

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
АВТОСЕРВІСУ
DEVELOPMENT OF CAR SERVICE SOFTWARE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-211сп
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ЛПЕЙКО П.В.

(ПРИЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЛПЕЙКА Павла Вікторовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програмного забезпечення для автосервісу.
Development of Car Service Software.

керівник проєкту (роботи) к.т.н., доцент, СТЕПАНЕНКО Олександр Олексійович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 24 ” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно
розробити) 1. Аналіз проблеми та постановка завдань .
2. Матеріали і методи розробки програмного забезпечення. 3. Опис програмного
забезпечення. 4. Тестування програмного забезпечення.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість
слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст.викладач		

7. Дата видачі завдання “ 15 ” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Павло ЛІПЕЙКО
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Олександр СТЕПАНЕНКО
(ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 76 с., 13 табл., 34 рис., 2 дод., 24 джерел.

АВТОСЕРВІС, ВЕБСАЙТ, ПРОЕКТУВАННЯ, РОЗРОБКА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, PHP, LARAVEL, REACT, MYSQL, MVC, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, SCRUM.

Об'єкт дослідження – програмне забезпечення, яке інтегрується з внутрішніми системами управління автосервісу та забезпечує взаємодію з клієнтами.

Предмет дослідження – процеси розробки веб орієнтованих систем для автосервісу.

Мета роботи – розробка вебсайту для автосервісу, який би задовольняв потреби сучасних автовласників у швидкому та ефективному сервісі.

Матеріали, методи та технічні засоби: методології розробки програмного забезпечення (Scrum), клієнт-серверна архітектура, шаблон проектування MVC, мови програмування (PHP, JavaScript), фреймворк Laravel, бібліотека React, система управління базами даних MySQL, локальний веб-сервер Open Server.

Результати. Розроблено повнофункціональний вебсайт автосервісу з можливостями перегляду інформації про послуги, онлайн-запису на ремонт та подання заявок на підбір запчастин.

Висновки. У даній дипломній роботі успішно розроблено програмне забезпечення для автосервісу, яке відповідає сучасним вимогам ринку та потребам клієнтів.

Галузь використання – автосервісні підприємства.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 76 pages, 13 tables, 34 figures, 2 appendices, 24 sources.

CAR SERVICE, WEBSITE, DESIGN, DEVELOPMENT, SOFTWARE, PHP, LARAVEL, REACT, MYSQL, MVC, CLIENT-SERVER ARCHITECTURE, SCRUM.

The object of study - software that integrates with internal management systems of a car service and ensures interaction with clients.

The subject of study - the processes of developing web-oriented systems for a car service.

The purpose of the work - development of a website for a car service that would satisfy the needs of modern car owners in fast and efficient service.

Materials, methods, and technical means: software development methodologies (Scrum), client-server architecture, MVC design pattern, programming languages (PHP, JavaScript), Laravel framework, React library, MySQL database management system, local web server Open Server.

Results. A fully functional car service website has been developed with the capabilities of viewing information about services, online booking for repairs, and submitting requests for the selection of spare parts.

Conclusions. In this diploma thesis, software for a car service has been successfully developed that meets modern market requirements and client needs.

Field of use - car service enterprises.

ЗМІСТ

	С.
Перелік скорочень та умовних познач 7	7
Вступ..... 8	8
1 Аналіз проблеми та постановка завдань 10	10
1.1 Аналіз предметної області 10	10
1.2 Аналіз вебзастосунків автосервісу 11	11
1.3 Вимоги до розробки програмного забезпечення 17	17
1.4 Висновки до розділу 1 20	20
2 Матеріали і методи розробки програмного забезпечення 22	22
2.1 Архітектура програмного забезпечення..... 22	22
2.2 Методи розробки програмного забезпечення 25	25
2.3 Аналіз технологій розробки програмного забезпечення..... 27	27
2.4 Висновки до розділу 30	30
3 Опис програмного забезпечення 31	31
3.1 Проєктування та розробка бази даних 31	31
3.2 Розробка програмного коду 33	33
3.3 Користування вебсайтом..... 38	38
3.4 Висновки до розділу 47	47
4 Тестування програмного забезпечення 49	49
4.1 Тестування функціоналу вебсайту 49	49
4.2 Тестування бази даних вебсайту 52	52
4.3 Висновки до розділу 54	54
Висновки..... 56	56
Перелік джерел посилення 58	58
Додаток А Текст програми 61	61
Додаток Б Слайди презентації..... 70	70

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	- Application Programming Interface;
AOP	- Aspect-Oriented Programming;
AWS	- Amazon Web Services;
CRUD	- Create, Read, Update, Delete;
CRM	- Customer Relationship Management;
ERP	- Enterprise Resource Planning;
HTTPS	- Hypertext Transfer Protocol Secure;
IoC	- Inversion of Control;
MVC	- Model-View-Controller;
MySQL	- My Structured Query Language;
ORM	- Object-Relational Mapping;
PHP	- PHP: Hypertext Preprocessor;
SCRUM	- Sustainable Evolutionary Repeatable Unified Modular;
SQL	- Structured Query Language;
UML	- Unified Modeling Language;
UI	- User Interface;
UX	- User Experience;
XP	- Extreme Programming;
XSS	- Cross-Site Scripting;
СУБД	- Система управління базами даних.

ВСТУП

Розвиток інформаційних технологій останніми десятиліттями значно вплинув на всі сфери людської діяльності, включаючи автомобільний сервіс. Сучасні ІТ-технології відіграють ключову роль у підвищенні ефективності, доступності та якості обслуговування в автосервісних підприємствах. Розробка програмного забезпечення для автосервісу, яке дозволяє користувачам переглядати інформацію про послуги, записуватися на ремонт та залишати заявки на підбір запчастин через вебінтерфейс, є актуальним завданням, яке відповідає сучасним вимогам ринку [1].

Впровадження ІТ-технологій в автосервіс значно оптимізує процеси управління, логістики та клієнтського сервісу. Використання CRM-систем, мобільних додатків, ERP-систем, а також платформ для онлайн-бронювання і замовлення деталей, дозволяє підвищити зручність для клієнтів та ефективність роботи сервісних центрів. Цифрові технології, такі як штучний інтелект і машинне навчання, вже використовуються для аналізу великих даних, діагностики стану автомобілів та оптимізації запасів запчастин [2].

Метою даної дипломної роботи є розробка вебсайту для автосервісу, який би задовольняв потреби сучасних автовласників у швидкому та ефективному сервісі. Вебсайт повинен включати функціональні можливості для перегляду інформації про послуги, онлайн-запис на ремонт та систему заявок на підбір запчастин.

Предмет дослідження - процеси розробки веборієнтованих систем для автосервісу. Об'єкт дослідження - програмне забезпечення, яке інтегрується з внутрішніми системами управління автосервісу та забезпечує взаємодію з клієнтами.

Задачі, які треба вирішити:

- аналіз потреб користувачів та формування технічних вимог до програмного забезпечення;

- проектування архітектури вебсайту, забезпечення його масштабованості та безпеки;
- розробка функціоналу для перегляду інформації про послуги, запису на ремонт та подання заявок на деталі;
- тестування та оптимізація вебсайту для забезпечення високої продуктивності та користувацького досвіду.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ

1.1 Аналіз предметної області

Розробка програмного забезпечення для автосервісу є важливим кроком у напрямку підвищення ефективності та якості обслуговування клієнтів. В сучасному світі, де технології відіграють ключову роль у багатьох сферах життя, автосервіси не можуть залишатися осторонь цього процесу. Створення вебсайту, який надає клієнтам можливість ознайомитися з послугами автосервісу, записатися на ремонт та залишити заявку на підбір деталей, значно спрощує процес взаємодії між клієнтами та автосервісом [1].

Існуючі програмні рішення для автосервісів часто мають ряд недоліків, які ускладнюють їх використання та знижують ефективність роботи. Серед таких проблем можна виділити застарілий дизайн, відсутність зручної навігації, обмежений функціонал та недостатню інтеграцію з іншими системами. Ці недоліки призводять до того, що клієнти витрачають більше часу на пошук необхідної інформації та оформлення заявок, а співробітники автосервісу змушені працювати з неефективними інструментами.

Розробка нового програмного забезпечення для автосервісу дозволить вирішити ці проблеми та надати клієнтам і співробітникам зручний та функціональний інструмент. Сучасний дизайн та інтуїтивно зрозумілий інтерфейс дозволять клієнтам легко орієнтуватися на сайті та швидко знаходити потрібну інформацію. Можливість онлайн-запису на ремонт та подання заявки на підбір деталей значно спростить процес взаємодії з автосервісом та заощадить час клієнтів. Для співробітників автосервісу нове програмне забезпечення надасть можливість ефективніше керувати завданнями, відстежувати статус ремонту та оперативно обмінюватися інформацією з клієнтами [2].

Серед переваг розробки нового програмного забезпечення для автосервісу можна виділити підвищення якості обслуговування клієнтів, скорочення часу на оформлення заявок та запис на ремонт, а також

оптимізацію роботи співробітників автосервісу. Крім того, наявність сучасного вебсайту підвищує імідж автосервісу та дозволяє залучити нових клієнтів, які надають перевагу онлайн-взаємодії з сервісними центрами [3].

Актуальність розробки нового програмного забезпечення для автосервісу обумовлена зростаючою конкуренцією на ринку послуг з ремонту та обслуговування автомобілів. В умовах, коли клієнти мають широкий вибір автосервісів, наявність зручного та функціонального вебсайту може стати вирішальним фактором при виборі сервісного центру. Крім того, впровадження сучасних технологій дозволяє автосервісам оптимізувати свої бізнес-процеси, скоротити витрати та підвищити ефективність роботи.

Таким чином, розробка нового програмного забезпечення для автосервісу є необхідним кроком для забезпечення високої якості обслуговування клієнтів, підвищення ефективності роботи співробітників та зміцнення конкурентних позицій на ринку. Створення вебсайту, який надає можливість онлайн-запису на ремонт, подання заявки на підбір деталей та ознайомлення з послугами автосервісу, дозволить автосервісу йти в ногу з часом та відповідати зростаючим очікуванням клієнтів.

1.2 Аналіз вебзастосунків автосервісу

Сучасні програмні вебзастосунки автосервісу відіграють важливу роль у розвитку індустрії обслуговування автомобілів. Вони дозволяють автосервісам підвищувати ефективність, забезпечувати вищу якість обслуговування та задовольняти зростаючі вимоги клієнтів. Вебсайти автосервісів є невід'ємною частиною цих програмних рішень, оскільки вони забезпечують зручний доступ до інформації про послуги, можливість онлайн-запису на ремонт і замовлення запчастин. Такі сайти допомагають автосервісам залучати нових клієнтів і зберігати лояльність існуючих.

Проблеми, з якими стикаються сучасні автосервіси, включають недостатню автоматизацію процесів, низьку ефективність управління

замовленнями та запчастинами, а також відсутність інтегрованих систем обслуговування клієнтів. Багато автосервісів досі використовують застарілі методи управління, що ускладнює роботу та знижує продуктивність. Відсутність зручних онлайн-платформ для клієнтів призводить до втрати потенційних замовлень і зменшення прибутковості. Тому розробка сучасного програмного забезпечення є надзвичайно актуальною.

Одним з ключових переваг сучасних програмних рішень для автосервісів є їхня здатність інтегрувати різні аспекти діяльності в єдину систему. Це включає управління записами на ремонт, контроль запасів запчастин, фінансовий облік та взаємодію з клієнтами. Такі системи дозволяють знизити витрати, підвищити ефективність і забезпечити високий рівень обслуговування. Крім того, вони надають керівництву автосервісів доступ до аналітичних даних, що дозволяє приймати більш обґрунтовані рішення.

Актуальність розробки нового програмного забезпечення для автосервісу обумовлена зростаючими вимогами до якості та швидкості обслуговування. Сучасні клієнти очікують швидкого і зручного сервісу, можливості онлайн-запису та доступу до інформації в режимі реального часу. Тому автосервіси повинні постійно оновлювати свої технології, щоб відповідати цим вимогам. Розробка нового програмного забезпечення дозволить автосервісам залишатися конкурентоспроможними і задовольняти потреби своїх клієнтів.

Автосервіс «Вишка» є одним з провідних вебсайтів, що пропонує широкий спектр послуг для автосервісів. Вебсайт має такі модулі: інформація про послуги, онлайн-запис на ремонт, замовлення запчастин, блоги та новини. Переваги сайту включають зручний інтерфейс, швидкий доступ до інформації та можливість онлайн-запису. Недоліки юзабіліті включають відсутність мобільної версії та іноді повільне завантаження сторінок. Сайт існує понад 5 років і відповідає сучасним технологіям (рис. 1.1) [4].

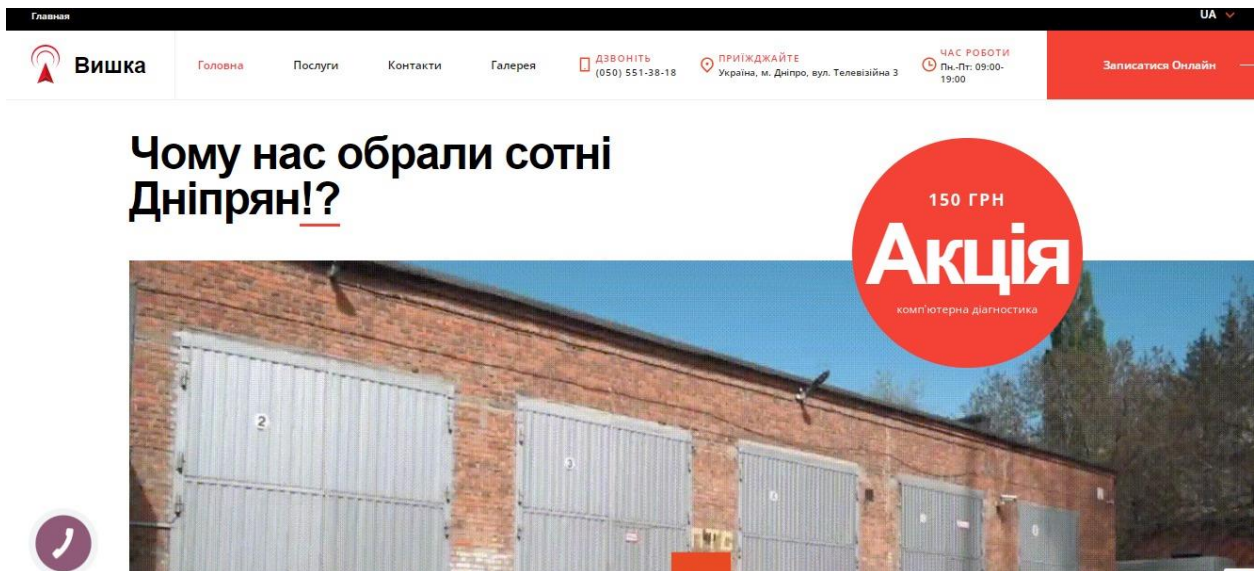


Рисунок 1.1 – Автосервіс Вишка [4]

Сервіс Автоексперт пропонує клієнтам можливість записатися на ремонт та замовити запчастини онлайн. Основні модулі включають каталог послуг, систему запису, відгуки клієнтів та контактну інформацію. Переваги сайту: простий і зрозумілий дизайн, легка навігація та актуальна інформація. Недоліки включають відсутність інтеграції з соціальними мережами та обмежені можливості пошуку. Сайт працює близько 3 років і використовує сучасні технології веброзробки (рис.1.2) [5]

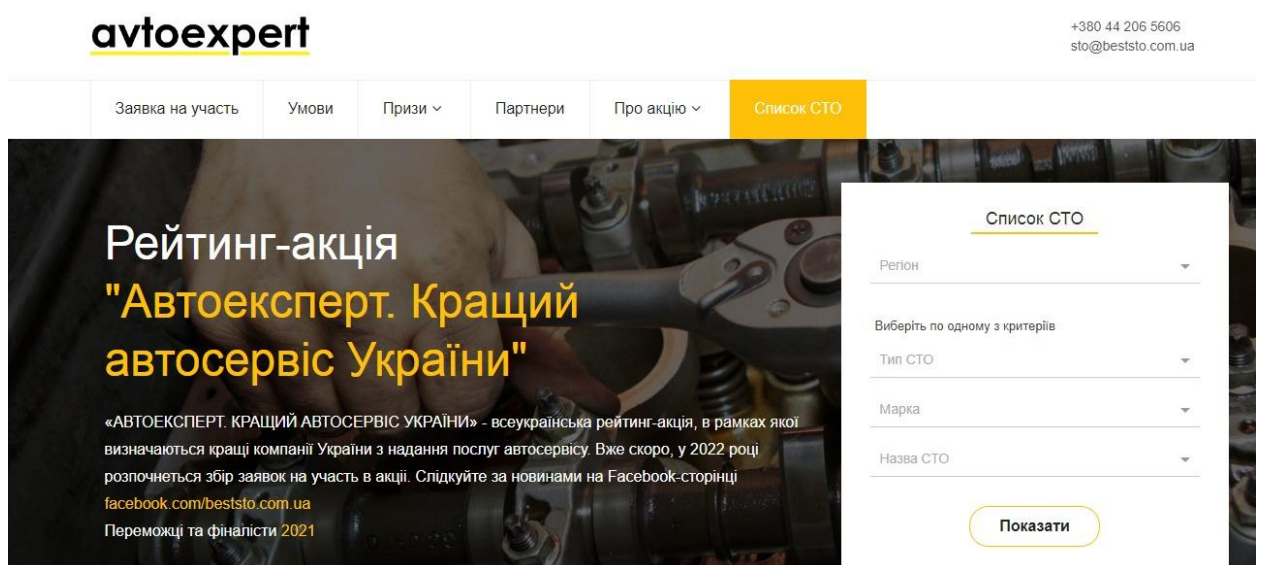


Рисунок 1.2 – Сервіс Автоексперт [5]

Прайм Моторс спеціалізується на пропонуванні послуг автосервісу з акцентом на швидкість обслуговування. Вебсайт включає такі модулі, як онлайн-запис, інформація про послуги, відгуки клієнтів та блоги. Переваги включають швидкий час завантаження, адаптивний дизайн та наявність чату підтримки. Недоліки: відсутність детального каталогу запчастин та іноді проблеми з безпекою. Сайт існує понад 4 роки і відповідає сучасним стандартам безпеки (рис. 1.3) [6]

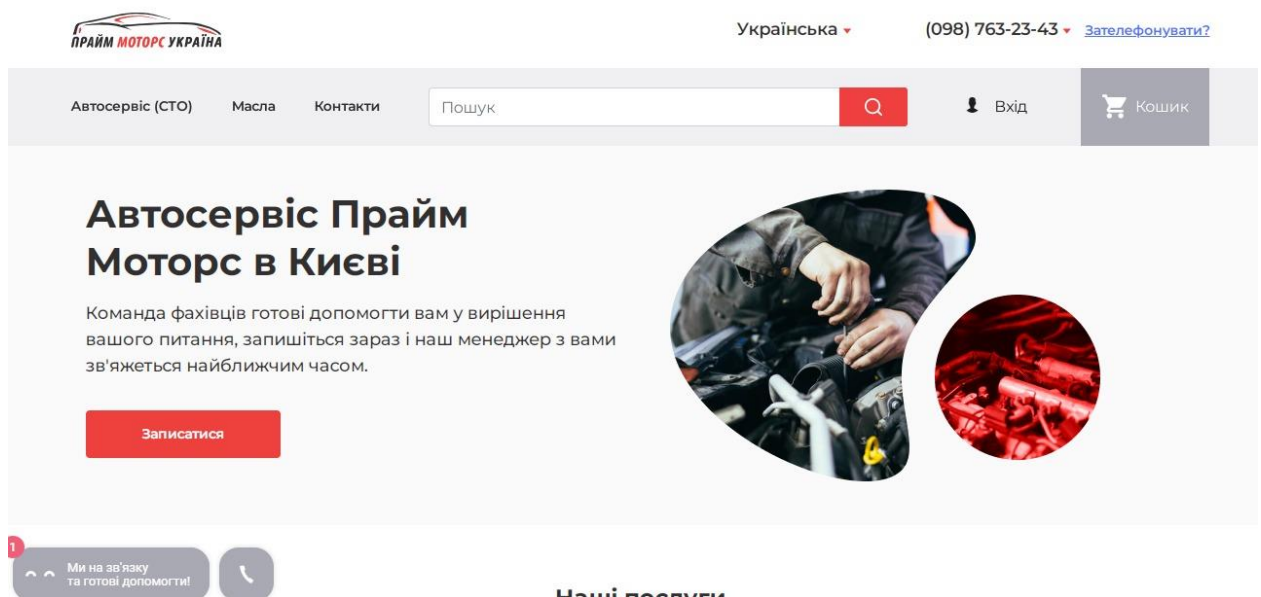


Рисунок 1.3 – Прайм Моторс [6]

Укравто надає повний спектр послуг для автосервісів, включаючи консультації та підбір запчастин. Основні модулі: каталог послуг, система запису, блог та форум. Переваги включають інтерактивний дизайн, наявність форуму для обговорень та можливість онлайн-консультацій. Недоліки: складний процес реєстрації та обмежені можливості для мобільних користувачів. Сайт працює вже більше 6 років і активно оновлюється (рис. 1.4) [7]

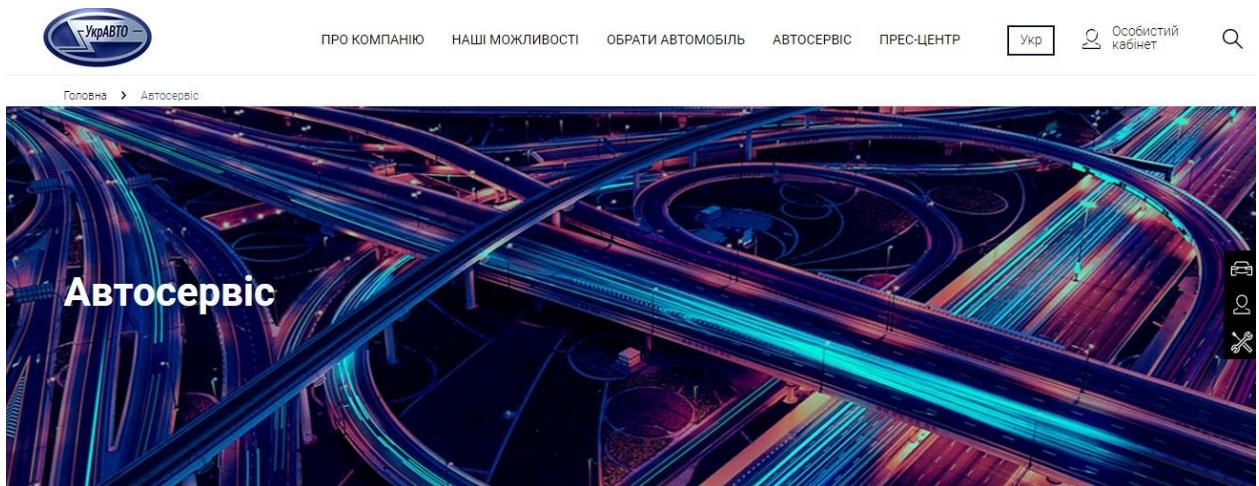


Рисунок 1.4 – Укравто [7]

Порівняльна таблиця 1.1 вебсайтів автосервісів містить детальну інформацію про модулі, переваги, недоліки юзабіліті, зручність сайту, тривалість існування та відповідність сучасним технологіям для кожного з них.

На основі аналізу наведеної таблиці можна зробити такі висновки щодо можливих покращень для нового програмного забезпечення автосервісу:

- зручний інтерфейс та швидкий доступ до інформації. Забезпечення простого та інтуїтивно зрозумілого дизайну, який дозволить користувачам легко знаходити потрібну інформацію;
- адаптивний дизайн. Розробка мобільної версії вебсайту для забезпечення зручного доступу з різних пристроїв;
- інтерактивний дизайн та наявність чату підтримки. Впровадження інтерактивних елементів та чату для надання оперативної допомоги користувачам;
- можливість онлайн-консультацій. Надання користувачам можливості отримувати консультації в режимі реального часу;

- інтеграція з соціальними мережами. Додавання можливості швидкого поширення інформації через соціальні мережі для залучення нових клієнтів;
- відсутність проблем з безпекою. Забезпечення високого рівня безпеки для захисту персональних даних користувачів.

Таблиця 1.1 – Порівняння вебсайтів автосервісів

Параметр	Автосервіс Вишка	Авто Експерт	Прайм Моторс	Укравто
Модулі	Інформація про послуги, Онлайн-запис на ремонт, Замовлення запчастин, Блоги та новини	Каталог послуг, Система запису, Відгуки клієнтів, Контактна інформація	Онлайн-запис, Інформація про послуги, Відгуки клієнтів, Блоги	Каталог послуг, Система запису, Блог, Форум
Переваги	Зручний інтерфейс, Швидкий доступ до інформації, Можливість онлайн-запису	Простий і зрозумілий дизайн, Легка навігація, Актуальна інформація	Швидкий час завантаження, Адаптивний дизайн, Наявність чату підтримки	Інтерактивний дизайн, Наявність форуму, Можливість онлайн-консультацій
Недоліки юзабіліті	Відсутність мобільної версії, Повільне завантаження сторінок	Відсутність інтеграції з соціальними мережами, Обмежені можливості пошуку	Відсутність детального каталогу запчастин, Проблеми з безпекою	Складний процес реєстрації, Обмежені можливості для мобільних користувачів
Зручність сайту	Висока	Висока	Висока	Середня
Тривалість існування	Понад 5 років	Близько 3 років	Понад 4 роки	Більше 6 років
Відповідність до сучасних технологій	Висока	Висока	Висока	Висока

Ці елементи дозволять створити конкурентоспроможний продукт, який відповідатиме потребам сучасних клієнтів автосервісу та забезпечить високий рівень обслуговування.

Аналізуючи наведені приклади, можна зробити висновок, що в новому розроблюваному програмному забезпеченні для автосервісу варто

врахувати переваги, такі як зручний інтерфейс, швидкий доступ до інформації, адаптивний дизайн та інтеграція з сучасними технологіями. Важливо забезпечити наявність мобільної версії, інтерактивного дизайну та можливості онлайн-консультацій для покращення користувацького досвіду. Ці елементи дозволять створити конкурентоспроможний продукт, який задовольнить потреби сучасних клієнтів автосервісу.

1.3 Вимоги до розробки програмного забезпечення

Першим етапом розробки програмного забезпечення для автосервісу є аналіз потреб користувачів. Необхідно визначити, які саме функції та сервіси потрібні кінцевим користувачам - автовласникам та співробітникам автосервісу. Це включає опитування, аналіз ринку та дослідження конкурентів. На основі отриманих даних буде сформульовано технічні вимоги до системи, що включатимуть перелік необхідних функцій та їхню пріоритетність.

Наступним кроком є проектування архітектури системи. Важливо визначити, які компоненти будуть включені в програмне забезпечення, як вони взаємодіятимуть між собою та з зовнішніми системами. Архітектура має бути масштабованою, щоб забезпечити можливість подальшого розширення функціоналу без суттєвих змін у структурі системи. Це також включає вибір технологічного стеку, який найкраще відповідає вимогам проекту.

Розробка інтерфейсу користувача (UI) є критично важливим етапом, оскільки від його зручності та інтуїтивності залежатиме задоволеність клієнтів. Інтерфейс має бути простим, але функціональним, забезпечуючи швидкий доступ до необхідної інформації та можливість виконання основних операцій (запис на ремонт, подання заявок на підбір деталей тощо). Важливо також забезпечити адаптивний дизайн, щоб користувачі могли комфортно користуватися сайтом з мобільних пристроїв.

Основна частина програмного забезпечення складається з функціональних модулів, кожен з яких відповідає за певний набір задач. Серед них: модуль запису на ремонт, модуль підбору запчастин, модуль управління клієнтськими даними, модуль управління замовленнями та інвентарем. Кожен модуль розробляється окремо, але з урахуванням інтеграції з іншими частинами системи.

Для забезпечення повноцінного функціонування нового програмного забезпечення необхідно забезпечити його інтеграцію з існуючими системами управління автосервісом. Це включає синхронізацію баз даних, інтеграцію з CRM-системами, ERP-системами та іншими внутрішніми платформами, що використовуються в автосервісі. Така інтеграція дозволить забезпечити цілісність даних та безперебійну роботу всіх компонентів системи.

Одним з найважливіших аспектів розробки програмного забезпечення є забезпечення безпеки даних користувачів. Необхідно передбачити всі можливі ризики та загрози і розробити відповідні заходи захисту. Це включає використання сучасних методів шифрування даних, впровадження системи автентифікації та авторизації користувачів, захист від атак типу SQL-ін'єкцій, XSS тощо.

Після завершення розробки всіх модулів та інтеграції їх у єдину систему, необхідно провести ретельне тестування програмного забезпечення. Це включає як функціональне тестування, так і тестування на відповідність вимогам безпеки та продуктивності. Верифікація системи дозволить виявити та усунути можливі помилки та недоліки до моменту впровадження програмного забезпечення у експлуатацію.

Вибір технологій для розробки програмного забезпечення залежить від вимог проекту, наявних ресурсів та компетенцій команди. Для створення вебсайту можуть бути використані сучасні фреймворки та мови програмування, такі як React або Angular для фронтенду, та Node.js або Django для бекенду. Важливо також вибрати надійну систему управління базами даних, таку як PostgreSQL або MySQL.

Після завершення розробки та тестування програмного забезпечення необхідно провести його впровадження у автосервісі. Це включає встановлення програмного забезпечення на робочі станції, налаштування системи та навчання співробітників автосервісу. Навчання повинно включати як загальні принципи роботи з системою, так і специфічні інструкції для кожного функціонального модуля.

Після впровадження програмного забезпечення важливо забезпечити його подальшу підтримку та оновлення. Це включає моніторинг роботи системи, виявлення та усунення можливих проблем, впровадження нових функцій та оновлення існуючих. Регулярні оновлення дозволять забезпечити відповідність програмного забезпечення сучасним вимогам та потребам користувачів, а також підтримувати високий рівень безпеки.

Ці етапи та їх завдання також описані в табл. 1.2.

Таблиця 1.2 - Вимоги до розробки програмного забезпечення

Етап	Завдання
Аналіз потреб користувачів	Опитування, аналіз ринку, дослідження конкурентів, формулювання технічних вимог
Визначення архітектури	Проектування компонентів, визначення взаємодії між ними, вибір технологічного стеку
Розробка інтерфейсу користувача	Створення простого та інтуїтивно зрозумілого дизайну, забезпечення адаптивності для мобільних пристроїв
Розробка функціональних модулів	Розробка модулів запису на ремонт, підбору запчастин, управління клієнтськими даними та інвентарем
Інтеграція з існуючими системами	Синхронізація баз даних, інтеграція з CRM та ERP-системами
Забезпечення безпеки	Використання шифрування, систем автентифікації та авторизації, захист від атак
Тестування та верифікація	Функціональне тестування, тестування безпеки та продуктивності, верифікація
Підтримка та оновлення	Моніторинг, виявлення та усунення проблем, регулярні оновлення та впровадження нових функцій

Ці завдання допоможуть створити якісне програмне забезпечення для автосервісу, яке задовольнить потреби як користувачів, так і співробітників автосервісу, забезпечуючи високий рівень обслуговування та ефективність роботи.

1.4 Висновки до розділу 1

У першому розділі було проведено детальний аналіз предметної області розробки програмного забезпечення для автосервісу. Розглянуто актуальність та необхідність створення сучасного вебсайту автосервісу, який дозволить спростити взаємодію з клієнтами, підвищити ефективність роботи та забезпечити конкурентні переваги на ринку.

Аналіз існуючих вебдодатків автосервісів показав, що більшість з них мають ряд недоліків, таких як застарілий дизайн, відсутність зручної навігації, обмежений функціонал та недостатню інтеграцію з іншими системами. Порівняння чотирьох популярних вебсайтів автосервісів в Україні дозволило виявити їх сильні та слабкі сторони, а також сформулювати рекомендації щодо покращення нового програмного забезпечення.

На основі проведеного аналізу були визначені ключові вимоги до розробки нового програмного забезпечення для автосервісу. Серед них: аналіз потреб користувачів, визначення архітектури системи, розробка зручного та адаптивного інтерфейсу користувача, створення функціональних модулів (запис на ремонт, підбір запчастин, управління клієнтськими даними та інвентарем), інтеграція з існуючими системами автосервісу, забезпечення безпеки даних, ретельне тестування та верифікація, а також подальша підтримка та оновлення системи.

Дотримання цих вимог дозволить створити якісне та ефективне програмне забезпечення, яке задовольнить потреби як клієнтів, так і

співробітників автосервісу. Новий вебсайт забезпечить зручний доступ до інформації про послуги, можливість онлайн-запису на ремонт та замовлення запчастин, що значно підвищить рівень обслуговування та лояльність клієнтів.

Таким чином, розробка сучасного програмного забезпечення для автосервісу є необхідним кроком для підвищення ефективності роботи, покращення якості обслуговування та зміцнення конкурентних позицій на ринку. Врахування виявлених вимог та рекомендацій дозволить створити продукт, який відповідатиме очікуванням користувачів та сприятиме успішному розвитку автосервісу.

2 МАТЕРІАЛИ І МЕТОДИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Архітектура програмного забезпечення

Для розробки програмного забезпечення автосервісу було вирішено використовувати клієнт-серверну архітектуру, схему якої надано в рис 2.1. Цей вибір обґрунтовується рядом переваг, які надає дана архітектура для вебзастосунків такого типу. По-перше, клієнт-серверна архітектура забезпечує чіткий розподіл функцій та обов'язків між клієнтською та серверною частинами. Клієнтська частина, яка виконується у веббраузері користувача, відповідає за відображення інтерфейсу та взаємодію з користувачем. Серверна частина, розташована на віддаленому сервері, займається обробкою запитів, роботою з базою даних та бізнес-логікою застосунку. Такий розподіл дозволяє оптимізувати навантаження та підвищити продуктивність системи.

По-друге, клієнт-серверна архітектура надає можливість централізованого зберігання та управління даними. Всі дані автосервісу, включаючи інформацію про послуги, записи на ремонт, заявки на підбір деталей тощо, зберігаються на сервері у базі даних. Це забезпечує цілісність та узгодженість даних, а також спрощує їх резервне копіювання та відновлення у разі необхідності.

По-третє, клієнт-серверна архітектура дозволяє забезпечити високий рівень безпеки даних. Оскільки всі чутливі дані зберігаються та обробляються на сервері, можна застосувати надійні методи автентифікації, авторизації та шифрування для захисту інформації від несанкціонованого доступу.

Всі перераховані переваги клієнт-серверної архітектури також описані в табл. 2.1.

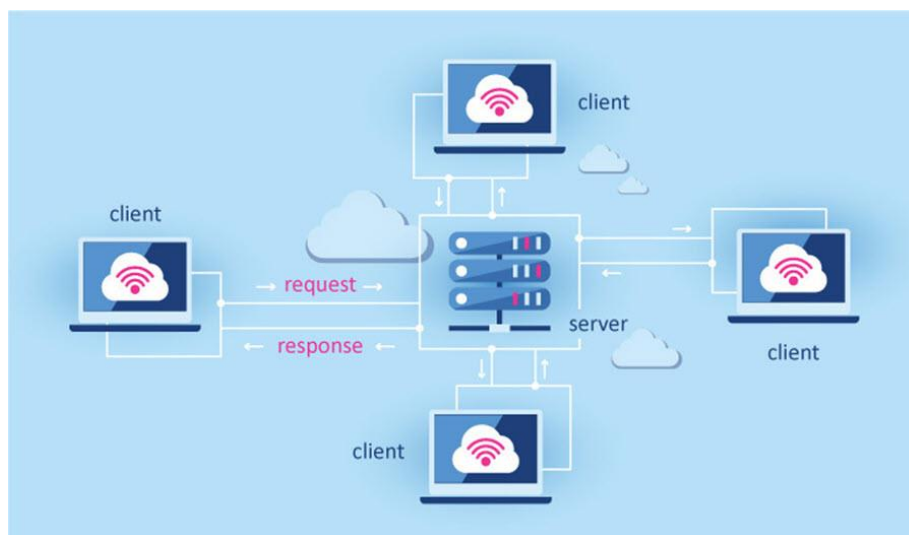


Рисунок 2.1 – Клієнт-серверна архітектура

Таблиця 2.1 - Переваги клієнт-серверної архітектури

Перевага	Опис
Розподіл обов'язків	Чітке розділення функцій між клієнтською (інтерфейс) і серверною (логіка і дані) частинами, що покращує організацію і продуктивність системи.
Централізоване управління даними	Усі дані зберігаються на сервері, що забезпечує їх цілісність, узгодженість та спрощує резервне копіювання.
Високий рівень безпеки	Чутливі дані обробляються на сервері, з можливістю використання сильних механізмів захисту інформації.

Для реалізації клієнт-серверної архітектури було обрано шаблон проєктування Model-View-Controller (MVC)(рис. 2.2). Цей шаблон дозволяє розділити застосунок на три основні компоненти: модель (Model), представлення (View) та контролер (Controller). Модель відповідає за роботу з даними та бізнес-логіку. Представлення займається відображенням даних та інтерфейсу користувача. Контролер забезпечує взаємодію між моделлю та представленням, обробляє запити користувача та оновлює модель і представлення відповідно.

Використання шаблону MVC надає ряд переваг. По-перше, він забезпечує чіткий поділ відповідальності між компонентами, що полегшує розробку, тестування та супровід застосунку. По-друге, MVC дозволяє досягти високого рівня модульності та повторного використання коду. Компоненти можуть бути розроблені та протестовані незалежно один від одного, що спрощує процес розробки та зменшує кількість потенційних помилок. По-третє, шаблон MVC сприяє масштабованості застосунку. У разі необхідності розширення функціоналу чи додавання нових модулів, зміни можуть бути внесені в окремі компоненти без впливу на інші частини системи.

Всі перераховані переваги шаблону проєктування MVC також описані в табл. 2.2.

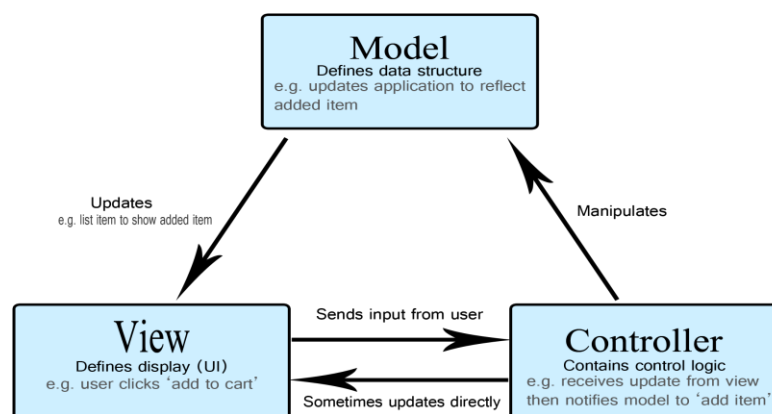


Рисунок 2.2 -Шалон проєктування

Отже, для розробки програмного забезпечення автосервісу було обрано клієнт-серверну архітектуру у поєднанні з шаблоном проєктування MVC. Такий підхід дозволяє створити ефективний, безпечний та масштабований вебзастосунок, який відповідатиме вимогам замовника та потребам користувачів.

Таблиця 2.2 - Переваги шаблону проєктування MVC

Перевага	Опис
Чіткий поділ відповідальності	Розмежування ролей між моделлю (дані, логіка), представленням (інтерфейс) і контролером (логіка управління), що полегшує розробку і супровід.
Висока модульність і повторне використання коду	Компоненти можна розробляти і тестувати незалежно, що сприяє легшій розробці і меншому числу помилок.
Масштабованість	Зміни можна вносити в окремі компоненти без впливу на інші частини системи, що забезпечує легке розширення функціоналу.

2.2 Методи розробки програмного забезпечення

Розробка програмного забезпечення - це складний процес, який вимагає вибору оптимальної методології та підходу в залежності від специфіки проєкту та вимог замовника. Серед найбільш поширених методів розробки можна виділити каскадну модель (Waterfall), ітеративну модель, гнучкі методології (Agile) та екстремальне програмування (XP).

Каскадна модель (Waterfall) - це послідовний підхід до розробки програмного забезпечення, в якому процес розбивається на чіткі фази, такі як аналіз вимог, проєктування, реалізація, тестування та впровадження. Кожна фаза повинна бути завершена перед переходом до наступної. Цей підхід є структурованим та передбачуваним, але не дуже гнучким, оскільки зміни вимог на пізніх етапах можуть призвести до значних витрат часу та ресурсів.

Ітеративна модель передбачає розбиття процесу розробки на кілька ітерацій, кожна з яких включає аналіз вимог, проєктування, реалізацію та тестування. Після кожної ітерації замовник отримує робочу версію продукту з обмеженою функціональністю, що дозволяє отримати зворотний зв'язок та

внести необхідні зміни. Цей підхід є більш гнучким порівняно з каскадною моделлю, але вимагає ретельного планування та управління проєктом.

Гнучкі методології (Agile) ґрунтуються на ітеративному та інкрементальному підході до розробки. Вони роблять акцент на швидкій розробці, постійному вдосконаленні та адаптації до мінливих вимог. Серед найбільш популярних гнучких методологій можна виділити Scrum, Kanban та Lean. Ці підходи дозволяють швидко реагувати на зміни, залучати замовника до процесу розробки та забезпечувати високу якість кінцевого продукту.

Екстремальне програмування (XP) - це гнучка методологія, яка робить акцент на простоті, комунікації та зворотному зв'язку. XP передбачає тісну співпрацю між замовником та командою розробників, часті релізи, парне програмування та рефакторинг коду. Цей підхід дозволяє швидко створювати високоякісне програмне забезпечення, але вимагає високої кваліфікації та дисципліни від команди розробників.

При виборі методу розробки для програмного забезпечення автосервісу було вирішено використовувати гнучку методологію Scrum. Цей вибір обумовлений необхідністю швидко реагувати на зміни вимог, забезпечувати постійний зворотний зв'язок з замовником та залучати його до процесу розробки. Scrum передбачає розбиття процесу на короткі ітерації (спринти), протягом яких команда працює над реалізацією обраних функцій. Після кожного спринту замовник отримує робочу версію продукту та може вносити корективи до вимог.

Використання Scrum для розробки програмного забезпечення автосервісу дозволяє забезпечити високу гнучкість та адаптивність процесу розробки. Це особливо важливо в умовах динамічного ринку та мінливих потреб клієнтів. Крім того, залучення замовника до процесу розробки дозволяє забезпечити повну відповідність кінцевого продукту його очікуванням та вимогам.

Таким чином, вибір методу розробки програмного забезпечення має ґрунтуватися на специфіці проєкту, вимогах замовника та особливостях

команди розробників. Для розробки програмного забезпечення автосервісу оптимальним вибором є використання гнучкої методології Scrum, яка дозволяє швидко реагувати на зміни, забезпечувати високу якість продукту та залучати замовника до процесу розробки.

2.3 Аналіз технологій розробки програмного забезпечення

Вибір технологій для розробки програмного забезпечення є критично важливим рішенням, яке впливає на продуктивність, масштабованість, безпеку та зручність супроводу застосунку. Серед найбільш поширених технологій для веброзробки можна виділити мови програмування (JavaScript, Python, PHP, Java), фреймворки (React, Angular, Vue.js, Django, Laravel), бази даних (MySQL, PostgreSQL, MongoDB) та інструменти для розгортання та управління інфраструктурою (Docker, Kubernetes, AWS, Azure) [8].

JavaScript - це мова програмування, яка використовується для створення інтерактивних вебсторінок та застосунків. Вона дозволяє розробникам додавати динамічну поведінку, обробляти події та взаємодіяти з користувачем на стороні клієнта. JavaScript також використовується на стороні сервера за допомогою платформи Node.js [9].

PHP - це серверна мова програмування, яка спеціально розроблена для веброзробки. PHP дозволяє створювати динамічні вебсторінки, обробляти запити користувачів та взаємодіяти з базами даних. PHP має велику екосистему фреймворків та бібліотек, таких як Laravel та Symfony [10].

React - це бібліотека JavaScript для створення користувацьких інтерфейсів. React дозволяє розробникам будувати інтерактивні та багаторазові компоненти інтерфейсу, які ефективно оновлюються та відображаються при зміні даних. React часто використовується у поєднанні з іншими бібліотеками та інструментами, такими як Redux та Webpack [11].

Spring - це потужний фреймворк для розробки корпоративних застосунків на мові Java. Spring надає широкий спектр функцій, таких як

інверсія управління (IoC), аспектно-орієнтоване програмування (AOP), управління транзакціями та інтеграція з іншими технологіями. Spring часто використовується для створення вебсервісів та мікросервісної архітектури [12].

Для розробки програмного забезпечення автосервісу було вирішено використовувати мову програмування PHP у поєднанні з фреймворком Laravel та бібліотекою React для розробки клієнтської частини. Цей вибір обумовлений рядом переваг, які надають ці технології для проєкту такого типу.

По-перше, PHP є широко використовуваною та перевіреною часом мовою програмування для веброзробки. PHP має простий синтаксис, велику кількість вбудованих функцій та розширень, що дозволяє швидко розробляти динамічні та інтерактивні вебзастосунки [13].

По-друге, Laravel є потужним та елегантним фреймворком PHP, який значно полегшує розробку. Laravel надає багато вбудованих функцій, таких як ORM Eloquent для роботи з базами даних, шаблонізатор для створення динамічних шаблонів, вбудований механізм автентифікації та авторизації. Це дозволяє швидко розробляти функціональний та надійний бекенд для застосунку автосервісу.

По-третє, використання React для розробки клієнтської частини дозволяє створити інтерактивний та динамічний інтерфейс користувача. React дозволяє будувати багаторазові компоненти, які можна легко комбінувати та використовувати в різних частинах застосунку. Це забезпечує високу швидкість розробки та зручність підтримки коду [14].

Для зберігання даних було обрано реляційну базу даних MySQL. MySQL є надійною, швидкою та масштабованою базою даних, яка добре інтегрується з PHP та Laravel. База даних буде використовуватися для зберігання інформації про послуги автосервісу, записи на ремонт, заявки на підбір деталей, дані клієнтів тощо [15].

Для розробки та тестування застосунку локально буде використовуватися вебсервер Open Server. Open Server - це портативна серверна платформа, яка включає в себе вебсервер Apache, базу даних MySQL, інтерпретатор PHP та інші необхідні інструменти для локальної розробки. Використання Open Server дозволяє швидко налаштувати локальне середовище розробки та тестувати застосунок перед розгортанням на production-сервері [16].

Таким чином, для розробки програмного забезпечення автосервісу було обрано мову програмування PHP, фреймворк Laravel, бібліотеку React, базу даних MySQL та локальний вебсервер Open Server для розробки. Ці технології надають потужні інструменти для швидкої та ефективної розробки, забезпечення безпеки та надійності застосунку. Вони дозволяють створити високоякісне програмне забезпечення, яке відповідатиме вимогам замовника та потребам користувачів автосервісу.

Таблиця 2.3 підсумовує вибрані технології для розробки програмного забезпечення автосервісу та пояснює їх роль і переваги.

Таблиця 2.3 - Вибрані технології для розробки програмного забезпечення автосервісу

Технологія	Категорія	Роль у проекті	Основні переваги
PHP	Мова програмування	Серверна логіка	Широка використаність, простий синтаксис, швидка розробка, велика екосистема фреймворків та бібліотек.
Laravel	Фреймворк (PHP)	Розробка бекенду	Організація коду, ORM Eloquent для роботи з базами даних, шаблонізатор, вбудовані механізми автентифікації та авторизації.
React	Бібліотека (JavaScript)	Розробка клієнтської частини	Побудова інтерактивних UI, ефективне оновлення компонентів, висока швидкість розробки і зручність підтримки.
MySQL	Система управління базами даних	Зберігання даних	Надійність, швидкість, масштабованість, добра інтеграція з PHP та Laravel.
Open Server	Локальний вебсервер	Розробка і тестування	Легке налаштування локального середовища, включає Apache, MySQL, PHP.

2.4 Висновки до розділу

У другому розділі дипломної роботи було детально розглянуто ключові аспекти розробки програмного забезпечення для автосервісу, включаючи вибір архітектури, методів та технологій розробки.

Після аналізу переваг та недоліків різних архітектурних підходів, було обґрунтовано вибір клієнт-серверної архітектури для реалізації вебзастосунку автосервісу. Клієнт-серверна архітектура забезпечує чіткий розподіл функцій та обов'язків між клієнтською та серверною частинами, централізоване зберігання та управління даними, а також високий рівень безпеки. Для ефективної організації коду та забезпечення масштабованості було обрано шаблон проєктування MVC (Model-View-Controller).

Розглянувши різні методології розробки програмного забезпечення, для проєкту автосервісу було обрано гнучку методологію Scrum. Scrum дозволяє швидко реагувати на зміни вимог, забезпечувати постійний зворотний зв'язок із замовником та залучати його до процесу розробки. Цей підхід є особливо ефективним в умовах динамічного ринку та мінливих потреб клієнтів.

Аналіз технологій розробки показав, що оптимальним вибором для реалізації програмного забезпечення автосервісу є використання мови програмування PHP у поєднанні з фреймворком Laravel для розробки серверної частини та бібліотеки React для створення клієнтського інтерфейсу. Ці технології надають потужні інструменти для швидкої та ефективної розробки, забезпечення безпеки та надійності застосунку. Для зберігання даних було обрано реляційну базу даних MySQL, а для локальної розробки та тестування - вебсервер Open Server.

3 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Проектування та розробка бази даних

Таблиця "Користувачі" містить інформацію про всіх користувачів системи. Ця таблиця використовується для авторизації та ідентифікації користувачів, визначення їхніх ролей і доступів у системі, а також для контакту з користувачами (табл. 3.1).

Таблиця 3.1 - Користувачі

Атрибут	Тип	Опис
_id	ObjectId	Унікальний ідентифікатор користувача
username	String	Ім'я користувача
password	String	Хеш пароля
email	String	Електронна адреса користувача
phone	String	Телефонний номер користувача
role	String	Роль користувача ('customer' або 'admin')
createdAt	Date	Дата створення запису

Таблиця "Послуги" зберігає інформацію про всі наявні послуги, які надає автосервіс. Включає деталі про вартість, тривалість та опис послуг (табл. .3.2).

Таблиця 3.2 - Послуги

Атрибут	Тип	Опис
_id	ObjectId	Унікальний ідентифікатор послуги
name	String	Назва послуги
description	String	Опис послуги
price	Number	Вартість послуги
duration	Number	Тривалість послуги в хвилинах
createdAt	Date	Дата створення запису
updatedAt	Date	Дата оновлення запису

Таблиця "Записи на ремонт" використовується для управління записами клієнтів на ремонтні роботи, включаючи статуси та дати записів (табл. 3.3).

Таблиця 3.3- Записи на ремонт

Атрибут	Тип	Опис
_id	ObjectId	Унікальний ідентифікатор запису
userId	ObjectId	ID користувача, що зробив запис
serviceId	ObjectId	ID послуги
appointmentDate	Date	Дата і час запису
status	String	Статус запису ('pending', 'confirmed', 'cancelled')
createdAt	Date	Дата створення запису
updatedAt	Date	Дата оновлення запису

Таблиця "Деталі" зберігає інформацію про запчастини, їх наявність, ціни, і відповідність до певних марок і моделей автомобілів (табл. 3.4).

Таблиця 3.4- Деталі

Атрибут	Тип	Опис
_id	ObjectId	Унікальний ідентифікатор деталі
carMake	String	Марка автомобіля
carModel	String	Модель автомобіля
year	Number	Рік випуску автомобіля
partName	String	Назва деталі
availability	Boolean	Чи є деталь в наявності
price	Number	Ціна деталі
createdAt	Date	Дата створення запису
updatedAt	Date	Дата оновлення запису

Таблиця "Відгуки" використовується для збереження відгуків від користувачів про послуги, з вказанням їхніх оцінок та коментарів (табл. 3.5).

Таблиця 3.5 - Відгуки

Атрибут	Тип	Опис
_id	ObjectId	Унікальний ідентифікатор відгуку
userId	ObjectId	ID користувача, що залишив відгук
serviceId	ObjectId	ID послуги, на яку залишений відгук
rating	Number	Оцінка від 1 до 5
comment	String	Коментар користувача
createdAt	Date	Дата створення запису

3.2 Розробка програмного коду

Головна сторінка вебсайту автосервісу включає меню навігації, яке допомагає користувачам легко орієнтуватися по ресурсу. У меню є розділ "Про нас", де представлена інформація про автосервіс, його історію та особливості. Розділ "Послуги" містить перелік всіх послуг, що надаються, включаючи їх детальний опис. Через розділ "Запис на ремонт" можна подати онлайн-заявку на ремонт, вибравши зручний час та дату. "Підбір деталей" пропонує форму для запиту на замовлення необхідних автозапчастин. Розділ "Контакти" надає всю необхідну контактну інформацію, включаючи адресу, телефон та графік роботи автосервісу (табл. 3.6).

На сторінці послуг користувачі можуть ознайомитись з детальними описами кожної послуги та зробити запис на конкретну послугу. Сторінка запису на ремонт містить форму для запису та інформацію про вільні терміни. Сторінка підбору деталей дозволяє подати запит на підбір деталей з урахуванням марки, моделі та року випуску авто, а також надає інформацію про наявність та можливість замовлення цих деталей.

Таблиця 3.6 - Структура та функціональність вебсайту автосервісу

Назва файлу	Опис
index.php	Головна сторінка сайту, що включає меню навігації з основними розділами: Про нас, Послуги, Запис на ремонт, Підбір деталей та Контакти. Служить вхідною точкою для користувачів, які бажають дізнатися більше про сервіси та можливості автосервісу.
services.php	Сторінка з детальним описом послуг, що надаються автосервісом. Забезпечує інформацію про кожну послугу та можливість запису на конкретну послугу.
booking.php	Сторінка для онлайн-запису на ремонт, де користувач може вибрати дату та час візиту. Також відображає інформацію про вільні терміни.
parts_request.php	Сторінка для запиту деталей з формою, куди вводяться дані про марку авто, модель, рік випуску та потрібні деталі. Включає інформацію про наявність та можливість замовлення деталей.
contact.php	Сторінка з контактною інформацією автосервісу, включаючи адресу, телефон, електронну пошту та графік роботи.
reviews.php	Сторінка відгуків, де відображається список відгуків клієнтів із оцінками. Включає форму для залишення відгуку.
manage_bookings.php	Сторінка управління записами, на якій можна переглядати всі зареєстровані записи, а також скасувати або змінити час візиту.
login.php	Сторінка для авторизації користувачів, що включає форму входу.
register.php	Сторінка для реєстрації нових користувачів, яка містить форму реєстрації.

Сторінка контактів пропонує всю важливу контактну інформацію та графік роботи сервісу. Сторінка відгуків дозволяє клієнтам залишити свої відгуки та переглянути відгуки інших користувачів. Сторінка управління записами забезпечує можливість переглядати всі зареєстровані записи та при необхідності змінювати час або навіть скасовувати їх. Нарешті, сторінка авторизації та реєстрації надає форми для входу та реєстрації на сайті, що робить доступ до послуг більш особистим та зручним.

Підключення до бази даних та робота з нею є критично важливими компонентами вебзастосунку, особливо для сервісу, як автосервіс, де потрібно зберігати і обробляти велику кількість даних, включаючи інформацію про клієнтів, замовлення, деталі для автомобілів та записи на сервіс. Розробка програмного забезпечення для автосервісу, як описано вище, було використано PHP та MySQL.

Для встановлення з'єднання з базою даних в PHP було використано PDO (PHP Data Objects) або mysqli (MySQL Improved). PDO надає більш гнучкі та безпечні засоби для роботи з різними СУБД, а mysqli спеціалізується на MySQL. При використанні фреймворку Laravel, робота з базою даних значно спрощується завдяки вбудованій ORM (Object-Relational Mapping) системі Eloquent, яка дозволяє взаємодіяти з базою даних за допомогою об'єктно-орієнтованого підходу.

Для створення з'єднання з базою даних у Laravel було налаштовано параметри з'єднання у файлі конфігурації `.env`, де вказано ім'я бази даних, ім'я користувача, пароль, адреса сервера, та інші параметри. Laravel автоматично використовує ці параметри для встановлення з'єднання через Eloquent.

Використання Eloquent дозволило розробникам виконувати запити до бази даних, не занурюючись в деталі SQL-синтаксису. Наприклад, для створення нового запису на сервіс було використано такий код (рис. 3.1).

```
$appointment = new Appointment;
$appointment->client_id = $client_id;
$appointment->service_id = $service_id;
$appointment->date = $date;
$appointment->time = $time;
$appointment->save();
```

Рисунок 3.1 - Створення нового запису на сервіс

Цей код створив новий запис у таблиці призначень, використовуючи об'єкти та методи Eloquent. Такий підхід забезпечив високий рівень

абстракції та безпеку, оскільки ORM автоматично обробив запити на створення, читання, оновлення та видалення даних (CRUD), захистивши від ін'єкцій SQL.

Для складніших запитів, які вимагали з'єднань між таблицями або спеціальних умов, у Laravel було використано білдер запитів, який надав більше контролю над SQL-операціями, при цьому зберігаючи зручність і безпеку (рис. 3.2)

```
$services = Service::where('available', true)->get();
```

Рисунок 3.2 - Білдери запитів

Цей код вибрав всі послуги, які були доступні для запису. Laravel також дозволив легко впровадити валідацію, аутентифікацію, і кешування, що було важливим для забезпечення швидкодії та безпеки вебзастосунку.

Для клієнтської частини вебзастосунку автосервісу було вирішено використовувати бібліотеку React, що дозволило створити інтерактивний та динамічний інтерфейс користувача. Використання React сприяло розробці багаторазових компонентів, які легко комбінувати та використовувати у різних частинах застосунку, забезпечуючи при цьому високу швидкість розробки та зручність підтримки коду.

Розробка інтерфейсу зосереджувалася на створенні зрозумілого та зручного користувацького досвіду. Були створені такі React компоненти: 'Home.js' для головної сторінки, 'Services.js' для переліку послуг, 'BookingForm.js' для форми запису на ремонт, 'PartRequestForm.js' для форми запиту на підбір деталей, 'Contact.js' для сторінки з контактною інформацією, та 'ReviewForm.js' для залишення відгуків. Дизайн відповідав сучасним вимогам UX/UI, забезпечуючи інтуїтивно зрозумілу навігацію та взаємодію (табл. 3.7).

Таблиця 3.7 - Клієнтська частина веб-застосунку автосервісу

Назва файлу	Компонент	Опис
Home.js	Головна сторінка	Компонент для відображення головної сторінки сайту, включає посилання на основні функції та послуги.
Services.js	Сторінка послуг	Компонент, який відображає перелік та опис послуг, які надає автосервіс.
BookingForm.js	Форма запису на ремонт	Компонент для створення форми запису на ремонт з вибором дати та часу.
PartRequestForm.js	Форма запиту на підбір деталей	Компонент для створення форми запиту деталей з можливістю вказати марку авто, модель, рік випуску.
Contact.js	Сторінка контактів	Компонент для відображення контактної інформації автосервісу, включаючи адресу, телефон та графік роботи.
ReviewForm.js	Форма відгуків	Компонент, що дозволяє користувачам залишати відгуки та переглядати існуючі оцінки і коментарі.

Клієнтська логіка була реалізована через взаємодію компонентів з сервером за допомогою асинхронних запитів API. Для цього використовувалась бібліотека Axios, яка дозволяла здійснювати GET та POST запити для обміну даними з сервером. Це забезпечувало динамічне оновлення контенту без перезавантаження сторінки.

Для стилізації використовувалися CSS-модулі, які допомагають ізолювати стилі компонентів, уникати конфліктів і покращувати масштабованість стилів. Зовнішній вигляд був адаптивним, що забезпечувало коректне відображення на різних пристроях та розмірах екранів.

Безпека клієнтської частини забезпечувалася використанням сучасних практик, таких як HTTPS для захищених запитів та токенів для аутентифікації сесій користувачів. Реакт також сприяв оптимізації продуктивності, завдяки ефективному управлінню станами компонентів та оптимізованому перерендерингу тільки тих частин інтерфейсу, які зазнали змін.

Таким чином, клієнтська частина вебзастосунку автосервісу була розроблена з використанням сучасних технологій та практик, забезпечуючи високу продуктивність, безпеку та відмінний користувацький досвід.

3.3 Користування вебсайтом

Головна сторінка вебсайту автосервісу "AutoCarePro" представлена в червоних та чорних тонах з використанням яскравого, звертаючого увагу заголовка "AutoCarePro - Ваш надійний автосервіс". Під заголовком розташований підзаголовок, що розповідає про професійний ремонт та підбір деталей для онлайн-запису на сервіс автомобілів. На сторінці також є кнопки для запису на ремонт та дізнатися більше про сервіси, що підкреслює зручність і доступність сервісних послуг. Верхня частина містить навігаційне меню з посиланнями на головну сторінку, контакти та управління записами, а також кнопку звернення "Contact us" (рис. 3.3).

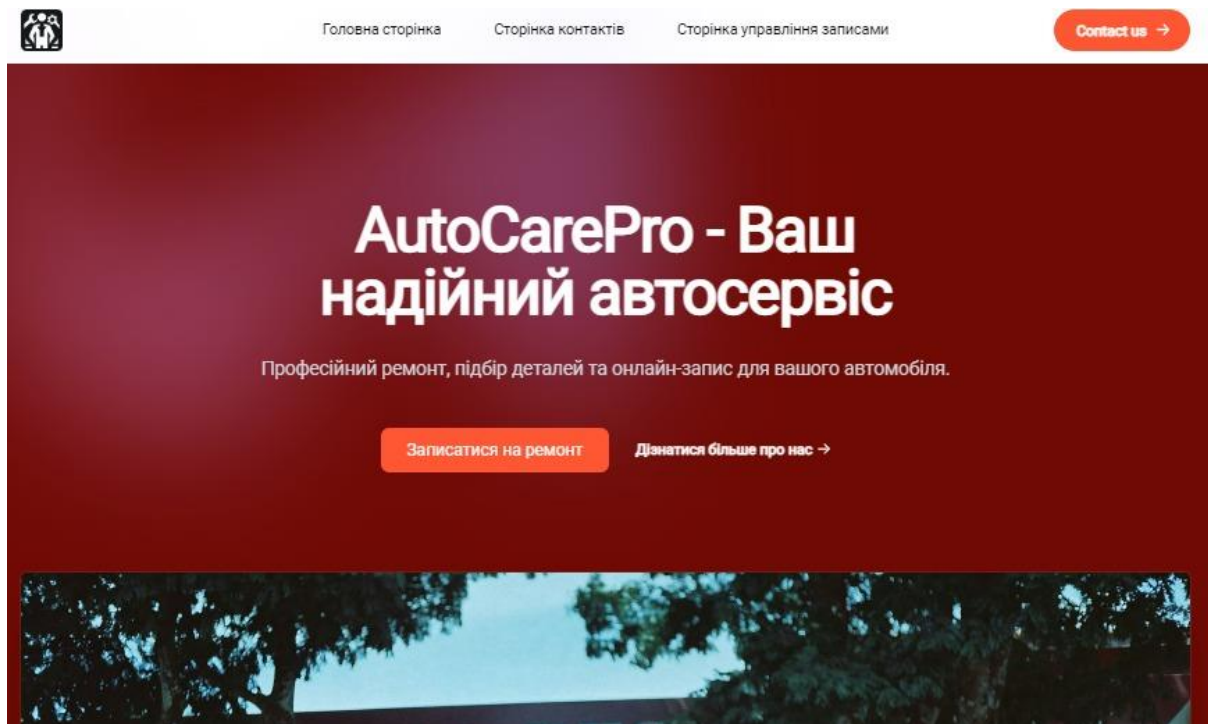


Рисунок 3.3 – Головна сторінка

Блок вебсайту автосервісу "AutoCarePro" пропонує користувачам чіткий огляд основних послуг, які забезпечують все необхідне для автомобіля. Заголовок "Наші послуги забезпечують все необхідне для вашого автомобіля" встановлює контекст для трьох ключових послуг: "Професійний ремонт", "Підбір деталей", та "Онлайн-запис". Кожна послуга підкреслена іконкою для кращої візуальної ідентифікації, що допомагає користувачам швидше знайти потрібну інформацію та сервіси на сайті (рис. 3.4).

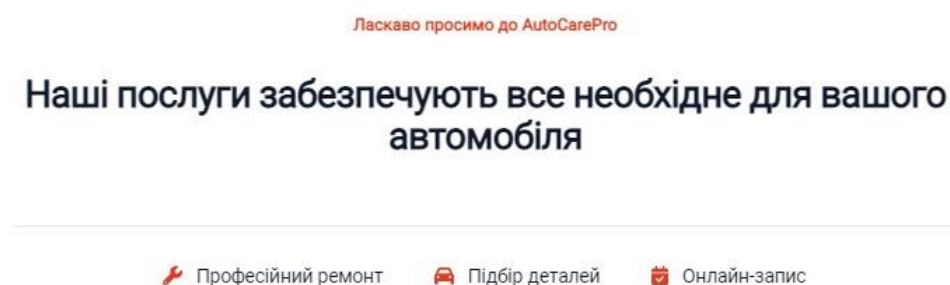


Рисунок 3.4 – Блок інформації

Блок вебсайту автосервісу "AutoCarePro" інформує користувачів про доступні локації сервісу в різних містах: Київ, Львів та Одеса. Для кожного міста надано адресу сервісного центру, що допомагає клієнтам знайти найближчий до них сервіс. Вказано також спеціалізовані електронні адреси для зв'язку з конкретним центром, що сприяє зручності та оперативності обслуговування. Така організація інформації сприяє легкому знаходженню потрібної локації та спілкуванню з відповідними відділеннями (рис. 3.5).



Рисунок 3.5 - Сторінка послуг з детальним описом розташування філій

Блок вебсайту "AutoCarePro" представляє команду професіоналів, які забезпечують якісне обслуговування автомобілів. У цьому розділі зображено п'ять членів команди, кожен з яких представлений із своєю роллю та фотографією на тлі робочого процесу. Олександр Іванов - головний механік, Марія Петрова - менеджер з обслуговування клієнтів, Андрій Сидоров - фахівець з підбору деталей, Ольга Коваль - директора з продажу, та Дмитро Шевченко - асистент майстерні. Всі вони ретельно підібрані за своїми навичками та досвідом, щоб забезпечити найкращий сервіс для клієнтів (рис. 3.6)

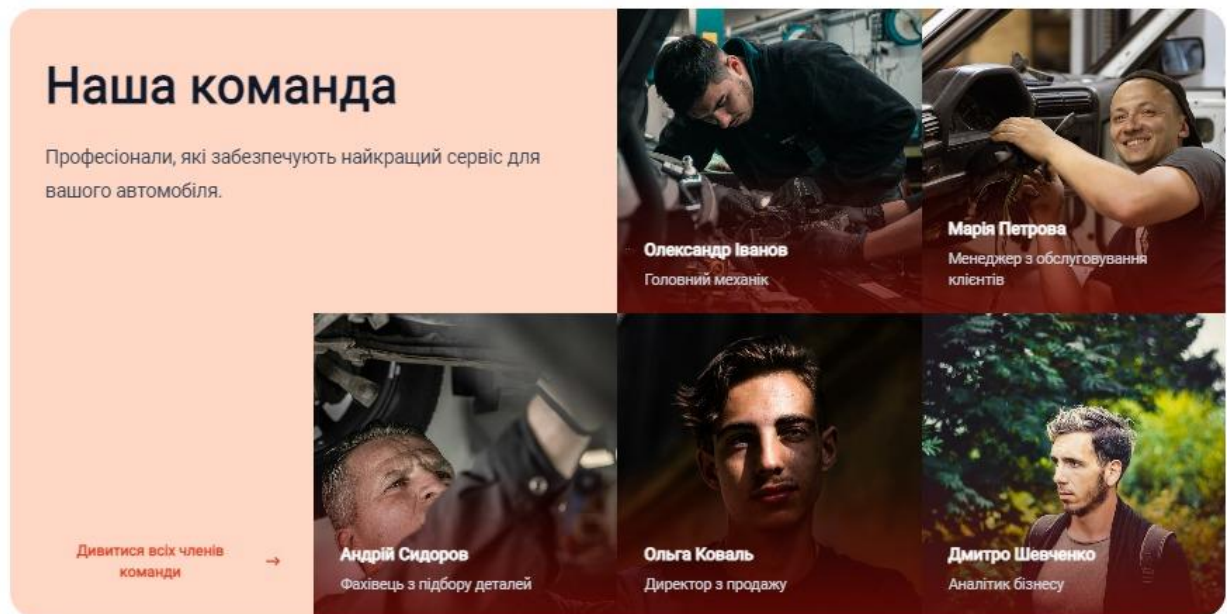


Рисунок 3.6 – Склад команди

Блок вебсайту "AutoCarePro" під назвою "Зв'яжіться з нами" містить важливу контактну інформацію та години роботи автосервісу. На сторінці вказано години роботи з понеділка по п'ятницю, з 8:00 до 17:00, а також фізичну адресу сервісу, розташовану на вулиці Шевченка, 123 у Києві, Україна. Для зв'язку з автосервісом надано два телефонних номери. Візуальний елемент сторінки включає фотографію автомобіля в русі на дорозі, що додає динаміки та акцентує на мобільності послуг, які пропонує компанія (рис. 3.7).

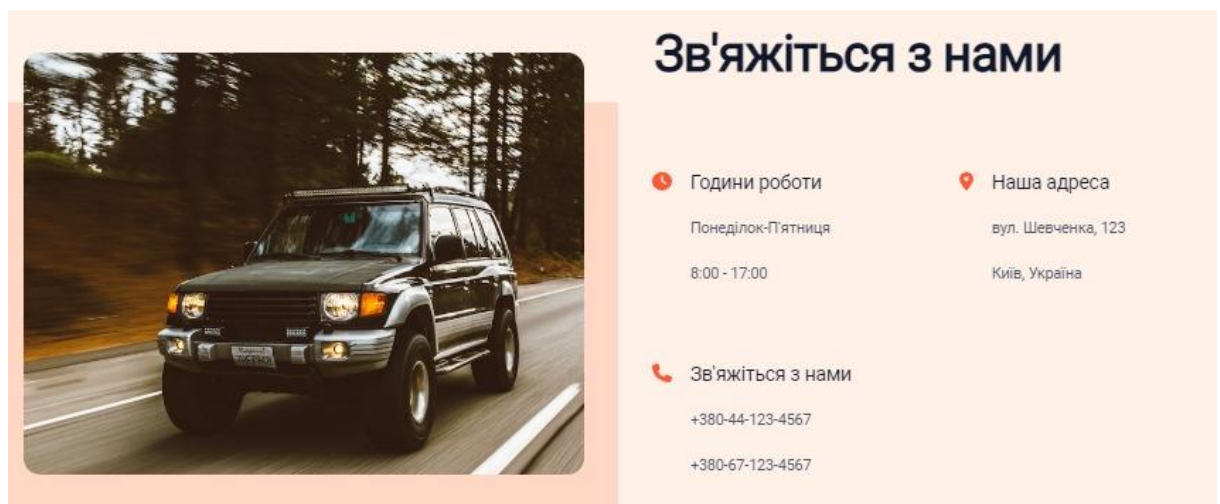


Рисунок 3.7 - Детальна сторінка конкретного сервісу з описом

Блок вебсайту "AutoCarePro" з назвою "Зв'яжіться з нами" надає користувачам декілька варіантів для взаємодії з сервісом, розроблений для зручного керування зв'язком з компанією (рис. 3.8, рис. 3.9).

Зв'яжіться з нами

AutoCarePro - ваш надійний партнер у світі автосервісу. Зв'яжіться з нами для будь-яких запитань або консультацій.

Загальні запитання

info@autocarepro.ua
+380 (44) 123-4567

Підбір деталей

parts@autocarepro.ua
+380 (44) 234-5678

Запис на ремонт

service@autocarepro.ua
+380 (44) 345-6789

Підтримка клієнтів

support@autocarepro.ua
+380 (44) 456-7890

Рисунок 3.8 – Керування зв'язком із сервісом

Зв'яжіться з AutoCarePro

Ми завжди готові допомогти вам з усіма вашими потребами в автосервісі. Звертайтеся до нас для отримання професійної допомоги.



РОБОЧІ ГОДИНИ

Понеділок-П'ятниця 9:00 до 18:00

НАША АДРЕСА

вул. Автосервісна, 123, Київ, Україна

КОНТАКТИ

+380-123-456-789
+380-987-654-321

Рисунок 3.9 – Сторінка «Про нас»

Інтерфейс форми "Запит деталей" на вебсайті автосервісу "AutoCarePro" дозволяє користувачам вводити інформацію для замовлення специфічних автомобільних деталей. Форма включає поля для введення

марки авто, моделі, року випуску та конкретних потрібних деталей. Додатково користувач може обрати дату та час, коли він бажає забрати деталі. Все це зібрано в чітко структурованому інтерфейсі, який забезпечує зручність та ефективність процесу замовлення. Button "Відправити" на кінці форми використовується для подання запиту після заповнення всіх даних (рис. 3.10).

Запит деталей

Марка авто

Модель

Рік випуску

Потрібні деталі

Дата

Час

Відправити

Рисунок 3.10 – Форма замовлення деталей

Блок вебсайту "AutoCarePro" включає інтерактивну карту для зручного розташування автосервісу. Заголовок "Розташування нашого автосервісу" служить введенням, а підзаголовок "Знайдіть нас на карті та відвідайте наш сервісний центр для професійного обслуговування вашого

автомобіля" надає користувачам додатковий контекст. Карта відображає точне місцезнаходження сервісу в Києві, дозволяючи користувачам легко знайти дорогу до центру. Інтерфейс також включає пошукову панель для введення адреси та кнопки для отримання напрямків, забезпечуючи зручність доступу та навігацію (рис. 3.11).

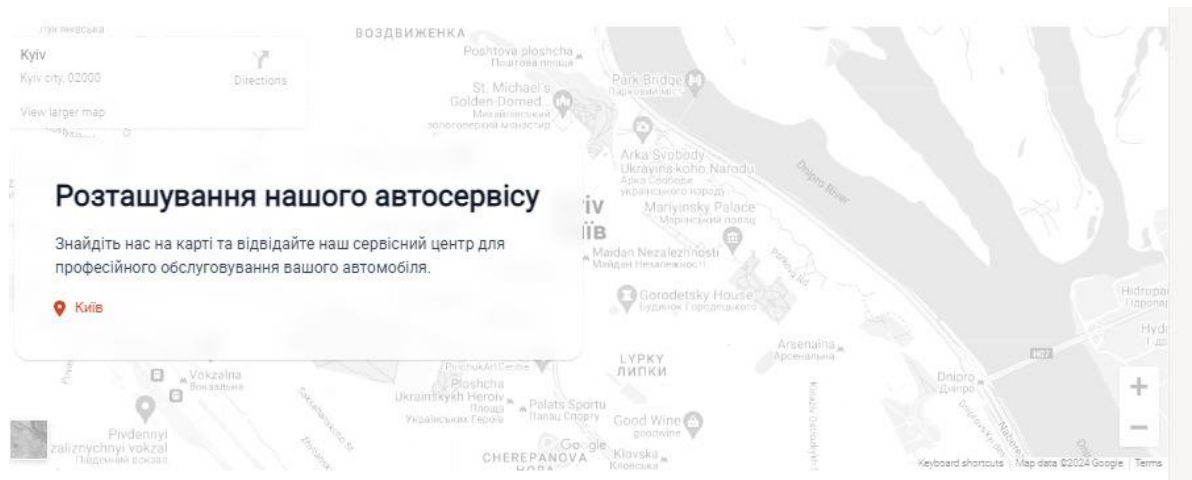


Рисунок 3.11 – Карта

На рисунку 3.12 представлено розділ відгуків на вебсайті автосервісу "AutoCarePro". Тут користувачі можуть прочитати позитивні коментарі від клієнтів, які високо оцінюють якість обслуговування та майстерність техніків. Відгуки відображають задоволення клієнтів, такі як похвала чудовому сервісу та рекомендації сервісу іншим.

Відгуки клієнтів

Дізнайтеся, що наші клієнти говорять про AutoCarePro.

Ім'я	Відгук
 Олександр Іванов 5 зірок	Чудовий сервіс! Дуже задоволений роботою майстрів.
 Марія Петрівна 4 зірки	Швидко та якісно. Рекомендую!
 Іван Коваленко 5 зірок	Найкращий автосервіс в місті!

Рисунок 3.12 – Відгуки

На рисунку 3.13 зосереджено увагу на розділі частих питань (FAQ), який відповідає на основні запитання клієнтів щодо сервісних послуг, такі як можливості онлайн-запису, види ремонтів, що виконуються, та робочі години. Цей розділ допомагає клієнтам отримати важливу інформацію швидко та ефективно, сприяючи кращому розумінню того, що вони можуть очікувати від автосервісу.

ЧАСТІ ПИТАННЯ

Часті питання

Знайдіть відповіді на найпоширеніші питання щодо наших послуг.

Чи можу я записатися на ремонт онлайн?

Так, ви можете записатися на ремонт через наш сайт, використовуючи онлайн-форму.

Які види ремонту ви пропонуєте?

Ми пропонуємо широкий спектр послуг, включаючи діагностику, ремонт двигуна, заміну шин та багато іншого.

Які ваші робочі години?

Ми працюємо з понеділка по п'ятницю з 9:00 до 18:00.

[Переглянути всі питання](#)

Рисунок 3.13 – Часті питання

Рисунок 3.14 демонструє підвал сайту, який містить посилання на основні розділи сайту, такі як команда, допомога, ресурси, та додаткові послуги. Також тут є іконки соціальних мереж, що полегшують перехід та взаємодію з автосервісом у соціальних медіа. Це зручний спосіб для користувачів залишатися зв'язаними з компанією.

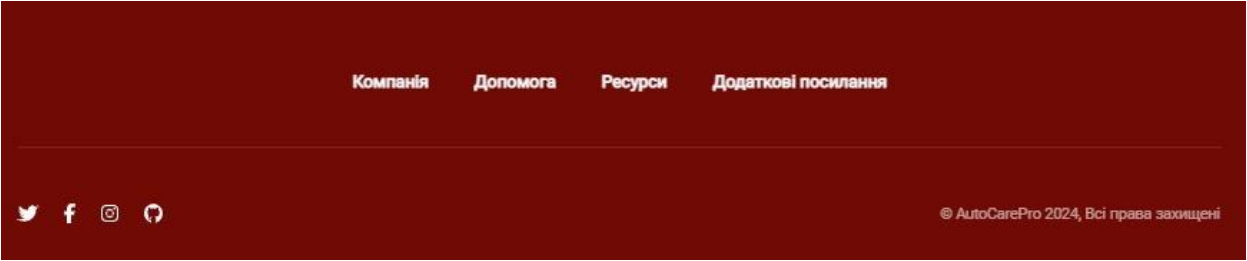


Рисунок 3.14 – Підвал вебсайту

Рисунок 3.15 демонструє таблицю доступних термінів, де клієнти можуть переглядати та обирати час для запису на сервіс. Ця таблиця включає інформацію про дату, час, вид послуги, майстра, який виконуватиме роботу, та статус доступності. Це робить процес планування візиту до сервісу простим та зручним, забезпечуючи клієнтам можливість легко зарезервувати необхідні послуги.

Доступні терміни

Перегляньте доступні терміни для запису на наші послуги.

Дата	Час	Послуга	Майстер	Статус
 2024-06-10 10:00	Заміна масла	Іван Петров	Доступно	
 2024-06-11 14:00	Технічний огляд	Олександр Коваль	Доступно	
 2024-06-12 09:00	Діагностика двигуна	Іван Іванов	Доступно	

Рисунок 3.15 – Запис на послугу в автосервіс

На рисунку 3.16 представлений інформаційний блок "Як це працює?" від автосервісу "AutoCarePro", який пояснює процес обслуговування автомобілів. На початку розташовано загальний опис, який стверджує, що сервіс покликаний забезпечити елементи, необхідні для професійного обслуговування авто

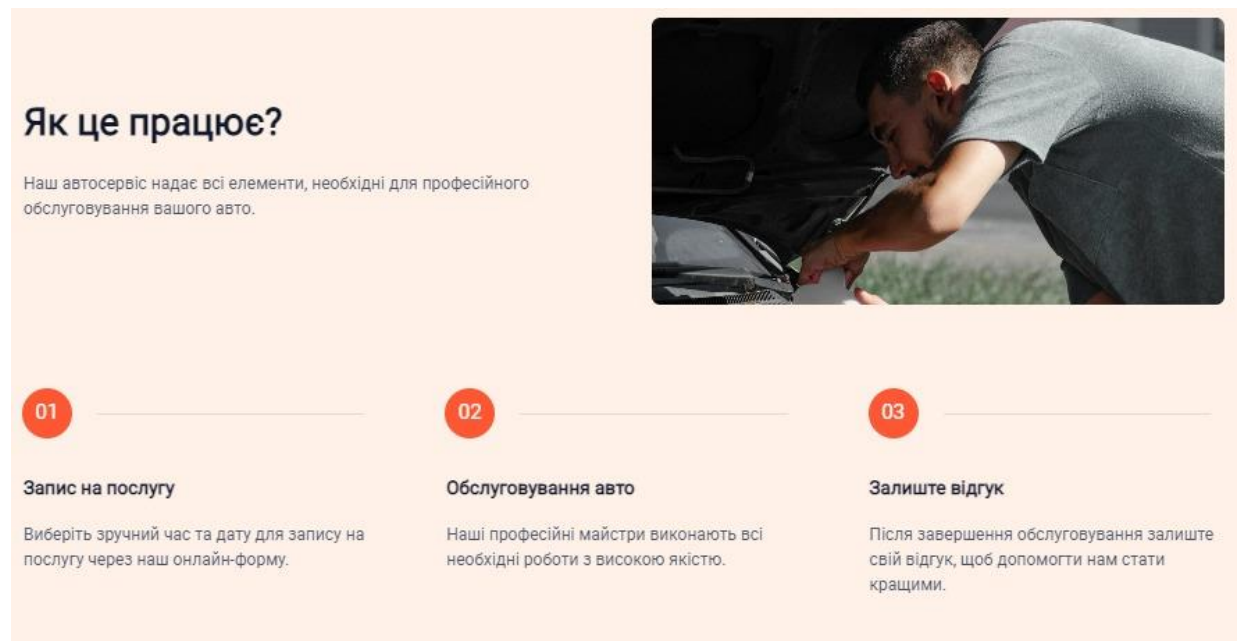


Рисунок 3.16 – Правила користування вебсайтом

3.4 Висновки до розділу

У третьому було детально розглянуто процес розробки програмного забезпечення для автосервісу, включаючи проектування та реалізацію бази даних, розробку програмного коду та опис користування вебсайтом.

При проектуванні бази даних було створено п'ять основних таблиць: "Користувачі", "Послуги", "Записи на ремонт", "Деталі" та "Відгуки". Кожна таблиця містить необхідні атрибути для зберігання відповідної інформації, забезпечуючи ефективну організацію та управління даними в системі.

Розробка програмного коду включала створення структури та функціональності вебсайту автосервісу. Було реалізовано основні сторінки, такі як головна сторінка, сторінка послуг, запис на ремонт, підбір деталей, контакти та управління записами. Використання мови програмування PHP та фреймворку Laravel дозволило ефективно працювати з базою даних та реалізувати серверну логіку. Для клієнтської частини було використано бібліотеку React, що забезпечило створення інтерактивного та динамічного інтерфейсу користувача.

При описі користування вебсайтом було представлено основні блоки та функціональні можливості системи. Головна сторінка надає загальну інформацію про автосервіс та дозволяє швидко перейти до ключових послуг. Блок з послугами детально описує доступні варіанти ремонту та обслуговування. Форма запису на ремонт дозволяє користувачам зручно обрати дату та час візиту. Форма запиту деталей надає можливість замовити необхідні запчастини. Також було розглянуто розділи з відгуками клієнтів, частими питаннями та контактною інформацією.

Таким чином, у третьому розділі було продемонстровано комплексний підхід до розробки програмного забезпечення для автосервісу, що включає проектування бази даних, реалізацію серверної та клієнтської частини, а також створення зручного та інформативного інтерфейсу користувача. Розроблений вебсайт відповідає вимогам замовника та забезпечує ефективну взаємодію між клієнтами та автосервісом. Використання сучасних технологій та дотримання кращих практик розробки гарантує високу якість, безпеку та продуктивність системи.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Тестування функціоналу вебсайту

Для забезпечення якості та надійності розробленого вебсайту автосервісу було проведено ретельне тестування його функціоналу. Тестування базувалося на методології, яка включала наступні етапи.

Планування тестування.

На цьому етапі було визначено цілі тестування, які полягали у перевірці відповідності вебсайту заявленим вимогам, забезпеченні коректної роботи всіх функцій та зручності використання для користувачів. Було розроблено детальні тестові сценарії, які охоплювали всі основні функціональні можливості вебсайту, такі як реєстрація та авторизація користувачів, запис на ремонт, підбір деталей, залишення відгуків тощо. Також були встановлені критерії успішності тестування, які включали відсутність критичних помилок, відповідність очікуваним результатам та забезпечення безперебійної роботи системи.

Тестування функціональності.

На цьому етапі було проведено ретельну перевірку відповідності функціональних можливостей вебсайту заявленим вимогам. Кожна функція, включаючи реєстрацію користувачів, авторизацію, запис на ремонт, підбір деталей та залишення відгуків, була протестована на предмет коректної роботи та відповідності очікуваним результатам. Були перевірені як позитивні, так і негативні сценарії, такі як введення коректних та некоректних даних, обробка виняткових ситуацій та граничних значень. Результати тестування показали, що всі основні функції працюють правильно та відповідають вимогам.

Тестування зручності використання (юзабіліті).

На цьому етапі було оцінено зручність та інтуїтивність інтерфейсу користувача вебсайту. Тестувальники, які виступали в ролі потенційних користувачів, перевіряли, наскільки легко орієнтуватися на сайті, знаходити

потрібну інформацію та виконувати основні дії, такі як реєстрація, запис на ремонт чи підбір деталей. Були оцінені такі аспекти, як розташування елементів на сторінках, зрозумілість навігації, читабельність тексту та загальна естетика дизайну. За результатами тестування були внесені необхідні корективи для покращення зручності використання вебсайту.

Тестування сумісності.

Для забезпечення коректного відображення та функціонування вебсайту на різних платформах було проведено тестування сумісності. Вебсайт був протестований на найпопулярніших веббраузерах, таких як Google Chrome, Mozilla Firefox, Safari та Microsoft Edge, щоб переконатися у правильному відображенні всіх елементів та відсутності візуальних чи функціональних відмінностей. Також було проведено тестування на різних пристроях, включаючи ПК, планшети та смартфони з різними розмірами екранів та операційними системами. Результати тестування підтвердили, що вебсайт коректно працює та відображається на всіх основних платформах.

Тестування безпеки.

Безпека вебсайту є критично важливим аспектом, тому на цьому етапі були ретельно перевірені механізми захисту від потенційних загроз. Було проведено тестування на наявність вразливостей, таких як SQL-ін'єкції та міжсайтовий скриптинг (XSS), щоб переконатися у неможливості несанкціонованого доступу до даних або виконання шкідливого коду. Також були перевірені механізми автентифікації та авторизації користувачів, щоб забезпечити надійний захист облікових записів та персональних даних. Результати тестування показали, що вебсайт має належний рівень безпеки та стійкість до поширених типів атак.

Тестування продуктивності.

Для забезпечення оптимальної швидкодії вебсайту було проведено тестування продуктивності при різних навантаженнях. Були оцінені такі параметри, як час завантаження сторінок, швидкість обробки запитів та стабільність роботи при великій кількості одночасних користувачів.

Використовувалися спеціальні інструменти для симуляції навантаження та вимірювання часу відгуку сервера. Результати тестування дозволили виявити потенційні вузькі місця та оптимізувати продуктивність вебсайту, забезпечивши швидке та плавне функціонування навіть при високих навантаженнях.

Регресійне тестування.

Після внесення змін до функціоналу або виправлення виявлених помилок було проведено регресійне тестування. Метою цього етапу було переконатися, що нові зміни не вплинули негативно на вже існуючий функціонал вебсайту. Були повторно виконані ключові тестові сценарії та перевірено, чи не виникли нові помилки або непередбачувані побічні ефекти. Результати регресійного тестування підтвердили, що виправлення помилок та додавання нових функцій не призвели до порушення роботи існуючих компонентів вебсайту.

Дотримання цієї методології тестування дозволило забезпечити високу якість розробленого вебсайту автосервісу (табл. 4.1).

Таблиця 4.1 – Тестування вебсайту

Етап тестування	Що було зроблено	Результати
Планування тестування	Визначені цілі тестування, розроблені детальні тестові сценарії, встановлені критерії успішності.	Чіткий план і сценарії для виявлення критичних помилок та забезпечення відповідності вебсайту вимогам.
Тестування функціональності	Проведено перевірку основних функцій вебсайту, включаючи реєстрацію, авторизацію, запис на ремонт, підбір деталей, залишення відгуків.	Усі основні функції працюють правильно та відповідають заявленим вимогам, що підтверджено позитивними та негативними тестами.
Тестування зручності використання	Оцінка інтуїтивності та зручності інтерфейсу користувача.	Внесені корективи для покращення навігації, читабельності та загальної естетики дизайну.
Тестування сумісності	Тестування вебсайту на різних браузерах та пристроях.	Вебсайт правильно відображається та функціонує на всіх основних платформах без візуальних чи функціональних відмінностей.

Продовження таблиці 4.1

Етап тестування	Що було зроблено	Результати
Тестування безпеки	Перевірка на вразливості до SQL-ін'єкцій, XSS, а також перевірка механізмів автентифікації та авторизації.	Вебсайт має належний рівень безпеки, стійкий до поширених атак, безпека облікових записів та персональних даних забезпечена.
Тестування продуктивності	Оцінка часу завантаження сторінок, швидкості обробки запитів, стабільності при великій кількості користувачів.	Оптимізація продуктивності вебсайту, швидке та плавне функціонування під високими навантаженнями.
Регресійне тестування	Повторне виконання ключових тестових сценаріїв після внесення змін для перевірки впливу на існуючий функціонал.	Зміни та виправлення не вплинули на існуючий функціонал, робота інших компонентів вебсайту не порушена.

4.2 Тестування бази даних вебсайту

Тестування бази даних є важливою складовою процесу забезпечення якості та надійності вебсайту автосервісу. Метою цього етапу було перевірити коректність структури бази даних, цілісність даних, ефективність запитів та стійкість до потенційних помилок. Тестування бази даних включало наступні кроки.

Перевірка структури бази даних.

На цьому етапі було перевірено відповідність структури бази даних вимогам проекту та логічній моделі даних. Були перевірені назви таблиць, типи даних полів, обмеження цілісності (первинні та зовнішні ключі) та індекси. Також було перевірено, чи всі необхідні таблиці та зв'язки між ними були створені коректно. Результати перевірки показали, що структура бази даних повністю відповідає вимогам та забезпечує ефективне зберігання та доступ до даних.

Тестування цілісності даних.

Цей етап передбачав перевірку коректності та узгодженості даних у базі даних. Були проведені тести на наявність дублікатів, невідповідностей між зв'язаними таблицями та порушень обмежень цілісності. Також було перевірено коректність збереження та відображення спеціальних символів та

великих обсягів даних. Результати тестування підтвердили, що дані в базі даних зберігаються коректно, без втрат чи спотворень, та відповідають встановленим правилам цілісності.

Тестування продуктивності запитів.

Для забезпечення швидкого доступу до даних та мінімізації часу відгуку вебсайту було проведено тестування продуктивності запитів до бази даних. Були створені та виконані різні типи запитів, включаючи вибірку, вставку, оновлення та видалення даних, з використанням реальних сценаріїв використання вебсайту. Були виміряні час виконання запитів та використання ресурсів сервера. Результати тестування дозволили виявити потенційні проблеми з продуктивністю та оптимізувати запити для забезпечення швидкого та ефективного доступу до даних.

Тестування стійкості до помилок.

На цьому етапі було перевірено стійкість бази даних до потенційних помилок та збоїв. Були змодельовані різні сценарії, такі як втрата з'єднання з базою даних, некоректні вхідні дані, одночасний доступ великої кількості користувачів та апаратні збої. Метою було переконатися, що база даних здатна коректно обробляти помилки, відновлюватися після збоїв та забезпечувати цілісність даних. Результати тестування показали, що база даних має високу стійкість до помилок та здатна продовжувати роботу навіть у разі виникнення нештатних ситуацій.

Тестування безпеки бази даних.

Безпека даних є критично важливою для вебсайту автосервісу, тому було проведено ретельне тестування механізмів захисту бази даних. Були перевірені права доступу користувачів, щоб переконатися, що кожен користувач має доступ лише до дозволених даних відповідно до своєї ролі. Також було протестовано захист від ін'єкцій SQL, щоб унеможливити виконання шкідливих запитів та несанкціонований доступ до даних. Результати тестування підтвердили, що база даних має надійні механізми безпеки та захищена від потенційних загроз.

Тестування бази даних вебсайту автосервісу дозволило переконатися у коректності структури, цілісності даних, продуктивності запитів, стійкості до помилок та безпеці. Завдяки ретельному тестуванню були виявлені та усунені потенційні проблеми, що забезпечило надійну та ефективну роботу бази даних. Результати тестування підтвердили, що база даних відповідає вимогам проекту та здатна забезпечити стабільну роботу вебсайту автосервісу в умовах реального використання.

4.3 Висновки до розділу

У четвертому розділі було проведено ретельне тестування розробленого програмного забезпечення для автосервісу, що включало тестування функціоналу вебсайту та тестування бази даних.

Тестування функціоналу вебсайту базувалося на методології, яка охоплювала планування тестування, тестування функціональності, зручності використання, сумісності, безпеки, продуктивності та регресійне тестування. Кожен етап тестування був спланований та виконаний для виявлення та усунення потенційних помилок, недоліків та проблем з продуктивністю. Результати тестування підтвердили високу якість розробленого програмного забезпечення та відповідність заявленим вимогам.

Тестування бази даних включало перевірку структури бази даних, цілісності даних, продуктивності запитів, стійкості до помилок та безпеки. Були перевірені назви таблиць, типи даних полів, обмеження цілісності, індекси, коректність та узгодженість даних. Також було протестовано продуктивність запитів, стійкість до потенційних помилок та збоїв, а також механізми захисту бази даних. Результати тестування підтвердили коректність структури, цілісність даних, оптимальну продуктивність запитів, стійкість до помилок та надійність механізмів безпеки бази даних.

Завдяки ретельному тестуванню функціоналу вебсайту та бази даних були виявлені та усунені потенційні проблеми, що забезпечило надійну та

ефективну роботу розробленого програмного забезпечення. Результати тестування підтвердили високу якість вебсайту автосервісу та його здатність відповідати вимогам замовника та потребам користувачів у реальних умовах експлуатації. Дотримання кращих практик тестування дозволило мінімізувати ризики виникнення помилок та збоїв у роботі системи та забезпечити її стабільність і продуктивність.

ВИСНОВКИ

В роботі бакалавра було успішно розроблено програмне забезпечення для автосервісу, що дозволяє користувачам переглядати інформацію про послуги, записуватися на ремонт та залишати заявки на підбір запчастин через вебінтерфейс. Розробка такого програмного забезпечення є актуальним завданням, яке відповідає сучасним вимогам ринку та потребам клієнтів автосервісних підприємств.

У першому розділі було проведено аналіз предметної області, розглянуто актуальність та необхідність створення сучасного вебсайту автосервісу. Аналіз існуючих веб-застосунків автосервісів дозволив виявити їх переваги та недоліки, а також сформулювати вимоги до розробки нового програмного забезпечення.

У другому розділі було обґрунтовано вибір клієнт-серверної архітектури та шаблону проектування MVC для розробки веб-застосунку. Для ефективної організації процесу розробки було обрано гнучку методологію Scrum. Також було проаналізовано та обрано оптимальний стек технологій, що включає мову програмування PHP, фреймворк Laravel, бібліотеку React, базу даних MySQL та локальний веб-сервер Open Server.

У третьому розділі було детально описано процес розробки програмного забезпечення, включаючи проектування бази даних, розробку програмного коду та опис користування веб-сайтом. База даних була спроектована з урахуванням вимог до зберігання та управління даними автосервісу. Програмний код реалізовано з використанням обраних технологій та забезпечує функціональність та зручність використання веб-сайту.

У четвертому розділі було проведено ретельне тестування розробленого програмного забезпечення, що охоплювало тестування функціоналу веб-сайту та бази даних. Результати тестування підтвердили

високу якість, надійність та продуктивність розробленої системи, а також її відповідність заявленим вимогам та потребам користувачів.

Таким чином, розроблене програмне забезпечення для автосервісу є повноцінним та ефективним рішенням, яке дозволяє підвищити якість обслуговування клієнтів, оптимізувати бізнес-процеси та забезпечити конкурентні переваги на ринку. Використання сучасних технологій, дотримання кращих практик розробки та ретельне тестування гарантують високу якість, безпеку та продуктивність системи. Впровадження такого програмного забезпечення сприятиме успішному розвитку автосервісних підприємств та підвищенню задоволеності клієнтів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гуржій А.М. Методи та засоби розробки програмного забезпечення: Навчальний посібник. / Гуржій А.М., Жуков І.М., Куш О.О. - Видавництво "Академвидав", 2020. – 312 с.
2. Barnes B. Software Engineering: An Advanced Approach. / [eds.: Barnes B., Bollinger T.] – New York: McGraw-Hill, 2021. – 456 с.
3. Василенко Т.В. Проектування систем на базі вимог: методи і практики./ Василенко Т.В. - Видавничий дім "Техніка", 2022. – 288 с.
4. Автосервіс Вишка [Електронний ресурс]. - Режим доступу: <http://www.sto.repair/ua/>
5. Сервіс Автоексперт [Електронний ресурс]. - Режим доступу: <https://service-auto-expert.com.ua/>
6. Прайм Моторс [Електронний ресурс]. - Режим доступу: <https://primemotors.com.ua/>
7. Укравто [Електронний ресурс]. - Режим доступу: <https://ukravto.ua/>
8. Software Engineering. – 10th ed. / [eds.: Sommerville I.] - Boston: Addison-Wesley, 2019. – 800 с.
9. Черняк В.С. Аналіз і тестування програмного забезпечення. / Черняк В.С., Петренко А.Г. – Львів: Видавництво "Новий Світ-2000", 2023. – 350 с.
10. Fitzgerald B. Continuous Software Engineering. / [eds.: Fitzgerald B., Stol K. J.] – New York: Springer, 2019. – 235 с.
11. Корнієнко В.М. Менеджмент програмних проєктів: Український контекст. / Корнієнко В.М. – Каравела, 2022. – 198 с.
12. McConnell S. Rapid Development: Taming Wild Software Schedules. / [eds.: McConnell S.] – Redmond, WA: Microsoft Press, 2018. – 680 с.
13. Поданенко О.П. Основи програмування на Java./ Поданенко О.П. – Видавництво "Либідь", 2021. – 440 с.

14. Martin R. Clean Code: A Handbook of Agile Software Craftsmanship. / [eds.: Martin R.] – Upper Saddle River, NJ: Prentice Hall, 2020. – 464 с.
15. Захарченко В.І. Розробка мобільних додатків. / Захарченко В.І - Видавництво "Комп'ютер", 2022. – 512 с.
16. Hunt A. The Pragmatic Programmer. – 2nd ed. / [eds.: Hunt A., Thomas D.] – Raleigh, NC: The Pragmatic Bookshelf, 2019. – 320 с.
17. Петренко А.П. Віртуалізація та хмарні обчислення. / Петренко А.П. - Видавничий дім "Академія", 2023. – 260 с.
18. Freeman E., Robson E. Head First Design Patterns. / [eds.: Freeman E., Robson E.]– Sebastopol, CA: O'Reilly Media, 2020. – 694 с.
19. Мельник М.Г. Основи кібербезпеки. / Мельник М.Г. ІТ - Академія, 2021. – 378 с.
20. Beck K. Extreme Programming Explained: Embrace Change. – 2nd ed. / [eds.: Beck K., Andres C.] Boston: Addison-Wesley, 2018. – 190 с.
21. Волкова В.Н. Сучасні технології в програмуванні. / Волкова В.Н., Семенцов О.Е. - Факт, 2021. – 324 с.
22. Fowler M. Refactoring: Improving the Design of Existing Code. – 2nd ed. / [eds.: Fowler M.] – Boston: Addison-Wesley, 2019. – 455 с.
23. Кравчук О.А. Розробка веб-додатків. / Кравчук О.А. - Юніверс, 2020. – 310 с.
24. Ambler S. W. Refactoring Databases: Evolutionary Database Design. / [eds.: Ambler S. W., Sadalage P. J.] - Boston: Addison-Wesley, 2022. – 356 с.

ДОДАТОК А
Текст програми

```
php artisan make:migration create_users_table --create=users
php artisan make:migration create_services_table --create=services
php artisan make:migration create_appointments_table --
create=appointments
php artisan make:migration create_parts_table --create=parts
php artisan make:migration create_reviews_table --create=reviews
Заповніть міграції відповідними атрибутами.
```

php

```
// database/migrations/yyyy_mm_dd_create_users_table.php
```

```
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->id();
        $table->string('username');
        $table->string('email')->unique();
        $table->string('password');
        $table->string('phone');
        $table->string('role');
        $table->timestamps();
    });
}
```

```
// database/migrations/yyyy_mm_dd_create_services_table.php
```

```
public function up()
{
    Schema::create('services', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->text('description');
        $table->decimal('price', 8, 2);
        $table->integer('duration');
        $table->timestamps();
    });
}
```

```
// database/migrations/yyyy_mm_dd_create_appointments_table.php
```

```
public function up()
{
    Schema::create('appointments', function (Blueprint $table) {
        $table->id();
        $table->foreignId('user_id')->constrained('users');
        $table->foreignId('service_id')->constrained('services');
```

```

        $table->dateTime('appointment_date');
        $table->string('status')->default('pending');
        $table->timestamps();
    });
}

```

// database/migrations/yyyy_mm_dd_create_parts_table.php

```

public function up()
{
    Schema::create('parts', function (Blueprint $table) {
        $table->id();
        $table->string('car_make');
        $table->string('car_model');
        $table->integer('year');
        $table->string('part_name');
        $table->boolean('availability')->default(true);
        $table->decimal('price', 8, 2);
        $table->timestamps();
    });
}

```

// database/migrations/yyyy_mm_dd_create_reviews_table.php

```

public function up()
{
    Schema::create('reviews', function (Blueprint $table) {
        $table->id();
        $table->foreignId('user_id')->constrained('users');
        $table->foreignId('service_id')->constrained('services');
        $table->tinyInteger('rating');
        $table->text('comment');
        $table->timestamps();
    });
}

```

```

php artisan make:model User
php artisan make:model Service
php artisan make:model Appointment
php artisan make:model Part
php artisan make:model Review

```

```

php artisan make:controller UserController
php artisan make:controller ServiceController
php artisan make:controller AppointmentController
php artisan make:controller PartController
php artisan make:controller ReviewController

```

```
php artisan make:controller AuthController
```

```
use App\Http\Controllers\UserController;
use App\Http\Controllers\ServiceController;
use App\Http\Controllers\AppointmentController;
use App\Http\Controllers\PartController;
use App\Http\Controllers\ReviewController;
use App\Http\Controllers\AuthController;

Route::get('/', [ServiceController::class, 'index'])->name('home');
Route::get('/services', [ServiceController::class, 'list'])->name('services');
Route::get('/booking', [AppointmentController::class, 'create'])->
>name('booking');
Route::post('/booking', [AppointmentController::class, 'store']);
Route::get('/parts', [PartController::class, 'create'])->name('parts');
Route::post('/parts', [PartController::class, 'store']);
Route::get('/contact', [ContactController::class, 'index'])->name('contact');
Route::get('/reviews', [ReviewController::class, 'index'])->name('reviews');
Route::post('/reviews', [ReviewController::class, 'store']);
Route::get('/login', [AuthController::class, 'showLoginForm'])->
>name('login');
Route::post('/login', [AuthController::class, 'login']);
Route::get('/register', [AuthController::class, 'showRegisterForm'])->
>name('register');
Route::post('/register', [AuthController::class, 'register']);
7
```

```
<!-- resources/views/welcome. .php -->
@extends('layouts.app')

@section('content')
    <h1>Welcome to AutoService</h1>
    <nav>
        <ul>
            <li><a href="{{ route('services') }}">Services</a></li>
            <li><a href="{{ route('booking') }}">Book an
Appointment</a></li>
            <li><a href="{{ route('parts') }}">Order Parts</a></li>
            <li><a href="{{ route('contact') }}">Contact Us</a></li>
            <li><a href="{{ route('reviews') }}">Reviews</a></li>
        </ul>
    </nav>
@endsection

<!-- resources/views/services/list. .php -->
```

```

@extends('layouts.app')

@section('content')
    <h1>Our Services</h1>
    <ul>
        @foreach($services as $service)
            <li>
                <h2>{{ $service->name }}</h2>
                <p>{{ $service->description }}</p>
                <p>Price: ${{ $service->price }}</p>
                <p>Duration: {{ $service->duration }} minutes</p>
            </li>
        @endforeach
    </ul>
@endsection

<!-- resources/views/appointments/create. .php -->
@extends('layouts.app')

@section('content')
    <h1>Book an Appointment</h1>
    <form action="{{ route('booking') }}" method="POST">
        @csrf
        <label for="service_id">Service:</label>
        <select name="service_id" id="service_id">
            @foreach($services as $service)
                <option value="{{ $service->id }}">{{ $service->name
}}</option>
            @endforeach
        </select>
        <label for="appointment_date">Date and Time:</label>
        <input type="datetime-local" name="appointment_date"
id="appointment_date">
        <button type="submit">Book Now</button>
    </form>
@endsection

<!-- resources/views/parts/create. .php -->
@extends('layouts.app')

@section('content')
    <h1>Order Parts</h1>
    <form action="{{ route('parts') }}" method="POST">
        @csrf
        <label for="car_make">Car Make:</label>
        <input type="text" name="car_make" id="car_make">

```

```
        <label for="car_model">Car Model:</label>
        <input type="text" name="car_model" id="car_model">
        <label for="year">Year:</label>
        <input type="number" name="year" id="year">
        <label for="part_name">Part Name:</label>
        <input type="text" name="part_name" id="part_name">
        <button type="submit">Submit</button>
    </form>
@endsection
```

```
users = db['users']
```

```
user_schema = {
    "_id": "ObjectId",
    "username": "String",
    "password": "String",
    "email": "String",
    "phone": "String",
    "role": "String",
    "createdAt": "Date"
}
```

```
services = db['services']
```

```
service_schema = {
    "_id": "ObjectId",
    "name": "String",
    "description": "String",
    "price": "Number",
    "duration": "Number",
    "createdAt": "Date",
    "updatedAt": "Date"
}
```

```
appointments = db['appointments']
```

```
appointment_schema = {
    "_id": "ObjectId",
    "userId": "ObjectId",
    "serviceId": "ObjectId",
    "appointmentDate": "Date",
    "status": "String",
    "createdAt": "Date",
    "updatedAt": "Date"
}
```

```

parts = db['parts']

part_schema = {
    "_id": "ObjectId",
    "carMake": "String",
    "carModel": "String",
    "year": "Number",
    "partName": "String",
    "availability": "Boolean",
    "price": "Number",
    "createdAt": "Date",
    "updatedAt": "Date"
}

reviews = db['reviews']

review_schema = {
    "_id": "ObjectId",
    "userId": "ObjectId",
    "serviceId": "ObjectId",
    "rating": "Number",
    "comment": "String",
    "createdAt": "Date"
}

from bson.objectid import ObjectId

# Create
def create_user(username, password, email, phone, role):
    user = {
        "username": username,
        "password": password,
        "email": email,
        "phone": phone,
        "role": role,
        "createdAt": datetime.now()
    }
    return users.insert_one(user).inserted_id

# Read
def get_user(user_id):
    return users.find_one({"_id": ObjectId(user_id)})

# Update

```

```

def update_user(user_id, update_fields):
    update_fields["updatedAt"] = datetime.now()
    return users.update_one({"_id": ObjectId(user_id)}, {"$set":
update_fields})

# Delete
def delete_user(user_id):
    return users.delete_one({"_id": ObjectId(user_id)})

# Create
def create_service(name, description, price, duration):
    service = {
        "name": name,
        "description": description,
        "price": price,
        "duration": duration,
        "createdAt": datetime.now(),
        "updatedAt": datetime.now()
    }
    return services.insert_one(service).inserted_id

# Read
def get_service(service_id):
    return services.find_one({"_id": ObjectId(service_id)})

# Update
def update_service(service_id, update_fields):
    update_fields["updatedAt"] = datetime.now()
    return services.update_one({"_id": ObjectId(service_id)}, {"$set":
update_fields})

# Delete
def delete_service(service_id):
    return services.delete_one({"_id": ObjectId(service_id)})

# Create
def create_appointment(user_id, service_id, appointment_date, status):
    appointment = {
        "userId": ObjectId(user_id),
        "serviceId": ObjectId(service_id),
        "appointmentDate": appointment_date,
        "status": status,
        "createdAt": datetime.now(),

```

```

        "updatedAt": datetime.now()
    }
    return appointments.insert_one(appointment).inserted_id

# Read
def get_appointment(appointment_id):
    return appointments.find_one({"_id": ObjectId(appointment_id)})

# Update
def update_appointment(appointment_id, update_fields):
    update_fields["updatedAt"] = datetime.now()
    return appointments.update_one({"_id": ObjectId(appointment_id)},
{"$set": update_fields})

# Delete
def delete_appointment(appointment_id):
    return appointments.delete_one({"_id": ObjectId(appointment_id)})

# Create
def create_part(car_make, car_model, year, part_name, availability, price):
    part = {
        "carMake": car_make,
        "carModel": car_model,
        "year": year,
        "partName": part_name,
        "availability": availability,
        "price": price,
        "createdAt": datetime.now(),
        "updatedAt": datetime.now()
    }
    return parts.insert_one(part).inserted_id

# Read
def get_part(part_id):
    return parts.find_one({"_id": ObjectId(part_id)})

# Update
def update_part(part_id, update_fields):
    update_fields["updatedAt"] = datetime.now()
    return parts.update_one({"_id": ObjectId(part_id)}, {"$set":
update_fields})

# Delete
def delete_part(part_id):

```

```
return parts.delete_one({"_id": ObjectId(part_id)})
```

```
# Create
```

```
def create_review(user_id, service_id, rating, comment):
```

```
    review = {  
        "userId": ObjectId(user_id),  
        "serviceId": ObjectId(service_id),  
        "rating": rating,  
        "comment": comment,  
        "createdAt": datetime.now()  
    }
```

```
    return reviews.insert_one(review).inserted_id
```

```
# Read
```

```
def get_review(review_id):
```

```
    return reviews.find_one({"_id": ObjectId(review_id)})
```

```
# Update
```

```
def update_review(review_id, update_fields):
```

```
    update_fields["updatedAt"] = datetime.now()
```

```
    return reviews.update_one({"_id": ObjectId(review_id)}, {"$set":  
update_fields})
```

```
# Delete
```

```
def delete_review(review_id):
```

```
    return reviews.delete_one({"_id": ObjectId(review_id)})
```

ДОДАТОК Б
Слайди презентації

Розробка програмного забезпечення для автосервісу.

- Виконав студент групи КНТ-211сп
- Керівник проєкту, к.т.н., доцент

Павло ЛІПЕЙКО

Олександр СТЕПАНЕНКО

Рисунок Б.1 – Слайд 1

Мета та задачі

Метою даної дипломної роботи є розробка вебсайту для автосервісу, який би задовольняв потреби сучасних автовласників у швидкому та ефективному сервісі. Вебсайт повинен включати функціональні можливості для перегляду інформації про послуги, онлайн-запис на ремонт та систему заявок на підбір запчастин.

Задачі, які треба вирішити:

- аналіз потреб користувачів та формування технічних вимог до програмного забезпечення;
- проектування архітектури вебсайту, забезпечення його масштабованості та безпеки;
- розробка функціоналу для перегляду інформації про послуги, запису на ремонт та подання заявок на деталі;
- тестування та оптимізація вебсайту для забезпечення високої продуктивності та користувацького досвіду.

Рисунок Б.2 – Слайд 2

Аналіз вебзастосунків автосервісу

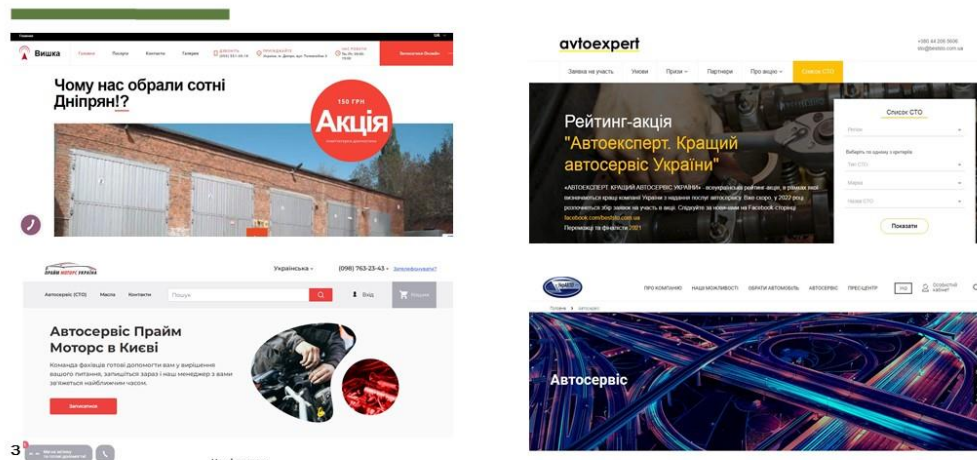
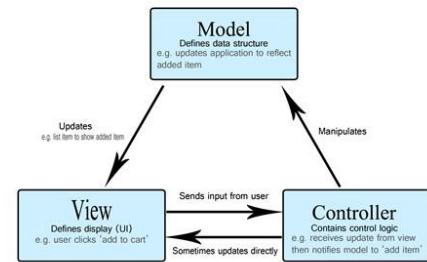


Рисунок Б.3 – Слайд 3

Порівняння вебсайтів автосервісів

Параметр	Автосервіс Вишка	Авто Експерт	Прайм Моторс	Укравто
Модулі	Інформація про послуги, Онлайн-запис на ремонт, Замовлення запчастин, Блоги та новини	Каталог послуг, Система Відгуки клієнтів, Контактна інформація	Онлайн-запис, Інформація про послуги, Відгуки клієнтів, Блоги	Каталог послуг, Система запису, Блог, Форум
Переваги	Зручний інтерфейс, Швидкий доступ до інформації, Можливість онлайн-запису	Простий і зрозумілий дизайн, Легка навігація, Актуальна інформація	Швидкий час завантаження, Адаптивний дизайн, Наявність чату підтримки	Інтерактивний дизайн, Наявність форуму, Можливість онлайн-консультацій
Недоліки/язвільі	Відсутність мобільної версії, Повільне завантаження сторінок	Відсутність інтеграції з соціальними мережами, Обмежені можливості пошуку	Відсутність детального каталогу запчастин, Проблеми з безпекою	Складний процес реєстрації, Обмежені можливості для мобільних користувачів
Зручність сайту	Висока	Висока	Висока	Середня
Тривалість існування	Понад 5 років	Близько 3 років	Понад 4 роки	Більше 6 років
Відповідність до сучасних технологій	Висока	Висока	Висока	Висока

Рисунок Б.4 – Слайд 4



Архітектура програмного забезпечення

Рисунок Б.5 – Слайд 5

Вибрані технології для розробки програмного забезпечення автосервісу

Технологія	Категорія	Роль у проєкті	Основні переваги
PHP	Мова програмування	Серверна логіка	Широка використаність, простий синтаксис, швидка розробка, велика екосистема фреймворків та бібліотек.
Laravel	Фреймворк (PHP)	Розробка бекенду	Організація коду, ORM Eloquent для роботи з базами даних, шаблонізатор Blade, вбудовані механізми автентифікації та авторизації.
React	Бібліотека (JavaScript)	Розробка клієнтської частини	Побудова інтерактивних UI, ефективне оновлення компонентів, висока швидкість розробки і зручність підтримки.
MySQL	Система управління базами даних	Зберігання даних	Надійність, швидкість, масштабованість, добра інтеграція з PHP та Laravel.
OpenServer	Локальний вебсервер	Розробка і тестування	Легке налаштування локального середовища, включає Apache, MySQL, PHP.

Рисунок Б.6 – Слайд 6

Таблиця 3.1 - Користувачі

Атрибут	Тип	Опис
_id	ObjectId	Унікальний ідентифікатор користувача
username	String	Ім'я користувача
password	String	Хеш пароля
email	String	Електронна адреса користувача
phone	String	Телефонний номер користувача
role	String	Роль користувача ('customer' або 'admin')
createdAt	Date	Дата створення запису

Таблиця 3.2 - Послуги

Атрибут	Тип	Опис
_id	ObjectId	Унікальний ідентифікатор послуги
name	String	Назва послуги
description	String	Опис послуги
price	Number	Вартість послуги
duration	Number	Тривалість послуги в хвилинах
createdAt	Date	Дата створення запису
updatedAt	Date	Дата оновлення запису

Таблиця 3.3- Записи на ремонт

Атрибут	Тип	Опис
_id	ObjectId	Унікальний ідентифікатор запису
userId	ObjectId	ID користувача, що зробив запис
serviceId	ObjectId	ID послуги
appointmentDate	Date	Дата і час запису
status	String	Статус запису ('pending', 'confirmed', 'cancelled')
createdAt	Date	Дата створення запису
updatedAt	Date	Дата оновлення запису

Таблиця 3.4- Деталі

Атрибут	Тип	Опис
_id	ObjectId	Унікальний ідентифікатор деталі
carMake	String	Марка автомобіля
carModel	String	Модель автомобіля
year	Number	Рік випуску автомобіля
partName	String	Назва деталі
availability	Boolean	Чи є деталь в наявності
price	Number	Ціна деталі
createdAt	Date	Дата створення запису
updatedAt	Date	Дата оновлення запису

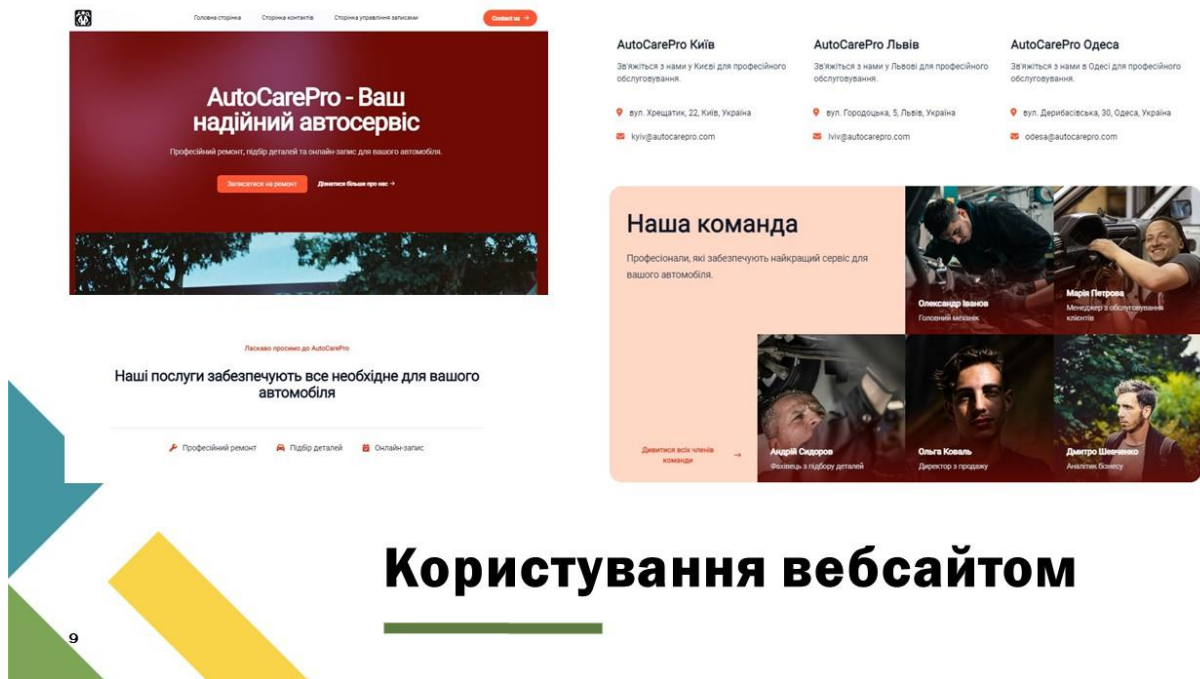
Проектування та розробка бази даних

Рисунок Б.7 – Слайд 7

Структура та функціональність вебсайту автосервісу

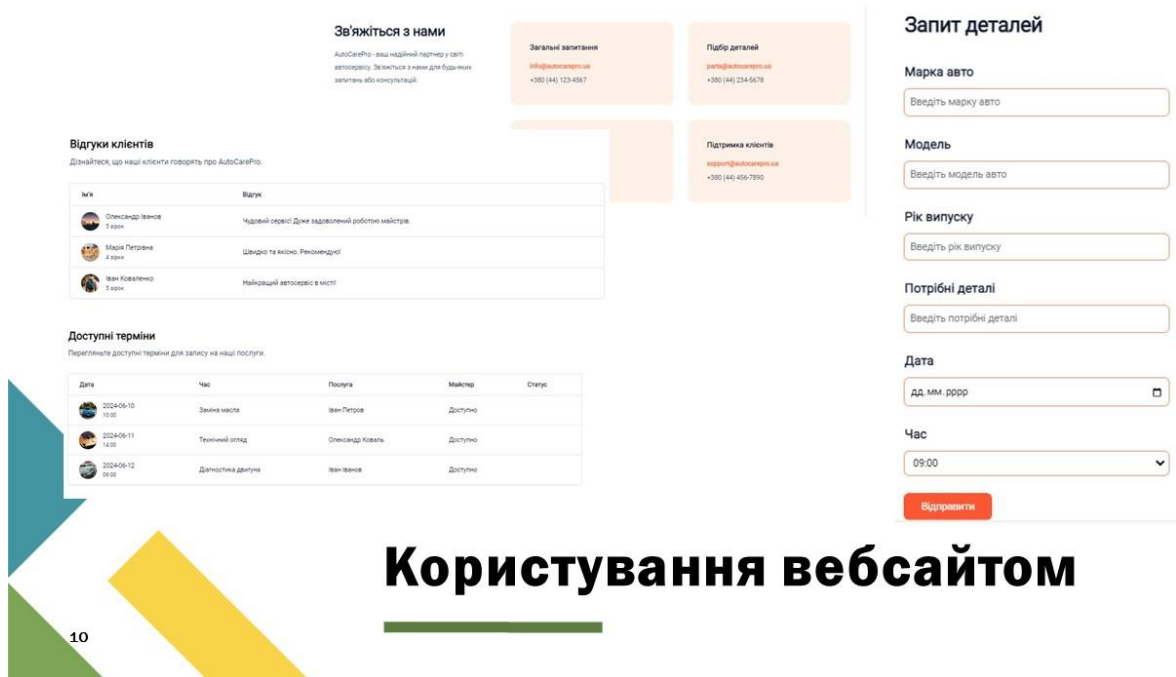
Назва файлу	Опис
index.php	Головна сторінка сайту, що включає меню навігації з основними розділами: Про нас, Послуги, Запис на ремонт, Підбір деталей та Контакти. Служить вхідною точкою для користувачів, які бажають дізнатися більше про сервіс та можливості автосервісу.
services.php	Сторінка з детальним описом послуг, що надаються автосервісом. Забезпечує інформацію про кожну послугу та можливість запису на конкретну послугу.
booking.php	Сторінка для онлайн-запису на ремонт, де користувач може вибрати дату та час візиту. Також відображає інформацію про вільні терміни.
parts_request.php	Сторінка для запиту деталей з формою, куди вводяться дані про марку авто, модель, рік випуску та потрібні деталі. Включає інформацію про наявність та можливість замовлення деталей.
contact.php	Сторінка з контактною інформацією автосервісу, включаючи адресу, телефон, електронну пошту та графік роботи.
reviews.php	Сторінка відгуків, де відображається список відгуків клієнтів із оцінками. Включає форму для залишення відгуку.
manage_bookings.php	Сторінка управління записами, на якій можна переглядати всі зареєстровані записи, а також скасувати або змінити час візиту.
login.php	Сторінка для авторизації користувачів, що включає форму входу.
register.php	Сторінка для реєстрації нових користувачів, яка містить форму реєстрації.

Рисунок Б.8 – Слайд 8



Користування вебсайтом

Рисунок Б.9 – Слайд 9



Користування вебсайтом

Рисунок Б.10 – Слайд 10

Тестування вебсайту автосервісу

11

Етап тестування	Що було зроблено	Результати
Планування тестування	Визначені цілі тестування, розроблені детальні тестові сценарії, встановлені критерії успішності.	Чіткий план і сценарії для виявлення критичних помилок та забезпечення відповідності вебсайту вимогам.
Тестування функціональності	Проведено перевірку основних функцій вебсайту, включаючи реєстрацію, авторизацію, запис на ремонт, підбір деталей, залишення відгуків.	Усі основні функції працюють правильно та відповідають заявленим вимогам, що підтверджено позитивними та негативними тестами.
Тестування зручності використання	Оцінка інтуїтивності та зручності інтерфейсу користувача.	Внесені корективи для покращення навігації, читабельності та загальної естетики дизайну.
Тестування сумісності	Тестування вебсайту на різних браузерях та пристроях.	Вебсайт правильно відображається та функціонує на всіх основних платформах без візуальних чи функціональних відмінностей.
Тестування безпеки	Перевірка на вразливості до SQL-ін'єкцій, XSS, а також перевірка механізмів автентифікації та авторизації.	Вебсайт має належний рівень безпеки, стійкий до поширених атак, безпека облікових записів та персональних даних забезпечена.
Тестування продуктивності	Оцінка часу завантаження сторінок, швидкості обробки запитів, стабільності при великій кількості користувачів.	Оптимізація продуктивності вебсайту, швидке та плавне функціонування під високими навантаженнями.
Регресійне тестування	Повторне виконання ключових тестових сценаріїв після внесення змін для перевірки впливу на існуючий функціонал.	Зміни та виправлення не вплинули на існуючий функціонал, робота інших компонентів вебсайту не порушена.

Рисунок Б.11 – Слайд 11

Висновки

У дипломній роботі успішно розроблено програмне забезпечення для автосервісу, яке дозволяє користувачам переглядати інформацію про послуги, записуватися на ремонт та залишати заявки на підбір запчастин через вебінтерфейс.

Основні задачі, які були вирішені в ході роботи:

- Проаналізовано предметну область та сформульовано вимоги до розробки програмного забезпечення.
- Обґрунтовано вибір архітектури, методології та технологій розробки.
- Спроектовано та реалізовано базу даних для зберігання та управління даними автосервісу.
- Розроблено функціональний та зручний у використанні вебсайт автосервісу.
- Проведено ретельне тестування функціоналу вебсайту та бази даних для забезпечення якості та надійності системи.

Мета дипломної роботи, яка полягала в розробці програмного забезпечення для автосервісу, що відповідає сучасним вимогам ринку та потребам клієнтів, була успішно досягнута.

12

Рисунок Б.12 – Слайд 12