Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для організації практичних занять в автошколі. Frontend
(тема)

Виконала:
студентка 4 курсу, групи ПЗПІ 20-4

Ляшенко М.А.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Олійник О.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ляшенко Марії Андріївні
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для організації практичних занять в автошколі. Frontend

Затверджена наказом по університету від 20.05. 2024р. № 471 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20.06.2024

3. Вихідні дані до роботи Розробити frontend частину додатку для комплексної програмної системи, яка призначена для організації практичних занять в автошколі. Використовувати javascript та бібліотеку react.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, перелік джерел посилань, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	21.05.2024	*виконано*
2	Створення специфікації ПЗ	22.05.2024	*виконано*
3	Проектування ПЗ	24.05.2024	*виконано*
4	Розробка ПЗ	08.05.2024	*виконано*
5	Тестування ПЗ	09.05.2024	*виконано*
6	Оформлення пояснювальної записки	11.06.2024	*виконано*
7	Підготовка презентації та доповіді	13.06.2024	*виконано*
8	Попередній захист	14.06.2024	*виконано*
9	Нормоконтроль, рецензування	15.06.2024	*виконано*
10	Здача роботи у електронний архів	16.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	19.06.2024	*виконано*

Дата видачі завдання 21 травня 2024р.

Студент (ка)_____ Ляшенко М.А.

(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Олійник О.В.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 78 стор., 48 рис., 1 табл., 10 джерел.

АВТОМАТИЗАЦІЯ, АВТОШКОЛА, ІНСТРУКТОРИ, НАВЧАННЯ, СТУДЕНТИ АВТОШКОЛИ, FRONT END, I18NEXT, REACT, REDUX, SCSS.

Об’єкт розробки – веб сайт для автоматизації проведення практичних занять в автошколі.

Мета розробки – створення простого та зрозумілого сайту для комфортного навчання практичним заняттям в автошколах як учням так і інструкторам.

Методи рішення завдання – використання бібліотеки React, scss для написання стилей. Додаток взаємодіє з бекендом через HTTP-запити для синхронізації та оновлення інформації.

Результатом цієї роботи є створений веб-сайт, який враховує потреби як інструкторів, так і студентів автошколи. Для інструкторів на сайті доступні інструменти для управління та планування практичних занять, відстеження прогресу учнів та доступ до відповідних ресурсів і матеріалів. Студенти, у свою чергу, можуть користуватися зручним інтерфейсом для перегляду розкладу занять, доступу до навчальних матеріалів та відстеження свого навчального прогресу.

AUTOMATION, DRIVING SCHOOL, INSTRUCTORS, TRAINING, DRIVING SCHOOL STUDENTS, FRONT END, I18NEXT, REACT, REDUX, SCSS.

Development object: a website for automating practical lessons at driving schools.

Development goal: to create a simple and understandable website for comfortable learning of practical lessons in driving schools for both students and instructors.

Solution methods: using the React library, SCSS for styling. The application interacts with the backend through HTTP requests for synchronization and updating of information.

The outcome of this project is the development of a website tailored to meet the needs of both driving instructors and students. For instructors, the website provides tools for managing and scheduling practical lessons, tracking student progress, and accessing relevant resources and materials. Students, on the other hand, benefit from a user-friendly interface for viewing their lesson schedules, accessing study materials, and tracking their learning progress.

Я, Ляшенко Марія Андріївна, студентка гр. ПЗПІ-20-4, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для організації практичних занять в автошколі. Frontend, що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Оскільки віднедавно практичний іспит може складатися тільки після успішного проходження теоретичної частини, впровадження сучасних технологій у практичне навчання в автошколах стає не тільки актуальним, але й необхідним кроком на шляху до підвищення якості освіти водіїв [1]. В умовах постійно зростаючих вимог до безпеки дорожнього руху, важливо забезпечити максимально ефективне та зручне навчання майбутніх водіїв. Серед інноваційних підходів можна виділити розробку програмних систем, які інтегруються з реальним досвідом водіння та надають структуроване та цілісне освітнє середовище.

На сьогоднішній день існують різноманітні додатки та програмні засоби для водіння, але часто вони обмежуються лише теоретичною підготовкою або базовими практичними вправами. Відсутність глибокої інтеграції практичного навчання з цифровими інструментами є прогалиною, яку наш проект прагне заповнити.

Метою даного проекту є створення програмної системи для автошкіл, що використовує сучасні технології для оптимізації практичних занять. Ця система буде сприяти покращенню навчального процесу через інтерактивність, адаптивність і візуалізацію, забезпечуючи більш глибоке і занурене засвоєння матеріалу. В рамках дослідження було проаналізовано існуючі програмні рішення, спроектована інноваційна архітектура системи та розроблені алгоритми для вирішення специфічних завдань, які стоять перед сучасною автошколою. Завершенням проекту стало створення функціонального прототипу, який вже демонструє здатність до зміни стандартів навчання водіїв.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Тема якісної підготовки водіїв залишається завжди актуальною у суспільному обговоренні. Кожен учасник дорожнього руху прагне бути безпечним та компетентним на дорозі. Останнім часом ця тема стала особливо популярною, оскільки безпека дорожнього руху стає пріоритетом у багатьох країнах. Особливо в Україні часто виникають проблеми з електропостачанням та громадським транспортом, що унеможливлює комфортне переміщення по місту.

У світі, де технології постійно розвиваються, соціальні медіа та інтернет сайти стали невід'ємною частиною нашого життя. Вони допомагають не тільки ділитися особистим досвідом, але й використовуватися в навчальних цілях, зокрема у підготовці водіїв.

На сьогодні існує багато програмних засобів для теоретичного навчання в автошколах, проте практична підготовка часто залишається традиційною і може не включати всі можливості, які надають сучасні технології. Наприклад, автоматизація запису на урок або бронювання чи відміна уроку через сайт могло б значно підвищити ефективність навчання та безпеку під час навчальних занять. Також відстеження прогресу учня за допомогою спеціалізованих додатків може допомогти інструкторам краще адаптувати навчальний процес до індивідуальних потреб учнів.

Провідними методами підтримки ефективної практичної підготовки є:

- використання інтерактивних симуляторів для відпрацювання складних маневрів в безпечному середовищі;
- застосування збору даних про процес навчання та адаптації курсів відповідно до потреб студентів;
- впровадження гнучких навчальних планів, що дозволяють студентам повноцінно готуватися до викликів сучасного дорожнього руху.

Розробка інноваційної програмної системи для автошкіл, яка б включала ці технології, могла б кардинально змінити підхід до навчання водіїв, зробити його

більш сучасним, безпечним та ефективним. В рамках даного проекту було проведено дослідження наявних методів підготовки, розроблені спеціалізовані алгоритми для вирішення конкретних завдань навчання водіння, а також створено прототип програмної системи, яка вже демонструє свій потенціал у підвищенні кваліфікації водіїв.

1.2 Аналіз ринку

У сучасному світі автошколи грають ключову роль у підготовці відповідальних та кваліфікованих водіїв. Важливість надання якісної освіти в цій сфері не може бути переоцінена, адже безпека на дорогах безпосередньо залежить від майстерності та обізнаності водіїв. Вибір правильної автошколи є критично важливим, а їхня презентація через інтернет стає першим кроком у взаємодії між школою та потенційними студентами. Сучасні технології дозволяють автошколам не тільки інформувати про свої послуги, а й спрощувати процеси реєстрації та навчання через інтеграцію електронних систем запису та оплати. Аналіз веб-сайтів трьох автошкіл в Україні демонструє, як цифрові рішення можуть впливати на зручність та доступність навчальних послуг.

Сайт автошколи «Україна».

При відвідуванні нашого сайту ви можете одразу побачити основне меню нашої автошколи, яке містить всі доступні розділи, такі як "Про нас", "Навчання", "Записатися", "Галерея" та "Контакти" (рис 1.1). Розділ "Про нас" надасть вам можливість отримати більш детальну інформацію про нашу автошколу (рис 1.2). В розділі "Навчання" ви можете ознайомитися з категоріями навчання та отримати інформацію щодо теоретичного та практичного навчання. Тут також представлена інформація про навчальну літературу, включаючи рекомендації щодо посібників, рекомендованих нашою автошколою (рис 1.3).

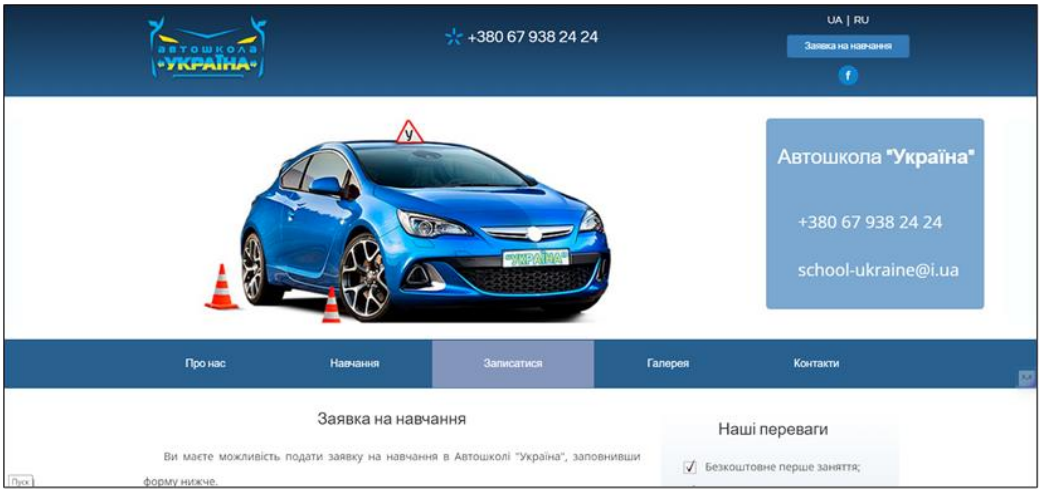


Рисунок 1.1 – Початкова сторінка

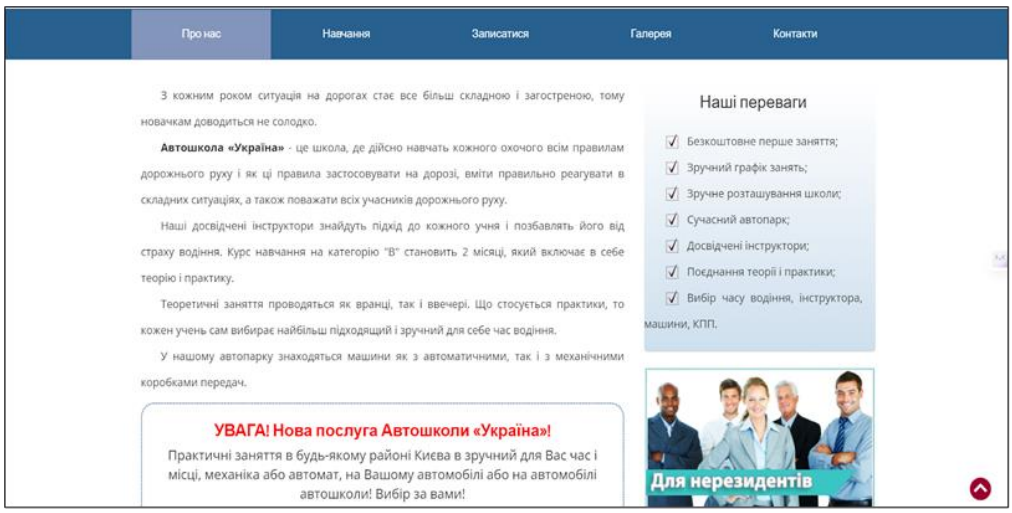


Рисунок 1.2 – Сторінка «Про нас»

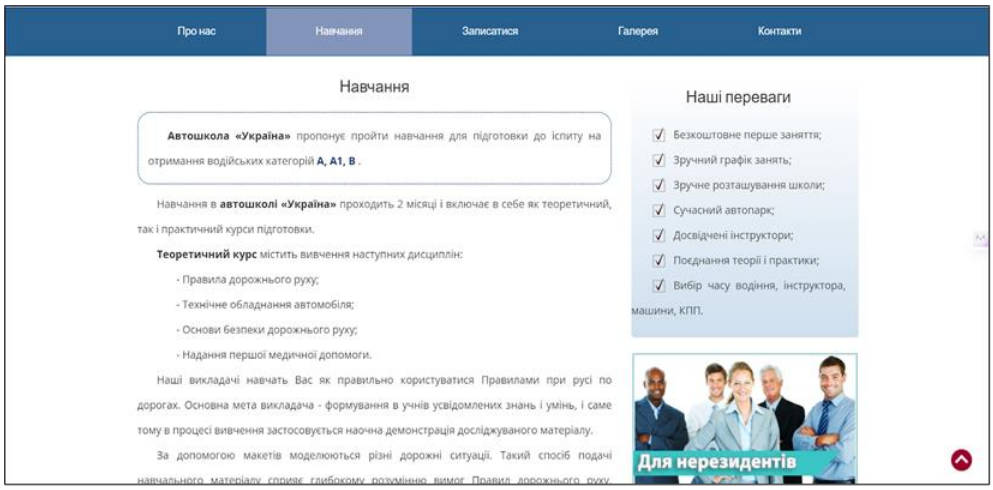


Рисунок 1.3 – Сторінка «Навчання»

На вкладці "Записатися" доступна спеціальна форма для реєстрації на навчання, яка дає можливість користувачам вибирати коробку передач та графік занять, що є надзвичайно зручним для наших клієнтів (рис 1.4). Крім того, на вкладці "Галерея" ви зможете оглянути фотографії наших навчальних класів та автомобілів, на яких проводиться навчання (рис 1.5). Переходячи до розділу "Контакти", ви зможете заповнити форму для зворотного зв'язку і дізнатися наші контактні номери телефонів, за допомогою яких ви можете зв'язатися з нами (рис 1.6).

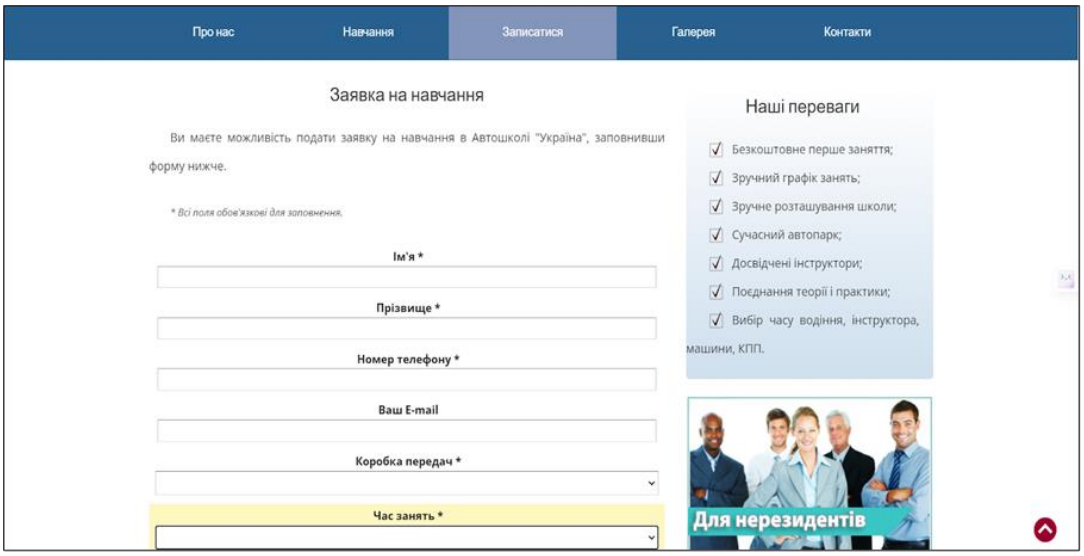


Рисунок 1.4 – Сторінка “Записатися”

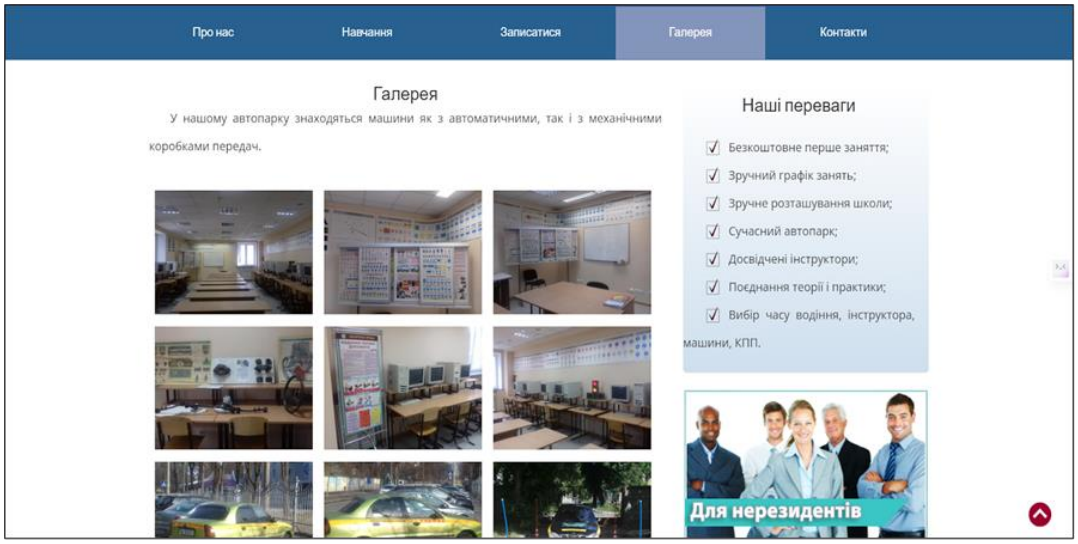


Рисунок 1.5 – Сторінка “Галерея”

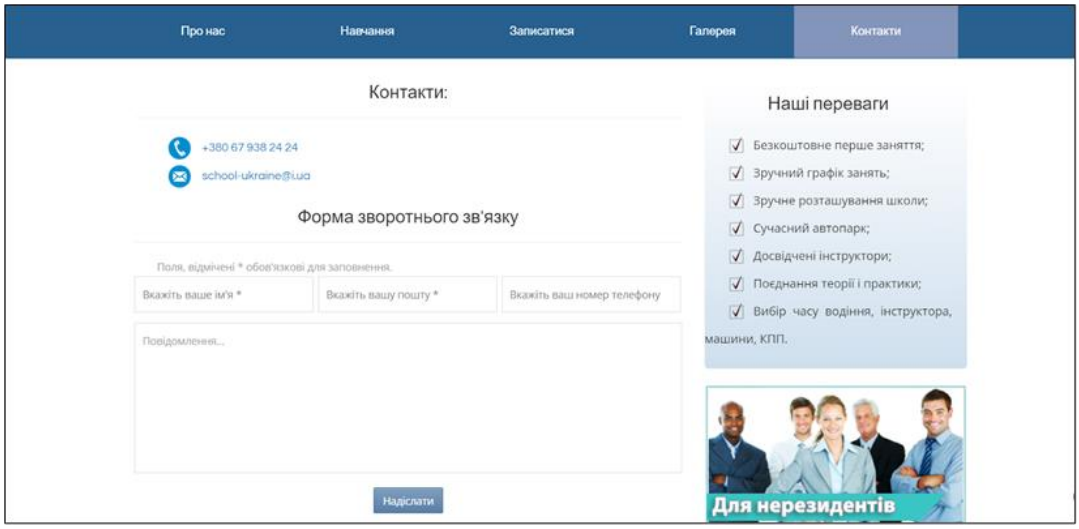


Рисунок 1.6 – Сторінка “Контакти”

Підсумувавши можна виділити позитивні та негативні характеристики.

Позитивні характеристики автошколи:

- зручний інтерфейс веб-сайту з легким доступом до інформації;
- наявність детальної інформації про автошколу та навчальні програми;
- можливість перегляду фотографій навчальних класів та автомобілів;
- легкий спосіб зв'язку з автошколою через форму зворотнього зв'язку та контактні номери телефонів.

Негативні характеристики автошколи:

- відсутність можливості обирати конкретного інструктора для навчання, що може бути важливим для багатьох студентів, які бажають мати певного інструктора або отримувати рекомендації щодо вибору;
- не наявність системи онлайн бронювання практичних занять ускладнює процес реєстрації та може вимагати більше часу та зусиль від клієнтів.

Жіноча школа водіння «АвтоЛеді».

Зайшовши на сайт одразу попадаємо на Головну сторінку. На ній можна побачити основні переваги цієї автошколи, ознайомитися з частими питаннями, подивитися відгуки на цю автошколу, а також перелік міст де можна знайти автошколу (рис 1.8).

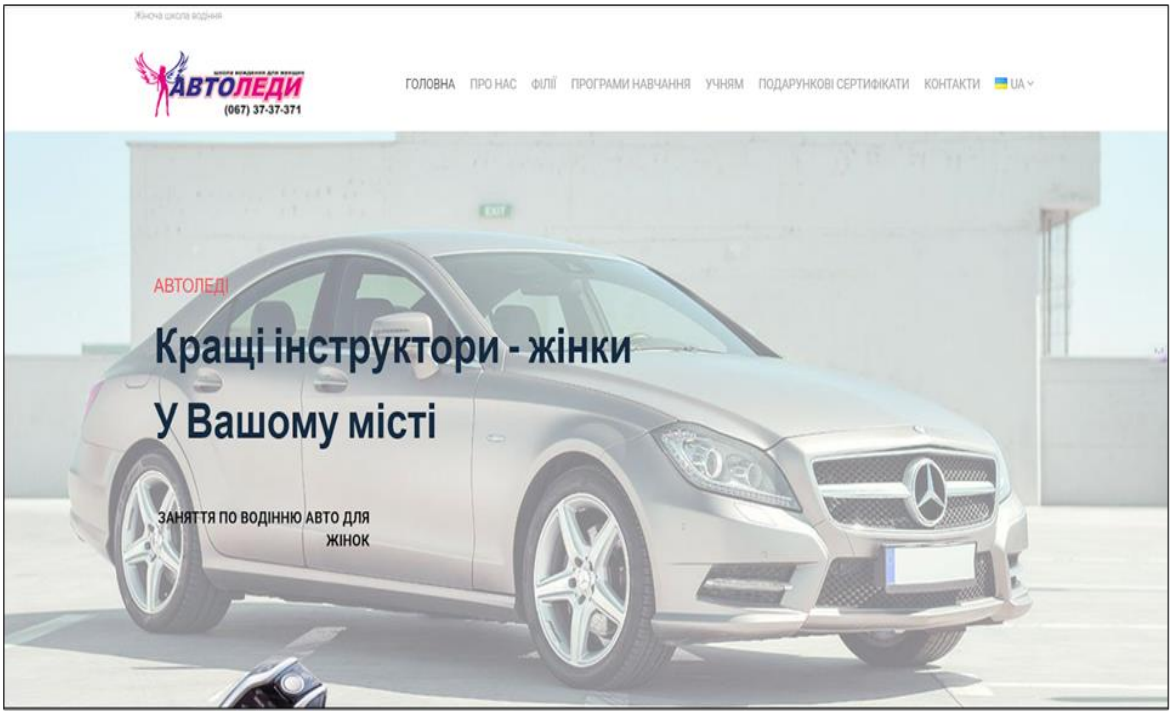


Рисунок 1.8 – Сторінка «Головна»

На вкладці «Про нас» можна прочитати більш детальну інформацію про автошколу (рис 1.9).

Рисунок 1.9 – Сторінка «Про нас»

На вкладці «Філія» можна прочитати більш детальну інформацію про автошколу в кожному місті України де вони розташовані (рис 1.10).

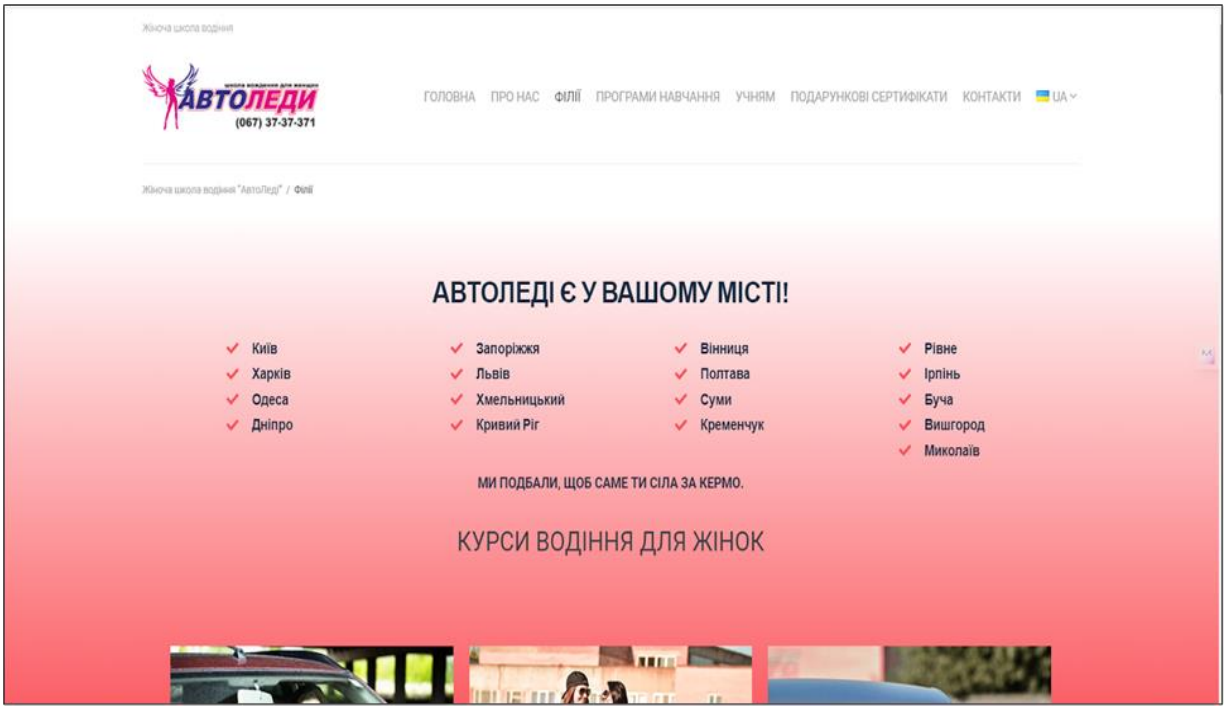


Рисунок 1.10 – Сторінка «Філія»

На вкладці “Програми навчання” можна прочитати більш детальну інформацію про усі програми та їх довжину навчання в годинах (рис 1.11).

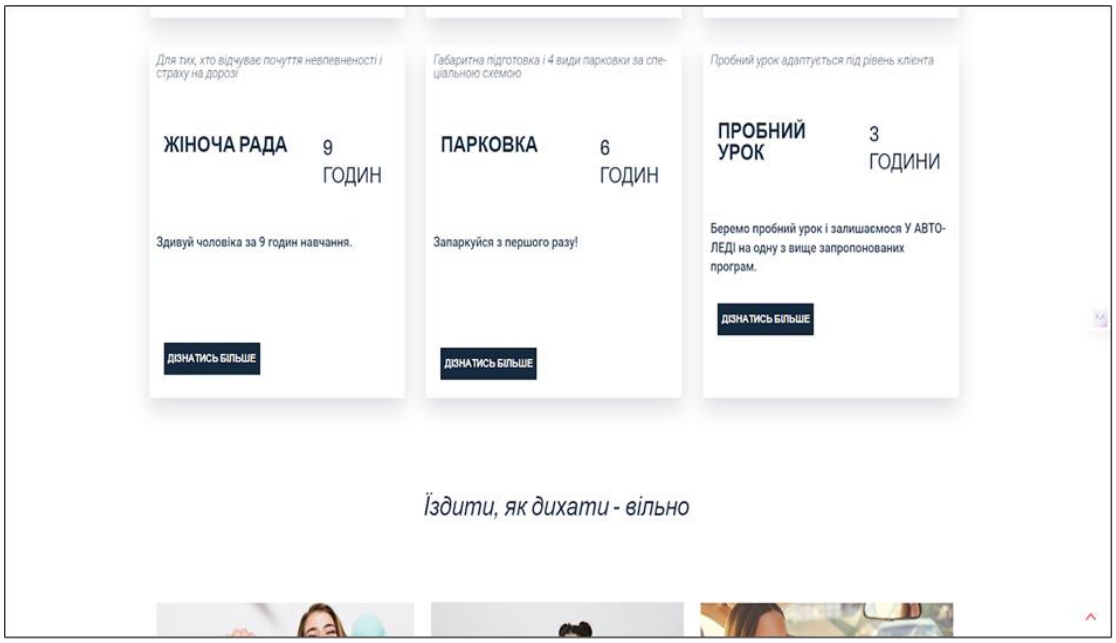


Рисунок 1.11 – Сторінка “Програми навчання”

На вкладці “Учням” можна прочитати правила з безпечного водіння а також оглянути найчастіші питання які виникають при вступі в автошколу (рис 1.12).

Рисунок 1.12 – Сторінка “ Учням”

На вкладці “Подарункові сертифікати” можна ознайомитися з подарунковими сертифікатами та інформацією про них (рис 1.13).

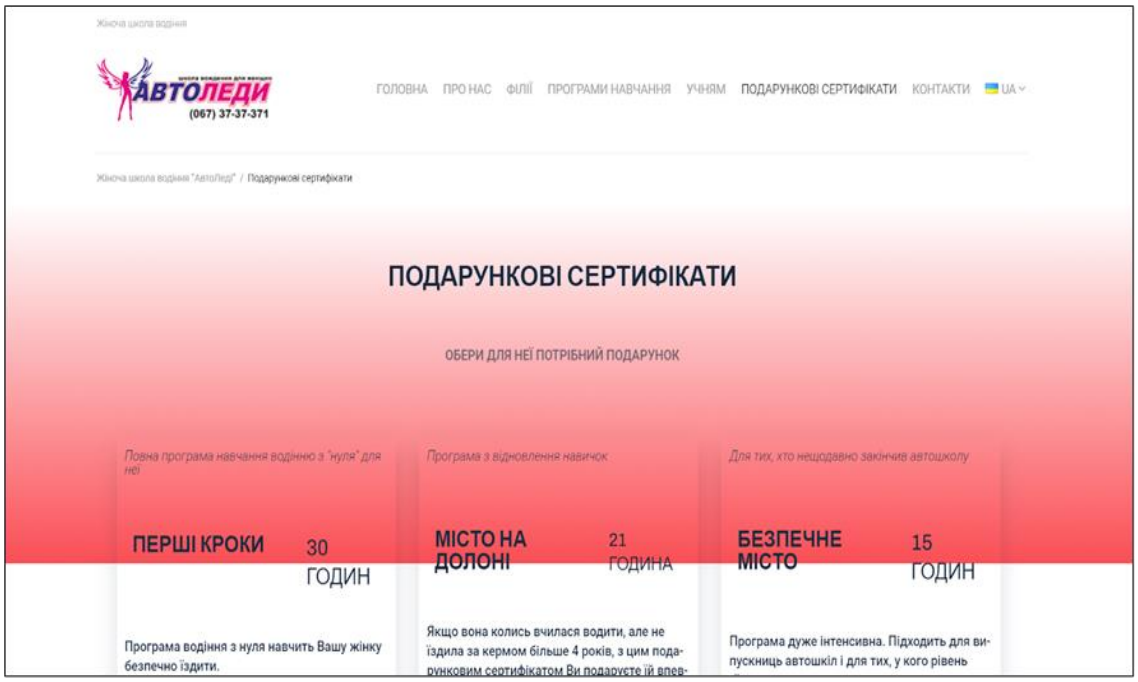


Рисунок 1.13 – Сторінка “ Подарункові сертифікати”

На вкладці “Контакти” представлені контактні номера телефону. Також є можливість залишити свої контакти або перейти до різних соцмереж (рис1.14).

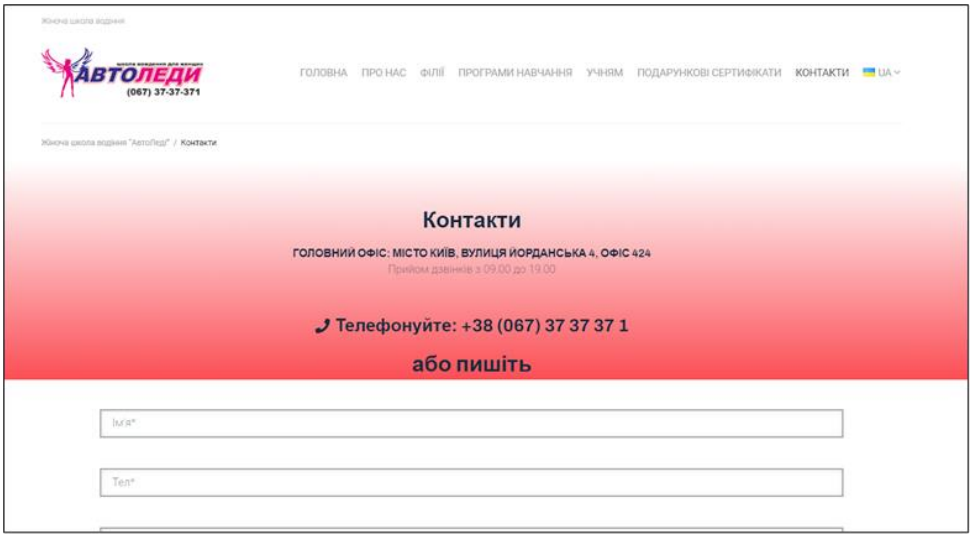


Рисунок 1.14 – Сторінка “ Контакти”

Плюси автошколи, де навчання проводиться жінками:

- зручність та комфорт: Багато жінок відчувають себе більш комфортно в навчанні в жіночому колективі. Це може збільшити їхню віру у себе і сприяти успішному навчанню;
- присутня локалізація сайту;
- зрозумілий інтерфейс;
- зменшення стереотипів: Автошкола для жінок сприяє подоланню стереотипів, що пов'язані з жіночою водійською майстерністю. Вона показує, що жінки можуть бути відмінними водіями;
- зручне розташування: Такі автошколи можуть бути розташовані у багатьох містах України, що забезпечує зручний доступ для більшої кількості жінок.

Мінуси автошколи:

- відсутність електронного запису: Відсутність онлайн-системи запису та оплати може бути не дуже зручною для клієнток, які бажають зареєструватися або оплатити курси в зручний для них час;
- не дуже зрозумілий інтерфейс для користувача;

– оплата за курс: Якщо автошкола не надає можливості оплатити курс онлайн, це може вимагати додаткового зусилля від клієнтів, наприклад, відвідування фізичної точки оплати.

Сайт автошколи «Драйв-мастер».

Зайшовши на сайт одразу попадаємо на Головну сторінку. На ній можна побачити картинки та також що школа знаходиться в Харкові (рис 1.15).

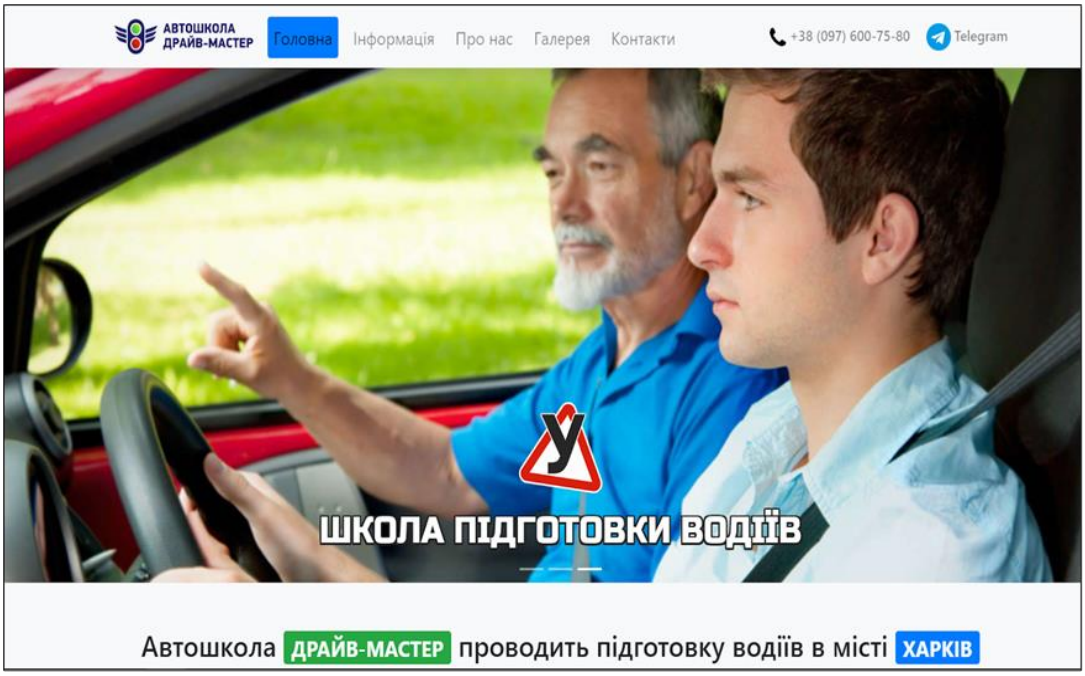


Рисунок 1.15 – Сторінка “головна”

Скролячи вниз, бачимо інформацію про автошколу та ціни (рис 1.16).

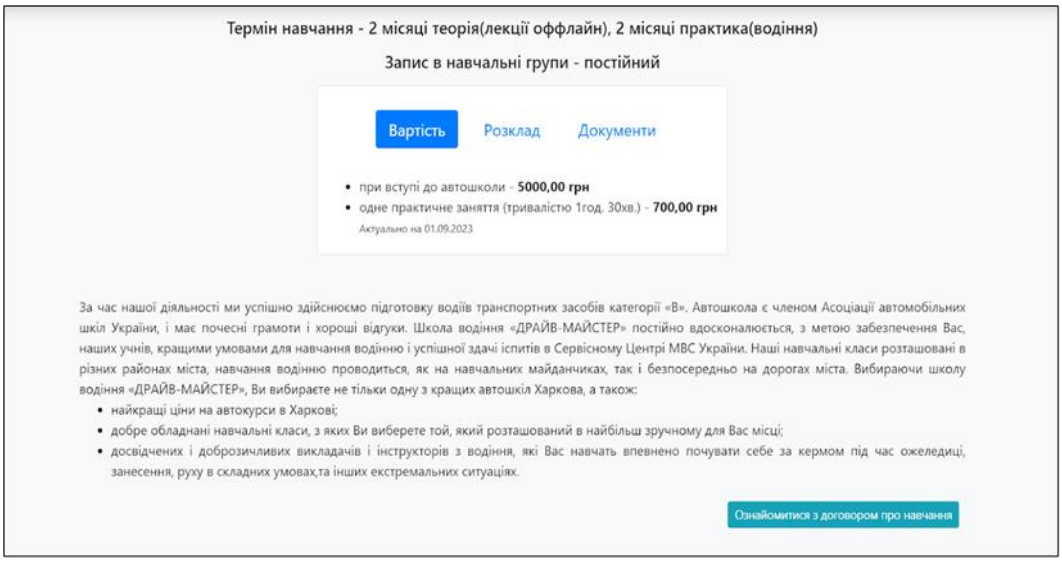


Рисунок 1.16 – Сторінка “головна”

Ще нижче ми можемо ознайомитися з сертифікатами автошколи та її опис (рис 1.17).

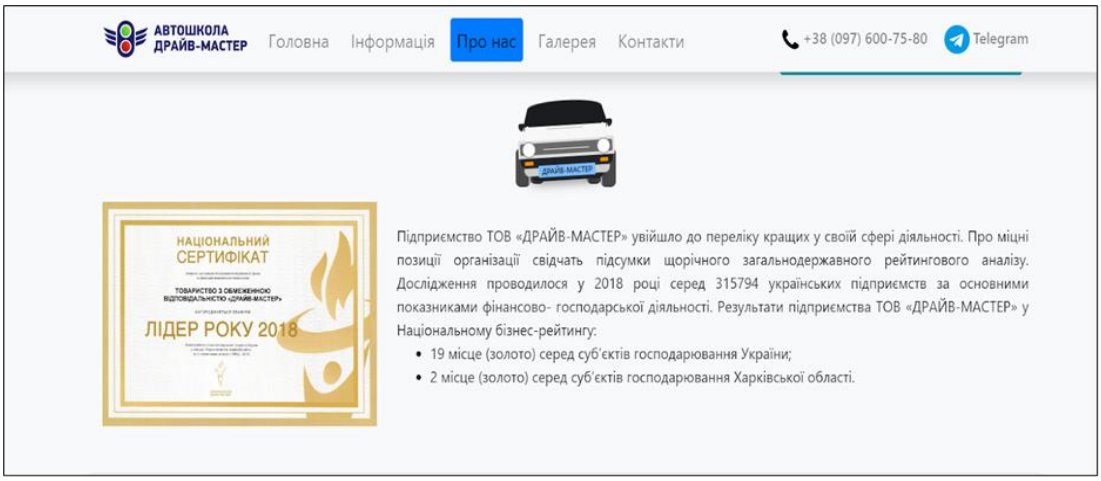


Рисунок 1.17 – Сторінка “Про нас”

Після цього ми можемо переглянути різні фотографії автошколи: класів, машин, дорожніх знаків (рис 1.18).

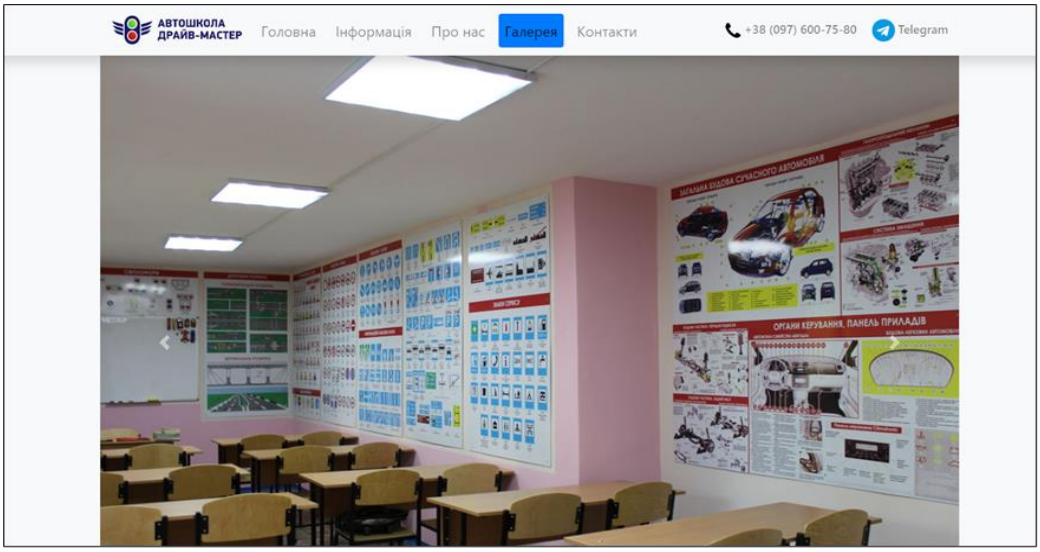


Рисунок 1.18 – Сторінка “Галерея”

Після галереї ми бачимо підрозділ контактів та адреса працюючих шкіл в місті Харкові. Також наведено карту міста та мітки шкіл для більшої зручності розуміння місцезнаходження. Також можна побачити відгуки про школу (рис 1.19).

У цій таблиці ми представили оцінки функціональності додатків, використовуючи десятибальну шкалу.

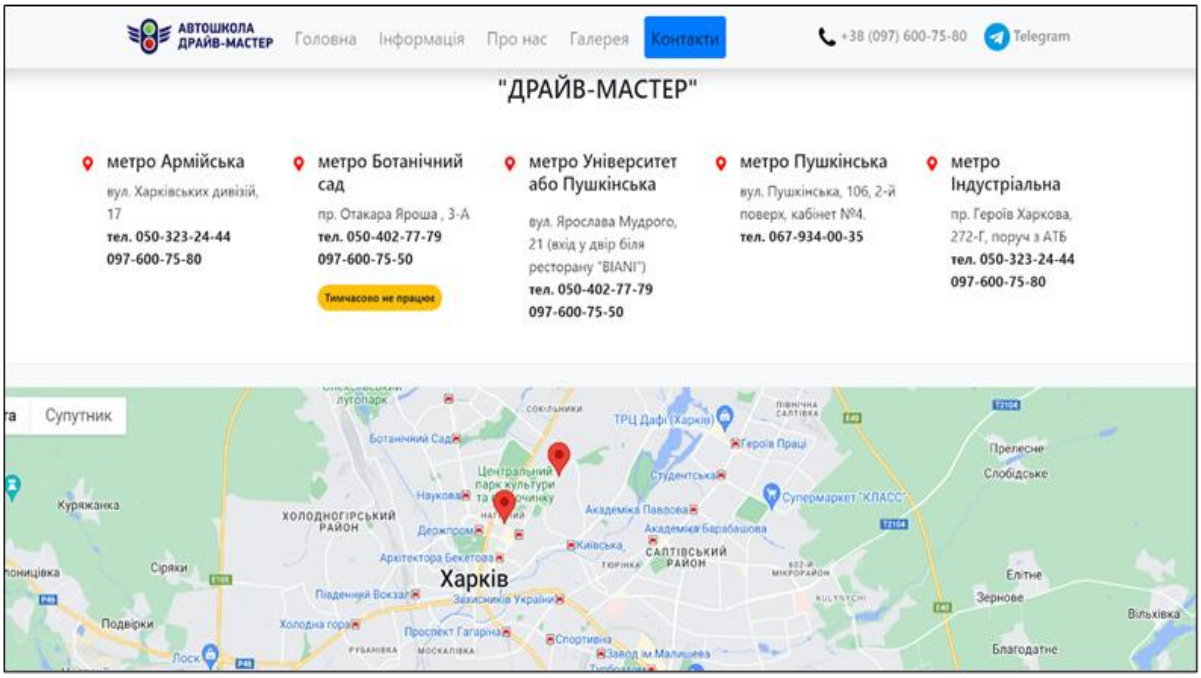


Рисунок 1.19 – Сторінка “Контакти”

Плюси автошколи «Драйв – мастер»:

- наявність в усіх районах: Можливість навчання в автошколі в усіх районах Харкова забезпечує зручність і доступність навчання для багатьох мешканців міста;
- простий та лаконічний дизайн сайту;
- гарні відгуки: Позитивні відгуки і репутація автошколи свідчать про якісне навчання та задоволеність від студентів.

Мінуси автошколи “Драйв - мастер”:

- відсутність електронного запису: Відсутність онлайн-системи запису може бути не дуже зручною для клієнтів, оскільки вони можуть бажати записатися в школу або перевірити доступність курсів без необхідності відвідування фізичної точки;
- відсутність онлайн-оплати: Також, відсутність можливості онлайн-оплати за курс може ускладнити процес реєстрації та сплати грошей для студентів.

Аналогічні системи, згадані раніше, не задовольняють всі потреби користувачів у повному обсязі, що демонструється в таблиці 2.1.

Таблиця 2.1. Порівняння нашого продукту з іншими конкурентами

	Назви продуктів			
Показник	Створюваний додаток	Автошкола “Україна”	Жіноча школа водіння “АвтоЛеді”	Автошкола “Драйв - мастер”
Дизайн	7	7	7	7
Функціональність	9	7	7	5
Обслуговування	9	5	5	6
Надійність	9	8	8	8
Продуктивність	9	7	8	7
Інноваційність	5	3	2	2
Клієнтське задоволення	9	8	7	8
Підтримка	8	6	6	7
Загалом	67	51	49	50

Згідно з даними, представленими у таблиці, наша система має перевершувати конкурентів за більшістю параметрів.

1.3 Постановка задач

Основна ціль цього проєкту полягає у розробці фронтенд-частини веб-версії сайту. Розробка фронтенду має забезпечити інтуїтивно зрозумілий і привабливий

користувацький інтерфейс, що спростить взаємодію користувачів з веб-сайтом. Основні завдання фронтенду включають створення зручної навігації, оптимізацію для різних пристроїв та браузерів, а також інтеграцію з бекенд-системами для забезпечення динамічного вмісту та інтерактивності. Фронтенд має включати такі елементи як форми реєстрації та входу, дашборди для керування навчальними планами, а також візуально привабливі засоби для відображення прогресу користувачів.

Бекенд-частина системи відповідатиме за обробку даних, збереження інформації та забезпечення функціональності веб-сайту і мобільного додатку. Він включатиме сервери, бази даних і API для забезпечення взаємодії між фронтендом та даними, що зберігаються на сервері. Бекенд також буде забезпечувати безпеку даних, обробку запитів і повернення відповідей до користувацьких запитів. Основні аспекти включатимуть автентифікацію користувачів, авторизацію доступу до ресурсів та виконання бізнес-логіки додатків.

Мобільний застосунок має стати гнучким і доступним рішенням для користувачів на платформі Android. Він повинен синхронізуватися з веб-версією та бекендом, забезпечуючи однакову функціональність. Мобільний застосунок включатиме такі функції, як перегляд розкладу занять, відстеження прогресу навчання та управління особистими даними. Також передбачено реалізацію сповіщень для інформування про зміни в розкладі або нові вимоги в навчальному процесі.

Таким чином, інтеграція між трьома основними компонентами проєкту (фронтенд, бекенд, мобільний додаток) повинна забезпечити злагоджену та ефективну роботу всієї системи, відповідаючи сучасним вимогам до навчальних додатків і сервісів.

2 ПЕРЕЛІК ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Постановка мети

Метою цього проекту є створення фронтенд-частини веб-версії для сайту, яка забезпечить користувачам зручний та інтуїтивно зрозумілий інтерфейс для взаємодії з системою. Завдання фронтенду включатиме розробку адаптивних веб-сторінок, які будуть коректно відображатися, інтеграцію з бекендом для динамічного відображення даних, а також створення візуально привабливого дизайну, що відповідає сучасним тенденціям веб-розробки.

Для імплементації фронтенд частини веб-застосунку було обрано React, що дозволяє створювати інтерактивні користувацькі інтерфейси з використанням компонентного підходу. React забезпечує високу продуктивність та зручність розробки завдяки своїй архітектурі, яка дозволяє ефективно управляти станом застосунку та оновлювати користувацький інтерфейс у відповідь на зміни даних.

Для забезпечення багатомовності та локалізації застосунку використовується i18next. Ця бібліотека дозволяє легко додавати підтримку різних мов, керувати перекладами та динамічно змінювати мову інтерфейсу. Використання i18next дозволяє нам забезпечити користувачам з різних країн зручний досвід взаємодії з застосунком, незалежно від їх мови.

Для інтеграції платіжної системи було обрано LiqPay. LiqPay надає простий та зручний спосіб реалізації онлайн-платежів, що дозволяє користувачам швидко та безпечно здійснювати транзакції. Використання LiqPay забезпечує високий рівень безпеки платежів, підтримує різні платіжні методи та забезпечує зручну інтеграцію з нашим застосунком.

Для здійснення HTTP-запитів до серверної частини застосунку використовується бібліотека Axios. Axios дозволяє легко виконувати запити різних типів, таких як GET, POST, PUT, DELETE, та інші. Вона забезпечує ефективну обробку запитів і відповідей, а також підтримує роботу з промісами, що спрощує управління асинхронними операціями у React-застосунку.

Для керування станом застосунку та збереження контексту користувача між сторінками було обрано Redux. Redux дозволяє централізовано зберігати стан застосунку та забезпечує передбачуване оновлення стану через дії та редюсери. Використання Redux дозволяє легко керувати складним станом застосунку, забезпечує масштабованість та полегшує тестування компонентів.

Архітектура фронтенд частини застосунку повинна бути масштабованою та легко тестованою. Для досягнення цих цілей у поєднанні з React було обрано підхід, що поєднує використання компонентів, бібліотеки для роботи з даними та управління станом. Це забезпечує високу продуктивність, зручність розробки та стабільну роботу застосунку.

2.2 Загальний опис системи

Основна ціль цієї роботи полягає у наданні можливості швидкого та зручного пошуку і вибору інструктора. Учні можуть завжди отримати доступ до розкладу свого інструктора, записатися на майбутні заняття, відмінити заняття якщо час їм не підходить, а також одразу мають можливість оплатити це заняття, тоді як інструктори отримують інструмент для управління своїм розкладом. Адміністратори сайту мають можливість додавати нових інструкторів та їх машини, а також підтверджувати або спростовувати запис учнів до інструкторів , щоб виключити людей які поки не можуть навчатися водінню. Це дуже спрощує організацію навчального процесу і полегшує взаємодію як між інструкторами та учнями, так і між адмінами та учнями і інструкторами. Важливим акцентом робиться на розробці інтуїтивно зрозумілого і привабливого інтерфейсу, який забезпечить зручність та ефективність користування сервісом. Важливим аспектом є адаптивність дизайну, щоб сайт коректно відображався на різних пристроях та роздільних здатностях. Розробка включає впровадження функцій пошуку, фільтрації, і сортування даних, які значно покращать навігацію та доступність інформації. Також важливо забезпечити безпечне та швидке завантаження веб-сторінок, інтеграцію з соціальними мережами і підтримку мультиязичності.

Що стосується бекенду, його розробка передбачає створення стабільної та безпечної серверної частини, яка забезпечуватиме ефективну обробку даних і взаємодію з базами даних. Бекенд повинен підтримувати високий рівень продуктивності та масштабування, щоб витримувати високе навантаження користувачів.

У контексті мобільного застосунку, його розробка фокусується на створенні функціонального та оптимізованого додатку для Android, який забезпечить користувачам доступ до всіх необхідних функцій, таких як перегляд розкладу, відстеження прогресу, та зручне управління особистими налаштуваннями.

2.3 Основний функціонал системи

Розроблена програмна система для автошкіл має на меті автоматизацію процесів навчання і управління, спрямована на задоволення потреб двох головних груп користувачів: студентів та інструкторів.

Для забезпечення широкого спектру потреб, система включає наступний функціонал:

- реєстрація та авторизація для студентів і інструкторів;
- підтвердження електронної пошти через gmail
- управління особистими даними в профілі;
- перегляд інформації про інструкторів;
- запис на навчальні заняття;
- оплата забронюваного заняття;
- перегляд розкладу інструкторів погодинно з можливістю обирати час а також бачити які заняття вже заброньовані учнями або відмінені інструктором;
- управління записами на заняття, включаючи можливість перенесення або скасування занять;
- перегляд власного профілю;
- налаштування мови інтерфейсу.

Окрім цього, роль адміністратора автошколи в системі дозволяє:

- підтвердження або скасування запиту учня на додавання до інструктора;
- редагування інструкторів та їх транспортних засобів;
- перегляд та редагування міст доступних у застосунку;
- перегляд усіх користувачів системи.

Такий підхід сприяє створенню комплексної системи, яка ефективно вирішує задачі автоматизації управління навчальними процесами та підтримки взаємодії між усіма учасниками навчального процесу.

2.4 Загальні обмеження

Розроблена програмна система для автошкіл має такі загальні обмеження:

- користувач має підтвердити свою електронну пошту щоб в подальшому мати доступ до всього іншого функціоналу, наприклад запис до інструктора;
- у інструкторів доступна лише обмежена кількість місць для запису на практичні заняття. Це обмеження може виникнути через обмежену кількість доступних інструкторів або обмежену кількість автомобілів, доступних для навчання;
- програмний додаток працює тільки при наявності підключення до бездротової локальної мережі або мобільного зв’язку. Без цього зв'язку користувачі не зможуть користуватися додатком та його функціоналом;
- програмний додаток може бути доступний тільки на певних мовах, що може створити проблеми для користувачів, які не розмовляють мовою, на якій доступний додаток. Наприклад, якщо програма підтримується лише англійською мовою, це може стати перешкодою для користувачів, які володіють іншими мовами.

2.5 Припущення та залежності

- припускається, що користувачі, які реєструються у системі, вже пройшли теоретичне навчання та успішно склали іспит у сертифікованому

сервісному центрі МВС. Це гарантує, що студенти мають необхідні знання та навички для отримання практичного навчання в автошколі.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ

3.1 Проектування архітектури програмної системи

Для розробки фронтенду веб-сайту було обрано сучасні технології та інструменти, що дозволяють створити ефективний та гнучкий інтерфейс для користувачів. Було використано React, інноваційну бібліотеку для створення користувацьких інтерфейсів, яка надає можливість швидко реагувати на зміни та створювати складні UI-компоненти з легкістю . Він гарний тим, що його можна використовувати в web-додатках будь-якого масштабу, але саме в невеликих додатках із перспективою росту він показує себе максимально добре.[2].

Також, для підтримки міжнародизації та локалізації нашого веб-сайту, використано бібліотеку i18next [3]. Це дозволило легко інтегрувати різноманітні мовні ресурси та забезпечити зручність використання сайту для користувачів з різних країн та мовних середовищ.

Для структурування та стилізації веб-сайту використано SCSS (Sass)[4], що дозволяє ефективно організовувати CSS-код та надає більше можливостей для розробки та підтримки стилів.

Управління станом додатку було забезпечено за допомогою Redux, що є потужним інструментом для управління складним станом додатку та забезпечення консистентності даних на різних рівнях застосунку.

Цей підхід дозволив створити веб-сайт з високим рівнем функціональності та користувацького досвіду, забезпечуючи при цьому ефективну та швидку розробку та підтримку (див. рисунок 3.1).

Для більшої зручності було створено окремі папки для адміна та клієнта. Адмін папка відповідає за фронтенд частину для адміністратора автошколи де можна побачити та редагувати інформацію про користувачів, доступна тільки робітникам автошкіл, які мають статус адміністратора. А папка клієнт містить собі фронтенд частину вже безпосередньо для користувачів ресурсу.

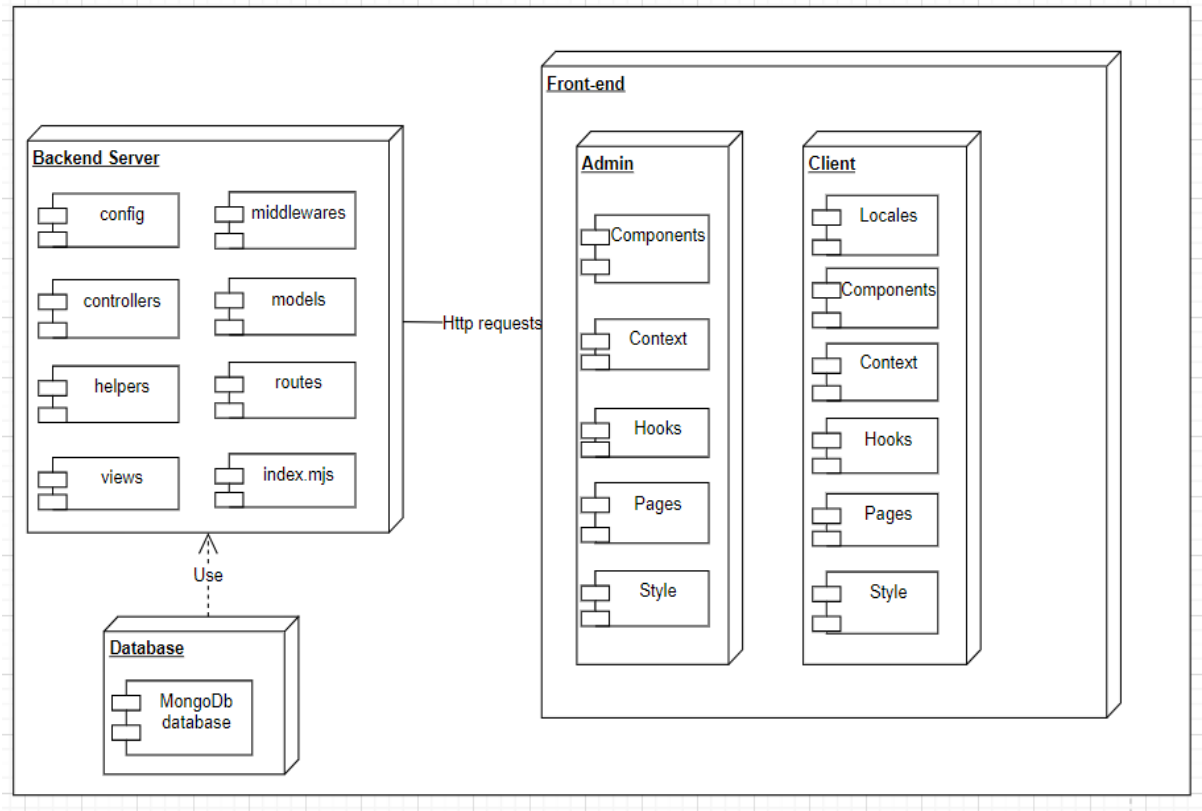


Рисунок 3.1 – Схема архітектури програмної системи

3.2 Проектування бази даних

В якості бази даних було обрано MongoDB. Її гнучка структура даних дозволяє легко відстежувати інформацію про студентів, інструкторів та автомобілі, а також швидко адаптувати систему до змінних вимог навчального процесу. MongoDB забезпечує швидкий доступ до великого обсягу даних, що дозволяє ефективно управляти розкладом занять, веденням обліку успішності студентів та іншими аспектами організації навчання. Безпека та надійність MongoDB гарантує безперебійну роботу системи, забезпечуючи високу якість обслуговування для користувачів автошколи [5].

Щодо взаємодії з фронтендом, для комунікації з MongoDB було використано Axios – бібліотеку JavaScript для виконання запитів з клієнта. Фронтенд надсилає запити до сервера, який потім взаємодіє з базою даних MongoDB для забезпечення необхідної інформації. Цей підхід дозволяє ефективно керувати даними і забезпечувати потрібну функціональність для користувачів нашої системи.

Для реалізації цих запитів було створено кастомний хук для управління станом та обробки запитів. Він також забезпечує перевірку та оновлення токенів автентифікації, оскільки багато запитів до бази даних дозволені лише для зареєстрованих користувачів.

3.3 Прецеденти використання

Доступ до додатку буде надано всім користувачам сайту. Проте, важливо, не зареєстрованим користувачам буде обмежено доступ до більшості функцій. Нові користувачі можуть зареєструватися або через мобільний додаток, або відвідавши веб-сайт. Додаток передбачає три ролі: студент, який збирається отримувати навчання в автошколі, інструктор, який проводить навчання, а також адміністратор, який може вносити зміни. Це дозволить керувати процесом навчання з відповідними правами та функціоналом для кожного типу користувача.

Для ілюстрації сценаріїв використання системи було побудовано Use-Case діаграму [6], що спрощує візуалізацію взаємодії різних акторів з системою та окреслює, які конкретні функціональні можливості вони використовують.

Діаграма прецедентів для студентів надає уявлення про функції, які доступні користувачеві та який шлях йому слід пройти, щоб скористатися цими функціями (див. рисунок 3.2).

Серед доступних функцій для користувача студента:

- реєстрація;
- авторизація;
- перегляд та редагування власного профілю;
- перегляд інструкторів;
- перегляд інформації про автомобіль для навчання;
- відправлення запиту на призначення до інструктора;
- перегляд розкладу;
- запис на заняття з обраним інструктором;
- відміна уроку менше ніж за день;

– можливість оплати онлайн.

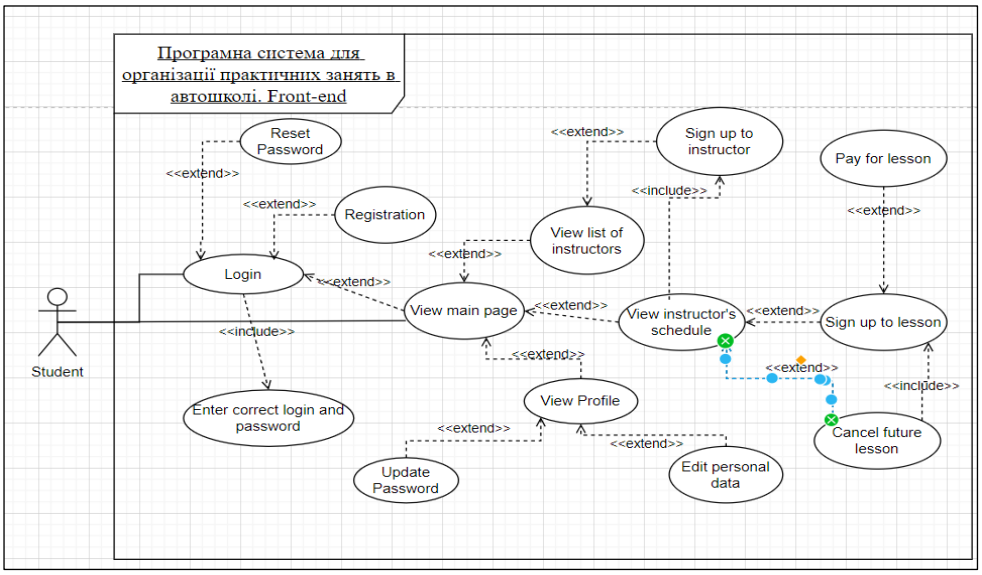


Рисунок 3.2 – Use Case діаграма для користувача-студента

Серед доступних функцій для користувача інструктора (рисунок 4.3):

– реєстрація;

– авторизація;

– перегляд власного профілю;

– перегляд розкладу;

– можливість відміни уроків якщо туди ще не записані користувачі.

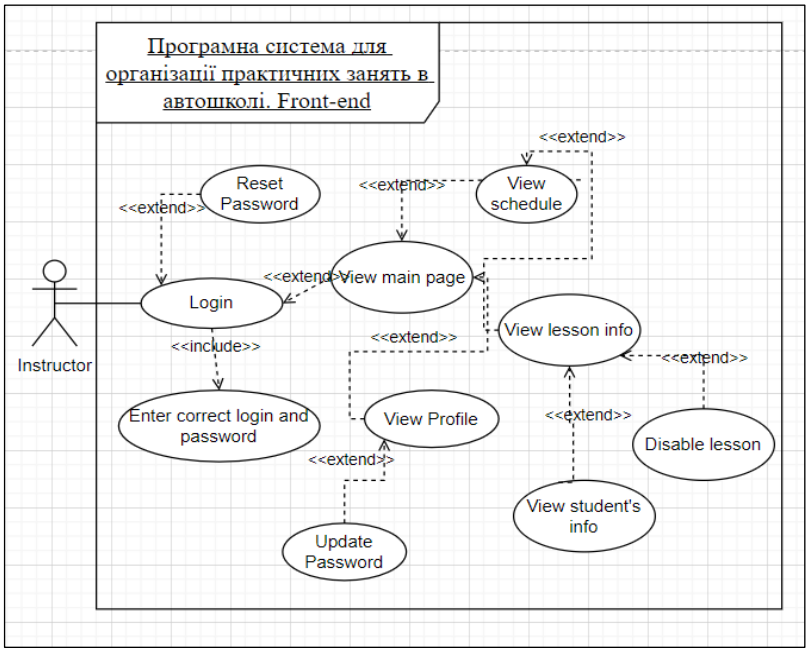


Рисунок 3.3 – Use Case діаграма для користувача-інструктора

Серед доступних функцій для користувача адміна(див. рисунок 3.4):

- авторизація;
- перегляд та редагування інструкторів та користувачів;
- підтвердження запиту на призначення до інструктора;
- перегляд міст;
- перегляд та редагування інформації про інструкторів.

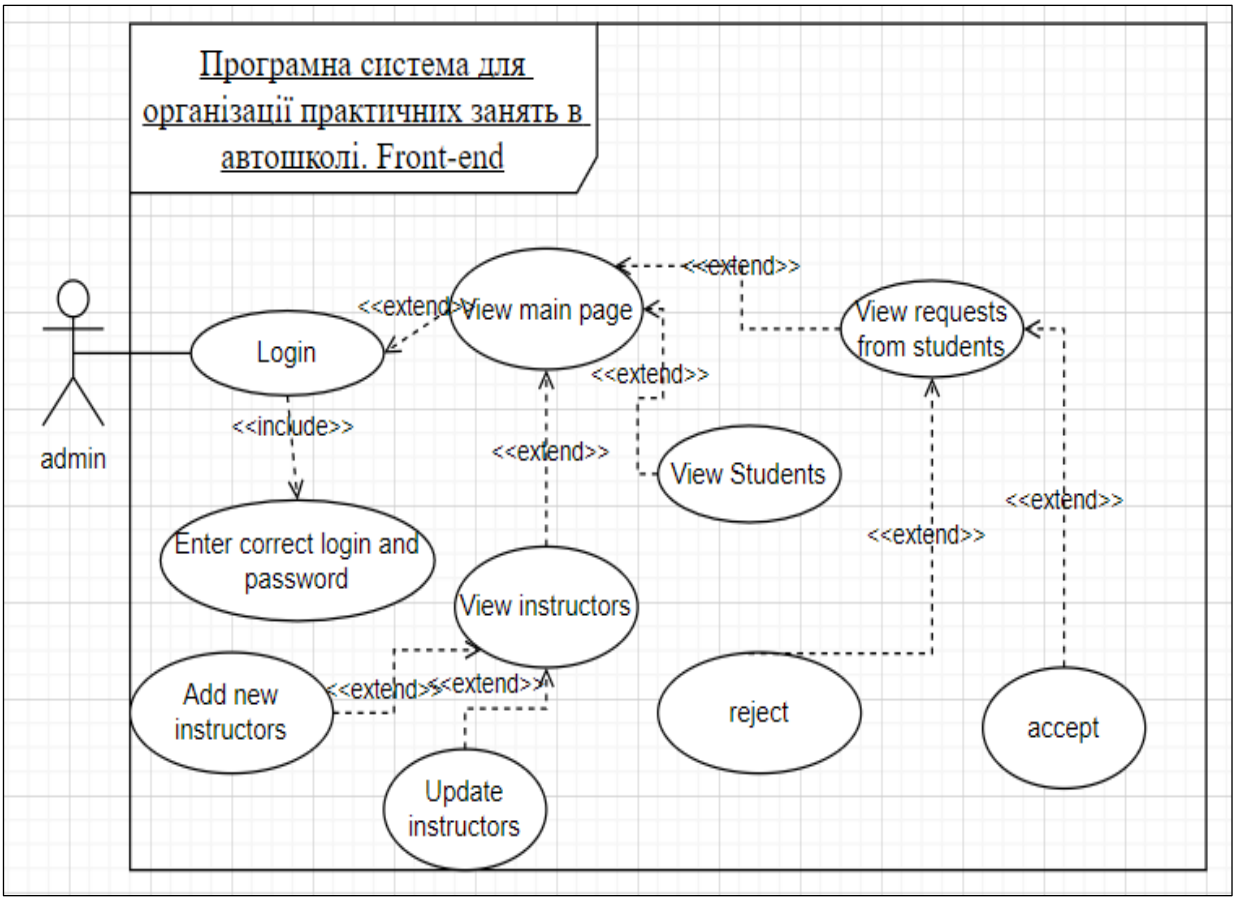


Рисунок 3.4 – Use Case діаграма для користувача-адміна

Для більш зрозумілої логіки запису на урок, було побудовано діаграму активності на рисунку 3.5. Діаграма активності на рисунку 3.5 відображає кроки процесу запису на урок і його оплати з точки зору користувача, що спрощує їм навігацію та робить процес зрозумілим. Завдяки такій візуалізації користувачі можуть чітко уявити послідовність дій і виконувати їх без зайвих запитань. Це підвищує загальний комфорт використання системи та сприяє позитивному враженню від взаємодії з нею.

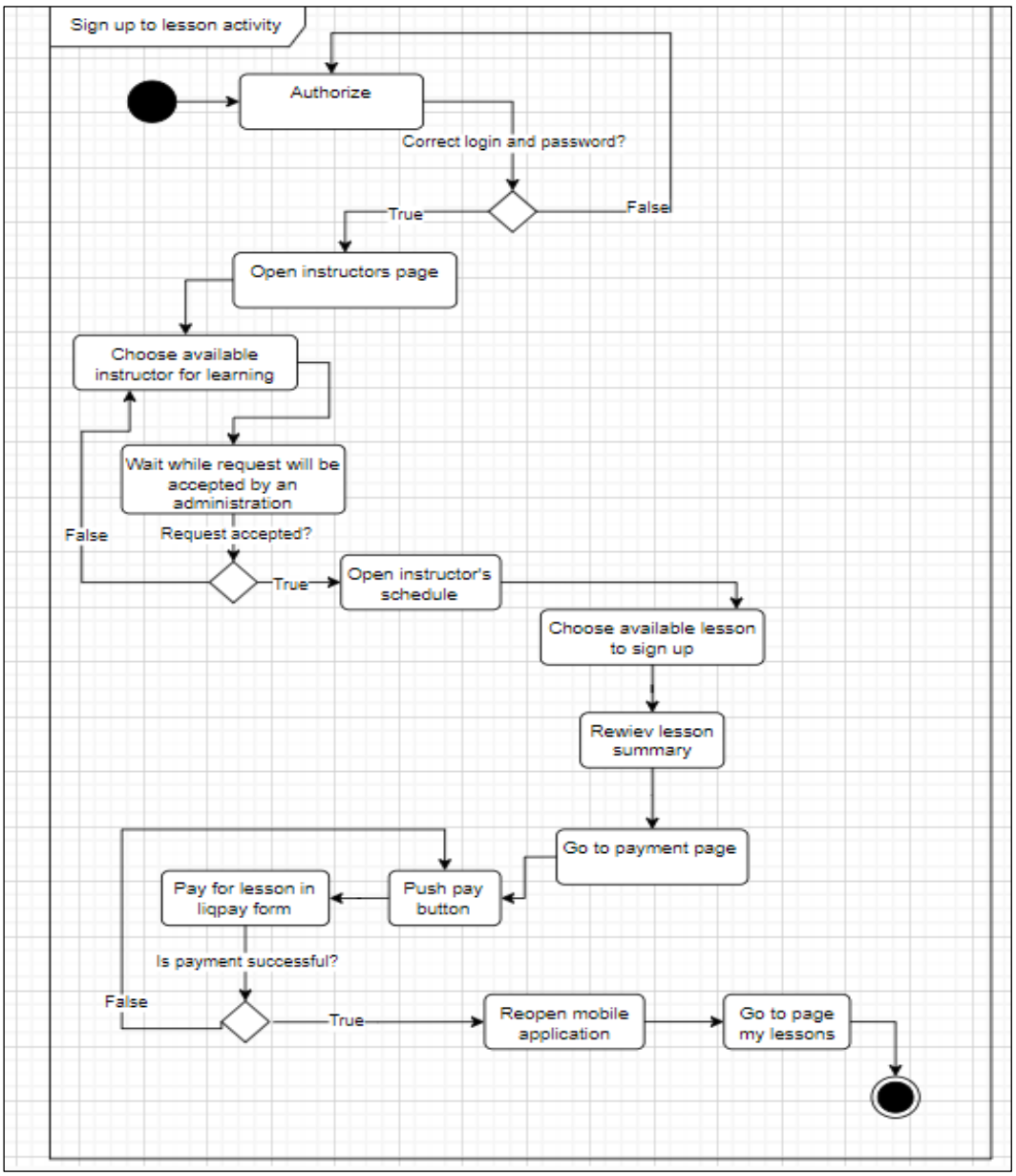


Рисунок 3.5 – Діаграма активності запису на урок

3.4 Розробка візуальної частини

Для фронтенду було використано низку інструментів і технологій, щоб забезпечити високу продуктивність та ефективність розробки. Основним фреймворком для створення користувацького інтерфейсу був React, який дозволяє легко організовувати компоненти та забезпечувати їхню перевикористовуваність.

Для локалізації текстових ресурсів було використано бібліотеку i18next, що дозволяє забезпечити підтримку різних мов у нашому додатку. Redux використовувався для глобального керування станом додатку, що дозволяє ефективно керувати даними та їх взаємодією між компонентами.

Для стилізації компонентів було використано препроцесор SCSS, який дозволяє швидко та зручно створювати зрозумілий та підтримуваний CSS код. Крім того, для управління глобальним станом додатку та передачі даних між компонентами ми використовували контексти та хуки.

Для взаємодії з сервером та отримання даних ми використовували бібліотеку Axios, яка надає можливість легко виконувати HTTP запити. І для навігації між сторінками додатку ми використовували бібліотеку React Router DOM, яка дозволяє нам динамічно маніпулювати URL та відображати різні компоненти на основі шляху.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Вибір засобів програмної реалізації

Розроблювана програмна система є великою та комплексною. Для забезпечення високої потужності та надійності роботи, а також для легкості підтримки написаного програмного коду, було обрано сучасні програмні засоби та інструменти. Для розробки клієнтської веб-частини використовувались мова програмування JavaScript та фреймворк React.

React призначений для створення користувацьких інтерфейсів і має численні переваги. Однією з основних є використання віртуального DOM, що дозволяє знизити навантаження на реальний DOM і покращити продуктивність програми. React оптимізує оновлення тільки тих частин інтерфейсу, які змінилися, замість повного оновлення всього дерева DOM. Компонентний підхід React дозволяє розбивати користувацький інтерфейс на невеликі незалежні компоненти, що робить код більш організованим, легким для розуміння і повторного використання.

Для локалізації інтерфейсу було обрано бібліотеку i18next. Вона дозволяє легко перекладати додаток на різні мови, забезпечуючи гнучкість та масштабованість у створенні мультимовного інтерфейсу. i18next має зручні інструменти для налаштування перекладів і підтримує динамічне завантаження мовних ресурсів.

Для реалізації платіжної системи використовується LiqPay.LiqPay – платіжний сервіс, який надає функції інтернет-еквайрингу – прийом платежів на веб-сайтах, у мобільних додатках, підключених до Інтернету [7]. Цей інструмент забезпечує безпечні та зручні оплати, підтримуючи різні способи платежів.LiqPay надає простий у використанні API, що дозволяє інтегрувати платіжну систему у веб-додаток з мінімальними зусиллями.

Для виконання HTTP-запитів використовується бібліотека Axios. Вона забезпечує простий та інтуїтивно зрозумілий інтерфейс для роботи з API, підтримує обробку асинхронних запитів і автоматично керує помилками. Axios

дозволяє легко налаштовувати запити та обробляти відповіді, що робить її ідеальним вибором для роботи з мережевими запитами.

Для управління станом додатку обрано Redux – бібліотеку JS для передбачуваного та підтримуваного глобального управління станом [8]. Вона дозволяє ефективно керувати станом додатку та забезпечує однаковість і передбачуваність у потоці даних. Redux зберігає контекст користувача між сторінками, що робить перехід між ними плавним і безперервним. Для ще більш швидкої розробки та надійнішого керування станом була використана бібліотека Redux, яка пропонує зручні інструменти для визначення дій, маніпулювання і зберігання стану, а також автоматично вирішує безліч повсякденних завдань.

4.2 Опис фронтенд частини системи

Структура файлів проекту зображена на рисунку 4.1. У проекті є дві основні папки: public і src, причому основний код зберігається у папці src. У цій папці організовано три рівні структури:

- assets: знаходяться необхідні картинки та іконки для нашої системи;
- components: У цій папці зберігаються всі компоненти інтерфейсу користувача. Кожна підпапка містить код, пов'язаний з конкретним розділом UI. Це рівень, де відбувається рендеринг та взаємодія з користувачем;
- fonts: розміщені шрифти проекту;
- hooks: розміщені кастомні хуки проекту;
- i18n: функції для локалізації;
- context: Тут розміщений рівень BLL (Business Logic Layer), який відповідає за управління станом системи та логіку обробки. У цій папці викликаються методи з рівня DAL, і потім дані передаються на рівень UI;
- pages:сторінки додатку;
- styles: Ця папка містить допоміжні файли, такі як код для локалізації або загальні стилі системи SCSS. Це місце для зберігання утиліт, які використовуються в різних частинах проекту.

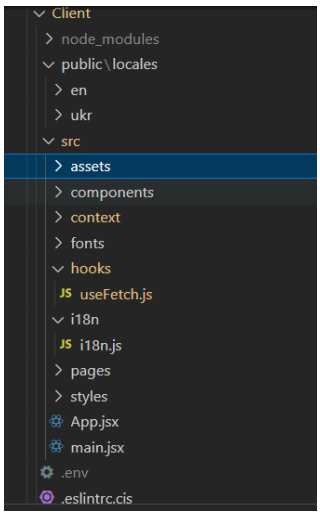


Рисунок 4.1 – Структура файлів проекту

Всі основні компоненти зібрані у файлі App.js, а головний файл JavaScript – це index.js, який служить точкою входу для додатку.

Додаток розроблений для обслуговування чотирьох типів користувачів: звичайних незареєстрованих студентів,студентів автошкол інструкторів, які проводять уроки, а такоє адміністраторів які управляють процесами на сайті. Після входу в додаток, кожен користувач буде перенаправлений на відповідний інтерфейс, спеціально адаптований для його потреб. Першими розглянемо весь функціонал звичайного користувача який не зареєстрований на сайті.

Перша сторінка яку бачить користувач після відкриття застосунку це головна сторінка (див. рисунок 4.2). Ця сторінка відкривається як користувачам так і інструкторам як початкова. На сторінці є навбар, в якому можна обирати мову застосунку, це було зроблено за допомогою інструмента i18next.

Програмний код для реалізації:

```
import i18n from "i18next";
import { initReactI18next } from "react-i18next";
import LanguageDetector from "i18next-browser-languagedetector";
import HttpApi from "i18next-http-backend";

i18n.use(initReactI18next)
    .use(LanguageDetector)
    .use(HttpApi)
    .init({
        debug: true,
        supportedLngs: ["ukr", "en"],
```

```
        fallbackLng: "ukr",
        ns: ["common", "pages", "components", "validation"],
        defaultNS: "pages",
        detection: {
            order: ["cookie", "navigator", "htmlTag"],
            caches: ["cookie"]
        },
        backend: {
            loadPath: "/locales/{{lng}}/{{ns}}.json"
        }
    });
```

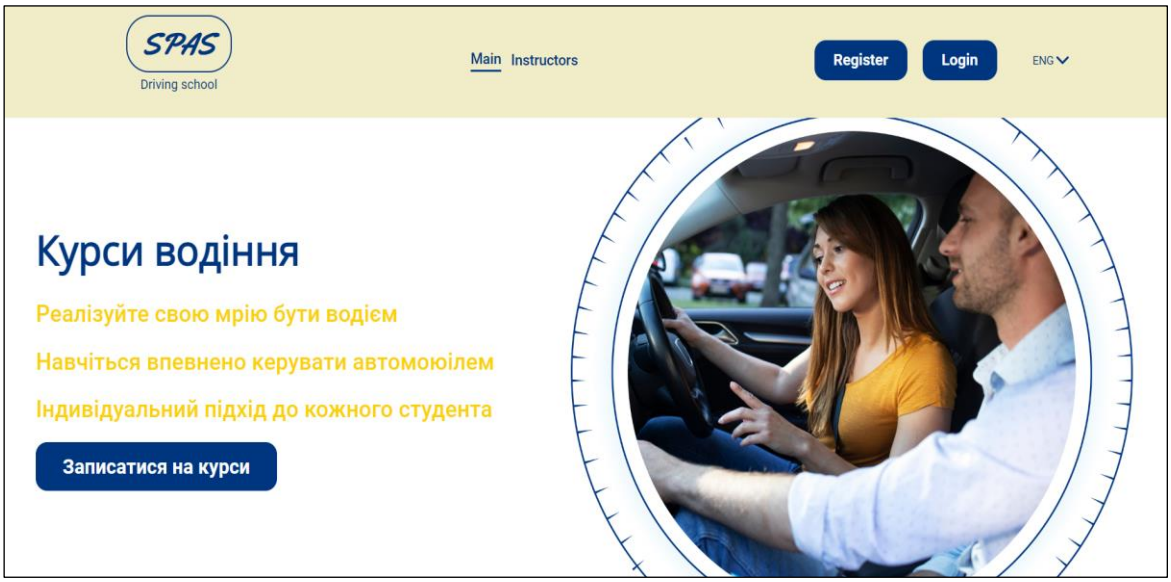


Рисунок 4.2 – Головний екран при вході користувача

Не зареєстрований користувач може тільки переглядати список всіх інструкторів (див. рисунок 4.3)

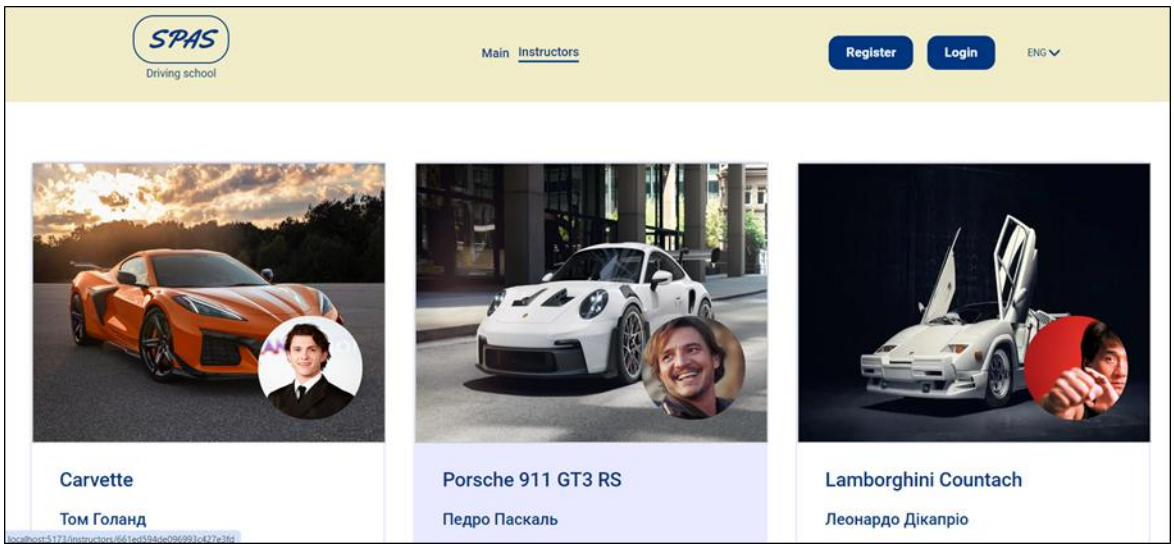


Рисунок 4.3 – Список інструкторів

Далі незареєстрований користувач може зареєструватися або увійти в систему відвідавши відповідні сторінки реєстрація або логін відповідно (див. рисунок 4.4)

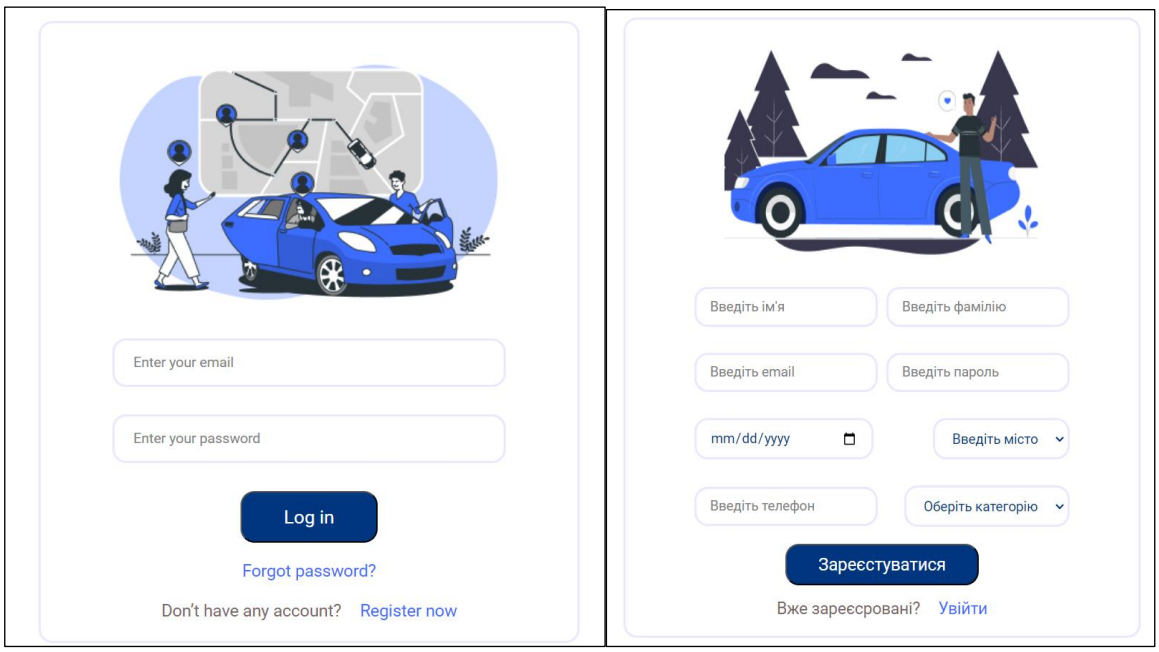


Рисунок 4.4 – Реєстрація та логін користувача

Для успішного збереження користувача було використано redux і написано таку функцію:

```
import { createContext, useEffect, useReducer } from "react"

const userFromLocalStorage = localStorage.getItem("user")
const user = userFromLocalStorage ? JSON.parse(userFromLocalStorage)
: null

const INITIAL_STATE = {
  user: JSON.parse(localStorage.getItem("user")) || null,
  loading: false,
  error: null
}
console.log(JSON.parse(localStorage.getItem("user")))


export const AuthContext = createContext(INITIAL_STATE)

const AuthReducer = (state, action) => {
  switch (action.type) {
    case "LOGIN_START":
      return {
        user: null,
        loading: true,
        error: null
```

```
      }
    case "LOGIN_SUCCESS":
      return {
        user: action.payload,
        loading: false,
        error: null
      }
    case "LOGIN_FAILURE":
      return {
        user: null,
        loading: false,
        error: action.payload
      }
    case "LOGOUT":
      return {
        user: null,
        loading: false,
        error: null
      }
    default:
      return state
  }
}

export const AuthContextProvider = ({ children }) => {
  const [state, dispatch] = useReducer(AuthReducer, INITIAL_STATE)

  useEffect(() => {
    localStorage.setItem("user", JSON.stringify(state.user))
  }, [state.user])

  return (
    <AuthContext.Provider
      value={{
        user: state.user,
        loading: state.loading,
        error: state.error,
        dispatch
      }}
    >
      {children}
    </AuthContext.Provider>
  )
}
```

Ця функція створена для управління станом автентифікації користувача у React-додатках, використовуючи контекст та редуктор. Вона забезпечує можливість зберігання даних про користувача, його стан завантаження та помилки. Крім того, вона визначає редуктор, який відповідає за зміну стану відповідно до дій користувача, таких як вхід, вихід та обробка помилок. Це

дозволяє легко та ефективно керувати станом автентифікації в додатку, забезпечуючи однорідність та простоту у роботі з даними користувача.

Після успішної реєстрації користувача (див. рисунок 4.5).

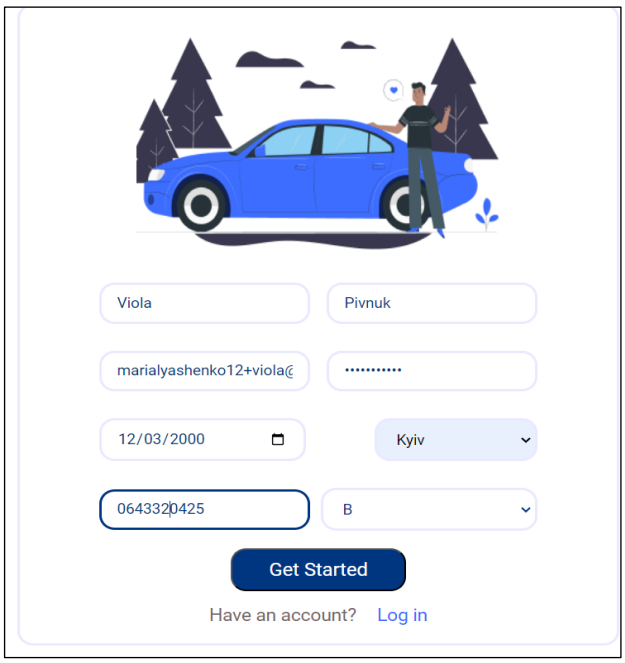


Рисунок 4.5 – Успішна реєстрація

Користувачу стає доступним також перегляд та редагування свого власного профілю але доступ до запису до інструктора відкриється лише після успішного підтвердження електроної пошти, про це свідчить віконечко на сторінці профілю користувача (див. рисунок 4.6)

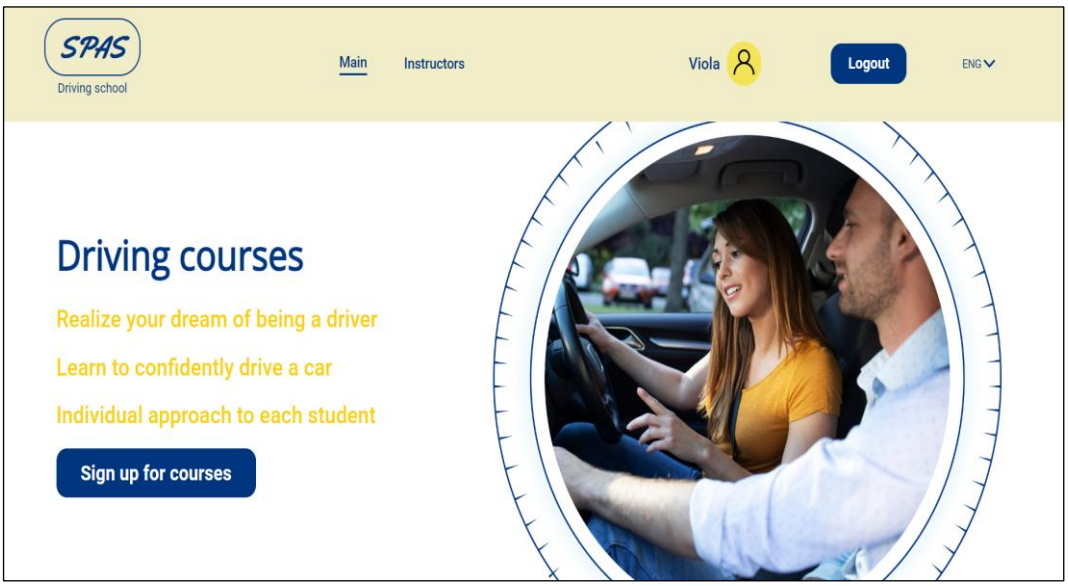


Рисунок 4.5 – Успішна реєстрація користувача на сторінці профілю користувача

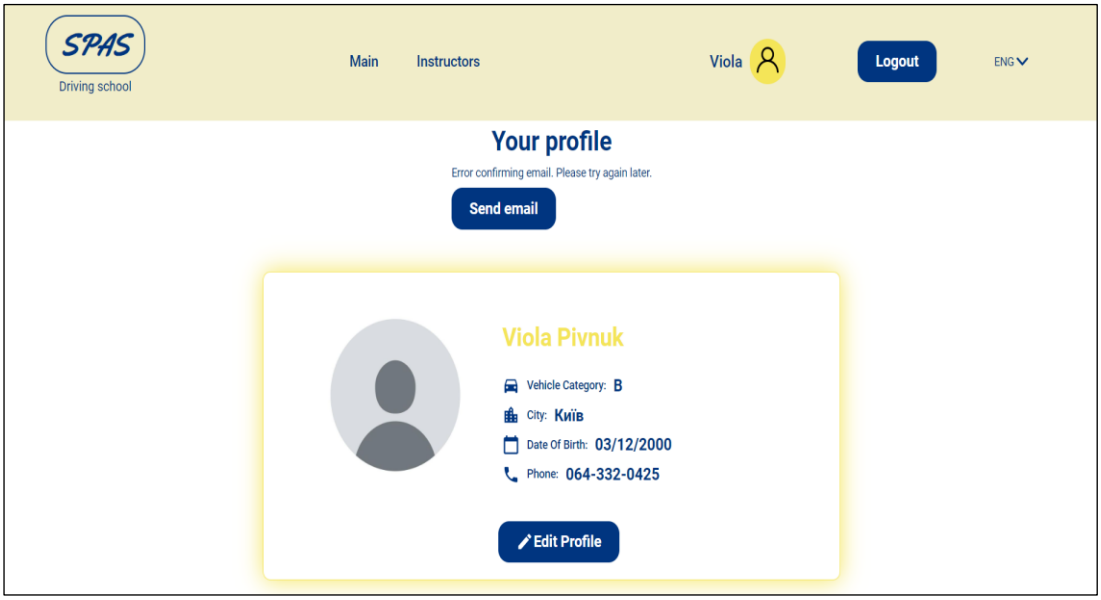


Рисунок 4.6 – Повідомлення про неверифіковану пошту та віконечко для редагування або перегляду сторінки користувача

Після реєстрації або при натисканні на кнопку повторної відправки імейла, на поштову скриньку приходить імейл щоб підтвердити пошту, (див. рисунок 4.7) після натискання на кнопку “verify email” , відкривається віконечко з кнопкою підтвердження. Після успішного підтвердження пошти, відкривається знову віконечко з логіном де користувач вже може авторизуватися в системі (див. рисунок 4.8).

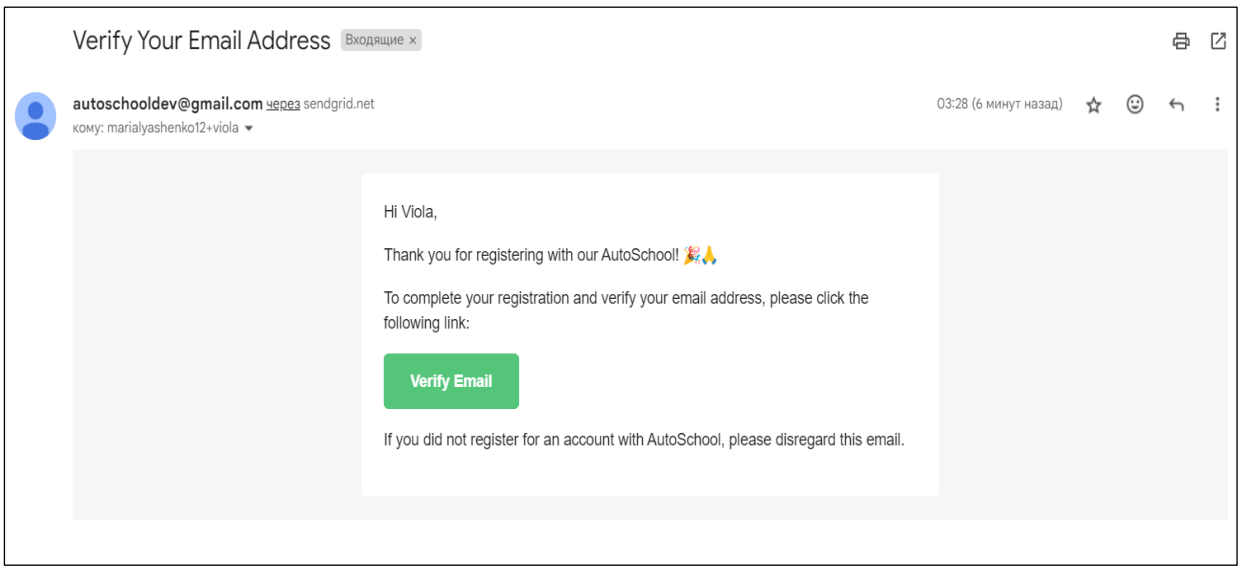


Рисунок 4.7 – Повідомлення яке приходить на пошту для підтвердження емейлу

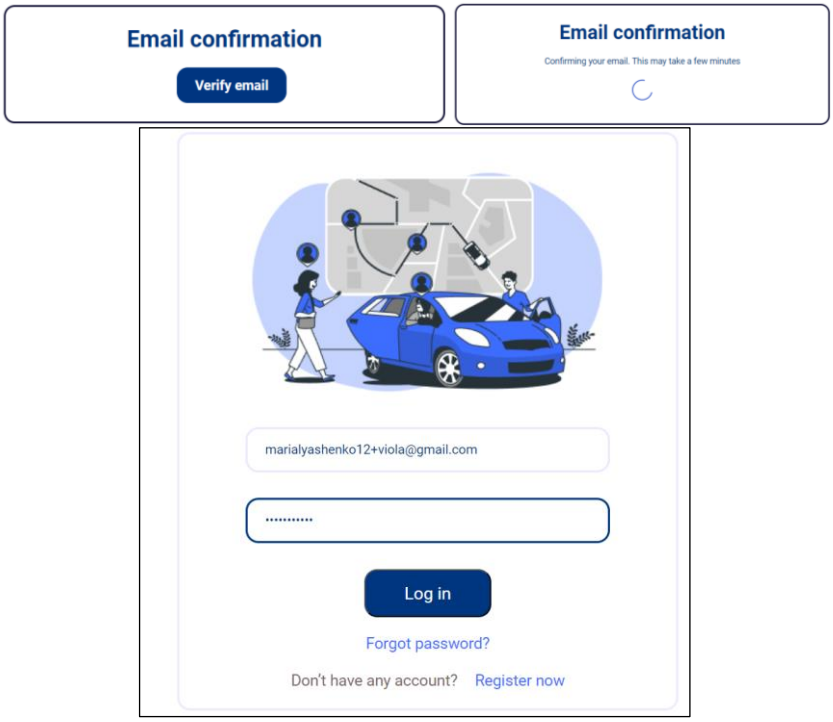


Рисунок 4.8 – Продовження логіки підтвердження імейлу

Після успішної авторизації користувачу відкривається можливість обрати свого інструктора шляхом натискання на кнопку обрати (див. рисунок 4.9)

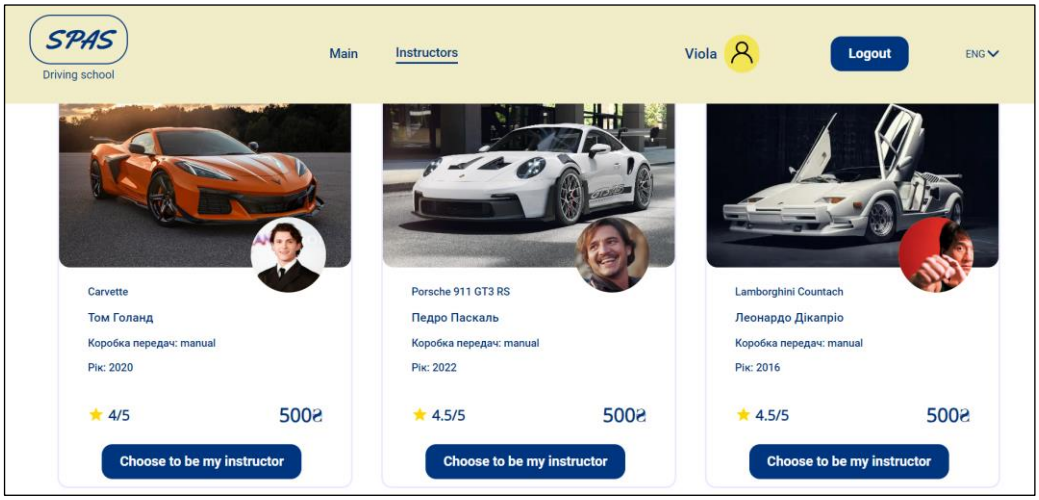


Рисунок 4.9 – Логіка обрання інструктора користувачем

Після того, як ми натиснули кнопку обрати інструктора, нам залишається тільки чекати на дозвіл з боку адміністратора . Це було зроблено таким принципом оскільки в реальному житті перш ніж проходити практику в автошколі учень має представити свої документи про успішне складання іспиту з теоретичного екзамену а також має бути старше 18 років та мати дозвіл на

керування автомобілем від лікарів. По логіці нашого застосунку ці всі папери перевіряє адміністратор та після цього вирішує чи дозволяти учню записатися до конкретного інструктора. Після успішного підтвердження адміністратора, користувачу відкриваються нові можливості переглядати розклад свого інструктора (див. рисунок 4.10) та список своїх забронюваних занять. Зайшовши на вкладку з розкладом в інструктора користувач може обрати день та час в який йому зручно було б прийти на заняття (див. рисунок 4.11).

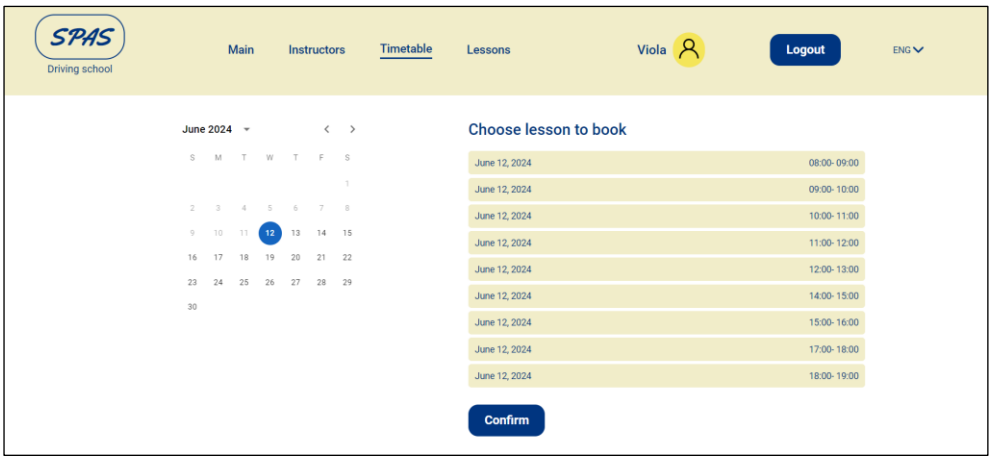


Рисунок 4.10 – Логіка перегляду розкладу інструктора

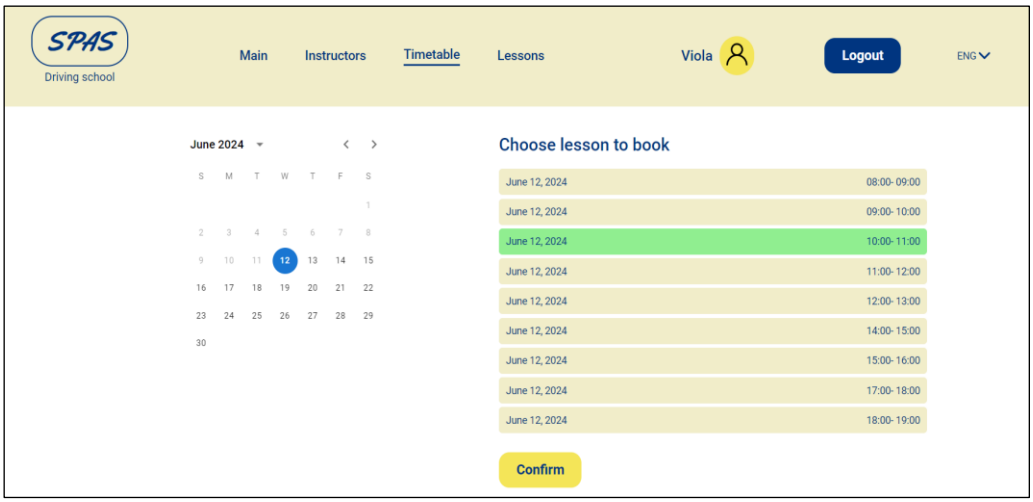


Рисунок 4.11 – Логіка обрання уроку

Після обрання зручного уроку, відкривається сторінка з підтвердженням уроку, та часу та посиланням на оплату уроку (див. рисунок 4.12).

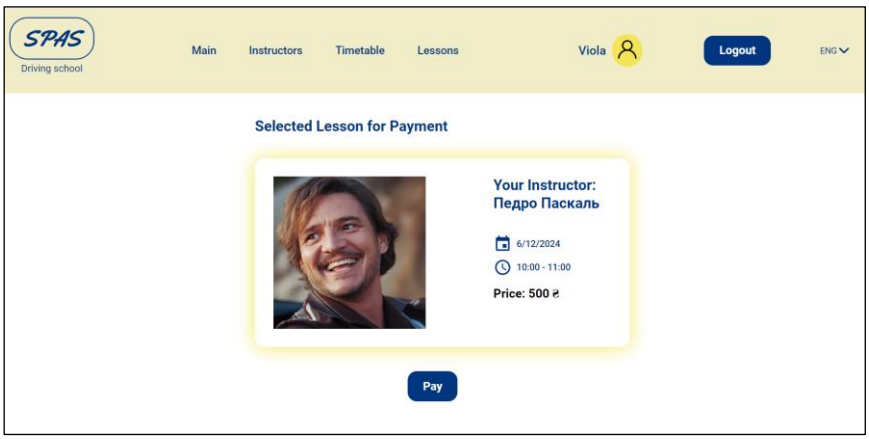


Рисунок 4.12 – Логіка підтвердження обраного уроку

Коли користувач натискає кнопку оплатити, йому відкривається вікно liqpay за допомогою якого він може сплатити заняття онлайн (див. рисунок 4.13).

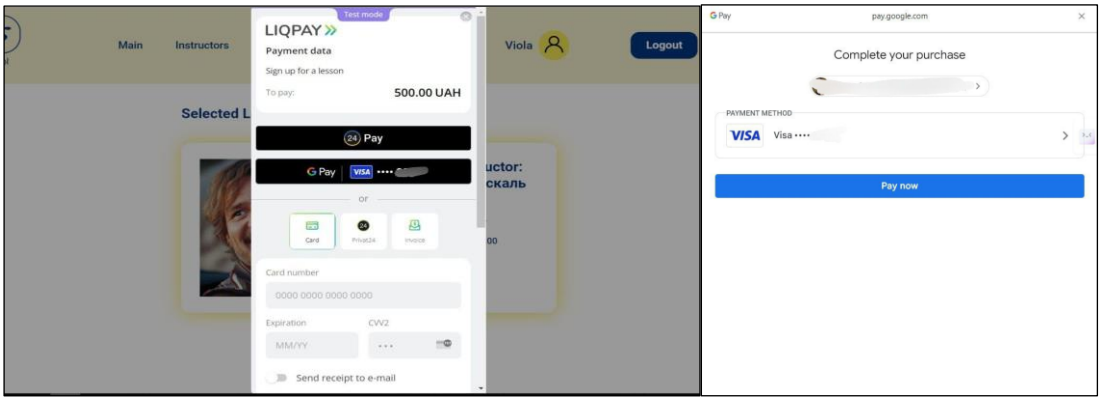


Рисунок 4.13 – Логіка процесу оплати обраного уроку

При успішній оплаті користувача одразу перекидає на головну сторінку та показує йому повідомлення що оплата успішна. (див. рисунок 4.14).

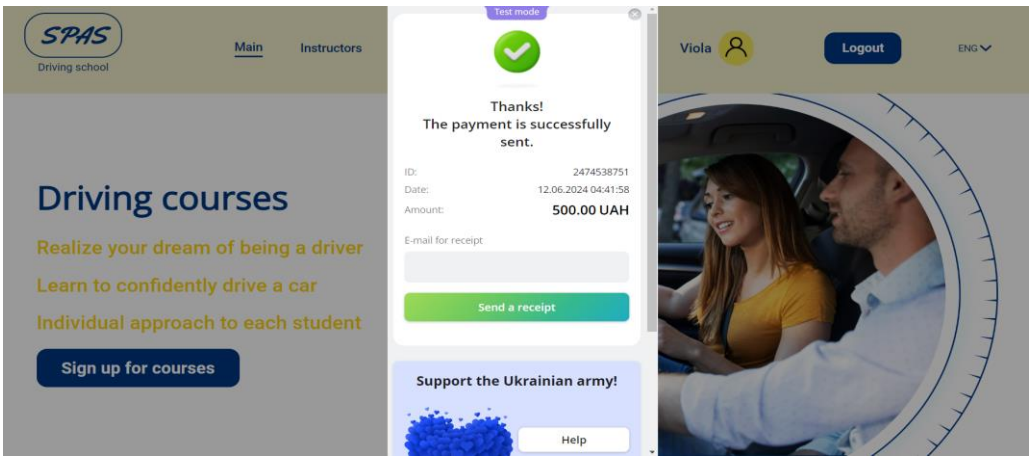


Рисунок 4.14 – Логіка успішного процесу оплати обраного уроку

Програмний код для реалізації оплати за допомогою liqpay:

```
const sha1Base64 = str => {
  const sha1Hash = CryptoJS.SHA1(str)
  const base64Hash = CryptoJS.enc.Base64.stringify(sha1Hash)
  return base64Hash
}

const processLiqPayPayment = (data, signature, successCb, errorCb,
cb) => {
  ;(window.LiqPayCheckoutCallback = function() {
    LiqPayCheckout.init({
      data: data,
      signature: signature,
      embedTo: "#liqpay_checkout",
      language: "en",
      mode: "popup" // embed || popup
    })
      .on("liqpay.callback", function(data) {
        if (data.status === "success") successCb()
        else if (data.status === "error") errorCb()
        const requestData = {
          status: data.status,
          amount: data.amount,
          paytype: data.paytype,
          end_date: data.end_date,
          order_id: data.order_id
        }
        cb(requestData)
      })
      .on("liqpay.ready", function(data) {
        // ready
      })
      .on("liqpay.close", function(requestData) {
        // close
      })
  })(
```

Цей код створює функцію sha1Base64, яка генерує хеш SHA1 заданого рядка і кодує його у формат Base64. Функція processLiqPayPayment відповідає за ініціювання оплати через LiqPay. Вона ініціалізує чекаут LiqPay, обробляє події під час процесу оплати і викликає зазначені функції залежно від результату операції.

Після цього користувач може переглянути всі заброньовані уроки у вкладці уроки (див. рисунок 4.15).

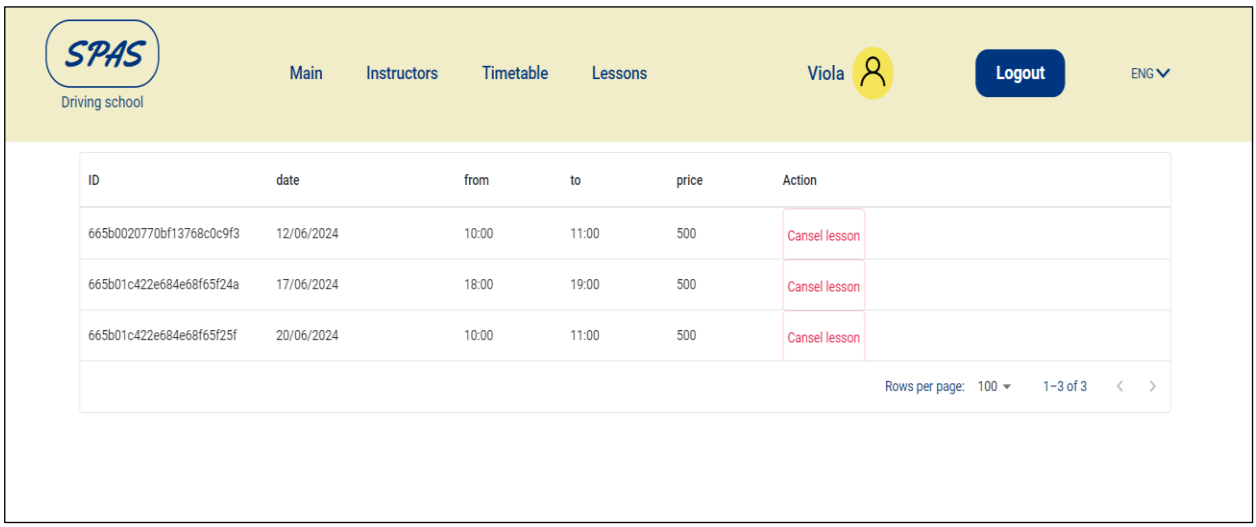


Рисунок 4.15 – Список уроків даного користувача

Також користувач має право скасовувати заняття, однак треба зауважити що заняття може бути відмінене не пізніше ніж за день до його проведення. При відміні заняття користувачем, воно знову стає вільним для запису інших користувачей (див. рисунок 4.16).

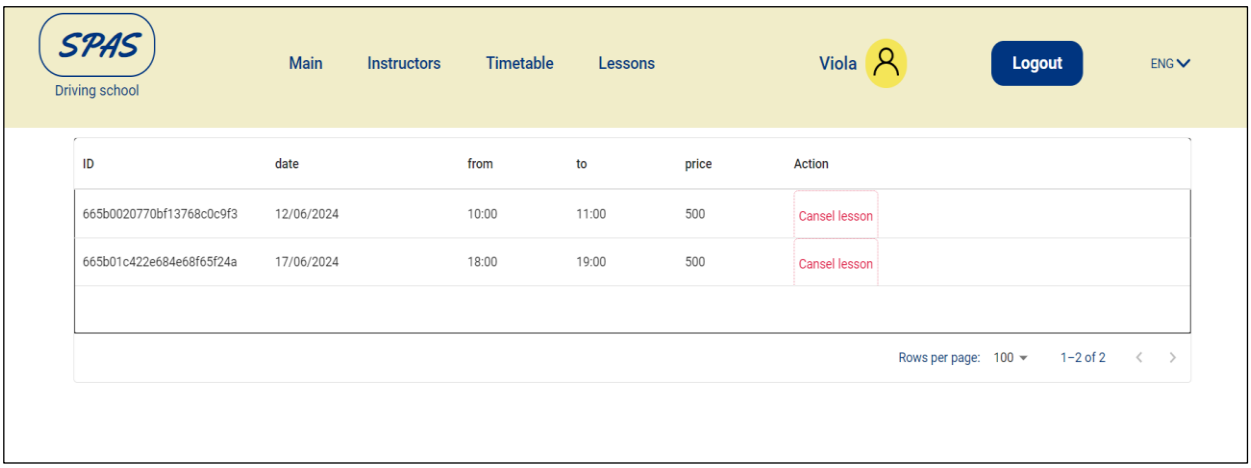


Рисунок 4.16 – Відміна урока користувачем

Також як зазначалося раніше, користувач може редагувати інформацію про себе та додавати фотокарточку, це можна зробити у вкладці мій профіль (див. рисунок 4.17). Натиснувши кнопку зберегти зміни, користувач автоматично змінить інформацію про себе і вона відобразиться одразу в нього в профілі (див. рисунок 4.18)

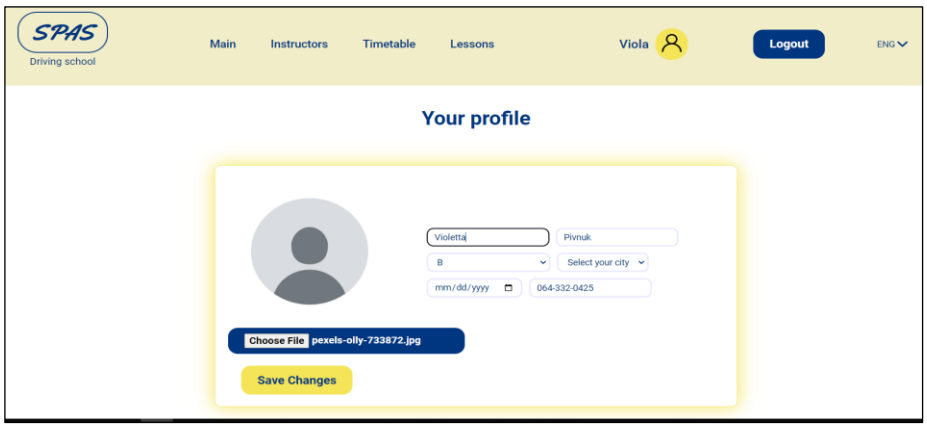


Рисунок 4.17 – Зміна користувачем інформації про себе

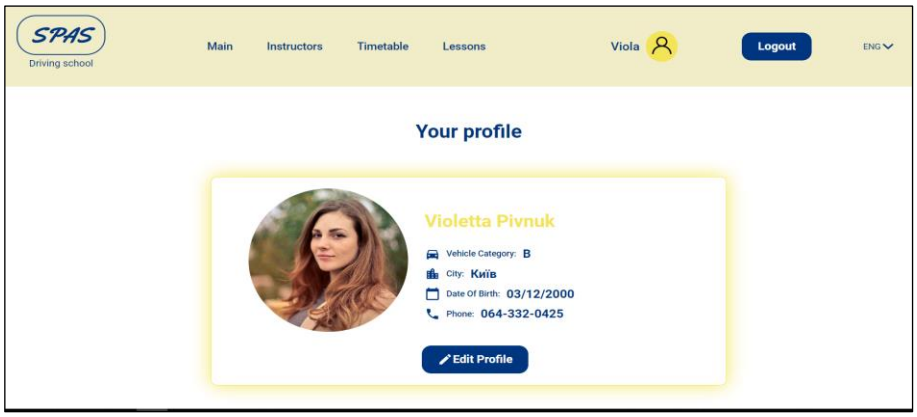


Рисунок 4.18 – Збереження зміни інформації

Також оскільки коли користувач заходить в систему в нього постійно оновлюється його токен, і на кожен запит до бд іде перевірка чи токен ще валідний, було створено кастомний хук, що надає зручний спосіб взаємодії з сервером для отримання, оновлення, створення та видалення даних. Крім того, він автоматично перевіряє та оновлює токен автентифікації, якщо потрібно, для забезпечення безпеки та доступу до захищених ресурсів.

Програмний код для реалізації цього хуку:

```
import { useEffect, useState } from "react";
import axios from "axios";
import Cookies from "js-cookie";

const useFetch = () => {
    const [data, setData] = useState([]);
    const [loading, setLoading] = useState(false);
    const [error, setError] = useState(false);
```

```
        const handleRequest = async (method, endpoint, payload =
null) => {
            setLoading(true);
            try {
                await checkAndRefreshTokenIfNeeded();
                   const config = {
                    method,
                    url: endpoint,
                    data: payload,
                    withCredentials: true
                };
                try {
                    const res = await axios(config);
                    console.log("Response from UseFetch: ", res);
                    setData(res.data);
                    return res.data;
                } catch (err) {
                    console.log("err from UseFetch: ", err);
                    console.log("err from UseFetch: ", err.response);

                    setError(err.response);
                    return err;
                }

            } catch (err) {
                setError(err);
            }
            setLoading(false);
        };

        const getData = async endpoint => {
            return await handleRequest("GET", endpoint);
        };
        const deleteData = async endpoint => {
            await handleRequest("DELETE", endpoint);
        };

        const putData = async (endpoint, payload) => {
            await handleRequest("PUT", endpoint, payload);
        };

        const postData = async (endpoint, payload) => {
            await handleRequest("POST", endpoint, payload);
        };

        const patchData = async (endpoint, payload) => {
            let res = await handleRequest("PATCH", endpoint, payload);
            return res;
        };

        const checkAndRefreshTokenIfNeeded = async () => {
            if (isTokenExpired()) {
                await refreshAuthToken();
            }
        };

        const isTokenExpired = () => {
            const storedData = JSON.parse(localStorage.getItem("user"));
```

```
            const tokenExpire = storedData.tokenExpire;
            return tokenExpire && new Date(tokenExpire) <= new Date();
        };

        const refreshAuthToken = async () => {
            try {
                    const response = await axios.post(
                    "http://localhost:3000/api/auth/token",
                    {},
                    {
                        withCredentials: true
                    }
                );
                console.log("new token: ", response);

                const                        storedData                        =
JSON.parse(localStorage.getItem("user"));
                storedData.tokenExpire = response.data.tokenExpire;
                localStorage.setItem("user", JSON.stringify(storedData));
                return response;
            } catch (error) {
                console.error("Error refreshing token", error);
                throw error;
            }
        };

        return { data, deleteData, putData, patchData, postData, error,
getData };
    };
```

Перейдемо до логіки яка доступна інструктору. При потраплянні на сайт, як вже зазначалося раніше інструктора теж перенаправляє на головну сторінку. Тому щоб дістатися до функціоналу доступному тілки користувачам з ролю інструктор, треба пройти авторизацію. Після успішної авторизації, інструктор бачить усі доступні йому сторінки (див. рисунок 4.19).

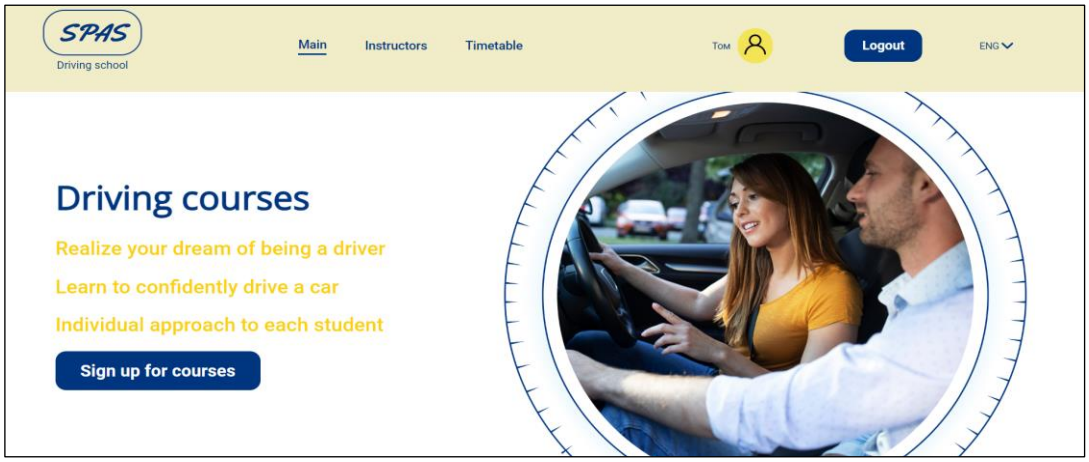


Рисунок 4.19 – Сторінка інструктора після успішної авторизації

В принципі, як можна побачити, в інструктора та в студента доволі схожий функціонал. Інструктор у вкладці розкладу може переглянути усі свої заняття та хто з учнів на них записан (див. рисунок 4.20), також інструктор може відмінити якесь заняття, але тільки якщо на нього ще не записан жоден учень (див. рисунок 4.21).

На рисунку 4.20 також можна помітити, що заняття на які записані учні висвічуються червоним як забронювані та з підписами імен учнів що їх забронювали, а заняття які були відмінені самим інструктором просто висвічуються червоним.

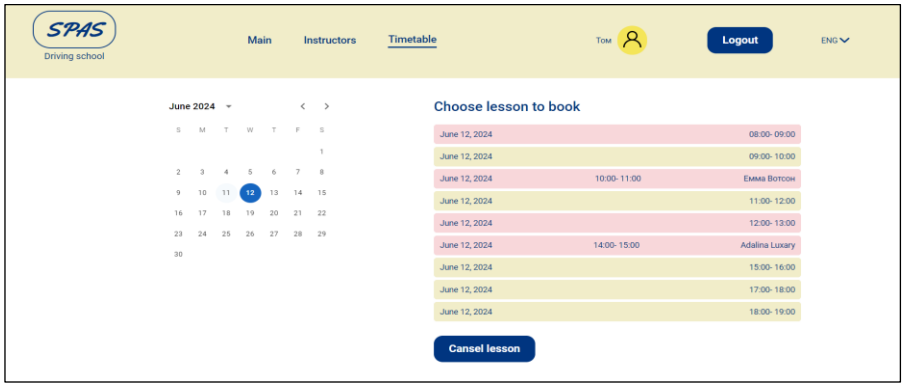


Рисунок 4.20 – Сторінка розкладу інструктора

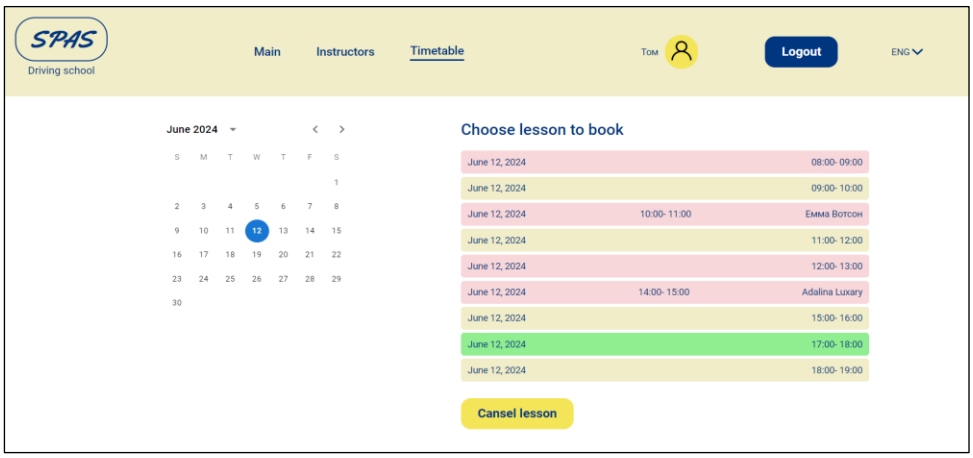


Рисунок 4.21 – Функціонал відміни заняття інструктором

Також інструктор має змогу переглядати інформацію про свій профіль однак редагування інформації про інструктора йому не доступне (див. рисунок 4.22).

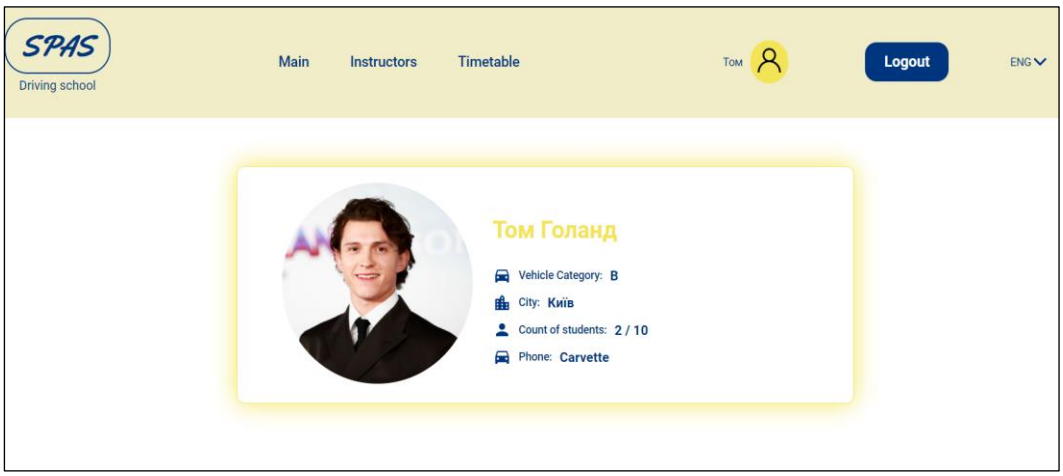


Рисунок 4.22 – Функціонал перегляду інформації про себе

Перейдемо до функціоналу адміністратора. Коли ми заходимо на адмінський сайт, нам першим пропонується форма авторизації (див. рисунок 4.23).

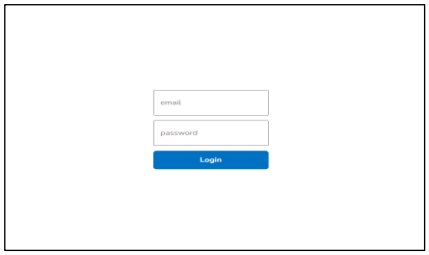


Рисунок 4.23 – Авторизація на адміні

Після успішної авторизації нам показується головна сторінка та в боковом меню ми можемо побачити всі наші доступні сторінки (див. рисунок 4.24).

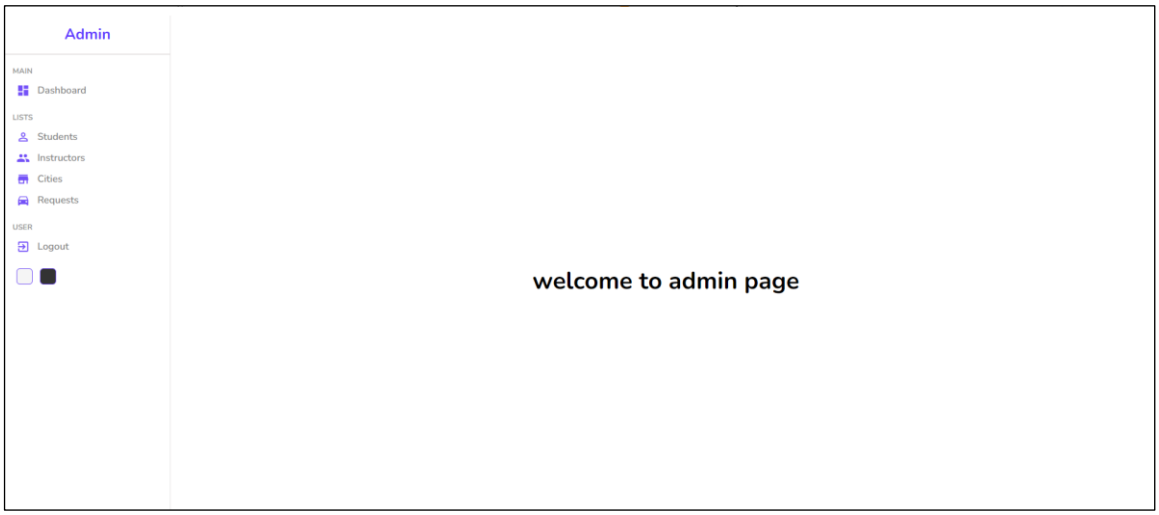


Рисунок 4.24 – Головна сторінка на адміні

Як вже видно по інтерфейсу, адміну доступні такі роути як, перегляд усіх студентів (рисунок 4.25), перегляд усіх інструкторів (рисунок 4.26) та додавання (рисунок 4.27) або видалення інструктора (видалення тільки за умови що до інструктора ніхто не записаний), перегляд списку міст (рисунок 4.28), а також перегляду списку запитів (рисунок 4.29) на запис від студентів до інструкторів які адмін може або погодити або заборонити.

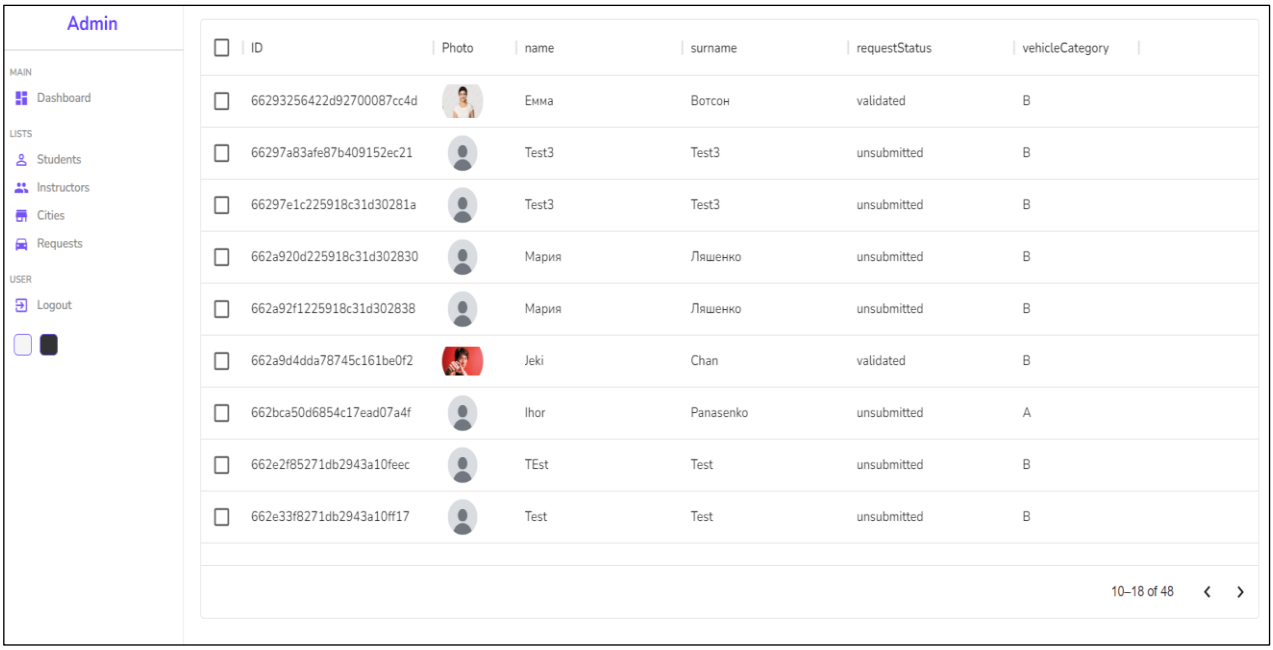


Рисунок 4.25 – Перегляд усіх студентів на адміні

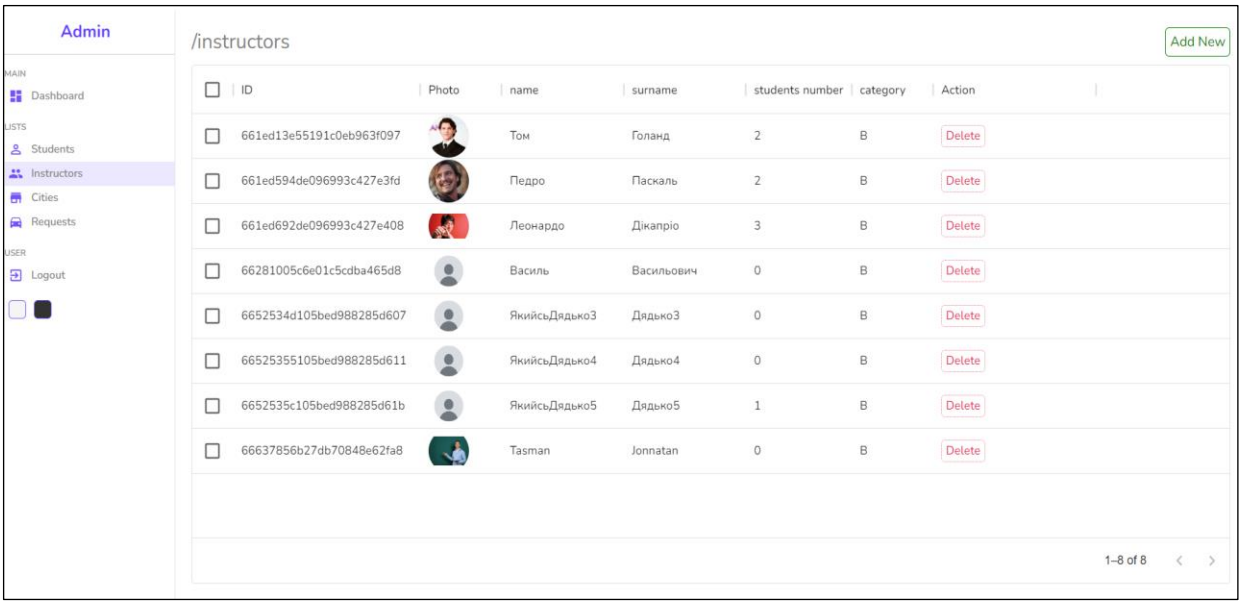


Рисунок 4.26 – Перегляд усіх інструкторів на адміні

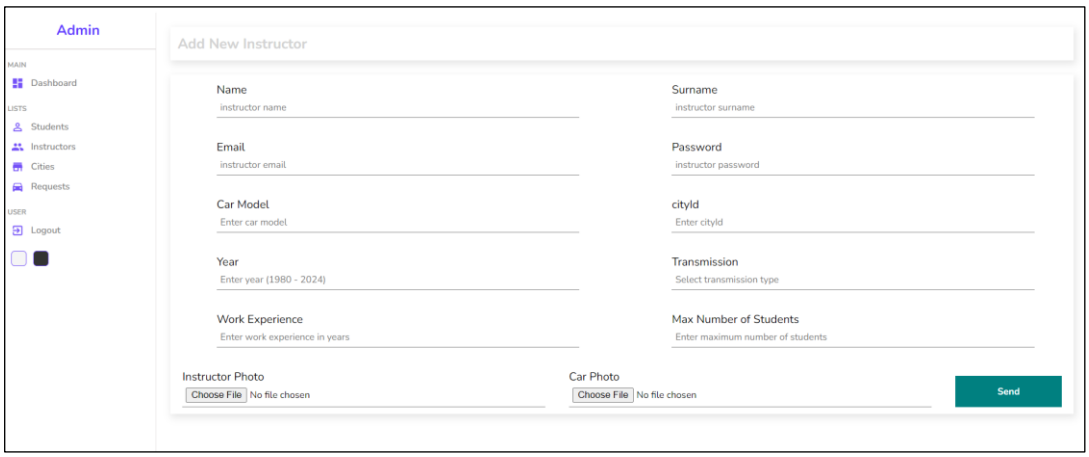


Рисунок 4.27 – Додавання нового інструктора на адміні

Рисунок 4.28 – Перегляд списку міст на адміні

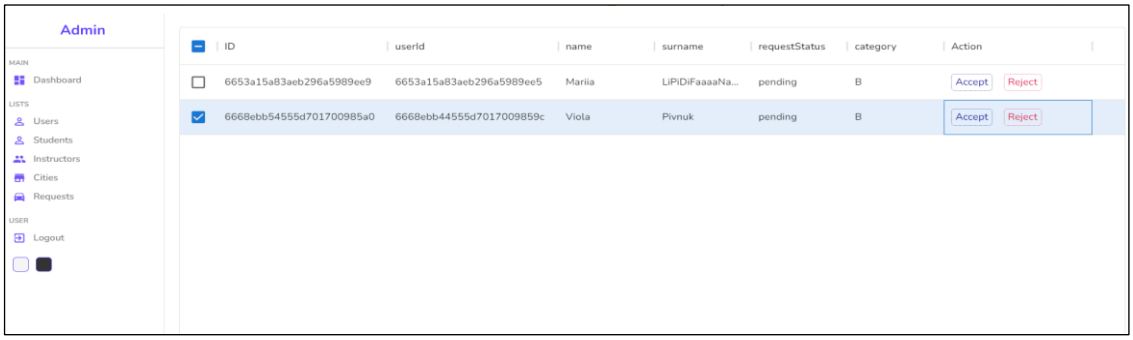


Рисунок 4.29 – Перегляд списку запитів на адміні

Також слід відмітити що для адмінського сайту також було створено кастомний хук, що надає зручний спосіб взаємодії з сервером для отримання, оновлення, створення та видалення даних за принципом як і на клієнті. І також

для збереження стану використовуючи контекст та редуктор було написано функцію, також схожу як і на клієнті.

Було розглянуто основні сторінки та функціональні можливості фронтенд частини застосунку для організації практичних занять в автошколі, який спрямований на задоволення потреб не тільки студентів, а й інструкторів та адмінів. Особливу увагу приділено найцікавішим частинам коду, які демонструють, як застосунок забезпечує ефективну та правильну роботу.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Обгрунтування вибору виду тестування

Для забезпечення якості та надійності мобільного додатку, призначеного для організації практичних занять в автошколі, було обрано відповідні методи тестування. Враховуючи особливості цього застосунку, ми вирішили використати кілька видів тестування, кожен з яких має свої переваги та обґрунтування.

Регресійне тестування – це набір тестів, спрямованих на виявлення дефектів у вже протестованих модулях додатка. Робиться це зовсім не для того, щоб остаточно переконатися у відсутності багів, а для пошуку та виправлення регресійних помилок. Регресійні помилки – ті ж баги, але з’являються вони не при написанні програми, а при додаванні в існуючий білд нової частини програми або виправлення інших багів, що і стає причиною виникнення нових дефектів у вже протестованому продукті [9].

Тестування реальними користувачами, також відоме як користувацьке тестування (user testing) або тестування з залученням кінцевих користувачів, є важливим етапом процесу розробки програмного забезпечення. Воно дозволяє виявити проблеми зручності використання, функціональні недоліки та інші аспекти, які можуть вплинути на досвід користувача.

Unit тести – це автоматичні тести, які перевіряють невеликі частини коду, такі як функції або методи, ізольовано від решти системи.Вони дають змогу розробникам переконатися, що кожна частина коду працює правильно і відповідає очікуваній поведінці. Вони пишуться мовою програмування і можуть бути запущені автоматично для швидкої перевірки коду при кожній його зміні [10].

Комплексне застосування різних видів тестування дозволяє виявити та виправити помилки на різних етапах розробки, забезпечуючи стабільну та ефективну роботу додатку у виробничому середовищі.

5.2 Опис тестування

В першу чергу було створено Юніт-тести оскільки вони є важливою складовою будь-якого розробленого програмного продукту. Наведемо приклад

Юніт тестів для компоненту Register . Ці тести включають перевірку правильності відображення компоненту, обробки змін у полях форми, валідації введених даних, перевірки відправки даних на сервер, обробки помилок під час реєстрації та перенаправлення користувача після успішної реєстрації.

Наведемо нижче програмний код тестів для компоненту Реєстрації:

```
      test("renders Register component correctly", () => {
              renderComponent();

expect(screen.getByPlaceholderText(/firstName/i)).toBeInTheDocument();

expect(screen.getByPlaceholderText(/lastName/i)).toBeInTheDocument();

expect(screen.getByPlaceholderText(/email/i)).toBeInTheDocument();

expect(screen.getByPlaceholderText(/password/i)).toBeInTheDocument();

expect(screen.getByPlaceholderText(/phoneNumber/i)).toBeInTheDocument();
          });

          test("handles form field changes and validates input fields",
async () => {
              renderComponent();

              fireEvent.change(screen.getByPlaceholderText(/firstName/i),  {
target: { value: "John" } });

expect(screen.getByPlaceholderText(/firstName/i)).toHaveValue("John");

              fireEvent.change(screen.getByPlaceholderText(/lastName/i),   {
target: { value: "Doe" } });

expect(screen.getByPlaceholderText(/lastName/i)).toHaveValue("Doe");

              fireEvent.change(screen.getByPlaceholderText(/email/i),      {
target: { value: "invalid email" } });

expect(screen.getByPlaceholderText(/email/i)).toHaveValue("invalid email");

              fireEvent.change(screen.getByPlaceholderText(/phoneNumber/i),
{ target: { value: "12345" } });

expect(screen.getByPlaceholderText(/phoneNumber/i)).toHaveValue("12345");

              fireEvent.change(screen.getByPlaceholderText(/password/i),   {
target: { value: "short" } });

expect(screen.getByPlaceholderText(/password/i)).toHaveValue("short");

              fireEvent.change(screen.getByPlaceholderText(/email/i),      {
target: { value: "john.doe@example.com" } });
```

```
expect(screen.getByPlaceholderText(/email/i)).toHaveValue("john.doe@example
.com");
          });

          test("prevents form submission with invalid data", async () => {
              renderComponent();

              fireEvent.change(screen.getByPlaceholderText(/firstName/i),  {
target: { value: "John" } });
              fireEvent.change(screen.getByPlaceholderText(/lastName/i),   {
target: { value: "Doe" } });

              // Invalid email
              fireEvent.change(screen.getByPlaceholderText(/email/i),      {
target: { value: "invalid email" } });

              // Invalid phone number
              fireEvent.change(screen.getByPlaceholderText(/phoneNumber/i),
{ target: { value: "12345" } });

              // Invalid password
              fireEvent.change(screen.getByPlaceholderText(/password/i),   {
target: { value: "short" } });

              fireEvent.submit(screen.getByRole("button",       {       name:
/register.link_registration/i }));

              // Ensure form does not submit with errors
              expect(screen.queryByText(/Form      data      is      valid,
submitting.../i)).not.toBeInTheDocument();
          });
          test("submits form data to the server on successful validation",
async () => {
              renderComponent();

              fireEvent.change(screen.getByPlaceholderText(/firstName/i),  {
target: { value: "John" } });
              fireEvent.change(screen.getByPlaceholderText(/lastName/i),   {
target: { value: "Doe" } });
              fireEvent.change(screen.getByPlaceholderText(/email/i),      {
target: { value: "john.doe@example.com" } });
              fireEvent.change(screen.getByPlaceholderText(/phoneNumber/i),
{ target: { value: "+380501234567" } });
              fireEvent.change(screen.getByPlaceholderText(/password/i),   {
target: { value: "StrongPassword123" } });

              fireEvent.submit(screen.getByRole("button",       {       name:
/register.link_registration/i }));

              // Simulate successful server response (mocked)
              await waitFor(() => {
                  expect(mock.history.post.length).toBe(1);

expect(mock.history.post[0].data).toEqual(expect.stringContaining("john.doe
@example.com"));
              });
          });
```

Ці юніт-тести призначені для перевірки компоненту реєстрації (Register) в React-додатку. Вони охоплюють різні аспекти роботи форми реєстрації, включаючи рендеринг, зміну полів, валідацію та відправку даних на сервер. Нижче наведено опис кожного тесту:

- рендеринг компонента реєстрації, цей тест перевіряє, чи всі поля форми реєстрації відображаються правильно. Використовується метод renderComponent для рендерингу компоненту, а потім перевіряється наявність кожного поля за допомогою screen.getByPlaceholderText. Він гарантує, що всі необхідні поля форми присутні на сторінці, що є основою для подальшої взаємодії користувача з формою;
- обробка змін полів форми та валідація, цей тест перевіряє, чи правильно обробляються зміни в полях форми і чи правильно відображаються введені значення. Він перевіряє, чи оновлюються значення полів після зміни їх користувачем. Він забезпечує правильну обробку введених користувачем даних і дає можливість негайного зворотного зв'язку, що покращує користувацький досвід;
- запобігання відправки форми з некоректними даними, цей тест перевіряє, чи запобігає форма відправці при наявності некоректних даних. Він змінює поля форми на некоректні значення і перевіряє, чи не відправляється форма. Він гарантує, що форма не буде відправлена на сервер з некоректними даними, що допомагає уникнути помилок на стороні сервера і покращує загальну стабільність додатку;
- відправка даних форми на сервер після успішної валідації, цей тест перевіряє, чи відправляються дані форми на сервер після успішної валідації. Він змінює поля форми на коректні значення і перевіряє, чи відправляються вони на сервер. Він гарантує, що дані форми правильно відправляються на сервер після успішної валідації, що є критичним для процесу реєстрації користувача.

Наступним видом тестування було регресійне тестування, що дозволило підтримувати високу якість та стабільність веб-додатку, мінімізуючи ризики

виникнення критичних багів. Завдяки цьому можна переконатися, що користувачі отримають надійний та безпечний продукт, який працює коректно навіть після внесення нових змін. Це тестування також сприяло зменшенню витрат на виправлення багів на пізніх етапах розробки, що позитивно вплинуло на загальну ефективність процесу розробки.

Завершальним було тестування реальними користувачами є важливим етапом у процесі розробки програмного забезпечення, оскільки дозволяє виявити та усунути проблеми, які можуть вплинути на задоволеність користувачів продуктом. Залучення реальних користувачів до тестування допомагає створити більш інтуїтивний, зручний та ефективний продукт, який відповідає потребам та очікуванням його аудиторії.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи була розроблена візуальна частина програмної системи для автоматизації проведення практичних занять в автошколі. Під час проектування був проведений аналіз предметної галузі, сформульовані вимоги до системи, описана архітектура та проектування фронтенду. На основі аналізу ринку та предметної галузі були визначені основні конкуренти, необхідний функціонал та цільова аудиторія додатку.

Основним функціоналом розробленого фронтенду є управління розкладом, з можливістю запису на заняття та їх оплати, а також відстеження власного прогресу з можливістю відмічення вивчених вправ. Для розробки фронтенду був використаний набір технологій, зокрема React, який забезпечує легку організацію компонентів та їх перевикористування. Для глобального керування станом додатку використовувався Redux, що дозволило ефективно керувати даними та їх взаємодією між компонентами.

Для стилізації компонентів був використаний препроцесор SCSS, що дозволив швидко та зручно створити зрозумілий та підтримуваний CSS код. Крім того, для управління глобальним станом додатку та передачі даних між компонентами використовувалися контексти та хуки. Для взаємодії з сервером та отримання даних була використана бібліотека Axios, що надає можливість легко виконувати HTTP запити.

Спроектований фронтенд має задовольнити потреби як інструкторів, так і учнів автошколи, надаючи зручний та ефективний інструмент для планування, ведення та аналізу практичних занять. Таким чином, розроблений фронтенд має потенціал стати необхідним інструментом для автошколи, спрощуючи та оптимізуючи процес проведення практичних занять для всіх учасників навчального процесу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Що варто знати про практичний іспит в сервісному центрі МВС? URL: https://hsc.gov.ua/index/poslugi/faq/shho-varto-znati-pro-praktichnij-ispit-v-servisnomu-tsentri-mvs/ (дата звернення: 12.05.2024).

2. Що таке Реакт? web developer Дмитро Берестень URL: https://web-developer.in.ua/assets/articles/react/react-introduction/react-introduction.html (дата звернення: 12.05.2024).

3. i18next documentation URL: https://www.i18next.com/ (дата звернення: 12.05.2024).

4. Знати CSS – недостатньо: що таке SCSS та як з ним працювати. Розбір синтаксису URL: https://highload.today/uk/scss/ (дата звернення: 12.05.2024).

5. MongoDB URL: https://www.mongodb.com/ (дата звернення: 12.05.2024).

6. Діаграма прецедентів, wikipedia вільна енциклопедія URL: https://uk.wikipedia.org/wiki/%D0%94%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0_%D0%BF%D1%80%D0%B5%D1%86%D0%B5%D0%B4%D0%B5%D0%BD%D1%82%D1%96%D0%B2 (дата звернення: 12.05.2024).

7. Liqpay URL: https://www.liqpay.ua/en/about/info (дата звернення: 13.06.2024).

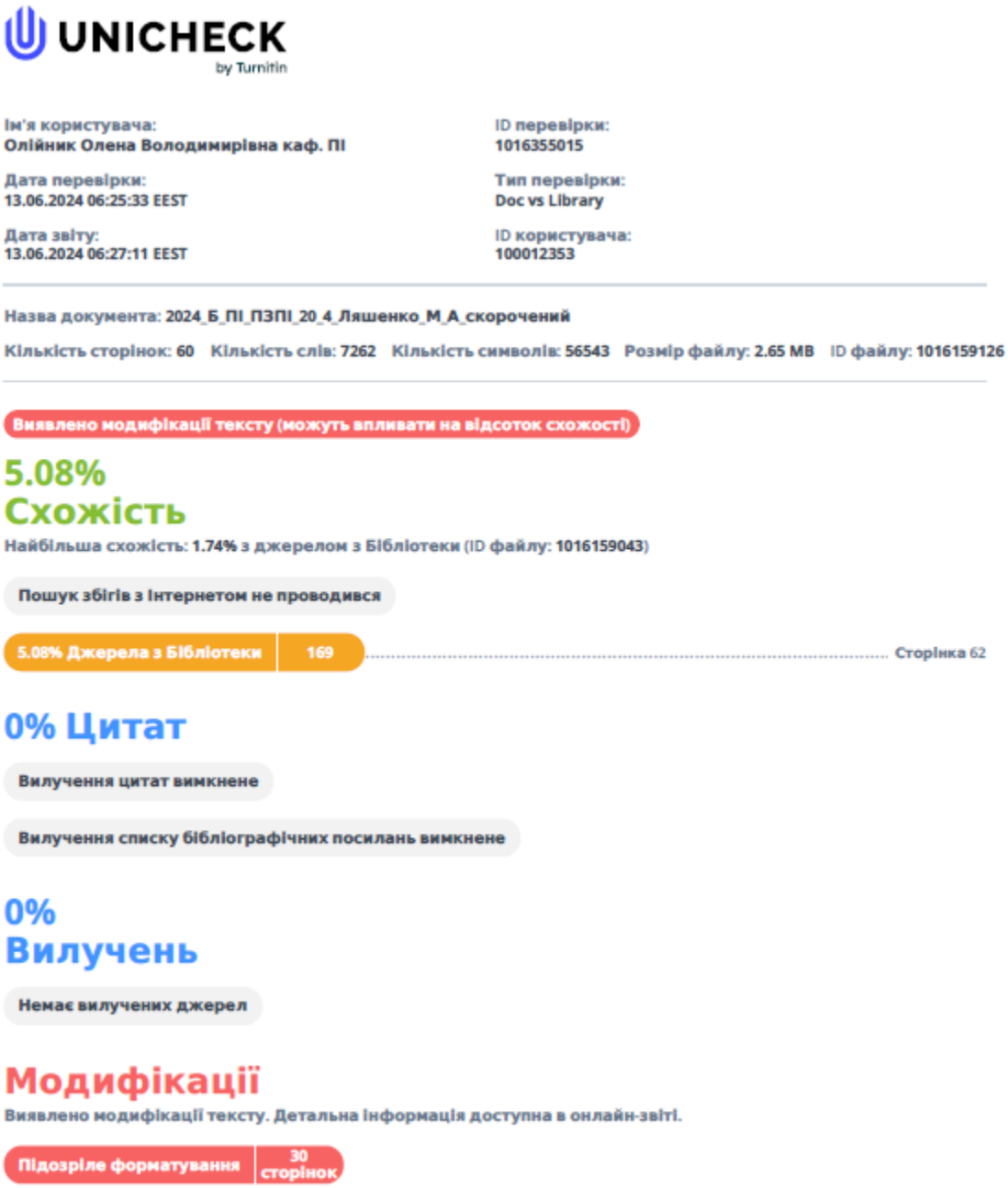
8. Redux URL: https://redux.js.org/ (дата звернення: 13.06.2024).

9. Регресійне тестування, qalight Центр підготовки IT фахівців URL: https://qalight.ua/baza-znaniy/regresijne-testuvannya/ (дата звернення: 13.06.2024).

10. Що таке Unit тести і як їх писати foxminded URL: https://foxminded.ua/yunit-testy/ (дата звернення: 13.06.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016355015

Дата перевірки:
13.06.2024 06:25:33 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
13.06.2024 06:27:11 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ_20_4_Ляшенко_М_А_скорочений

Кількість сторінок: 60 Кількість слів: 7262 Кількість символів: 56543 Розмір файлу: 2.65 MB ID файлу: 1016159126

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.08%
Схожість

Найбільша схожість: 1.74% з джерелом з Бібліотеки (ID файлу: 1016159043)

Пошук збігів з інтернетом не проводився

5.08% Джерела з Бібліотеки 169 ..... Сторінка 62

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

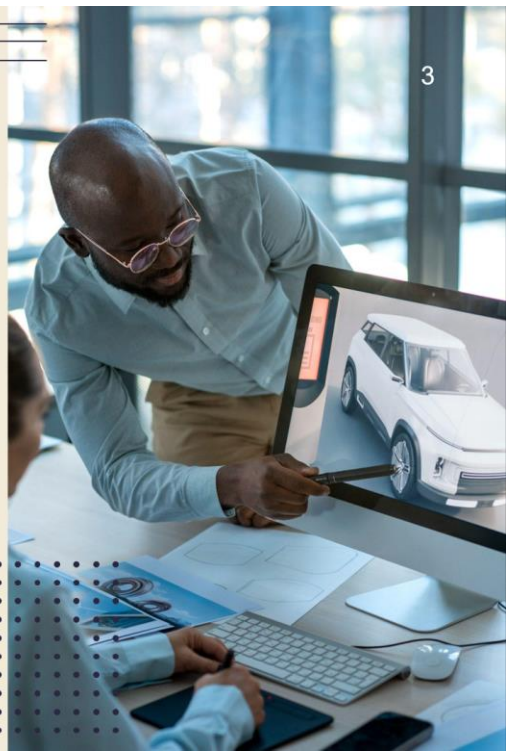
Підозріле форматування 30 сторінок

ДОДАТОК Б

Слайди презентації

Аналіз ринку

- потреба у цифровізації навчальних процесів, що дозволяє підвищити ефективність організації занять
- спрощення взаємодії між учнями та інструкторами
- забезпечення зручний і зрозумілий інтерфейс для керування розкладом

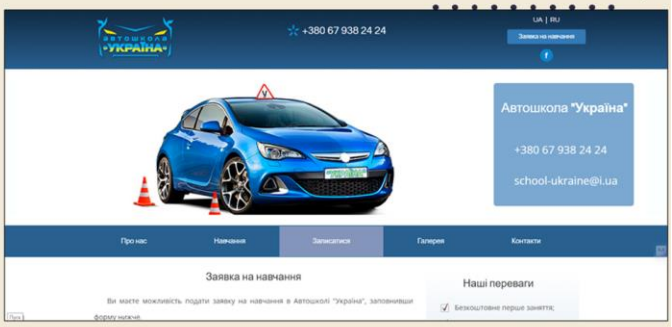


Аналіз конкурентів

Сайт автошколи «Україна»

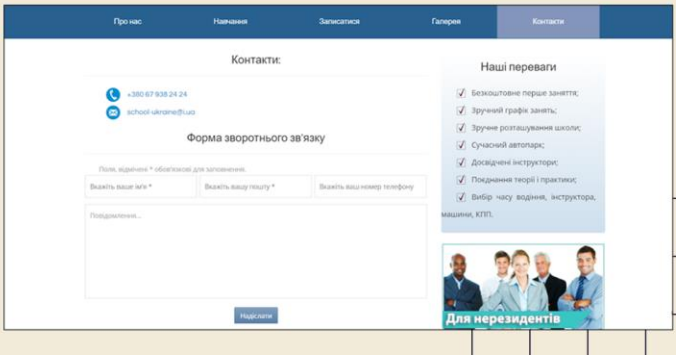
+

- зручний інтерфейс;
- наявність детальної інформації;
- можливість перегляду фотографій;
- легкий спосіб зв'язку з автошколою через форму зворотнього зв'язку.



–

- відсутність можливості обирати конкретного інструктора для навчання;
- не наявність системи онлайн бронювання.



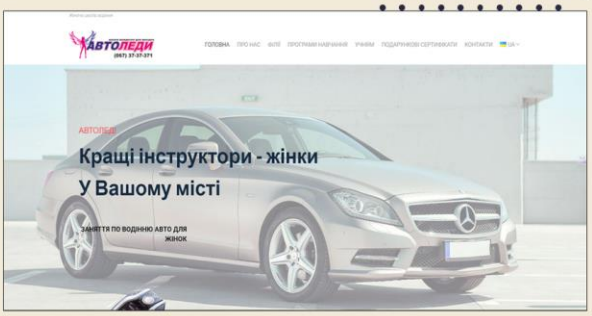
Аналіз конкурентів

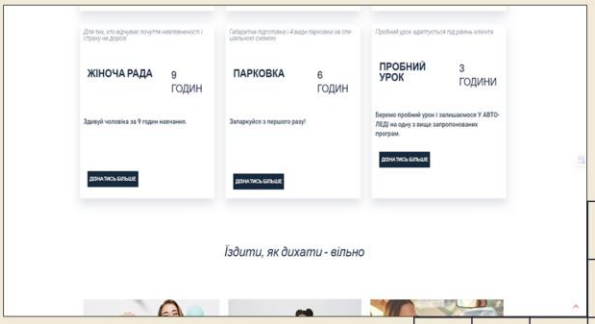
Жіноча школа водіння «АвтоЛеді».

\+
- зручність та комфорт;
- присутня локалізація сайту;
- зрозумілий інтерфейс;
- зменшення стереотипів;
- зручне розташування:

\-
- відсутність електронного запису;
- відсутність оплати за курс.

5





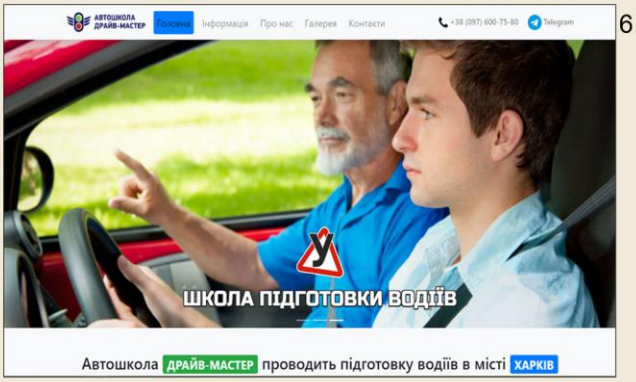
Аналіз конкурентів

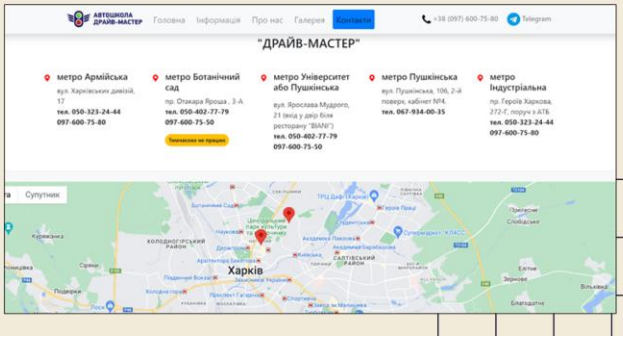
Сайт автошколи «Драйв-мастер»

\+
- наявність в усіх районах;
- простий та лаконічний дизайн сайту;
- гарні відгуки.

\-
- відсутність електронного запису;
- відсутність оплати за курс.

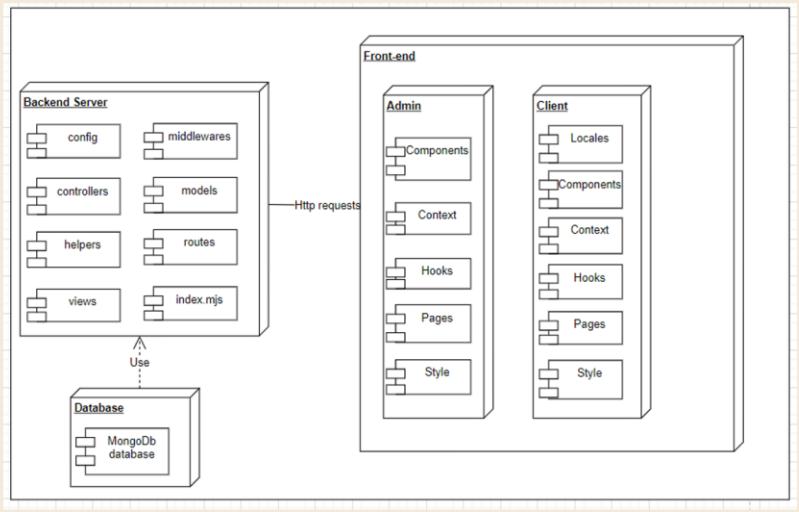
6





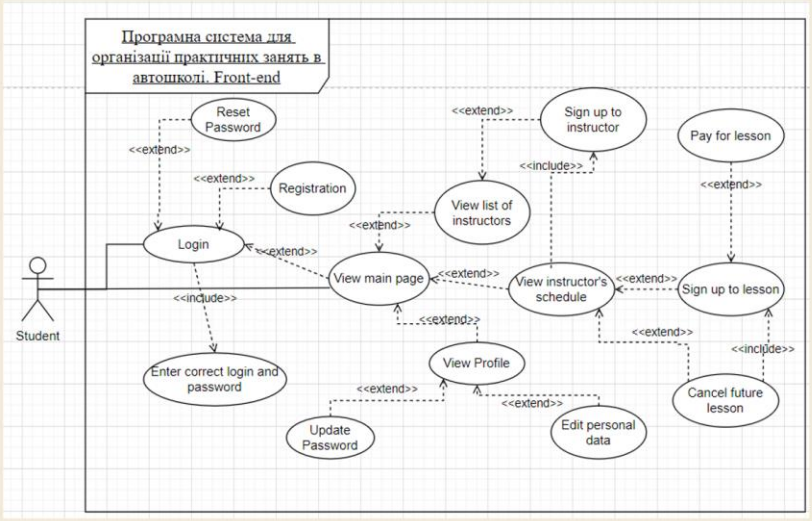
7

Схема архітектури програмної системи



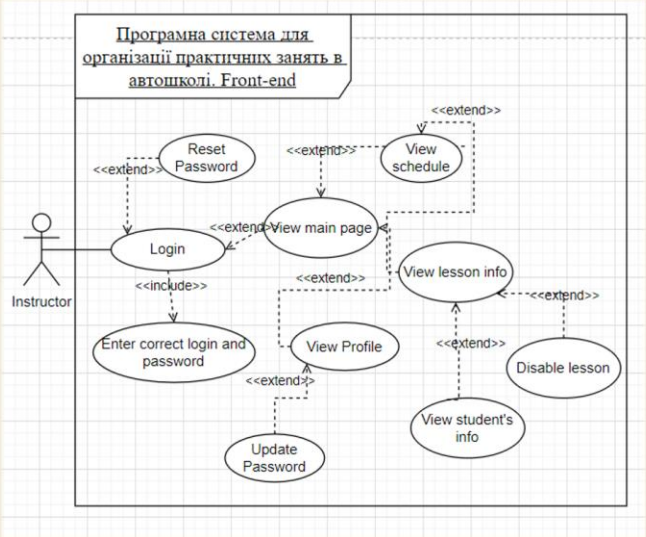
8

Use Case діаграма для студента



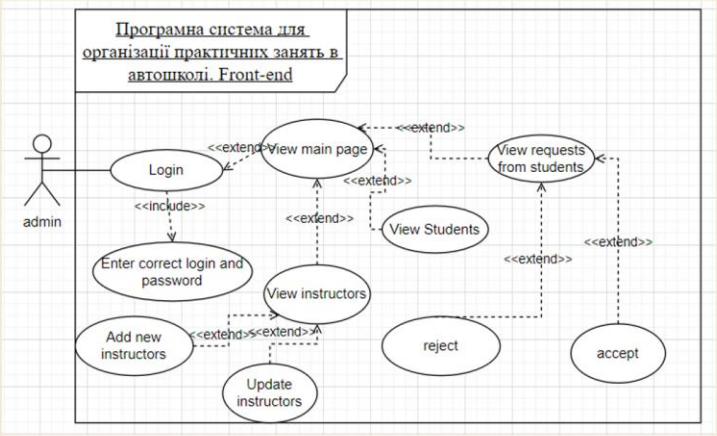
9

Use Case діаграма для інструктора



10

Use Case діаграма для адміна



11

Технології

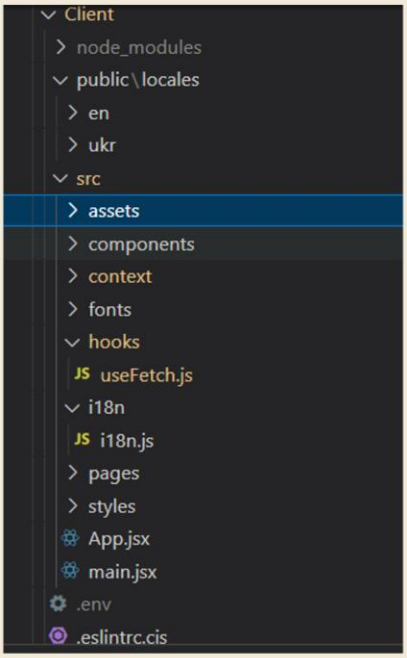
LIQPAY

AXIOS

JS

Redux

12

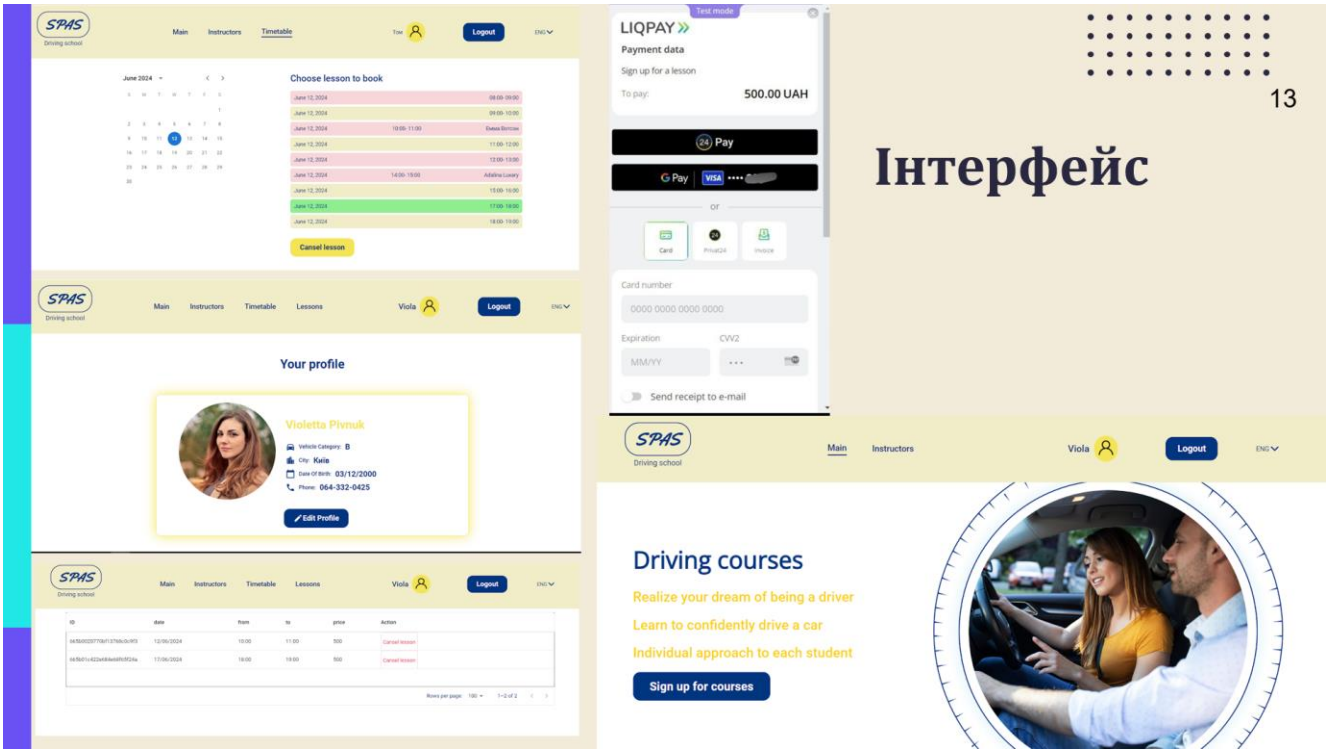


Структура файлів проекту

1. assets;
2. components;
3. fonts;
4. hooks;
5. i18n;
6. context;
7. pages;
8. styles.

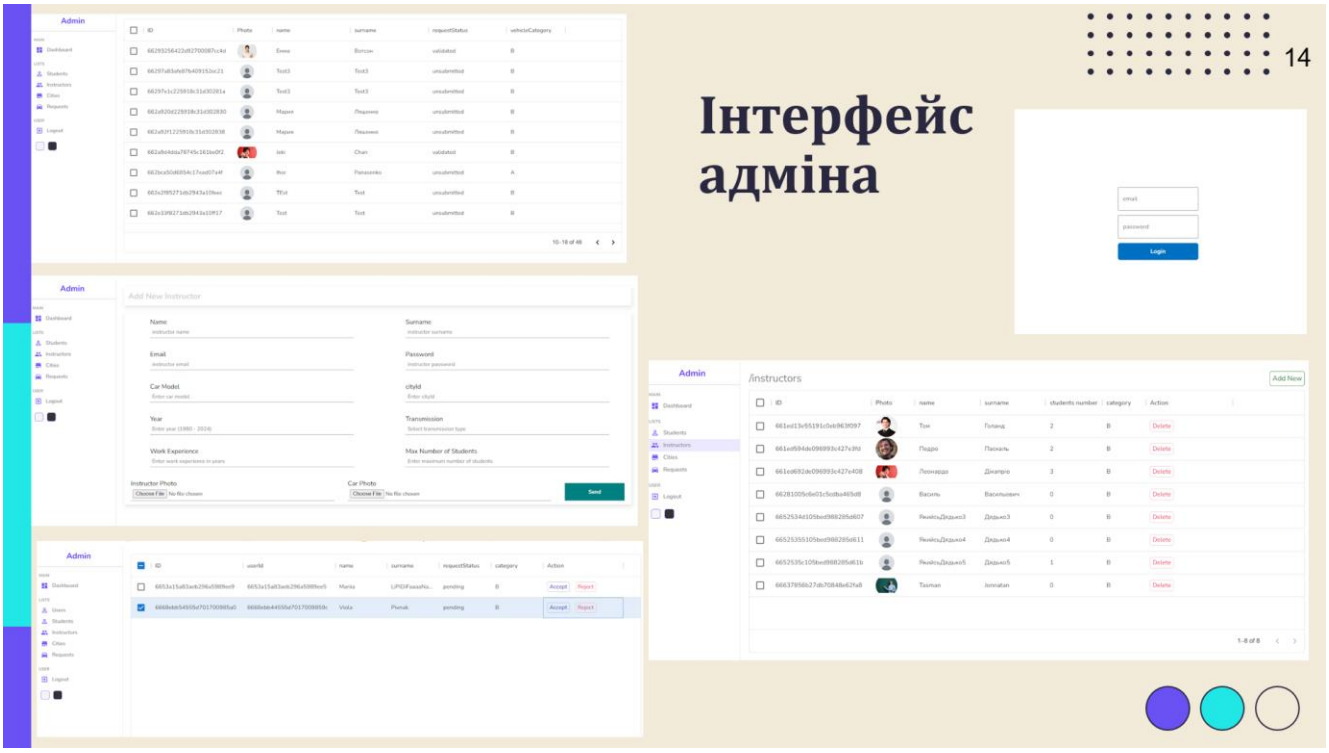
13

Інтерфейс



14

Інтерфейс адміна



15

Код реалізації контексту

```
import { createContext, useEffect, useReducer } from "react"

const userFromLocalStorage = localStorage.getItem("user")
const user = userFromLocalStorage ? JSON.parse(userFromLocalStorage) : null

const INITIAL_STATE = {
  user: JSON.parse(localStorage.getItem("user")) || null,
  loading: false,
  error: null
}
console.log(JSON.parse(localStorage.getItem("user")))

export const AuthContext = createContext(INITIAL_STATE)

const AuthReducer = (state, action) => {
  switch (action.type) {
    case "LOGIN_START":
      return {
        user: null,
        loading: true,
        error: null
      }
    case "LOGIN_SUCCESS":
      return {
        user: action.payload,
        loading: false,
        error: null
      }
    case "LOGIN_FAILURE":
      return {
        user: null,
        loading: false,
        error: action.payload
      }
    case "LOGOUT":
      return {
        user: null,
        loading: false,
        error: null
      }
    default:
      return state
  }
}
```

```
export const AuthContextProvider = ({ children }) => {
  const [state, dispatch] = useReducer(AuthReducer, INITIAL_STATE)

  useEffect(() => {
    localStorage.setItem("user", JSON.stringify(state.user))
  }, [state.user])

  return (      You, 2 months ago • add register and login …
    <AuthContext.Provider
      value={{
        user: state.user,
        loading: state.loading,
        error: state.error,
        dispatch
      }}
    >
      {children}
    </AuthContext.Provider>
  )
}
```

16

Код реалізації оплати

```
const processLiqPayPayment = (data, signature, successCb, errorCb, cb) => {
  ;(window.LiqPayCheckoutCallback = function() {
    LiqPayCheckout.init({
      data: data,
      signature: signature,
      embedTo: "#liqpay_checkout",
      language: "en",
      mode: "popup" // embed || popup
    })
      .on("liqpay.callback", function(data) {
        if (data.status === "success") successCb()
        else if (data.status === "error") errorCb()
        const requestData = {
          status: data.status,
          amount: data.amount,
          paytype: data.paytype,
          end_date: data.end_date,
          order_id: data.order_id
        }
        cb(requestData)
      })
      .on("liqpay.ready", function(data) {
        // ready
      })
      .on("liqpay.close", function(requestData) {
        // close
      })
  })()
}
export default PaymentButton
```

17

Тестування програмного забезпечення

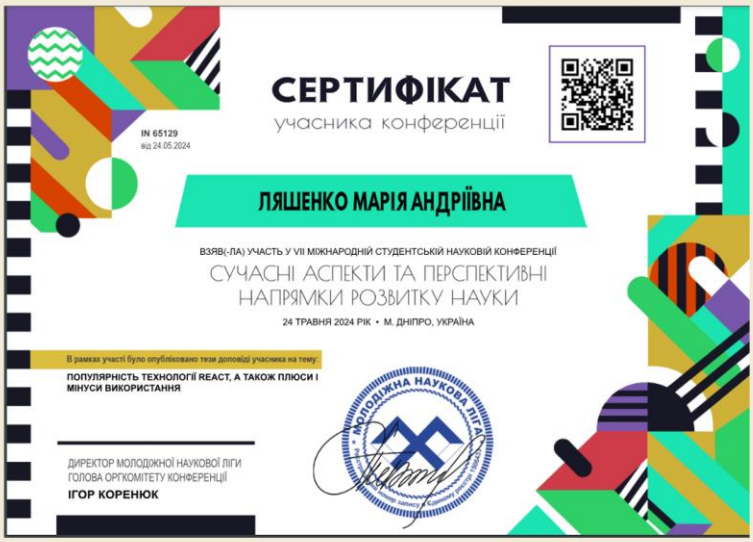
Регресійне тестування

Тестування реальними користувачами

Unit тестування

18

Апробація результатів

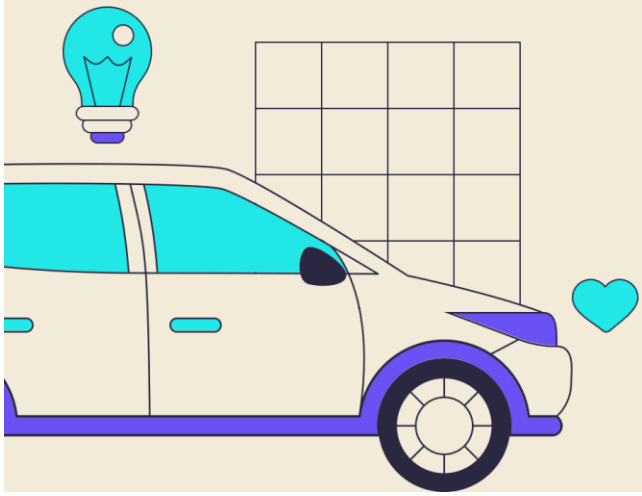


19

Висновки

Було розроблено фронтенд для автоматизації проведення практичних занять в автошколі. Під час проектування проаналізовано предметну галузь, сформульовано вимоги до системи, описано архітектуру та функціонал. Основний функціонал включає управління розкладом, запис на заняття, оплату, відстеження прогресу та відмічення вивчених вправ. Використано технології React для організації компонентів та Redux для керування станом додатку. Для стилізації застосовано SCSS, а для взаємодії з сервером - бібліотеку Axios. Фронтенд задовольняє потреби інструкторів та учнів автошколи, спрощуючи та оптимізуючи процес проведення практичних занять.

20



Дякую за увагу!

ДОДАТОК В

Тези доповіді на VII міжнародній студентській конференції “Сучасні аспекти та перспективні напрямки розвитку науки”

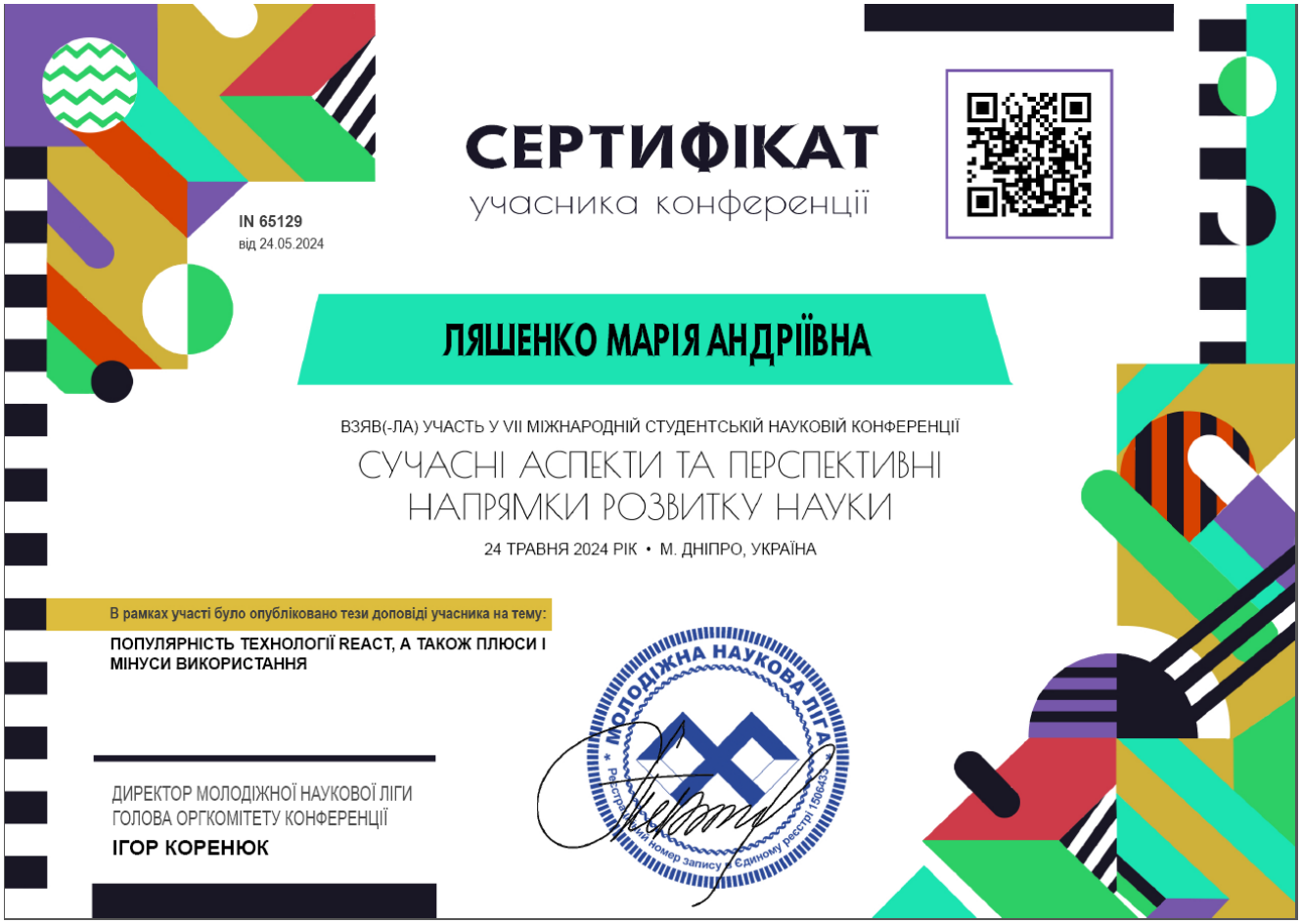


Рисунок В.1 – Сертифікат учасника конференції

ПОПУЛЯРНІСТЬ ТЕХНОЛОГІЇ REACT, А ТАКОЖ ПЛЮСИ І МІНУСИ ВИКОРИСТАННЯ

Ляшенко Марія Андріївна

здобувач вищої освіти факультету комп’ютерних наук

«Харківський національний університет радіоелектроніки», Україна

Науковий керівник: Олійник Олена Володимирівна

Старший викладач кафедри програмної інженерії

«Харківський національний університет радіоелектроніки», Україна

React js – це інтерфейсна бібліотека, яка стала популярним середовищем для сучасної веб розробки у спільноті програмування JavaScript. Він використовується для швидкого та ефективного створення інтерактивних користувацьких інтерфейсів і веб додатків із застосуванням значно меншої кількості коду, ніж під час використання звичайного JavaScript [1].

React став надзвичайно популярним серед розробників з кількох важливих причин.

По-перше, його простота у використанні. React має зрозумілий та логічний API, що дозволяє розробникам швидко зрозуміти, як працює бібліотека і почати створювати ефективні інтерфейси.

По-друге, компонентний підхід [2], який лежить в основі React дозволяє розробникам створювати незалежні та багаторазові компоненти. Це не тільки спрощує структуру коду, але й робить його більш модульним та легким для підтримки.

Ще однією важливою причиною є висока продуктивність React. Бібліотека використовує віртуальний DOM (Document Object Model) [2], що значно підвищує ефективність оновлення інтерфейсу. Замість того, щоб оновлювати весь реальний DOM при кожній зміні, React оновлює лише ті його частини, які були змінені. Це забезпечує швидку реакцію інтерфейсу та зменшує навантаження на браузер, що особливо важливо для великих та складних додатків.

В цілому, комбінація простоти використання, компонентного підходу, високої продуктивності, великої спільноти та підтримки від великої компанії зробили React одним з найпопулярніших інструментів для розробки інтерфейсів користувача. Тепер розглянемо переваги та недоліки цієї бібліотеки.

React зарекомендував себе як ключова бібліотека для створення динамічних і адаптивних веб-додатків з наступних причин:

1. Компонентна архітектура. Традиційні додатки на JavaScript часто стикаються з проблемами управління станом при масштабуванні. Проте React пропонує складні, незалежно підтримувані та повторно використовувані компоненти, що дозволяє розробникам оновлювати

частини веб-сторінки, не зачіпаючи інші, забезпечуючи слабку зв'язність та функціональність спільної роботи.

2. Перспективний вибір для розробників. Перспективність React – одна з найбільш переконливих переваг, які він пропонує розробникам. Гнучка архітектура React задовольняє поточні потреби веб-розробки, а також легко адаптується до нових технологій, що будуть визначати найближче майбутнє галузі [4]. Одним із найбільш яскравих прикладів є TensorFlow.js, бібліотека машинного навчання, що використовується для розпізнавання зображень та образів. Аналогічно, React дозволяє інтегрувати чат-ботів на базі машинного навчання та навіть функції рекомендацій. Крім того, WebAssembly може стати корисним помічником, дозволяючи додаткам машинного навчання, написаним на Rust, Python або C++ існувати в рідному додатку.

3. Redux для управління станом [3]. В односторінкових додатках, також відомих як SPA, де декілька компонентів розташовані на одній сторінці, управління станом і міжкомпонентним зв'язком може швидко стати складним завданням – саме в цьому Redux для React і проявляється. Будучи невід'ємною частиною React, він служить «менеджером» для забезпечення узгодженого та точного потоку даних між компонентами, централізуючи управління станом та сприяючи незалежності компонентів, що значно покращує стабільність даних та зручність використання.

Хоча React надає багато переваг розробникам різного рівня кваліфікації, він не позбавлений відповідних недоліків, серед яких:

1. Складні концепції та просунуті шаблони: React представляє кілька просунутих концепцій та шаблонів, які спочатку можуть бути складними для новачків. Розуміння JSX, компонентів, властивостей, управління станом, методів життєвого циклу та перехоплювачів вимагає чіткого розуміння основ JavaScript.

2. Складність інтеграції з іншими технологіями [4]. React часто використовується у поєднанні з іншими інструментами та технологіями,

такими як Redux, React Router і різним проміжним програмним забезпеченням, і розуміння того, як інтегрувати ці технології з React, може виявитися складним завданням для новачків.

3. Бар'єр для розробників, що не використовують JavaScript: Сильна залежність React від JavaScript може стати бар'єром для розробників, не знайомих із JavaScript [2]. Хоча JavaScript є універсальною та широко використовуваною мовою, розробникам з іншим досвідом програмування може бути складно адаптуватися до парадигм JavaScript та способів його використання в React.

4. Не повноцінна платформа [4]. React в першу чергу обробляє «представлення» частини MVC, також відомої як архітектура контролера представлення моделі. Для «Моделі» та «Контролера» потрібні додаткові бібліотеки, що в кінцевому підсумку може призвести до менш структурованого та потенційно більш хаотичного коду порівняно з повнофункціональними фреймворками, такими як Angular.

5. Роздування коду. React.js, характеризуючись високими вимогами до бібліотек і залежностей, відомий своїми роздутими додатками. Це роздування часто проявляється у збільшенні часу завантаження, особливо у складних проектах [4]. Структура платформи, значною мірою заснована на віртуальному DOM, вимагає завантаження цілих бібліотек навіть для незначних функцій, що значно збільшує цифровий слід додатку та знижує його ефективність.

В кінцевому підсумку, незважаючи на деякі проблеми, React залишається ключовою бібліотекою в розробці веб-сайтів та мобільних додатків завдяки його гнучкості та надійній екосистемі. Він продовжить служити надійним інструментом для розробників, дозволяючи їм без проблем впроваджувати передові функції у свої додатки. Однак варто відзначити, що складність React та його залежність від передових концепцій JavaScript можуть потребувати значного часу на навчання, особливо для початківців або тих, хто ще не дуже добре розбирається в JavaScript. Також важливо враховувати, що React в першу чергу

стосується аспекту "Вигляд" архітектури MVC, що потребує додаткових інструментів для повної розробки додатків, що може призвести до більш складних та менш структурованих кодових баз.

Незважаючи на ці проблеми, активне та обширне співтовариство React відіграє важливу роль у його розвитку, що робить його надійним вибором для розробників у майбутньому.

Список використаних джерел:

1. Projector Creative & Tech Institute. React: Що таке React? Як почати вивчати Реакт? Основні навички. *CASES*. URL: https://cases.media/en/article/sho-take-react-js-yak-pochati-vivchati-reakt-navichki-dlya-react-developer (дата звернення: 18.05.2024).

2. Ibrahim M. S. S. ReactJS–Advantages & Disadvantages. *Medium*. URL: https://medium.com/@muhamed.salih/reactjs-advantages-disadvantages-16f479b3aa47 (дата звернення: 18.05.2024).

3. Why Use React Redux? | React Redux. *React Redux | React Redux*. URL: https://react-redux.js.org/introduction/why-use-react-redux (дата звернення: 18.05.2024).

4. Pros and Cons of Using React | Advantages & Disadvantages - DS. *Devstringx Technologies*. URL: https://www.devstringx.com/pros-and-cons-of-using-react (дата звернення: 18.05.2024).