предметної області

Запорука в успішному складанні іспитів на отримання водійського посвідчення залежить від рівня підготовки майбутніх водіїв. Оскільки правила дорожнього руху та питання в екзаменаційних білетах постійно змінюються, складно забезпечити їх постійну актуальність у друкованому вигляді. Завдяки розвитку та поширенню Інтернету, доступ до актуальної інформації більше не є проблемою. Створення веб–сайту з корисними матеріалами для водіїв допоможе учням автошкіл готуватись до екзаменів і поліпшить знання вже досвідчених водіїв.

Основні складові веб–застосунку з підготовки водіїв включають:

– Авторизація та реєстрація.
– Тести за різними темами ПДР.
– Тести за екзаменаційними білетами.
– Можливість перегляду Статистики результатів.

Для отримання доступу до даних веб–сайту необхідно буде зареєструватися у системі. Користувач ідентифікується за допомогою свого електронного адресу та пароля. У разі відсутності облікового запису користувач має можливість зареєструватися в системі, вводячи свою електронну адресу та придуманий пароль. На сторінці профілю користувача відображаються його особисті дані, такі як ім'я, електронна адреса та фотографія.

Тестування користувача може здійснюватися як повноцінний екзаменаційний тест із 20 випадковими питаннями, так і відповісти на одне випадкове питання. Також надається інформація про його прогрес у

проходженні білетів, включаючи кількість правильних, неправильних відповідей та пропущених питань.

1.2 Огляд існуючих рішень

На сьогоднішній день в україномовному просторі існує безліч різних тренувальних веб–застосунків для підготовки майбутніх водіїв. Серед них для порівняння з розроблюваною системою були вибрані найпопулярніші та найзручніші на думку користувачів сервіси, а саме: «Vodiy.ua», «Green Way» та «Road Rules» .

Для порівняльного огляду даних сервісів було прийнято рішення розробити таблицю у якій міститься усі необхідні критерії до веб–застосунку аби він був успішним серед користувачів та учнів, а саме: зручність та зрозумілість інтерфейсу, доступність інформації, відсутність зайвої інформації, зручність проходження тестування, та актуальність тестів.

Таблиця 1.1 – Огляд існуючих рішень

	Vodiy.ua	**Green Way**	**Road Rules**
Зручність інтерфейсу	Складний та нагромаджений, надто багато реклами	Зрозумілий, але перенасичений	Інтерфейс розуміється інтуїтивно, приємний оку
Доступність інформації	Інформація постійно оновлюється, тести посортовані по категоріях водіїв, але видаються випадково. Аналіз помилок платний.	Інформація доступна, тести видаються кількома способами: одиночні, 20 випадкових та іспит. Аналіз помилок та іспит	Тести посортовані по основних темах, інтерфейс розуміється інтуїтивно, проходження та аналіз відповідей безкоштовні

Продовження таблиці 1.1

Відсутність зайвої інформації	Перенасичений інформацією	Присутньою багато реклами та продаж продукції однойменної школи	Зайва інформація відсутня
Зручність проходження тестування	Середня	Середня	Висока
Актуальність тестів	Тести постійно оновлюються	Тести постійно оновлюються	Інформація є застарілою, останнє оновлення 2021 року

В результаті порівняльної характеристики можна зробити висновок що в більшості такі веб–застосунки розробляють вже існуючі школи водіння для тренування свої студентів та як рекламна компанія для майбутніх. Єдине що засмучує, на таких сервісах надто багато лишньої інформації, реклами та продажу своїх книжок та продукції.

На відміну від Vodiy.ua та Green Way, Road Rules не є дочірнім сервісом водійської школи, а є самостійним продуктом, містить багато корисної та добре структурованої інформації, але нажаль навчальний матеріал є застарілим, останнє оновлення на сайті проводилося ще 2021 року.

1.3 Вибір середовища розробки

Для полегшення розробки веб–застосунків програмісти мають можливість вибрати з різних середовищ розробки (IDE), які надають зручний інструментарій для програмування. На сьогоднішній день існує надзвичайно велика кількість різних IDE, під різні мови програмування та задачі і вимоги розробників. Для створення даного проекту вибір стояв між WebStorm та Visual Studio Code. Для того щоб обрати найкраще середовище для даного проекту, потрібно більше про них дізнатися та порівняти наскільки вони зручні в написанні веб–застосунків.

WebStorm є платним інтегрованим середовищем розробки від компанії JetBrains, спеціалізованою на розробці веб–застосунків мовою JavaScript[10]. Надає розширені можливості для написання, редагування та налагодження коду JavaScript, а також підтримку інших технологій, таких як HTML, CSS, Node.js та фреймворки JavaScript, наприклад React і Angular[8].

На рисунку 1.1 зображено базовий інтерфейс середовища розробки WebStorm.

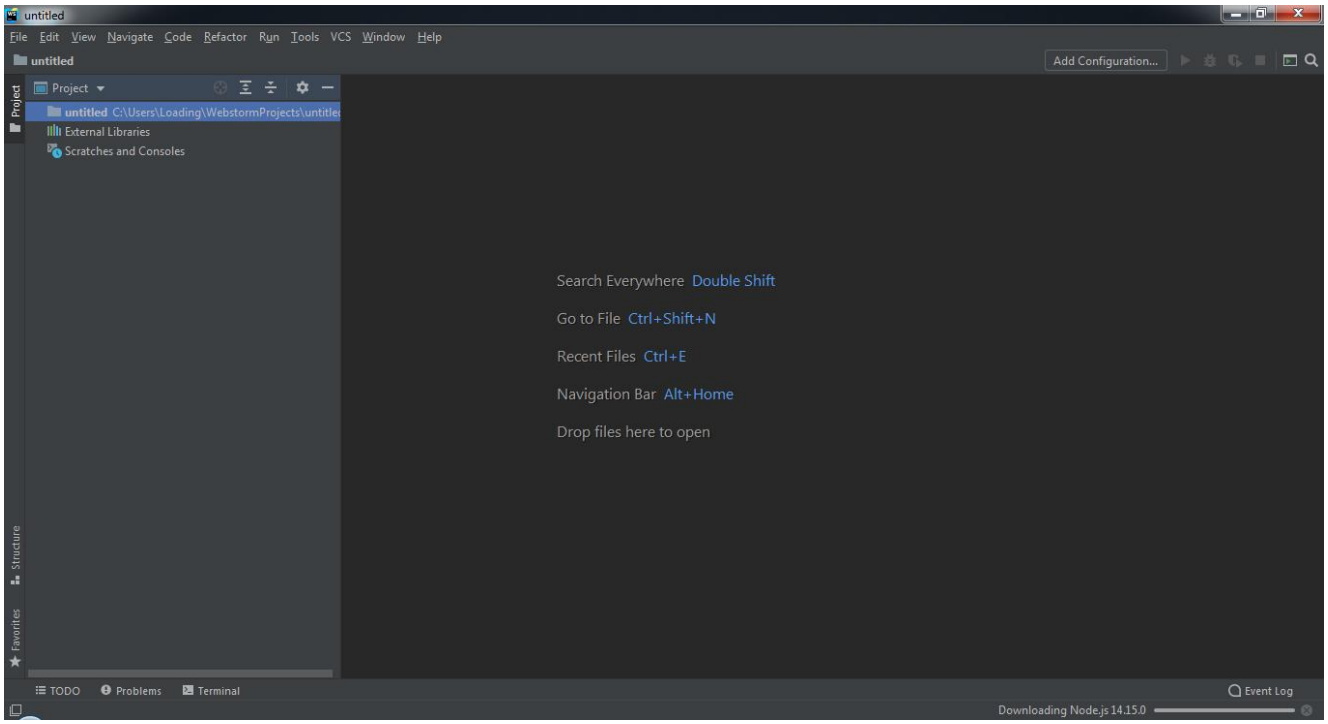

Рисунок 1.1 – Інтерфейс WebStorm

Visual Studio Code є безкоштовним редактором від компанії Microsoft, який також надає потужні інструменти для розробки веб–застосунків мовою JavaScript[10], у вигляді розширень. Середовище має широкий набір функцій, розширень та плагінів, які дозволяють розширити його функціональність та настроїти його під потреби розробника. Це IDE доступне для популярних операційних систем, таких як Windows, macOS і Linux, а також є безкоштовним і з відкритим вихідним кодом. Розробники, які є користувачами GitHub, можуть зробити свій власний внесок у подальший розвиток цієї програми[25].

На рисунку 1.2 зображено інтерфейс середовища розробки Visual Studio Code у процесі розробки.

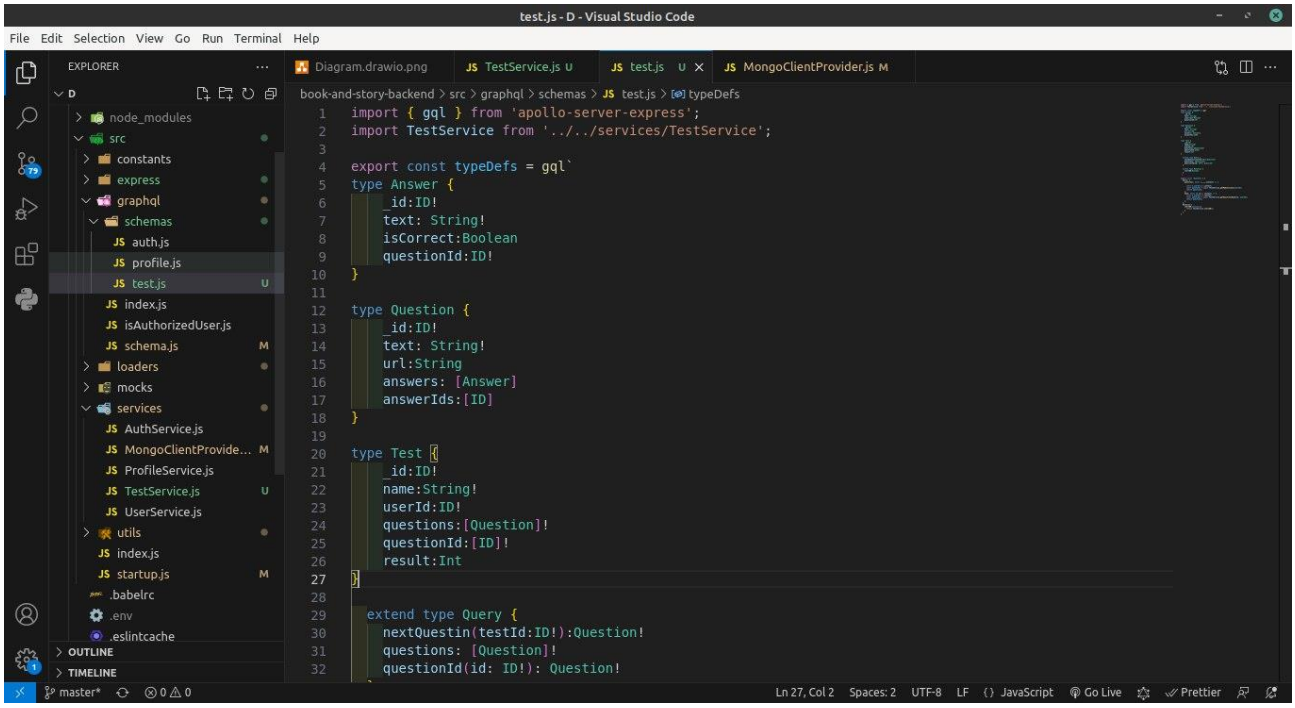

Рисунок 1.2 – Інтерфейс Visual Studio Code

Порівняння WebStorm та Visual Studio Code для розробки веб–застосунків мовою JavaScript:

– WebStorm має вбудовану підтримку фреймворків, інструментів та систем контролю версій, що полегшує розробку веб–застосунків. Середовище надає розширений рефакторинг, автодоповнення коду, налагоджування та інші корисні інструменти. VS Code також має широкий спектр функціональності та може бути настроєний за допомогою розширень. Середовище надає базові можливості для редагування, налагодження та автодоповнення коду.

– Обидва середовища розробки працюють досить швидко, але можуть вимагати більше ресурсів порівняно з легшими текстовими редакторами. Використання швидкого та потужного комп'ютера може покращити продуктивність в роботі з обома середовищами.

– VS Code має велику кількість розширень та плагінів, що дозволяють розширити його функціональність та настроїти робочий процес. Можна знайти

розширення для різних задач, включаючи розробку веб–застосунків мовою JavaScript, а також багато інших розширень які тим чи іншим способом допомагають у розробці. WebStorm також має розширення та плагіни, але їх вибір помітно менший порівняно з VS Code.

– WebStorm готовий до початку розробки зразу після встановлення, тоді як VS Code потребує встановлення потрібних розширень. Через велику кількість розширень, недосвідченому розробнику може бути важко швидко налаштувати VS Code під веб розробку.

Обидва середовища розробки, WebStorm та Visual Studio Code, мають свої переваги та підходять для розробки веб–застосунків мовою JavaScript. Якщо потрібне потужне та повнофункціональне IDE з вбудованою підтримкою фреймворків та розширеними інструментами, WebStorm може бути кращим варіантом. З іншого боку, потрібен легкий та настроюваний редактор з великою кількістю розширень та плагінів, які можна використовувати для налаштування функціональності під свої потреби, Visual Studio Code може стати кращим варіантом. Більшість розробників початківців обирають VS Code, у зв’язку з тим що дане середовище розробки є безкоштовним.

Для даного проекту, в якості середовища розробки було обрано Visual Studio Code, основні чинники які привели до цього вибору це велика кількість розширень і плагінів за допомогою яких, можна налаштувати середовище під розробника. Також великим плюсом VS Code є його доступність, так як середовище безкоштовне.

1.4 Огляд використаних технологій

В якості платформи для розробки, було обрано Node.js[14]. Це відкрита платформа, яка дозволяє виконувати JavaScript–код на стороні сервера. В основі Node.js лежить рушій V8[20], який до цього використовується в браузері Google Chrome для виконання JavaScript, але в подальшому його змогли адаптувати під розробку на любих пристроях.

Одним з ключових принципів Node.js[14] є асинхронний та дієвий підхід до програмування. Це означає, що платформа може обробляти багато запитів одночасно без блокування виконання інших операцій, що забезпечує високу продуктивність та швидкодію веб–додатків. Node.js дозволяє розробляти повноцінні серверні додатки за допомогою JavaScript. Це забезпечує єдиноформованість мови програмування між клієнтською та серверною частиною додатків, що в свою чергу значно полегшує розробку. Платформа має вбудовану модульну систему, яка дозволяє організувати код в окремі модулі, що також спрощує розробку та підтримку додатків.

Node.js використовує пакетний менеджер npm[15], який є одним з найбільших репозиторіїв відкритих пакетів програмного забезпечення. За допомогою npm розробники можуть швидко встановлювати, оновлювати та використовувати різноманітні пакети, що додають функціональність до їхніх додатків.

Платформа Node.js широко використовується для розробки серверних додатків, веб–додатків, API, real–time додатків та багатьох інших сценаріїв. Він став популярним в програмістській спільноті завдяки своїй продуктивності, простоті використання та широкому спектру інструментів та пакетів.

1.4.1 Огляд використаних технологій для розробки серверної частини

Розробка серверної частини передбачає поєднання потужних технологій для створення надійної та ефективної системи. GraphQL, Apollo Server, Express, MongoDB і Studio 3T є ключовими компонентами, які відіграють значну роль у цьому процесі.

GraphQL є мовою запитів та середовищем, яке дозволяє описувати структуру та типи даних API[7] і робити запити до сервера. Замість традиційного REST–інтерфейсу, де клієнт отримує всю доступну інформацію, GraphQL дозволяє клієнту вказати, які саме дані йому потрібні, що зменшує надмірні завантаження та покращує продуктивність[6].

Apollo Server – це серверна реалізація GraphQL. Він надає засоби для створення та розгортання GraphQL API на стороні сервера. Apollo Server дозволяє визначати схему даних, обробляти запити, виконувати логіку бізнес–процесів та повертати відповіді на клієнтські запити[1].

Express є легким фреймворком для створення серверних додатків з використанням JavaScript. Express може використовуватись разом з Apollo Server для створення серверної частини веб-застосунків. Даний фреймворк надає різноманітні інструменти для маршрутизації, обробки запитів, роботи зі статичними та динамічними маршрутами[5].

MongoDB – це база даних, яка зберігає дані у вигляді JSON–подібних[26] документів. Вона є популярним вибором для зберігання даних у сучасних веб–додатках[13].

Studio 3T – це клієнт для роботи з MongoDB, який надає графічний інтерфейс для керування базою даних MongoDB. Studio 3T використовується для взаємодії з БД, створення та виконання запитів, керування колекціями та іншими операціями з даними[21].

У даному проєкті Apollo Server використовується для обробки запитів GraphQL. Коли клієнт виконує запит до сервера, Apollo Server відповідає на запит, виконуючи логіку бізнес–процесів та взаємодіючи з базою даних MongoDB за необхідності. Express використовується як обгортка навколо Apollo Server, надаючи додаткові можливості для маршрутизації та обробки запитів.

Ці технології співпрацюють разом, надаючи потужні інструменти для розробки та розгортання веб застосунків. GraphQL використовується для опису структури API, Apollo Server для обробки запитів, Express для обгортки та маршрутизації, MongoDB для зберігання даних, а Studio 3T для роботи з БД.

1.4.2 Огляд використаних технологій для розробки клієнтської частини

Розробка клієнтської частини передбачає використання таких технологій, як React.js, axios та Apollo Client, для створення динамічного та інтерактивного інтерфейсу користувача.

React.js є відкритою JavaScript бібліотекою для розробки користувацьких інтерфейсів. Він дозволяє створювати складні інтерфейси з використовуючи компонентний підхід. React використовує віртуальний DOM для ефективного оновлення і відображення компонентів на стороні клієнта[17].

Axios є популярною бібліотекою для виконання HTTP–запитів[24] з браузера або з сервера з використанням JavaScript. Вона надає простий та зрозумілий інтерфейс для взаємодії з API та обміну даними. Axios підтримує різні методи запиту, такі як GET, POST, PUT, DELETE тощо, і дозволяє обробляти відповіді сервера та передавати дані між клієнтом та сервером[16].

Apollo Client є бібліотекою для управління станом та взаємодії з GraphQL–сервером. Він надає інструменти для виконання GraphQL–запитів, управління кешем, підписки на зміни даних та багато іншого[2]. Apollo Client інтегрується з React і дозволяє легко отримувати, оновлювати та відображати дані з GraphQL–сервера в компонентах React.

Axios може використовується в React–компонентах для виконання HTTP–запитів до сервера, наприклад, для отримання або оновлення даних з API. Інтегрований в React–додаток Apollo Client використовується для взаємодії з GraphQL–сервером та отримання даних з нього. Він використовує Axios для виконання фактичних HTTP–запитів до GraphQL–сервера. React компоненти використовують дані, отримані від Axios або Apollo Client, для відображення і оновлення відповідних даних в інтерфейсі. Ці технології використовуються разом для створення потужних та ефективних веб–додатків на основі JavaScript.

1.5 Формування структури веб–застосунку

Після проведення аналізу предметної області, можна зробити висновок, що для даного проєкту найбільше підійде структура яка базується на трирівневій архітектурі.

На рисунку 1.3 зображено структуру веб–застосунку.

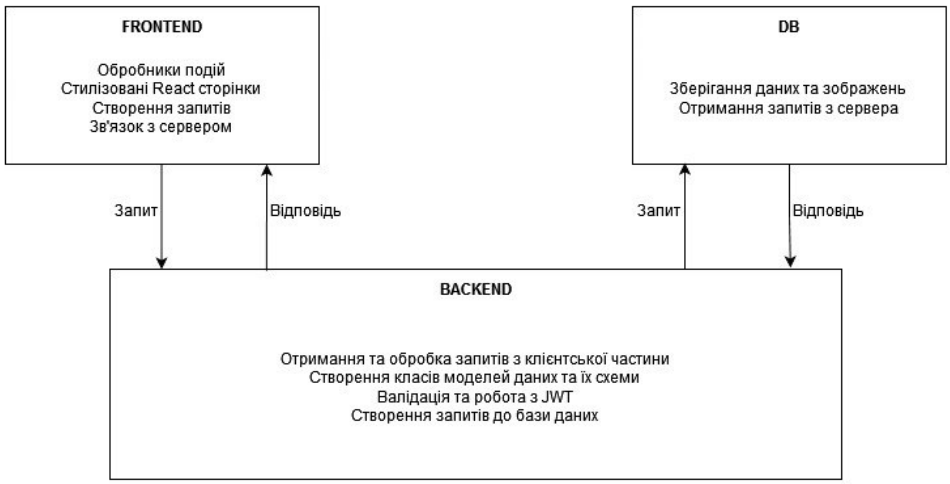

Рисунок 1.3 – Структура веб–застосунку

Структуру веб застосунку можна умовно поділити на три рівні:

1. Frontend – на цьому рівні реалізується інтерфейс, іменно з ним співпрацює користувач. Він відображає компоненти і сторінки React. А також створює запити до сервера[23].

2. Backend – структурно-логічний рівень який являє собою основне ядро веб–застосунку. Всі схеми, модулі, шифрування та обробка запитів реалізується на цьому рівні. Він взаємодіє як з клієнтською частиною, так і з базою даних[22].

3. Рівень даних – відповідає за збереження всіх даних які використовує сервер. До них відносяться дані користувача для авторизації і редагування профілю, а також в ньому зберігаються питання, відповіді і зображення.

1.6 Висновки до першого розділу

В першому розділі кваліфікаційної роботи було проаналізовано обрану предметну область і аргументовано її актуальність. Аналіз існуючих сервісів показав, що існує потреба у зручному та надійному додатку з актуальними матеріалами.

У цьому розділі були визначені основні компоненти веб-застосунку, такі як авторизація, реєстрація, тести з правил дорожнього руху, тести з білетами та статистика результатів. Для доступу до сайту користувач повинен зареєструватися, ідентифікувавшись за допомогою електронної адреси та пароля. Сторінка профілю містить особисті дані користувача. Тестування може бути як повним, так і з одним питанням, з детальним звітом про прогрес у проходженні білетів.

У процесі розробки веб-застосунку використовується середовище Visual Studio Code, а також технології, які дозволяють створювати ефективні веб-додатки. Node.js використовується для розробки серверної частини, використовуючи GraphQL, Apollo Server, Express, MongoDB і Studio 3T. Для клієнтської частини використовуються React.js, Axios і Apollo Client. Ці технології дозволяють створювати динамічні та інтерактивні веб-додатки з використанням JavaScript.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ–ЗАСТОСУНКУ ДЛЯ ПЕРЕВІРКИ ЗНАНЬ ПДР

2.1 Розробка серверної частини на базі Node.js

Після визначення усіх необхідних технологій для створення клієнтської та серверної частини, можна приступати до реалізації самого веб–застосунку. У даному пункті буде описано всі необхідні кроки та елементи для створення серверної частини сайту по перевірці знань ПДР користувача, за допомогою технологій Apollo Server, Express та MongoDB.

Express для швидкого створення маршрутизації та обробки запитів. MongoDB виступатиме у ролі нереляційної бази даних, що ідеально підходить для розробки на Node.js, а Apollo Server визначає схему даних, обробляє запити користувачів тощо.

2.1.1 Ініціалізація проєкту

Для створення проєкту потрібно відкрити теку в якій буде знаходитися сам проєкт, також потрібно відкрити її у терміналі. В командній строці прописуємо команду 'npm init', яка ініціалізує проект. Після виконання команди, у теці буде створено файл 'package.json'. Цей файл буде містити інформацію про проект та його залежності.

Наступним кроком є встановлення необхідних залежностей В даному випадку це будуть Apollo Server, Express і MongoDB.

На рисунку 2.1 зображено процес встановлення залежностей.

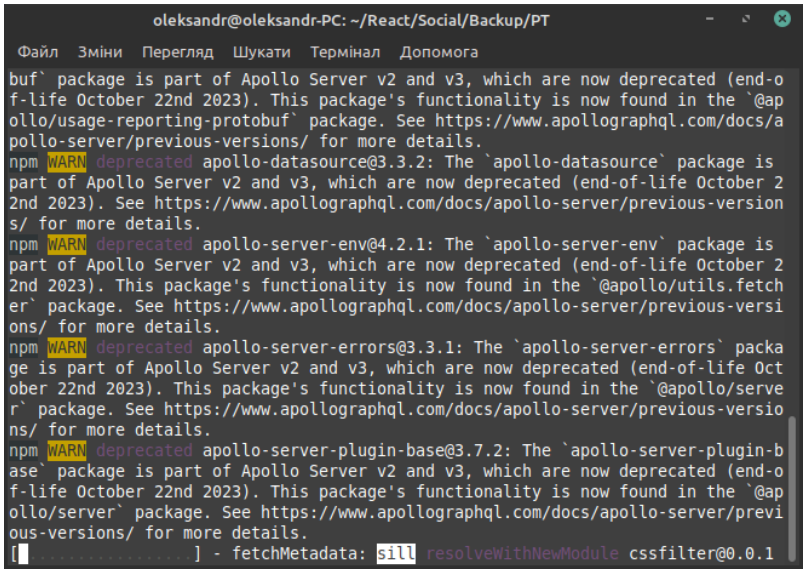

Рисунок 2.1 – Процес встановлення залежностей

Для встановлення залежностей виконується команда в командному рядку – npm install apollo–server–express mongoose.

2.1.2 Налаштування сервера

Створюється файл 'startup.js' в кореневій папці проєкту. В файлі підключаються необхідні модулі і налаштовується базовий сервер Express. Наступним кроком є використання схем GraphQL в яких описані типи та запити в схемі, а також резольвери і мутації. Створюється Apollo Server який потім передається до Express. Прописується порт на якому буде запущено сервер, і після цього від готовий до запуску.

На рисунку 2.2 зображено повідомлення в консолі яке свідчить про успішний запуск сервера.

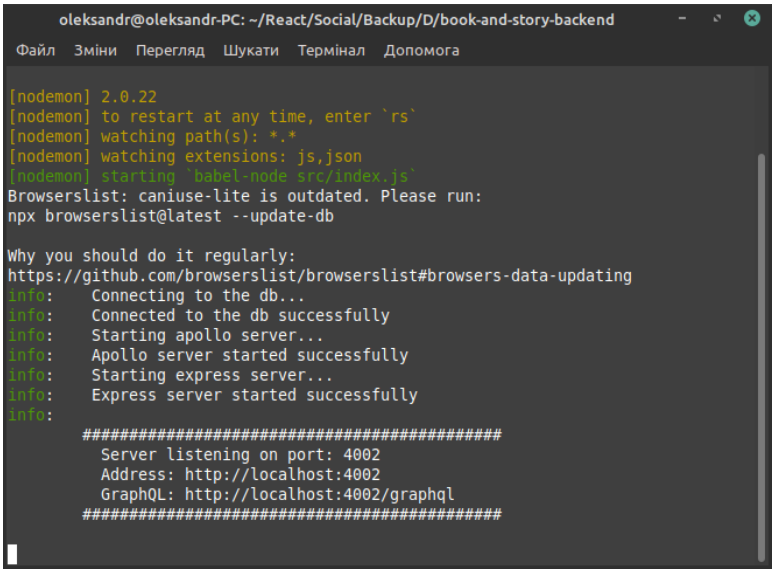

Рисунок 2.2 – Повідомлення про успішний запуск сервера.

Після успішного запуску сервера, можна приступати до написання основної логіки.

2.1.3 Підключення до MongoDB

Для повноцінної роботи сервера йому необхідна співпраця з базою даних. В даному випадку використовується MongoDB[13]. Першим етапом сервер підключається до бази даних, підключення відбувається в файлі 'startup.js' і додаються обробники помилок. Коли БД підключено, можна починати створювати. моделі даних в яких буде описано структуру та взаємозв'язки між даними. В схемах GraphQL описано структура та типи даних які будуть зберігатися в БД, а також резольвери й мутації. Резольвери відповідають за запити до БД, тоді як мутації відповідають за оновлення, додавання, видалення даних. Для випадків коли адміністратору сервера вручну змінити дані в MongoDB, використовується спеціальний інтерфейс – Studio 3T.

На рисунку 2.3 зображено редагування даних в середовищі Studio 3T.

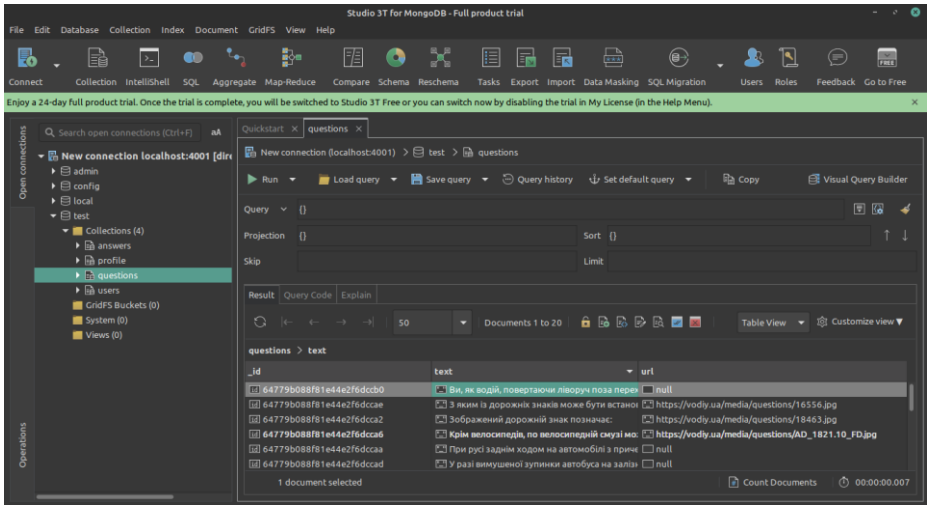

Рисунок 2.3 – Редагування даних в середовищі Studio 3T

Studio 3T дозволяє переглядати, редагувати та створювати дані і таблиці в MongoDB

2.1.5 Структура файлів і каталогів

Для розробки серверної частини була створена наступна структура файлів і каталогів. Кожен елемент цієї структури відповідає певному функціоналу та відіграє важливу роль у роботі проекту. Далі буде наведено короткий опис кожного елемента:

– Каталог «rutes»

– auth.js – Модуль який встановлює маршрути для обробки HTTP–запитів, пов'язаних з аутентифікацією та авторизацією.

– test.js – Модуль який містить обробник помилок, визначення маршрутів і контролерів для обробки HTTP–запитів пов'язаних з питаннями на сервері.

– Каталог «schemas»

– auth.js – Схема GraphQL для автентифікації та керування користувачами. Вона включає типи запитів, резольвери і мутації, такі як вхід, вихід, реєстрація та оновлення токенів.

– profile.js – Схема GraphQL для оновлення профілю користувача. Основне завдання схеми це оновлення і отримання профілю користувача.

– test.js – Даний файл є схемою GraphQL, він описує типи запитів, резольвери і мутації, основне його завдання полягає в отриманні питань та відповідей.

– Каталог «services»

– AuthService.js – Даний файл є класом AuthService, який відповідає за автентифікацію, генерацію та перевірку JWT–токенів[9].

– ProfileService.js – У цьому файлі реалізовано клас ProfileService, який містить методи для: створення профілю, оновлення профілю і отримання профілю за ідентифікатором користувача.

– TestService.js – В даному файлі знаходиться клас TestService, який надає методи для взаємодії з колекціями документів questions і answers в базі даних MongoDB. Він може повертати як цілі колекції questions чи answers, так і знаходити question чи answer за ідентифікатором і повертає його.

– UserService.js – В даному файлі реалізовано клас UserService, який надає різний функціонал для роботи з користувачем. Він містить методи для збереження, отримання та авторизації даних пов'язаних з користувачем.

– Каталог «utils»

– logger.js – В цьому файлі відбувається налаштування логування. Тут визначається налаштування логування в залежності від середовища виконання. Він також включає налаштування, щоб не виходити з програми при помилці.

– startup.js – Це файл який запускає сервер, в ньому реалізовано налаштування сервера і підключення MongoDB.

– env – Даний файл включає в себе різні конфігураційні параметри.

В свою чергу файл env виконує відповідні завдання:

– встановлює URL–рядок[11] підключення до локальної бази даних MongoDB, яка працює на порту 4001;

– встановлює рівень журналювання логів на рівень 1;

– встановлює режим розробки;

– встановлює секретні ключі для підписування та перевірки доступного і оновлюючого токену JWT;

– встановлює порт 4002 для запуску сервера.

– В даний список не було включено каталог «node_modules» і інші файли які були ініціалізовані при створені проєкту.

2.2 Розробка клієнтської частини на базі React.js

Після створення серверної частини проєкту на базі Node.js, можна приступати до роботи над реалізації клієнтської частини веб–застосунку за допомогою популярних бібліотек React.js та Axios, а також потурбуватися про злагоджену взаємодію між серверною та клієнтською частиною проєкту.

2.2.1 Ініціалізація проєкту

Щоб створити новий React.js проєкт, потрібно відкрити папку, в якій у майбутньому буде знаходитися сам проект, у терміналі і прописати команду «npx create–react–app». Після виконання команди, у теці буде ініціалізовано всі початкові файли для створення React проєкту.

На рисунку 2.4 зображено процес ініціалізації React проєкту.

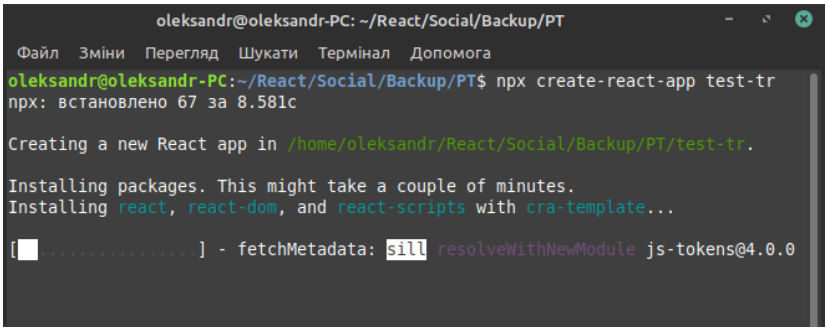

Рисунок 2.4 – Ініціалізація React проєкту

Для даного проекту потрібно ще встановити багато сторонніх бібліотек, які будуть допомагати у побудові клієнтської частини веб–сайту, для приклду

на рисунку 2.5 можна побачити процес інсталювання однієї із найголовніших бібліотек для коректної роботи проєкту. Для встановлення сторонніх бібліотек, в даному випадку, використовується команда 'npm i library–name', де library–name – це назва бібліотеки.

На рисунку 2.5 зображено процес інсталювання бібліотеки «Axios».

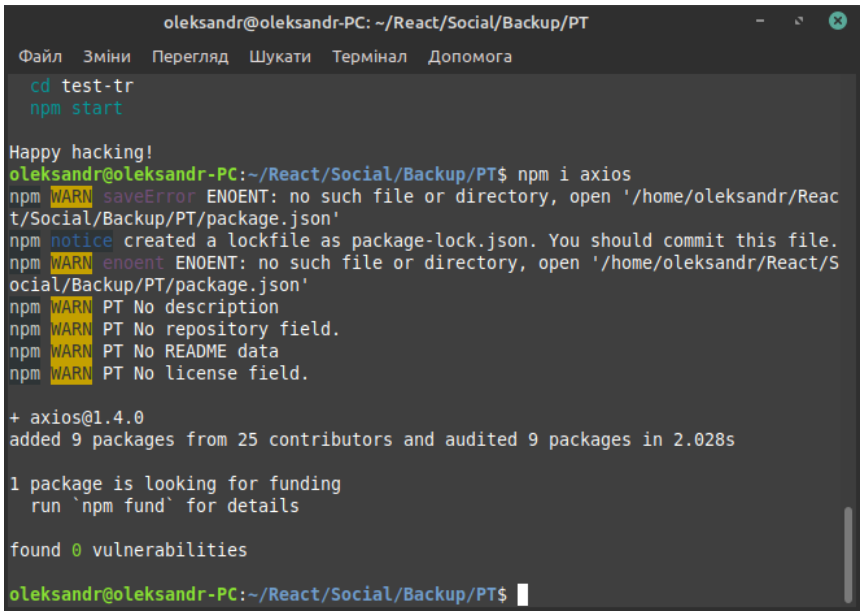

Рисунок 2.5 – Процес інсталяції бібліотеки «Axios».

Після перевірки коректності роботи бібліотек і виправленні помилок що виникли у процесі, можна приступати до вирішення задачі їхньої взаємодії із серверною частиною застосунка.

2.2.2 Взаємодія з сервером

Так як немає єдиного правильного порядку створення клієнтської частини, було вирішено почати з налаштування запитів до серверної частини. Для цього створюються окремі файли для сервісів, які будуть відповідальні за взаємодію з сервером. В них буде описано функції які будуть виконувати різні запити до сервера, такі як GET, POST, PUT, DELETE. Для здійснення запитів використовується бібліотека Axios. В сервісах будуть налаштовані базові URL,

за допомогою яких можна буде звертатися до сервера з клієнтської частини. Для перевірки коректності отриманих даних, в коді використовуємо команду 'console.log(data)', де 'data' є даними які були отримані з сервера. Після запуску проекту, в консолі браузера будуть виводитися отримані дані з сервера.

2.2.3 Розробка компонентів і сторінок

Наступним етапом розробки є створення сторінок які вже можна буде бачити в браузері. Для початку потрібно створити сторінку яка буде відмальовувати інші сторінки і компоненти. Створюється файл 'App.js' в якому оголошується функція 'App()', вона буде експортуватися по замовчуванню, дана функція буде відмальовувати компоненти 'RootRouter' і 'ToastContainer'.

На рисунку 2.6 зображено головну сторінку сайту

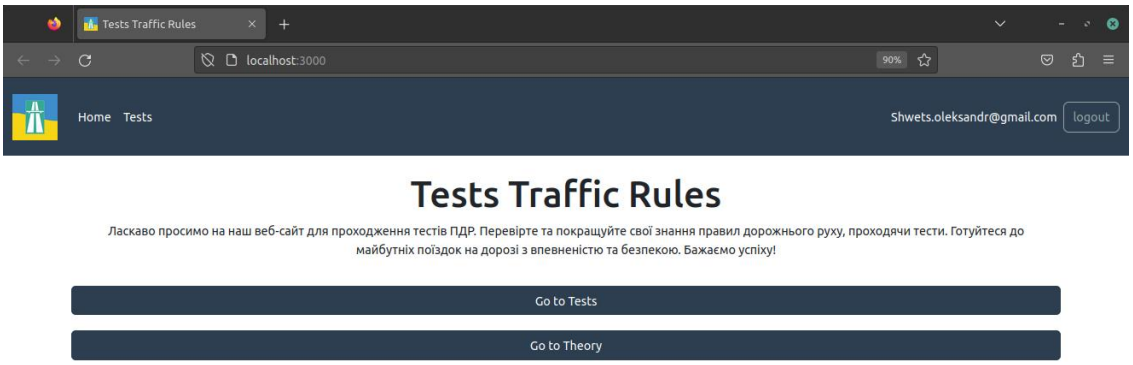

Рисунок 2.6 – Головна сторінка.

Компонента 'ToastContainer' імпортується з бібліотеки 'react–toastify', і дозволяє легко додавати спливаючі сповіщення. Компонента 'RootRouter' отримує дані з сервера і перевіряє чи є користувач авторизованим, також дана компонента перенаправляє користувача на інші сторінки в залежності від його

дій. Якщо користувач авторизований, він має доступ до всіх інших сторінок веб застосунку, проте якщо користувач не авторизований, його перенаправить на сторінку авторизації. Для взаємодії з сервером створено хуки, які будуть викликати методи класів що знаходяться в серверній частині проєкту. Таким чином за допомогою хуків можна надсилати, і отримувати потрібні дані з сервера, також можна оновлювати або змінювати дані.

2.2.4 Оформлення та стилізація

Для оформлення елементів і стилізації сайту було обрано бібліотеку 'react–bootstrap', вона дозволяє одночасно стилізувати всі елементи які обгорнуті спеціальними тегами, які пропонує сама бібліотека. Завдяки такому підходу можна за лічені хвилини кардинально змінити дизайн сайту. Для встановлення того чи іншого дизайну, можна скачати файл CSS[12], і підключити до свого проєкту[18].

2.2.5 Структура файлів і каталогів

Для розробки клієнтської частини була створена велика структура файлів і каталогів. Кожен елемент цієї структури відповідає певному функціоналу та відіграє важливу роль у роботі проекту. Нижче наведено короткий опис важливих елементів:

– Каталог 'components' – містить в собі компоненти які сторінки відмальовують на веб сторінці. Далі наведено компоненти які знаходяться в каталозі:

– AuthForm.js – Виводить форму для реєстрації нових користувачів.

– ButtonWithSpinner.js – Показує елемент загрузки якщо інша компонента ще не відмалювалася.

– AppLayout.js – Відмальовує шапку сайту якщо користувач авторизований.

– AuthLayout.js – Відмальовує шапку сайту якщо користувач неавторизований.

– OneTestForm.js – Відмальовує Одне питання і варіанти відповіді.

– Profile.js – Відмальовує дані користувача на сторінці профілю.

– ProfileForm.js – Відмальовує поля для редагування профілю.

– AnswerAnalysis.js – Будує кругову діаграму де зображено кількість правильних і неправильних відповідей, а також кількість питань на які не відповіли.

– RandomTicketForm.js Відмальовує 20 рандомних питань разом з варіантами відповідей.

– apolloClient.js – Встановлює з'єднання між клієнтом і сервером, різні Apollo – посилання для налаштування автентифікації і оновлення токенів доступу.

– Каталог 'hooks' – Містить в собі хуки які співпрацюють з сервером, а саме: приймають, надсилають, оновлюють, змінюють дані на сервері.

– Каталог 'pages' – Даний каталог містить в собі сторінки на які може переходити користувач. Нижче наведено список сторінок які знаходяться в даному каталозі:

– EditProfile.js – Показує сторінку на якій відбувається редагування профілю, також в даному файлі викликається хук для взаємодії з сервером.

– Home.js – Головна сторінка на яку користувач попадає зразу після авторизації, чи реєстрації, містить в собі вітальний текст і 2 кнопки, одна перенаправляє користувача на сторінку 'Tests', а друга перенаправляє на сторонній сайт із правилами дорожнього руху.

– Login.js – Сторінка логанізації.

– MyProfile.js – Сторінка профілю.

– OneTest.js – Сторінка яка відображає компоненту, в яку передає одне питання з відповідями, містить 2 кнопки. Перша кнопка відповідає за зміну кольору відповідей в залежності від відповіді. Друга заново перемальовує компоненту з іншим питанням.

– RandomTicket.js – Сторінка яка малює компоненту з 20 запитаннями. Також містить кнопку, після натискання якої, на полях з відповідями показується правильна відповідь і малюється кругова діаграма з аналізом відповідей.

– SingUp.js – Сторінка реєстрації.

– Tests.js – Сторінка яка відмальовує сторінки 'RandomTicket.js' та 'OneTest.js', а також 2 кнопки які на них перенаправляють.

– Каталог 'router' – Містить файли для керування перенаправленнями.

– path.js – Зберігає в собі об’єкт, елементи якого містять в собі частини URL адрес.

– RootRouter.js – Перевіряє чи користувач авторизований, якщо так то перенаправляє на головну сторінку, якщо ні то на сторінку логанізації. Також даний файл відповідає за переадресацію на сторінки в залежності від зміни URL.

– App.js – Файл який відмальовує всі сторінки і компоненти. Також підключає стилізацію сайту.

2.3 Опис типової схеми використання веб застосунку

Коли користувач вперше потрапить на сайт, йому буде запропоновано пройти авторизацію. Так як це перше відвідування, користувачу потрібно зареєструватися щоб на сервері збереглися його дані. Після реєстрації користувач може авторизуватися за допомогою електронної пошти і паролю.

Далі у користувача буде можливість редагувати його особистий акаунт на сайт, встановити рисунок, змінити пошту та пароль, а також за бажанням ввести ім’я та прізвище користувача (див. Додаток Ґ). Також для перевірки знань можна буде пройти як одиночний тест так і тест екзаменаційного формату із випадковими питаннями.

На рисунку 2.7 зображено сторінку реєстрації користувача.

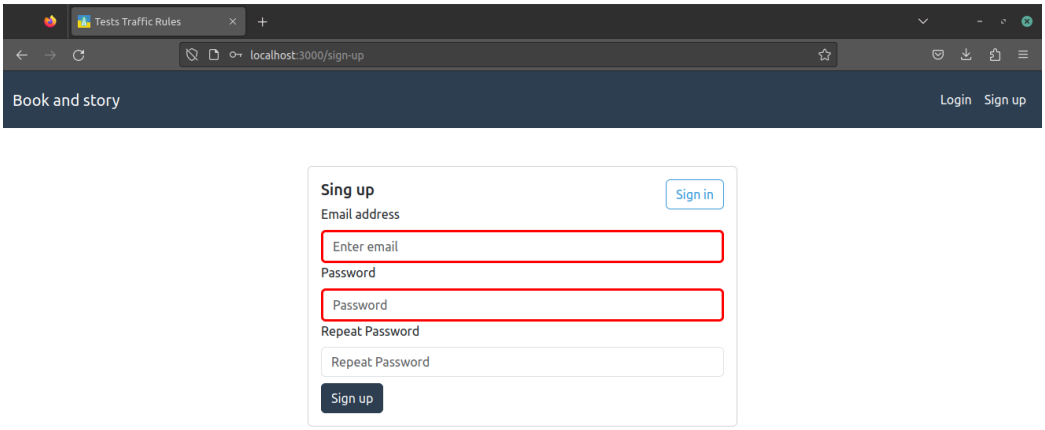

Рисунок 2.7 – Сторінка реєстрації.

Після того як користувача було авторизовано, його перенаправляє на головну сторінку, з якої він може зайти на сторінку 'Tests' і натиснути кнопку 'RandomTicket'. Після чого на екрані появляться питання.

На рисунку 2.8 зображено сторінку з випадковим білетом

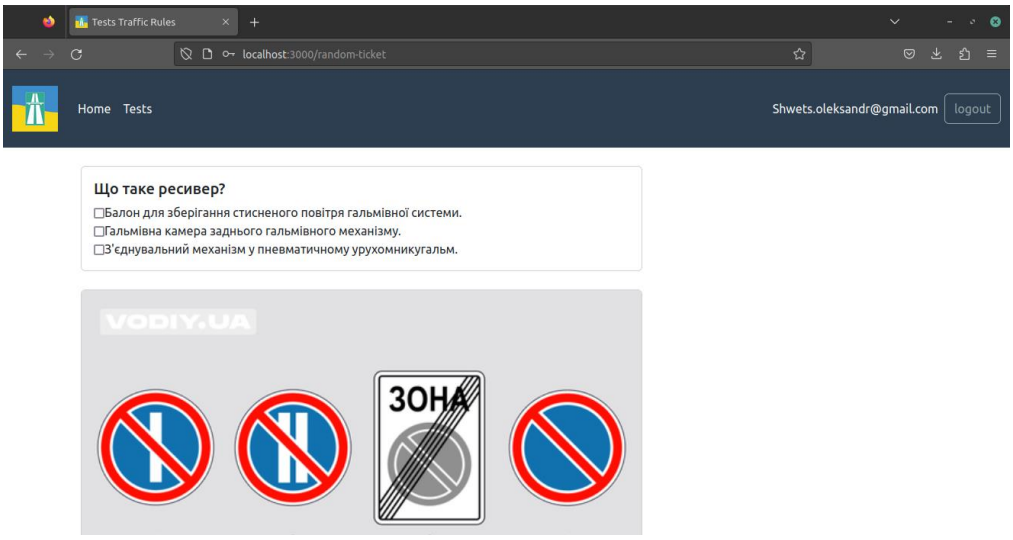

Рисунок 2.8 – Сторінка з випадковим білетом

Коли користувач пройде тест і натисне кнопку 'Check Answers', його перемістить на верх сторінки, правильні питання буде перемальовано зеленим кольором, а неправильні червоним, також над питаннями появиться кругова діаграма зі статистикою відповідей.

На рисунку 2.9 зображено сторінку з пройденим випадковим білетом і діаграмою.

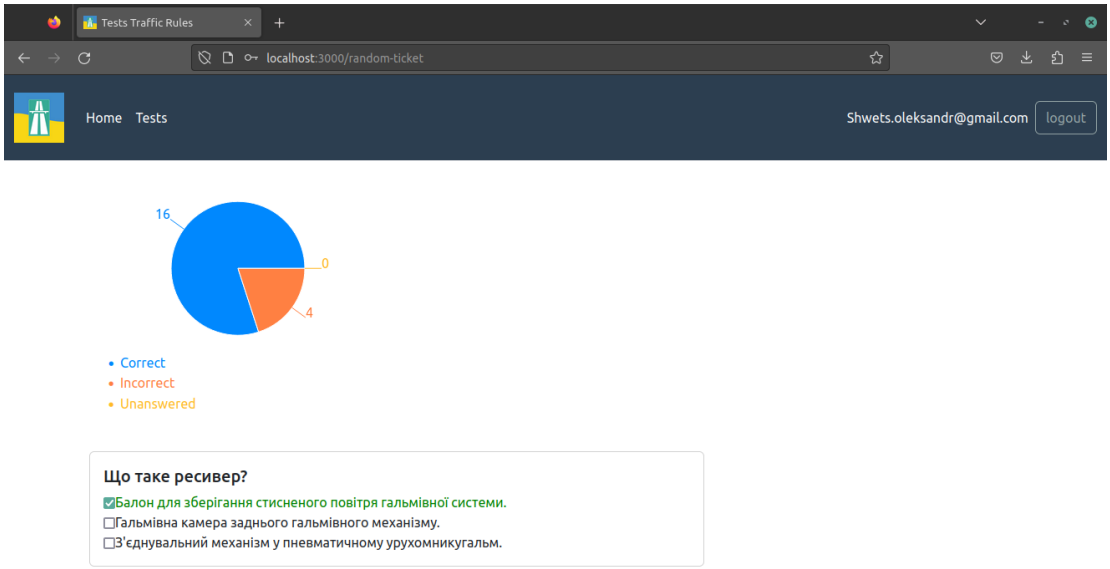

Рисунок 2.9 – Сторінка пройденого випадкового білету з діаграмою.

У результаті розроблено повністю робочий веб застосунок для перевірки знань з правил дорожнього руху, з можливістю редагування профілю, декількох варіантів способів тестувань та аналізу відповідей учня.

2.4 Висновки до другого розділу

У даному розділі описано розробку серверної та клієнтської частини веб–проєкту. Для серверної частини використовується Node.js, з допомогою якого створюються і налаштовуються необхідні модулі, з'єднання з базою даних MongoDB та реалізується схема GraphQL. Серверна частина має структуру файлів і каталогів, яка включає файли для маршрутизації, схем, сервісів та утиліт.

У клієнтській частині використовується React.js, де процес розробки починається з ініціалізації проєкту та встановлення необхідних бібліотек. Далі відбувається взаємодія з серверною частиною за допомогою Axios та створення компонентів і сторінок. Оформлення і стилізація елементів веб–сторінки

здійснюється за допомогою бібліотеки react–bootstrap. Клієнтська частина також має свою структуру файлів і каталогів, включаючи файли для компонентів, хуків та інших важливих елементів.

У результаті розробки серверної та клієнтської частини забезпечується функціональність веб–проєкту, включаючи з'єднання з базою даних, взаємодію з сервером, створення сторінок та компонентів, а також оформлення та стилізацію веб–інтерфейсу.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Безпека при надзвичайній ситуації в будівлі сервісного центру МВС

Перш ніж приступати до розгляду правил безпеки при надзвичайних ситуаціях в сервісному центрі МВС варто зазначити, що дана будівля відноситься до приміщень підвищеної небезпеки, особливо у час воєнного стану в країні. Тому дані правила необхідно знати як і працівникам даних служб так і цивільним громадянам які з тих чи інших причин перебувають у даній будівлі, або просто проходять поряд[4].

Надзвичайна ситуація - порушення нормальних умов життя та діяльності людей на окремій території чи об'єкті на ній або на водному об'єкті, спричинене аварією, катастрофою, стихійним лихом чи іншою небезпечною подією, зокрема епідемією, епізоотією, епіфітотією, пожежею, що призвело до виникнення великої кількості постраждалих, загрози життю та здоров'ю людей, їх загибелі, значних матеріальних утрат, а також до неможливості проживання населення на території чи об'єкті, ведення там господарської діяльності.

Подальші дії людей залежать від характеру надзвичайної ситуації, це може бути як надзвичайна ситуація техногенного характеру (пожежі, вибухи, аварії тощо), так і природнього (повені, землетруси, виверження вулкану), соціального (протиправні терористичні напади) та воєнного (військові конфлікти та війни). У даній ситуації розглядитиметься надзвичайна ситуація воєнного характеру, адже нажаль це є сьогоденням сучасних українців. Отже, дії людей при даній ситуації у будівлі сервісного центру МВС наступні:

1) Від лиця цивільної людини яку надзвичайна ситуація застала будівлі сервісного центру МВС, слід уважно слухати оголошення та настанови персоналу.

2) Зберігати спокій та уникати паніку наскільки це можливо, адже паніка породжує хаос і штовханину що може тільки погіршити становище.

3) Робітники МВС та відвідувачі повинні негайно закрити штори або жалюзі у вікнах.

4) Вимкнути світло у приміщенні та закрити вікна та двері. Це допоможе зменшити видимість та наскільки це можливо збільшити рівень безпеки приміщення.

5) Скористуватися правилом двох стін, яке свідчить що для збільшення безпеки людей, від небезпеки їм має розділяти щонайменше 2 стіни, бо одна, ймовірно, зруйнується від удару, а друга візьме на себе уламки стіни, віконного скла тощо це може бути комора, ванна чи передпокій, у даному випадку вбиральня або який кабінет в глибині приміщення без вікон та скляних поверхонь. Це забезпечить більшу захищеність від можливих пострілів.

6) Негайно проінформувати про небезпеку близьких і знайомих. Використати будь–які доступні засоби сповіщення, такі як мобільні телефони або системи попередження.

7) Якщо це дозволяє ситуація варто негайно покинути будівлю та знайдіть безпечний притулок, наприклад, підвал або найближче укриття.

Як найшвидше відійти до захисної споруди або знайти безпечне місце на місцевості. Дотримуватись спокою, не залишати безпечне місце перебування без крайньої потреби. Та слідкувати за офіційними повідомленнями для отримання актуальної інформації.

3.2 Протипожежна безпека в сервісному центру МВС

Протипожежна безпека є невід'ємною складовою організаційної роботи на об'єктах господарювання, включаючи будівлі сервісного центру МВС. Забезпечення безпеки в разі пожежі вимагає відповідної підготовки, розробки та впровадження протипожежних заходів та контролю дотримання вимог і норм законодавства.

На будівлі МВС накладається відповідальність за створення безпечних умов праці та мінімізацію ризику виникнення пожеж. Це означає, що будівля

має відповідати протипожежним вимогам та нормам законодавства. Посадові особи, згідно з рішенням керівництва, зобов'язані розробити та впровадити протипожежний режим і інструкції, які охоплюють конкретні території, ділянки, зони, об'єкти та їх частини у будівлі сервісного центру МВС[30].

Протипожежний режим і інструкції містять встановлені порядки та правила користування місцями спеціального призначення, такими як евакуаційні шляхи, "курилки", місця складування продукції та сировини, стоянки транспорту. Крім того, регламентуються порядок роботи та технічне обслуговування вентиляційного устаткування, засобів пожежогасіння і захисту від загорянь, нагрівальних приладів, електрообладнання[29].

На будівлі МВС передбачається розробка і впровадження порядку дій при виникненні пожежі, а також забезпечення належного контролю дотримання протипожежних вимог і норм законодавства. Важливою складовою протипожежного режиму є наявність плану евакуації та опису процедур відключення електроустановок та інших важливих вимкнень[28].

Керівник установи повинен визначити обов'язки посадових осіб щодо забезпечення пожежної безпеки, призначити відповідальних за пожежну безпеку окремих будівель, споруд, приміщень, дільниць тощо, технологічного та інженерного устаткування, а також за утримання і експлуатацію технічних засобів протипожежного захисту.

Керівник зобов'язаний вживати відповідних заходів реагування на факти порушень чи невиконання посадовими особами, іншими працівниками підприємства встановленого протипожежного режиму, вимог правил пожежної безпеки та інших нормативно-правових актів, що діють у цій сфері.

Для кожного об'єкта, приміщення та виду робіт у будівлі МВС розробляються інструкції, які визначають порядок роботи персоналу та виконання окремих видів робіт. Ці інструкції включають навчання персоналу з питань пожежної безпеки, проведення інструктажів та перевірку знань пожежно-технічного мінімуму. Навчання може проводитись як на внутрішніх

лекціях, семінарах та тренінгах на підприємстві, так і на зовнішніх навчальних центрах з використанням кваліфікованих викладачів.

Для підвищення рівня протипожежної безпеки на будівлі сервісного центру МВС може бути створений підрозділ добровільної пожежної охорони та пожежно–рятувальна команда. Ці підрозділи відповідають за виконання спеціальних завдань у разі виникнення пожежі, евакуації працівників та порятунку майна.

Відповідальність за дотримання вимог пожежної безпеки покладається на кожного співробітника підприємства. Організаційна складова виконання цих вимог лежить на посадових особах, згідно з рішеннями керівництва та відповідними посадовими інструкціями та положеннями структурних підрозділів.

На кожному підприємстві має бути опрацьована загальнооб'єктова інструкція про заходи пожежної безпеки та інструкції для всіх вибухопожежонебезпечних та пожежонебезпечних приміщень.

Ці інструкції мають вивчатися під час проведення протипожежних інструктажів, проходження пожежно-технічного мінімуму, а також в системі виробничого навчання і вивішуватися на видних місцях.

На території підприємства на видних місцях повинні бути встановлені таблички із зазначенням порядку виклику пожежної охорони, знаки місць розміщення первинних засобів пожежогасіння, схема руху транспорту, в якій слід вказувати розміщення будівель, водойм, гідрантів, пірсів та градирень

Протипожежна безпека є важливим аспектом функціонування будівлі сервісного центру МВС. Запровадження протипожежних заходів, контроль їх дотримання та належне навчання персоналу є необхідними для забезпечення безпеки працівників та майна на об'єкті. Розробка режиму, інструкцій та плану евакуації сприяє ефективному управлінню пожежною безпекою та підвищенню рівня готовності до дій у надзвичайних ситуаціях.

3.3 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи описано рекомендації щодо заходів безпеки у випадку надзвичайних ситуацій в будівлі сервісного центру МВС, як діяти виникненні загрози ураження стрілецькою зброєю, бойовими діями або повітряної небезпеки. В розділі міститься інформація про план дій при виникненні надзвичайної ситуації, та про потребу забезпечення основних матеріальних потреб.

Також описано протипожежну безпеку та необхідні заходи для забезпечення безпеки в будівлі сервісного центру МВС. Наведено вимоги щодо створення безпечних умов праці та мінімізації ризику пожежі, розробку протипожежних заходів і контроль їх дотримання, а також розробку протипожежного режиму, інструкцій та плану евакуації.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи, було спроектовано та реалізовано веб застосунок для перевірки знань ПДР. При аналізу та огляді переваг та недоліків потенційних конкурентів, було доведено актуальність проекту. В якості середовища для розробки було обрано Visual Studio Code з використанням платформи Node.js.

Для розробки серверної частини були використані GraphQL, Apollo Server, Express, MongoDB і Studio 3T, а для клієнтської частини - React.js, axios і Apollo Client.

– Перший розділ роботи присвячений:
– Проведенню аналізу предметної області.
– Огляду існуючих рішень.
– Огляді переваг та недоліків потенційних конкурентів.
– Вибору середовища розробки.
– Огляду використаних технологій як для серверної, так і для клієнтської частини.
– Структурі веб застосунку.

Другий розділ роботи включає:

– Розробку серверної частини на базі Node.js.
– Ррозробку клієнтської частини на базі React.js.
– Опис типової схеми використання веб застосунку.

Були розглянуті основні аспекти розробки серверної та клієнтської частин, використання відповідних технологій та аргументовано вибір підходів до реалізації.

У розділі «Безпека життєдіяльності, основи охорони праці» описано протипожежна безпека в будівлях МВС, а саме забезпечення безпеки в разі пожежі, відповідність будівлі нормам протипожежної безпеки та ознайомленість працівників з інструкціями щодо пожежної безпеки. Також описано план дій у разі надзвичайно ситуації в залежності від її контексту.

ПЕРЕЛІК ДЖЕРЕЛ

1. Apollo Docs – Get started with Apollo Server. [Електронний ресурс] – Режим доступу до ресурсу: https://www.apollographql.com/docs/apollo–server/

2. Apollo Docs – Introduction to Apollo Client. [Електронний ресурс] – Режим доступу до ресурсу: https://www.apollographql.com/docs/react/

3. Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack / E. Brown. – 2nd edit. – Cambridge: O`Reilly Media, 2018. – 243 Пасічник О. Г. Основи веб-дизайну [Текст]: навч. посібн. / О. Г. Пасічник, О.

4. EllasCookies – МВС - це одна з найважливіших силових структур країни. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.ellas-cookies.com/zakon/112463-mvd-eto-odna-iz-vazhneyshih-silovyh-struktur-strany.html

5. Express - Node.js web application framework. [Електронний ресурс] – Режим доступу до ресурсу: https://expressjs.com/en/5x/api.html

6. GraphQL – Introduction to GraphQL. [Електронний ресурс] – Режим доступу до ресурсу: https://graphql.org/learn/

7. IBM – What is an Application Programming Interface (API)? [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/topics/api

8. JetBrains – What’s New in WebStorm 2023.1. [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/help/webstorm/meet–webstorm.html

9. JWT.IO – JSON Web Tokens Introduction. [Електронний ресурс] – Режим доступу до ресурсу: https://jwt.io/introduction/

10. MDN Web Docs – JavaScript. [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript

11. MDN Web Docs – What is a URL? [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_URL

12. MDN Web Docs – What is CSS? [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS

13. MongoDB: The Developer Data Platform – Itroduction to MongoDB. [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/docs/manual/introduction/

14. Node JS – Dependencies. [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en/docs/meta/topics/dependencies

15. NPM – About npm. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.npmjs.com/about-npm

16. Npmjs – Axios. [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/package/axios

17. React A JavaScript library – Getting Started. [Електронний ресурс] – Режим доступу до ресурсу: https://legacy.reactjs.org/docs/getting–started.html

18. React Bootstrap – React Bootstrap. [Електронний ресурс] – Режим доступу до ресурсу: https://react-bootstrap.github.io/

19. Simpson K. You Don`t Know JS: Up & Going / K. Simpson. – 1st edit. – USA: O`Reilly Media, 2020. – 88 p.

20. StackPath – What is V8 JavaScript Engine? [Електронний ресурс] – Режим доступу до ресурсу: https://www.stackpath.com/edge-academy/what-is-v8-javascript-engine/

21. Studio 3T – Studio 3T Licensing Archives. [Електронний ресурс] – Режим доступу до ресурсу: https://studio3t.com/knowledge–base/articles/time–series–collections–in–studio–3t/

22. TechTerms.com – Backend. [Електронний ресурс] – Режим доступу до ресурсу: https://techterms.com/definition/backend

23. TechTerms.com – Frontend. [Електронний ресурс] – Режим доступу до ресурсу: https://techterms.com/definition/frontend

24. TechTerms.com – HTTP. [Електронний ресурс] – Режим доступу до ресурсу: https://techterms.com/definition/http

25. Visual Studio Code – Documentation for Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs

26. W3Schools – What is JSON? [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/whatis/whatis_json.asp

27. В. Пасічник, І. В. Стеценко. – К. : Вид. група BHV, 2009 – 336 с. р.

28. Закон України – «Про охорону праці» [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

29. Маньківська райдержадміністрація – Правила безпечної поведінки в надзвичайних ситуаціях. . [Електронний ресурс] – Режим доступу до ресурсу: http://mankrda.gov.ua/pravila-bezpechnoi-povedinki-v-nadzvichajnih-situaciyah/

30. Офіційний вебпортал парламенту України – Про затвердження Правил пожежної безпеки в Україні. [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0252-15#Text

ДОДАТКИ

Додаток А

Програмний код JS для запуску сервера «startup.js»

```
import express from 'express';
import RootLoader from './loaders';
import logger from './utils/logger';

export default async function startServer() {
  try {
    const app = express();
    const port = process.env.PORT;

    await RootLoader(app);

    app.listen(port, () => {
      logger.info(`
        ################################################
          Server listening on port: ${port}
          Address: http://localhost:${port}
          GraphQL: http://localhost:${port}/graphql
        ################################################
      `);
    });
  } catch (e) {
    console.log(e.messages);
    process.exit(1);
  }
}
```

Додаток Б

Програмний JS код схеми GraphQL для автентифікації та керування користувачами «auth.js»

```
import { gql } from 'apollo-server-express';
import UserService from '../../services/UserService';

export const typeDefs = gql`
  extend type Query {
    me: User
  }

  type AuthResponse {
    accessToken: String!
    refreshToken: String
  }

  type User {
    _id: ID!
    email: String!
  }

  extend type Mutation {
    login(email: String!, password: String!): AuthResponse!
    logout(token: String!): Boolean
    registration(email: String!, password: String!): AuthResponse!
    token(refreshToken: String!): AuthResponse!
  }
`;

export const resolvers = {
  Query: {
    me: async (root, params, context) => {
      const user = await context.getUser();
      return user;
    },
  },

  Mutation: {
    login: async (root, { email, password }) => {
      const { accessToken, refreshToken } = await
UserService.loginWithPassword({ email, password });
      return { accessToken, refreshToken };
    },
    logout: (root, { token }) => {
      UserService.logout(token);
      return true;
    },
    registration: async (root, { email, password }) => {
      await UserService.createAccount({ email, password });
```

```
      const { accessToken, refreshToken } = await
UserService.loginWithPassword({ email, password });
      return { accessToken, refreshToken };
    },
    token: async (root, { token }) => {
      const accessToken =
UserService.loginWithRefreshToken(token);
      return { accessToken };
    },
  },
};
```

Додаток В

Програмний JS код для перенаправлення користувача, та перевірки на авторизацію «RootRouter»

```
import React, { useEffect, useState, useCallback } from
'react';
import { Route, BrowserRouter as Router, Switch, Redirect }
from 'react-router-dom';
import AppLayout from '../components/layouts/AppLayout';
import AuthLayout from '../components/layouts/AuthLayout';
import Home from '../pages/Home';
import Login from '../pages/Login';
import SignUp from '../pages/SignUp';
import AuthManager from '../services/AuthManager';
import paths from './paths';
import MyProfile from '../pages/MyProfile';
import EditProfile from '../pages/EditProfile';
import UserProfile from '../pages/UserProfile';
import Tests from '../pages/Tests';
import OneTest from '../pages/OneTest'
import RandomTicket from '../pages/RandomTicket';


const authRouts = [
  {
    path: paths.login,
    exact: true,
    Component: Login,
  },
  {
    path: paths.signUp,
    exact: true,
    Component: SignUp,
  },
];

const appRouts = [
  {
    path: paths.home,
    exact: true,
    Component: Home,
  },
  {
    path: paths.myProfile,
    exact: true,
    Component: MyProfile,
  },
  {
    path: paths.editProfile,
    exact: true,
    Component: EditProfile,
  },
```

```
  {
    path: paths.userProfile,
    exact: true,
    Component: UserProfile,
  },
  {
    path: paths.Tests,
    exact: true,
    Component: Tests,
  },
  {
    path: paths.oneTest,
    exact: true,
    Component: OneTest,
  },
  {
    path: paths.randomTicket,
    exact: true,
    Component: RandomTicket,
  },
];

const useIsLoggedIn = () => {
  const [loggedIn, setLoggedIn] =
useState(AuthManager.isLoggedIn());

  const subscriber = useCallback((token) => {
    setLoggedIn(!!token);
  }, []);

  useEffect(() => {
    AuthManager.onLoginStatusChange(subscriber);
    return () => {
      AuthManager.offLoginStatusChange(subscriber);
    };
  }, [subscriber]);
  return loggedIn;
};
const RootRouter = () => {
  const loggedIn = useIsLoggedIn();
  return (
    <Router>
      {loggedIn ? (
        <AppLayout>
          <Switch>
            {appRouts.map(({ path, exact, Component }) => {
              return (
                <Route key={paths} exact={exact} path={path}>
                  <Component />
                </Route>
              );
            })}
```

```
                <Redirect to={paths.home} />
              </Switch>
            </AppLayout>
          ) : (
            <AuthLayout>
              <Switch>
                {authRouts.map(({ path, exact, Component }) => {
                  return (
                    <Route key={paths} exact={exact} path={path}>
                      <Component />
                    </Route>
                  );
                })}
                <Redirect to={paths.login} />
              </Switch>
            </AuthLayout>
          )}
        </Router>
      );
    };

export default RootRouter;
```

Додаток Г

Програмний JS код для відмальовування даних користувача на сторінці профілю «Profile.js»

```
import { faCog } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-
fontawesome';
import { Card, Container, Alert, Row, Col, ListGroupItem,
ListGroup} from 'react-bootstrap';
import { Link } from 'react-router-dom';
import { PROFILE_PHOTO } from '../../constants/settings';
import paths from '../../router/paths';

export default function Profile({ error, isLoading, profile,
isMyProfile }) {

  if (error) {
    return (
      <Container className="mt-4">
        <Alert variant="danger">{error.message}</Alert>
      </Container>
    );
  }

  if (isLoading && !profile) {
    return <Container className="mt-4">Loading...</Container>;
  }
  return (
    <Container className="mt-4">
      <>
        <Row>
          <Col lg="3" md="3" sm="4" xs="5">
            <Card bg="dark" text="white">
              <Card.Img
                src={profile.profilePhoto || PROFILE_PHOTO}
                variant="top"
                style={{ maxWidth: 'auto', maxHeight: '400px',
borderRadius: '50%' }}
                className="p-4"
              />
              <Card.Body>
                <Card.Title style={{ textAlign: 'center'
}}>{profile.nickname || 'none'}</Card.Title>
              </Card.Body>
            </Card>
          </Col>
          <Col lg="9" md="9" sm="8" xs="7">
            <Card>
              <Card.Body>
                <Card.Title
                  style={{ fontSize: '25px', display: 'flex',
flexDirection: 'row', justifyContent: 'space-between'
```

```
                    >
                      <div>
                        {profile.nickname || 'none'}
                      </div>
                      {isMyProfile && (
                        <Link style={{ cursor: 'pointer' }}
    to={paths.editProfile}>
                          <FontAwesomeIcon title="Edit profile
    information" icon={faCog} />
                        </Link>
                      )}
                    </Card.Title>
                    <ListGroup className="list-group-flush">
                      <ListGroupItem>Full name: {profile.fullName
    || 'none'}</ListGroupItem>
                      <ListGroupItem>Email: {profile.email ||
    'none'}</ListGroupItem>
                    </ListGroup>
                  </Card.Body>
                </Card>
              </Col>
            </Row>
          </>
        </Container>
      );
}
```

Додаток Ґ

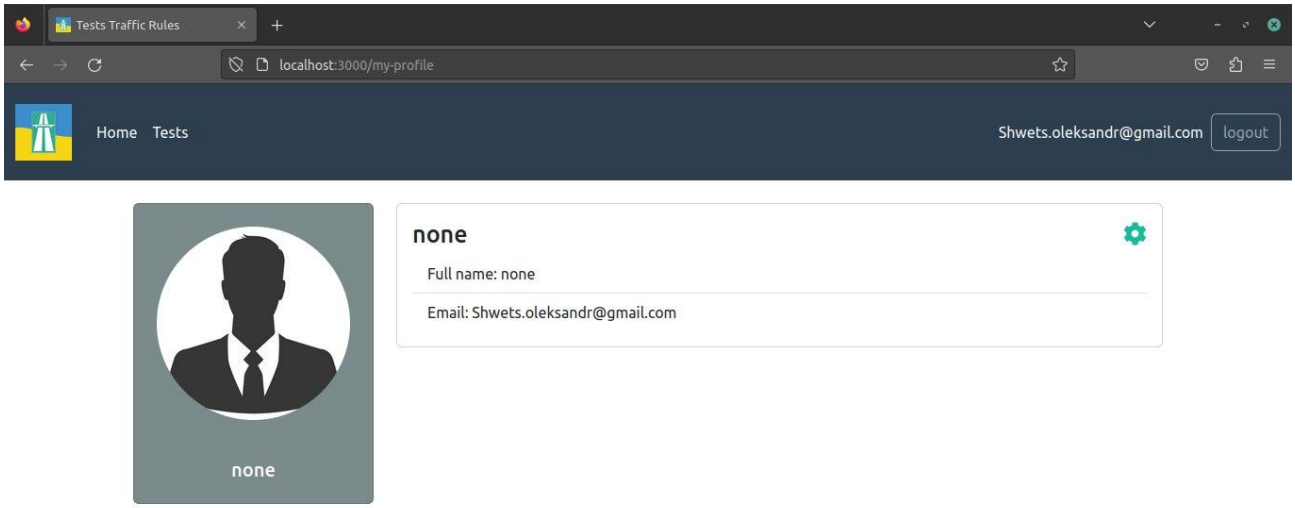

Рисунок Ґ.1 – Акаунт зареєстрованого користувача за замовчуванням

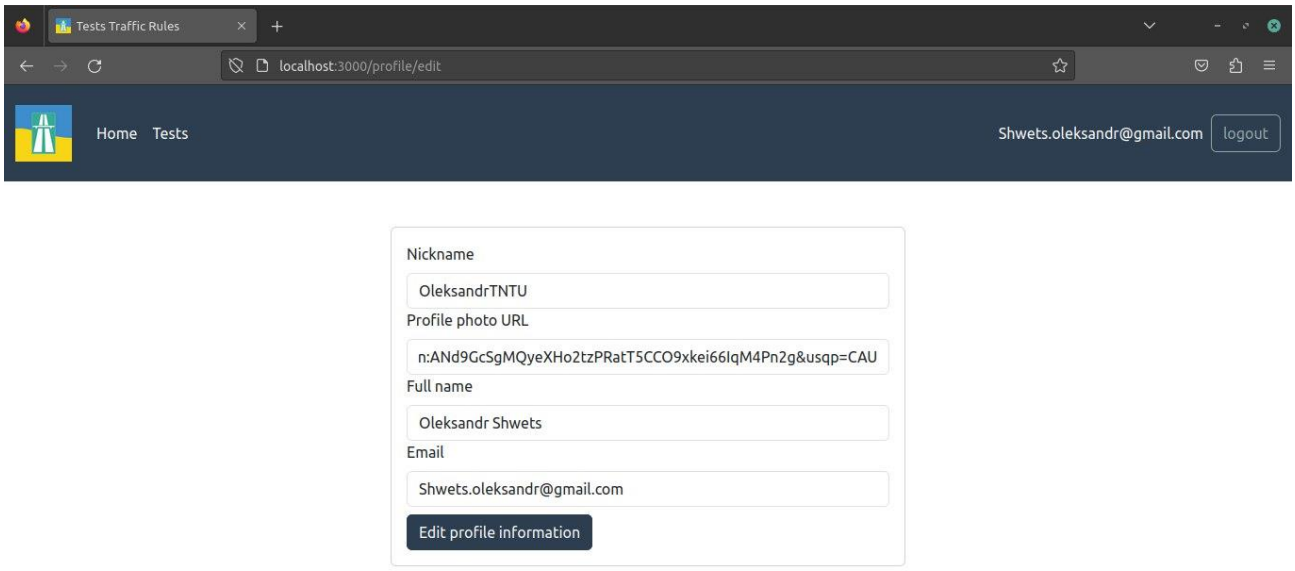

Рисунок Ґ.2 – Вікно налаштування акаунту користувача

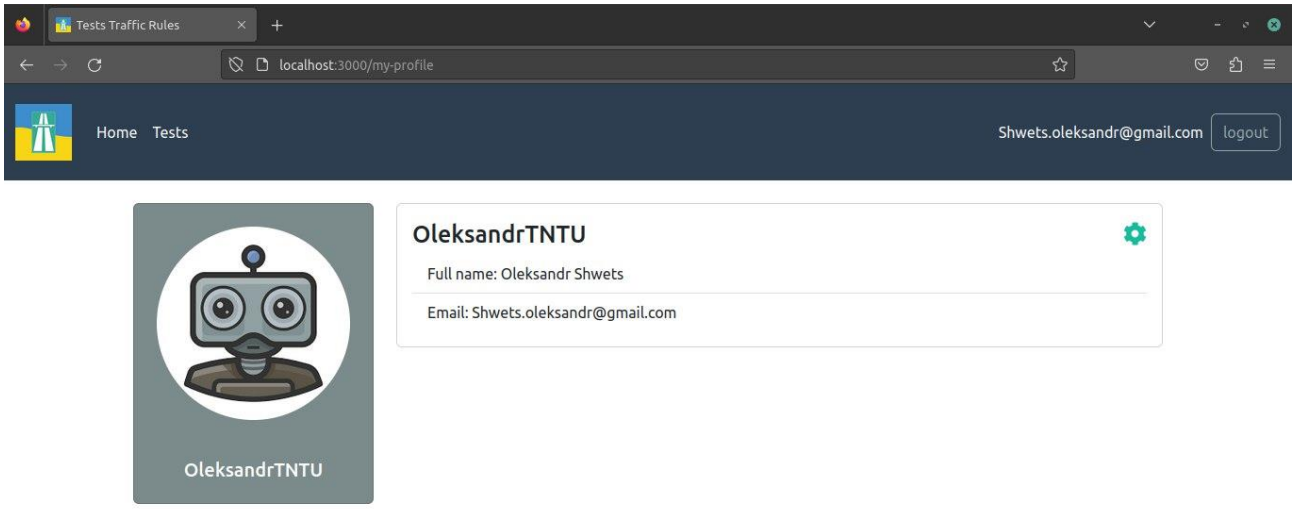

Рисунок Ґ.3 – Результат налаштувань акаунту