Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

# Розробка веб-сайту для підготовки водіїв з використанням HTML5 Web Workers

Текстова частина до курсової роботи
за спеціальністю “Інженерія програмного забезпечення” 121

Керівник курсової роботи
к. ф-м. н., с.в. Гречко А. В.
_____
(підпис)
“____” _____ 2021 р.
Виконала студентка Ровніна Т. Д.
“____” _____ 2021 р.

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри мережних технологій,
д. ф-м. н., професор
_____Г. І. Малашонок
“____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентці 4-го курсу, факультету інформатики, Ровніній Тетяні Дмитрівні

ТЕМА: Розробка веб-сайту для підготовки водіїв з використанням HTML5 Web Workers

Вихідні дані:

- використання HTML5 Web Workers;
- підтримка браузеру Google Chrome.

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1. Аналіз предметної області. Постановка завдання курсової роботи

2. Теоретичні відомості

3. Опис реалізації програмного продукту

4. Тестування програми і результати її виконання

Висновки

Список використаних джерел

Додатки

Дата видачі “___” _____ 2020 р. Керівник _____

Завдання отримав _____

**Тема:** Розробка веб-сайту для підготовки водіїв з використанням HTML5 Web Workers

**Календарний план виконання роботи:**

| № п/п | Назва етапу курсової роботи | Термін виконання етапу | Примітка |
|---|---|---|---|
| 1. | Отримання завдання на курсову роботу. | 02.10.2020 | |
| 2. | Огляд технічної літератури за темою роботи. | 12.12.2020 | |
| 3. | Проектування системи. | 01.01.2021 | |
| 4. | Розробка серверної частини веб-сайту. | 15.02.2021 | |
| 5. | Розробка клієнтської частини веб-сайту. | 14.03.2021 | |
| 6. | Розгортання проекту | 17.03.2021 | |
| 7. | Написання текстової частини. | 24.03.2021 | |
| 8. | Створення слайдів для доповіді та написання доповіді. | 03.04.2021 | |
| 9. | Обговорення отриманих результатів з керівником. | 30.03.2021 | |
| 11. | Остаточне оформлення текстової частини та слайдів. | 11.04.2021 | |
| 12. | Захист курсової роботи. | 19.04.2021 | |

**Студент** _____

**Керівник** _____

“_____”_____

**Зміст**

## Перелік умовних позначень

| | |
|---|---|
| БД | – база даних; |
| СКБД | – система керування базами даних; |
| ER-модель | – модель Entity-Relationship (модель «сутність-зв'язок»); |
| DDD | – Domain-driven design (Предметно-орієнтоване проектування); |
| EF | – Entity-Framework; |
| IoC | – Inversion of Control IoC; |
| AWS | – Amazon Web Services; |
| ПДР | – правила дорожнього руху. |

## Анотація

Курсова робота Ровніної Т. Д. на тему “Розробка веб-сайту для підготовки водіїв з використанням HTML5 Web Workers”, 146 с., 1 табл., 38 рис., 15 джерел, 3 додатки.

Об’єктом дослідження є веб-системи, які пропонують матеріали для підготовки українських водіїв до здачі екзамену на отримання права керування транспортним засобом.

Метою роботи є створення інформаційної веб-системи з необхідною інформацією для підготовки водіїв.

Створений програмний продукт містить такі складові:

- базу даних з необхідною інформацією;
- сервер для обробки запитів;
- веб-сайт для роботи з клієнтом.

Для демонстрації роботи системи створено веб-сайт “Світлофор”, який містить достовірний матеріал з правил дорожнього руху та тести з екзаменаційних білетів.

В результаті виконання роботи були досліджені теоретичні основи та прикладні підходи до реалізації процесу створення веб-застосувань та використання засобів паралельного програмування на стороні клієнта.

.NET, ASP.NET CORE, C#, DDD, DATABASE, MYSQL, ENTITY FRAMEWORK, CLIENT-SERVER, WEB, JAVASCRIPT, VUE.JS, HTML5 WEB WORKERS

# Вступ

Актуальність даної роботи полягає в тому, що в сучасному світі дуже поширене використання інтернет ресурсів для навчання. Це дає велику кількість переваг: доступ до даних у будь-який зручний час і з будь-якої точки світу, можливість задавати питання широкій аудиторії користувачів тощо. Вивчення правил дорожнього руху не є виключенням. Але підвищення доступності призводить до появи низки проблем, одна з яких – це велике навантаження на ресурси. Тому також актуальним є пошук шляхів підвищення ефективності використаня інтернет ресурсів в умовах великого навантаження. Одним із таких шляхів є використання потоків.

Метою роботи є отримання готової інформаційної веб-системи для підготовки водіїв, яка буде ефективно працювати в умовах високого навантаження завдяки асинхронному виклику деяких функцій.

Об’єктом дослідження є веб-системи організацій, які пропонують послуги з навчання водіїв, та технологія HTML5 Web Workers.

Обрані наступні методи дослідження:

– спостереження – вивчення послуг організацій з підготовки водіїв і прикладів застосування багато поточного програмування на стороні клієнта;

– аналіз – розбиття досліджуваного об’єкта на частини.

Серед джерел дослідження можна виділити такі основні:

– веб-сайти організацій «Vodiy.ua» і «Greenway»;

– електроні статті з матеріалами по HTML5 Web Workers: «Основные сведения об объектах Web Worker», «Как работает JavaScript: создание блоков Web Workers + 5 примеров, как их использовать»;

– веб-сайт з навчальним матеріалом по .NET «Metanit.com»

Створений програмний продукт буде містити такі складові:

– базу даних з необхідною інформацією;

– сервер для обробки запитів;

– веб-сайт для роботи з клієнтом.

Результат роботи системи та використання HTML5 Web Workers буде демонструватись на веб-сайті вигаданої організації для підготовки водіїв «Світлофор». Сайт буде містити достовірну інформацію з ПДР.

Виконання даної роботи зачіпає такі проблеми спеціальності, як веб-програмування, інтеграція з СКБД і сховищами даних, застосування розподіленої архітектури програмного забезпечення, створення зрозумілих і простих користувацьких інтерфейсів, використання багатопоточного програмування у розробці веб-сайтів.

Під час розробки проекту було використано таке програмне забезпечення:

– мови програмування JavaScript і C#;
– середовища розробки WebStorm, VisualStudio;
– платформа для створення серверу – ASP.NET Core;
– фреймворк Vue.js;
– продукти MySQL для взаємодії з базою даних;
– хмарні провайдери AWS та Heroku.

Робота складається зі вступу, чотирьох основних розділів, висновку, списку використаних джерел та додатків.

# 1. Аналіз предметної області. Постановка завдання курсової роботи

## 1.1. Аналіз сучасного стану питання та обґрунтування теми

Успіх здачі екзаменів на право керувати транспортним засобом залежить від рівня підготовки майбутнього водія. Правила дорожнього руху (ПДР) та питання в екзаменаційних білетах постійно змінюються і контролювати їх актуальність у друкованому вигляді доволі складно. З розвитком та розповсюдженням інтернету доступ до актуальної інформації перестав бути проблемою. На сьогодні створення веб-сайту з корисними матеріалами для водіїв допоможе підготуватись до екзаменів учням автошкіл, а також покращить знання вже бувалих водіїв.

До основних складових сайту з підготовки водіїв можна віднести:

– правила дорожнього руху;
– тести за темами ПДР;
– тести за білетами;
– інформація про актуальні штрафи за порушення ПДР.

Для автоматизації проведення тестування користувача передбачений інтерфейс з можливістю обрати тест за категорією, слідкувати за часом, який залишився на складання екзамену, а також вивід результату тестування.

Система дозволить модераторам сайту надавати доступ до актуальної інформації з ПДР та консультувати користувачів щодо незрозумілих питань у тестах.

## 1.2. Огляд існуючих аналогів розробки

На сьогодні існує доволі багато систем, які пропонують матеріали з ПДР. Серед них в якості аналогів створюваної системи обрано та проаналізовано найбільш популярні українські веб-сайти: «Vodiy.ua», «Greenway».

«Vodiy.ua» — найбільш популярний ресурс для підготовки водіїв в України. Окрім матеріалів для підготовки, він пропонує навчання у власній

автошколі. Також наявне застосування для Android iOS. Інтерфейс головної сторінки веб-сайту зображений на рис. 1.1.

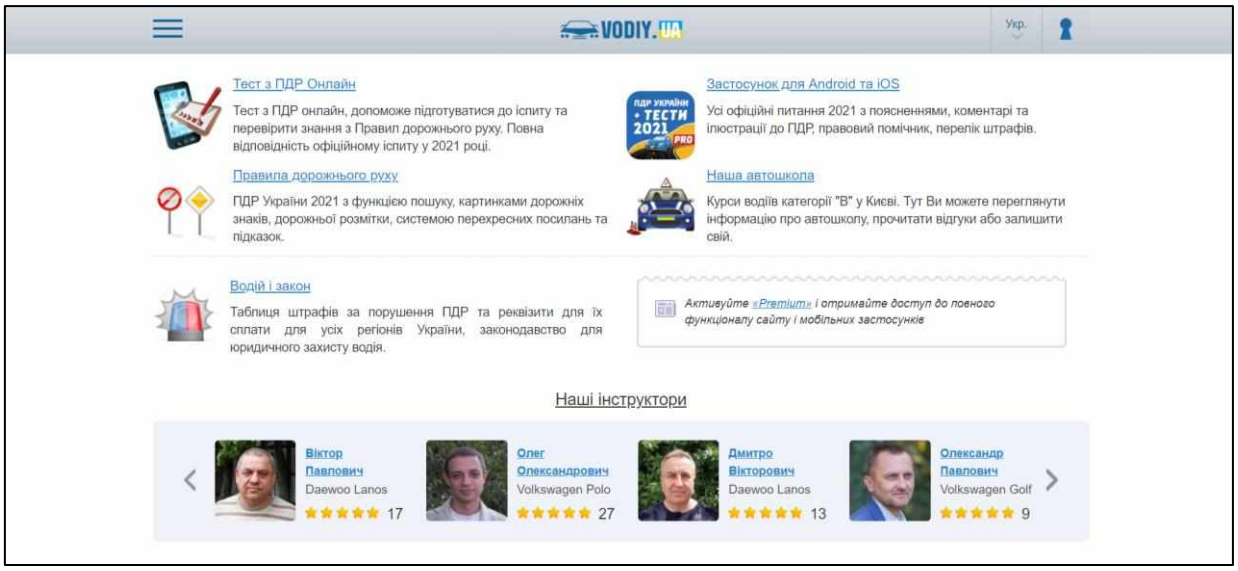


Рисунок 1.1 - Інтерфейс сайту «Vodiy.ua»

Веб-сайт «Vodiy.ua» має такі позитивні сторони:

- актуальні ПДР;
- таблиця штрафів з порушення ПДР;
- відеоуроки;
- реєстрація на сайті;
- тести за темами та білетами;
- збереження прогресу;
- можливість задати питання по тесту.

Але не зважаючи на велику кількість переваг, також у даного засобу наявні такі недоліки:

- не зручне меню фільтрації;
- багато корисного матеріалу доступно лише для преміум акаунтів;
- користувачі не можуть відповідати на коментарі;
- забагато реклами власної продукції.

Окрему увагу можна звернути на дизайн веб-сайту. Основні кольори – сірий та білий. Для виділення інформації використовуються відтінки синього кольорів. Через кольорову гамму сайт виглядає просто і не привертає увагу. Сторінки містять забагато зайвої інформації, такої як реклама ресурсів організації та додаткові меню.

«Greenway» — позиціонує себе як єдиний офіційний ресурс з тестами ПДР. Як і попередній ресурс має свою автошколу. На рис. 1.2 наведено зображення головної сторнки цього сайту.

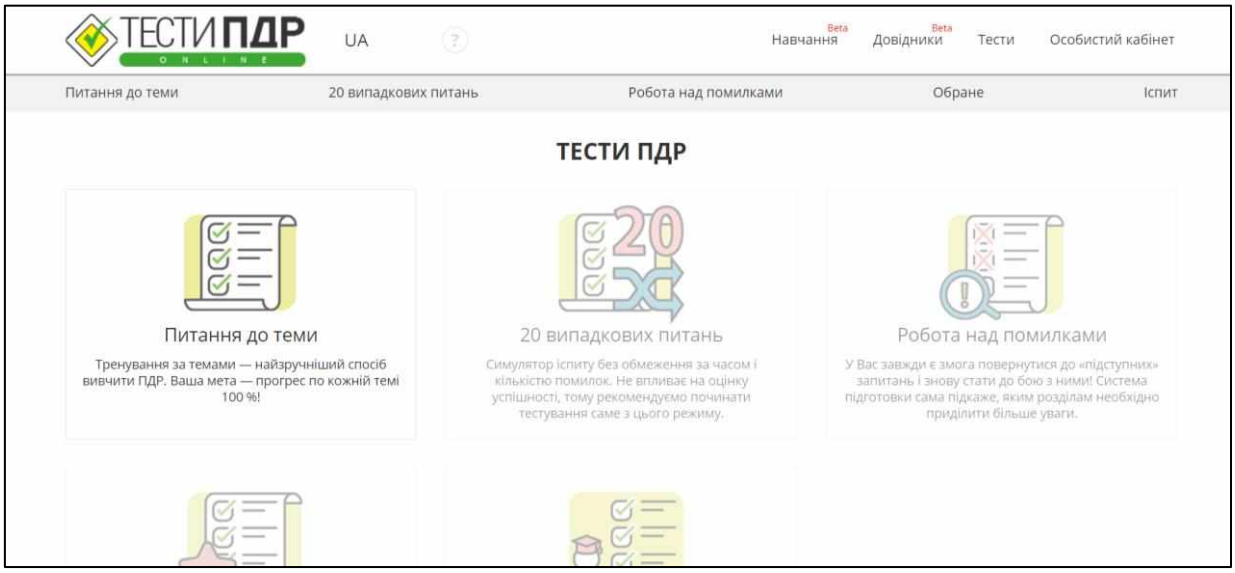


Рисунок 1.2 - Інтерфейс сайту «Greenway»

Веб-сайт має схожі характеристики, що і «Vodiy.ua». Позитивні сторони «Greenway»:

– актуальні ПДР;
– реєстрація на сайті;
– тести за темами;
– можливість задати питання;
– порівняння свого прогресу з іншими користувачами.

Недоліки веб-сайту:

– багато корисного матеріалу доступно лише для преміум акаунтів;

– немає питань за білетами.

Дизайн сайту має забагато білого кольору, але розмітка виглядає привабливо і стильно. Не зручно читати правила, оскільки кожне правило розміщено на новій сторінці. Аналогічна проблема з тестами. Картинки для тестів не відображаються через блокування реклами.

Проаналізувавши ці веб-сайти можна порівняти їх за такими основними характеристиками:

– зручність інтерфейсу;
– дизайн сайту;
– відсутність зайвої інформації;
– доступність інформації;
– зручність проходження тестування;
– фільтрація тестів;
– додаткові матеріали для навчання;
– можливість задавати питання.

Наявність чи відсутність цих характеристик у кожному з розглянутих сайтів наведена у табл. 1.1.

Таблиця 1.1 – Таблиця результатів порівняння існуючих систем

| **Характеристики** | **Vodiy.ua** | **Greenway** |
|---|---|---|
| зручність інтерфейсу | + | - |
| дизайн сайту | - | + |
| відсутність зайвої інформації | - | - |
| доступність інформації | - | - |
| зручність проходження тестування | + | - |

| | | |
|---|---|---|
| фільтрація тестів | + | - |
| додаткові матеріали | + | - |
| можливість задати питання | + | + |

### 1.3. Постановка завдання

Метою роботи є створення інформаційної веб-системи з необхідною інформацією для підготовки водіїв. Для забезпечення функціонування сайту потрібно створити базу даних, яка буде містити актуальну інформацію, та сервер для зв'язку між базою та клієнтом. Для пришвидшення роботи системи та оптимізації використання ресурсів на стороні клієнта треба використати технологію HTML5 Web Workers.

Веб-сайт має містити актуальні ПДР та надавати наступні функціональні можливості:

- перегляд таблиці штрафів за порушення ПДР;
- перевірка знань шляхом проходження тестування;
- фільтрація тестів за темами чи білетами;
- наявність таймеру під час тестування;
- збереження прогресу користувачів;
- порівняння прогресу з іншими користувачами;
- можливість залишити коментарі до тестів;
- реєстрація та авторизація користувачів.

Сайт має відповідати наступним вимогам:

- надавати інформацію українською мовою;
- забезпечувати контроль інформації, яку вводить користувач під час реєстрації та авторизації, з метою запобігання введення помилкових даних;

- контент сторінки має бути написаний на зрозумілій для користувача мові;
- забезпечувати естетичну і функціональну однаковість окремих сторінок сайту. Наприклад, кольорова гама сайту має бути функціональною, тобто виділяти деталі кольором, який буде гармонічно поєднуватись з фоном сайту.

## 2. Теоретичні відомості

### 2.1. Паралельне програмування. HTML5 Web Workers

JavaScript – поширена, об'єктно-орієнтована прототипна мова програмування. Проблема в тому, що це однопоточне середовище, в якому кілька скриптів не можуть виконуватись одночасно. Технологія HTML5 Web Workers може вирішити цю проблему.

Специфікація об'єктів Web Worker визначає API для створення фонових скриптів у веб-застосуваннях. Ці об'єкти дозволяють запускати довготривалі скрипти для виконання завдань, що вимагають великого обсягу обчислень, не вдаючись до блокування інтерфейсу користувача або інших скриптів, керуючих взаємодією користувача з системою. Об'єкти Web Worker використовують потокову передачу повідомлень для реалізації паралельності [14].

Можливості доступні Web Worker:

- об’єкт navigator;
- об’єкт location (тільки для читання);
- XMLHttpRequest;
- setTimeout()/clearTimeout(), а також setInterval()/clearInterval();
- кеш застосунку;
- імпорт зовнішніх скриптів importScripts();
- створення інших Web Workers.

Web Workers також мають обмеження. Вони не мають доступу до об’єктів DOM, window, document, parent оскільки вони є потоконебезпечними.

Зазвичай прийнято використовувати Web Workers в таких ситуаціях: трасування променів, шифрування, попередня вибірка даних, прогресивні веб-застосування, перевірка орфографії [13].

## 2.2. Опис засобів розробки

### 2.2.1. СКБД

Система керування базами даних (СКБД) – це програмне забезпечення, за допомогою якого користувачі можуть визначати, створювати та підтримувати базу даних, а також здійснювати контрольований доступ до неї.

Реляційна база даних – база даних, заснована на реляційній моделі даних. Для роботи з реляційними БД застосовують реляційні СКБД. Щоб зберігати і працювати з даними, такий тип  повинен мати певну структуру таблиці і відповідати аспектам цілісності [15].

MySQL – це найпопулярніша повноцінна серверна СКБД. Функціонал MySQL доволі широкий. Навчитися використанню цієї СКБД не складно оскільки існує велика кількість матеріалів про неї в мережі [10].

Плюси використання MySQL:

– простота у використанні;

– багатий функціонал: MySQL підтримує більшість функціоналу SQL;

– велика кількість функцій забезпечують безпеку;

– може працювати з великим обсягом даних і підходить для масштабованих додатків;

– нехтування деякими стандартами дозволяє MySQL працювати продуктивніше.

Недоліки MySQL:

– присутні певні обмеження функціоналу;

– немає вбудованої підтримки для XML;

– застій в розробці: робота над цим продуктом сильно загальмована.

### 2.2.2. Серверна частина

#### 2.2.2.1. ASP.NET Core

ASP.NET Core – це платформа від компанії Microsoft, призначена для створення різного роду веб-додатків. Вона є розширеною версією ASP.NET [3]. Можна виділити наступні переваги ASP.NET Core:

– легкий і модульний конвеєр HTTP-запитів;

– можливість розгортати додаток як на IIS, так і в рамках свого власного процесу;

– використання платформи .NET Core і її функціональності;

– інтегрована підтримка для створення та використання пакетів NuGet;

– конфігурація для спрощеного використання в хмарі;

– вбудована підтримка для впровадження залежностей;

– можливість розширення;

– кросплатформеність: можливість розробки і розгортання додатків ASP.NET на Windows, Mac і Linux.

#### 2.2.2.2. Entity Framework Core

Entity Framework (EF) Core – це проста, кросплатформена версія технології доступу до даних, що дозволяє автоматично зв'язати звичайні класи з таблицями в базі даних. Ця технологія є чудовим рішенням для об’єктно-реляційного відображення даних [3].

Переваги EF Core:

– дозоляє розробникам .NET працювати з БД за допомогою об’єктів .NET;

– спрощує код програми для доступа до даних;

– підтримує сумісність з більшістю СКБД.

Технологія EF Core використовується у «Code First» підході написання програм. Це робить БД звичайним сховищем даних, а систему менш залежною від неї. Також підхід «Code First» дозволяє легко носити зміни у БД і вирішувати проблеми, які з'являються під час розробки системи.

#### 2.2.2.3. AutoMapper

AutoMapper – це бібліотека, завдяки якій можна проектувати одну модель даних на іншу. Це дозволяє спростити програму, зменшивши написання великої кількості нудного коду.

Перетворення може відбуватися в багатьох місцях програми, але переважно в межах між рівнями представлення даних. Наприклад, між рівнями інтерфейсу користувача та домену або рівнями сервісів і домену.

Проблеми рівнів можуть мати конфлікти один з одним. Проектування моделі на іншу модель вирішить це і призведе до згенерованих моделей, де проблеми кожного рівня впливатимуть лише на поточний рівень представлення даних [6].

#### 2.2.2.4. Autofac

Inversion of Control – це принцип проектування, який передбачає використання інверсії різних елементів управління в об'єктно-орієнтованому дизайні для досягнення вільного зв'язку між класами застосунку. Ідея полягає в тому, що замість того, щоб зв'язувати класи у програмі і дозволяти їм оновлювати свої залежності, перемикаються інверсії, а залежності передаються під час побудови класу [8].

Autofac – це Inversion of Control контейнер, за допомогою якого програма може використовувати принцип IoC. Він забезпечить керованість залежностями і зменшить написання коду.

### 2.2.3. Клієнтська частина

#### 2.2.3.1. Фреймворк Vue.js

Vue – це прогресивний фреймворк для побудови інтерфейсів на стороні клієнта. На відміну від інших популярних фреймворків, Vue розроблений з нуля, щоб користувач міг поступово його застосовувати. Основна бібліотека орієнтована лише на рівень представлення та її легко можна інтегрувати з іншими бібліотеками або існуючими проектами. Vue є чудовою технологією для написання складних односторінкових програм, коли використовується у поєднанні з сучасними засобами розробки та бібліотеками, що його підтримують [5]. Переваги Vue над іншими популярними фреймворками:

– Оптимізація: залежності компонента автоматично відслідковуються під час відтворення, тому система точно знає, які компоненти дійсно необхідно повторно малювати при зміні стану, а які залишити без змін.

– Використання HTML-шаблонів: вони полегшують поступову міграцію існуючих додатків для використання можливостей реактивності Vue і роблять код легшим для розуміння.

– Модульний CSS: однофайловий компоненти надають кожному шаблону повний доступ до CSS в тому ж файлі, що і решта коду компонента.

#### 2.2.3.2. Бібліотека Axios

Axios – це HTTP-клієнт, розроблений з використанням промісів, для браузера. Він дозволяє легко відправляти асинхронні HTTP-запити до кінцевих точок REST та виконувати CRUD-операції. Його можна використовувати як в простому JavaScript так і з фреймворками Vue або React [11]. Переваги Axios:

– відправка XMLHttpRequests із браузера;

– підтримує API Promise;

– автоматично перетворює дані в JSON;

– наявна можливість скасовувати запити;

– перехоплює запити та відповіді.

#### 2.2.3.3. Набір інструментів Bootstrap

Bootstrap – це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-додатків. Він містить шаблони CSS та HTML для представлення форм, кнопок, навігації та інших компонентів інтерфейсу. Також він містить додаткові розширення JavaScript [12].

BootstrapVue – набір інструментів Bootstrap, який містить шаблони компонентів та плагінів для роботи з Vue.js.

Використання компонентів Bootstrap пришвидшує верстку сайту та робить його інтерфейс більш привабливим.

## 3. Опис реалізації програмного продукту

### 3.1. Аналіз технічного завдання

У цьому розділі докладно розглянуто пункти технічного завдання, викладеного у підрозділі 1.3.

Доступ до даних веб-сайту можуть мати лише зареєстровані користувачі. Користувач ідентифікується за емейлом і паролем. Якщо такого аккаунту не існує, то надається можливість зареєструватись у системі. Для реєстрації потрібно ввести email, ім'я, придумати пароль і підтвердити його.

Після авторизації користувачу має надаватись токен. Якщо токен стає недійсним, користувач не зможе переглядати матеріал веб сайту і повертається на сторінку авторизації.

Правила дорожнього руху мають бути відсортовані за темами і містити достовірні дані. У таблиці штрафів за порушення ПДР мають відображатись номер статті закону України, формулювання закону та грошове стягнення.

Тестування користувача може проводитись як за конкретними темами ПДР так і за екзаменаційними білетами. Якщо користувач обрав білет, то його результат зберігається в прогрес даного білету. Під час проходження тесту має бути видно час, який залишився до кінця тестування. По закінченню виділеного часу тестування завершується автоматично. Також тестування можна завершити за бажанням користувача. У результаті тестування має відображатись час витрачений користувачем, процент правильних відповідей від загальної кількості питань, кількість правильних і неправильних відповідей.

Коментарі до тестів відображаються на сторінці тестування. Окрім самого коментаря вони мають номер тесту, якому належать, данні про користувача та дату створення. Користувач може як задати нове питання до тесту так і відповісти на вже існуючі.

На сторінці профілю користувача має відображатись його дані такі як ім'я, email та фотографія. Також має бути наявний його прогрес у проходженні білетів. Має відображатись кількість правильних і неправильних відповідей,

кількість спроб складання білету та процент правильних відповідей від усіх завдань.

Користувач має можливість знайти інших користувачів, за прогресом яких він хоче слідкувати. Має відображатись фотографія і ім'я «друга», а також його прогрес, а саме кількість пройдених білетів, виділяючи кількість тих, що не набрали прохідні 90% прогресу, і тих, що мають більше ніж 90% правильних відповідей.

Щоб запобігти введення помилкових значень потрібно надавати підказки до полів та робити валідацію даних.

## 3.2. Архітектура системи

У цьому підрозділі наводиться опис архітектури системи та взаємодії її елементів. Архітектура системи веб-сайту «Світлофор» представлена на рис. 3.1.

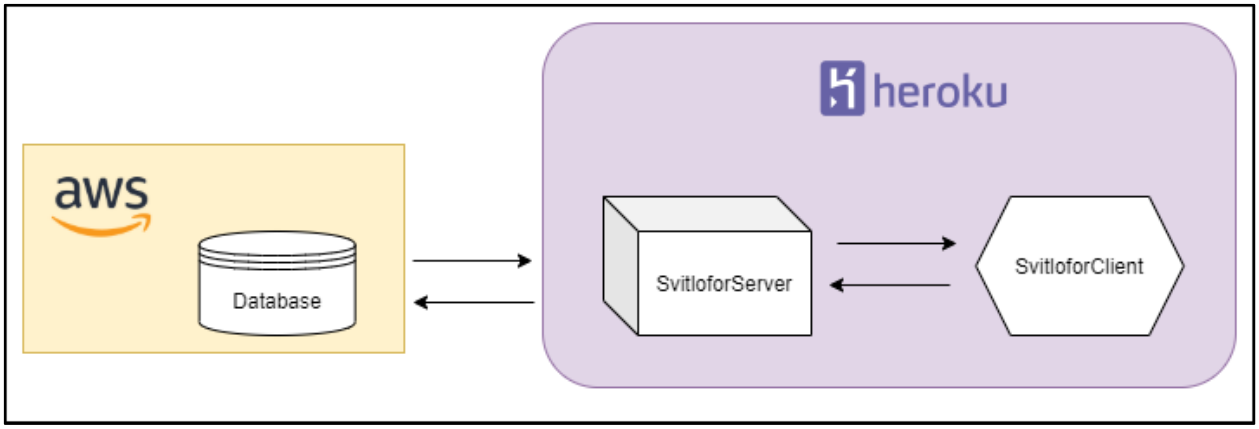


Рисунок 3.1 – Архітектура веб-системи «Світлофор»

До складу системи входять такі компоненти:

– база даних MySQL, розміщена на AWS;
– веб-сервер ASP.NET Core, розгорнений на Heroku;
– веб-сайт на стороні клієнта, розгорнений на Heroku.

Один з найбільш популярних шаблонів проектування архітектури веб-серверу є DDD (Domain-driven design). Предметно-орієнтоване проектування — це набір принципів і схем, спрямованих на створення оптимальної системи, шляхом розділення реалізації та моделі, що розроблюється. Основний підхід цього шаблону полягає в тому, що класи системи моделюються на основі реальних бізнес процесів. DDD використовується за наступних умов:

– акцент проекту спрямований на предметну область;
– увага до розробки дизайну предметної області;
– співпраця між експертами технічної та предметної областей проекту з метою вдосконалення концептуальної моделі системи.

Архітектура DDD складається з таких рівнів: рівень представлення, програмний рівень, рівень домену, рівень інфраструктури [4].

**Рівень представлення:** надає інтерфейс користувача. Використовує програмний рівень для досягнення взаємодії користувачів.

**Програмний рівень:** посередник між рівнями представлення та домену. Реалізує логіку програми. Оперує моделями для виконання конкретних прикладних завдань.

**Рівень домену:** включає моделі даних та основні бізнес-правила.

**Рівень інфраструктури:** надає загальні технічні можливості, які підтримують вищі рівні, переважно за допомогою сторонніх бібліотек.

Здебільшого під час реалізації шаблону DDD використовують лише програмний рівень і рівень домену. На рис. 3.2 зображена архітектура веб-серверу системи «Світлофор».

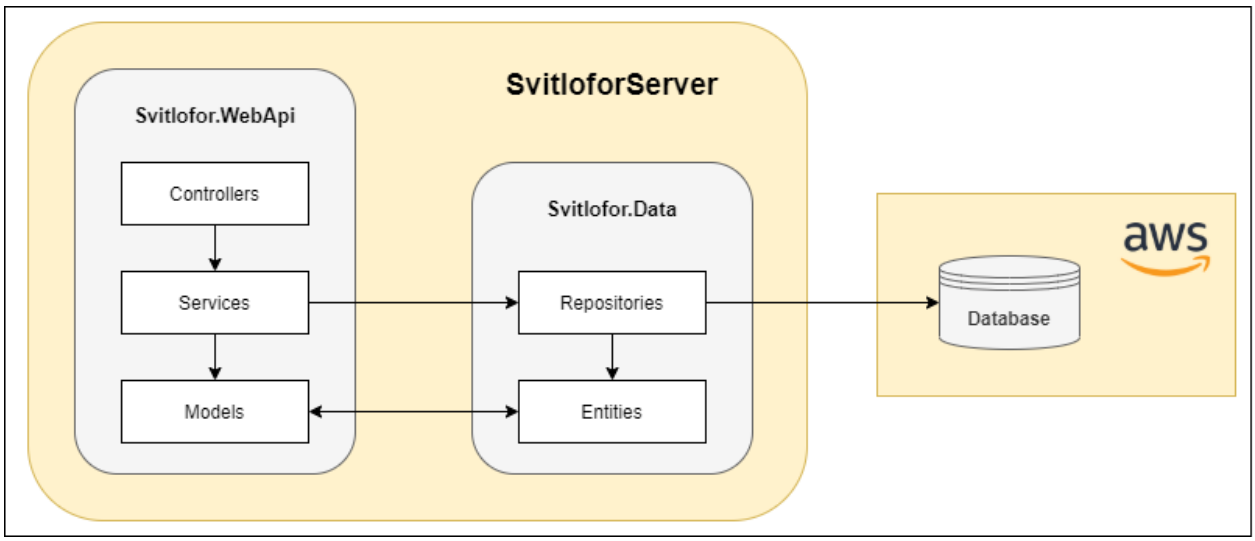


Рисунок 3.2 – Архітектура веб-серверу системи «Світлофор»

Програмний рівень системи представлений частиною Svitlofor.WebApi. В нього входять контролери, сервіси, а також моделі даних.

**Контролер –** є основним компонентом серверу. Він забезпечує зв'язок між користувачем та системою, приймаючи вхідні HTTP-запити. В залежності від результатів обробки відправляє користувачеві певний висновок.

**Сервіс –** забезпечує логіку обробки вхідних даних. Їх використовують для зв'язку з рівнем домену, а саме з репозиторіями системи.

**Модель даних** – використовуються для передачі даних між програмним рівнем та рівнем представлення. Модель відповідає певній сутності у БД, але відображає об’єкт у тому вигляді, в якому його використовує програма, з метою більш повного відображення даних.

За рівень домену системи відповідає частина Svitlofor.Data. Він складається з сутностей та репозиторіїв.

**Сутність** – базове поняття, за допомогою якого моделюється клас однотипних об'єктів, які розглядаються в конкретній предметній області. Ці класи описують таблиці, що містяться у БД.

**Репозиторій** – використовуються для виконання операцій над БД для об’єктів домену. Кожна сутність програми використовує власний репозиторій.

### 3.3. Реалізація бази даних

Першим етапом у розробці бази даних є побудова її ER-моделі.

ER (Entity-Relationship) - модель (модель «сутність-зв'язок») — графічна мова моделювання, яка дозволяє за допомогою узагальнених конструкцій блоків описувати об’єкти БД, їх властивості, зв’язки і обмеження.

На рис. 3.2 зображена схема ER-моделі бази даних системи для підготовки водіїв «Світлофор».

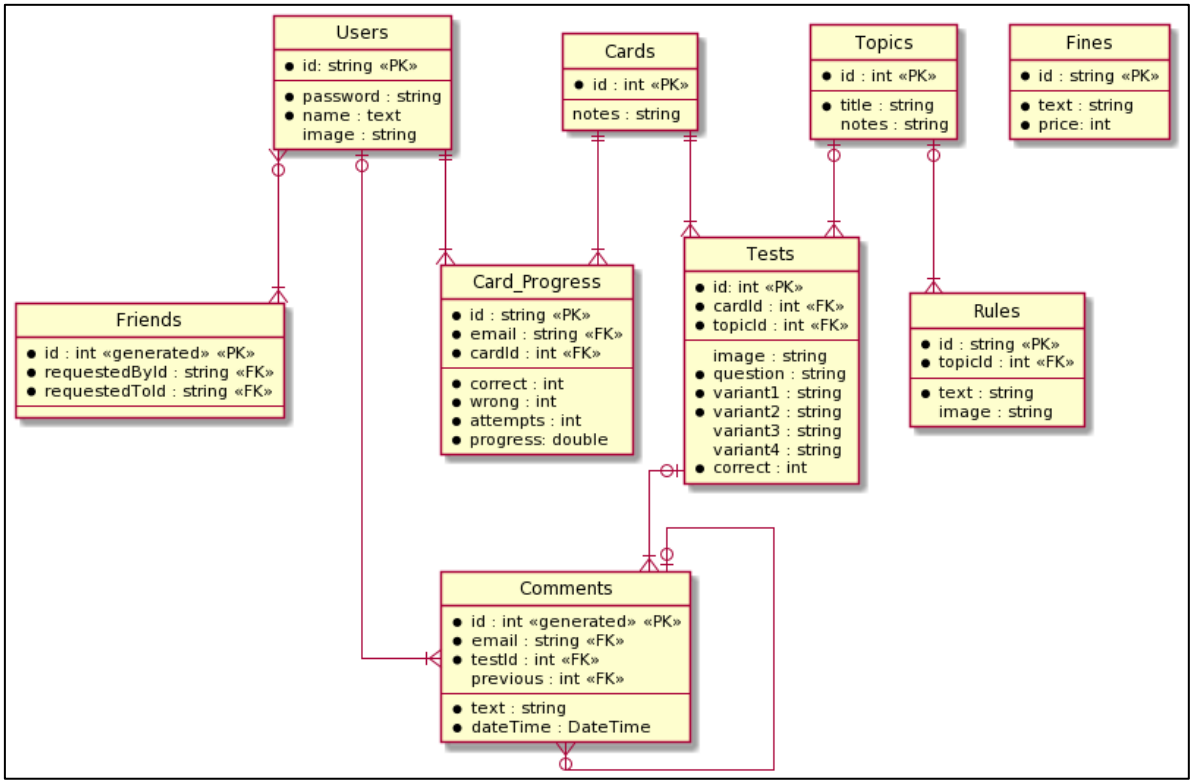


Рисунок 3.3 – ER-модель бази даних системи

Сутність **«Users»** відповідає за користувачів системи.

id – ключ, емейл користувача.

password – атрибут пароль користувача.

name – атрибут ім’я користувача.

image – необов’язковий атрибут фотографія.

Сутність **«Friends»** відповідає за зв’язок між користувачами, які є друзями.

id – автозгенерований ключ.

requestedById – зовнішній ключ, який посилається на користувача, від якого надійшов запит на дружбу.

requestedToId – зовнішній ключ, який посилається на користувача, якому надійшов запит на дружбу.

Сутність **«Comments»** відповідає за коментарі, які залишають користувачі системи до тестів.

id – автозгенерований ключ.

email – зовнішній ключ, який посилається на користувача, що написав коментар.

testId – зовнішній ключ, який посилається на тест, до якого написаний коментар.

previous – необов’язковий зовнішній ключ, який посилається на попередній коментар.

text – атрибут текст коментаря.

dateTime – атрибут дата і час, коли був написаний коментар.

Сутність **«Cards»** відповідає за екземенаційні білети.

id – ключ, номер білету.

notes – необов’язковий атрибут замітки то білету.

Сутність **«Topics»** відповідає за теми ПДР.

id – ключ, номер теми ПДР.

title – атрибут назва теми.

notes – необов’язковий атрибут замітки до теми.

Сутність **«Rules»** відповідає за правила дорожнього руху.

id – ключ, номер правила дорожнього руху.

topicId – зовнішній ключ, який посилається на тему, якій належить правило.

text – атрибут текст правила.

image – необов’язковий атрибут картинка до правила.

Сутність **«Tests»** відповідає за тести по темам ПДР.

id – ключ, номер тесту.

cardId – зовнішній ключ, який посилається на білет, якому належить тест.

topicId – зовнішній ключ, який посилається на тему, якій належить тест.

image – необов’язковий атрибут картинка до тесту.

question – атрибут питання тесту.

variant1 – атрибут перший варіант відповіді.

variant2 – атрибут другий варіант відповіді.

variant3 – необов’язковий атрибут третій варіант відповіді.

variant4 – необов’язковий атрибут четвертий варіант відповіді.

correct – атрибут номер правильної відповіді.

Сутність **«CardProgress»** відповідає за прогрес користувача у проходженні тестів за екзаменаційними білетами.

id – ключ, який містить номер білету та email користувача.

email – зовнішній ключ, який посилається на користувача, якому належить прогрес.

cardId – зовнішній ключ, який посилається на білет, якому належить прогрес.

correct – атрибут кількість правильних відповідей.

wrong – атрибут кількість неправильних відповідей.

attempts – атрибут кількість спроб проходження білету.

progress – атрибут процент правильних відповідей.

Сутність **«Fines»** відповідає за штрафи за порушення ПДР.

id – ключ, номер статті закону України.

text – атрибут текст закону.

price – атрибут сума, яку треба сплатити за порушення закону.

### 3.4. Опис розробки серверної частини

Серверна частина системи побудована за допомогою технології ASP.NET Core та має DDD архітектуру.

Сервер поділяється на два проекти:

**Svitlofor.Data** – має доступ до БД. Діаграма класів представлена на рис. 3.4. Код проекту наведено у Додатку А.

**Svitlofor.WebApi** – містить функціонал для взаємодії з клієнтом. Діаграма класів представлена на рис. 3.5. Код проекту наведено у Додатку Б.

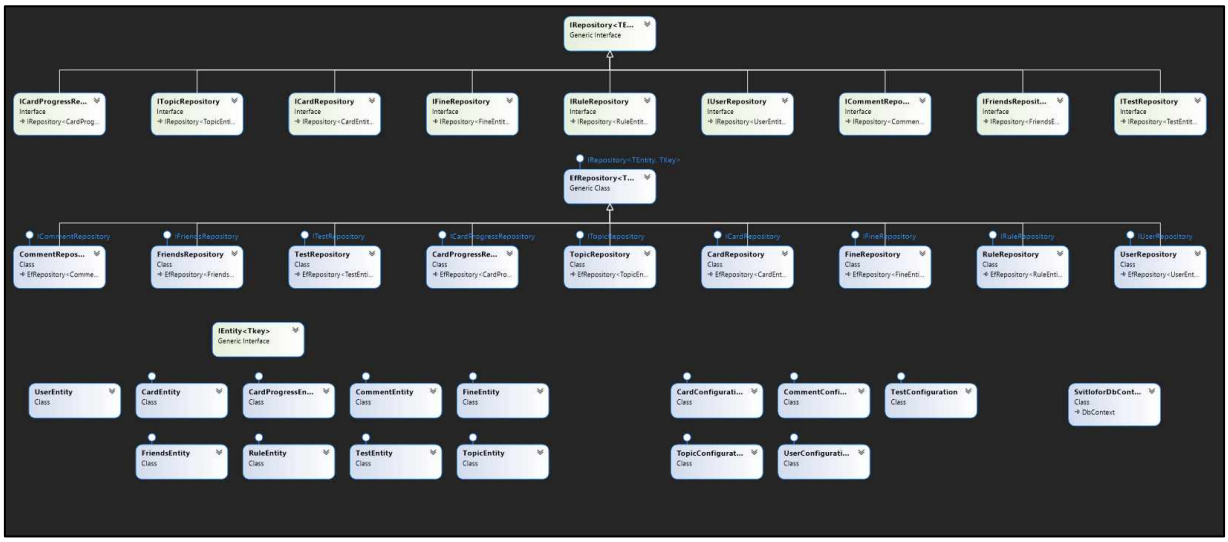


Рисунок 3.4 – Діаграма класів проекту Svitlofor.Data на сервері

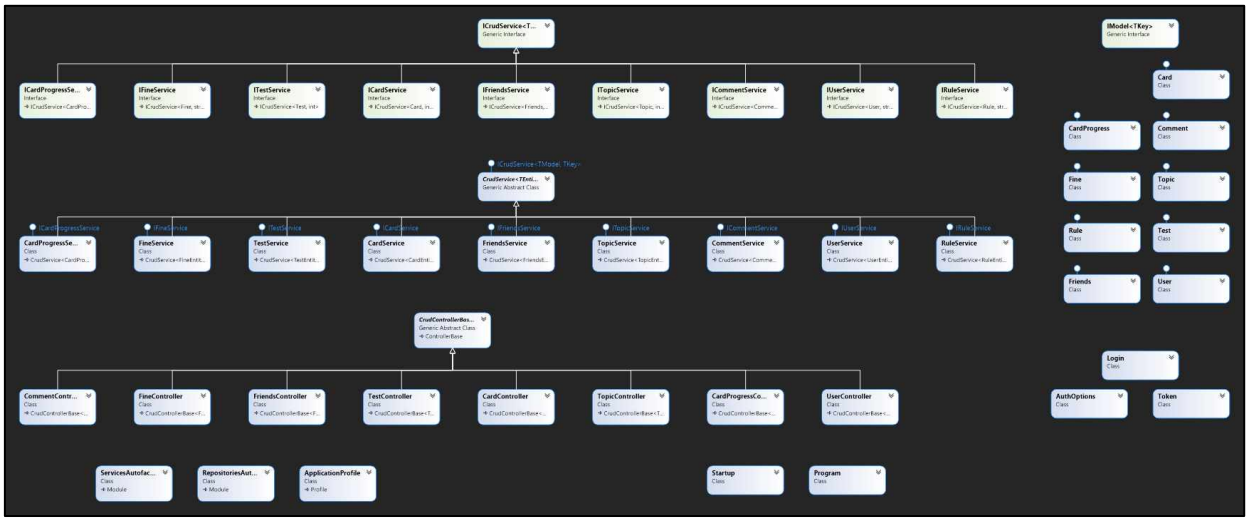


Рисунок 3.5 – Діаграма класів проекту Svitlofor.WebApi на сервері

Для зручного відображення даних програмою на основі класів сутностей були розроблені класи, що відповідають за моделі об’єктів і які програма використовує для обміну даними. Нижче наведено схеми цих моделей та відповідні API.

Реалізація функціоналу для моделі «Card» зображеної на рис. 3.6:

GET – ‘/api/Card’ – отримання всіх білетів.

POST – ‘/api/Card’ – створення білету. Також для кожного користувача створюється прогрес за цим білетом.

GET – ‘/api/Card/{id}’ – отримання білету за його id.

PUT – ‘/api/Card/{id}’ – редагування білету за його id.

DELETE – ‘/api/Card/{id} – видалення білету за його id. Також видаляється прогрес користувачів за цим білетом.

```
Card ⌄ {
    id                      integer($int32)
    notes                   string
                            nullable: true
}
```

Рисунок 3.6 – Схема моделі Card

Реалізація функціоналу для моделі «CardProgress» зображеної на рис. 3.7:

GET – ‘/api/CardProgress’ – отримання всіх збережених прогресів користувачів за білетами.

POST – ‘/api/CardProgress’– створення прогресу користувача.

GET – ‘/api/CardProgress/{id}’ – отримання прогресу за його id.

PUT – ‘/api/CardProgress/{id}’ – редагування прогресу за його id.

DELETE – ‘/api/CardProgress/{id}’ – видалення прогресу за його id.

```
CardProgress ⌄ {
    id                      string
                            nullable: true
    email                   string
                            nullable: true
    cardId                  integer($int32)
    correct                 integer($int32)
    wrong                   integer($int32)
    attempts                integer($int32)
    progress                number($double)
}
```

Рисунок 3.7 – Схема моделі CardProgress

Реалізація функціоналу для моделі «Comment» зображеної на рис. 3.8:

GET – ‘/api/Comment’ – отримання всіх коментарів.

POST – ‘/api/Comment’ – створення коментару.

GET – ‘/api/Comment/{id}’ – отримання коментару за його id.

PUT – ‘/api/Comment/{id}’ – реагування коментару за його id.

DELETE – ‘/api/Comment/{id}’ – видалення коментару за його id.

```
Comment ⌄ {
    id                  integer($int32)
    text                string
                        nullable: true
    testId              integer($int32)
    email               string
                        nullable: true
    login               string
                        nullable: true
    image               string
                        nullable: true
    dateTime            string($date-time)
    previousId          integer($int32)
                        nullable: true
    next
                         ⌄ [
                        nullable: true
                         > {...}]
}
```

Рисунок 3.8 – Схема моделі Comment

Реалізація функціоналу для моделі «Fine» зображеної на рис. 3.9:

GET – ‘/api/Fine’ – отримання всіх штрафів.

POST – ‘/api/Fine’ – створення штрафу.

GET – ‘/api/Fine/{id}’ – отримання штрафу за його id.

PUT – ‘/api/Fine/{id}’ – редагування штрафу за його id.

DELETE – ‘/api/Fine/{id}’ – видалення штрафу за його id.

```
Fine ⌄ {
    id                  string
                        nullable: true
    text                string
                        nullable: true
    price               integer($int32)
}
```

Рисунок 3.9 – Схема моделі Fine

Реалізація функціоналу для моделі «Friends» зображеної на рис. 3.10:

GET – ‘/api/Friends’ – отримання всіх пар друзів.

POST – ‘/api/Friends’ – створення всіх пари друзів.

GET – ‘/api/Friends/{id}’ – отримання пари друзів за її id.

PUT – ‘/api/Friends/{id}’ – редагування пари друзів за її id.

DELETE – ‘/api/Friends/{id}’ – видалення пари друзів за її id.

```
Friends ⌄ {
    id                  integer($int32)
    requestedById       string
                        nullable: true
    requestedToId       string
                        nullable: true
    friend              User > {...}
    isAccepted          boolean
}
```

Рисунок 3.10 – Схема моделі Friends

Реалізація функціоналу для моделі «Test» зображеної на рис. 3.11:

GET – ‘/api/Test’ – отримання всіх тестів та коментарів до них.

POST – ‘/api/Test’– створення тесту.

GET – ‘/api/Test/{id}’ – отримання тесту за його id та коментарів до нього.

PUT – ‘/api/Test/{id}’ – редагування тесту за його id.

DELETE – ‘/api/Test/{id}’ – видалення тесту за його id.

```
Test ⌄ {
    id                  integer($int32)
    image               string
                        nullable: true
    question            string
                        nullable: true
    variant1            string
                        nullable: true
    variant2            string
                        nullable: true
    variant3            string
                        nullable: true
    variant4            string
                        nullable: true
    correct             integer($int32)
    notes               string
                        nullable: true
    cardId              integer($int32)
    topicId             integer($int32)
    comments             ⌄ [
                        nullable: true
                        Comment > {...}]
}
```

Рисунок 3.11 – Схема моделі Test

Реалізація функціоналу для моделі «Topic» зображеної на рис. 3.12:

GET – ‘/api/Topic’ – отримання всіх тем та правил до них.

POST – ‘/api/Topic’ – створення теми.

GET – ‘/api/Topic/{id}’ – отримання теми за її id та правил до неї.

PUT – ‘/api/Topic/{id}’ – редагування теми за її id та її правил.

DELETE – ‘/api/Topic/{id}’ – видалення теми за її id та правил, що їй належать.

```
Topic ⌄ {
    id                  integer($int32)
    title               string
                        nullable: true
    notes               string
                        nullable: true
    rules                ⌄ [
                        nullable: true
                        Rule ⌄ {
                            id              string
                                            nullable: true
                            topicId         integer($int32)
                            text            string
                                            nullable: true
                            image           string
                                            nullable: true
                         }]
}
```

Рисунок 3.12 – Схема моделі Topic

Реалізація функціоналу для моделі «User» зображеної на рис. 3.13:

POST – ‘/api/User/login’ – авторизація користувача.

GET – ‘/api/User’ – отримання всіх користувачів.

POST – ‘/api/User’ – створення користувача. Також створюється його прогрес для кожного білету.

GET – ‘/api/User/{id}’ – отримання користувача за його id. А також отримання його друзів.

PUT – ‘/api/User/{id}’ – редагування користувача за його id.

DELETE – ‘/api/User/{id}’ – видалення користувача за його id. Також видаляється його прогрес.

```
User ⌄ {
    id                      string
                            nullable: true
    name                    string
                            nullable: true
    password                string
                            nullable: true
    image                   string
                            nullable: true
    confirmed               boolean
    progress                ⌄ [
                            nullable: true
                            CardProgress > {...}]
    sentFriendRequests      ⌄ [
                            nullable: true
                            Friends > {...}]
    receivedFriendRequests  ⌄ [
                            nullable: true
                            Friends > {...}]
}
```

Рисунок 3.13 – Схема моделі User

### 3.5. Опис розробки клієнтської частини

Клієнтська частина застосунку «Світлофор» реалізована за допомогою фреймворку Vue.js з використанням платформи Node.js. Вона складається з

компонентів і функціональних файлів написаних на мові JavaScript. Код проекту наведено у Додатку В.

### 3.5.1. Опис інтерфейсу

Веб-сайт «Світлофор» містить декілька сторінок, які побудовані за допомогою компонентів Vue. З кожної сторінки можливо потрапити на головну, а також на сторінки доступні у верхньому меню. Мапу сайту зображено на рис. 3.14. Далі буде наведено детальний опис кожної сторінки.

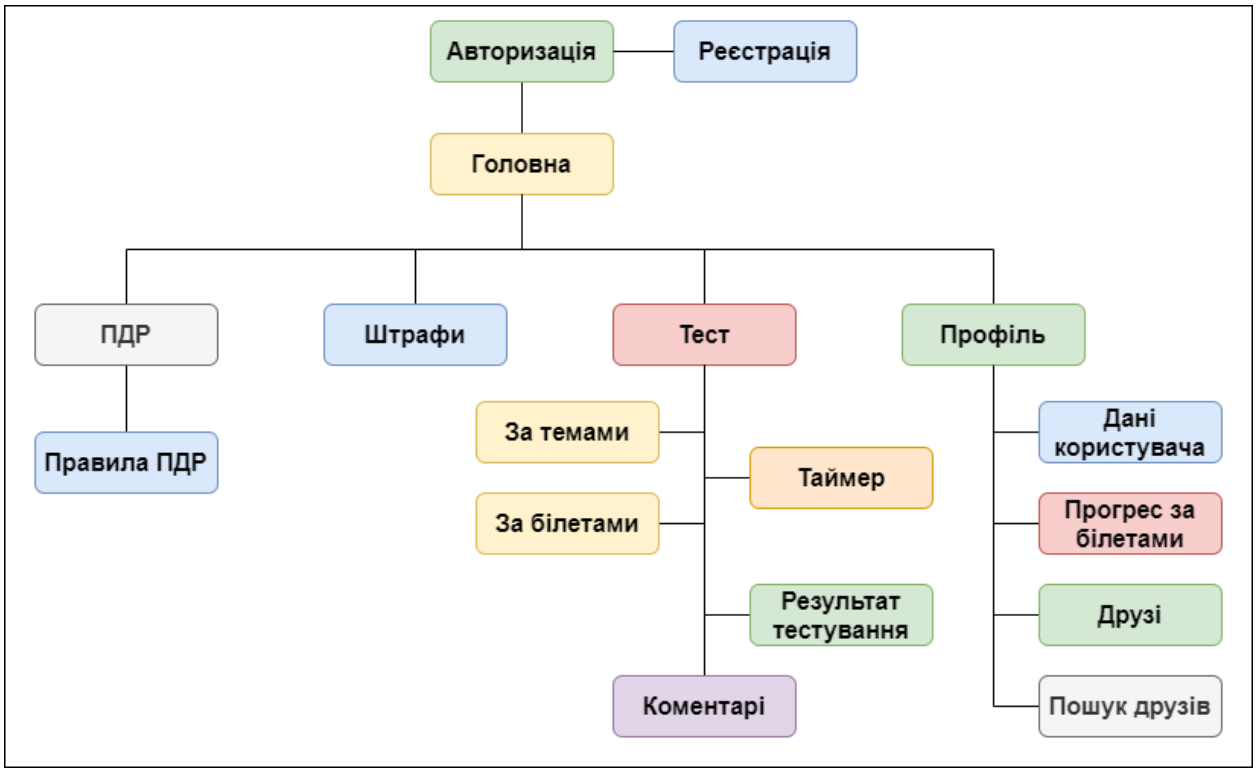


Рисунок 3.14 – Мапа сайту «Світлофор»

**Авторизація.** Сторінка, яка містить форму для входу в систему. Користувач має ввести свій email та пароль. Також на сторінці знаходиться посилання на сторінку реєстрації, якщо користувач хоче створити аккаунт.

**Реєстрація.** На сторінці знаходиться форма реєстрації користувача. Щоб створити аккаунт потрібно ввести email, ім’я, придумати пароль та підтвердити його. Після успішної реєстрації буде виведено повідомлення з привітанням.

**Головна сторінка.** Містить банер, який сповіщає, що сайт містить достовірний матеріал.

**Теми ПДР.** На сторінці знаходиться список тем з правил дорожнього руху.

**Правила ПДР.** Містить правила дорожнього руху, які належать певній темі. Правило складається з номеру, тексту правила та може містити картинку.

**Штрафи.** Сторінка, на якій заходиться таблиця із штрафами за порушення ПДР. Штраф складається з номеру статті закону України, формулювання закону та грошового стягнення.

**Тест.** Сторінка складається з декількох компонентів. Перший компонент – меню для фільтрації тестів. Користувач може обрати тему або білет, за яким хоче пройти тестування. Другий – тест. Він складається зі списку питань, які мають від 2 до 4 варіантів відповідей. Також тест може мати картинку. Знизу вікна закріплений таймер, який показує час, що залишився до кінця тестування. Третій компонент – блок коментарів до тесту. Користувач може як задати своє питання так і відповісти на вже існуючі коментарі. Коментар містить номер тесту, дату створення, текст, ім’я та фото користувача.

**Профіль.** На сторінці відображена основна інформація про користувача, а саме email, ім’я та фото. Також представлений прогрес за білетами. Він складається з номеру білету, кількості спроб проходження тестування, останнього результату тестування: кількість правильних і неправильних відповідей та процент правильних відповідей. Окрім цього сторінка містить список друзів користувача та їх прогрес. Опис друга складається з фото, імені, шкали прогресу, на якій відображено кількість білетів, що мають результат тестування більше ніж 90% і менше за нього. Якщо натиснути на клавішу «Додати», то відкриється вікно із списком користувачів, яких можна додати у друзі.

Кольори веб-сайту складаються з жовтого, темного сірого, світло-сірого та зеленого кольорів, оскільки вони найбільше нагадують світлофор, що є логотипом створеного веб-сайту.

### 3.5.2. Опис функціональної частини

Функціонал системи поділяється на спільний для всіх компонентів і знаходиться в окремих js файлах та на власний для кожного компонента. Серед спільного функціоналу можна виділити навігацію та зв'язок із сервером.

Для навігації між сторінками використовується маршрутизатор VueRouter. Усі шляхи і налаштування маршрутизатора прописані у файлі router/index.js, який розміщено у Додатку В. Нижче наведено список реалізованих шляхів:

| | | |
|---|---|---|
| authorization: '/' | tests: '/tests' | topic: '/topic/:topicId' |
| registration: '/registration' | fines: '/fines' | profile: '/profile' |
| main: '/main' | pdr: '/pdr' | |

Для зв'язку з сервером сайт використовує HTTP-клієнта Axios. Щоб отримати дані необхідно бути авторизованим користувачем, а саме токен сесії користувача має бути валідним. У даному проекті використовуються такі типи запитів: GET, POST, PUT. Далі наведено перелік запитів, що реалізує система.

– login (email, password) – авторизація користувача. Серверу передається email і пароль, а у відповідь, якщо такий користувач існує, надсилається email користувача і токен для сесії.

– registration (email, name, password) – реєстрація користувача в системі. Серверу надсилаються дані нового користувача. У відповідь приходить email користувача і токен для сесії.

– getFines – отримання усіх штрафів за порушення ПДР. У відповідь приходить список штрафів.

– getTopicsList – отримання усіх тем з ПДР. У відповідь приходить список тем.

– getTopic (id) – отримання інформації теми з номером id. У відповідь приходить об’єкт з даними про тему, якщо тема з таким id існує.

– getCardList – отримання усіх білетів з ПДР. У відповідь приходить список білетів.

– getTestList – отримання усіх тестів. У відповідь приходить список тестів.

– getUsersList – отримання усіх користувачів системи. У відповідь приходить список користувачів.

– getUser – отримати особисті дані поточного авторизованого користувача.

– updateProgress (progress) – оновлення прогресу користувача за білетом. Спочатку отримується прогрес за id. Старі дані змінюються на нові з об’єкту progress. Серверу відправляється оновлений прогрес користувача.

– addFriend (id) – відправка запиту на дружбу від поточного користувача користувачу з емейлом id. Створюється об’єкт friends і відправляється на сервер. У відповідь повертається об’єкт friends, якщо створення було успішним.

– addComment (comment) – додати до системи коментар. На сервер відправляється об'єкт comment. У відповідь повертається об’єкт comment, якщо створення було успішним.

Кожна сторінка має свій власний функціонал. Перелік реалізованих функцій наведено нижче.

Авторизація:

– авторизація користувача;

– перехід на сторінку реєстрації.

Реєстрація:

– зареєструвати користувача.

Теми ПДР:

– відкриття певної теми ПДР.

Тест:

– відкриття тесту за темою;

– відкриття тесту за білетом;

– контроль часу (таймер);

– перевірка тестування;

– збереження прогресу;

– додати коментар до тесту;

– відповісти на існуючий коментар.

Профіль:

– додати друга.

### 3.5.3. Використання HTML5 Web Workers

Для підвищення ефективності системи було розроблено функціонал, який використовує потоки, а саме HTML5 Web Workers. Основний ресурс для роботи з потоками – «workers/index.js». Код файлу наведено у Додатку В. Веб-сайт використовує потоки у трьох випадках:

**Таймер.** Worker знаходиться у файлі «workers/timerWorker.js» і приймає на вхід інтервал часу тестування. На кожному кроці повертає нове значення таймера.

**Фільтрація тестів.** Worker знаходиться у файлі «workers/filterWorker.js» і приймає на вхід список тестів, критерій фільтрації та його значення. На кожному кроці повертає тест з відповідними даними.

**Перевірка результатів тестування.** Worker знаходиться у файлі «workers/checkTestWorker.js» і приймає на вхід список тестів та список відповідей користувача. На кожному кроці повертається результат перевірки.

Як приклад використання HTML5 Web Workers розглянемо випадок з відображенням таймера. Таймер викликається на сторінці з тестами як тільки розпочинається тестування. Для створення взаємодії з потоками у файлі компонента використовується ресурс «workers/index.js», доступ до якого можна отримати через змінну worker. Він містить насупні функції: startTimerWorker(), getTimerWorker(), stopTimerWorker(), sendTimerWorker().

Функція **startTimerWorker()**, зображена на рис. 3.15, відповідає за створення та ініціалізацію потоку з таймером. Якщо таймер вже існує, то він зупиняється. Створення потоку відбувається через виклик конструктора Worker(), до якого передається скрипт, в якому описаний порядок дій потоку.

```
let timerWorker;

const startTimerWorker = () => {
    if(typeof(timerWorker) != "undefined")
        stopTimerWorker();

    timerWorker = new Worker( stringUrl: './timerWorker.js',  options: { type: 'module' });
}
```

Рисунок 3.15 – Функція startTimerWorker() у файлі «workers/index.js»

Функція **getTimerWorker()**, зображена на рис. 3.16, відповідає за доступ до потоку таймеру.

```
10  const getTimerWorker = () => {
11      return timerWorker
12  }
```

Рисунок 3.16 – Функція getTimerWorker() у файлі «workers/index.js»

Функція **stopTimerWorker()**, зображена на рис. 3.17, відповідає за завершення роботи потоку. Для цього використовується метод terminate().

```
19  const stopTimerWorker = () => {
20      if(typeof(timerWorker) != "undefined"){
21          timerWorker.terminate();
22          timerWorker = undefined;
23      }
24  }
```

Рисунок 3.17 – Функція stopTimerWorker() у файлі «workers/index.js»

Функція **sendTimerWorker()**, зображена на рис. 3.18, відповідає за відправку повідомлень потоку таймера через метод postMessage().

```
14  const sendTimer = message =>
15      timerWorker.postMessage({
16          message
17      })
```

Рисунок 3.18 – Функція sendTimerWorker() у файлі «workers/index.js»

Потік таймера приймає повідомлення та виконує свою роботу використовуючи обробник подій onmessage. Код наведений на рис. 3.19 ілюструє цю взаємодію.

```
onmessage = (event : MessageEvent ) => {
    let timeLimit = event.data.message.limit;

    setInterval( handler: () => {
        timeLimit--;
        postMessage(timeLimit);
    }, timeout: 1000)
}
```

Рисунок 3.19 – Метод onmessage у файлі «workers/timerWorker.js»

У наведоному коді, рис. 3.19, відлік часу здійснюється за допомогою методу setInterval() і на кожному кроці основному потоку відправляється значення таймера через функцію postMessage().

Основний потік компонента з тестами приймає результат роботи потоку таймера і перевіряє чи не закінчився час відведенний на тестування. Код обробника події наведено на рис. 3.20.

```
worker.getTimerWorker().onmessage = (event) => {
    this.timer = event.data;
    if (this.timer % this.timeLimit === 0)
        this.checkAnswers();
}
```

Рисунок 3.20 – Метод onmessage у компонента «Test-Page.vue»

## 4. Тестування програми і результати її виконання

Нижче наведено опис кінцевого варіанту користувацького інтерфейсу та його тестування.

Щоб потрапити на веб-сайт «Світлофор», потрібно в стрічку адреси ввести посилання «https://svitlofor.herokuapp.com/». Завантажуючи сайт користувач потрапляє на сторінку авторизації, яка представлена на рис. 4.1.

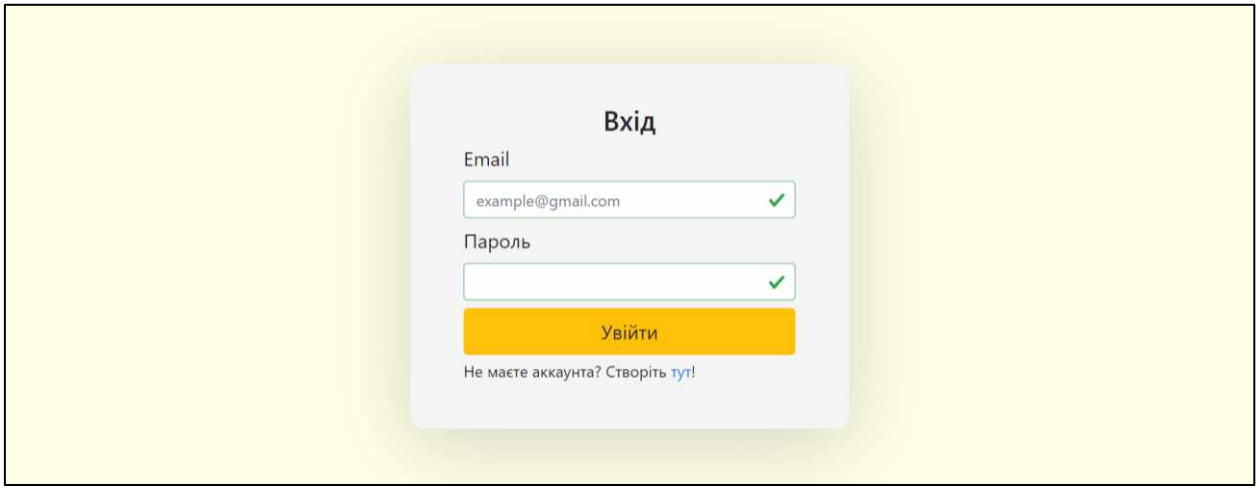


Рисунок 4.1 – Форма авторизації користувача

Для авторизації потрібно ввести email та пароль. Система перевіряє валідність введених даних. Якщо користувач ввів дані неправильно, видається повідомлення про помилку зображену на рис. 4.2.

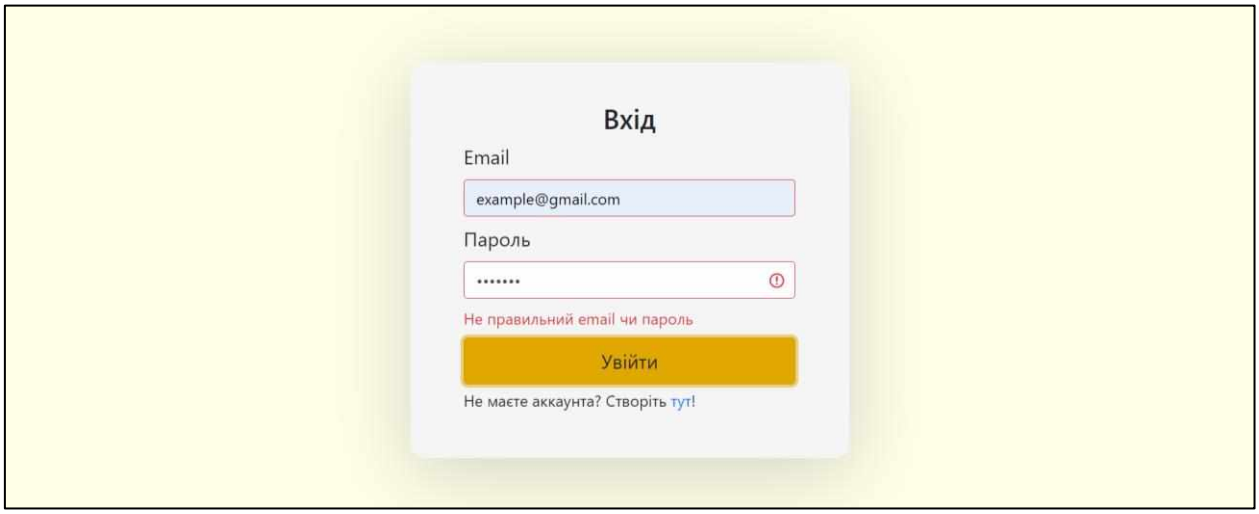


Рисунок 4.2 – Повідомлення про помилку в авторизації користувача

Якщо дані вірні і такий користувач існує у системі, то відбувається перехід на головну сторінку веб-сайту, яка зображена на рис. 4.3. Також на цю сторінку можна потрапити натиснувши на логотип або назву веб-сайту.

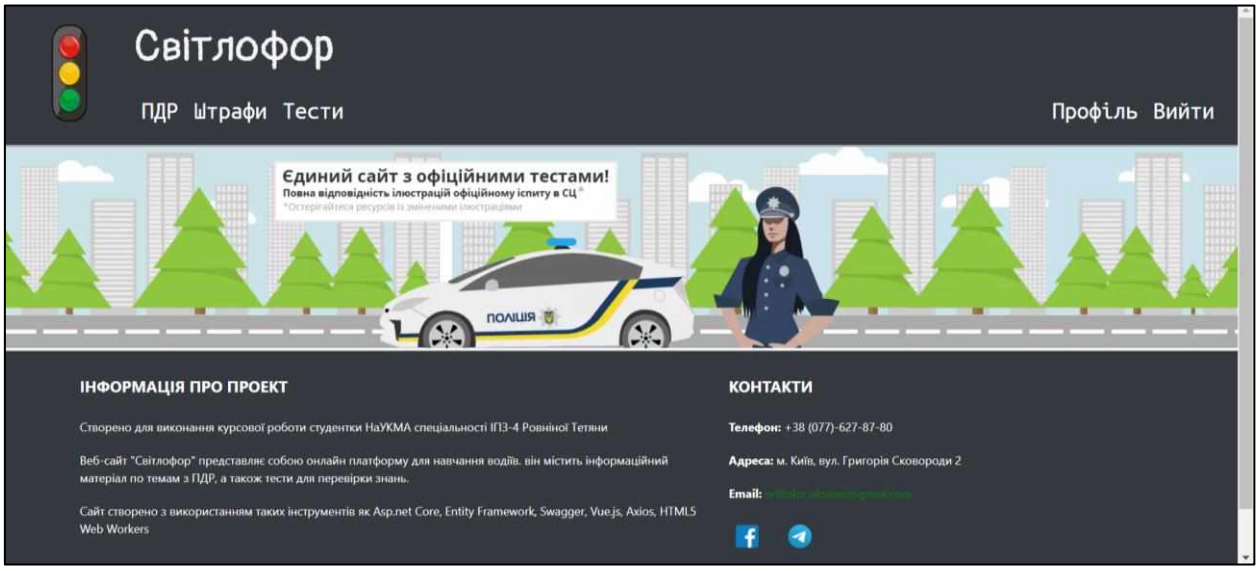


Рисунок 4.3 – Головна сторінка

Вгорі кожної сторінки розташоване меню. А знизу – футер з інформацією про веб-сайт. Наявна можливість зв’язатись з адміністрацією сайту через пошту або соціальні мережі.

Обравши в меню «ПДР», користувач потрапить на сторінку зображену на рис.  4.4, яка містить теми з правил дорожнього руху.

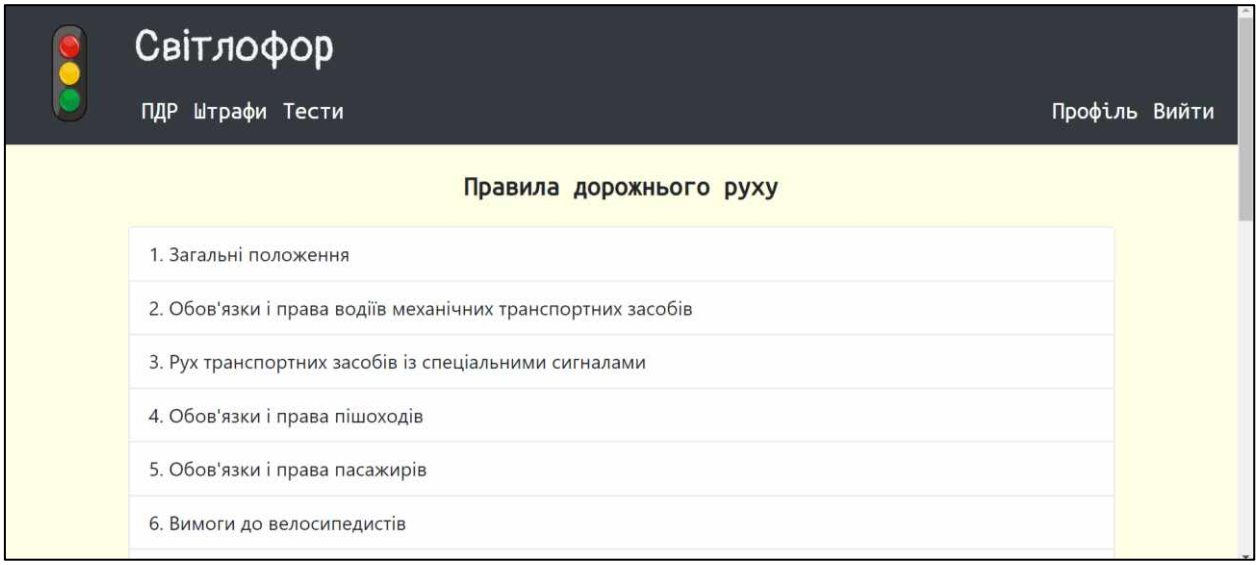


Рисунок 4.4 – Сторінка з темами ПДР

Натиснувши на одну з запропонованих тем, користувачу будуть доступні ПДР, які стосуються обраної теми. Інтерфейс сторінки представлено на рис. 4.5.

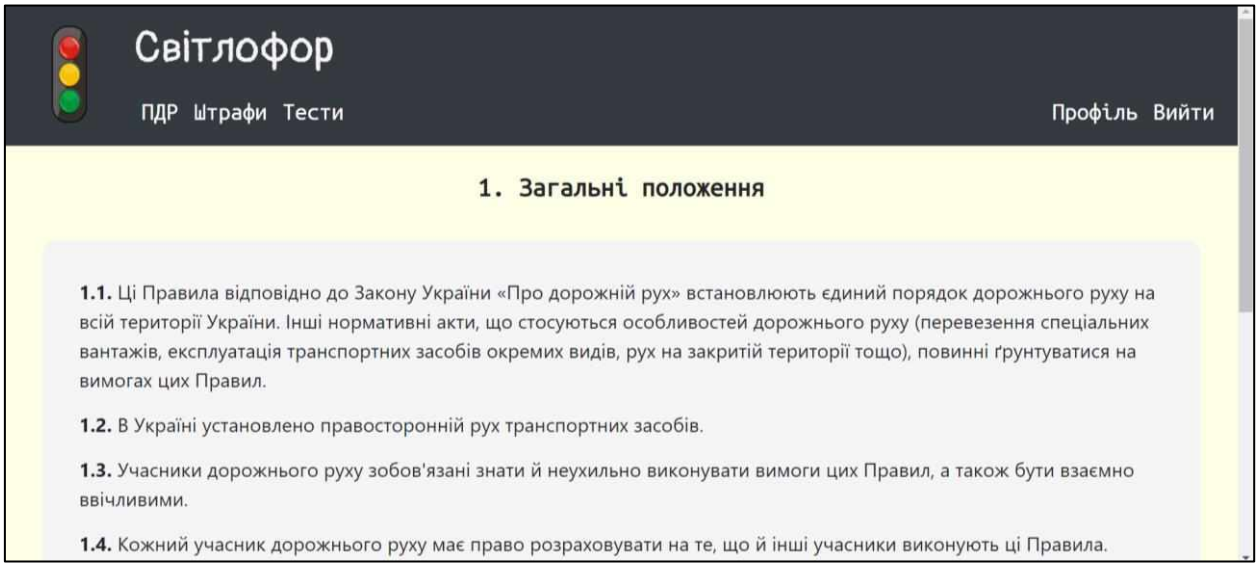


Рисунок 4.5 – Сторінка з правилами певної теми

На сторінці «Штрафи», яка зображена на рис. 4.6, представлена таблиця, яка містить номер статті, текст закону України та грошовий штраф.

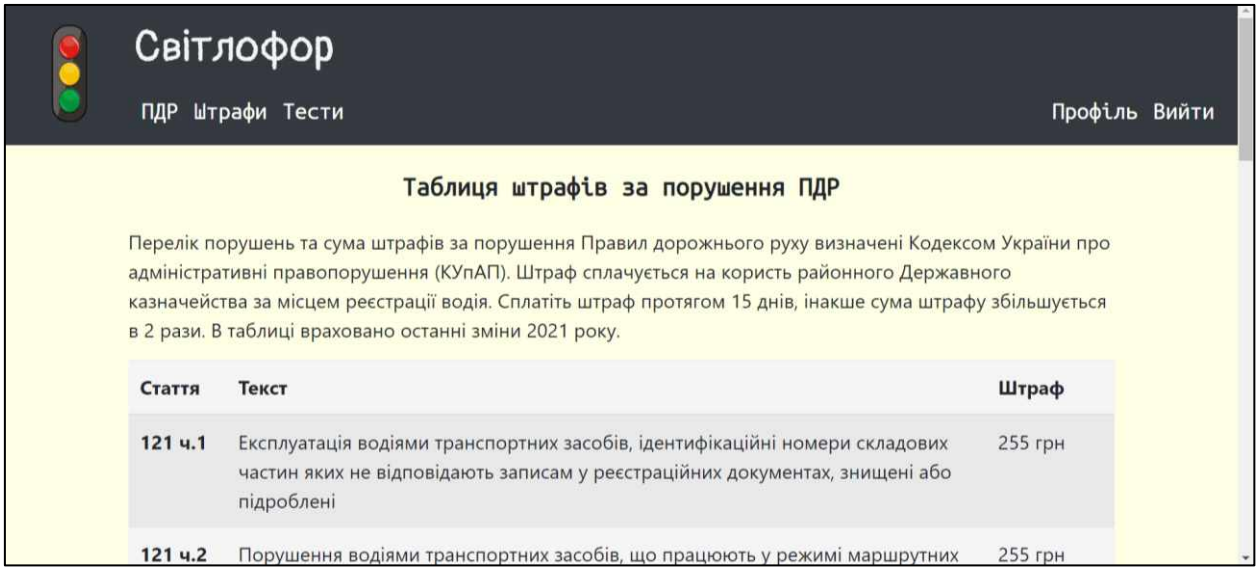


Рисунок 4.6 – Сторінка із штрафами за порушення ПДР

Якщо обрати категорію «Тести», користувач потрапить на сторінку тестування зображену на рис. 4.7, на якій зможе обрати тест за темами або за білетами.

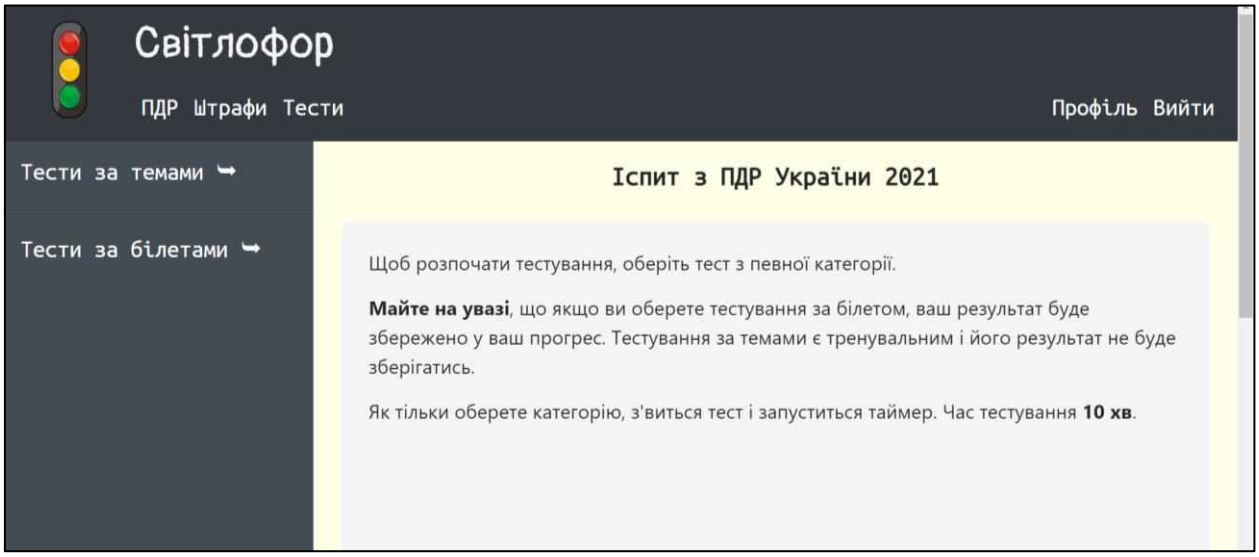


Рисунок 4.7 – Пуста сторінка з тестами

Після того, як користувач обере категорію, розпочиннеться тестування і одразу з’являться тестові питання. Інтерфейс сторінки з тестом зображений на рис. 4.8.

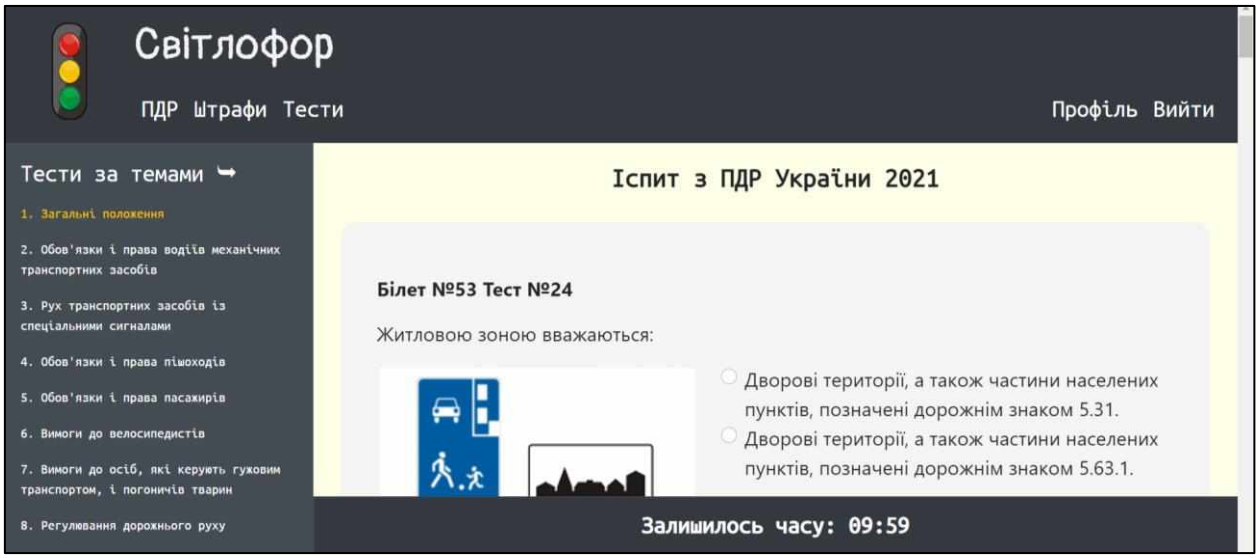


Рисунок 4.8 – Пуста сторінка з тестами

Після того, як користувач натисне на кнопку «Перевірити» чи час тестування дійде до 0, тест завершиться і користувач побачить свій результат. У результаті присутня інформація про те, скільки було правильних і

неправильних відповідей, час витрачений на тестування та процент правильних відповідей від загальної кількості питань. На рис. 4.9 зображено результат, коли процент правильних відповідей користувача менший за 90%, а на рис. 4.10 – результат, коли процент більший за 90%. Також на сторінці тесту правильні відповіді будуть виділені зеленим кольором, а неправильні червоним.

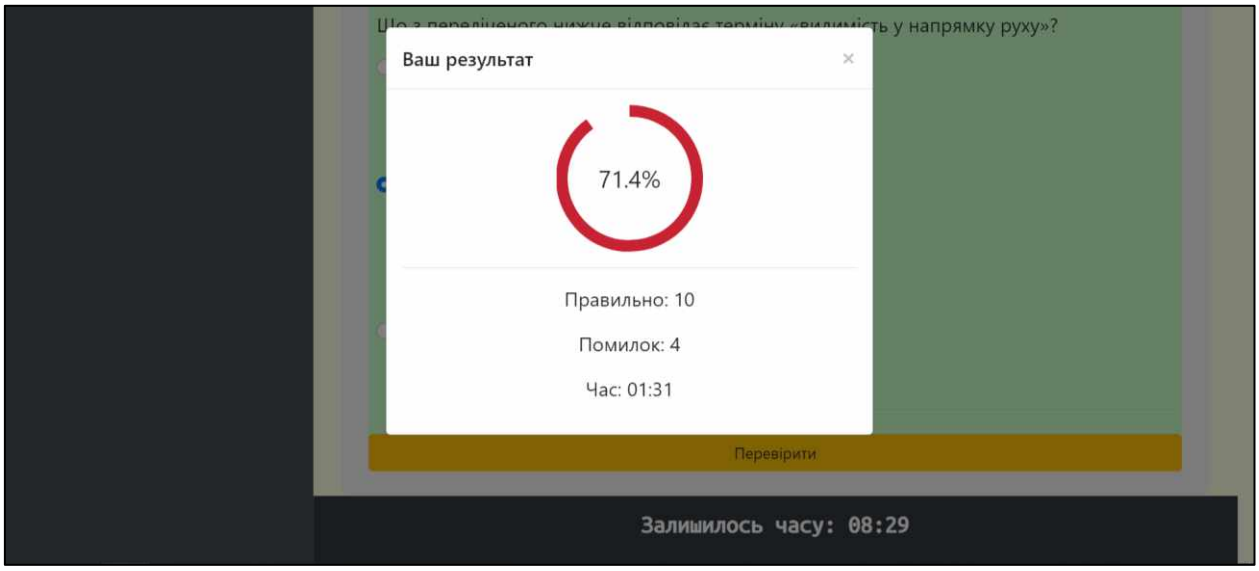


Рисунок 4.9 – Поганий результат тестування

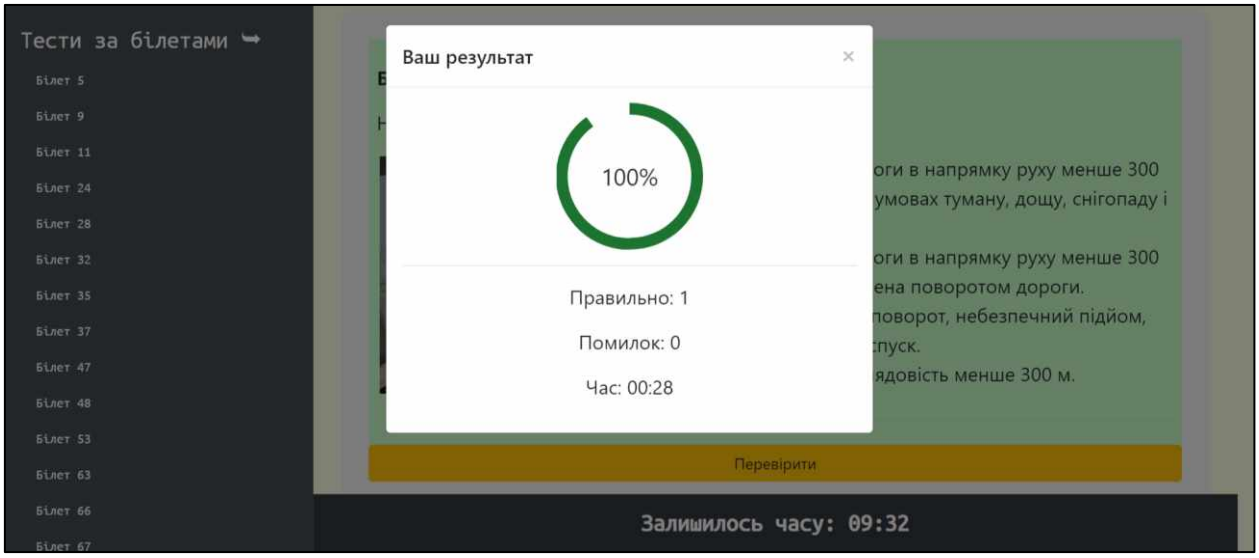


Рисунок 4.10 – Гарний результат тестування

Також на даній сторінці представленні коментарі до тесту. Блок коментарів зображено на рис. 4.11. Окрім тексту коментар містить інформацію про користувача, що його залишив, номер тесту, час і дату створення.

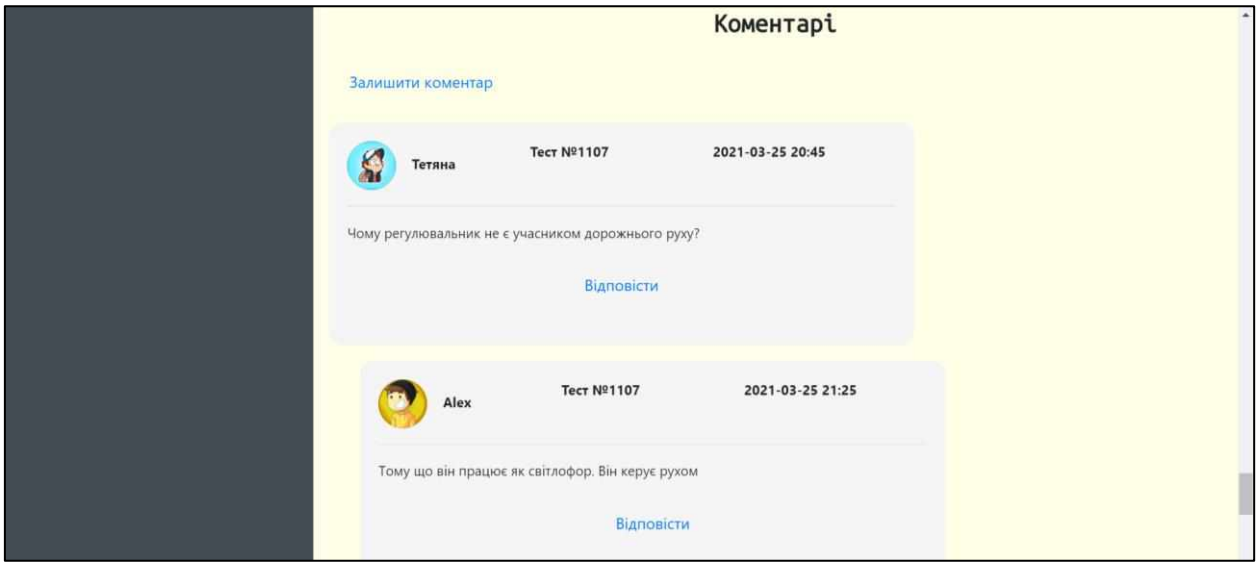


Рисунок 4.10 – Блок коментарів до тесту

Користувач може залишити свій коментар до тесту, використавши фурму, зображену на рис. 4.11.

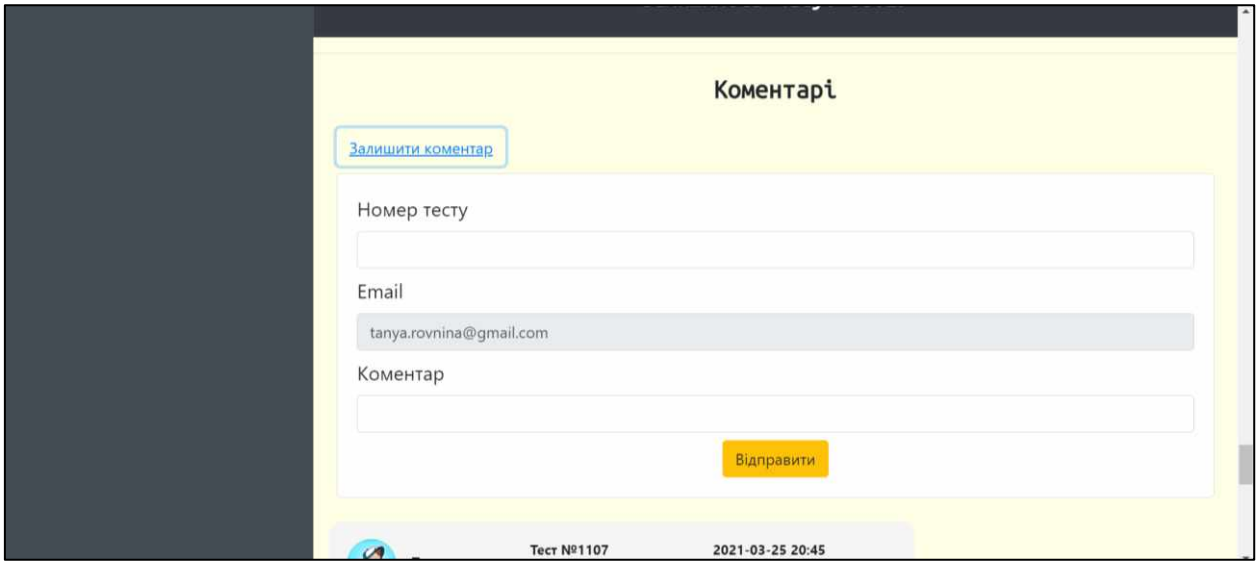


Рисунок 4.11 – Форма створення коментарю

Також можна відповісти на вже існуючий коментар, що зображено на рис. 4.12. Тоді поле з номером тесту буде заповнено автоматично.

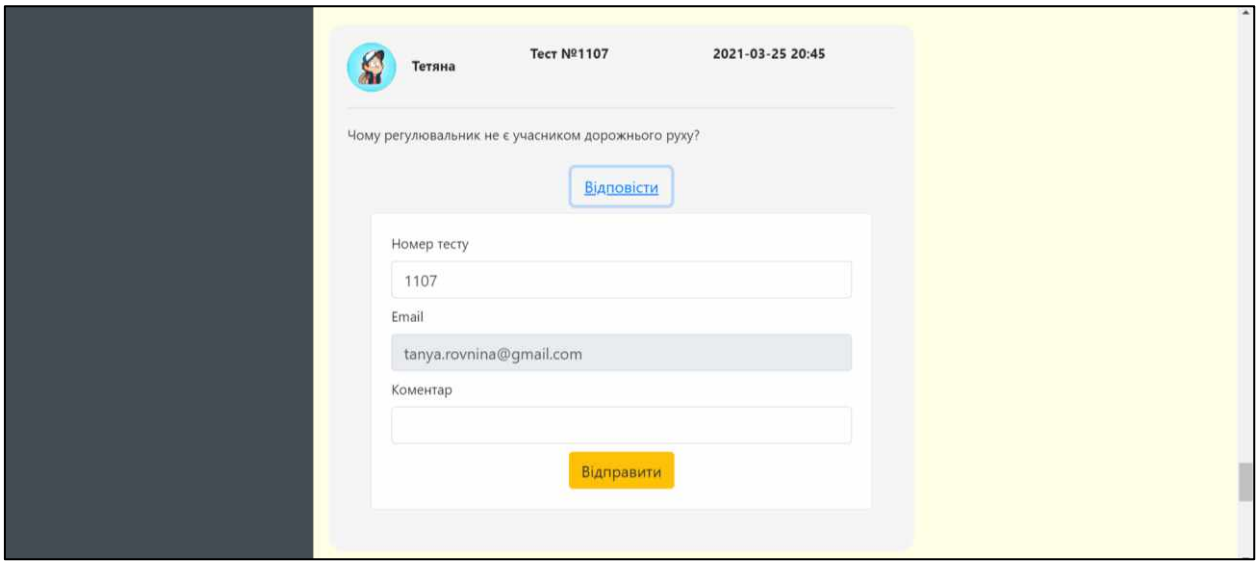


Рисунок 4.12 – Форма відповіді на коментар

Обравши в меню «Профіль», користувач потрапить на сторінку з інформацією про себе. Також на цій сторінці міститься перелік друзів користувача та їх прогрес. Прогрес друга складається з кількості білетів, результат яких має більше 90% правильних відповідей (зелений колір), та менше 90% (червоний). Інтерфейс сторінки зображено на рис. 4.13.

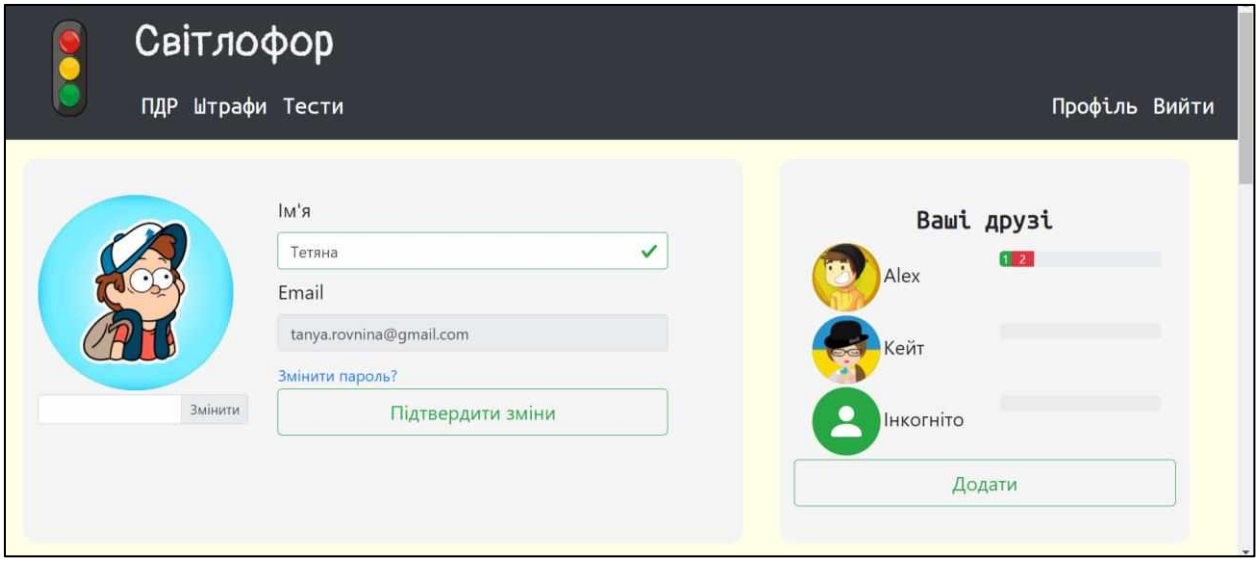


Рисунок 4.13 – Профіль користувача

Знизу сторінки знаходиться прогрес користувача за білетами, який зображено на рис. 4.14. Прогрес містить номер білету, кількість спроб його

складання, процент правильних відповідей від загальної кількості, кількість правильних та неправильних відповідей.

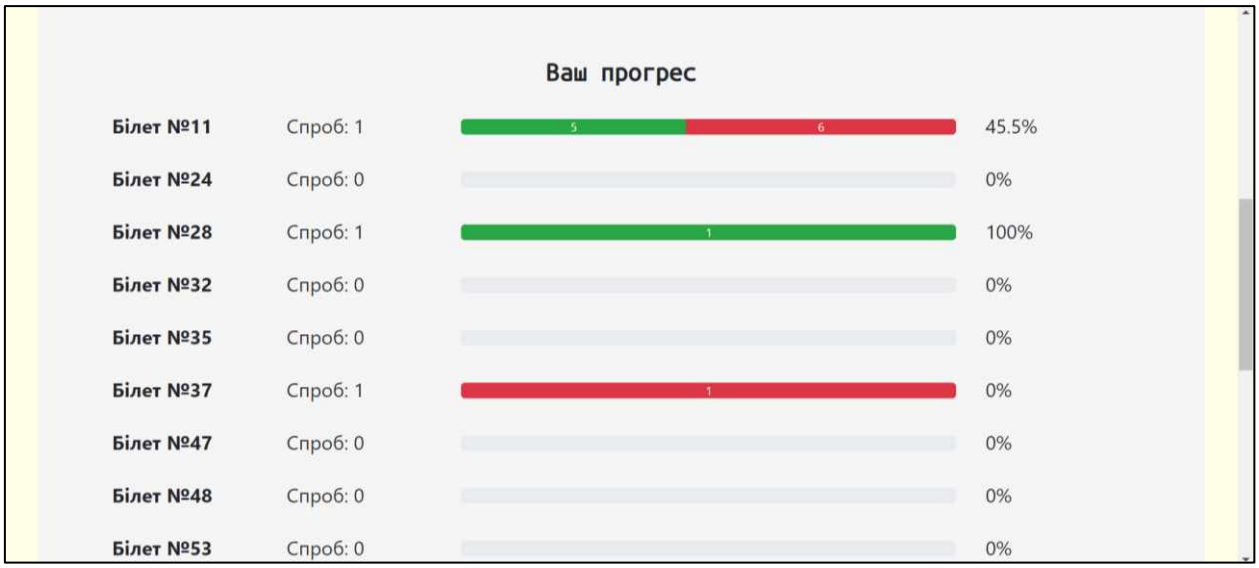


Рисунок 4.14 – Прогрес користувача за білетами

Реалізована можливість додати друга, натиснувши на кнопку «Додати» у блоці друзів. Вікно пошуку друзів зображено на рис. 4.15.

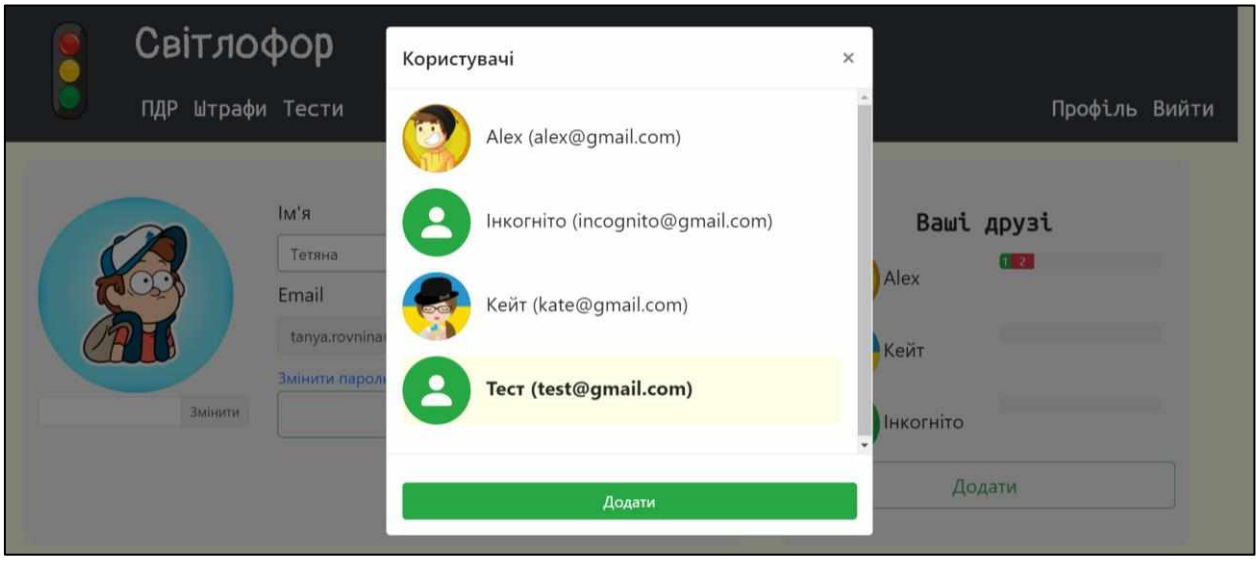


Рисунок 4.13 – Вікно пошуку друзів

Якщо користувача не існує в системі, він може створити аккаунт, натиснувши на посилання «Створіть тут» внизу форми авторизації, що зображена на рис. 4.1. Форма реєстрації користувача зображена на рис. 4.14.

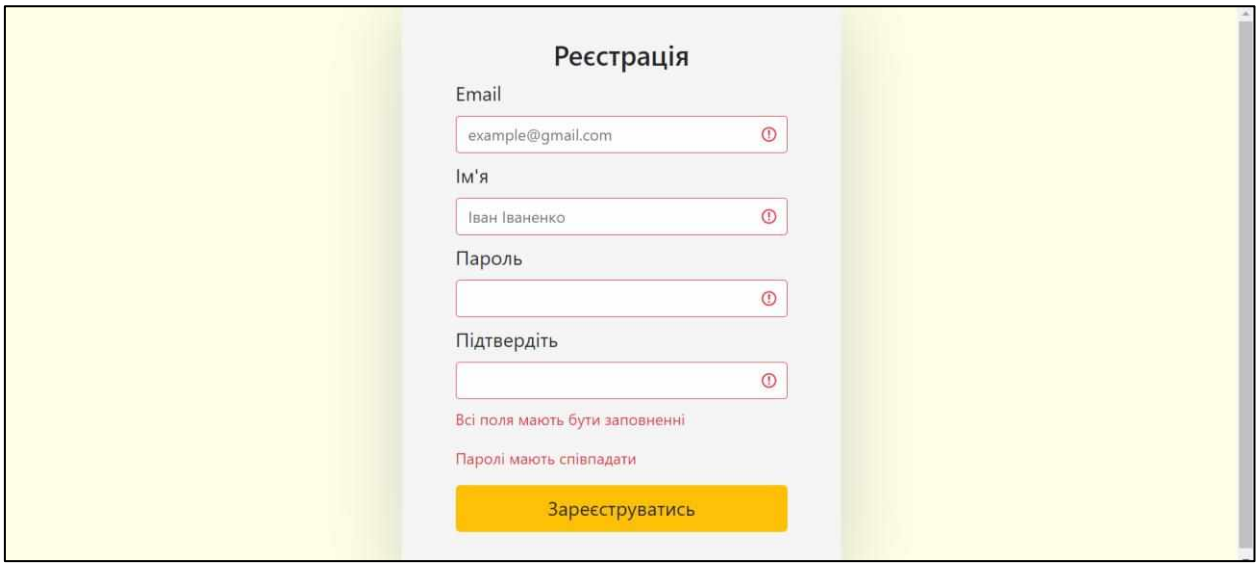


Рисунок 4.14 – Форма реєстрації користувача

Для реєстрації необхідно заповнити всі поля та підтвердити пароль. Валідна форма зображена на рис. 4.15.

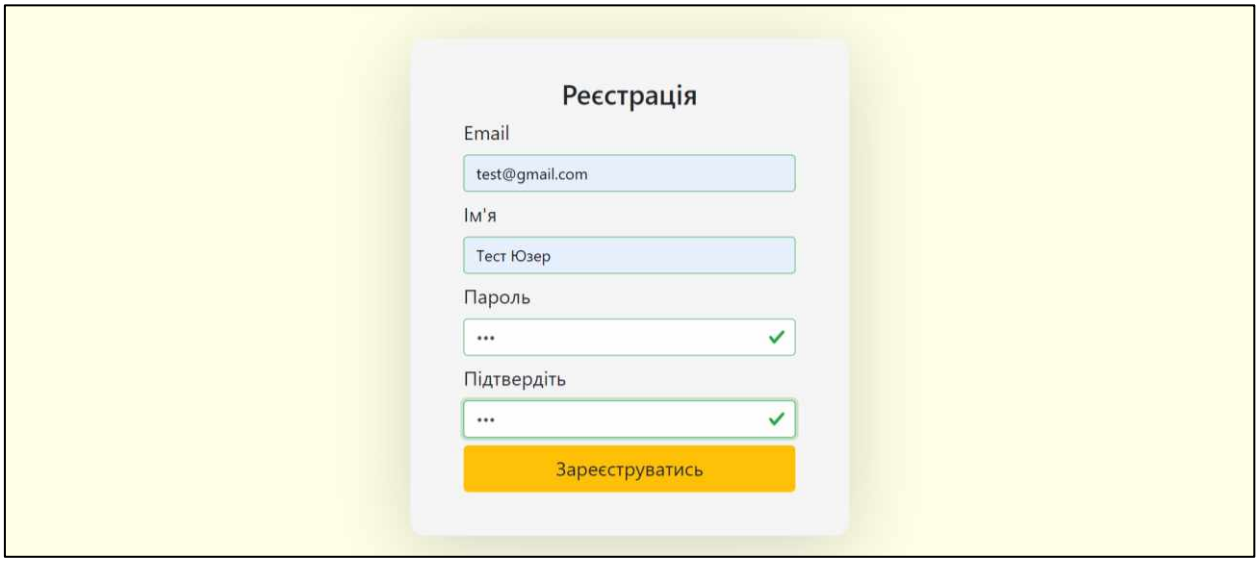


Рисунок 4.15 – Валідна форма реєстрації

Після успішної реєстрації користувач побачить повідомлення, зображене на рис. 4.16, і потрапить на сторінку свого профілю.

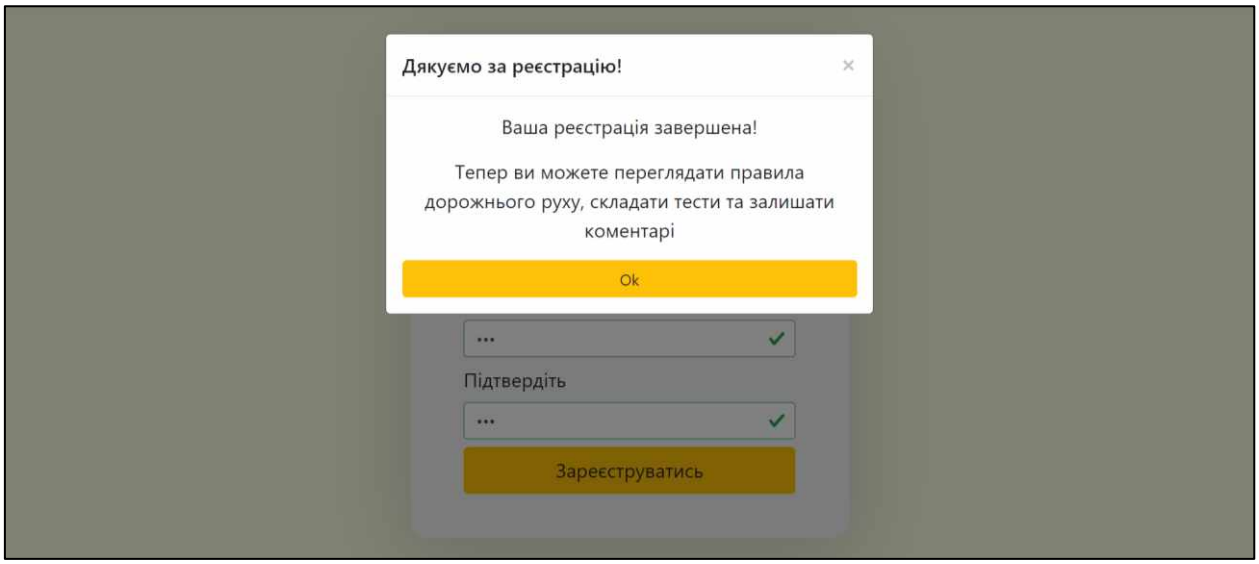


Рисунок 4.16 – Повідомлення про успішну реєстрацію

Обравши в меню пункт «Вийти», користувач повернеться на сторінку авторизації.

## Висновки

При виконанні курсової роботи була створена інформаційна веб-система для підготовки водіїв, яка використовує засоби багатопоточного програмування на стороні клієнта, а саме HTML5 Web Workers.

Дизайн сайту відповідає вимогам мінімальності та містить контент українською мовою. Веб-сайт розроблено для авторизованих користувачів і надає матеріали з ПДР. Користувач може перевірити свої знання за допомогою тестування за різними категоріями та задати питання до тесту, не виходячи з дому. Також система надає користувачу можливість зберігати свій прогрес та переглядати прогрес інших користувачів.

Веб-сайт використовує технологію HTML5 Web Workers для перевірки відповідей користувача на тестові питання, фільтрації тестів за категоріями та відображення таймеру під час тестування.

Сервер застосування розроблений з використанням технології ASP.NET Core. Клієнтська частина створена з використанням фреймворку Vue.js і платформи Node.js. В якості СКБД був обраний MySQL. Для розгортання системи був задіяний Docker. Сервер та клієнт системи розміщені у хмарному середовищі Heroku, а база даних на веб-сервісі AWS.

Застосування може бути покращено наступними шляхами:

- розробка системи для адміністрації сайту;
- додавання можливості редагування профілю користувача;
- додавання підтвердження реєстрації користувача;
- реалізація можливості відновлення паролю.

## Список використаних джерел


1. Как работает JavaScript: создание блоков Web Workers + 5 примеров, как их использовать [Електронний ресурс]. – Режим доступу: URL: https://www.html5rocks.com/ru/tutorials/workers/basics/.– Назва з екрану.

2. Основные сведения об объектах Web Worker [Електронний ресурс]. – Режим доступу: URL: https://webformyself.com/kak-rabotaet-javascript-sozdanie-blokov-web-workers-5-primerov-kak-ix-ispolzovat/.– Назва з екрану.

3. Матеріали з лекцій курсу «Реляційні бази даних» – основні теоретичні відомості. [Електронний ресурс]. – Режим доступу: URL: https://distedu.ukma.edu.ua/course/view.php?id=50. – Назва з екрану.

4. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems [Електронний ресурс]. – Режим доступу: URL: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. – Назва з екрану.

5. «Metanit.com». Веб-сайт з навчальним матеріалом по .NET [Електронний ресурс]. – Режим доступу: URL: https://metanit.com/.– Назва з екрану.

6. Документація бібліотеки AutoMapper [Електронний ресурс]. – Режим доступу: URL: https://docs.automapper.org/en/stable/Getting-started.html/.– Назва з екрану.

7. Inversion of Control Tutorials [Електронний ресурс]. – Режим доступу: URL: https://www.tutorialsteacher.com/ioc/introduction/.– Назва з екрану.

8. «Vue.js». Документація фреймворку Vue.js [Електронний ресурс]. – Режим доступу: URL: https://vuejs.org/.– Назва з екрану.

9. Документація інструменту Axios [Електронний ресурс]. – Режим доступу: URL: https://zetcode.com/javascript/axios/.– Назва з екрану.

10. Веб-сайт бібліотеки «Bootstrap» [Електронний ресурс]. – Режим доступу: URL: https://getbootstrap.com/. – Назва з екрану.

11. Domain Driven Design [Електронний ресурс]. – Режим доступу: URL: https://docs.abp.io/en/abp/latest/Domain-Driven-Design/.– Назва з екрану.

12. Документація корпорації Microsoft [Електронний ресурс]. – Режим доступу: URL: https://docs.microsoft.com/.– Назва з екрану.

13. Документація бібліотеки Autofac [Електронний ресурс]. – Режим доступу: URL: https://autofaccn.readthedocs.io/en/latest/getting-started/.– Назва з екрану.

14. «Vodiy.ua». Веб-сайт для підготовки водіїв [Електронний ресурс]. – Режим доступу: URL: https://vodiy.ua/.– Назва з екрану.

15. «Greenway». Веб-сайт для підготовки водіїв [Електронний ресурс]. – Режим доступу: URL: https://green-way.com.ua/.– Назва з екрану.

# Додаток А. Код проекту Svitlofor.Data

## А.1. Entities

### А.1.1. IEntity.cs

```
namespace Svitlofor.Data.Entities
{
    public interface IEntity<Tkey>
    {
        Tkey Id { get; set; }
    }
}
```

### А.1.2. CardEntity.cs

```
using System.Collections.Generic;
namespace Svitlofor.Data.Entities
{
   public class CardEntity : IEntity<int>
   {
     public int Id { get; set; }
     public string Notes { get; set; }
     public virtual ICollection<TestEntity> Tests { get; set; }
     public virtual ICollection<CardProgressEntity> CardProgress { get; set; }
   }
}
```

### А.1.3. CardProgressEntity.cs

```
namespace Svitlofor.Data.Entities
{
    public class CardProgressEntity : IEntity<string>
    {
        public string Id { get; set; }
        public string Email { get; set; }
        public int CardId { get; set; }
        public int Correct { get; set; }
        public int Wrong { get; set; }
        public int Attempts { get; set; }
        public double Progress { get; set; }
    }
}
```

### А.1.4. CommentEntity.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;

namespace Svitlofor.Data.Entities
```

```
{
    public class CommentEntity : IEntity<int>
    {
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public string Text { get; set; }
        public int TestId { get; set; }
        public string Email { get; set; }
        public string Login { get; set; }

        public string Image { get; set; }
        public int? PreviousId { get; set; }
        public DateTime DateTime { get; set; }
        public virtual ICollection<CommentEntity> Next { get; set; }
    }
}
```

### A.1.5. FineEntity.cs

```
namespace Svitlofor.Data.Entities
{
    public class FineEntity : IEntity<string>
    {
        public string Id { get; set; }
        public string Text { get; set; }
        public int Price { get; set; }
    }
}
```

### A.1.6. FriendsEntity.cs

```
using System.ComponentModel.DataAnnotations.Schema;
namespace Svitlofor.Data.Entities
{
    public class FriendsEntity : IEntity<int>
    {
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public string RequestedById { get; set; }
        public string RequestedToId { get; set; }
        public bool IsAccepted { get; set; }
    }
}
```

### A.1.7. RuleEntity.cs

```
namespace Svitlofor.Data.Entities
{
    public class RuleEntity : IEntity<string>
    {
        public string Id { get; set; }
```

```
        public int TopicId { get; set; }
        public string Text { get; set; }
        public string Image { get; set; }
    }
}
```

### A.1.8. TestEntity.cs

```
using System.Collections.Generic;
namespace Svitlofor.Data.Entities
{
    public class TestEntity : IEntity<int>
    {
        public int Id { get; set; }
        public string Image { get; set; }
        public string Question { get; set; }
        public string Variant1 { get; set; }
        public string Variant2 { get; set; }
        public string Variant3 { get; set; }
        public string Variant4 { get; set; }
        public int Correct { get; set; }
        public int CardId { get; set; }
        public int TopicId { get; set; }
        public virtual ICollection<CommentEntity> Comments { get; set; }
    }
}
```

### A.1.9. TopicEntity.cs

```
using System.Collections.Generic;
namespace Svitlofor.Data.Entities
{
    public class TopicEntity : IEntity<int>
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public string Notes { get; set; }
        public virtual ICollection<RuleEntity> Rules { get; set; }
        public virtual ICollection<TestEntity> Tests { get; set; }

    }
}
```

### A.1.10. UserEntity.cs

```
using System.Collections.Generic;
namespace Svitlofor.Data.Entities
{
    public class UserEntity
    {
      public string Id { get; set; }
```

```
     public string Name { get; set; }
     public string Password { get; set; }
     public string Image { get; set; }
     public virtual ICollection<CardProgressEntity> Progress {get; set;}
     public virtual ICollection<FriendsEntity> SentFriendRequests {get; set;}
   public virtual ICollection<FriendsEntity> ReceivedFriendRequests {get; set;}
     public virtual ICollection<CommentEntity> Comments { get; set; }
    }
}
```

## A.2. EntityConfiguration

### A.2.1. CardConfiguration.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using Svitlofor.Data.Entities;

namespace Svitlofor.Data.EntityConfiguration
{
    public class CardConfiguration : IEntityTypeConfiguration<CardEntity>
    {
        public void Configure(EntityTypeBuilder<CardEntity> builder)
        {
            builder.HasMany(x    =>    x.Tests).WithOne().HasForeignKey(x    =>
x.CardId);
            builder.HasMany(x  =>  x.CardProgress).WithOne().HasForeignKey(x  =>
x.CardId);
        }
    }
}
```

### A.2.2. CommentConfiguration.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using Svitlofor.Data.Entities;

namespace Svitlofor.Data.EntityConfiguration
{
    public              class              CommentConfiguration              :
IEntityTypeConfiguration<CommentEntity>
    {
        public void Configure(EntityTypeBuilder<CommentEntity> builder)
        {
            builder.HasMany(x    =>    x.Next).WithOne().HasForeignKey(x    =>
x.PreviousId);
        }
    }
}
```

### A.2.3. TestConfiguration.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using Svitlofor.Data.Entities;

namespace Svitlofor.Data.EntityConfiguration
{
    public class TestConfiguration : IEntityTypeConfiguration<TestEntity>
    {
        public void Configure(EntityTypeBuilder<TestEntity> builder)
        {
            builder.HasMany(x   =>   x.Comments).WithOne().HasForeignKey(x   =>
x.TestId);
        }
    }
}
```

### A.2.4. TopicConfiguration.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using Svitlofor.Data.Entities;

namespace Svitlofor.Data.EntityConfiguration
{
    public class TopicConfiguration : IEntityTypeConfiguration<TopicEntity>
    {
        public void Configure(EntityTypeBuilder<TopicEntity> builder)
        {
            builder.HasMany(x    =>    x.Tests).WithOne().HasForeignKey(x    =>
x.TopicId);
            builder.HasMany(x    =>    x.Rules).WithOne().HasForeignKey(x    =>
x.TopicId);
        }
    }
}
```

### A.2.5. UserConfiguration.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using Svitlofor.Data.Entities;

namespace Svitlofor.Data.EntityConfiguration
{
    public class UserConfiguration : IEntityTypeConfiguration<UserEntity>
    {
        public void Configure(EntityTypeBuilder<UserEntity> builder)
        {
            builder.HasMany(x   =>   x.Comments).WithOne().HasForeignKey(x   =>
x.Email);
```

```
            builder.HasMany(x   =>   x.Progress).WithOne().HasForeignKey(x   =>
x.Email);
            builder.HasMany(x                                                   =>
x.ReceivedFriendRequests).WithOne().HasForeignKey(x => x.RequestedToId);
            builder.HasMany(x                                                   =>
x.SentFriendRequests).WithOne().HasForeignKey(x => x.RequestedById);
        }
    }
}
```

## A.3. Repositories

### A.3.1. IRepository.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Svitlofor.Data.Repositories
{
    public interface IRepository<TEntity, TKey>
        where TEntity : class
    {
        Task<TEntity> Add(TEntity item);
        Task<List<TEntity>> GetList();
        Task<TEntity> GetById(TKey id);
        Task Delete(TEntity item);
        Task Update(TEntity item);
    }
}
```

### A.3.2. EfRepository.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;

namespace Svitlofor.Data.Repositories
{
    public class EfRepository<TEntity, TKey> : IRepository<TEntity, TKey>
        where TEntity : class
    {
        protected readonly DbContext _context;

        public EfRepository(DbContext context)
        {
            _context = context;
        }

        public virtual async Task<List<TEntity>> GetList()
        {
            return await _context.Set<TEntity>().ToListAsync();
        }
```

```
        public virtual async Task<TEntity> GetById(TKey id)
        {
            return await _context.Set<TEntity>().FindAsync(id);
        }

        public virtual async Task<TEntity> Add(TEntity item)
        {
            _context.Set<TEntity>().Add(item);
            await SaveChanges();
            return item;
        }

       public virtual async Task Update(TEntity item)
        {
            _context.Entry(item).State = EntityState.Modified;

            await SaveChanges();
        }

        public virtual async Task Delete(TEntity item)
        {
            _context.Set<TEntity>().Remove(item);

            await SaveChanges();
        }

        public virtual async Task SaveChanges()
        {
            await _context.SaveChangesAsync();
        }
    }
}
```

### A.3.3. CardProgressRepository.cs

```
using Svitlofor.Data.Entities;
namespace Svitlofor.Data.Repositories
{
    public interface ICardProgressRepository : IRepository<CardProgressEntity,
string>
    {
    }

    public class CardProgressRepository : EfRepository<CardProgressEntity,
string>, ICardProgressRepository
    {
        public CardProgressRepository(SvitloforDbContext context) :
base(context)
        {
        }
```

```
    }
}
```

### A.3.4. CardRepository.cs

```
using Svitlofor.Data.Entities;
namespace Svitlofor.Data.Repositories
{
    public interface ICardRepository : IRepository<CardEntity, int>
    {
    }

    public class CardRepository : EfRepository<CardEntity, int>,
ICardRepository
    {
        public CardRepository(SvitloforDbContext context) : base(context)
        {
        }
    }
}
```

### A.3.5. CommentRepository.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using Svitlofor.Data.Entities;

namespace Svitlofor.Data.Repositories
{
    public interface ICommentRepository : IRepository<CommentEntity, int>
    {
    }

    public class CommentRepository : EfRepository<CommentEntity, int>,
ICommentRepository
    {
        public CommentRepository(SvitloforDbContext context) : base(context)
        {
        }

        public override async Task<List<CommentEntity>> GetList()
        {
            var comments = await _context.Set<CommentEntity>().ToListAsync();
            foreach (var comment in comments)
            {
                var user = await GetUser(comment.Email);
                if (user != null)
                {
                    comment.Login = user.Name;
                    comment.Image = user.Image;
```

```
                }

            }

            return comments;
        }

        public override async Task<CommentEntity> GetById(int id)
        {
            var comment = await _context.Set<CommentEntity>().FindAsync(id);
            if (comment != null)
            {
                var user = await GetUser(comment.Email);
                if (user != null)
                {
                    comment.Login = user.Name;
                    comment.Image = user.Image;
                }
            }

            return comment;
        }

        private async Task<UserEntity> GetUser(string id)
        {
            return await _context.Set<UserEntity>().FindAsync(id);
        }
    }
}
```

### A.3.6. FineRepository.cs

```
using Svitlofor.Data.Entities;
namespace Svitlofor.Data.Repositories
{
    public interface IFineRepository : IRepository<FineEntity, string>
    {
    }

    public class FineRepository : EfRepository<FineEntity, string>,
IFineRepository
    {
        public FineRepository(SvitloforDbContext context) : base(context)
        {
        }
    }
```

### A.3.7. FriendsRepository.cs

```
using Svitlofor.Data.Entities;
namespace Svitlofor.Data.Repositories
```

```
{
    public interface IFriendsRepository : IRepository<FriendsEntity, int>
    {
    }

    public class FriendsRepository : EfRepository<FriendsEntity, int>,
IFriendsRepository
    {
        public FriendsRepository(SvitloforDbContext context) : base(context)
        {
        }
    }
}
```

### A.3.8. RuleRepository.cs

```
using Svitlofor.Data.Entities;
namespace Svitlofor.Data.Repositories
{
    public interface IRuleRepository : IRepository<RuleEntity, string>
    {
    }

    public class RuleRepository : EfRepository<RuleEntity, string>,
IRuleRepository
    {
        public RuleRepository(SvitloforDbContext context) : base(context)
        {
        }
    }
}
```

### A.3.9. TestRepository.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using Svitlofor.Data.Entities;

namespace Svitlofor.Data.Repositories
{
    public interface ITestRepository : IRepository<TestEntity, int>
    {
    }

    public class TestRepository : EfRepository<TestEntity, int>,
ITestRepository
    {
        private ICommentRepository _commentRepository;
```

```
        public TestRepository(SvitloforDbContext context, ICommentRepository
commentRepository) : base(context)
        {
            _commentRepository = commentRepository;
        }

        public override async Task<List<TestEntity>> GetList()
        {
            var tests = await _context.Set<TestEntity>().ToListAsync();
            var comments = await _commentRepository.GetList();

            foreach (var test in tests)
            {
                test.Comments = await GetComments(test.Id, comments);
            }

            return tests;
        }

        public override async Task<TestEntity> GetById(int id)
        {
            var test = await _context.Set<TestEntity>().FindAsync(id);

            if (test != null)
                test.Comments = await GetComments(id, null);

            return test;
        }

        private   async   Task<List<CommentEntity>>   GetComments(int   id,
List<CommentEntity> comments)
        {
            if(comments == null)
                comments = await _commentRepository.GetList();

            var commentsList = new List<CommentEntity>();
            var testComments = comments.Where(com => com.TestId.Equals(id));
            commentsList.AddRange(testComments);

            return commentsList;
        }
    }
}
```

### A.3.10. TopicRepository.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using Svitlofor.Data.Entities;
```

```
namespace Svitlofor.Data.Repositories
{
    public interface ITopicRepository : IRepository<TopicEntity, int>
    {
    }

    public class TopicRepository : EfRepository<TopicEntity, int>,
ITopicRepository
    {
        public TopicRepository(SvitloforDbContext context) : base(context)
        {
        }

        public override async Task<List<TopicEntity>> GetList()
        {
            var topics = await _context.Set<TopicEntity>().ToListAsync();
            var rules = await _context.Set<RuleEntity>().ToListAsync();

            foreach (var topic in topics)
            {
                var rulesList = new List<RuleEntity>();
                var topicRules = rules.Where(rule =>
rule.TopicId.Equals(topic.Id));
                rulesList.AddRange(topicRules);
                topic.Rules = rulesList;
            }

            return topics;
        }

        public override async Task<TopicEntity> GetById(int id)
        {
            var topic = await _context.Set<TopicEntity>().FindAsync(id);

            if (topic != null)
            {
                var rules = await _context.Set<RuleEntity>().ToListAsync();

                var rulesList = new List<RuleEntity>();
                var topicRules = rules.Where(rule => rule.TopicId.Equals(id));
                rulesList.AddRange(topicRules);
                topic.Rules = rulesList;
            }

            return topic;
        }
    }
}
```

### A.3.11. UserRepository.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using Svitlofor.Data.Entities;

namespace Svitlofor.Data.Repositories
{
    public interface IUserRepository : IRepository<UserEntity, string>
    {

    }

    public class UserRepository : EfRepository<UserEntity, string>,
IUserRepository
    {
        private ICardProgressRepository _cardProgressRepository;
        private IFriendsRepository _friendsRepository;
        public UserRepository(SvitloforDbContext context,
ICardProgressRepository cardProgressRepository, IFriendsRepository
friendsRepository) : base(context)
        {
            _cardProgressRepository = cardProgressRepository;
            _friendsRepository = friendsRepository;
        }

        public override async Task<List<UserEntity>> GetList()
        {
            var users = await _context.Set<UserEntity>().ToListAsync();
            var progress = await _cardProgressRepository.GetList();
            var friends = await _friendsRepository.GetList();

            foreach (var user in users)
            {
                user.Progress = await GetProgress(user.Id, progress);

                var sentRequests = new List<FriendsEntity>();
                var userSent = friends.Where(sr =>
sr.RequestedById.Equals(user.Id));
                sentRequests.AddRange(userSent);
                user.SentFriendRequests = sentRequests;

                var receivedRequests = new List<FriendsEntity>();
                var userReceived = friends.Where(rr =>
rr.RequestedToId.Equals(user.Id));
                receivedRequests.AddRange(userReceived);
                user.ReceivedFriendRequests = receivedRequests;
            }

            return users;
```

```
        }

        public override async Task<UserEntity> GetById(string id)
        {
            var user = await _context.Set<UserEntity>().FindAsync(id);

            if (user != null)
            {
                var friends = await _friendsRepository.GetList();

                user.Progress = await GetProgress(id, null);

                var sentRequests = new List<FriendsEntity>();
                var        userSent        =        friends.Where(sr        =>
sr.RequestedById.Equals(user.Id));
                sentRequests.AddRange(userSent);
                user.SentFriendRequests = sentRequests;

                var receivedRequests = new List<FriendsEntity>();
                var       userReceived       =       friends.Where(rr       =>
rr.RequestedToId.Equals(user.Id));
                receivedRequests.AddRange(userReceived);
                user.ReceivedFriendRequests = receivedRequests;
            }

            return user;
        }

        private  async  Task<List<CardProgressEntity>>  GetProgress(string  id,
List<CardProgressEntity> progress)
        {
            if (progress == null)
                progress = await _cardProgressRepository.GetList();

            var progressList = new List<CardProgressEntity>();
            var userProgress = progress.Where(pr => pr.Email.Equals(id));
            progressList.AddRange(userProgress);

            return progressList;
        }
    }
}
```

## A.4. SvitloforDbContext.cs

```
using Microsoft.EntityFrameworkCore;
using Svitlofor.Data.Entities;
using Svitlofor.Data.EntityConfiguration;

namespace Svitlofor.Data
{
```

```
    public class SvitloforDbContext : DbContext
    {
        public SvitloforDbContext()
        {
        }

        public         SvitloforDbContext(DbContextOptions<SvitloforDbContext>
options)
            : base(options)
        {
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
            modelBuilder.ApplyConfiguration(new CardConfiguration());
            modelBuilder.ApplyConfiguration(new CommentConfiguration());
            modelBuilder.ApplyConfiguration(new TestConfiguration());
            modelBuilder.ApplyConfiguration(new TopicConfiguration());
            modelBuilder.ApplyConfiguration(new UserConfiguration());
        }

        //entities
        public DbSet<TopicEntity> Topics { get; set; }
        public DbSet<RuleEntity> Rules { get; set; }
        public DbSet<CardEntity> Cards { get; set; }
        public DbSet<UserEntity> Users { get; set; }
        public DbSet<TestEntity> Tests { get; set; }
        public DbSet<CommentEntity> Comments { get; set; }
        public DbSet<CardProgressEntity> CardProgress { get; set; }
        public DbSet<FriendsEntity> Friends { get; set; }
        public DbSet<FineEntity> Fines { get; set; }
    }
}
```

# Додаток Б. Код проекту Svitlofor.WebApi

## Б.1. Controllers

### Б.1.1. CrudControllerBase.cs

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Svitlofor.WebApi.Models;
using Svitlofor.WebApi.Services;

namespace Svitlofor.WebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public abstract class CrudControllerBase<TModel, TKey> : ControllerBase
        where TModel : IModel<TKey>
    {

        protected readonly ICrudService<TModel, TKey> _service;

        public CrudControllerBase(ICrudService<TModel, TKey> service)
        {
            _service = service;
        }


        // GET: api/<CrudControllerBase>
        [Authorize]
        [HttpGet]
        [ProducesResponseType(StatusCodes.Status200OK)]
        public async Task<ActionResult<List<TModel>>> GetList()
        {
            var result = await _service.GetList();

            return Ok(result);
        }

        // GET api/<CrudControllerBase>/5
        [AllowAnonymous]
        [HttpGet("{id}")]
        [ProducesResponseType(StatusCodes.Status200OK)]
        [ProducesResponseType(typeof(string), StatusCodes.Status404NotFound)]
        public virtual async Task<ActionResult<TModel>> Get(TKey id)
        {
            Console.WriteLine("Get " + id);
```

```
            var model = await _service.Get(id);

            if (model == null)
                return NotFound($"{typeof(TModel).Name}.id={id} not found");

            return model;
        }

        // POST api/<CrudControllerBase>
        [Authorize]
        [HttpPost]
        [ProducesResponseType(StatusCodes.Status201Created)]
        [ProducesResponseType(StatusCodes.Status409Conflict)]
        public virtual async Task<ActionResult<TModel>> Add([FromBody] TModel
model)
        {
            var existModel = await _service.Get(model.Id);

            if (existModel != null)
                return Conflict($"{typeof(TModel).Name}.id={model.Id} already
exist");

            var result = await _service.Add(model);

            return CreatedAtAction(nameof(Get), new { id = result.Id}, result);

        }

        // PUT api/<CrudControllerBase>/5
        [Authorize]
        [HttpPut("{id}")]
        [ProducesResponseType(StatusCodes.Status404NotFound)]
        [ProducesResponseType(StatusCodes.Status200OK)]
        public virtual async Task<ActionResult<TModel>> Update([FromBody]
TModel model)
        {
            var existModel = await _service.Get(model.Id);

            if (existModel == null)
                return NotFound($"{typeof(TModel).Name}.id={model.Id} not
found");

            var result = await _service.Update(model);

            return Ok(result);
        }

        // DELETE api/<CrudControllerBase>/5
        [Authorize]
        [HttpDelete("{id}")]
```

```
        [ProducesResponseType(StatusCodes.Status204NoContent)]
        [ProducesResponseType(StatusCodes.Status200OK)]
        public virtual async Task Delete(TKey id)
        {
            var model = await _service.Get(id);

            if (model != null)
                await _service.Delete(id);
        }
    }
}
```

### Б.1.2. CardController.cs

```
using Microsoft.AspNetCore.Mvc;
using Svitlofor.WebApi.Models;
using Svitlofor.WebApi.Services;

namespace Svitlofor.WebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CardController : CrudControllerBase<Card, int>
    {
        public CardController(ICardService service) : base(service)
        {
        }
    }
}
```

### Б.1.3. CardProgressController.cs

```
using Microsoft.AspNetCore.Mvc;
using Svitlofor.WebApi.Models;
using Svitlofor.WebApi.Services;

namespace Svitlofor.WebApi.Controllers
{

    [Route("api/[controller]")]
    [ApiController]
    public class CardProgressController : CrudControllerBase<CardProgress,
string>
    {
        public CardProgressController(ICardProgressService service) :
base(service)
        {
        }
    }
}
```

### Б.1.4. CommentController.cs

```
using Microsoft.AspNetCore.Mvc;
using Svitlofor.WebApi.Models;
using Svitlofor.WebApi.Services;

namespace Svitlofor.WebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CommentController : CrudControllerBase<Comment, int>
    {
        public CommentController(ICommentService service) : base(service)
        {
        }
    }
}
```

### Б.1.5. FineController.cs

```
using Microsoft.AspNetCore.Mvc;
using Svitlofor.WebApi.Models;
using Svitlofor.WebApi.Services;

namespace Svitlofor.WebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class FineController : CrudControllerBase<Fine, string>
    {
        public FineController(IFineService service) : base(service)
        {
        }
    }
}
```

### Б.1.6. FriendsController.cs

```
using Microsoft.AspNetCore.Mvc;
using Svitlofor.WebApi.Models;
using Svitlofor.WebApi.Services;

namespace Svitlofor.WebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class FriendsController : CrudControllerBase<Friends, int>
    {
        public FriendsController(IFriendsService service) : base(service)
        {
        }
```

```
    }
}
```

### Б.1.7. TestController.cs

```
using Microsoft.AspNetCore.Mvc;
using Svitlofor.WebApi.Models;
using Svitlofor.WebApi.Services;

namespace Svitlofor.WebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class TestController : CrudControllerBase<Test, int>
    {
        public TestController(ITestService service) : base(service)
        {
        }
    }
}
```

### Б.1.8. TopicController.cs

```
using Microsoft.AspNetCore.Mvc;
using Svitlofor.WebApi.Models;
using Svitlofor.WebApi.Services;

namespace Svitlofor.WebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class TopicController : CrudControllerBase<Topic, int>
    {
        public TopicController(ITopicService service) : base(service)
        {
        }
    }
}
```

### Б.1.9. UserController.cs

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Svitlofor.WebApi.Models;
using Svitlofor.WebApi.Services;

namespace Svitlofor.WebApi.Controllers
{
    [Route("api/[controller]")]
```

```
    [ApiController]
    public class UserController : CrudControllerBase<User, string>
    {
        protected IUserService service => (IUserService)_service;
        public UserController(IUserService service) : base(service)
        {
        }

        [HttpPost("/api/User/login")]
        [ProducesResponseType(StatusCodes.Status200OK)]
        [ProducesResponseType(typeof(string), StatusCodes.Status404NotFound)]
        public virtual async Task<ActionResult<Token>> Token([FromBody] Login
login)
        {
            var result = await service.Login(login.Username, login.Password);
            if (result == null)
            {
                return NotFound($"User was not found");
            }

            return Ok(result);
        }

        [AllowAnonymous]
        [HttpPost]
        [ProducesResponseType(StatusCodes.Status201Created)]
        [ProducesResponseType(StatusCodes.Status409Conflict)]
        public  override  async  Task<ActionResult<User>>  Add([FromBody]  User
model)
        {
            var existModel = await _service.Get(model.Id);

            if (existModel != null)
                return  Conflict($"{typeof(User).Name}.id={model.Id}  already
exist");

            var result = await _service.Add(model);

            return  CreatedAtAction(nameof(Get),  new  {  id  =  result.Id  },
result);

        }
    }
}
```

## Б.2. Models

### Б.2.1. IModel.cs

```
namespace Svitlofor.WebApi.Models
{
```

```
    public interface IModel<TKey>
    {
        TKey Id { get; set; }
    }
}
```

### Б.2.2. Card.cs

```
namespace Svitlofor.WebApi.Models
{
    public class Card : IModel<int>
    {
        public int Id { get; set; }
        public string Notes { get; set; }
    }
}
```

### Б.2.3. CardProgress.cs

```
namespace Svitlofor.WebApi.Models
{
    public class CardProgress : IModel<string>
    {
        public CardProgress()
        {
            Correct = 0;
            Attempts = 0;
            Wrong = 0;
            Progress = 0;
        }

        public string Id { get; set; }
        public string Email { get; set; }
        public int CardId { get; set; }
        public int Correct { get; set; }
        public int Wrong { get; set; }
        public int Attempts { get; set; }
        public double Progress { get; set; }
    }
}
```

### Б.2.4. Comment.cs

```
using System;
using System.Collections.Generic;
namespace Svitlofor.WebApi.Models
{
    public class Comment : IModel<int>
    {
        public int Id { get; set; }
        public string Text { get; set; }
```

```
        public int TestId { get; set; }
        public string Email { get; set; }
        public string Login { get; set; }
        public string Image { get; set; }
        public DateTime DateTime { get; set; }
        public int? PreviousId { get; set; }
        public virtual ICollection<Comment> Next { get; set; }
    }
}
```

### Б.2.5. Fine.cs

```
namespace Svitlofor.WebApi.Models
{
    public class Fine : IModel<string>
    {
        public string Id { get; set; }
        public string Text { get; set; }
        public int Price { get; set; }
    }
}
```

### Б.2.6. Friends.cs

```
namespace Svitlofor.WebApi.Models
{
    public class Friends : IModel<int>
    {
        public int Id { get; set; }
        public string RequestedById { get; set; }
        public string RequestedToId { get; set; }
        public User Friend { get; set; }
        public bool IsAccepted { get; set; }
    }
}
```

### Б.2.7. Login.cs

```
namespace Svitlofor.WebApi.Models
{
    public class Login
    {
        public string Username { get; set; }
        public string Password { get; set; }
    }
}
```

### Б.2.8. Rule.cs

```
namespace Svitlofor.WebApi.Models
```

```
{
    public class Rule : IModel<string>
    {
        public string Id { get; set; }
        public int TopicId { get; set; }
        public string Text { get; set; }
        public string Image { get; set; }
    }
}
```

### Б.2.9. Test.cs

```
using System.Collections.Generic;
namespace Svitlofor.WebApi.Models
{
    public class Test : IModel<int>
    {
        public int Id { get; set; }
        public string Image { get; set; }
        public string Question { get; set; }
        public string Variant1 { get; set; }
        public string Variant2 { get; set; }
        public string Variant3 { get; set; }
        public string Variant4 { get; set; }
        public int Correct { get; set; }
        public string Notes { get; set; }
        public int CardId { get; set; }
        public int TopicId { get; set; }
        public virtual ICollection<Comment> Comments { get; set; }
    }
}
```

### Б.2.10. Token.cs

```
namespace Svitlofor.WebApi.Models
{
    public class Token
    {
        public string Email { get; set; }
        public string AccessToken { get; set; }
    }
}
```

### Б.2.11. Topic.cs

```
using System.Collections.Generic;
namespace Svitlofor.WebApi.Models
{
    public class Topic : IModel<int>
    {
        public int Id { get; set; }
```

```
        public string Title { get; set; }
        public string Notes { get; set; }
        public virtual ICollection<Rule> Rules { get; set; }
    }
}
```

### Б.2.12. User.cs

```
using System.Collections.Generic;
namespace Svitlofor.WebApi.Models
{
    public class User : IModel<string>
    {
      public string Id { get; set; }
      public string Name { get; set; }
      public string Password { get; set; }
      public string Image { get; set; }
      public bool Confirmed { get; set; }
      public virtual ICollection<CardProgress> Progress { get; set; }
      public virtual ICollection<Friends> SentFriendRequests { get; set; }
      public virtual ICollection<Friends> ReceivedFriendRequests { get; set; }
    }
}
```

## Б.3. Services

### Б.3.1. ICrudService.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Svitlofor.WebApi.Services
{
    public interface ICrudService<TModel, TKey>
    {
        Task<List<TModel>> GetList();
        Task<TModel> Get(TKey modelId);
        Task<TModel> Add(TModel model);
        Task<TModel> Update(TModel model);
        Task Delete(TKey modelId);
    }
}
```

### Б.3.2. CrudService.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using AutoMapper;
using Microsoft.AspNetCore.Mvc;
using Svitlofor.Data.Repositories;
using Svitlofor.WebApi.Models;
```

```
namespace Svitlofor.WebApi.Services
{
    public abstract class CrudService<TEntity, TModel, TKey> : ICrudService<TModel, TKey>
        where TEntity : class
        where TModel : IModel<TKey>
    {
        protected readonly IRepository<TEntity, TKey> _repository;
        protected readonly IMapper _mapper;

        public CrudService(IRepository<TEntity, TKey> repository, IMapper mapper)
        {
            _repository = repository;
            _mapper = mapper;
        }


        public virtual async Task<List<TModel>> GetList()
        {
            var entityList = await _repository.GetList();
            var modelList = _mapper.Map<List<TModel>>(entityList);
            return modelList;
        }

        public virtual async Task<TModel> Get(TKey modelId)
        {
            var resultEntity = await _repository.GetById(modelId).ConfigureAwait(false);
            var model = _mapper.Map<TModel>(resultEntity);
            return model;
        }

        public virtual async Task<TModel> Add([FromBody] TModel model)
        {
            var entity = _mapper.Map<TEntity>(model);
            var resultEntity = await _repository.Add(entity).ConfigureAwait(false);
            var resultModel = _mapper.Map<TModel>(resultEntity);

            resultModel = await AfterAddOrUpdate(resultModel).ConfigureAwait(false);

            return resultModel;
        }

        public virtual async Task<TModel> Update(TModel model)
        {
            var existEntity = await _repository.GetById(model.Id).ConfigureAwait(false);
```

```
            var entity = _mapper.Map(model, existEntity);

            await _repository.Update(entity).ConfigureAwait(false);
            var resultModel = _mapper.Map<TModel>(entity);

            resultModel                        =                        await
AfterAddOrUpdate(resultModel).ConfigureAwait(false);

            return resultModel;
        }

        public virtual async Task Delete(TKey modelId)
        {
            var                 entity                 =                 await
_repository.GetById(modelId).ConfigureAwait(false);
            if (entity != null)
            {
                await _repository.Delete(entity).ConfigureAwait(false);
            }
        }

        protected Task<TModel> AfterAddOrUpdate(TModel savedModel)
        {
            return Task.FromResult(savedModel);
        }
    }
}
```

### Б.3.3. CardProgressService.cs

```
using AutoMapper;
using Svitlofor.Data.Entities;
using Svitlofor.Data.Repositories;
using Svitlofor.WebApi.Models;

namespace Svitlofor.WebApi.Services
{
    public interface ICardProgressService : ICrudService<CardProgress, string>
    {
    }

    public   class   CardProgressService   :   CrudService<CardProgressEntity,
CardProgress, string>, ICardProgressService
    {
        public CardProgressService(ICardProgressRepository repository, IMapper
mapper)
            : base(repository, mapper)
        {
        }
    }
}
```

### Б.3.4. CardService.cs

```
using System.Linq;
using System.Threading.Tasks;
using AutoMapper;
using Microsoft.AspNetCore.Mvc;
using Svitlofor.Data.Entities;
using Svitlofor.Data.Repositories;
using Svitlofor.WebApi.Models;

namespace Svitlofor.WebApi.Services
{
    public interface ICardService : ICrudService<Card, int>
    {
    }

    public class CardService : CrudService<CardEntity, Card, int>, ICardService
    {
        private ICardProgressRepository _cardProgressRepository;
        private IUserRepository _userRepository;

        public CardService(ICardRepository repository, ICardProgressRepository
cardProgressRepository, IUserRepository userRepository, IMapper mapper)
            : base(repository, mapper)
        {
            _cardProgressRepository = cardProgressRepository;
            _userRepository = userRepository;
        }

        public override async Task<Card> Add([FromBody] Card model)
        {
            var usersList = await _userRepository.GetList();
            foreach (var user in usersList)
            {
                CardProgress cp = new CardProgress
                {
                    Id = model.Id+""+ user.Id,
                    CardId = model.Id,
                    Email = user.Id
                };

                var cpEntity = _mapper.Map<CardProgressEntity>(cp);
                var resultCp = await
_cardProgressRepository.Add(cpEntity).ConfigureAwait(false);
            }

            var entity = _mapper.Map<CardEntity>(model);
            var resultEntity = await
_repository.Add(entity).ConfigureAwait(false);
            var resultModel = _mapper.Map<Card>(resultEntity);
```

```
            resultModel                        =                        await
AfterAddOrUpdate(resultModel).ConfigureAwait(false);

            return resultModel;
        }

        public override async Task Delete(int modelId)
        {
            var                 entity                 =                 await
_repository.GetById(modelId).ConfigureAwait(false);
            if (entity != null)
            {
                await _repository.Delete(entity).ConfigureAwait(false);
                var            cardProgressList            =            (await
_cardProgressRepository.GetList()).Where(cp => cp.CardId.Equals(modelId)); ;
                foreach (var cardProgress in cardProgressList)
                {
                    await
_cardProgressRepository.Delete(cardProgress).ConfigureAwait(false);
                }
            }
        }
    }
}
```

### Б.3.5. CommentService.cs

```
using System;
using System.Threading.Tasks;
using AutoMapper;
using Microsoft.AspNetCore.Mvc;
using Svitlofor.Data.Entities;
using Svitlofor.Data.Repositories;
using Svitlofor.WebApi.Models;

namespace Svitlofor.WebApi.Services
{
    public interface ICommentService : ICrudService<Comment, int>
    {
    }

    public class CommentService : CrudService<CommentEntity, Comment, int>,
ICommentService
    {
        public CommentService(ICommentRepository repository,  IMapper mapper)
            : base(repository, mapper)
        {
        }

        public override async Task<Comment> Add([FromBody] Comment model)
        {
```

```
            var now = DateTime.Now;
            model.DateTime = new DateTime(now.Year, now.Month, now.Day, now.Hour, now.Minute, 0);

            var entity = _mapper.Map<CommentEntity>(model);
            var resultEntity = await _repository.Add(entity).ConfigureAwait(false);
            var resultModel = _mapper.Map<Comment>(resultEntity);
            resultModel = await AfterAddOrUpdate(resultModel).ConfigureAwait(false);

            return await Get(resultModel.Id);
        }

    }
}
```

### Б.3.6. FineService.cs

```
using AutoMapper;
using Svitlofor.Data.Entities;
using Svitlofor.Data.Repositories;
using Svitlofor.WebApi.Models;

namespace Svitlofor.WebApi.Services
{
    public interface IFineService : ICrudService<Fine, string>
    {
    }

    public class FineService : CrudService<FineEntity, Fine, string>, IFineService
    {
        public FineService(IFineRepository repository, IMapper mapper)
            : base(repository, mapper)
        {
        }
    }
}
```

### Б.3.7. FriendsService.cs

```
using AutoMapper;
using Svitlofor.Data.Entities;
using Svitlofor.Data.Repositories;
using Svitlofor.WebApi.Models;

namespace Svitlofor.WebApi.Services
{
    public interface IFriendsService : ICrudService<Friends, int>
    {
    }
```

```
    public class FriendsService : CrudService<FriendsEntity, Friends, int>, IFriendsService
    {
        public FriendsService(IFriendsRepository repository, IMapper mapper)
            : base(repository, mapper)
        {
        }
    }
}
```

### Б.3.8. RuleService.cs

```
using AutoMapper;
using Svitlofor.Data.Entities;
using Svitlofor.Data.Repositories;
using Svitlofor.WebApi.Models;

namespace Svitlofor.WebApi.Services
{
    public interface IRuleService : ICrudService<Rule, string>
    {
    }

    public class RuleService : CrudService<RuleEntity, Rule, string>, IRuleService
    {
        public RuleService(IRuleRepository repository, IMapper mapper)
            : base(repository, mapper)
        {
        }
    }
}
```

### Б.3.9. TestService.cs

```
using AutoMapper;
using Svitlofor.Data.Entities;
using Svitlofor.Data.Repositories;
using Svitlofor.WebApi.Models;

namespace Svitlofor.WebApi.Services
{
    public interface ITestService : ICrudService<Test, int>
    {
    }

    public class TestService : CrudService<TestEntity, Test, int>, ITestService
    {
        public TestService(ITestRepository repository, IMapper mapper)
            : base(repository, mapper)
```

```
        {
        }
    }
}
```

### Б.3.10. TopicService.cs

```
using AutoMapper;
using Svitlofor.Data.Entities;
using Svitlofor.Data.Repositories;
using Svitlofor.WebApi.Models;

namespace Svitlofor.WebApi.Services
{
    public interface ITopicService : ICrudService<Topic, int>
    {
    }

    public class TopicService : CrudService<TopicEntity, Topic, int>,
ITopicService
    {
        public TopicService(ITopicRepository repository, IMapper mapper)
            : base(repository, mapper)
        {
        }
    }
}
```

### Б.3.11. UserService.cs

```
using System;
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Threading.Tasks;
using AutoMapper;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using Svitlofor.Data.Entities;
using Svitlofor.Data.Repositories;
using Svitlofor.WebApi.Models;

namespace Svitlofor.WebApi.Services
{
    public interface IUserService : ICrudService<User, string>
    {
        Task<Token> Login(string username, string password);

    }
```

```
    public class UserService : CrudService<UserEntity, User, string>,
IUserService
    {
        private ICardProgressService _cardProgressService;
        private ICardRepository _cardRepository;
        private IFriendsService _friendsService;

        public UserService(IUserRepository repository, ICardProgressService
cardProgressService, IFriendsService friendsService,
            ICardRepository cardRepository, IMapper mapper)
            : base(repository, mapper)
        {
            _cardProgressService = cardProgressService;
            _cardRepository = cardRepository;
            _friendsService = friendsService;
        }

        public override async Task<User> Get(string modelId)
        {
            var resultEntity = await
_repository.GetById(modelId).ConfigureAwait(false);
            var model = _mapper.Map<User>(resultEntity);

            if (model != null)
            {
                foreach (var friends in model.SentFriendRequests)
                {
                    var person = await
_repository.GetById(friends.RequestedToId).ConfigureAwait(false);
                    friends.Friend = _mapper.Map<User>(person);

                }

                foreach (var friends in model.ReceivedFriendRequests)
                {
                    var person = await
_repository.GetById(friends.RequestedById).ConfigureAwait(false);
                    person.Comments = null;
                    person.Password = null;
                    person.ReceivedFriendRequests = null;
                    person.SentFriendRequests = null;
                    friends.Friend = _mapper.Map<User>(person);
                }
            }

            return model;
        }

        public override async Task<User> Add([FromBody] User model)
        {
            model.Confirmed = false;
```

```
            model.Progress = new List<CardProgress>();
            model.SentFriendRequests = new List<Friends>();
            model.ReceivedFriendRequests = new List<Friends>();
            var cardsList = await _cardRepository.GetList();
            foreach (var card in cardsList)
            {
                model.Progress.Add(new CardProgress
                {
                    Id = card.Id + "" + model.Id,
                    CardId = card.Id,
                    Email = model.Id
                });
            }

            var entity = _mapper.Map<UserEntity>(model);
            var               resultEntity              =               await
_repository.Add(entity).ConfigureAwait(false);
            var resultModel = _mapper.Map<User>(resultEntity);

            resultModel                        =                        await
AfterAddOrUpdate(resultModel).ConfigureAwait(false);

            return resultModel;
        }

        public override async Task<User> Update(User model)
        {
            ICollection<CardProgress> progressList = model.Progress;
            foreach (var progress in progressList)
            {
                await _cardProgressService.Delete(progress.Id);
            }
            model.Progress = new List<CardProgress>();

            ICollection<Friends>             friendsReceivedList             =
model.ReceivedFriendRequests;
            foreach (var friends in friendsReceivedList)
            {
                await _friendsService.Delete(friends.Id);
            }
            model.ReceivedFriendRequests = new List<Friends>();

            ICollection<Friends> friendsSentList = model.SentFriendRequests;
            foreach (var friends in model.SentFriendRequests)
            {
                await _friendsService.Delete(friends.Id);
            }
            model.SentFriendRequests = new List<Friends>();
```

```
            var existEntity = await
_repository.GetById(model.Id).ConfigureAwait(false);
            var entity = _mapper.Map(model, existEntity);
            await _repository.Update(entity).ConfigureAwait(false);
            var resultModel = _mapper.Map<User>(entity);
            resultModel = await
AfterAddOrUpdate(resultModel).ConfigureAwait(false);


            resultModel.Progress =  new List<CardProgress>();
            foreach (var progress in progressList)
            {
                var newProgress = await _cardProgressService.Add(progress);
                resultModel.Progress.Add(newProgress);
            }

            resultModel.ReceivedFriendRequests = new List<Friends>();
            foreach (var fr in friendsReceivedList)
            {
                var newFriend = await _friendsService.Add(fr);
                resultModel.ReceivedFriendRequests.Add(newFriend);
            }

            resultModel.SentFriendRequests = new List<Friends>();
            foreach (var fr in friendsSentList)
            {
                var newFriend = await _friendsService.Add(fr);
                resultModel.SentFriendRequests.Add(newFriend);
            }

            return resultModel;
        }


        public override async Task Delete(string modelId)
        {
            var entity = await
_repository.GetById(modelId).ConfigureAwait(false);
            if (entity != null)
            {
                foreach (var progress in entity.Progress)
                {
                    await _cardProgressService.Delete(progress.Id);
                }

                foreach (var friends in entity.SentFriendRequests)
                {
                    await _friendsService.Delete(friends.Id);
                }

                foreach (var friends in entity.ReceivedFriendRequests)
```

```
                {
                    await _friendsService.Delete(friends.Id);
                }

                await _repository.Delete(entity).ConfigureAwait(false);
            }
        }


        public async Task<Token> Login(string username, string password)
        {
            Token response = null;

            var identity = await GetIdentity(username, password);
            if (identity != null) {
                var now = DateTime.UtcNow;
                // create JWT - Token
                var jwt = new JwtSecurityToken(
                    issuer: AuthOptions.ISSUER,
                    audience: AuthOptions.AUDIENCE,
                    notBefore: now,
                    claims: identity.Claims,
                    expires:
now.Add(TimeSpan.FromMinutes(AuthOptions.LIFETIME)),
                    signingCredentials:                                    new
SigningCredentials(AuthOptions.GetSymmetricSecurityKey(),
                        SecurityAlgorithms.HmacSha256));
                var               encodedJwt               =               new
JwtSecurityTokenHandler().WriteToken(jwt);

                response  =  new  Token  {AccessToken  =  encodedJwt,  Email  =
identity.Name};
            }

            return response;
        }

        private async Task<ClaimsIdentity> GetIdentity(string username, string
password)
        {
            var userEntityList = await _repository.GetList();
            var userList = _mapper.Map<List<User>>(userEntityList);
            var person = userList.Find(x => x.Id == username && x.Password ==
password);
            if (person != null)
            {
                var claims = new List<Claim>
                {
                    new Claim(ClaimsIdentity.DefaultNameClaimType, person.Id),
                    new Claim(ClaimsIdentity.DefaultRoleClaimType, "user")
                };
```

```
                ClaimsIdentity claimsIdentity =
                    new            ClaimsIdentity(claims,            "Token",
ClaimsIdentity.DefaultNameClaimType,
                        ClaimsIdentity.DefaultRoleClaimType);
                return claimsIdentity;
            }
            return null;
        }
    }
}
```

## Б.4. ApplicationProfile

```
using AutoMapper;
using Svitlofor.Data.Entities;
using Svitlofor.WebApi.Models;

namespace Svitlofor.WebApi.Mapping
{
    public class ApplicationProfile : Profile
    {
        public ApplicationProfile()
        {
            CreateMap<TestEntity, Test>().ReverseMap();
            CreateMap<CardEntity, Card>().ReverseMap();
            CreateMap<TopicEntity, Topic>().ReverseMap();
            CreateMap<RuleEntity, Rule>().ReverseMap();
            CreateMap<UserEntity, User>().ReverseMap();
            CreateMap<FriendsEntity, Friends>().ReverseMap();
            CreateMap<CommentEntity, Comment>().ReverseMap();
            CreateMap<CardProgressEntity, CardProgress>().ReverseMap();
            CreateMap<FineEntity, Fine>().ReverseMap();
        }
    }
}
```

## Б.5. AuthOptions

```
using System.Text;
using Microsoft.IdentityModel.Tokens;

namespace Svitlofor.WebApi
{
    public class AuthOptions
    {
        public const string ISSUER = "MyAuthServer";
        public const string AUDIENCE = "MyAuthClient";
        const string KEY = "mysupersecret_secretkey!123";
        public const int LIFETIME = 1;
        public static SymmetricSecurityKey GetSymmetricSecurityKey()
        {
```

```
            return new SymmetricSecurityKey(Encoding.ASCII.GetBytes(KEY));
        }
    }
}
```

## Б.6. RepositoriesAutofacModule

```
using Autofac;
using Svitlofor.Data.Repositories;
namespace Svitlofor.WebApi
{
    public class RepositoriesAutofacModule : Module
    {
        protected override void Load(ContainerBuilder builder)
        {
            builder.RegisterType<UserRepository>()
                .As<IUserRepository>()
                .InstancePerLifetimeScope();
            builder.RegisterType<TopicRepository>()
                .As<ITopicRepository>()
                .InstancePerLifetimeScope();
            builder.RegisterType<TestRepository>()
                .As<ITestRepository>()
                .InstancePerLifetimeScope();
            builder.RegisterType<CommentRepository>()
                .As<ICommentRepository>()
                .InstancePerLifetimeScope();
            builder.RegisterType<CardRepository>()
                .As<ICardRepository>()
                .InstancePerLifetimeScope();
            builder.RegisterType<CardProgressRepository>()
                .As<ICardProgressRepository>()
                .InstancePerLifetimeScope();
            builder.RegisterType<FriendsRepository>()
                .As<IFriendsRepository>()
                .InstancePerLifetimeScope();
            builder.RegisterType<RuleRepository>()
                .As<IRuleRepository>()
                .InstancePerLifetimeScope();
            builder.RegisterType<FineRepository>()
                .As<IFineRepository>()
                .InstancePerLifetimeScope();
        }
    }
}
```

## Б.7. ServicesAutofacModule

```
using Autofac;
using Svitlofor.WebApi.Services;
```

```
namespace Svitlofor.WebApi
{
    public class ServicesAutofacModule : Module
    {
        protected override void Load(ContainerBuilder builder)
        {
            builder.RegisterType<TopicService>()
                .As<ITopicService>()
                .InstancePerLifetimeScope();
            builder.RegisterType<CardService>()
                .As<ICardService>()
                .InstancePerLifetimeScope();
            builder.RegisterType<TestService>()
                .As<ITestService>()
                .InstancePerLifetimeScope();
            builder.RegisterType<UserService>()
                .As<IUserService>()
                .InstancePerLifetimeScope();
            builder.RegisterType<CommentService>()
                .As<ICommentService>()
                .InstancePerLifetimeScope();
            builder.RegisterType<FriendsService>()
                .As< IFriendsService>()
                .InstancePerLifetimeScope();
            builder.RegisterType<CardProgressService>()
                .As<ICardProgressService>()
                .InstancePerLifetimeScope();
            builder.RegisterType<RuleService>()
                .As<IRuleService>()
                .InstancePerLifetimeScope();
            builder.RegisterType<FineService>()
                .As<IFineService>()
                .InstancePerLifetimeScope();
        }
    }
}
```

## Б.8. Program

```
using System;
using Autofac.Extensions.DependencyInjection;
using Microsoft.AspNetCore.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Svitlofor.Data;

namespace Svitlofor.WebApi
{
```

```
    public class Program
    {
        public static void Main(string[] args)
        {
            var host = CreateHostBuilder(args).Build();
            CreateDbIfNotExists(host);
            host.Run();
        }


        private static void CreateDbIfNotExists(IHost host)
        {
            using (var scope = host.Services.CreateScope())
            {
                var services = scope.ServiceProvider;
                try
                {
                    var                          db                          =
services.GetRequiredService<SvitloforDbContext>();
                    db.Database.Migrate();
                }
                catch (Exception ex)
                {
                    var                        logger                        =
services.GetRequiredService<ILogger<Program>>();
                    logger.LogError(ex, "An error occurred while migrating the
database.");
                }
            }
        }


        public static IHostBuilder CreateHostBuilder(string[] args)
        {
            var port = Environment.GetEnvironmentVariable("PORT");

            return Host.CreateDefaultBuilder(args)
                .UseServiceProviderFactory(new
AutofacServiceProviderFactory())
                .ConfigureServices(services => services.AddAutofac())
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>().UseUrls("http://*:"     +
port);
                });
        }
    }
}
```

## Б.9. Startup

```
using Autofac;
```

```
using AutoMapper;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;
using Svitlofor.Data;
using Svitlofor.WebApi.Mapping;

namespace Svitlofor.WebApi
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add
services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddCors();
            services.AddDbContext<SvitloforDbContext>(options =>

options.UseMySql(Configuration.GetConnectionString("DefaultConnection")));


services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
                .AddJwtBearer(options =>
                {
                    options.RequireHttpsMetadata = false;
                    options.TokenValidationParameters          =          new
TokenValidationParameters
                    {
                        ValidateIssuer = true,
                        ValidIssuer = AuthOptions.ISSUER,
                        ValidateAudience = true,
                        ValidAudience = AuthOptions.AUDIENCE,
                        ValidateLifetime = true,
                        IssuerSigningKey                                    =
AuthOptions.GetSymmetricSecurityKey(),
                        ValidateIssuerSigningKey = true,
                    };
                });
```

```
            services.AddControllers();
            services.AddHealthChecks();
            // register AutoMapper
            services.AddAutoMapper(configuration                                =>
configuration.AddProfile(new ApplicationProfile()));
            // register Swagger
            services.AddSwaggerGen(options =>
            {
                options.SwaggerDoc("v1", new OpenApiInfo
                {
                    Version = "v1",
                    Title = "My API"
                });
            });
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment()){
                app.UseDeveloperExceptionPage();
            }
            app.UseHttpsRedirection();
            app.UseRouting();
            app.UseCors(
         options => options.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader()
            );
            app.UseAuthentication();
            app.UseAuthorization();
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
                endpoints.MapHealthChecks("/api/health");
            });

            app.UseSwagger();
            app.UseSwaggerUI(options =>
            {
                options.SwaggerEndpoint("/swagger/v1/swagger.json", "My API");
            });

        }
        public virtual void ConfigureContainer(ContainerBuilder builder)
        {
builder.RegisterAssemblyModules(System.Reflection.Assembly.GetExecutingAssemb
ly());
        }
    }
}
```

# Додаток В. Код проекту SvitloforClient

## В.1. index.html

```
<!DOCTYPE html>
<html lang="">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width,initial-scale=1.0">
  <link rel="icon" href="/images/svitlofor.png">
  <link rel="stylesheet" type="text/css" href="../node_modules/bootstrap/dist/css/bootstrap.min.css"/>
  <link rel="stylesheet" href="style.css">

  <title><%= htmlWebpackPlugin.options.title %></title>

</head>
<body class="body_style">
<noscript>
  <strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn't work properly without JavaScript enabled.
    Please enable it to continue.</strong>
</noscript>

<div id="app"></div>
</body>
</html>
```

## В.2. style.css

```
@font-face {
    font-family: 'Pangolin';
    src:  url('../src/fonts/Pangolin-Regular.ttf');
}
@font-face {
    font-family: 'UbuntuMono';
    src:  url('../src/fonts/UbuntuMono-Regular.ttf');
}

* {
    box-sizing: border-box;
}

body {
    background: #ffffe6 !important;
    min-height: 100vh;
    display: flex;
    font-weight: 400;
    font-size: 20px !important;
}
```

```
body,
html,
.App,
#app,
.vertical-center {
    width: 100%;
    height: 100%;
}

.vertical-center {
    display: flex;
    text-align: left;
    justify-content: center;
    flex-direction: column;
}

.inner-block {
    width: 450px;
    margin: auto;
    background: whitesmoke;
    box-shadow: 0px 14px 80px rgba(34, 35, 58, 0.2);
    padding: 40px 55px 45px 55px;
    border-radius: 15px;
    transition: all .3s;
}

.card-back {
    background: whitesmoke;
    border-radius: 15px;
    margin: 3%;
    padding: 3%;
}

.nav-link {
    font-size: 25px;
    color: white !important;
    font-family: 'UbuntuMono';
    cursor: pointer;
}

.nav-link:hover{
    color: #f5c100 !important;
}

.page-title {
    font-family: 'UbuntuMono';
    font-weight: bold;
    text-align: center;
    margin: 2%;
```

```
}

.profile-page, .pdr-page, .main-page, .fines-page {
    min-height: 200px;
}
```

## B.3. App.vue

```
<template>
  <div id="app">
    <router-view />
  </div>
</template>

<script>

export default {
  name: 'App'
}
</script>

<style>
</style>
```

## B.4. main.js

```
import '/public/style.css'

import { BootstrapVue } from 'bootstrap-vue'
// Import Bootstrap an BootstrapVue CSS files
import 'bootstrap/dist/css/bootstrap.css'
import 'bootstrap-vue/dist/bootstrap-vue.css'
import 'bootstrap/dist/js/bootstrap.min.js'
// Make BootstrapVue available throughout project
Vue.use(BootstrapVue)

import Vue from 'vue'
import App from './App.vue'
import router from './js/router'

Vue.config.productionTip = false;

new Vue({
  router,
  render: h => h(App)
}).$mount('#app')
```

## B.5. components

### B.5.1. Progress-Round.vue

```
<template>
    <div v-if="value >= 90" class="progress border-correct">
        <span class="progress-left">
            <span class="progress-bar"></span>
        </span>
        <span class="progress-right">
            <span class="progress-bar"></span>
        </span>
        <div class="progress-value">{{Number((value).toFixed(1))}}%</div>
    </div>

    <div v-else class="progress border-wrong">
        <span class="progress-left">
            <span class="progress-bar"></span>
        </span>
        <span class="progress-right">
            <span class="progress-bar"></span>
        </span>
        <div class="progress-value">{{value}}%</div>
    </div>
</template>

<script>
    export default {
        name: "Progress-Round",
        props: {
            value: Number
        }
    }
</script>

<style scoped>
    .progress{
        width: 150px;
        height: 150px;
        line-height: 150px;
        background: none;
        margin: 0 auto;
        box-shadow: none;
        position: relative;
    }
    .progress:after{
        content: "";
        width: 100%;
        height: 100%;
        border-radius: 50%;
        border: 12px solid #fff;
```

```
    position: absolute;
    top: 0;
    left: 0;
}
.progress > span{
    width: 50%;
    height: 100%;
    overflow: hidden;
    position: absolute;
    top: 0;
    z-index: 1;
}
.progress .progress-left{
    left: 0;
}
.progress .progress-bar{
    width: 100%;
    height: 100%;
    background: none;
    border-width: 12px;
    border-style: solid;
    position: absolute;
    top: 0;
}
.progress .progress-left .progress-bar{
    left: 100%;
    border-top-right-radius: 80px;
    border-bottom-right-radius: 80px;
    border-left: 0;
    transform-origin: left;
}
.progress .progress-right{
    right: 0;
}
.progress .progress-right .progress-bar{
    left: -100%;
    border-top-left-radius: 80px;
    border-bottom-left-radius: 80px;
    border-right: 0;
    transform-origin: right;
    animation: loading-1 1.8s linear forwards;
}
.progress .progress-value{
    width: 90%;
    height: 90%;
    border-radius: 50%;
    font-size: 24px;
    line-height: 135px;
    text-align: center;
    position: absolute;
```

```
    top: 5%;
    left: 5%;
}
.progress.border-correct .progress-bar{
    border-color: #1c7430;
}

.progress.border-wrong .progress-bar{
    border-color: #c82333;
}
.progress .progress-left .progress-bar{
    animation: loading-2 1.5s linear forwards 1.8s;
}

@keyframes loading-1{
    0%{
        -webkit-transform: rotate(0deg);
        transform: rotate(0deg);
    }
    100%{
        -webkit-transform: rotate(180deg);
        transform: rotate(180deg);
    }
}
@keyframes loading-2{
    0%{
        -webkit-transform: rotate(0deg);
        transform: rotate(0deg);
    }
    100%{
        -webkit-transform: rotate(144deg);
        transform: rotate(144deg);
    }
}
@keyframes loading-3{
    0%{
        -webkit-transform: rotate(0deg);
        transform: rotate(0deg);
    }
    100%{
        -webkit-transform: rotate(90deg);
        transform: rotate(90deg);
    }
}
@keyframes loading-4{
    0%{
        -webkit-transform: rotate(0deg);
        transform: rotate(0deg);
    }
    100%{
```

```
            -webkit-transform: rotate(36deg);
            transform: rotate(36deg);
        }
    }
    @keyframes loading-5{
        0%{
            -webkit-transform: rotate(0deg);
            transform: rotate(0deg);
        }
        100%{
            -webkit-transform: rotate(126deg);
            transform: rotate(126deg);
        }
    }
</style>
```

### B.5.2. Navbar.vue

```
<template>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a      class="navbar-brand"      href="/main"><img      class="logo"
src="/images/svitlofor.png" alt="Logo"/></a>
        <div class="w-100">
            <a             class="navbar-brand"             href="/main"><span
class="title">Світлофор</span></a>
            <button      class="navbar-toggler"      type="button"      data-
toggle="collapse" data-target="#navbarNavAltMarkup"
                    aria-controls="navbarNavAltMarkup"   aria-expanded="false"
aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div                class="collapse                navbar-collapse"
id="navbarNavAltMarkup"><br/>
                <ul class="navbar-nav mr-auto">
                    <li      class="nav-link"      @click="openPDR()">ПДР<span
class="sr-only" current="current"></span></li>
                    <li class="nav-link" @click="openFines()">Штрафи</li>
                    <li class="nav-link" @click="openTests()">Тести</li>
                </ul>
                <ul class="navbar-nav  flex-row ml-md-auto d-none d-md-flex">
                    <li class="nav-link" @click="openProfile()">Профіль</li>
                    <li class="nav-link" @click="exit()">Вийти</li>
                </ul>
            </div>
        </div>
    </nav>
</template>

<script>
    import router from "@/js/router";
    import API from "@/js/api";
```

```
    export default {
        name: 'Navbar',
        methods: {
            openPDR () {
                router.push({name: 'pdr'});
            },
            openFines () {
                router.push({name: 'fines'});
            },
            openTests () {
                router.push({name: 'tests'});
            },
            openProfile () {
                router.push({name: 'profile', });
            },
            exit () {
                sessionStorage.setItem(API.tokenKey, null);
                sessionStorage.setItem('user', null);
                router.push({name: 'authorization'});
            },
        }
    }
</script>

<style scoped>
    .logo {
        width: 100px;
        height: 100px;
    }

    .title {
        color: whitesmoke;
        font-family: 'Pangolin';
        font-size: 45px;
        margin-right: 2em;
    }
</style>
```

### B.5.3. Main.vue

```
<template>
    <div>
        <Navbar/>

        <div class="main-page">
            <img class="w-100" src="/images/road.png" alt="Title image"/>

        </div>

        <Footer/>
```

```
    </div>
</template>

<script>
    import Navbar from './Navbar.vue'
    import Footer from './Footer.vue'

    export default {
        name: "Main",
        components: {
            Navbar,
            Footer
        }
    }
</script>

<style scoped>

</style>
```

### B.5.4. Footer.vue

```
<template>
    <footer>
        <div class="container text-center text-md-left">
            <div class="row">
                <div class="col-md-8 col-lg-7 text-center text-md-left mb-3 mb-
md-0">
                    <h6   class="font-weight-bold   text-uppercase   mt-3   mb-
4">Інформація про проект</h6>
                    <p>Створено для виконання курсової роботи студентки НаУКМА
спеціальності ІПЗ-4 Ровніної Тетяни</p>
                    <p>Веб-сайт "Світлофор" представляє собою онлайн платформу
для навчання водіїв. він містить інформаційний матеріал по темам з ПДР, а також
тести для перевірки знань.</p>
                    <p>Сайт створено з використанням таких інструментів як
Asp.net Core, Entity Framework, Swagger, Vue.js, Axios, HTML5 Web Workers</p>
                </div>
                <hr class="clearfix w-100 d-md-none">

                <div class="col-md-5 col-lg-4 text-center text-md-left mb-3 mb-
md-0">
                    <h6   class="font-weight-bold   text-uppercase   mt-3   mb-
4">Контакти</h6>
                    <p><b>Телефон:</b> +38 (077)-627-87-80</p>
                    <p><b>Адреса:</b> м. Київ, вул. Григорія Сковороди 2</p>
                    <p><b>Email:</b>                                    <a
href="mailto:svitlofor@gmail.com">svitlofor.ukraine@gmail.com</a></p>
                    <!-- Social buttons -->
                    <div>
                        <a                                      class="fa"
href="https://www.facebook.com/profile.php?id=100015201939759"
target="_blank">
```

```
                            <img src="/images/fb.png" alt="facebook">
                        </a>
                        <a    class="fa"    href="https://t.me/sleep_walker11'"
target="_blank">
                            <img src="/images/telegram.png" alt="telegram">
                        </a>
                    </div>
                </div>
            </div>
        </div>

        <!-- Copyright -->
        <div class="footer-copyright text-center py-3">© 2021 Copyright:
            <a    href="https://github.com/TRovnina/SvitloforServer">    Back-
end</a>
            <a href="https://github.com/TRovnina/Svitlofor-web-client"> Front-
end</a>
        </div>
        <!-- Copyright -->

    </footer>
</template>

<script>
    export default {
        name: "Footer"
    }
</script>

<style scoped>
    footer {
        background: #343a40;
        color: white;
        font-size: 12px;
        left: 0;
        bottom: 0;
        padding: 10px;
        width: 100%;
    }

    footer a {
        color: green;
    }

    .fa {
        display: inline-block;
        width: 40px;
        margin: 2%;
    }

    .fa img {
```

```
        width: 60%;
    }
</style>
```

### B.5.5. Authorization/Authorization.vue

```
<template>

    <div class="vertical-center">
        <div class="inner-block">
            <b-overlay :show="show" rounded="sm">
                <b-form :aria-hidden="show ? 'true' : null">
                    <h3 class="text-center">Вхід</h3>

                    <b-form-group     label="Email"     label-for="inputEmail"
class="mb-2">
                        <b-form-input id="inputEmail" v-model="email"
                                placeholder="example@gmail.com"
:state="validation"
                                required />
                    </b-form-group>

                    <b-form-group   label="Пароль"   label-for="inputPassword"
class="mb-2">
                        <b-form-input id="inputPassword" v-model="password"
                                type="password"  :state="validation"  required
/>
                    </b-form-group>

                    <b-form-invalid-feedback :state="validation" class="mb-2">
                        Не правильний email чи пароль
                    </b-form-invalid-feedback>

                    <button type="button" class="btn btn-warning btn-lg btn-
block" @click="sendRequest">Увійти</button>

                    <small>Не маєте аккаунта? Створіть <a
href="/registration">тут</a>!</small>
                </b-form>
            </b-overlay>
        </div>
    </div>

</template>

<script>
    import router from '@/js/router'
    import API from "@/js/api";

    export default {
        name: 'Authorization',
        data (){
```

```
            return {
                email: "",
                password: "",
                show: false,
                valid: true
            }
        },
        computed: {
            validation() {
                return this.valid
            }
        },
        methods: {
            async sendRequest() {
                this.show = true;

                API.login(this.email, this.password)
                    .then((response) => {
                        this.valid = true;
                        sessionStorage.setItem(API.tokenKey,
response.accessToken);
                        sessionStorage.setItem('user', response.email);
                        this.show = false;
                        router.push({name: 'main'});
                    }).catch(error => {
                        this.valid = false;
                        this.show = false;
                        console.log('error: ' + error);
                    });
            }
        }
    }
</script>

<style scoped>
</style>
```

### B.5.6. Authorization/Registration.vue

```
<template>
    <div class="vertical-center">
        <div class="inner-block">
            <b-overlay :show="show" rounded="sm">
                <b-form :aria-hidden="show ? 'true' : null">
                    <h3 class="text-center">Реєстрація</h3>

                    <b-form-group     label="Email"     label-for="inputEmail"
class="mb-2">
                        <b-form-input id="inputEmail" v-model="email"
                                      placeholder="example@gmail.com"
:state="validation"
```

```
                                    required/>
                    </b-form-group>

                    <b-form-group      label="Ім'я"      label-for="inputName"
class="mb-2">
                        <b-form-input      id="inputName"      v-model="name"
placeholder="Іван Іваненко"
                                    :state="validation" required/>
                    </b-form-group>

                    <b-form-group  label="Пароль"  label-for="inputPassword1"
class="mb-2">
                        <b-form-input  id="inputPassword1"  v-model="password1"
type="password"
                                    :state="validation" required/>
                    </b-form-group>


                    <b-form-group          label="Підтвердіть"          label-
for="inputPassword2" class="mb-2">
                        <b-form-input id="inputPassword2" v-model="password2"
                                    type="password"      :state="validation"
required/>
                    </b-form-group>

                    <b-form-invalid-feedback :state="validation" class="mb-2">
                        <p>Всі поля мають бути заповненні</p>
                        <p>Паролі мають співпадати</p>
                    </b-form-invalid-feedback>

                    <button  type="button"  class="btn  btn-warning  btn-lg  btn-
block" @click="sendRequest"
                            :aria-disabled="!validation">Зареєструватись
                    </button>

                </b-form>
            </b-overlay>
            <b-modal id="registration-result" hide-footer>
                <template #modal-title>
                    Дякуємо за реєстрацію!
                </template>
                <template #modal-header-close>
                    <b-button-close @click="openProfile"/>
                </template>
                <div class="d-block text-center">
                    <p>Ваша реєстрація завершена!</p>
                    <p>Тепер  ви  можете  переглядати  правила  дорожнього  руху,
складати тести та залишати коментарі</p>
                </div>
                <b-button     class="mt-3"     block     @click="openProfile"
variant="warning">OK</b-button>
            </b-modal>
```

```
        </div>
    </div>
</template>

<script>
    import router from '@/js/router'
    import API from "@/js/api";

    export default {
        name: 'Registration',
        data() {
            return {
                email: "",
                password1: "",
                password2: "",
                name: "",
                show: false,
            }
        },
        computed: {
            validation() {
                return this.email != '' && this.name != ''
                    && this.password1 != '' && this.password2 != ''
                    && this.password1 === this.password2;
            }
        },
        methods: {
            async sendRequest() {
                this.show = true;
                API.registration(this.email, this.name, this.password1)
                    .then((response) => {
                        API.login(response.id, response.password)
                            .then((response) => {
                                sessionStorage.setItem(API.tokenKey,
response.accessToken);
                                sessionStorage.setItem('user',
response.email);
                                this.show = false;
                                this.$bvModal.show('registration-result');
                            }).catch(error => {
                                console.log(error);
                            });
                    }).catch(error => {
                        console.log(error);
                    });
            },
            openProfile() {
                this.$bvModal.hide('registration-result');
                router.push({name: 'profile'});
            }
        }
```

```
    }
</script>

<style scoped>

</style>
```

### В.5.7. PDR/FinesPage.vue

```
<template>
    <div>
        <Navbar/>
        <div class="fines-page">
            <h3 class="page-title">Таблиця штрафів за порушення ПДР</h3>
            <div class="fines-list">
            <p>Перелік порушень та сума штрафів за порушення Правил дорожнього руху визначені Кодексом України про адміністративні правопорушення (КУпАП). Штраф сплачується на користь районного Державного казначейства за місцем реєстрації водія. Сплатіть штраф протягом 15 днів, інакше сума штрафу збільшується в 2 рази. В таблиці враховано останні зміни 2021 року.</p>
            <table class="table table-striped">
                <thead>
                <tr>
                    <th scope="col" width="10%">Стаття</th>
                    <th scope="col">Текст</th>
                    <th scope="col" width="13%">Штраф</th>
                </tr>
                </thead>
                <tbody>
                <tr :id="item.id" v-for="item in fines" :key="item.id">
                    <th scope="row" width="10%">{{ item.id }}</th>
                    <td>{{ item.text }}</td>
                    <td width="13%">{{ item.price }} грн</td>
                </tr>
                </tbody>
            </table>
            </div>
        </div>
        <Footer/>
    </div>
</template>

<script>
    import Navbar from '../Navbar.vue'
    import Footer from '../Footer.vue'
    import API from "@/js/api";

    export default {
        name: "FinesPage",
        components: {
            Navbar,
```

```
            Footer
        },
        data () {
            return {
              fines: []
            }
        },
        mounted () {
            API.getFines()
                .then(function( response ){
                    this.fines = response;
                }.bind(this));
        }
    }
</script>

<style scoped>
    .fines-list {
        margin: 2% 10%;
    }

    .fines-list table {
        background-color: whitesmoke;
    }
</style>
```

### B.5.8. PDR/ PDR-Page.vue

```
<template>
    <div>
        <Navbar/>
        <div class="pdr-page">
            <h3 class="page-title">Правила дорожнього руху</h3>
            <ul class="list-group">
                <li            class="list-group-item"            :id="item.id"
@click="openTopic($event)" v-for="item in topics" :key="item.id">
                    {{ item.id }}. {{ item.title }}
                </li>
            </ul>
        </div>
        <Footer/>
    </div>
</template>

<script>
    import Navbar from '../Navbar.vue'
    import Footer from '../Footer.vue'
    import router from "@/js/router";
    import API from '@/js/api'
```

```
    export default {
        name: "PDR-Page",
        components: {
            Navbar,
            Footer
        },
        data () {
            return {
                topics: []
            }
        },
        mounted () {
            API.getTopicsList()
                .then(function( response ){
                    this.topics = response;
                }.bind(this));
        },
        methods: {
            openTopic (event) {
                let id = event.currentTarget.id;
                router.push({ name: 'topic', params: { topicId: id }});
            }
        }
    }
</script>

<style scoped>
    .list-group {
        margin: 2% 10%;
    }

    .list-group-item:hover {
        font-weight: bold;
        color: #f5c100;
        cursor: pointer;
    }
</style>
```

### B.5.9. PDR/ Topic.vue

```
<template>
    <div>
        <Navbar/>
        <h3 class="page-title">{{ topic.id }}. {{ topic.title }}</h3>
        <div class="card-back">
            <p    class=""    :id="item.id"    v-for="item    in    topic.rules"
:key="item.id">
                <b>{{ item.id }}.</b> {{ item.text }}
            </p>
        </div>
```

```
    </div>
</template>

<script>
    import Navbar from '../Navbar.vue'
    import API from '@/js/api'

    export default {
        name: "Topic",
        components: {
            Navbar
        },
        data () {
            return {
                topic: {}
            }
        },
        mounted () {
            API.getTopic(this.$route.params.topicId)
                .then((response) => {
                    this.topic = response
                }).catch(error => {
                    console.log('error: ' + error);
            });
        }
    }
</script>

<style scoped>
</style>
```

### B.5.10. Profile/CardProgress.vue

```
<template>
    <div>
        <h3 class="page-title">Ваш прогрес</h3>
        <div class="row m-4" v-for="item in progress" :key="item.id">
            <span class="col-2"><b>Білет №{{ item.cardId }}</b></span>
            <span class="col-2">Спроб: {{ item.attempts }}</span>
            <div class="col-6">
                <b-progress class="mt-2" v-if="item.correct+item.wrong != 0"
:max="item.correct+item.wrong" show-value>
                    <b-progress-bar :value="item.correct"
variant="success"></b-progress-bar>
                    <b-progress-bar :value="item.wrong" variant="danger"></b-
progress-bar>
                </b-progress>

                <b-progress v-else class="mt-2" max="20">
                    <b-progress-bar/>
                </b-progress>
```

```
            </div>
            <span class="col-2">{{ item.progress }}%</span>
        </div>
    </div>
</template>

<script>
    export default {
        name: "CardProgress",
        props: {
            progress: Array
        }
    }
</script>

<style scoped>
</style>
```

### B.5.11. Profile/Friend.vue

```
<template>
    <div class="friend-row row" :id="item.id">
        <div class="col-6">
            <b-avatar  v-if="item.friend.image  ==  null"  variant="success"
size="70px"/>
            <b-avatar  v-else  variant="success"  :src="item.friend.image"
size="70px"/>
            <span>{{item.friend.name}}</span>
        </div>
        <div class="col-6">
            <b-progress class="mt-4" :max="item.friend.progress.length" show-
value>
                <b-progress-bar     :value="getCorrect(item.friend.progress)"
variant="success"></b-progress-bar>
                <b-progress-bar       :value="getWrong(item.friend.progress)"
variant="danger"></b-progress-bar>
            </b-progress>
        </div>
    </div>
</template>

<script>
    export default {
        name: "Friend",
        props: {
            item: Object
        },
        methods: {
            getProgress(list){
                let count = 0;
                list.forEach(item=>{
                    if(item.attempts != 0)
```

```
                        count++;
                })
                return count;
            },
            getCorrect(list){
                let count = 0;
                list.forEach(item=>{
                    if(item.progress >= 90)
                        count++;
                })
                return count;
            },
            getWrong(list){
                let count = 0;
                list.forEach(item=>{
                    if(item.progress < 90 && item.attempts != 0)
                        count++;
                })
                return count;
            }
        }
    }
</script>

<style scoped>
    .friend-row {
        margin: 1% 0;
    }
    .friend-row span{
        margin-left: 2%;
    }
</style>
```

### B.5.12. Profile/Friends.vue

```
<template>
    <div>
        <h3 class="page-title">Ваші друзі</h3>
        <Friend :item="item" v-for="item in sentRequest" :key="item.id" />
        <Friend :item="item" v-for="item in receivedRequest" :key="item.id" />

        <div>
            <b-button class="btn-lg btn-block mt-4" variant="outline-success"
@click="openUsers">
                Додати
            </b-button>

            <b-modal id="modal-scrollable" scrollable title="Користувачі">
                <div>
                    <p  v-for="item  in  users"  :id="item.id"  :key="item.id"
@click="selectItem($event)">
```

```
                        <b-avatar v-if="item.image == null" variant="success"
size="70px"/>
                        <b-avatar v-else variant="success" :src="item.image"
size="70px"/>
                        <span class="ml-3">{{item.name}}</span>
                        <span> ({{item.id}})</span>
                    </p>
                </div>
                <template #modal-footer>
                    <b-button    class="mt-3"    block    @click="addFriend"
variant="success">Додати</b-button>
                </template>
            </b-modal>
        </div>
    </div>
</template>

<script>
    import Friend from './Friend.vue'
    import API from "@/js/api";

    export default {
        name: "Friends",
        components: {
            Friend
        },
        props: {
            sentRequest: Array,
            receivedRequest: Array
        },
        data () {
            return {
                users: [],
                select: null
            }
        },
        methods: {
            openUsers() {
                API.getUsersList()
                    .then(function (response) {
                        this.users = [];
                        const user = sessionStorage.getItem('user');
                        response.forEach(item => {
                            if(item.id != user)
                                this.users.push(item);
                        })
                    }.bind(this));

                this.$bvModal.show('modal-scrollable');
            },
            selectItem (event) {
```

```
                if(this.select != null)

document.getElementById(this.select).classList.remove("select");

                const id = event.currentTarget.id;
                document.getElementById(id).classList.add("select");
                this.select = id;
            },
            addFriend () {
                API.addFriend(this.select)
                    .then(function (response) {
                        console.log(response);
                        API.getUser()
                            .then(function (response) {
                                this.sentRequest                                =
response.sentFriendRequests;
                            }.bind(this));
                    }.bind(this))
                    .catch(error => {
                        console.log('error: ' + error);
                    });
                this.$bvModal.hide('modal-scrollable');
            }
        }
    }
</script>

<style scoped>
    .select {
        background-color: #ffffe6;
        font-weight: bold;
    }

    #modal-scrollable p:hover{
        cursor: pointer;
        font-weight: bold;
    }
</style>
```

### B.5.13. Profile/Profile.vue

```
<template>
    <div>
        <Navbar/>
        <b-overlay :show="user == null" rounded="sm">
            <div class="container-fluid profile-page">
                <div class="row">
                    <div class="col-7 card-back" v-if="user != null">
                        <UserForm :user="user"/>
                    </div>
```

```
                    <div class="col-4 card-back friends-card" v-if="user != null">
                        <Friends :sentRequest="user.sentFriendRequests" :receivedRequest="user.receivedFriendRequests"/>
                    </div>
                </div>

                <div class="card-back" v-if="user != null">
                    <CardProgress :progress="user.progress"/>
                </div>
            </div>
        </b-overlay>
        <Footer/>
    </div>
</template>

<script>
    import Navbar from '../Navbar.vue'
    import Footer from '../Footer.vue'
    import UserForm from './User-Form.vue'
    import CardProgress from './CardProgress.vue'
    import Friends from './Friends.vue'
    import API from "@/js/api";

    export default {
        name: "Profile",
        components: {
            Navbar,
            Footer,
            UserForm,
            CardProgress,
            Friends
        },
        data() {
            this.loadData();
            return {
                user: null,
            }
        },
        methods: {
            async loadData() {
                API.getUser()
                    .then(function (response) {
                        this.user = response;
                    }.bind(this));

            }

        }
    }
</script>
```

```
<style scoped>
    .profile-page .card-back {
        margin: 1.5% !important;
    }
    .profile-page .friends-card.card-back {
        padding: 3% 0;
    }
</style>
```

### B.5.14. Profile/User-Form.vue

```
<template>
    <div class="row">
        <div class="col-4">
            <b-avatar     v-if="user.image     ==     null"     size="200px"
variant="warning"/>
            <b-avatar       v-else       :src="user.image"       size="200px"
variant="success"/>
            <b-form-file id="file-small" v-model="avatar"
                         size="sm" accept="image/*"
                         placeholder="" drop-placeholder="Перетягніть сюди"
                         browse-text="Змінити"/>
        </div>
        <div class="col-7">
            <b-form-group label="Ім'я" label-for="inputName" class="mb-2">
                <b-form-input id="inputName" v-model="name"
                              :state="validation" required />
            </b-form-group>

            <b-form-group label="Email" label-for="inputEmail" class="mb-2">
                <b-form-input id="inputEmail" v-model="email" disabled/>
            </b-form-group>

            <small><a>Змінити пароль?</a></small>

            <b-button  disabled  class="btn-lg  btn-block"  variant="outline-
success" @click="editProfile">
                Підтвердити зміни
            </b-button>
        </div>
    </div>
</template>

<script>
    export default {
        name: "User-Form",
        props: {
            user: Object
        },
        data () {
```

```
            return {
                avatar: null,
                name: this.user.name,
                email: this.user.id
            }
        },
        computed: {
            validation() {
                return this.name != ''
            }
        },
        methods: {
            editProfile() {
                return true;
            }
        }
    }
</script>

<style scoped>
</style>
```

### B.5.15. TestPage/Comment.vue

```
<template>
    <div>
        <div :class="'comment-card mb-3 ml-'+(3+step)">
            <div class="comment-header row">
                <div class="col-4">
                    <b-avatar  v-if="comment.image  ==  null"  variant="success"
size="50px"/>
                    <b-avatar  v-else  variant="success"  :src="comment.image"
size="50px"/>
                    <span class="ml-3">{{comment.login}}</span>
                </div>
                <span class="col-4">Тест №{{comment.testId}}</span>
                <span class="col-4">{{parseDateTime(comment.dateTime)}}</span>
            </div>
            <hr/>
            <p>{{comment.text}}</p>
            <div class="m-4">
                <div align="center">
                    <b-button              v-b-toggle="'collapse-'+comment.id"
variant="link">Відповісти</b-button>
                </div>
                <b-collapse :id="'collapse-'+comment.id" class="mt-2">
                    <b-card>
                        <CommentForm                    :test="comment.testId"
:previous="comment.id" ref="form"/>
                        <div align="center">
                            <b-button                      @click="addComment"
variant="warning">Відправити</b-button>
```

```
                        </div>
                    </b-card>
                </b-collapse>
            </div>
        </div>

        <div>
            <Comment    :comment="item"    :step="step+2"    v-for="item    in
comment.next" :key="item.id"/>
        </div>
    </div>
</template>

<script>
    import Comment from '@/components/TestPage/Comment.vue'
    import CommentForm from '@/components/TestPage/Comment-Form.vue'

    export default {
        name: "Comment",
        components: {
            Comment,
            CommentForm
        },
        props: {
            comment: Object,
            step: Number
        },
        methods:{
            parseDateTime(dateTime) {
                return dateTime.replace('T', ' ').slice(0, dateTime.length-3);
            },
            addComment() {
                this.$refs.form.onSubmit()
                    .then((response) => {
                        this.comment.next.push(response)
                    }).catch(error => {
                    console.log(error);
                });
            }
        }

    }
</script>

<style scoped>
    .comment-card {
        background: whitesmoke;
        border-radius: 15px;
        max-width: 600px;
        padding: 2%;
```

```
        font-size: 14px !important;
    }

    .comment-card .comment-header {
        width: 100%;
        font-weight: bold;
    }
</style>
```

### B.5.16. TestPage/Comment-Form.vue

```
<template>
    <div>
        <b-form-group  label="Номер тесту" label-for="inputTest" class="mb-2">
            <b-form-input id="inputTest" v-model="testId"/>
        </b-form-group>
        <b-form-group label="Email" label-for="inputEmail" class="mb-2">
            <b-form-input id="inputEmail" v-model="user" disabled/>
        </b-form-group>

        <b-form-group label="Коментар" label-for="inputText" class="mb-2">
            <b-form-input id="inputText" v-model="commentText" required/>
        </b-form-group>
    </div>
</template>

<script>
    import API from '@/js/api'
    export default {
        name: "Comment-Form",
        props: {
            test: Number,
            previous: Number
        },
        data (){
            return {
                testId: this.test,
                commentText: '',
                user: sessionStorage.getItem('user')
            }
        },
        methods: {
            onSubmit(){
                const comment = {
                    text: this.commentText,
                    testId: parseInt(this.testId),
                    email: this.user,
                    previousId: this.previous
                };
                return  API.addComment(comment)
```

```
            }
        }
    }
</script>

<style scoped>
</style>
```

### B.5.17. TestPage/ Test-Page.vue

```
<template>
    <div>
     <Navbar/>
     <div class="container-fluid">
     <div class="row">

         <div class="col-3 left-nav">
         <div class="accordion filter" id="accordionExample">
         <div>
            <h2 id="headingOne" class="mb-0 mt-2">
            <li       class="nav-link"       data-toggle="collapse"       data-
target="#collapseOne"
                                    aria-expanded="true"                aria-
controls="collapseOne">
            Тести за темами &#11177;
            </li>
            </h2>

         <div      id="collapseOne"      class="collapse      show"      aria-
labelledby="headingOne" data-parent="#accordionExample">
             <li class="nav-link" :id="item.id" @click="openTopicTest($event)"
                                    v-for="item in topics" :key="item.id">
                                    {{ item.id }}. {{ item.title }}
              </li>
        </div>
        </div>
        <hr/>
        <div>
              <h2 id="headingTwo" class="mb-0 mt-2">
              <li class="nav-link" type="button" data-toggle="collapse"
                                    data-target="#collapseTwo"          aria-
expanded="false"
                                    aria-controls="collapseTwo">
               Тести за білетами &#11177;
              </li>
              </h2>
        <div id="collapseTwo" class="collapse" aria-labelledby="headingTwo"
                                data-parent="#accordionExample">
             <li          class="nav-link          ml-3"          :id="item.id"
@click="openCardTest($event)"
                                    v-for="item in cards" :key="item.id">
```

```
                Білет {{ item.id }}
              </li>
            </div>
            </div>
           </div>
          </div>

          <div class="col-9 p-0">
          <b-overlay :show="show" rounded="sm">
          <div class="test-page">
              <h3 class="page-title">Іспит з ПДР України 2021</h3>

                  <div id="choose-test" class="card-back">
                     <div v-if="tests.length == 0">
                     <p>
                     Щоб розпочати тестування, оберіть тест з певної категорії.
                     </p>
                     <p>
                     <b>Майте на увазі</b>, що якщо ви оберете тестування за
білетом, ваш результат буде збережено у ваш прогрес. Тестування за темами є
тренувальним і його результат не буде зберігатись.
                     </p>
                     <p>
                     Як тільки оберете категорію, з'виться тест і запуститься
таймер. Час тестування <b>10 хв</b>.
                      </p>
                  </div>


                                  <div class="test-block" :id="'topic'+item.id"
v-for="(item, index) in tests" :key="item.id">
                                      <p      class="test-title">Білет      №{{
item.cardId }} Тест №{{ item.id }}</p>
                                      <p>{{ item.question }}</p>

                                      <div class="row">
                                          <div    v-if="item.image    !=    null"
class="col-md-6 col-lg-5">
                                              <img             :src="item.image"
alt="image"/>
                                          </div>
                                          <div class="col-md-8 col-lg-7">
                                              <b-form-group v-slot="{variants}">
                                                  <b-form-radio              v-
model="res[index]" :aria-describedby="variants" :id="item.id+'-1'"

:name="item.id+''" value="1">{{item.variant1}}
                                                  </b-form-radio>
                                                  <b-form-radio              v-
model="res[index]" :aria-describedby="variants" :id="item.id+'-2'"

:name="item.id+''" value="2">{{item.variant2}}
                                                  </b-form-radio>
```

```
                                                <b-form-radio                v-
if="item.variant3 != null" v-model="res[index]" :aria-describedby="variants"
                                                                  :id="item.id+'-
3'" :name="item.id+''" value="3">{{item.variant3}}
                                                </b-form-radio>
                                                <b-form-radio                v-
if="item.variant4 != null" v-model="res[index]" :aria-describedby="variants"
                                                                  :id="item.id+'-
4'" :name="item.id+''" value="4">{{item.variant4}}
                                                </b-form-radio>
                                            </b-form-group>
                                        </div>
                                    </div>
                                    <hr/>
                                </div>

                                <div>
                                    <b-button    v-if="tests.length    !=    0"
@click="checkAnswers" block variant="warning">Перевірити</b-button>
                                    <b-modal id="test-result" hide-footer>
                                        <template #modal-title>
                                            Ваш результат
                                        </template>
                                        <template #modal-header-close>
                                            <b-button-close
@click="saveProgress"/>
                                        </template>
                                        <div class="d-block text-center">
                                            <Progress :value="getProgress" />
                                            <hr/>
                                            <p>Правильно: {{ right }}</p>
                                            <p>Помилок: {{ wrong }}</p>
                                            <p>Час:                          {{
formattedTime(timeLimit-timer) }}</p>
                                        </div>
                                    </b-modal>
                                </div>

                            </div>

                            <div v-if="tests.length != 0" id="timer-block">
                                <span>Залишилось часу: </span>
                <span class="timer-numbers">{{ formattedTime(timer) }}</span>
                            </div>
                        </div>

                        <div class="comment-page">
                            <hr/>
                            <h3 class="page-title">Коментарі</h3>
                            <div class="m-4">
                      <b-button v-b-toggle.collapse-1  variant="link">Залишити
коментар</b-button>
```

```
                                <b-collapse id="collapse-1" class="mt-2">
                                    <b-card>
                                        <CommentForm              :test="null"
:previous="null" ref="form"/>
                                        <div align="center">
                                            <b-button      @click="addComment"
variant="warning">Відправити</b-button>
                                        </div>
                                    </b-card>
                                </b-collapse>
                            </div>

                            <div v-for="item in tests" :key="item.id">
                                <div    v-for="comment    in    item.comments"
:key="comment.id">
                                    <Comment v-if="comment.previousId == null"
:comment="comment" :step="0"/>
                                </div>
                            </div>
                        </div>
                    </b-overlay>
                </div>
            </div>
        </div>

        <Footer />
    </div>
</template>

<script>
    import Navbar from '../Navbar.vue'
    import Footer from '../Footer.vue'
    import API from '@/js/api'
    import worker from "@/js/workers";
    import Progress from '@/components/Progress-Round';
    import Comment from '@/components/TestPage/Comment.vue'
    import CommentForm from '@/components/TestPage/Comment-Form.vue'

    export default {
        name: "TestPage",
        components: {
            Navbar,
            Footer,
            Progress,
            Comment,
            CommentForm
        },
        data() {
            return {
                topics: [],
                cards: [],
```

```
        tests: [],
        res: [],
        show: false,

        wrong: -1,
        right: 0,

        timeLimit: 600,
        timer: 0,

        id_card: 0
    }
},
computed: {
    getProgress() {
        return Number((this.right*100 / this.tests.length).toFixed(1))
    }
},
mounted() {
    API.getTopicsList()
        .then(function (response) {
            this.topics = response;
        }.bind(this));

    API.getCardList()
        .then(function (response) {
            this.cards = response;
        }.bind(this));
},
methods: {
    openTopicTest(event) {
        this.id_card = 0;
        this.openTest('topicId', event.currentTarget.id);
    },
    openCardTest(event) {
        this.id_card = event.currentTarget.id;
        this.openTest('cardId', event.currentTarget.id);
    },
    openTest(criterion, id){
        this.show = true;

        API.getTestList()
            .then(function (response) {
                this.filterWorker(response, criterion, id);
                this.startTest();
            }.bind(this));
    },
    startTest(){
        this.res = [];
        this.show = false;
```

```
        this.timer = 0;
        worker.startTimerWorker();
        worker.getTimerWorker().onmessage = (event) => {
            this.timer = event.data;
            if (this.timer % this.timeLimit === 0)
                this.checkAnswers();
        }

        worker.sendTimer({
            limit: this.timeLimit
        })

    },
    filterWorker(test_list, criterion, filterValue) {
        this.tests = [];

        let worker1 = worker.getFilterWorker();
        let worker2 = worker.getFilterWorker();

        worker1.onmessage = (event) => {
            this.tests.push(event.data);
        }

        worker2.onmessage = (event) => {
            this.tests.push(event.data);
        }

        const half = Math.ceil(test_list.length / 2);
        worker1.postMessage({
            test_list: test_list.splice(0, half),
            criterion: criterion,
            filterValue: filterValue
        })

        worker2.postMessage({
            test_list: test_list.splice(-half),
            criterion: criterion,
            filterValue: filterValue
        })
    },
    checkAnswers() {
        worker.stopTimerWorker();

        this.right = 0;
        this.wrong = 0;

        let worker1 = worker.getTestWorker();

        worker1.onmessage = (event) => {
```

```
            let test = event.data.test;
            let elem = document.getElementById('topic' + test.id);
            if (event.data.flag == 1){
                this.right++;
                elem.classList.add("correct");
            }
            else {
                this.wrong++;
                elem.classList.add("wrong");
            }
        }

        worker1.postMessage({
            test_list: this.tests,
            answers_list: this.res
        })

        this.$bvModal.show('test-result');
    },
    formattedTime(time) {
        let minutes = Math.floor(time / 60)
        if (minutes < 10)
            minutes = '0'+minutes
        let seconds = time % 60
        if (seconds < 10)
            seconds = '0'+seconds
        // The output in MM:SS format
        return minutes+':'+seconds
    },
    saveProgress(){
        if(this.id_card != 0){
            API.updateProgress({
                cardId: this.id_card,
                correct: this.right,
                wrong: this.wrong,
                progress: this.getProgress
            });
        }
    },
    addComment(){
        this.$refs.form.onSubmit()
            .then((response) => {
                this.tests.forEach(item => {
                    if (item.id == response.testId)
                        item.comments.push(response)
                })
            }).catch(error => {
                console.log(error);
            });
    }
```

```
        }
    }
</script>

<style scoped>
    .left-nav {
        position: sticky;
        top: 0;
        overflow-y: hidden;
        background-color: #454d54 !important;
        padding: 0;
    }

    .filter {
        align-items: flex-start;
        justify-content: flex-start;
        flex-wrap: nowrap;
        flex-direction: column;
        height: 100%;
        background-color: #454d54 !important;
    }

    #collapseOne .nav-link, #collapseTwo .nav-link {
        font-size: 14px;
    }


    #choose-test {
        min-height: 350px;
    }

    .test-block {
        padding: 1%;
    }

    .test-block img {
        width: 100%;
        margin: 1%;
    }

    .test-block .test-title {
        font-weight: bold;
    }

    .test-block p {
        margin: 2% 0;
    }

    #timer-block {
        z-index: 100;
```

```
        position: sticky;
        bottom: 0;
        right: 0;
        height: 70px;
        width: 100%;
        background: #343a40;
        color: white;
        border: 1px solid #454d54;

        font-family: 'UbuntuMono';
        font-weight: bold;
        font-size: 25px;
        text-align: center;
        padding: 1%;
    }

    .correct {
        background: #ccffcc;
    }

    .wrong {
        background: #ffb3b3;
    }
</style>
```

## B.6. js

### B.6.1. api/index.js

```
import axios from 'axios';
import router from '@/js/router'

const baseURL = 'https://svitlofor-server.herokuapp.com'
let tokenKey = 'accessToken'

const login = (email, password) => {
    return new Promise((resolve, reject) => {
        axios.post(baseURL + '/api/User/login', {
            username: email,
            password: password
        })
            .then((response) => {
                resolve(response.data);
            })
            .catch(error => {
                reject(error.response.data);
            });
    });
}

const registration = (email, name, password) => {
```

```
    return new Promise((resolve, reject) => {
        axios.post(baseURL + '/api/User', {
            id: email,
            name: name,
            password: password
        })
            .then((response) => {
                resolve(response.data);
            })
            .catch(error => {
                reject(error.response.data);
            });
    });
}

const getFines = () => {
    return new Promise((resolve, reject) => {
        axios.get(baseURL + '/api/Fine', {
            headers: {
                "Authorization": "Bearer " + sessionStorage.getItem(tokenKey)
            }
        })
            .then((response) => {
                resolve(response.data);
            })
            .catch(error => {
                if(error.response.status == 401)
                    router.push({name: 'authorization'});
                reject(error.response.data);
            });
    });
}

const getTopicsList = () => {
    return new Promise((resolve, reject) => {
        axios.get(baseURL + '/api/Topic', {
            headers: {
                "Authorization": "Bearer " + sessionStorage.getItem(tokenKey)
            }
        })
            .then((response) => {
                resolve(response.data);
            })
            .catch(error => {
                if(error.response.status == 401)
                    router.push({name: 'authorization'});
                reject(error.response.data);
            });
    });
}
```

```
const getTopic = (id) => {
    return new Promise((resolve, reject) => {
        axios.get(baseURL + '/api/Topic/' + id, {
            headers: {
                "Authorization": "Bearer " + sessionStorage.getItem(tokenKey)
            }
        })
            .then((response) => {
                resolve(response.data);
            })
            .catch(error => {
                if(error.response.status == 401)
                    router.push({name: 'authorization'});
                reject(error.response.data);
            });
    });
}

const getCardList = () => {
    return new Promise((resolve, reject) => {
        axios.get(baseURL + '/api/Card', {
            headers: {
                "Authorization": "Bearer " + sessionStorage.getItem(tokenKey)
            }
        })
            .then((response) => {
                resolve(response.data);
            })
            .catch(error => {
                if(error.response.status == 401)
                    router.push({name: 'authorization'});
                reject(error.response.data);
            });
    });
}

const getTestList = () => {
    return new Promise((resolve, reject) => {
        axios.get(baseURL + '/api/Test', {
            headers: {
                "Authorization": "Bearer " + sessionStorage.getItem(tokenKey)
            }
        })
            .then((response) => {
                resolve(response.data);
            })
            .catch(error => {
                if(error.response.status == 401)
                    router.push({name: 'authorization'});
```

```
                reject(error.response.data);
            });
    });
}

const getUser = () => {
    return new Promise((resolve, reject) => {
        axios.get(baseURL + '/api/User/' + sessionStorage.getItem('user'), {
            headers: {
                "Authorization": "Bearer " + sessionStorage.getItem(tokenKey)
            }
        })
            .then((response) => {
                resolve(response.data);
            })
            .catch(error => {
                if(error.response.status == 401)
                    router.push({name: 'authorization'});
                reject(error.response.data);
            });
    });
}

const updateProgress = (progress) => {
    let id = progress.cardId +''+ sessionStorage.getItem('user')
    return new Promise((resolve, reject) => {
        axios.get(baseURL + '/api/CardProgress/' + id, {
            headers: {
                "Authorization": "Bearer " + sessionStorage.getItem(tokenKey)
            }
        })
            .then((response) => {

                let userProgress = response.data;
                userProgress.correct = progress.correct;
                userProgress.wrong = progress.wrong;
                userProgress.attempts = userProgress.attempts + 1;
                userProgress.progress = progress.progress;

                axios.put(baseURL + '/api/CardProgress/' + id, userProgress, {
                    headers: {
                        "Authorization":          "Bearer          "          +
sessionStorage.getItem(tokenKey)
                    }
                })
                    .then((response) => {
                        resolve(response.data);
                    })
                    .catch(error => {
                        if(error.response.status == 401)
```

```
                            router.push({name: 'authorization'});
                        reject(error.response.data);
                    });
            })
            .catch(error => {
                if(error.response.status == 401)
                    router.push({name: 'authorization'});
                reject(error.response.data);
            });
    });
}


const addComment = (comment) => {
    return new Promise((resolve, reject) => {
        axios.post(baseURL + '/api/Comment', comment, {
            headers: {
                "Authorization": "Bearer " + sessionStorage.getItem(tokenKey)
            }
        })
            .then((response) => {
                resolve(response.data);
            })
            .catch(error => {
                if(error.response.status == 401)
                    router.push({name: 'authorization'});
                reject(error.response.data);
            });
    });
}


const getUsersList = () => {
    return new Promise((resolve, reject) => {
        axios.get(baseURL + '/api/User', {
            headers: {
                "Authorization": "Bearer " + sessionStorage.getItem(tokenKey)
            }
        })
            .then((response) => {
                resolve(response.data);
            })
            .catch(error => {
                if(error.response.status == 401)
                    router.push({name: 'authorization'});
                reject(error.response);
            });
    });
}
```

```
const addFriend = (id) => {
    const friends = {
        requestedById: sessionStorage.getItem('user'),
        requestedToId: id
    };
    return new Promise((resolve, reject) => {
        axios.post(baseURL + '/api/Friends', friends, {
            headers: {
                "Authorization": "Bearer " + sessionStorage.getItem(tokenKey)
            }
        })
            .then((response) => {
                resolve(response.data);
            })
            .catch(error => {
                if(error.response.status == 401)
                    router.push({name: 'authorization'});
                reject(error.response.data);
            });
    });
}

export default {
    tokenKey,
    baseURL,
    login,
    registration,
    getFines,
    getTopicsList,
    getTopic,
    getCardList,
    getTestList,
    getUsersList,
    getUser,
    updateProgress,
    addFriend,
    addComment
}
```

### B.6.2. router/index.js

```
import Vue from 'vue'
import VueRouter from 'vue-router'

Vue.use(VueRouter);

const routes = [
    {
        path: '/',
        name: 'authorization',
```

```
        component:                              ()                              =>
import('../../components/Authorization/Authorization.vue')
    },
    {
        path: '/registration',
        name: 'registration',
        component:                              ()                              =>
import('../../components/Authorization/Registration.vue')
    },
    {
        path: '/main',
        name: 'main',
        component: () => import('../../components/Main.vue')
    },
    {
        path: '/tests',
        name: 'tests',
        component: () => import('../../components/TestPage/Test-Page.vue')
    },
    {
        path: '/fines',
        name: 'fines',
        component: () => import('../../components/PDR/Fines-Page.vue')
    },
    {
        path: '/pdr',
        name: 'pdr',
        component: () => import('../../components/PDR/PDR-Page.vue')
    },
    {
        path: '/topic/:topicId',
        name: 'topic',
        component: () => import('../../components/PDR/Topic.vue')
    },
    {
        path: '/profile',
        name: 'profile',
        component: () => import('../../components/Profile/Profile.vue')
    }
]

const router = new VueRouter({
    mode: 'history',
    base: process.env.BASE_URL,
    routes
})

export default router
```

### B.6.3. workers/index.js

```
let timerWorker;

const startTimerWorker = () => {
    if(typeof(timerWorker) != "undefined")
        stopTimerWorker();

    timerWorker = new Worker('./timerWorker.js', { type: 'module' });
}

const getTimerWorker = () => {
    return timerWorker
}

const sendTimer = message =>
    timerWorker.postMessage({
        message
    })

const stopTimerWorker = () => {
    if(typeof(timerWorker) != "undefined"){
        timerWorker.terminate();
        timerWorker = undefined;
    }
}

const getFilterWorker = () => {
    return new Worker('./filterWorker.js', { type: 'module' });
}

const getTestWorker = () => {
    return new Worker('./checkTestWorker.js', { type: 'module' });
}

export default {
    getTimerWorker,
    startTimerWorker,
    sendTimer,
    stopTimerWorker,
    getFilterWorker,
    getTestWorker
}
```

### B.6.4. workers/checkTestWorker.js

```
onmessage = (event) => {
    let test_list = event.data.test_list;
    let answers_list = event.data.answers_list;

    for (let i = 0; i < test_list.length; i++) {
```

```
        if (answers_list[i] == null || answers_list[i] != test_list[i].correct)
            postMessage({
                test: test_list[i],
                flag: 0
            });
        else
            postMessage({
                test: test_list[i],
                flag: 1
            });
    }
}
```

### B.6.5. workers/filterWorker.js

```
onmessage = (event) => {
    let test_list = event.data.test_list;
    let criterion = event.data.criterion;
    let filterValue = event.data.filterValue;

    test_list.forEach(item => {
        if (item[criterion] == filterValue)
            postMessage(item)
    })
}
```

### B.6.6. workers/timerWorker.js

```
onmessage = (event) => {
    let timeLimit = event.data.message.limit;

    setInterval(() => {
        timeLimit--;
        postMessage(timeLimit);
    }, 1000)
}
```