МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ Андрій ФОРСЮК	_____ Сергій ГРИБКОВ
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«02» ____червня____ 2025р.	«02» ____червня_____ 2025р.

КВАЛІФІКАЦІЙНА РОБОТАХ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)

освітньо-професійної програми Комп'ютерні науки

на тему : «Розроблення веб-орієнтованої системи для моніторингу технічного стану автомобілів "AutoCheck Solutions"»

Виконав: здобувач 4 курсу, групи КН-4-3

Мохонько Ілля Костянтинович _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Мошенський Андрій Олександрович _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти _____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент _____ _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач _____
(підпис)

Київ - 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки

Освітній ступінь Бакалавр
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій, штучного інтелекту і кібербезпеки _____Сергій ГРИБКОВ
«28» _____квітня_____ 2025 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Мохонька Іллі Костянтиновича
(прізвище, ім'я, по батькові)

1. Тема роботи «Розроблення веб-орієнтованої системи для моніторингу технічного стану автомобілів "AutoCheck Solutions"»,
керівник роботи Мошенський Андрій Олександрович, доцент, к. т. н.
затверджені наказом закладу вищої освіти від «28» квітня 2025 р. № 254-кс
2. Строк подання здобувачем роботи: 30.05.2025 р.
3. Вихідні дані до роботи: загальні відомості про ринок інформаційних систем для моніторингу технічного стану автомобілів.
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):

1. Аналіз ринку предметної області, 2. Опис проблем, 3. Пошук і аналіз типових систем, 4. Переваги та недоліки систем, 5. Аналіз проблем типових систем, 6. Обґрунтування доцільності проектування та розробки системи, 7. Постановка задачі, 8. Розрахунок економічного ефекту, 9. Технічне завдання, 10. Опис та обґрунтування вибору програмно-технічних засобів, 11. Проєктування та створення бази даних, 12. Реалізація функцій системи, 13. Інструкція користувача, 14. Тестування програмного продукту

5. Перелік графічного матеріалу:
Приклади інтерфейсу типових систем, графік окупності проєкту, діаграма Ганта, скріншоти інтерфейсу системи, загальна схема архітектури, UML діаграма використання системи.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	доц. Мошенський А.О.	28.04.25	10.05.25
2	доц. Мошенський А.О.	28.04.25	17.05.25
3	доц. Мошенський А.О.	28.04.25	19.05.25

7. Дата видачі завдання: 28 квітня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження об’єкта автоматизації	01.05.2025	Виконано
2	Розробка технічного завдання	05.05.2025	Виконано
3	Проєктування БД	06.05.2025	Виконано
4	Реалізація програмного забезпечення	08.05.2025	Виконано
5	Тестування системи	10.05.2025	Виконано
6	Підготовка документації	12.05.2025	Виконано
7	Встановлення та налаштування системи	15.05.2025	Виконано
8	Оформлення документації	18.05.2025	Виконано

Здобувач _____ Ілля МОХОНЬКО
(підпис) (прізвище та ініціали)

Керівник роботи _____ Андрій МОШЕНСЬКИЙ
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Головною метою цієї кваліфікаційної роботи є створення вебзастосунку, призначеного для моніторингу технічного стану транспортних засобів. Розроблена система AutoCheck Solutions покликана забезпечити користувачам можливість вести облік автомобілів, зберігати історію технічного обслуговування, додавати події, пов’язані з ремонтом чи обслуговуванням, а також здійснювати аналітику витрат у зручному візуальному форматі. Для адміністратора чи власника автопарку система дозволяє централізовано керувати всіма записами, контролювати статус подій, переглядати звіти у форматі PDF і отримувати аналітичні діаграми.

У межах реалізації проєкту було впроваджено функціонал реєстрації, авторизації, додавання автомобілів із фотографіями, інтерактивного календаря з подіями, фільтрації записів, редагування даних, автоматичного визначення статусу подій за датою, а також функцію експорту звіту по обслуговуванню. Програмна реалізація базується на технологіях Node.js[17] + Express для серверної частини, MongoDB[18] як системи зберігання даних, React[16] — для створення інтерфейсу користувача, а також використано бібліотеки для побудови діаграм (Recharts), генерації PDF-документів та валідації форм.

Кваліфікаційна робота містить 84 сторінок, 5 таблиць, 38 рисунків і 30 використаних джерел.

КЛЮЧОВІ СЛОВА: МОНІТОРИНГ, АВТОМОБІЛЬ, ВЕБОРІЄНТОВАНА ІНФОРМАЦІЙНА СИСТЕМА, ТЕХНІЧНЕ ОБСЛУГОВУВАННЯ, КАЛЕНДАР, ДІАГРАМА, БАЗА ДАНИХ, ЗВІТ, ВЕБЗАСТОСУНОК.

SUMMARY

The main objective of this qualification thesis is the development of a web-based application designed for monitoring the technical condition of vehicles. The developed system, AutoCheck Solutions, aims to provide users with the ability to manage vehicle records, store maintenance history, add service-related events, and visualize spending analytics in a user-friendly format. For fleet owners or administrators, the system offers centralized control over all records, monitoring of event statuses, report generation in PDF format, and access to visual analytics.

The project includes functionality for user registration and authentication, adding vehicles with photos, an interactive event calendar, record filtering, data editing, automatic event status detection based on the current date, and the ability to export service reports. The software is built using Node.js[17] + Express for the backend, MongoDB[18] for data storage, and React[16] for the frontend interface. Additional libraries were used for chart generation (Recharts), PDF creation, and form validation.

This qualification work contains 84 pages, 5 tables, 38 figures, and references to 30 sources.

KEYWORDS: MONITORING, VEHICLE, WEB-ORIENTED INFORMATION SYSTEM, MAINTENANCE, CALENDAR, CHART, DATABASE, REPORT, WEB APPLICATION.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку цифрових технологій усе більше галузей потребують ефективних засобів автоматизації й аналітики. Однією з таких сфер є технічне обслуговування автомобілів, де своєчасне виявлення проблем, збереження історії обслуговування та контроль витрат відіграють ключову роль у забезпеченні надійності транспорту.

Потреба в централізованому управлінні інформацією про транспортні засоби зумовлює актуальність розробки веб-орієнтованих інформаційних систем. Такі системи повинні забезпечувати зручний і швидкий доступ до даних про стан авто, дозволяти вести облік обслуговувань, створювати календар подій і генерувати аналітичні звіти.

Метою цієї кваліфікаційної роботи є створення вебсистеми AutoCheck Solutions, яка дозволяє користувачам додавати автомобілі з фото, реєструвати події технічного обслуговування, автоматично визначати статуси виконання робіт, фільтрувати записи та переглядати візуалізовану статистику. Для підприємств або автопарків це рішення забезпечує прозорий облік технічного стану транспорту й дозволяє швидко отримувати зведені звіти.

У ході розробки системи було виконано повний цикл програмної інженерії: від аналізу предметної області, постановки задачі, вибору технологій і проєктування архітектури — до реалізації, тестування та підготовки інструкції для користувача. Система побудована з використанням Node.js, React, MongoDB і низки допоміжних бібліотек для форм, діаграм і PDF-звітів.

Результатом є універсальне вебзастосування, яке може бути використане як приватними автовласниками, так і організаціями для ефективного моніторингу технічного стану автомобілів.

РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ ЗАДАЧІ НА РОЗОРОЗБУ ВЕБОРІЄНТОВАНОЇ ПЛАТФОРМИ

1.1 Аналіз сучасного ринку досліджуваної предметної області

У сучасних умовах цифрової трансформації різні галузі економіки активно впроваджують інформаційні технології для оптимізації внутрішніх процесів і підвищення якості управління. Особливого значення ці зміни набули у сфері транспортного забезпечення, де контроль технічного стану автомобілів є критичним фактором безпеки, надійності та ефективності експлуатації.

Автоматизація обліку та планування технічного обслуговування автотранспорту дозволяє не лише уникнути помилок, пов’язаних із людським фактором, а й забезпечити прозорість ведення всієї історії робіт. Зокрема, впровадження веб-орієнтованих систем створює можливість централізованого зберігання та обробки даних, доступу до них із будь-якого пристрою, формування статистичних звітів і аналітики в реальному часі.

Нині все більше організацій прагнуть позбутись застарілих підходів до ведення документації та переходять від локальних програм або паперових журналів до інтегрованих рішень, які дозволяють автоматизувати не лише облік автомобілів, а й контроль витрат, планування ремонту, нагадування про ключові дати, а також формування звітів для аналізу ефективності експлуатації техніки.

Однак, попри наявність готових рішень на ринку, чимало з них мають обмежений функціонал або не враховують специфіку малого й середнього бізнесу. У результаті підприємства змушені користуватись кількома окремими сервісами, що призводить до дублювання даних, затримок в обробці інформації та неузгодженості між відділами.

Розробка єдиної веб-системи для моніторингу технічного стану автопарку, яка об’єднує облік авто, події обслуговування, візуалізацію витрат і календарні нагадування, є актуальним і доцільним рішенням для

підприємств, які прагнуть зменшити витрати, покращити керованість процесами й підвищити загальний рівень безпеки транспортних операцій.

Таким чином, створення інноваційної системи AutoCheck Solutions є відповіддю на виклики сучасного ринку та дозволяє забезпечити прозорий, ефективний і технологічно гнучкий підхід до управління технічним станом автомобілів.

1.2 Опис проблем

У сфері обслуговування та технічного контролю автотранспорту залишається низка суттєвих труднощів, які негативно впливають на ефективність управління автопарком. Однією з основних проблем є відсутність універсального цифрового рішення, що дозволяє систематизовано збирати, зберігати й аналізувати інформацію про стан транспортних засобів.

Часто власники авто не мають централізованого доступу до сервісної історії: записи про ремонти або технічні огляди зберігаються в паперовому вигляді, у різних документах або взагалі втрачаються. Це ускладнює перевірку фактичного стану автомобіля, особливо при передачі в інше користування, продажу або під час гарантійного обслуговування.

Окрему проблему становить відсутність зручного способу збереження та доступу до супровідної документації — чеків, актів виконаних робіт, гарантійних талонів. За відсутності структурованого електронного архіву користувачі часто стикаються з ситуаціями, коли відновити інформацію стає неможливо.

Ще одним слабким місцем є контроль за термінами виконання обов’язкових технічних робіт. Через відсутність нагадувань або календарного планування багато власників виконують обслуговування із запізненням, що може призвести до несправностей, зниження безпеки руху та збільшення витрат на усунення поломок.

Не менш важливою проблемою є відсутність зведеної статистики по витратах на утримання транспортного засобу. Неможливість аналізу динаміки

витрат або прогнозування бюджету технічного обслуговування ускладнює фінансове планування, особливо для компаній, які обслуговують великий автопарк.

Усі ці чинники вказують на необхідність створення інтегрованої системи, яка б забезпечила єдиний підхід до зберігання сервісних записів, автоматизувала контроль за технічними роботами, дозволяла формувати звіти та аналізувати витрати в реальному часі. Саме така мета покладена в основу розробки системи AutoCheck Solutions — сучасного веб-застосунку, покликаного вирішити актуальні проблеми, з якими щодня стикаються власники авто та підприємства, що експлуатують транспорт.

1.3 Пошук та аналіз подібних систем

Підприємства, які здійснюють експлуатацію або технічне обслуговування автотранспортних засобів, дедалі частіше стикаються з потребою впровадження програмних рішень, що забезпечують контроль за технічним станом автомобілів, облік сервісних операцій та організацію планових робіт. З метою вирішення цих завдань на ринку представлено чимало програмних продуктів — від простих електронних журналів до комплексних систем управління автопарком.

Серед поширених рішень можна виділити такі системи, як Fleetio, Chevin FleetWave та Simply Auto. Ці платформи пропонують базовий або розширений функціонал для ведення обліку транспортних засобів, збереження історії ТО, формування нагадувань, зберігання документів і побудови аналітичних звітів. Частина з них підтримує інтеграцію з GPS-трекерами, сторонніми CRM-системами чи мобільними додатками.

Однак попри наявність широких можливостей, багато із згаданих систем мають певні обмеження. Деякі орієнтовані лише на великі автопарки і мають складний інтерфейс для звичайного користувача. Інші — не підтримують українську мову або не мають адаптації до локальних вимог і стандартів. Також зустрічаються платформи, де зберігання документації

реалізовано лише у вигляді прикріплених файлів без можливості їхньої категоризації чи швидкого пошуку.

Окремою проблемою є недостатній рівень автоматизації. Не всі продукти підтримують формування графіка обслуговування на основі пробігу або часу, а також не всі надають гнучкі інструменти для прогнозування витрат на ремонт. Для малого та середнього бізнесу бар'єром може стати ціна використання або складність налаштування таких систем.

Аналіз існуючих рішень дає змогу чітко окреслити ті функціональні характеристики, які має містити сучасний вебзастосунок для моніторингу технічного стану автомобілів. Зокрема, система повинна поєднувати зручність для користувача, підтримку української мови, можливість централізованого зберігання сервісної інформації, інтеграцію з календарем подій, формування звітів і автоматичні нагадування про планові роботи.

Саме ці аспекти були враховані при формуванні вимог до майбутньої системи AutoCheck Solutions, яка має на меті створити максимально адаптований до українських реалій, доступний та ефективний інструмент для контролю технічного стану транспортних засобів.

1.3.1 Аналіз системи управління «Chevin FleetWave»

Chevin FleetWave – це потужна веб-орієнтована система управління автопарком, яка допомагає компаніям контролювати всі аспекти експлуатації транспортних засобів (Рис. 1.1.). Програма використовується в різних галузях, включаючи логістику, державний сектор, будівництво та енергетику.

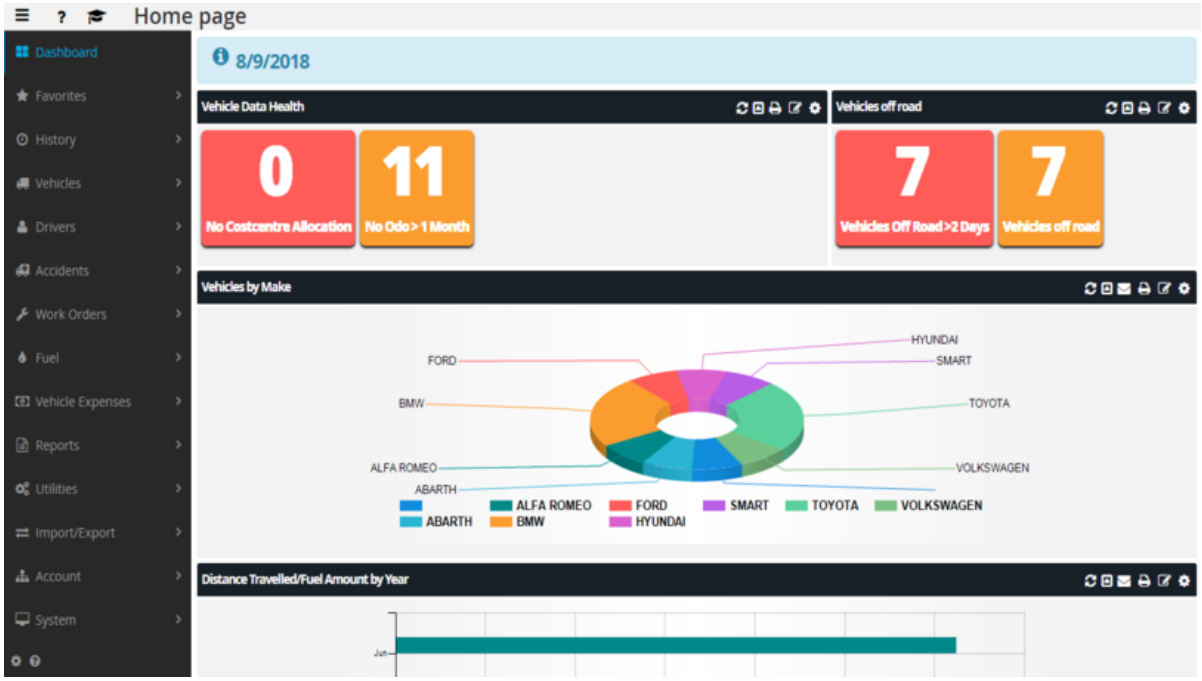

Рисунок 1.1 – Інтерфейс «Chevin FleetWave»

Chevin FleetWave — це одна з найбільш функціональних веб-орієнтованих систем для управління автопарком, яка орієнтована на середні та великі підприємства, що прагнуть комплексного контролю за технічним станом транспортних засобів.

Основні особливості системи:

- **Централізований облік і контроль:** система надає можливість вести повний електронний реєстр транспортних одиниць підприємства, де зберігається історія експлуатації, дані про пробіг, технічне обслуговування та витрати. Це дозволяє уникнути розрізненого зберігання інформації та пришвидшити доступ до неї.
- **Автоматизація сервісних процедур:** FleetWave дозволяє формувати графіки обслуговування, надсилати автоматичні нагадування про наближення ТО чи перевірки, а також реєструвати виконані ремонтні роботи із зазначенням дати, вартості та категорії послуги.
- **Гнучка інтеграція з іншими платформами:** розробники передбачили можливість підключення системи до зовнішніх сервісів за допомогою API. Зокрема, доступна синхронізація з бухгалтерськими обліковими

системами, модулями GPS-моніторингу, обліку пального та складських програм.

- **Функціонал звітності:** програмне забезпечення містить розвинений інструментарій для створення звітів. Користувач може отримати аналітику за витратами, ремонтами, споживанням пального, інтервалами між обслуговуваннями та іншими показниками.

Недоліки, виявлені під час огляду:

- інтерфейс системи може виявитися складним для нових користувачів без попереднього навчання;
- вартість ліцензії та підтримки — вище середньої, що робить її малодоступною для невеликих компаній;
- при обробці великих масивів даних трапляється зниження швидкодії;

FleetWave є надійним інструментом для управління великим автопарком, зосередженим на максимальній деталізації та контролі. Проте для невеликих компаній або користувачів без досвіду роботи з корпоративними системами її використання може вимагати адаптації або пошуку альтернатив з простішим функціоналом.

1.3.2 Аналіз системи управління «Fleetio»

Fleetio – це сучасна хмарна система для управління автопарком, яка допомагає компаніям контролювати технічний стан транспортних засобів, оптимізувати витрати та автоматизувати робочі процеси (Рис. 1.2.). Система орієнтована на підприємства різного масштабу, від малого бізнесу до великих корпорацій.

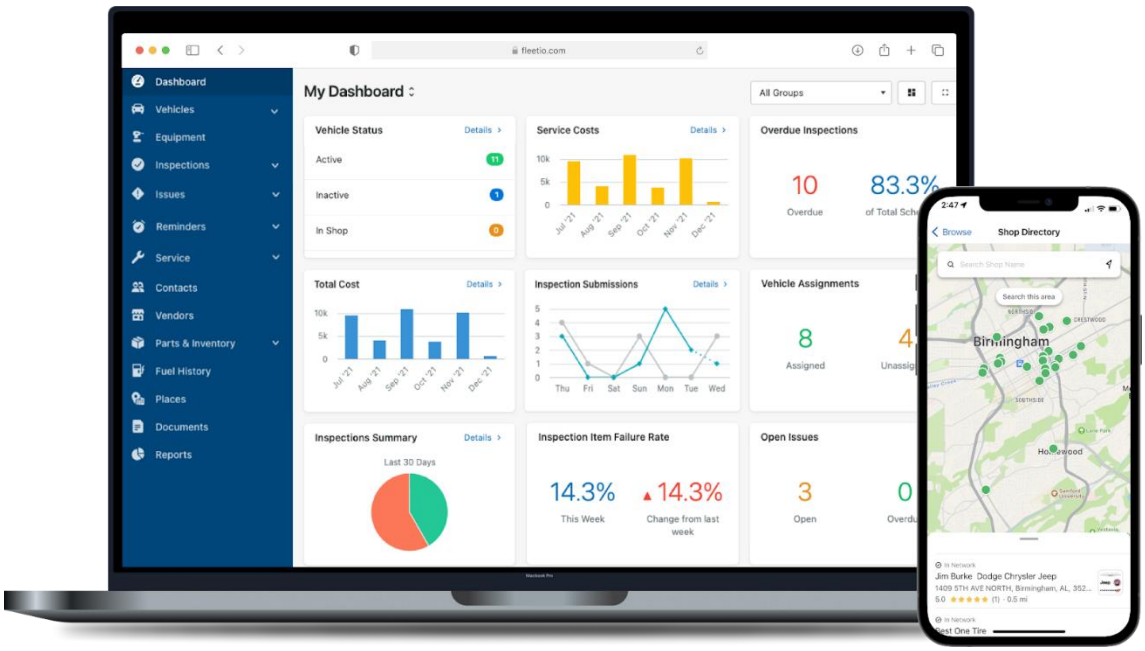

Рисунок 1.2 – Інтерфейс «Fleetio»

Основні особливості системи:

- Управління технічним обслуговуванням: Fleetio автоматично нагадує про заплановані перевірки, реєструє виконані роботи та дозволяє відстежувати історію обслуговування кожного транспортного засобу.
- Моніторинг витрат і пального: програма надає детальні звіти про витрати на паливо, ремонт і експлуатацію, що дозволяє контролювати бюджет автопарку.
- Мобільний додаток та інтеграція: Fleetio має зручний мобільний додаток для Android та iOS, що дозволяє механікам і водіям миттєво вводити дані. Також підтримує інтеграцію з GPS-трекерами, системами керування запасами та бухгалтерським ПЗ.
- Керування документами: система дозволяє зберігати всі важливі документи (страховки, ліцензії, гарантії, рахунки за ремонт) в електронному вигляді, усуваючи потребу в паперовому архівуванні.
- Персоналізовані звіти та аналітика: користувачі можуть створювати кастомні звіти для аналізу ефективності роботи автопарку, виявлення тенденцій витрат та прогнозування майбутніх потреб.

Можливі недоліки:

- вартість підписки може бути високою для малих компаній;
- деякі користувачі відзначають, що початкове налаштування системи займає певний час;
- для деяких функцій потрібна інтеграція з додатковими сервісами, що може бути незручним;

Загалом, Fleetio – це потужна та гнучка система, яка підійде як для малих, так і для великих компаній, які прагнуть отримати повний контроль над своїм автопарком у зручному онлайн-форматі.

1.3.3 Аналіз системи управління «Fleetio»

Simply Auto – це зручний мобільний застосунок для обліку витрат на транспортні засоби, що підходить як для індивідуальних користувачів, так і для малого бізнесу (Рис. 1.3.). Його основна мета – допомогти водіям і власникам автопарків вести детальний журнал витрат, технічного обслуговування та заправок.

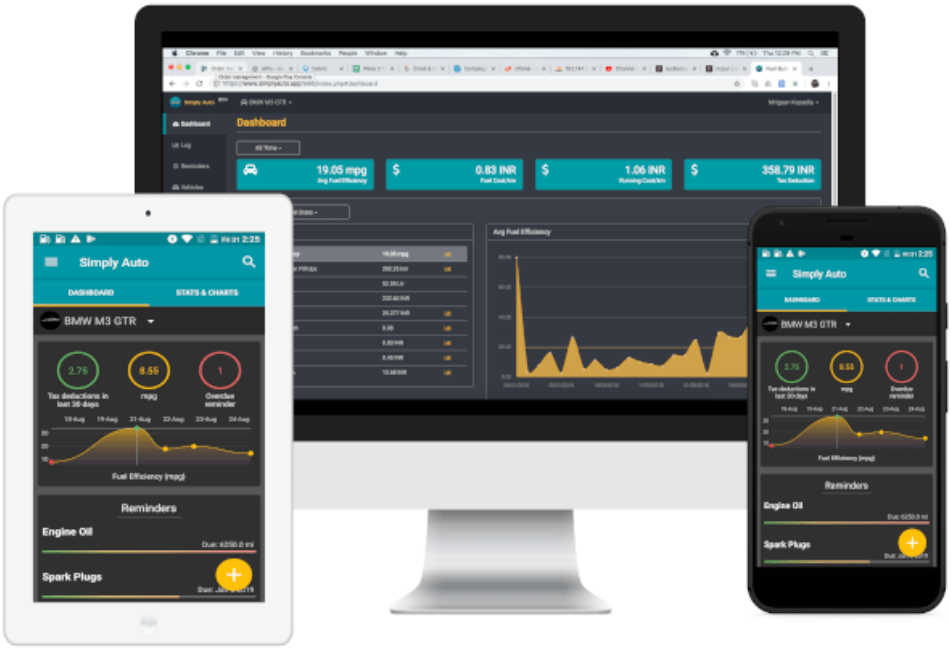

Рисунок 1.3 – Інтерфейс «Simply Auto»

Основні особливості системи:

- Облік витрат і пального: Simply Auto дозволяє вести детальну історію витрат на паливо, ремонт, страхування та інші потреби. Користувач може аналізувати витрати та ефективність використання пального через графіки та звіти.
- Журнал технічного обслуговування: програма надає можливість записувати проведені ремонтні роботи, заміну деталей, перевірки та ТО, а також встановлювати нагадування про наступне обслуговування.
- Сканування чеків і документів: Simply Auto підтримує функцію збереження чеків та інших документів у цифровому вигляді, що полегшує контроль витрат.
- Імпорт та експорт даних: користувачі можуть експортувати дані в Excel або PDF для зручного аналізу та збереження, а також імпортувати інформацію з інших.
- Мобільний додаток та хмарна синхронізація: застосунок працює на iOS та Android, а дані можна синхронізувати між різними пристроями, що зручно для тих, хто використовує кілька гаджетів.

Можливі недоліки:

- функціонал орієнтований більше на особисте користування або невеликі автопарки, що може бути обмеженням для великих підприємств;
- безкоштовна версія має обмежені можливості, а для доступу до розширених функцій потрібна підписка;
- інтерфейс менш гнучкий у порівнянні з більш професійними рішеннями для керування автопарком;

Загалом, Simply Auto – це чудове рішення для особистого використання або невеликих автопарків, які хочуть вести зручний цифровий облік витрат, технічного обслуговування та інших важливих аспектів експлуатації автомобілів.

1.3.4 Порівняльна характеристика типових систем управління

Аналіз систем управління автопарком дозволяє оцінити їхні можливості, сильні та слабкі сторони, а також визначити, яке рішення найкраще підходить для конкретних потреб підприємства або індивідуальних користувачів. У цьому розділі розглянемо три популярні системи: Chevin FleetWave, Fleetio та Simply Auto. Ми звернемо увагу на основні характеристики, такі як функціональність, зручність використання, автоматизація процесів, інтеграція з іншими платформами та доступність для користувачів.

Для детального аналізу представимо порівняльну таблицю 1.1, у якій порівняємо ключові можливості кожної з цих систем.

Таблиця 1.1. Порівняння типових систем

Характеристика	**Chevin**	**Fleetio**	**Simply Auto**
Тип системи	Хмарна система управління автопарком для великих підприємств	Хмарна система управління автопарком для бізнесу будь-якого масштабу	Додаток для індивідуального та малого бізнес-обліку автопарку
Основне призначення	Контроль і управління великим автопарком, оптимізація витрат	Автоматизація управління транспортом, аналіз ефективності	Облік витрат на транспорт, ведення історії технічного обслуговування
Ключові можливості	Моніторинг транспорту, управління витратами, ТО, звітність, аналітика	Відстеження автопарку, ТО, витрати, документообіг, API-інтеграції	Запис витрат, планування ТО, розрахунок витрати пального
Мобільна версія	Так, адаптований мобільний інтерфейс	Так, мобільний додаток для iOS та Android	Так, мобільний додаток з хмарною синхронізацією

Продовження Таблиці 1.1.

Характеристика	Chevin	Fleetio	Simply Auto
Інтерфейс	Простий, орієнтований на особисте використання	Інтуїтивний, простий у використанні	Простий, орієнтований на особисте використання
Цільова аудиторія	Великі компанії, державні установи, логістичні підприємства	Малі, середні та великі компанії	Індивідуальні користувачі, невеликі автопарки
Підтримка інтеграцій	API для підключення до інших бізнес-систем	Широкий вибір інтеграцій (GPS, бухгалтерія, телематика)	Обмежена, можливість експорту даних
Моніторинг транспорту	Так, детальна аналітика, GPS-відстеження	Так, інтеграція з GPS-трекерами	Ні, тільки ручний запис пробігу
Автоматизація процесів	Високий рівень автоматизації, планування обслуговування	Автоматичні нагадування, цифровий документообіг	Базові нагадування про ТО та витрати
Звіти та аналітика	Детальні звіти, прогнозування витрат	Гнучка система звітності, інтеграція з BI	Базові звіти про витрати на паливо та ТО
Методи оплати	Корпоративні ліцензії, підписка	Підписка, корпоративні тарифи	Безкоштовна версія з обмеженим функціоналом
Лояльність та бонуси	Індивідуальні умови для великих клієнтів	Гнучкі тарифи для різного масштабу бізнесу	Безкоштовна версія з базовим функціоналом
Додаткові послуги	Інтеграція з бухгалтерією, телематика, GPS	Підключення до сторонніх сервісів, API	Імпорт/експорт даних, збереження чеків

Продовження Таблиці 1.1.

Характеристика	Chevin	Fleetio	Simply Auto
Адаптація під пристрої	Веб-версія, мобільний додаток	Веб-версія, мобільний додаток	Мобільний додаток, синхронізація з хмарою

Це порівняння дозволяє оцінити, яка система найкраще відповідає вимогам бізнесу або індивідуального користувача. Вибір оптимального рішення залежить від масштабу автопарку, необхідного рівня автоматизації, інтеграцій з іншими сервісами та зручності використання.

1.4 Переваги та недоліки систем

У цьому розділі буде детально розглянуто переваги та недоліки кожної з типових систем, що ми розглядали в минулому підрозділі, щоб зробити певні висновки.

1.4.1 Переваги та недоліки системи «Chevin FleetWave»

Переваги:

- Гнучкість і масштабованість – система підходить для великих автопарків і підприємств, які керують великим обсягом даних.
- Широкий функціонал – включає управління технічним обслуговуванням, відстеження витрат, телеметрію та аналіз ефективності автопарку.
- Хмарне зберігання – всі дані зберігаються в хмарі, що дозволяє отримувати доступ до інформації з будь-якого пристрою.
- Можливість інтеграції – підтримує інтеграцію з GPS-трекерами, бухгалтерськими програмами та іншими бізнес-системами.
- Високий рівень аналітики – система генерує детальні звіти щодо витрат, технічного стану транспорту та інших показників.

Недоліки:

- Складний інтерфейс – потребує певного навчання для ефективного використання.

- Висока вартість – система орієнтована на великі компанії, що може бути дорогим варіантом для малого бізнесу.
- Обмежена мобільна версія – функціонал мобільного додатку не такий повний, як у веб-версії.

1.4.2 Переваги та недоліки системи «Fleetio»

Переваги:

- Інтуїтивно зрозумілий інтерфейс – проста у використанні платформа з сучасним дизайном.
- Автоматизація процесів – нагадування про ТО, ведення історії ремонтів, контроль витрат на паливо.
- Зручний мобільний додаток – підтримує сканування VIN-кодів, що спрощує облік транспорту.
- Інтеграція з GPS і телематикою – дозволяє отримувати дані в реальному часі про місцезнаходження і стан ТЗ.
- Гнучка система звітності – можна створювати індивідуальні звіти та аналізувати витрати.

Недоліки:

- Дорожча за Simply Auto – не завжди виправдана для невеликих компаній або приватних осіб.
- Потребує налаштування – для отримання максимального функціоналу необхідно налаштовувати інтеграції та параметри.
- Обмеження безкоштовної версії – базові можливості доступні безкоштовно, але розширені функції вимагають передплати.

1.4.3 Переваги та недоліки системи «Simply Auto»

Переваги:

- доступна ціна – має безкоштовну версію з основними функціями та недорогі платні плани;
- зручний для індивідуальних користувачів – підходить для особистого автопарку або невеликих підприємств;
- простий мобільний додаток – дозволяє швидко вводити дані про витрати, заправки, ТО та пробіг;

- синхронізація між пристроями – можна вести облік одночасно з телефону та ПК;
- автоматичне резервне копіювання – запобігає втраті даних;

Недоліки:

- обмежений функціонал для великих компаній – не підходить для масштабного управління автопарком;
- менше інтеграцій – у порівнянні з Fleetio та Chevin FleetWave, підтримує менше сторонніх сервісів;
- може не мати необхідних аналітичних інструментів – менш потужні звіти порівняно з конкурентами;

1.4.4 Загальний висновок аналізу переваг та недоліків

Аналіз систем управління автопарком, таких як Chevin FleetWave[6], Fleetio[5] та Simply Auto[4], є ключовим етапом у виборі ефективного рішення для контролю технічного стану та експлуатації транспортних засобів. Кожна з цих платформ має власні функціональні можливості, що впливають на зручність використання, інтеграцію з іншими сервісами та автоматизацію процесів. Проте, у кожної з них також є певні недоліки, які можуть обмежувати їхню ефективність у реальних умовах роботи.

Досліджуючи сильні та слабкі сторони цих систем, можна визначити ключові критерії, які необхідно враховувати при виборі або розробці оптимального рішення. Зручний інтерфейс, широкі можливості автоматизації, інтеграція з іншими програмами, детальна аналітика та мобільна доступність – це ті аспекти, які суттєво впливають на ефективність управління автопарком.

Врахування цих факторів дозволить підібрати або створити систему, що максимально відповідатиме потребам користувачів, покращить контроль за технічним станом транспорту та сприятиме зниженню експлуатаційних витрат. Оптимізація існуючих можливостей і усунення виявлених недоліків допоможуть зробити управління автопарком більш ефективним, сучасним і зручним.

1.5 Аналіз проблем типових систем

Аналіз популярних систем управління автопарком, таких як Chevin FleetWave, Fleetio та Simply Auto, дозволяє виявити низку проблем, що зустрічаються в багатьох подібних платформах. Розуміння цих недоліків є важливим для створення більш ефективного рішення, яке відповідатиме сучасним вимогам підприємств. Основні проблеми типових систем включають:

1. Обмежена інтеграція з іншими сервісами: багато платформ не забезпечують достатньо гнучкої інтеграції з іншими корпоративними системами, такими як бухгалтерські програми, GPS-трекери чи системи управління замовленнями. Це може ускладнювати автоматизацію процесів та зменшувати ефективність роботи автопарку.

2. Недостатня персоналізація: деякі системи не враховують специфічні потреби компаній, що працюють у різних галузях. Наприклад, відсутність можливості налаштовувати звіти, створювати кастомні шаблони документів або додавати унікальні параметри для моніторингу технічного стану транспорту може бути обмеженням для користувачів.

3. Складний інтерфейс: не всі платформи мають інтуїтивно зрозумілий дизайн. В деяких випадках система може бути перевантажена функціями, що ускладнює її освоєння та використання, особливо для нових користувачів або компаній, які не мають досвіду роботи з подібним програмним забезпеченням.

4. Проблеми з мобільною версією: деякі системи не мають повноцінної мобільної версії або ж вона має обмежений функціонал. Це створює незручності для користувачів, які потребують доступу до управління автопарком у режимі реального часу через смартфони чи планшети.

5. Недостатньо гнучкі налаштування обліку витрат: не всі системи пропонують зручні механізми для контролю витрат, зокрема на паливо, ремонт та страхування. Відсутність деталізованого аналізу витрат може призводити до недостатньо ефективного управління бюджетом компанії.

6. Обмежені можливості аналізу та звітності: деякі платформи не пропонують розширених можливостей аналітики та генерації гнучких звітів. Це може ускладнювати прийняття стратегічних рішень, оскільки компанія не отримує достатньо даних для оцінки ефективності використання транспорту.

7. Висока вартість впровадження та підтримки: деякі системи можуть бути досить дорогими у впровадженні та обслуговуванні. Для малого та середнього бізнесу витрати на програмне забезпечення можуть виявитися занадто високими, особливо якщо воно містить функціонал, який не використовується в повному обсязі.

8. Проблеми з технічною підтримкою: деякі користувачі стикаються з повільною або недостатньо кваліфікованою технічною підтримкою. Це може бути критично важливим у разі технічних збоїв або необхідності швидкого налаштування нових функцій.

Зважаючи на виявлені проблеми, при виборі або розробці нової системи управління автопарком варто звернути увагу на гнучкість інтеграції, персоналізацію, зручність інтерфейсу, якість мобільної версії, ефективність обліку витрат, розширені можливості аналізу та підтримку користувачів. Врахування цих аспектів дозволить створити платформу, яка буде максимально корисною для підприємств різного масштабу та сфер діяльності.

1.6 Обґрунтування доцільності проектування та розробки системи

Обґрунтування необхідності проектування та розробки нової системи управління автопарком базується на виявлених недоліках існуючих рішень, таких як Chevin FleetWave, Fleetio та Simply Auto. Як було зазначено в пункті 1.4, ці системи мають певні переваги, проте також містять суттєві обмеження, які негативно впливають на ефективність управління транспортом.

Основні проблеми, притаманні існуючим платформам, включають обмежену інтеграцію з іншими сервісами, недостатню персоналізацію, складний інтерфейс, труднощі з мобільним доступом, обмежені можливості аналізу витрат та звітності. Ці недоліки ускладнюють оперативне керування

автопарком і можуть призводити до неефективного використання ресурсів компанії.

Розробка нової інформаційної системи дозволить вирішити зазначені проблеми, запропонувавши гнучку, адаптивну та інтуїтивно зрозумілу платформу для управління транспортом. Вона сприятиме автоматизації процесів, покращенню аналітики, спрощенню контролю витрат та інтеграції з іншими корпоративними системами.

Таким чином, створення сучасної системи управління автопарком є доцільним кроком для підвищення ефективності роботи транспортних компаній, мінімізації витрат та забезпечення конкурентоспроможності на ринку логістичних і транспортних послуг.

1.7 Постановка задачі

Завданням проекту є розробка веб сайту для моніторингу технічного стану автомобілів, що дозволить користувачам зручно відстежувати всі операції, пов’язані з технічним обслуговуванням, витратами та плануванням ремонтних робіт.

Основні функції системи:

- Облік технічних робіт:
    - розроблення модуля для внесення інформації про проведення технічних робіт;
    - створення модуля для збереження електронних документів, пов’язаних із обслуговуванням;
    - впровадження модуля для підтвердження виконання операцій співробітниками сервісних станцій;
- Контроль витрат:
    - розроблення модуля статистики та моніторингу витрат;
    - створення алгоритму для розрахунку витрат на технічне обслуговування;
    - впровадження алгоритму для аналізу статистики технічних операцій над агрегатами транспортного засобу;

- Планування та сповіщення:
    - впровадження модуля для планування операцій над автомобілем;
    - розроблення модуля, що надсилає повідомлення про необхідність проведення технічного обслуговування;
    - створення алгоритму для автоматичної передачі історії технічного обслуговування автомобіля;
    - розроблення модуля для надсилання сповіщень на електронну адресу користувача;

Загальні вимоги до системи:

- мінімальні вимоги до технічних ресурсів, що забезпечать стабільну роботу додатку;
- висока швидкодія та ефективне використання ресурсів для зручності користувачів;
- інтуїтивно зрозумілий інтерфейс для швидкого доступу до інформації про транспортний засіб;
- надійність і безпека даних щодо технічного обслуговування та витрат;

Розробка такого додатку дозволить автоматизувати контроль стану транспортного засобу, підвищити ефективність управління витратами та запобігти несподіваним технічним несправностям.

1.8 Розрахунок економічного ефекту

Проєкт спрямований на автоматизацію процесу моніторингу технічного стану транспортних засобів, що дозволить значно покращити управління обслуговуванням, скоротити витрати на ремонти та збільшити експлуатаційну ефективність автопарку. Вебзастосунок надасть можливість вести журнал технічного обслуговування, планувати ремонти та аналізувати витрати на утримання транспорту.

Очікувані результати включають зменшення витрат на ремонти завдяки своєчасному обслуговуванню, зниження ризику несправностей, підвищення прозорості та зручності управління автопарком. Оцінка

економічної ефективності проєкту дозволить визначити його фінансову доцільність та очікуваний прибуток.

Основні етапи проєкту:

Аналіз вимог:

- трудомісткість (ТМ): 12 люд.-днів;
- чисельність працюючих (ЧП): 2 особи;
- тривалість (ТР): 6 днів;
- витрати: $1,800 (зарплата персоналу) + $700 (консультації) = $2,500;

Проєктування:

- трудомісткість (ТМ): 22 люд.-дні;
- чисельність працюючих (ЧП): 3 особи;
- тривалість (ТР): 8 днів;
- витрати: $3,200 (зарплата персоналу) + $800 (інструменти для проєктування) = $4,000;

Розробка та налаштування системи:

- трудомісткість (ТМ): 180 люд.-днів;
- чисельність працюючих (ЧП): 9 осіб;
- тривалість (ТР): 25 днів;
- витрати: $21,000 (зарплата розробників) + $5,000 (програмне забезпечення) = $26,000;

Тестування:

- трудомісткість (ТМ): 65 люд.-днів;
- чисельність працюючих (ЧП): 5 осіб;
- тривалість (ТР): 12 днів;
- витрати: $8,500 (зарплата тестувальників) = $8,500;

Впровадження:

- трудомісткість (ТМ): 35 люд.-днів;
- чисельність працюючих (ЧП): 3 особи;
- тривалість (ТР): 12 днів;

- витрати: $3,500 (зарплата) + $1,500 (сервери та налаштування) = $5,000;

Навчання персоналу:

- трудомісткість (ТМ): 25 люд.-днів;
- чисельність працюючих (ЧП): 2 особи;
- тривалість (ТР): 13 днів;
- витрати: $2,500 (тренінги та навчальні матеріали) = $2,500;

Підтримка та супровід:

- трудомісткість (ТМ): 250 люд.-днів;
- чисельність працюючих (ЧП): 5 осіб;
- тривалість (ТР): 60 днів;
- витрати: $15,000 (підтримка та обслуговування системи) = $15,000;

Загальні витрати проєкту:

2,500+4,000+26,000+8,500+5,000+2,500+15,000=**60,860;**

Джерела фінансування:

- власні кошти;
- банківські кредити;
- гранти;
- інвестиції;
- лізинг/оренда обладнання;
- розстрочка з постачальниками;

Очікуваний економічний ефект:

Зниження витрат на технічне обслуговування:

- річні витрати на систему: $85,000;
- очікуване зниження витрат: 18%;
- зниження витрат: $85,000 × 0.18 = $15,300;

Оптимізація використання транспорту:

- кількість транспортних засобів: 50;
- підвищення ефективності використання: 20%;
- додатковий економічний ефект: $12,000;

Скорочення простоїв через технічні несправності:

- середній час простою до автоматизації: 15 год./місяць;
- скорочення простоїв: 30%;
- економія: $8,500;

Загальний економічний ефект:

15,300+12,000+8,500= **$35,800;**

Окупність проєкту:

формула: окупність = загальні витрати / річний економічний ефект

$60,860 / $35,800 ≈ 1,7 років;

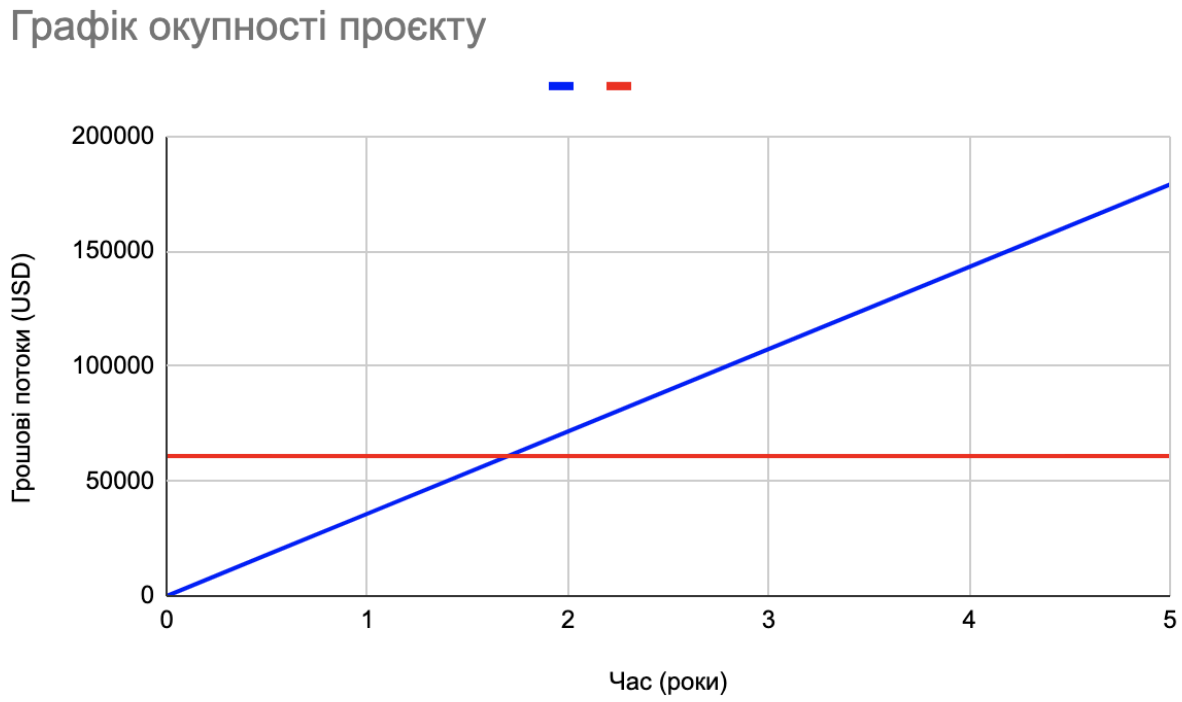

Рисунок 1.4 - Графік окупності проекту

Отже, термін окупності веб застосунку для ведення моніторингу технічного стану автомобілів становить **1.7 років**.

РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ НА ПРОЄКТУВАННЯ

2.1. Загальні положення

Дане технічне завдання визначає вимоги до створення вебзастосунку для моніторингу технічного стану транспортних засобів під назвою «AutoCheck Solutions». Основною метою проєкту є автоматизація процесів ведення обліку обслуговування автомобілів, управління інформацією про технічні події, формування звітності та підвищення ефективності контролю за технічним станом автопарку.

2.1.1. Відомості про проєкт

Розроблення інформаційної системи виконується в межах дипломного проєкту студента Мохонька Іллі Костянтиновича, спеціальності 122 – Комп’ютерні науки. У процесі реалізації використовуються сучасні засоби проєктування та розробки: мова програмування JavaScript, фреймворк React для фронтенду, Node.js з Express для серверної логіки, база даних MongoDB, а також API для інтеграції з іншими модулями системи.

2.1.2. Стандарти та гнучкість змін

Усі результати, створені в процесі розробки, оформлюються відповідно до вимог чинних державних стандартів у сфері створення програмного забезпечення. Послідовність і терміни виконання робіт визначаються графіком реалізації проєкту. У разі потреби, окремі положення технічного завдання можуть бути деталізовані або змінені в подальших етапах розробки.

2.2. Призначення системи та цілі системи

Розроблювана система AutoCheck Solutions призначена для централізованого ведення, контролю та аналізу інформації, пов’язаної з технічним станом автомобілів. Вона забезпечує зручне ведення цифрової історії обслуговування кожного транспортного засобу, включаючи дані про ремонтні роботи, діагностику, заміну комплектуючих, планові огляди та інші технічні події. Система орієнтована на збереження точних записів, надання

можливості додавання, редагування та видалення подій, а також автоматичне нагадування про майбутнє технічне обслуговування.

Основною метою створення AutoCheck Solutions є підвищення ефективності моніторингу технічного стану автомобілів та оптимізація взаємодії між водіями та сервісними центрами. Система дозволяє забезпечити достовірне збереження історії обслуговування, що сприяє зниженню ризиків пов’язаних з пропуском обов’язкових оглядів чи несвоєчасною заміною вузлів.

Сервіс підвищує рівень прозорості технічної інформації про автомобіль, що є важливою перевагою при купівлі-продажу транспортного засобу. Також цифровий підхід до обліку робіт дозволяє сформувати довіру до виконавців технічного обслуговування за рахунок фіксації та підтвердження виконаних процедур, що позитивно впливає на загальну безпеку експлуатації транспортних засобів.

2.3. Характеристика об’єкта автоматизації.

Об’єктом автоматизації виступає процес цифрового ведення, збереження та аналізу інформації щодо технічного стану транспортних засобів.

2.3.1. Основні функціональні можливості системи

Розроблена система «AutoCheck Solutions» реалізована як повнофункціональний вебзастосунок, який дозволяє працівникам станцій технічного обслуговування (СТО) додавати транспортні засоби, фіксувати їх технічні характеристики (модель, рік випуску, пробіг), а також прикріплювати зображення.

Система включає інтерактивний календар, що дає змогу створювати події, пов’язані з обслуговуванням — зокрема техогляд, ремонт, заміну шин, страхування та інші сервісні дії. Кожна подія фіксує тип, дату, витрати, опис, а також прив’язується до конкретного автомобіля.

Реалізовано автоматичне оновлення статусів подій на основі дати (наприклад, події зі строком у минулому отримують статус "виконано"), а

також модуль побудови аналітичних діаграм для аналізу витрат у розрізі місяців і категорій.

2.3.2. Основні функціональні можливості системи

Для зручності звітності користувач може згенерувати PDF-звіт, що включає інформацію про події для обраного автомобіля — з деталями про власника, типи обслуговування та витрати.

Таким чином, «AutoCheck Solutions» забезпечує ефективне та централізоване управління інформацією про технічне обслуговування транспорту, підвищує прозорість взаємодії між СТО та клієнтами, а також дозволяє формувати повну історію експлуатації автомобіля.

2.4 Вимоги до системи

2.4.1 Вимоги до системи в цілому

2.4.1.1 Вимоги до структури та функціонування системи

2.4.1.1.1. Система реалізується як веборієнтований додаток для моніторингу технічного стану автомобілів без поділу на ролі адміністратора чи директора. Основний функціонал надається зареєстрованим користувачам — переважно працівникам СТО або власникам авто.

2.4.1.1.2. Доступ до функцій системи забезпечується через вебінтерфейс з підтримкою сучасних браузерів. Інтерфейс адаптований для ПК, ноутбуків і планшетів.

2.4.1.1.3. Усі користувачі мають однаковий рівень доступу після авторизації. Основні можливості включають:

- додавання транспортних засобів (із фото, пробігом, роком випуску, власником);
- формування подій з обслуговування авто (з описом, вартістю, статусом, типом);
- календар подій з інтерактивною взаємодією;
- перегляд аналітики по витратах у вигляді діаграм;
- формування звітів за подіями з можливістю експорту в excel.

2.4.1.1.4. Система не інтегрується з платіжними сервісами, оскільки її призначення — ведення обліку, а не оплата.

2.4.1.1.5. Передбачено локалізацію інтерфейсу українською та англійською мовами.

2.4.1.2 Вимоги до кваліфікації користувачів

2.4.1.2.1. Користувачами є особи, які мають базові навички користування браузером. Не потрібно спеціальної IT-кваліфікації.

2.4.1.2.2. Функціональність надається через особистий кабінет, з якого можна:

- додавати/редагувати/видаляти авто;
- формувати та змінювати події ТО;
- переглядати графіки витрат;
- формувати excel-звіти.

2.4.1.3 Функціональні можливості системи

2.4.1.3.1. Система повинна:

- дозволяти додавати автомобілі з фото;
- фіксувати події ТО (із датою, назвою, типом, вартістю, описом);
- оновлювати статус подій автоматично за датою;
- виводити графіки витрат за категоріями і часом;
- генерувати звіти по подіях у вигляді excel.

2.4.1.3.2. Інтерфейс має бути швидким — не більше 2 секунд затримки при виконанні запиту.

2.4.1.4 Вимоги до надійності

2.4.1.4.1. Вебсайт має бути доступним 24/7. У разі відмови — час відновлення не повинен перевищувати 1 год.

2.4.1.4.2. Дані повинні автоматично зберігатися у MongoDB, а також резервно копіюватися періодично (через додаткові сервіси хостингу).

2.4.1.5 Вимоги до безпеки

2.4.1.5.1. Авторизація захищена JWT[19] токеном. Доступ до маршрутів API обмежується middleware.

2.4.1.5.2. Усі паролі хешуються. Персональні дані передаються через HTTPS.

2.4.1.6. Вимоги з ергономіки та технічної естетики

Загальні ергономічні і естетичні вимоги до вебсайту повинні відповідати держстандартам ДСТУ 8604:2015, ДСТУ 7298:2013. Освітленість робочого місця користувача повинна відповідати нормам ДСТУ EN 12464-1:2016, ДБН В.2.5-28-2006.

Засоби відображення інформації (екрани пристроїв) повинні розміщуватись таким чином, щоб кут спостереження не перевищував 45°, мінімальна відстань до екрана становила не менше 0,3 м, а рекомендована — 0,5 м.

Інтерфейс вебсайту повинен бути зручним, зрозумілим і естетично привабливим, що сприятиме зниженню втомлюваності користувача під час взаємодії з системою.

2.4.1.7. Вимоги по експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи

2.4.1.7.1. Види обслуговування системи визначаються у відповідності з ДСТУ EN 13306:2019. Загальні вимоги по експлуатації, технічному обслуговуванню і ремонту повинні відповідати ДСТУ 3576-97.

2.4.1.7.2. Для розміщення технічних засобів, які забезпечують функціонування вебсайту (сервери, мережеве обладнання тощо), необхідні площі, визначені в ДБН В.2.2-9-2009. Слід дотримуватися вимог експлуатаційної документації. Напруга живлення обладнання має становити 220/380 В змінного струму з частотою (50±1) Гц. Допустиме відхилення напруги — від +10% до –15%, тривалість перерв у живленні — не більше 0,001 с.

2.4.1.7.3. Кількість, кваліфікація і режими роботи обслуговуючого персоналу повинні відповідати рекомендаціям, зазначеним у технічних умовах і інструкціях з експлуатації апаратних засобів системи.

2.4.1.7.4. Склад, розміщення і умови зберігання компонентів технічних засобів, що забезпечують роботу вебсайту (серверів, пристроїв резервного живлення тощо), визначаються згідно з рекомендаціями експлуатаційної документації на відповідні елементи.

2.4.1.7.5. Регламент обслуговування компонентів повинен відповідати умовам їх експлуатації. У разі відмови системи повинна бути забезпечена її робота в аварійному режимі або відновлення з резервної копії.

2.4.1.8. Вимоги до захисту інформації від несанкціонованого доступу

Для забезпечення надійності збереження та контролю доступу до інформації, вебсайт повинен використовувати сучасні засоби захисту, зокрема:

1. засоби захисту серверних операційних систем (наприклад, Windows Server, Ubuntu Server) — використання стандартних політик безпеки, обмеження прав доступу, аудит доступу до файлів і системних ресурсів;
2. захист локальної мережі та інтернет-з'єднання за допомогою Firewall, NAT, засобів виявлення вторгнень (IDS/IPS);
3. захист на рівні клієнт-серверної СУБД, зокрема:
4. використання тригерів, представлень (views), збережених процедур і функцій;
5. створення груп користувачів із різними ролями доступу;
6. заборона на пряме редагування певних критичних таблиць.

Кожен сеанс роботи з вебсайтом повинен починатися з автентифікації користувача шляхом введення логіну та індивідуального паролю. Парольний захист має реалізовувати функції періодичної зміни пароля (через відповідні механізми у фреймворку або ОС).

Для захисту від несанкціонованого доступу, кожен користувач повинен мати унікальні облікові дані, а доступ до чутливої інформації (наприклад, історії технічного обслуговування) повинен бути обмежений відповідно до ролі користувача в системі.

2.4.1.9. Вимоги щодо збереження інформації при аваріях.

2.4.1.9.1. Необхідно передбачити засоби резервного копіювання бази даних вебсайту після будь-яких змін у ній, а також можливість її відновлення з резервної копії у випадку пошкодження або втрати.

2.4.1.9.2. Резервна копія бази даних повинна зберігатися на окремому фізичному або хмарному носії, незалежному від основного серверного середовища.

2.4.1.10. Вимоги по захисту від впливу зовнішніх діянь.

2.4.1.10.1. Електрична складова електромагнітного поля в приміщеннях, де встановлене серверне обладнання вебсайту, не повинна перевищувати 0,3 В/м² в діапазоні частот від 0,15 до 300 МГц. Для захисту від впливу електромагнітних полів та індустріальних завад слід застосовувати екрани та фільтри.

2.4.1.10.2. Засоби, що виключають вплив шкідливих факторів на функціонування технічних компонентів системи, повинні бути реалізовані відповідно до вимог ДБН В.2.2-9-2009. Обчислювальні засоби повинні відповідати вимогам ДСТУ 2506-94 щодо стійкості до зовнішніх впливів.

2.4.1.11. Вимоги до патентної чистоти.

При створенні вебсайту для моніторингу технічного стану автомобілів патентні дослідження не проводяться.

2.4.1.12. Вимоги по стандартизації і уніфікації.

Кодування та зберігання інформації у системі має відповідати чинним міжнародним класифікаторам і стандартам, а також забезпечувати можливість масштабування та інтеграції з іншими інформаційними системами.

2.4.2 Вимоги до функцій

Таблиця 2.1. Перелік функцій системи

№	Назва функції	Вхідна інформація	Вихідна інформація
1	Реєстрація користувача	Ім’я, email, номер телефону, пароль	Токен авторизації, обліковий запис
2	Авторизація	Email, пароль	Токен доступу, вхід у систему
3	Додавання авто	Власник, марка, модель, пробіг, рік, фото	Новий запис у БД
4	Редагування/видалення авто	ID авто, нові дані або дія "видалити"	Оновлений або видалений запис

Продовження Таблиці 2.1

5	Додавання події	Дата, назва, тип, опис, авто, вартість, статус	Запис у базу бронювання, підтвердження заявки
6	Редагування події	Подія, зміни	Оновлений запис
7	Видалення події	Подія	Видалення з бд
8	Автоматичне оновлення статусу	Дата події	Якщо дата < поточної — статус → "виконано"
9	Генерація excel-звітів	Події користувача	excel-файл із таблицею подій
10	Візуалізація витрат	Події з БД	Кругова/лінійна діаграма

2.4.3 Вимоги до видів забезпечення

2.4.3.1 Вимоги до математичного забезпечення (МЗ)

Для реалізації функціональності вебзастосунку «AutoCheck Solutions» не потрібно створення спеціалізованих математичних модулів. Всі операції, включаючи облік технічного обслуговування, побудову графіків витрат, обробку подій і взаємодію з базою даних, реалізовані через логіку, вбудовану у програмний код. Обчислення зводяться до простих агрегацій (підсумки, групування по категоріях або датах), які виконуються через запити до бази MongoDB або обробляються на рівні клієнта в React.

2.4.3.2 Вимоги до інформаційного забезпечення (ІЗ)

2.4.3.2.1 Система зберігає інформацію про автомобілі, технічні події, витрати, статуси обслуговування, користувачів і аналітичні показники. Всі дані організовані у відповідних колекціях MongoDB, з чіткою структурою, що відповідає логічній моделі бази.

Доступ до даних реалізовано через REST API, що дозволяє обмінюватися інформацією між клієнтом і сервером. Кожен користувач має доступ лише до власних автомобілів і пов’язаних із ними подій.

2.4.3.2.2 Передбачено автоматичне створення резервних копій бази даних, що дає змогу швидко відновити інформацію у разі збою чи втрати даних. У разі розгортання в хмарі (наприклад, через MongoDB Atlas) ці функції реалізуються на рівні хмарного провайдера.

2.4.3.3 Вимоги до лінгвістичного забезпечення (ЛЗ)

2.4.3.3.1 Розробка клієнтської частини виконувалась з використанням JavaScript і React, серверна — на Node.js (з Express), а робота з базою даних — через Mongoose (MongoDB ORM). Запити до бази не потребують SQL, адже застосовується NoSQL-підхід.

2.4.3.3.2 Інтерфейс користувача підтримує багатомовність через бібліотеку i18next, що дає змогу легко перемикатися між українською та англійською мовами. Діалоги, підказки і повідомлення максимально адаптовані для зручності сприйняття, з урахуванням принципів UX/UI.

2.4.3.4 Вимоги до програмного забезпечення (ПЗ)

2.4.3.4.1 Платформа підтримується в середовищі Node.js. На сервері може використовуватись як Ubuntu Server, так і Windows Server. Клієнтський додаток функціонує в сучасних браузерах на Windows, macOS, Android або iOS. Для зберігання даних використовується MongoDB, розгорнута локально або в хмарі.

2.4.3.4.2 Загальні вимоги до ПЗ:

- ефективна обробка користувацьких запитів;
- оптимізоване використання пам’яті;
- мінімальна затримка при роботі з формами, графіками, таблицями.

2.4.3.4.3 Операційна система має підтримувати Node.js, MongoDB, npm/yarn.

2.4.3.4.4 Вимоги до СУБД:

- підтримка авторизації і захисту даних;

- гнучке моделювання документів;
- можливість агрегації та фільтрації великих обсягів даних;
- наявність механізмів бекапу.

2.4.3.4.5 Система повинна забезпечувати:

- введення та редагування подій, автомобілів, профілів;
- валідацію даних у формах;
- побудову аналітичних графіків (Pie, LineChart);
- генерацію звітів у форматі PDF.

2.4.3.4.6 Весь код реалізовано у вигляді окремих логічних модулів та компонентів. Компоненти React реалізують повторно використовувані елементи UI, а серверні маршрути — RESTful API. Це спрощує масштабування та супровід проєкту.

2.4.3.5 Вимоги до технічного забезпечення

2.4.3.5.1 Система повинна стабільно функціонувати на сервері зі щонайменше 2 ядрами CPU, 4 ГБ оперативної пам’яті, дисковим простором від 50 ГБ. Клієнтська частина не потребує спеціального ПЗ, окрім сучасного браузера (Chrome, Firefox, Edge). Для тестування використовувались локальні середовища (Node.js + MongoDB на Windows).

Таблиця 2.2. Вимоги до технічного забезпечення системи

Основні характеристики обладнання	Тип
Сервер: Intel Xeon / 8–16 GB RAM / SSD / 1 Gbit LAN	Технічне забезпечення сервера
Клієнт: AMD/Intel 2.0 GHz, 4 GB RAM, HDD/SSD 250+ Gb	Технічне забезпечення сервера
Монітор 17"+, клавіатура, миша	Робоче місце користувача

2.4.3.6 Вимоги до метрологічного забезпечення

Оскільки програмний продукт AutoCheck Solutions не пов’язаний із вимірюванням фізичних величин або використанням спеціалізованих вимірювальних приладів, необхідність у метрологічному забезпеченні відсутня. Система не передбачає обробку параметрів, які підпадають під контроль метрологічних норм. Всі розрахунки в межах системи (наприклад, сума витрат, підсумкові аналітичні значення) виконуються у межах стандартної прикладної логіки без використання зовнішніх сенсорів або пристроїв збору даних.

2.4.3.7 Вимоги до організаційного забезпечення

2.4.3.7.1 Організаційне забезпечення роботи вебзастосунку AutoCheck Solutions базується на принципах децентралізованої взаємодії між користувачами, які мають обмежений набір ролей. Оскільки система не включає адміністративну панель із широкими правами доступу, а також не реалізує класичну багаторівневу ієрархію доступу, до складу організаційного забезпечення не входить централізоване адміністрування.

2.4.3.7.2 Запровадження системи не вимагає змін у штатній структурі підприємства або додаткового навчання персоналу. Достатньо базових навичок користування браузером, оскільки система реалізована як вебзастосунок із інтуїтивним інтерфейсом. Робота користувачів передбачає самостійне додавання транспортних засобів, створення подій у календарі, перегляд історії обслуговування та формування звітів.

2.4.3.7.3 До організаційних вимог входять:

- доступ до системи здійснюється лише після авторизації користувача за токеном доступу (JWT);
- кожен користувач працює лише зі своїми даними, немає можливості переглядати або змінювати інформацію інших користувачів;
- у разі розгортання системи на підприємстві, визначення відповідальної особи за технічну підтримку (локально або через провайдера хостингу) здійснюється адміністрацією компанії.

2.5. Склад і зміст робіт по створенню системи

2.5.1. Стадії створення системи і терміни виконання робіт наведені в таблиці 2.3.

Таблиця 2.3. Стадії створення системи і терміни виконання робіт

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження об’єкта автоматизації	01.05.2025	Виконано
2	Розробка технічного завдання	05.05.2025	Виконано
3	Проєктування БД	06.05.2025	Виконано
4	Реалізація програмного забезпечення	08.05.2025	Виконано
5	Тестування системи	10.05.2025	Виконано
6	Підготовка документації	12.05.2025	Виконано
7	Встановлення та налаштування системи	15.05.2025	Виконано
8	Оформлення документації	18.05.2025	Виконано

2.6. Порядок контролю і приймання системи

2.6.1. Вебзастосунок AutoCheck Solutions впроваджується в умовах реального використання користувачами — працівниками станцій технічного обслуговування та власниками транспортних засобів. На етапі здачі системи проводяться приймальні випробування для підтвердження її працездатності. Оцінювання функціональності здійснюється відповідно до технічного завдання, положень про забезпечення якості та вимог ДСТУ 3974-2000.

2.6.2. Перевірка відповідності програмного продукту вимогам здійснюється спільно розробником та користувачами шляхом запуску

системи у тестовому середовищі. Тестування охоплює основні модулі: реєстрацію, додавання авто, управління подіями в календарі, генерацію звітів, відображення аналітики. Програма випробувань формується розробником відповідно до реалізованого функціоналу, а результати фіксуються у протоколі.

2.6.3. Введення системи в дослідну експлуатацію проводиться на основі технічної документації (технічне завдання, технічний проєкт, інструкція користувача). За результатами дослідної експлуатації можуть бути сформульовані уточнення або рекомендації щодо покращення функціоналу (наприклад, розширення аналітики або інтерфейсні вдосконалення).

2.6.4. Після завершення тестування та усунення виявлених недоліків оформлюється акт прийому-передачі системи в продуктивне середовище. Документ підписується представниками розробника та замовника або відповідальною особою за впровадження.

2.7. Вимоги до складу і змісту робіт із підготовки до введення системи в дію

Перед запуском вебзастосунку AutoCheck Solutions у промислове використання замовник або відповідальна особа повинна виконати наступні дії:

- створення облікових записів для працівників СТО, які будуть додавати транспортні засоби клієнтів;
- внесення початкових даних про наявні автомобілі (бренд, модель, пробіг, рік випуску, ім’я власника);
- перевірка роботи основних функцій: додавання подій, редагування та фільтрація записів, генерація статистики;
- у разі необхідності — проведення ознайомчого інструктажу для користувачів;
- підготовка та передача користувачам короткої інструкції з описом базових дій.

2.8. Вимоги до документації

2.8.1. Для інформаційної системи AutoCheck Solutions розробляється повний комплект проєктної документації, до якого входить: технічне завдання, технічний проєкт, логічна схема бази даних, інструкція користувача та пояснювальна записка.

2.8.2. Уся документація створюється відповідно до стандартів серій ДСТУ 19 (Єдина система програмної документації) та ДСТУ 24 (Автоматизовані системи управління). Структура та обсяг документів відповідають рівню складності системи та охоплюють всі аспекти реалізованого функціоналу.

2.9. Джерела розробки

2.9.1. При розробленні технічного завдання на вебсайт для моніторингу технічного стану автомобілів використано такі нормативні документи:

- ДСТУ 3008-2015 — Документація. Звіти у сфері науки і техніки. Структура та правила оформлювання;
- ДСТУ 3973–2000 — Система розроблення та поставлення продукції на виробництво;
- ДСТУ Б В.2.5–82:2016 — Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом.
- Онлайн-сервіс для перевірки протоколу обов’язкового технічного контролю транспортних засобів[30] .
- Приклади відкритих систем для введення технічного огляду автомобілів.

РОЗДІЛ 3. ПРОЄКТУВАННЯ, СТВОРЕННЯ ТА АПРОБАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Опис та обґрунтування вибору програмно-технічних засобів

Під час створення веборієнтованої інформаційної системи AutoCheck Solutions, призначеної для ведення обліку технічного стану транспортних засобів, було обрано набір програмних та технічних засобів, що забезпечують стабільність, масштабованість і зручність у подальшій експлуатації та підтримці системи.

У процесі розробки було використано фреймворк Node.js у поєднанні з бібліотекою Express.js для побудови серверної частини. Такий стек технологій дозволив реалізувати RESTful API, необхідне для обміну даними між клієнтською частиною й базою даних. Express.js забезпечив гнучке налаштування маршрутизації та підтримку модульності коду.

У ролі бази даних обрано MongoDB — документно-орієнтовану СУБД NoSQL, яка ідеально підходить для роботи з гнучкими структурами даних. MongoDB дозволяє зберігати записи про автомобілі, історію обслуговування, події в календарі та інші сутності у вигляді документів формату JSON. Підключення до бази було реалізовано за допомогою ODM-бібліотеки Mongoose, яка дозволила ефективно працювати з моделями, валідацією та запитами до колекцій.

Фронтенд-частина реалізована на основі React — популярної JavaScript-бібліотеки для створення динамічних інтерфейсів. Компонентна архітектура React дала можливість організувати структуру додатку логічно, розділивши його на незалежні модулі (сторінки, компоненти, блоки). Для стилізації інтерфейсу використано Tailwind CSS, що забезпечило швидке створення адаптивних елементів дизайну та підтримку темної/світлої теми. Додатково застосовувались бібліотеки Recharts для побудови аналітичних графіків і React Calendar для інтеграції календарного планування.

Таким чином, обраний стек технологій дозволив створити повноцінну, стабільну, зручну у використанні та розширювану систему. Усі компоненти

проєкту взаємодіють за принципом клієнт–сервер. Архітектурна схема системи представлена в додатку А на рисунку А.1.

3.2 Проєктування та створення бази даних

У процесі розроблення веборієнтованої інформаційної системи для моніторингу технічного стану транспортних засобів AutoCheck Solutions було спроєктовано і реалізовано базу даних на основі документоорієнтованої СУБД MongoDB. Завдяки гнучкості структури документів JSON, система забезпечує зручне зберігання, пошук і оновлення інформації, пов’язаної з автопарком, подіями обслуговування, користувачами та календарними записами.

Базу даних було спроєктовано з використанням бібліотеки Mongoose, що дозволяє створювати схеми моделей, реалізовувати зв’язки між сутностями, накладати валідаційні обмеження та забезпечувати інтеграцію з API серверної частини.

3.2.1 Структура та призначення таблиць

Основні колекції системи наведені в таблиці 3.1.

Таблиця 3.1 – Основні колекції в базі даних AutoCheck

Назва таблиці	Призначення
users	Містить облікові записи користувачів (ПІБ, email, пароль, роль, телефон)
cars	Зберігає дані про транспортні засоби (бренд, модель, рік, пробіг, фото, власник)
events	Містить календар подій, пов’язаних із технічним обслуговуванням (назва, тип, дата, сума, опис, статус, прив'язка до авто)
serviceLogs	Архів дій з обслуговування (використовується для побудови статистики витрат)

Дані структури забезпечують централізоване ведення обліку технічних робіт, збереження історії по кожному авто, автоматичне формування графіків витрат, а також зберігають зв’язок між користувачем, автомобілем та обслуговуванням.

3.2.2 Зв’язки між таблицями

Взаємозв’язки реалізовано за допомогою ObjectId (аналог ForeignKey в SQL). Основні логічні зв’язки:

- один користувач (users) може володіти кількома автомобілями – users 1:N cars;
- один автомобіль (cars) може мати багато подій (events) – cars 1:N events;
- подія (event) обов’язково має бути пов’язана з конкретним авто (через carId) і користувачем (userId);
- дані з events агрегуються для побудови діаграм витрат у stats;

3.2.3 Фрагменти моделей (MongoDB + Mongoose)

У проєкті AutoCheck Solutions зберігання інформації реалізовано за допомогою MongoDB, де моделі описані з використанням бібліотеки Mongoose. Нижче подано приклади ключових моделей, які було створено в процесі розробки системи.

Модель користувача (User):

```
const UserSchema = new mongoose.Schema({
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  fullName: { type: String, required: true },
  phone: { type: String, required: true },
  role: { type: String, enum: ['user'], default: 'user' },
  createdAt: { type: Date, default: Date.now }
});
```

Ця модель зберігає облікові дані користувачів. Вона включає email, пароль (захешований), повне ім’я, телефон та роль користувача.

Модель автомобіля (Car):

```
const CarSchema = new mongoose.Schema({
  brand: { type: String, required: true },
  model: { type: String, required: true },
  year: { type: String, required: true },
  mileage: { type: String, required: true },
  image: { type: String },
  ownerName: { type: String, required: true },
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
  createdAt: { type: Date, default: Date.now }
});
```

Містить інформацію про транспортний засіб, включаючи марку, модель, рік випуску, пробіг, зображення, а також ім’я власника.

Модель події технічного обслуговування (Event):

```
const EventSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  carId: { type: mongoose.Schema.Types.ObjectId, ref: 'Car', required: true },
  title: { type: String, required: true },
  type: { type: String, enum: ['ТО', 'ремонт', 'страхування', 'шини', 'інше'], default: 'інше' },
  description: { type: String },
  cost: { type: Number, default: 0 },
  date: { type: Date, required: true },
```

```
  status: { type: String, enum: ['заплановано', 'виконано', 'скасовано'], default: 'заплановано' },
  createdAt: { type: Date, default: Date.now }
});
```

Використовується для збереження даних про технічні події, пов’язані з автомобілями: ТО, ремонт, заміна шин тощо.

Повні схеми моделей event.model.js, car.model.js, user.model.js розміщено у **додатку Г**.

3.2.4 Особливості реалізації та підтримки бази даних

Вся робота з базою даних реалізована через Mongoose, що дозволяє створювати та підтримувати структуру колекцій, проводити CRUD-операції, застосовувати валідацію та реалізовувати зв’язки між моделями.

Зв’язки між сутностями реалізовано за допомогою ObjectId. Наприклад, кожна подія Event прив’язується до конкретного авто Car, яке в свою чергу належить користувачу User.

В системі автоматично оновлюється статус подій — наприклад, якщо дата виконання події минула, статус змінюється на "виконано".

Для безпеки реалізовано резервне копіювання бази за допомогою команд mongodump та можливість відновлення з mongorestore.

Вся робота з БД контролюється через Node.js сервер, а запити надсилаються з фронтенд-частини, реалізованої на React.

Створення та оновлення об'єктів відбувається через REST API (/api/cars, /api/events, /api/users).

3.3 Реалізація функцій системи

Веборієнтована інформаційна система AutoCheck Solutions була реалізована з урахуванням усіх функціональних вимог, які визначені в технічному завданні. Основна логіка взаємодії реалізована на JavaScript (React.js) для клієнтської частини, а серверна логіка — на Node.js з

використанням фреймворку Express.js. Обробка запитів, взаємодія з базою даних MongoDB, авторизація, робота з подіями та автомобілями — усе реалізовано відповідно до REST-архітектури.

Для кращого розуміння структури та взаємодії компонентів системи створено UML-діаграму ролей користувачів, яку наведено в додатку Б, рисунок Б.1. Повні версії основних файлів проєкту (наприклад, auth.routes.js, car.routes.js, event.routes.js, user.model.js, client.jsx) наведені у додатку В.

3.3.1 Реєстрація та авторизація користувачів

Система підтримує механізм реєстрації та авторизації користувачів на основі JWT-токенів (Рис. 3.1.).

Під час реєстрації новий користувач заповнює наступні поля:

- ім’я;
- прізвище;
- email;
- номер телефону;
- пароль;
- країна (обирається з випадаючого списку з кодом);

Після надсилання даних система виконує базову валідацію (перевірка email, довжини пароля, формату телефону) та створює нового користувача в базі даних. Пароль зберігається у зашифрованому вигляді за допомогою бібліотеки bcrypt.

Одразу після виконання реєстрації або входу користувач отримує токен доступу (JWT), який зберігається у локальному сховищі браузера. Цей токен автоматично додається до всіх запитів до захищених маршрутів API, зокрема:

- /api/cars — додавання, редагування або перегляд авто;
- /api/events — робота з подіями обслуговування;
- /api/users/update — оновлення профілю.

Реалізація функції signup:

```
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const User = require('../models/User');

exports.register = async (req, res) => {
 try {
  const { email, password, firstName, lastName, phone } = req.body;

  if (!email || !password || !firstName || !lastName || !phone) {
   return res.status(400).json({ message: 'Усі поля обов’язкові' });
  }

  const existingUser = await User.findOne({ email });
  if (existingUser)
   return res.status(400).json({ message: 'Користувач вже існує' });

  const hashedPassword = await bcrypt.hash(password, 10);
  const fullName = `${firstName} ${lastName}`;

  const newUser = new User({
   email,
   password: hashedPassword,
   fullName,
   phone
  });

  await newUser.save();

  const token = jwt.sign(
```

```
    { id: newUser._id, email: newUser.email },
    process.env.JWT_SECRET,
    { expiresIn: '7d' }
  );

  res.status(201).json({
    token,
    user: {
      fullName: newUser.fullName,
      email: newUser.email,
      phone: newUser.phone
    }
  });
} catch (err) {
  console.error('❌ Реєстрація помилка:', err);
  res.status(500).json({ message: 'Помилка сервера при реєстрації' });
}};
```

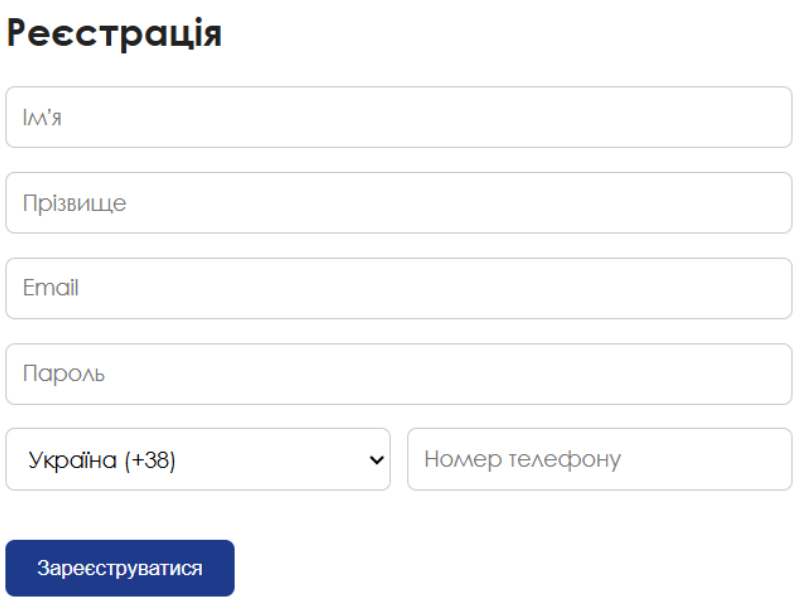

Рисунок 3.1 – Форма реєстрації

Реалізація Методу login:

```
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const User = require('../models/User');

exports.login = async (req, res) => {
  try {
    const { email, password } = req.body;

    // Перевірка заповнення полів
    if (!email || !password) {
      return res.status(400).json({ message: 'Вкажіть email та пароль' });
    }

    // Пошук користувача
    const user = await User.findOne({ email });
    if (!user) {
      return res.status(404).json({ message: 'Користувача не знайдено' });
    }

    // Перевірка пароля
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
      return res.status(401).json({ message: 'Невірний пароль' });
    }

    // Генерація JWT токена
    const token = jwt.sign(
      { id: user._id, email: user.email },
      process.env.JWT_SECRET,
```

```
    { expiresIn: '7d' }
  );

  res.status(200).json({
    token,
    user: {
      fullName: user.fullName,
      email: user.email,
      phone: user.phone
    }
  });
} catch (err) {
  console.error('❌ Помилка авторизації:', err);
  res.status(500).json({ message: 'Серверна помилка при авторизації' });
}};
```

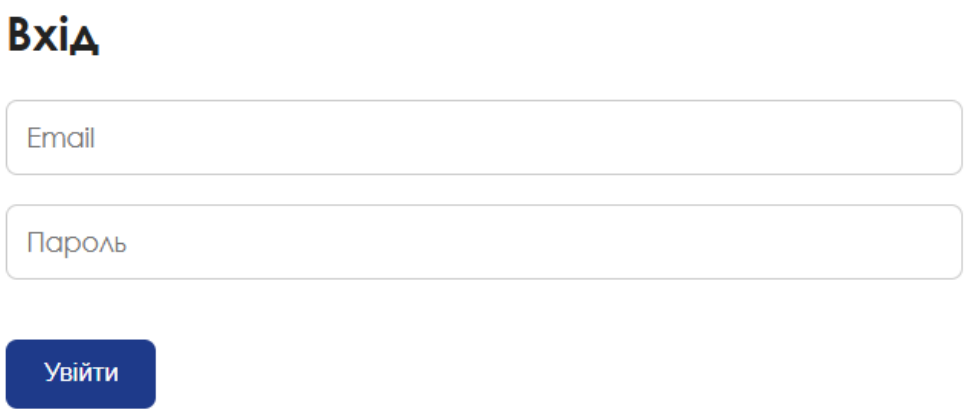

Рисунок 3.2 – Форма входу в обліковий запис

3.3.2 Додавання даних про авто

Після реєстрації або входу в обліковий запис працівник може додавати, редагувати і видаляти дані про авто які знаходяться в базі даних перейшовши до відповідної сторінки "Авто" (Рис. 3.3.).

Фрагмент коду зі сторінки CarsPage:

```
const CarsPage = () => {
  const [cars, setCars] = useState([]);
  const [form, setForm] = useState({
    ownerName: '',
    brand: '',
    model: '',
    year: '',
    mileage: '',
    image: null
  });
```

Рисунок 3.3 – Блок з додаванням авто

Після внесення у всіх потрібних даних працівник може фільтрувати авто за усіма видами сортувань (Рис. 3.4.).

Фрагмент коду з блоку Filters:

```
<div style={{ marginBottom: '2rem', display: 'flex', gap: '10px', flexWrap: 'wrap' }}>
        <input
          placeholder="Пошук марки/моделі"
          value={filters.search}
          onChange={(e) => setFilters({ ...filters, search: e.target.value })}
        />
        <input
          type="text"
          placeholder="Рік"
          value={filters.year}
          onChange={(e) => setFilters({ ...filters, year: e.target.value })}
        />
        <input
          type="number"
          placeholder="Пробіг від"
          value={filters.mileageFrom}
          onChange={(e) => setFilters({ ...filters, mileageFrom: e.target.value })}
        />
        <input
          type="number"
          placeholder="Пробіг до"
          value={filters.mileageTo}
```

```
    onChange={(e) => setFilters({ ...filters, mileageTo: e.target.value })}
  />
</div>
```

Рисунок 3.4 – Форма з усіма фільтрами та сортуванням

Для перевірки роботи фільтрів і сортувань задаємо марку автомобіля (BMW). Одразу після внесення марки авто виходить потрібний нам результат (Рис. 3.5.).

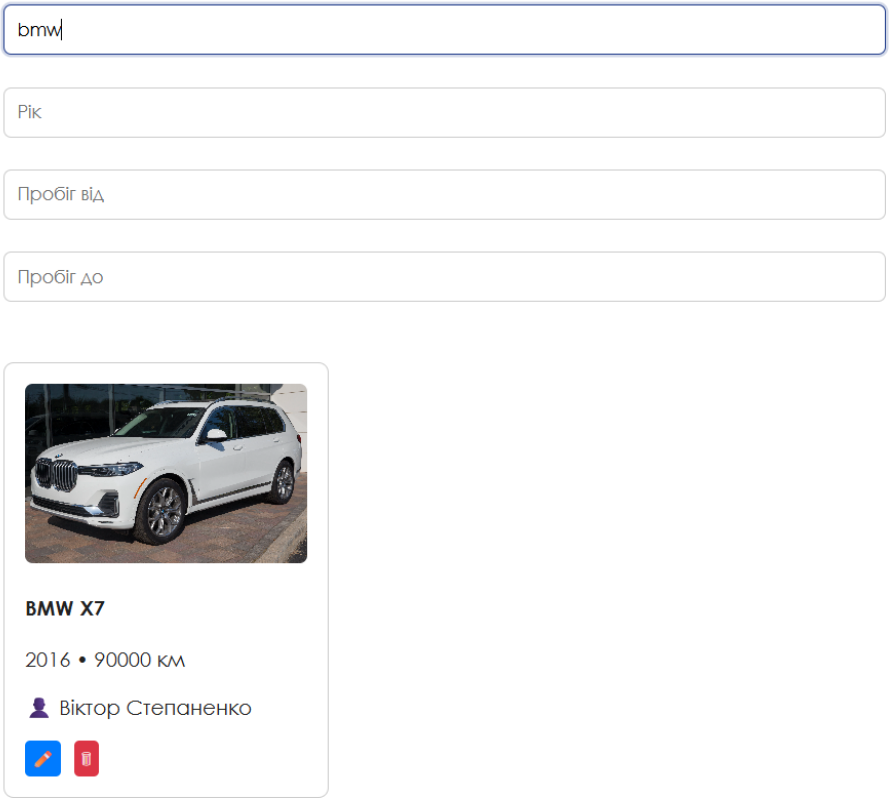

Рисунок 3.5 – Результат фільтрації і сортування на сторінці

3.3.3 Додавання подій до графіку. Видалення та редакція подій

Після занесення даних про авто працівник має додати подію з надянням послуг сервісного центру до календаря. Для цього потрібно перейти в розділ «Панель», де можна побачити сам календар та форму для додавання подій (Рис. 3.6.).

Фрагмент коду з AddEventForm:

```
useEffect(() => {
  if (eventToEdit) {
    setForm({
      carId: eventToEdit.carId?._id  '',
      date: eventToEdit.date?.slice(0, 10)  '',
      type: eventToEdit.type  'TO',
      cost: eventToEdit.cost  '',
      title: eventToEdit.title  '',
      description: eventToEdit.description  '',
      status: eventToEdit.status || 'заплановано'
    });
    setMessage('');
    setError('');
  }
}, [eventToEdit]);
```

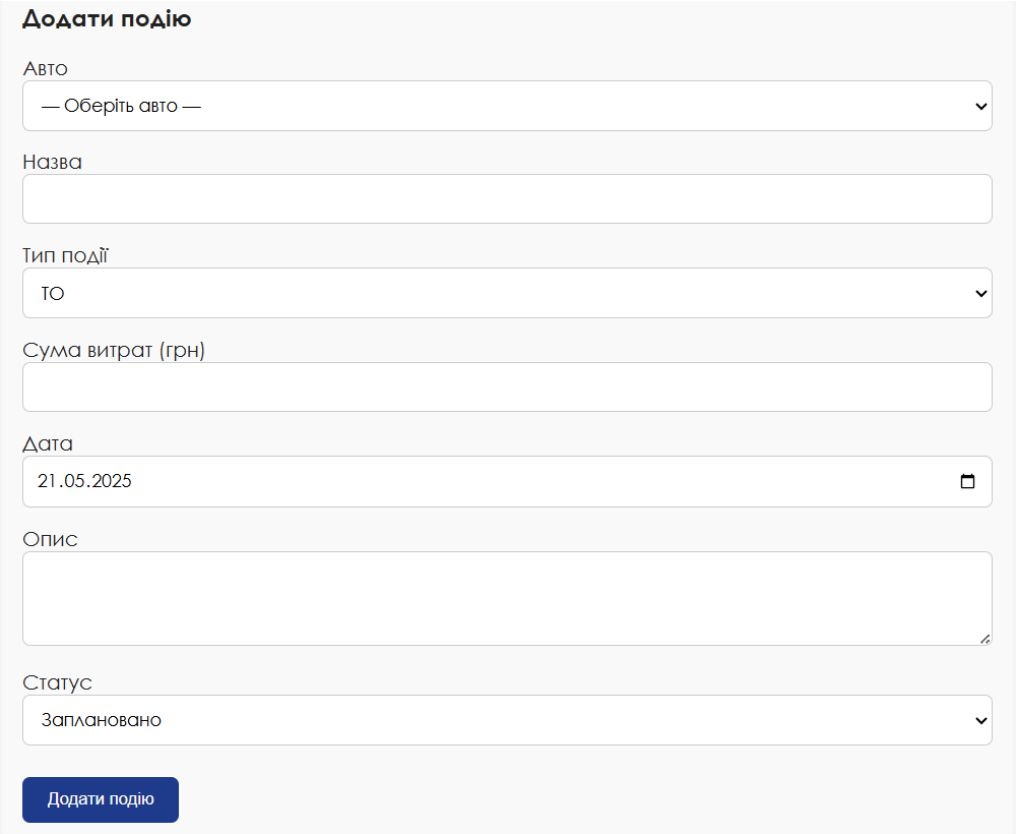

Рисунок 3.6 – Форма для для додавання подій

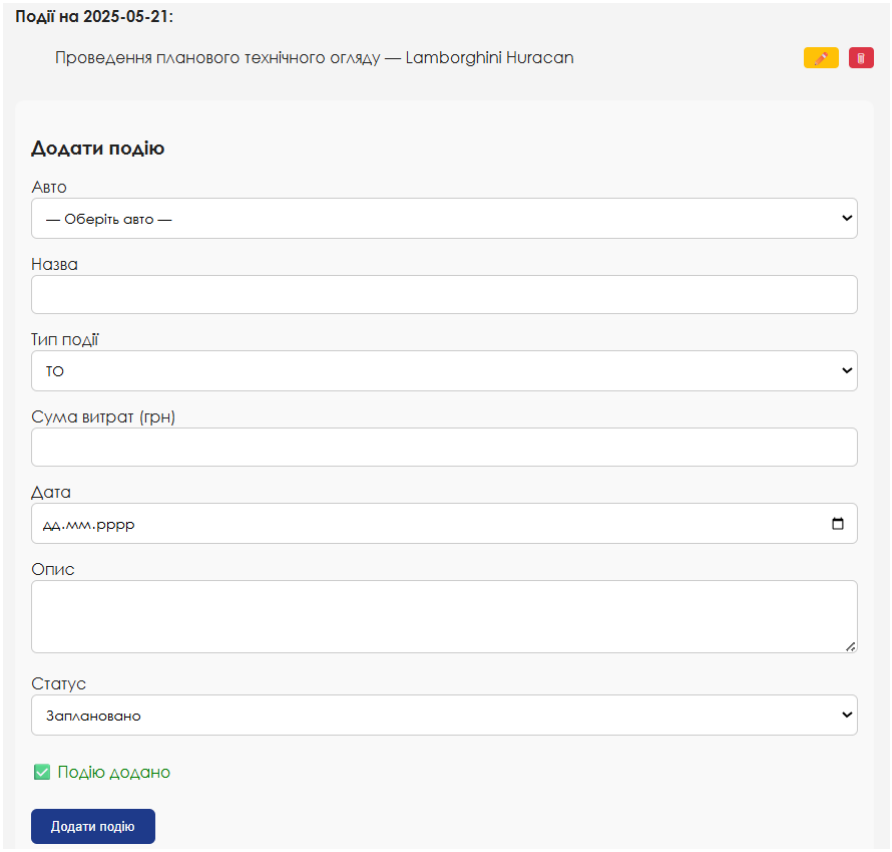

Рисунок 3.7 – Успішне додавання

На даному етапі працівник може переглянути подію у календарі, редагувати або видалити дану подію (Рис. 3.8.).

Фрагмент коду з блоку CalendarBlock:

```
<Calendar
    onChange={setDate}
    value={date}
    locale={i18n.language}
    calendarType="gregory"
    tileClassName={({ date }) => {
      const d = format(date, 'yyyy-MM-dd');
      return events.some(e => e.date?.slice(0, 10) === d) ? 'event-day' :
null;
    }}
```

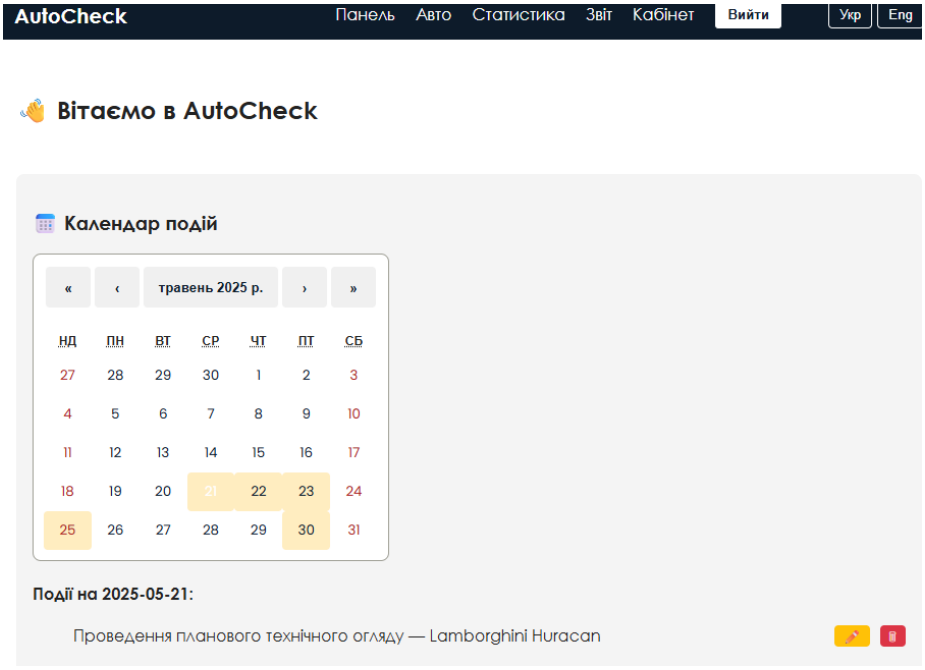

Рисунок 3.8 – Відображення календаря з подіями

Для демонстрації функцій спробуємо для початку редагувати подію, після натискання кнопки «Редагування» інформація вивантажується на екран

і готова до редагування, після цього натискаємо кнопку «Оновити дані», після чого вони успішно зберігаються (Рис. 3.9.).

Код 3.10. Представлення функції редагування:

```
try {
    if (eventToEdit?._id) {
      await API.put(/api/events/${eventToEdit._id}, form);
      setMessage('✅ Подію оновлено');
      clearEdit();
    } else {
      await API.post('/api/events', form);
      setMessage('✅ Подію додано');
    }

    setForm({
      carId: '',
      date: '',
      type: 'TO',
      cost: '',
      title: '',
      description: '',
      status: 'заплановано'
    });
    if (onEventAdded) onEventAdded();
  } catch (err) {
```

```
    console.error('❌ Помилка при збереженні події:', err);
    setMessage('❌ Помилка при збереженні');
  }
};
```

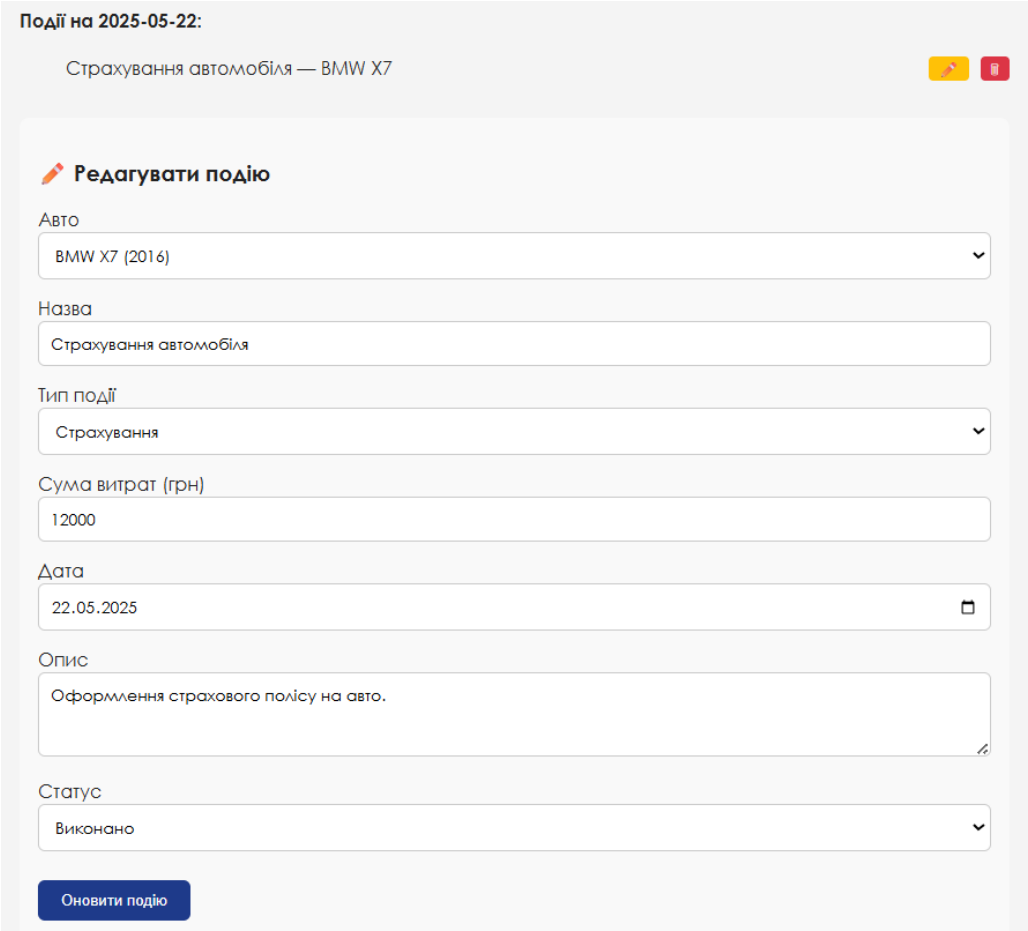

Рисунок 3.9 – Успішне оновлення

Далі при потребі можемо видалити подію з календаря, потрібно натистуни на кнопку «Видалити». Після цього запланована подія видаляється (Рис. 3.10.).

Фрагмент коду з блоку CalendarBlock:

```
<select name="carId" value={form.carId} onChange={handleChange}
required>
<option value="">— Оберіть авто —</option>
```

```
{cars.map(car => (
  <option key={car._id} value={car._id}>
    {car.brand} {car.model} ({car.year})
  </option>
))}
</select>
```

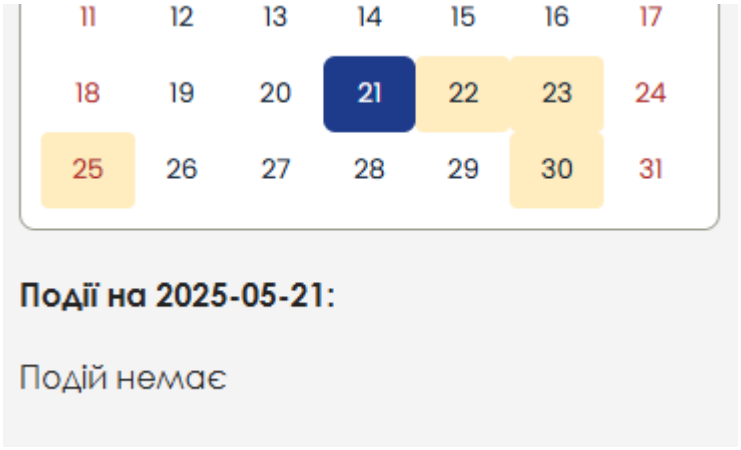

Рисунок 3.10 – Успішне видалення

Після проведення усіх робіт з додаванням і редагуванням даних усі автомобілі відображаються на сторінці «Авто» з додатковими даними про власника авто, маркою, моделлю, роком випуску та пробігом (Рис. 3.11.).

Фрагмент коду з блоку AddedCars:

```
filteredCars.map((car) => (
    <div key={car._id} style={{ border: '1px solid #ccc', borderRadius: '8px', padding: '1rem', width: '220px' }}>
      {car.image && (
        <img
          src={car.image}
          alt="auto"
          style={{ width: '100%', height: '140px', objectFit: 'cover', borderRadius: '6px' }}
        />
```

```
                )}
                <h4>{car.brand} {car.model}</h4>
                <p>{car.year} • {car.mileage} км</p>
                <p>👤 {car.ownerName}</p>
                <div style={{ display: 'flex', gap: '10px' }}>
                  <button
                    onClick={() => handleEdit(car)}
                    style={{ backgroundColor: '#007BFF', color: 'white', border:
'none', padding: '5px', borderRadius: '4px' }}
                  >
                    ✏️
                  </button>
                  <button
                    onClick={() => handleDelete(car._id)}
                    style={{ backgroundColor: '#dc3545', color: 'white', border:
'none', padding: '5px', borderRadius: '4px' }}
                  >
                    🗑️
                  </button>
                </div> </div>
              ))
```

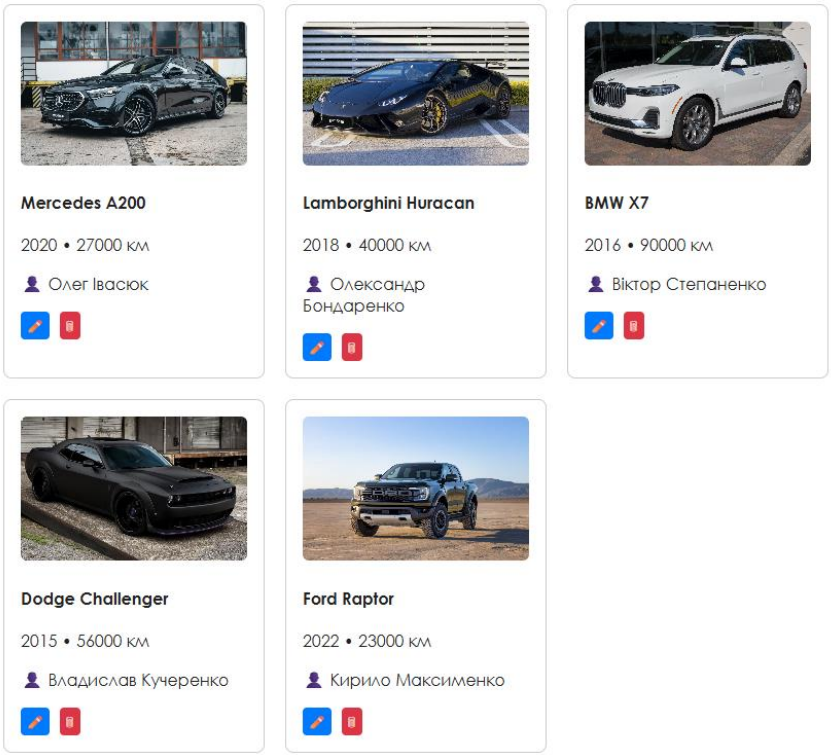

Рисунок 3.11 – Усі додані авто

3.3.4 Сторінка зі статистикою

На сторінці зі статистикою про витрати, працівник може переглянути детальну інформацію про розподіл витрат по категоріях та динаміку витрат по місяцях. Це дозволяє фірмі слідкувати за наданими роботами та фінансовими потоками (Рис. 3.12.).

Фрагмент коду зі сторінки StatsPage:

```
//  Витрати по категоріях
    const groupedByType = events.reduce((acc, ev) => {
      acc[ev.type] = (acc[ev.type]  0) + ev.cost;
      return acc;
    }, {});

    const pieData = Object.entries(groupedByType).map(([type, value]) => ({
      name: type,
```

```
  value
}));
setCategoryData(pieData);

//  Витрати по місяцях
const groupedByMonth = events.reduce((acc, ev) => {
  const month = ev.date?.slice(0, 7); // yyyy-mm
  acc[month] = (acc[month]  0) + ev.cost;
  return acc;
}, {});
```

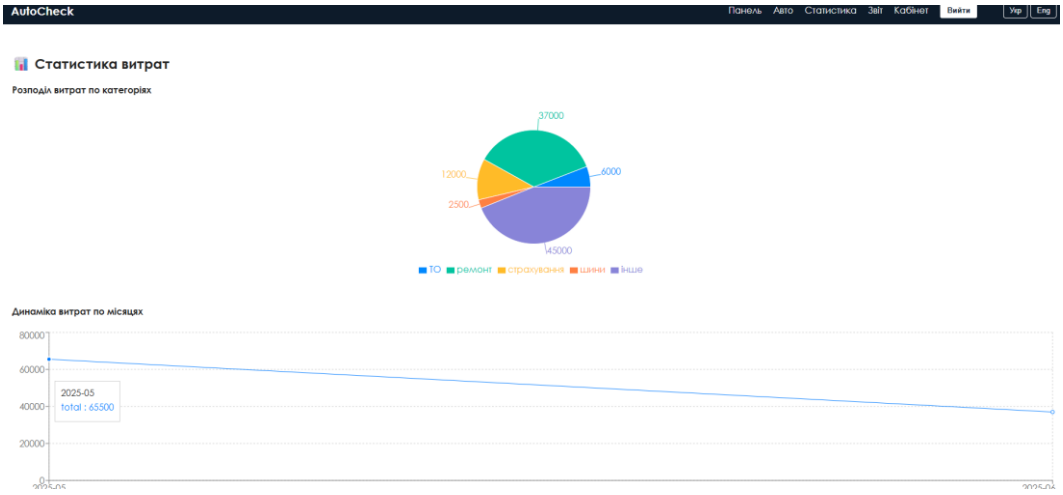

Рисунок 3.12 – сторінка зі статистикою

3.3.5 Перемикання мови інтерфесу

В дану веб-орієнтовану програму було інтегровано перемикач мови. Працівник може запросто переключити мову інтерфейсу на навігаційній панелі. Потрібно лише обрати бажану мову інтерфейсу і натиснути на кнопку (Рис. 3.13.).

Фрагмент коду з навігаційної панелі Navbar:

```
const changeLang = (lng) => {
```

```
i18n.changeLanguage(lng);

};

<div style={{ marginLeft: '20px' }}>

<button onClick={() => changeLang('uk')} style={styles.langBtn}>Укр</button>

<button onClick={() => changeLang('en')} style={styles.langBtn}>Eng</button>

</div>
```

Рисунок 3.13 – можливість зміни мови інтерфейсу

3.3.6 Редагування персональних даних

На сторінці «Кабінет» користувач бачить свої облікові дані які також може змінити. Після зміни даних потрібно натиснути «Зберегти зміни» після чого дані оновляться (Рис. 3.14.).

Фрагмент коду зі сторінки ProfilePage:

```
<div style={{ padding: '2rem' }}>
    <h2>👤 Кабінет користувача</h2>
    <form onSubmit={handleSubmit}>
      <input
        name="fullName"
```

```
  placeholder="Повне ім’я"
  value={form.fullName}
  onChange={handleChange}
  required
/>
<input
  name="email"
  placeholder="Email"
  value={form.email}
  onChange={handleChange}
  required
/>
<input
  name="phone"
  placeholder="Номер телефону"
  value={form.phone}
  onChange={handleChange}
  required
/>
<input
  name="password"
  type="password"
  placeholder="Новий пароль (необов’язково)"
  value={form.password}
  onChange={handleChange}
/>
<button type="submit" style={{ marginTop: '1rem' }}>
  Зберегти зміни
</button>
```

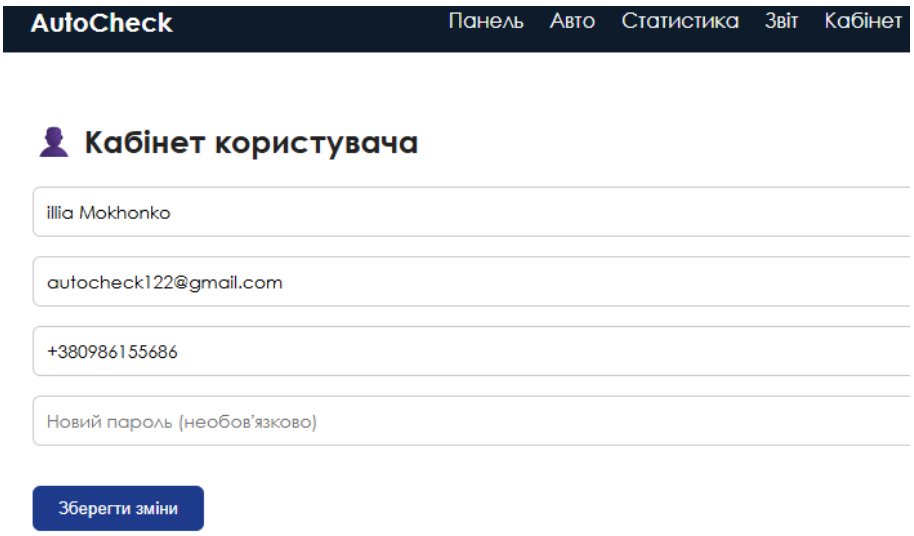

Рисунок 3.14 – Кабінет користувача

3.4 Інструкція користувача

Цей розділ створений як повноцінна інструкція для використання веб-орієнтованої системи для моніторингу технічного стану автомобілів "AutoCheck Solutions". Взаємодія з системою починається зі сторінки авторизації, де користувач має зареєструватись (Рис. 3.15.) або залогінитись (Рис. 3.16.). Валідація на цій сторінці контролює процес правильного занесення даних.

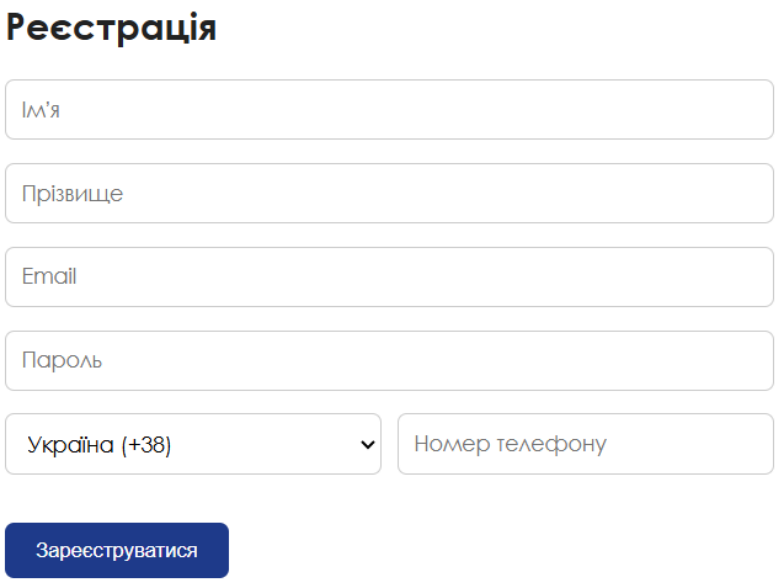

Рисунок 3.15 – Форма реєстрації

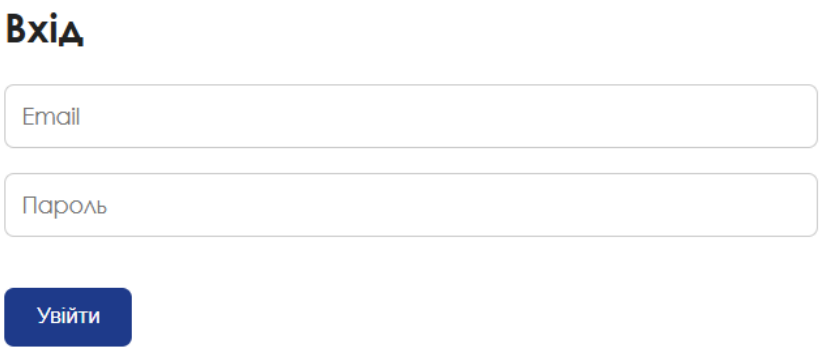

Рисунок 3.16 – Форма входу до акаунту

Після процесу верифікації користувач переходить до головної частини сайту. Зручна навігація дозволяє працівникам легко переходити на інші сторінки системи, і легко змінювати мову інтерфейсу (Рис. 3.17.). На сторінці з календарем ми бачимо заплановані події на наближчий час. Також на цій сторінці можна додати нові події або відредагувати та видалити старі (Рис. 3.18.).

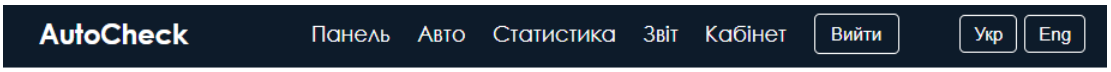

Рисунок 3.17 – Навігаційна панель

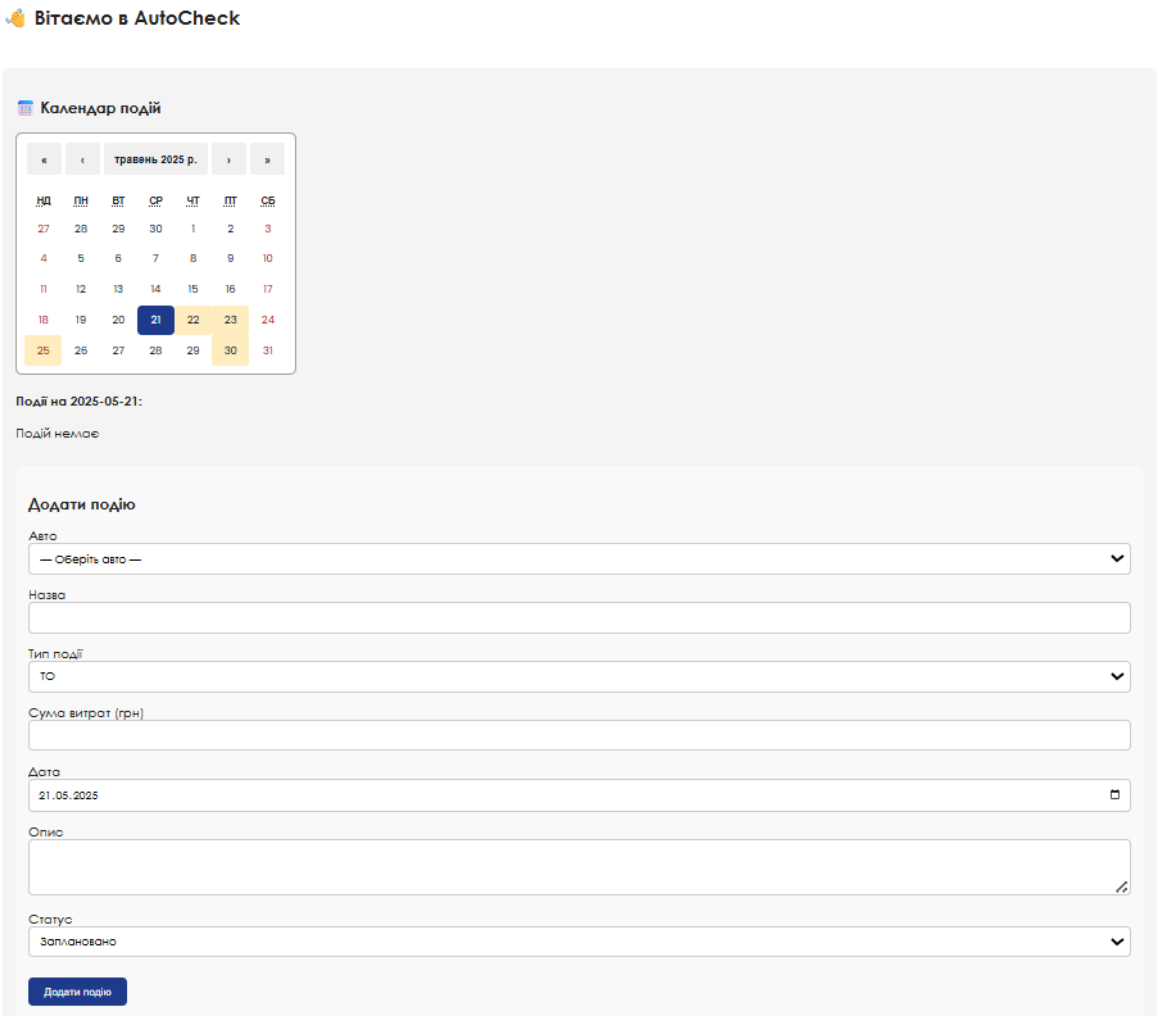

Рисунок 3.18 – Сторінка з календарем та формою

В навігації також є кнопка з переходом до секції з автомобілями. Тут ми бачимо блок з додаванням авто (Рис. 3.19.), під яким знаходиться секція завдяки якій можна виконати фільтрацію з пошуку конкретного авто. Після закінчення виконання поставлених задач з технічних робіт статус робіт змінюється на "Виконано" (Рис. 3.20.).

Рисунок 3.19 – Блок з додаванням авто

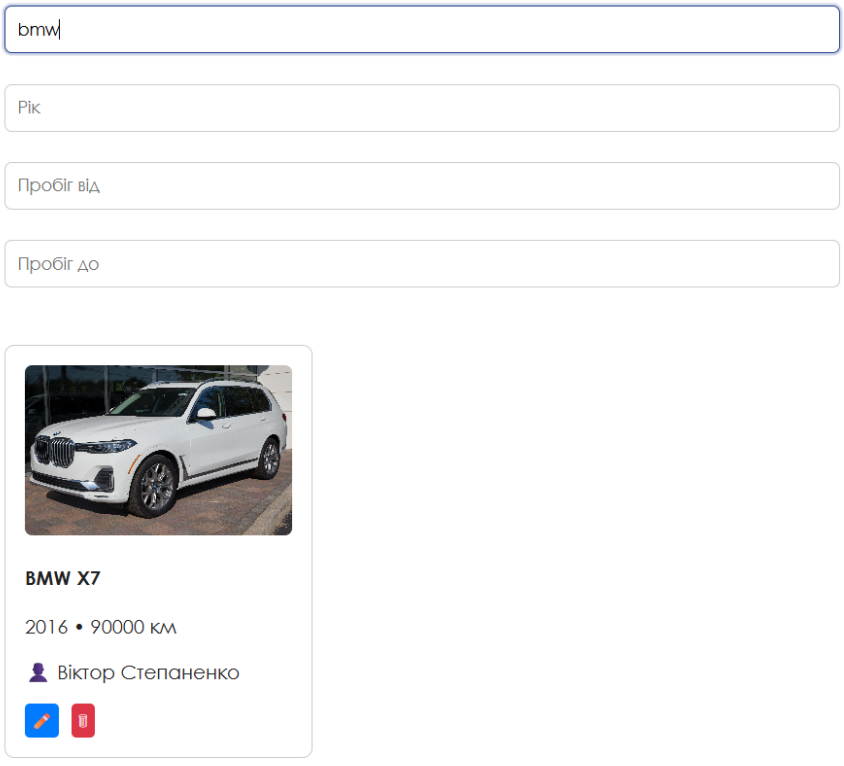

Рисунок 3.20 – Результат фільтрації за маркою авто

В нижній частині сайту знаходяться вже додані авто (Рис. 3.21.), інформацію про які також можна змінити завдяки кнопкам на панелях (Рис. 3.22.).

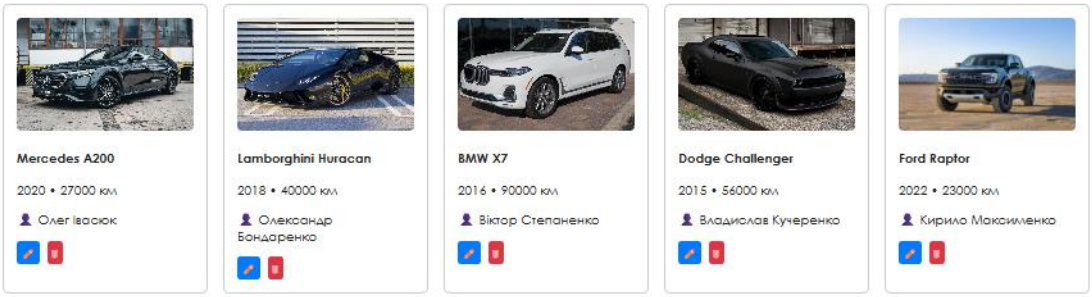

Рисунок 3.21 – додані автомобілі

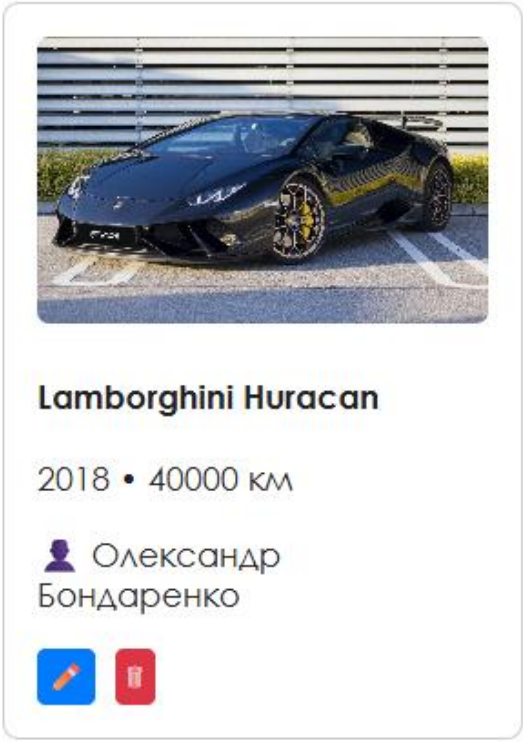

Рисунок 3.22 – кнопки з редагуванням інформації про авто

Наступною в навігації є сторінка з аналітикою. Завдяки ній можна проаналізувати детальну інформацію про розподіл витрат по категоріях (Рис. 3.23.) та динаміку витрат по місяцях (Рис. 3.24.). Це дозволяє контролювати фінанси компанії.

Рисунок 3.23 – Кругова діаграма з розподілом витрат

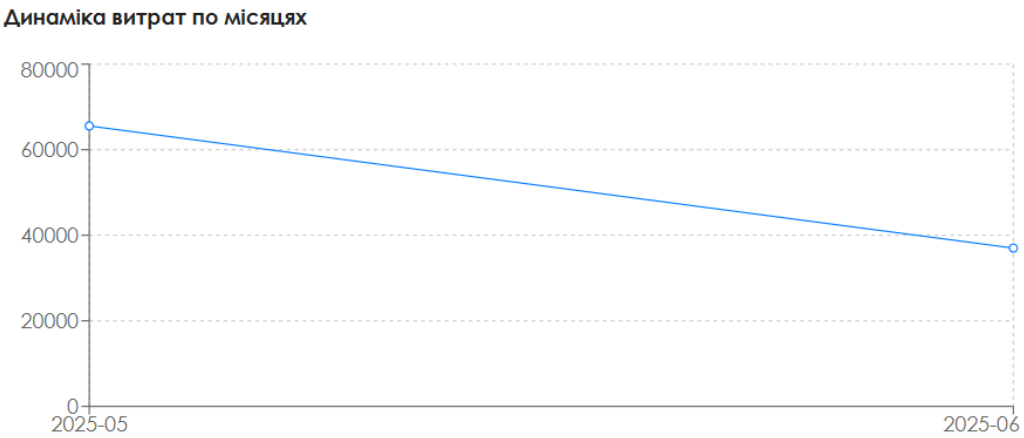

Рисунок 3.24 – Діаграма з динамікою витрат

Ще одним елементом навігації є персональний кабінет працівника. На цій сторінці можна редагувати дані про користувача. Валідація дає можливість надання правильних даних (Рис. 3.25.).

Рисунок 3.25 – Персональний кабінет користувача

Останнім але не менш важливим елементом цієї системи є завантажження звітів (Рис. 3.26.). Після переходу на сторінку "Звіт" бачимо кнопку "Завантажити Excel". Одразу після цього можемо переглянути повний звіт (Рис. 3.27.).

Рисунок 3.26 – Таблиця зі вітом

№	Назва	Тип	Дата	Сума	Авто	Власник
1	Технічний огляд двигуна	ТО	2025-05-2	6000 грн	Mercedes A200 (2020)	Олег Івасюк
2	Ремонт вихлопної системи	ремонт	2025-06-1	37000 грн	Lamborghini Huracan (2018)	Олександр Бондаренко
3	Страхування автомобіля	страхуван	2025-05-2	12050 грн	BMW X7 (2016)	Віктор Степаненко
4	Заміна шин	шини	2025-05-2	2500 грн	Dodge Challenger (2015)	Владислав Кучеренко
5	Тюнінгування авто	інше	2025-05-3	45000 грн	Ford Raptor (2022)	Кирило Максименко

Рисунок 3.27 – Результати експорту в excel

3.5 Тестування програмного продукту

3.5.1 Загальні принципи тестування

Тестування є невіддільною частиною життєвого циклу програмного забезпечення і забезпечує виявлення можливих дефектів ще до впровадження системи. У межах цього проєкту тестування мало на меті переконатися у стабільності, правильності роботи та відповідності функціональності веборієнтованої системи «AutoCheck Solutions» до технічного завдання. Особливу увагу було приділено коректності взаємодії між користувачем (працівником СТО) та основними модулями системи, включаючи додавання транспортних засобів, подій техобслуговування, фільтрацію, генерацію звітів та обробку профільних даних.

Для документування результатів тестування було використано шаблон Excel з чеклістом, у якому відображені всі протестовані функції, статус перевірки та виявлені зауваження. Тестування проводилося у популярних веббраузерах із залученням декількох користувачів на одному сервері.

3.5.2 Тестування основних функцій. Чекліст (Test Checklist)

Під час тестування перевірялися всі основні функціональні можливості, реалізовані у системі згідно технічного завдання. Зокрема, було протестовано:

- реєстрацію та авторизацію користувачів (з перевіркою валідації);
- додавання, редагування та видалення транспортних засобів з фото;
- створення та редагування подій технічного обслуговування;
- перегляд та фільтрацію подій у календарі;

- динамічне відображення графіків по витратах (Pie та Line Chart);
- формування звітів у форматі Excel;
- редагування профілю користувача;
- автоматичне оновлення статусу подій залежно від дати;
- перевірку захищеності доступу (JWT, приватні маршрути);
- адаптивність та коректність інтерфейсу у різних браузерах.

Тестове середовище:

- операційна система: Windows 11 Pro;
- база даних: MongoDB;
- фреймворк бекенду: Node.js (Express.js);
- середовище розробки: Visual Studio Code;
- фреймворк фронтенду: React.js;
- мови програмування: JavaScript (ES6+), HTML5, CSS3;
- бібліотеки: Axios[21], Recharts, jsPDF, xlsx[22], file-saver[23];
- браузери для тестування: Google Chrome (v124), Opera, Microsoft Edge;

Усі ключові функції пройшли ручне функціональне тестування, яке підтвердило працездатність системи в основних сценаріях взаємодії. Виявлені незначні помилки були усунені до завершення етапу тестування. Система визнана готовою до використання.

На даному етапі перейдемо до результатів тестування за чек-листом використовуючи відповідний шаблон в середовищі Excel:

1. Рестрація користувачів на сайті (Рис. 3.28.).

1.

Назва	Реєстрація користувача
Опис	Відкриття сторінки /register
Вимоги	
Статус	Виконано

№	Приорітет	Перевірка	Очікуваний результат	Результат перевірки		
Реєстрація користувача				Chrome	Opera	Edge
Перевірки для реєстрації						
1	високий	Відкрити сторінку реєстрації /register	Форма реєстрації відображається	Passed	Passed	Passed
2	високий	Ввести валідні ім'я, email, логін, пароль	Поля приймаються, не підсвічуються помилками	Passed	Passed	Passed
3	високий	Натиснути кнопку «Зареєструватися»	Користувач створюється в базі	Passed	Passed	Passed
4	високий	Відбувається редирект на головну сторінку	Відображаються усі блоки та календар	Passed	Passed	Passed

Рисунок 3.28 – Тестування «Реєстрація користувача на сайті»

2. Авторизація користувача на сайті (Рис. 3.29.).

2.

Назва	Авторизація користувача
Опис	Відкриття сторінки /login
Вимоги	
Статус	Виконано

№	Приорітет	Перевірка	Очікуваний результат	Результат перевірки		
Авторизація користувача				Chrome	Opera	Edge
Перевірки для авторизації						
1	високий	Відкрити сторінку входу /login	Відображається форма логіну	Passed	Passed	Passed
2	високий	Ввести правильний логін і пароль	Поля не мають помилок	Passed	Passed	Passed
3	високий	Натиснути «Увійти»	Відбувається авторизація	Passed	Passed	Passed
4	високий	Користувач перенаправляється на головну сторінку	Відображається головна сторінка	Passed	Passed	Passed
5	високий	При оновленні сторінки сесія зберігається	Користувач залишається в системі	Passed	Passed	Passed

Рисунок 3.29 – Тестування «авторизація користувача на сайті»

3.Додавання та редагування даних стосовно автомобіля (Рис. 3.30.).

3.

Назва	Додавання та редагування даних стосовно автомобіля
Опис	Перевірка додавання даних до системи
Вимоги	
Статус	Виконано

№	Приорітет	Перевірка	Очікуваний результат	Результат перевірки		
Додавання авто в систему та створення події в календарі				Chrome	Opera	Edge
Перевірки для додавання авто в систему та створення події в календарі						
1	високий	Перейти на сторінку додавання авто	Завантажується форма для заповнення	Passed	Passed	Passed
2	високий	Заповнити усі необхідні поля	Інформація підходить під опис	Passed	Passed	Passed
3	високий	Натиснути кнопку «Додати авто»	Авто відображається на сторінці	Passed	Passed	Passed
4	високий	редагування даних про авто після додавання	інформація редагується та зберігається успіш	Passed	Passed	Passed

Рисунок 3.30 – Тестування «Додавання та редагування даних стосовно автомобіля»

4. Фільтрація авто за маркою (Рис. 3.31.).

4.

Назва	Фільтрація авто за маркою
Опис	Перевірка роботи фільтрів на сторінці додавання авто
Вимоги	
Статус	Виконано

№	Приорітет	Перевірка	Очікуваний результат	Результат перевірки		
Додавання авто в систему та створення події в календарі				Chrome	Opera	Edge
Перевірки для додавання авто в систему та створення події в календарі						
1	середній	Відкрити сторінку /cars	Виводиться повний список доданих авто	Passed	Passed	Passed
2	середній	Обрати марку (наприклад, «BMW»)	Список машин автоматично оновлюється	Passed	Passed	Passed
3	середній	Перевірити, що лишились авто відповідної марки	У кожного авто марка = «BMW»	Passed	Passed	Passed
4	середній	Обрати іншу марку (наприклад, «Ford»)	Список оновлюється повторно	Passed	Passed	Passed
5	середній	Очистити фільтр	Відображаються всі доступні автомобілі	Passed	Passed	Passed

Рисунок 3.31 – Тестування «Фільтрація авто за маркою»

5. Додавання події в календар (Рис. 3.32.).

5.

Назва	Додавання події в календар
Опис	Перевірка функції додавання події в календар
Вимоги	
Статус	Виконано

№	Приорітет	Перевірка	Очікуваний результат	Результат перевірки		
Додавання авто в систему та створення події в календарі				Chrome	Opera	Edge
Перевірки для додавання авто в систему та створення події в календарі						
1	високий	Перейти до сторінки /dashboard	Відображається календар з подіями	Passed	Passed	Passed
2	високий	Перейти до форми для додавання подій	Форма працює і готова до заповнення	Passed	Passed	Passed
3	високий	Заповнення форми	Форма заповнюється успішно	Passed	Passed	Passed
4	високий	Натиснути кнопку яка додає дані в базу	Дані завантажились успішно	Passed	Passed	Passed
5	високий	Перевірки чи додалась нова подія до календаря	Подія відображається в календарі	Passed	Passed	Passed

Рисунок 3.32 – Тестування «Додавання події в календар»

3.5.3 Оцінка результатів тестування

У межах тестування інформаційної системи "AutoCheck Solutions" було перевірено ключові функції, реалізовані у програмному продукті. Зокрема, проведено перевірку таких можливостей: реєстрація та авторизація користувачів; додавання й редагування даних про автомобілі; фільтрація транспортних засобів за маркою; створення подій у календарі технічного обслуговування.

Тестування проводилось у популярних браузерах — Google Chrome, Microsoft Edge та Opera. У всіх випадках система показала стабільну роботу: сторінки коректно завантажувались, інтерактивні елементи функціонували без збоїв, дані успішно оброблялися і записувалися до бази.

У тесті на реєстрацію система правильно обробляє введення даних, реалізована перевірка введеної інформації (валідація), після чого користувач автоматично перенаправляється до кабінету. Авторизація перевіряє правильність даних та реалізує контроль доступу до захищених сторінок.

При додаванні автомобіля перевірялися валідація введених значень (рік, пробіг, марка, модель, ім’я власника) та можливість прикріплення зображення. Редагування працює без помилок, зміни зберігаються.

Фільтрація авто за назвою (маркою) виконується у реальному часі, з урахуванням нечутливості до регістру. Тест на календар показав, що подія

створюється із зазначенням дати, назви, витрат, типу та опису, а також пов’язується з конкретним авто.

Усі перевірки підтвердили, що система працює відповідно до технічного завдання, забезпечуючи повний цикл взаємодії з базою даних, коректний розподіл доступу та зручний інтерфейс для користувачів. Продукт виявився стабільним, функціональним і готовим до впровадження в реальні умови експлуатації.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було поставлено основну мету – спроєктувати та реалізувати веборієнтовану інформаційну систему для моніторингу технічного стану автомобілів "AutoCheck Solutions", яка дозволяє працівникам станцій технічного обслуговування додавати транспортні засоби клієнтів, фіксувати події з обслуговування, зберігати історію технічних робіт і формувати звіти. Поставлену мету було досягнуто, а ключові завдання – виконано.

Розроблена система успішно реалізовує набір важливих функцій, серед яких: реєстрація та авторизація користувачів (працівників СТО); додавання, редагування та видалення автомобілів з фото; створення подій обслуговування з описом, вартістю, датою, типом і статусом; інтегрований календар з подіями; перегляд і фільтрація записів за критеріями; статистика витрат у вигляді графіків; формування звітів у форматі PDF та Excel; редагування профілю користувача.

Система була розроблена з використанням сучасних вебтехнологій, зокрема Node.js (Express) для серверної частини, MongoDB як системи керування базами даних, React.js для клієнтської частини, а також бібліотек Recharts, jspdf, file-saver та xlsx для візуалізації та експорту даних. Вона протестована та повністю функціонує в умовах локального середовища, з перспективою подальшого розгортання на віддаленому сервері.

Очікується, що впровадження розробленої системи у роботу автосервісу дозволить централізувати облік технічного обслуговування, підвищити точність ведення записів, автоматизувати обробку подій, спростити генерацію звітності та покращити комунікацію між працівниками і клієнтами.

Проєкт реалізовано в середовищі Visual Studio Code. Усі компоненти взаємодіють через REST API, структура бази даних побудована на основі схем Mongoose.

Отриманий результат є повноцінним вебзастосунком і може бути базою для розширення системи: розробки мобільної версії, інтеграції з GPS або обліком запасних частин, а також впровадження багаторівневого доступу з розмежуванням прав.

Рекомендовано використовувати систему "AutoCheck Solutions" для автоматизації роботи малого або середнього СТО, оскільки вона є гнучкою, зручною у використанні та адаптованою до реальних потреб галузі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методичні рекомендації до виконання кваліфікаційної роботи на здобуття освітнього ступеня «бакалавр» спеціальності 122 «Комп'ютерні науки» [Електрон. ресурс] / уклад. С. В. Грибков, Н. В. Ліманська. – К.: НУХТ, 2025. – 43 с.

2. Шевченко, О. В. Сучасний стан і тенденції розвитку транспортної інфраструктури в Україні / О. В. Шевченко // Економіка і суспільство. – 2023. – № 49. – С. 132–139.

3. Гриненко А. І., Панченко О. І. Економіка автомобільного сервісу : підручник. – Харків : Факт, 2020. – 240 с.

4. Simply Auto. Vehicle Management App [Електронний ресурс]. – Режим доступу: https://www.simplyauto.app/ (дата звернення: 21.05.2025).

5. Fleetio. Fleet Maintenance Software [Електронний ресурс]. – Режим доступу: https://www.fleetio.com/ (дата звернення: 21.05.2025).

6. Chevin FleetWave. Fleet Management System [Електронний ресурс]. – Режим доступу: https://www.chevinfleet.com/ (дата звернення: 21.05.2025).

7. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE). – Geneva: ISO, 2011.

8. Література з аналізу інформаційних систем: збірник наукових праць / за ред. І. І. Даниленка. – Київ : КПІ ім. Ігоря Сікорського, 2019. – 224 с.

9. Чучаєв А. П. Управління IT-проектами. – Київ : КНУ, 2019. – 320 с.

10. ДСТУ 3973:2000. Система розроблення та поставлення продукції на виробництво. – Київ : Держстандарт України, 2001. – 14 с.

11. ДСТУ Б В.2.5–82:2016. Електробезпека в будівлях і спорудах. – Київ : Мінрегіон України, 2016. – 35 с.

12. ISO/IEC 27001:2022. Information security management systems — Requirements. – Geneva : ISO, 2022.

13. Шевченко В. І. Проектування інформаційних систем : підручник. – Харків : ХНУРЕ, 2021. – 280 с.

14. JavaScript. MDN Web Docs [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/ (дата звернення: 21.05.2025).

15. React documentation [Електронний ресурс]. – Режим доступу: https://reactjs.org/docs/ (дата звернення: 21.05.2025).

16. Node.js documentation [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/docs/ (дата звернення: 21.05.2025).

17. MongoDB documentation [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/docs/ (дата звернення: 21.05.2025).

18. JWT Authentication. Auth0 Docs [Електронний ресурс]. – Режим доступу: https://auth0.com/docs/ (дата звернення: 21.05.2025).

19. Express.js Guide [Електронний ресурс]. – Режим доступу: https://expressjs.com/ (дата звернення: 21.05.2025).

20. Axios. Бібліотека для HTTP-запитів [Електронний ресурс]. – Режим доступу: https://axios-http.com/ (дата звернення: 21.05.2025).

21. XLSX. SheetJS Docs [Електронний ресурс]. – Режим доступу: https://sheetjs.com/ (дата звернення: 21.05.2025).

22. FileSaver.js Documentation [Електронний ресурс]. – Режим доступу: https://github.com/eligrey/FileSaver.js/ (дата звернення: 21.05.2025).

23. Ханенко Н. С. Основи баз даних : навч. посіб. – Дніпро : НГУ, 2019. – 185 с.

24. Жук Ю. О. Основи HTML, CSS та JavaScript : навч. посіб. – Київ : Ліра-К, 2022. – 198 с.

25. Bootstrap documentation [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/ (дата звернення: 21.05.2025).

26. MySQL Workbench: Офіційна документація [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/workbench/en/ (дата звернення: 21.05.2025).

27. Canva. Онлайн-сервіс для створення діаграм [Електронний ресурс]. – Режим доступу: https://www.canva.com/ (дата звернення: 21.05.2025).

28. Канер, С. Тестування програмного забезпечення: фундаментальні принципи : пер. з англ. – Львів : БаК, 2017. – 320 с.

29. Шевченко, І. В. Якість програмного забезпечення та тестування : навч. посіб. – Харків : ХНУРЕ, 2022. – 145 с.

30. Онлайн-сервіс для перевірки протоколу обов’язкового технічного контролю транспортних засобів [Електронний ресурс]. – Режим доступу: https://hsc.gov.ua/2024/03/20/onlajn-servis-dlya-perevirki-protokolu-obov-yazkovogo-tehnichnogo-kontrolyu-transportnih-zasobiv (дата звернення: 21.05.2025).

ДОДАТКИ

Додаток А. Схема архітектури системи

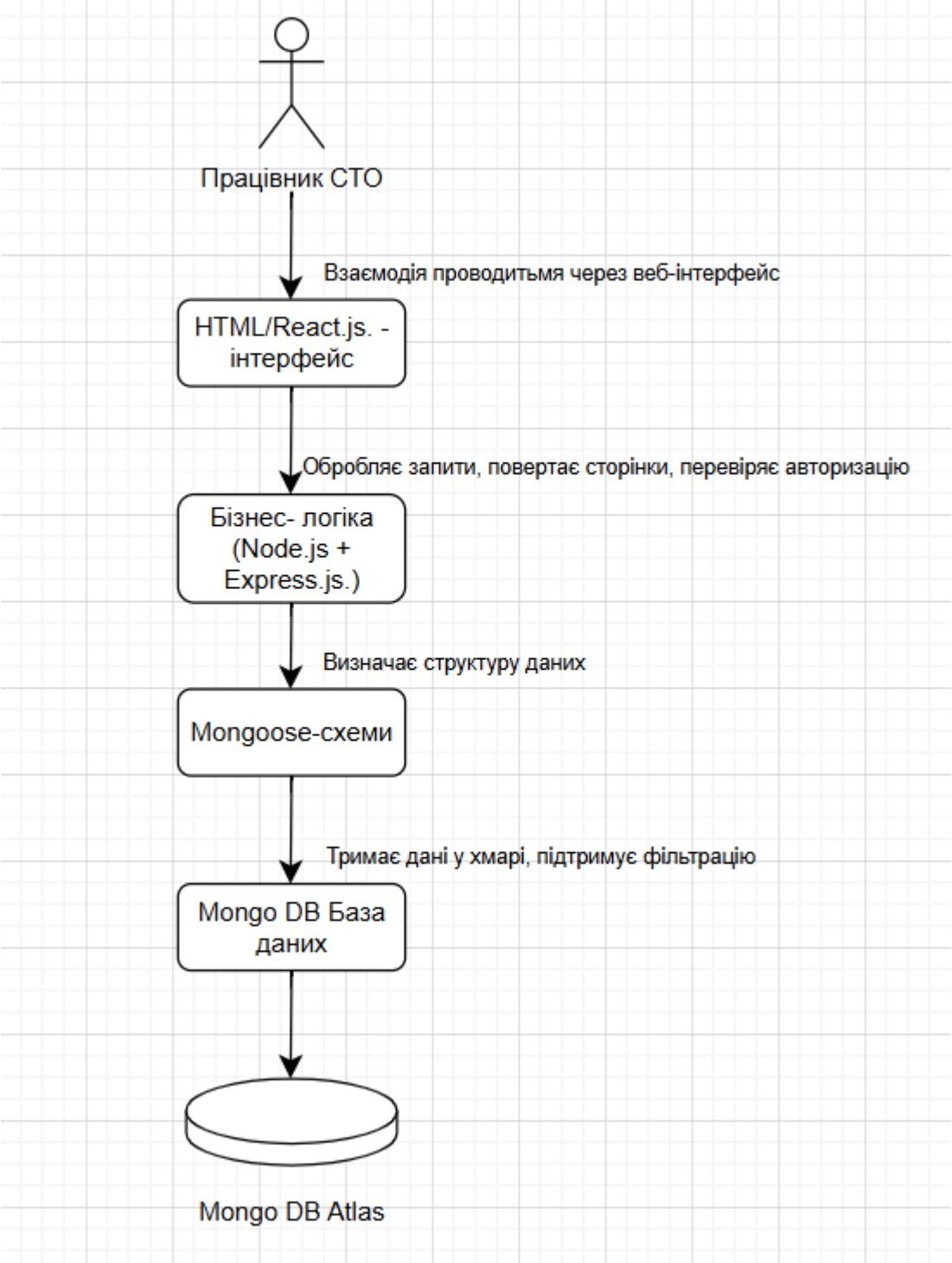

Рисунок А.1 – Схема архітектури системи

Додаток Б. UML діаграма використання системи

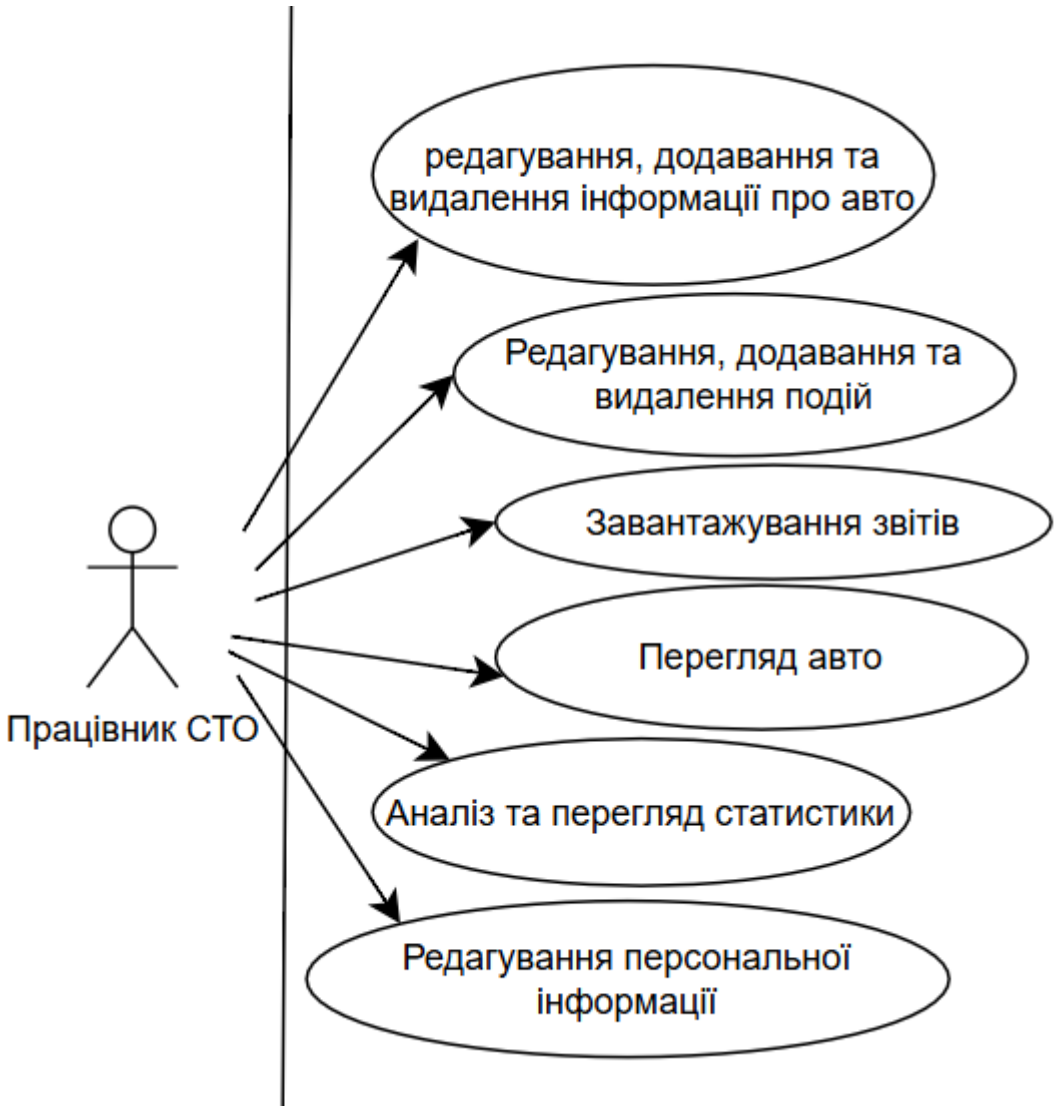

Рисунок Б.1 – UML діаграма використання системи

Додаток В. Діаграма Ганта

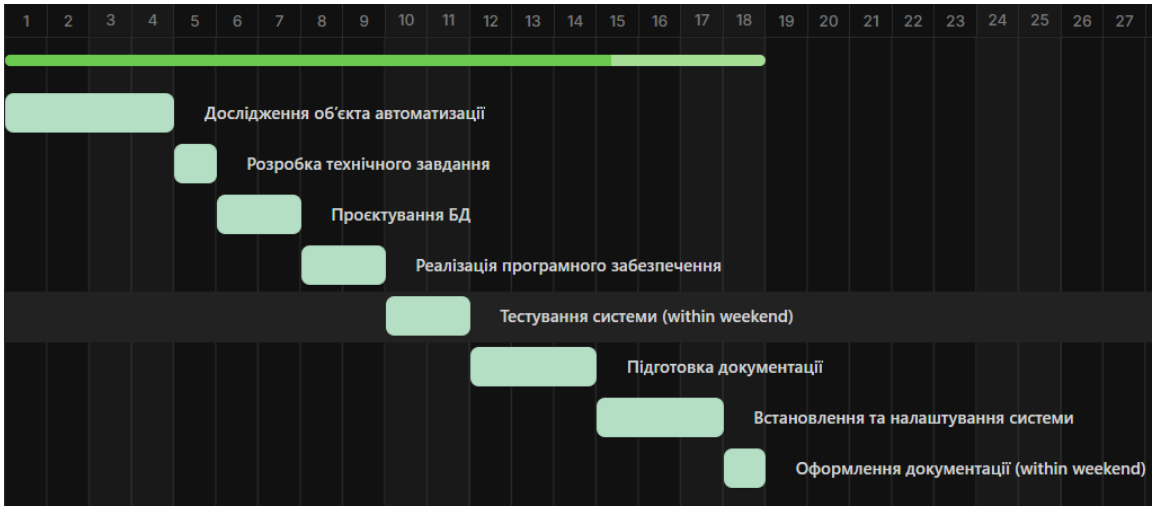

Рисунок В.1 – діаграма Ганта

Додаток Г. Фрагменти коду програми

Код Г.1. ReportPage:

```
import React, { useEffect, useState } from 'react';
import API from '../services/axios';
import * as XLSX from 'xlsx';
import { saveAs } from 'file-saver';

const ReportPage = () => {
  const [events, setEvents] = useState([]);

  useEffect(() => {
    API.get('/api/events')
      .then(res => setEvents(res.data))
      .catch(err => console.error('❌ Помилка при завантаженні подій:', err));
  }, []);

  const exportToExcel = () => {
    const data = events.map((e, i) => ({
      '№': i + 1,
      'Назва': e.title,
      'Тип': e.type,
      'Дата': e.date?.slice(0, 10),
      'Сума': ${e.cost} грн,
      'Авто': e.carId ? ${e.carId.brand} ${e.carId.model} (${e.carId.year}) : '—',
      'Власник': e.carId?.ownerName  '—'
    }));
```

```
    const worksheet = XLSX.utils.json_to_sheet(data);
    const workbook = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(workbook, worksheet, 'Звіт');

    const excelBuffer = XLSX.write(workbook, {
      bookType: 'xlsx',
      type: 'array'
    });

    const blob = new Blob([excelBuffer], {
      type: 'application/octet-stream'
    });
    saveAs(blob, 'AutoCheck_Report_All_Cars.xlsx');
  };

  return (
    <div style={{ padding: '2rem' }}>
      <h2>📋 Звіт по всіх автомобілях</h2>

      {events.length === 0 ? (
        <p>Немає подій для формування звіту.</p>
      ) : (
        <>
          <table style={{ width: '100%', borderCollapse: 'collapse', marginTop:
'1rem' }}>
            <thead>
              <tr style={{ backgroundColor: '#f0f0f0' }}>
                <th style={cellStyle}>Назва</th>
                <th style={cellStyle}>Тип</th>
                <th style={cellStyle}>Дата</th>
```

```
                <th style={cellStyle}>Сума</th>
                <th style={cellStyle}>Авто</th>
                <th style={cellStyle}>Власник</th>
              </tr>
            </thead>
            <tbody>
              {events.map((e) => (
                <tr key={e._id}>
                  <td style={cellStyle}>{e.title}</td>
                  <td style={cellStyle}>{e.type}</td>
                  <td style={cellStyle}>{e.date?.slice(0, 10)}</td>
                  <td style={cellStyle}>{e.cost} грн</td>
                  <td style={cellStyle}>
                    {e.carId ? `${e.carId.brand} ${e.carId.model} (${e.carId.year})` : '—'}
                  </td>
                  <td style={cellStyle}>{e.carId?.ownerName || '—'}</td>
                </tr>
              ))}
            </tbody>
          </table>

          <button onClick={exportToExcel} style={{ marginTop: '1rem' }}>
            📥 Завантажити Excel
          </button>
        </>
      )}
    </div>
  );
};
```

```
const cellStyle = {
  padding: '8px',
  border: '1px solid #ccc'
};


export default ReportPage;
```

Код Г.2. ProfilePage:

```
import React, { useEffect, useState } from 'react';
import API from '../services/axios';
import { jwtDecode } from 'jwt-decode';

const ProfilePage = () => {
  const [form, setForm] = useState({
    fullName: '',
    email: '',
    phone: '',
    password: ''
  });
  const [error, setError] = useState('');
  const [success, setSuccess] = useState('');

  useEffect(() => {
    const token = localStorage.getItem('token');
    if (token) {
      const decoded = jwtDecode(token);
      setForm({
        fullName: decoded.fullName  '',
```

```
        email: decoded.email  ",
        phone: decoded.phone  ",
        password: "
      });
    }
  }, []);

  const validate = () => {
    const { fullName, email, phone, password } = form;

    if (!fullName  fullName.split(' ').length < 2 || fullName.split(' ').some(n => n.length < 2)) {
      return 'Ім’я та прізвище мають містити принаймні по 2 символи';
    }

    if (!/\S+@\S+\.\S+/.test(email)) {
      return 'Введіть коректний email';
    }

    if (!/^\+\d{6,15}$/.test(phone)) {
      return 'Телефон має містити лише цифри та починатися з коду країни, напр. +380...';
    }

    if (password && password.length < 6) {
      return 'Пароль має містити щонайменше 6 символів';
    }

    return '';
  };
```

```
const handleChange = (e) => {
  setForm(prev => ({ ...prev, [e.target.name]: e.target.value }));
};

const handleSubmit = async (e) => {
  e.preventDefault();
  setError('');
  setSuccess('');

  const validationMsg = validate();
  if (validationMsg) {
    setError(validationMsg);
    return;
  }

  try {
    await API.put('/api/users/update', form);
    setSuccess('✅ Профіль оновлено');
    setForm({ ...form, password: '' });
  } catch (err) {
    setError('❌ Помилка при оновленні');
  }
};

return (
  <div style={{ padding: '2rem' }}>
    <h2>👤 Кабінет користувача</h2>
    <form onSubmit={handleSubmit}>
      <input
```

```
  name="fullName"
  placeholder="Повне ім’я"
  value={form.fullName}
  onChange={handleChange}
  required
/>
<input
  name="email"
  placeholder="Email"
  value={form.email}
  onChange={handleChange}
  required
/>
<input
  name="phone"
  placeholder="Номер телефону"
  value={form.phone}
  onChange={handleChange}
  required
/>
<input
  name="password"
  type="password"
  placeholder="Новий пароль (необов’язково)"
  value={form.password}
  onChange={handleChange}
/>
<button type="submit" style={{ marginTop: '1rem' }}>
  Зберегти зміни
</button>
```

```
            </form>
            {error && <p style={{ color: 'red', marginTop: '10px' }}>{error}</p>}
            {success && <p style={{ color: 'green', marginTop: '10px' }}>{success}</p>}
          </div>
        );
      };

      export default ProfilePage;
```

Код Г.3. AddServicePage:

```
      import React, { useEffect, useState } from 'react';
      import API from '../services/axios';
      import {
        PieChart, Pie, Cell, Tooltip, Legend, ResponsiveContainer,
        LineChart, Line, XAxis, YAxis, CartesianGrid
      } from 'recharts';
      import { useTranslation } from 'react-i18next';

const COLORS = ['#0088FE', '#00C49F', '#FFBB28', '#FF8042', '#8884D8'];

      const StatsPage = () => {
        const { t } = useTranslation();
        const [categoryData, setCategoryData] = useState([]);
        const [monthlyData, setMonthlyData] = useState([]);

        useEffect(() => {
          const fetchData = async () => {
            try {
              const res = await API.get('/api/events');
```

```
      const events = res.data;

      // 📊 Group by type
      const groupedByType = events.reduce((acc, ev) => {
        acc[ev.type] = (acc[ev.type]  0) + ev.cost;
        return acc;
      }, {});

  const pieData = Object.entries(groupedByType).map(([type, value]) => ({
name: t(`types.${type}`, { defaultValue: type }), // fallback if no translation
        value
      }));
      setCategoryData(pieData);

      // 📈 Group by month
      const groupedByMonth = events.reduce((acc, ev) => {
        const month = ev.date?.slice(0, 7); // yyyy-mm
        acc[month] = (acc[month]  0) + ev.cost;
        return acc;
      }, {});

const lineData = Object.entries(groupedByMonth).map(([date, total]) => ({
        date,
        total
      }));

      setMonthlyData(lineData);
    } catch (err) {
      console.error(t('error.fetchStats'), err);
    }
```

```
  };

  fetchData();
}, [t]);

return (
  <div style={{ padding: '2rem' }}>
    <h2>{t('stats.pageTitle')}</h2>

    {/* Pie Chart */}
    <div style={{ marginBottom: '3rem' }}>
      <h4>{t('stats.categoryDistribution')}</h4>
      {categoryData.length === 0 ? (
        <p>{t('stats.noData')}</p>
      ) : (
        <ResponsiveContainer width="100%" height={300}>
          <PieChart>
            <Pie
              data={categoryData}
              dataKey="value"
              nameKey="name"
              outerRadius={100}
              label
            >
              {categoryData.map((_, index) => (
            <Cell key={index} fill={COLORS[index % COLORS.length]} />
              ))}
            </Pie>
            <Tooltip />
            <Legend />
```

```
            </PieChart>
          </ResponsiveContainer>
        )}
      </div>

      {/* Line Chart */}
      <div>
        <h4>{t('stats.monthlyTrend')}</h4>
        {monthlyData.length === 0 ? (
          <p>{t('stats.noData')}</p>
        ) : (
          <ResponsiveContainer width="100%" height={300}>
            <LineChart data={monthlyData}>
              <CartesianGrid strokeDasharray="3 3" />
              <XAxis dataKey="date" />
              <YAxis />
              <Tooltip />
              <Line type="monotone" dataKey="total" stroke="#007BFF" />
            </LineChart>
          </ResponsiveContainer>
        )}
      </div>
    </div>
  );
};

export default StatsPage;
```