МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНИЙ БАКАЛАВРСЬКИЙ ПРОЕКТ

на тему

ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВИЗНАЧЕННЯ ПОТРЕБИ В КОМПЛЕКТУЮЧИХ ДЛЯ РЕМОНТУ АВТОМОБІЛІВ

здобувача Кукояшного Кирила Олеговича _____

Науковий керівник:
к.е.н., доцент
_____Помазун О.М.

Проект допущений до захисту перед екзаменаційною комісією з атестації здобувачів вищої освіти
завідувач кафедри:
к.е.н., доцент.
_____Тішков Б.О.

Київ 2024

Міністерство освіти і науки України
Київський національний економічний університет імені Вадима Гетьмана
Навчально-науковий інститут «Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО_ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

ПОГОДЖЕНО:
Керівник проектної групи(гарант)
освітньо-професійної програми
_____Іванченко Г.Ф.
"____" _____2024 р.

ЗАТВЕРДЖУЮ:
Завідувач кафедри
_____Тішков Б.О.
"____" _____2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти ***Кукояшного Кирила Олеговича***

очної (денної) форми навчання

на підготовку кваліфікаційного бакалаврського проекту
на тему: *«Проектування інформаційної системи для визначення потреби в комплектуючих для ремонту автомобілів»*

Тему затверджено наказом ректора Університету від «11» березня 2024 р. № 529-ст.

Кваліфікаційний бакалаврський проект виконується на матеріалах
інтернет-ресурсів, технічної документації

План кваліфікаційного бакалаврського проекту

Розділ І Характеристика та аналіз предметної області

Розділ ІІ Розробка вимог і моделювання інформаційної системи

Розділ ІІІ Проєктування та реалізація прототипу, та модулів системи

Об'єкт дослідження діяльність автосервісних підприємств пов'язаних з використанням мережі інтернет для надання послуг з ремонту та обліку ремонтних робіт автомобілів

Предмет дослідження теорія та методи аналізу, проєктування, прототипування та реалізації інформаційної системи для визначення потреби в комплектуючих для ремонту автомобілів

Мета кваліфікаційного бакалаврського проекту Проєктування та створення прототипу та реалізація частини модулів системи для визначення потреби в комплектуючих для ремонту автомобілів, ціль якого полегшення процесу запису на ремонт та ведення обліку робіт

Конкретні завдання, які студент повинен виконати для досягнення поставленої мети:

У розділі I Описати предметну область надання послуг ремонту автомобілів у вигляді автосервісів. Навести характеристику об'єкта дослідження. Подати результати аналізу літературних джерел та існуючих варіантів систем з надання послуг ремонту авто на основі яких визначити бажаний функціонал.

У розділі II Встановити бізнес вимоги, функціональні та нефункціональні вимоги до системи. За допомогою CASE-засобів створити моделі поведінки та структури системи у вигляді діаграм. Навести характеристику задачі та запропонувати її вирішення. Визначити вхідну та вихідну інформацію інформаційної системи

У розділі III Провести опис використаних програмних продуктів, побудувати схеми технічного забезпечення функціонування системи. Спроєктувати і створити базу даних, даталогічну та інфологічні схеми бази даних, обґрунтувати вибір СКБД та іншого програмного забезпечення. Описати рекомендовані технології лінії зв'язку щодо розміщення системи та бази даних. Розробити прототип інформаційної системи та реалізувати модулі системи.

Завдання підготував
науковий керівник _____ **Помазун Оксана Миколаївна**

"17" березня 2024 р.

Завдання одержав
студент _____ **Кукояшний Кирило Олегович**

"17" березня 2024 р.

Відгук
про кваліфікаційний бакалаврський проект
здобувача навчально-наукового інституту
«Інститут інформаційних технологій в економіці»
освітньо-професійної програми
«Комп'ютерні науки»

Кукояшного Кирила Олеговича

на тему
«Проектування інформаційної системи для визначення потреби в комплектуючих для ремонту автомобілів»

1. Актуальність теми: тема проекту «Проектування інформаційної системи для визначення потреби в комплектуючих для ремонту автомобілів» є досить актуальною та потрібною. Ремонт автомобілів дуже поширене явище що є невід'ємною частиною обов'язків власника авто. Більшість сервісів надання послуг з ремонту авто існують без веб-ресурсів і дізнатися інформацію про послуги з ремонту складно. Проєктування такої системи дозволить власникам авто мати сервіс який буде відображати на веб-ресурсі історію заявок на ремонт та бачити в чому була проблема з автомобілем.

2. Позитивні риси кваліфікаційного бакалаврського проекту: автор провів якісне порівняння існуючих систем з ремонту авто та виокремив найнеобхідніші функції з метою проєктування власної системи

3. Наявність самостійних розробок автора автором було представлено інтерфейс системи який при мінімальній кількості тексту дозволяє зрозуміти який функціонал надає та чи інша частина системи опираючись на піктограми.

4. Цінність теоретичних висновків та практичних рекомендацій: проект містить теоретичну інформацію стосовно проєктування систем які можна використати для проєктування систем іншого типу. Практичні рекомендації містять інформацію про апаратне та програмне забезпечення.

5. Наявність недоліків: спроєктована система має можливість впровадження більш сучасного функціоналу що може бути інтегрований в майбутньому, хоча автор зазначив що в перспективі є можливість розширення, але не вказав конкретику, реалізовано не весь функціонал системи.

6. Загальна оцінка кваліфікаційного бакалаврського проекту та його допущення до захисту перед ЕК:

Науковий керівник
_____ доцент, к.е.н. Помазун О.М.
(підпис)

" 6 " червня 2024 р.

Рецензія
на кваліфікаційний бакалаврський проект
Кукояшного Кирила Олеговича

тема
«Проектування інформаційної системи для визначення потреби в комплектуючих для ремонту автомобілів»

Актуальність теми кваліфікаційного бакалаврського проекту і доцільність його розроблення. Тема бакалаврського проекту не є надзвичайно важливою, автосервісів існує багато, але сервісів з веб-ресурсами де можна додавати власне авто в кабінет мало. Звідси і випливає доцільність створення ще однієї системи автосервісу з власним кабінетом

Якість проведеного дослідження. Автор детально оглянув існуючі системи та вивів вимоги. На основі цих досліджень автором було спроєктовано всі необхідні супровідні діаграми, схеми та графіки. Зміг обрати забезпечення для власної розробки системи та запропонувати для майбутнього розміщення інформаційної системи

Позитивні риси кваліфікаційного бакалаврського проекту. Автор доступно описав інформаційні процеси в системі та провів детальне дослідження функціоналу існуючих систем. Перед реалізацією компонентів системи створив прототип в якості опорної точки.

Зауваження. Автором було виконано проектування системи з точки зору користувача, це передбачає що система має і адміністративну частину, яка має бути доступна з того ж самого веб-ресурсу для доцільності створення системи в цілому.

Практична значимість висновків і рекомендацій Результати отримані в ході виконання даного проекту можуть бути використані для проєктування інших систем або створення ще однієї системи в якості конкурента, а роботу з проектування даної системи використовувати як опорну точку для створення власних вимог, розширення функціоналу та впровадження нових технологій, можливо навіть і автоматизації більшої частини функціоналу. Впровадження даної системи та її розширення можуть викликати потенційну конкуренцію сервісам що надають послуги на веб-ресурсах, що мають кабінет користувача та можливість додавати своє авто чи дивитись статус звернень по ремонту чи обслуговуванню. Це в свою чергу, змусить компанії почати впроваджувати нові технології, це сприяє розвитку конкуренції та технологій, тому робота заслуговує на оцінку «Відмінно»

Місце роботи та посада рецензента Український державний університет імені Михайла Драгоманова, доцент кафедри інформаційних технологій і програмування факультету математики, інформатики та фізики

к.пед.н. доцент _____ Умрик М.А.
(підпис) (ПІБ)

Підпис Умрик М.А.
ЗАВІРЯЮ завідувачка канцелярією Українського державного університету імені Михайла Драгоманова
06 2024 р.

АНОТАЦІЯ

кваліфікаційного бакалаврського проекту студента 4 курсу
Навчально-наукового інституту «Інститут інформаційних технологій в економіці»
Кукояшного Кирила Олеговича, виконаної на тему:
«Проектування інформаційної системи для визначення потреби в комплектуючих для ремонту автомобілів»
Київ: кафедра інформаційних систем в економіці, 2024р.

Кваліфікаційний бакалаврський проект присвячений актуальній проблемі системі визначення потреб у комплектуючих для ремонту авто, яку пропонується розв’язувати та удосконалювати з використанням сучасних інформаційних технологій.

Кваліфікаційний бакалаврський проект складається з трьох розділів, логічно пов’язаних між собою.

В першому розділі дана характеристика предметної галузі й об’єкта дослідження, наведено аналіз задач, що розв’язуються, а також наводиться перелік існуючих програмних засобів даної предметної галузі.

Другий розділ є проектним і присвячений обґрунтуванню методу проєктування системи, розробленню її архітектури, виконанню постановки та розробленню алгоритму розв’язання задач.

Третій розділ – конструктивний. Тут наведено інформаційне, технічне програмне та організаційне забезпечення для підсистеми системи для визначення потреби в комплектуючих для ремонту автомобілів. Висвітлені питання організації інформаційного забезпечення: розроблена структура інформаційного забезпечення, описані та наведені в додатках форми вхідних та вихідних документів. Обґрунтований комплекс технічних засобів, а також програмне забезпечення, що використовується для створення системи. Вказані структури інформаційних масивів, які використовуються під час вирішення задачі.

Висновки містять рекомендації щодо доцільності розроблення та впровадження системи для визначення потреби в комплектуючих для ремонту автомобілів.

РЕФЕРАТ

Кваліфікаційний бакалаврський проект містить 96 сторінок, 9 таблиць, 67 рисунків, список літератури з 30 джерел посилань, 2 додатки.

«ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВИЗНАЧЕННЯ ПОТРЕБИ В КОМПЛЕКТУЮЧИХ ДЛЯ РЕМОНТУ АВТОМОБІЛІВ»

Перелік ключових слів: інформаційна система, проєктування, прототип, ремонт авто, діаграма, огляд.

Предметом дослідження є теорія та методи аналізу, проєктування, прототипування та реалізації системи визначення потреби в комплектуючих для ремонту автомобілів.

Об'єктом дослідження є діяльність автосервісних підприємств пов’язаних з використанням мережі інтернет для надання послуг з ремонту та обліку ремонтних робіт автомобілів.

Мета кваліфікаційного бакалаврського проекту полягає в проектуванні та створенні прототипу системи, та модулів системи для визначення потреби в комплектуючих для ремонту автомобілів, ціль якого полегшення процесу запису на ремонт та ведення обліку робіт.

Завданням кваліфікаційного бакалаврського проекту є дослідження діяльності сервісів що надають послуги з ремонту авто та встановлення вимог, функціоналу для проектування системи для визначення потреби в комплектуючих для ремонту автомобілів.

Результатом досягнутим в процесі роботи стало створення необхідної документації для розробки системи для визначення потреби в комплектуючих для ремонту автомобілів та прототипу інтерфейсу інформаційної системи.

Одержані результати можуть бути використані в комерційних цілях з метою створення системи та за рахунок чого, конкуренції на ринку надання послуг з ремонту авто. Можлива адаптація системи під військові цілі з метою ведення обліку та ремонту пошкодженої техніки.

Рік виконання кваліфікаційного бакалаврського проекту – 2024.

Рік захисту кваліфікаційного бакалаврського проекту – 2024

ЗМІСТ

ВСТУП

Автомобілі стали невід'ємною частиною нашого життя, автомобільна промисловість стрімко розвивається, а сучасні транспортні засоби стають дедалі складнішими та технологічно досконалішими. Потреба в комплексній системі ремонту автомобілів стала більш критичною, ніж будь-коли.

Інформаційна система дозволить користувачам створювати власний кабінет, додавати свої автомобілі, та оформлювати заявки на діагностику для визначення потреби в комплектуючих, та безпосередньому ремонті авто.

Створення системи з таким функціоналом вирішує питання необхідності постійних дзвінків з метою уточнення статусу діагностики чи ремонту, це все можна перевірити за допомогою веб-ресурсу інформаційної системи.

Метою роботи бакалаврського проєкту є проєктування системи для розробки прототипу системи для визначення потреби у комплектуючих для ремонту автомобілів, що дозволить користувачам мати особистий кабінет в якому вони зможуть переглядати історію ремонтів своїх автомобілів, та саму можливість реєстрації автомобілів в системі. Також, необхідно провести міркування про можливе вдосконалення системи в майбутньому. Одержані результати можуть бути використані в комерційних цілях. В рамках вдосконалення та перспектив розвитку проєкту розглядаються рішення автоматизація, подібним дослідженням займався А.І. Сірий [1] та Д.В. Шип [2] в своїх роботах.

Об'єктом дослідження проєкту є дослідження діяльності автосервісних підприємств пов'язаних з використанням мережі інтернет для надання послуг з ремонту та обліку ремонтних робіт автомобілів для визначення ключових потреб при проєктуванні власної системи в межах проєкту, одним з використаних матеріалів для дослідження є робота І.А. Устенко [3] про організаційні процедури в автосервісі.

Предметом дослідження проєкту є теорія та методи аналізу, проєктування, прототипування та реалізації системи визначення потреб у комплектуючих для ремонту авто.

Завданням кваліфікаційного бакалаврського проекту є дослідження діяльності сервісів що надають послуги з ремонту авто та встановлення вимог, функціоналу для проектування системи для визначення потреби в комплектуючих для ремонту автомобілів, а саме наступні завдання:

Дослідження предметної області та аналіз існуючих рішень;
Встановлення всіх необхідних вимог до системи;
Проєктування супровідних діаграм для опису внутрішньої структури, поведінки та операцій в інформаційній системі;
Описати програмне, апаратне та технічне забезпечення;
Представити результати роботи.

Для проєктування та реалізації прототипу інформаційної системи буде використано програмні засоби: OPCAT, для побудови OPD діаграм в першому розділі, Enterprise Architect 13.5, в межах другого розділу проєкту для моделювання внутрішньої структури системи та її поведінки, онлайн редактори діаграм Lucidchart, drawio для побудови схем та алгоритмів забезпечення системи та задач. Для створення прототипу інформаційної системи буде використано середовище Figma. Для реалізації модулів інформаційної системи було прийнято рішення про використання комп'ютера на базі операційної системи Windows 10 22h2, в середовищі розробки Microsoft Visual Studio Code, в якості бази даних обрано реляційну СКБД MySQL та навчальним посібником щодо баз даних Н.В. Ситник [4], мова програмування системи Python з використанням фреймворку Django, з необхідними додатковими бібліотеками для роботи з базою даних.

РОЗДІЛ 1 ХАРАКТЕРИСТИКА ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Характеристика предметної галузі та об’єкта дослідження

У наші дні інтернет дарує можливість з легкістю знайти автосервіс: лише введіть запит у браузері, і перед вами відкриється безліч варіантів ремонтних закладів - від невеликих майстерень до великих спеціалізованих СТО. Однак у минулому, коли автомобілі були розкішшю та рідкістю, ніхто не передбачав потребу у їх ремонті, заміні запчастин і робочих рідинах.

Незважаючи на технічний прогрес у США, перші автосервіси з'явилися в Німеччині. Баварці вирізнялися педантичністю та дисциплінованістю, і коли в техпаспорті автомобіля було вказано, що робочі рідини слід замінювати через певний пробіг, німці точно це виконували. Проте вони ніколи не робили цього самостійно, довіряючи ремонт виключно спеціалістам, які отримали відповідну освіту. Так почалося зростання попиту на автомеханіків.

Спочатку, автомайстерні надавали обмежений перелік послуг, таких як перевірка двигуна, заміна свічок і мастила, а також заправка пального, іноді навіть надавали послуги бензозаправних станцій для зручності клієнтів. Ці процедури не вимагали високого рівня експертизи від механіків. Проте у 40-х роках виробники автомобілів відмовилися від використання дерев'яних конструкцій (так, до цього часу частини кузова виготовлялися з дерева, наприклад, двері та стеля), що призвело до необхідності в наявності кузовних майстрів. Створення окремих малярських і кузовних цехів у сервісах вимагала наявності фахівців у цих конкретних галузях. В цей період професія автомеханіка почала розділятися на вузькі спеціалізації.

Зі стрімким розвитком автомобільного виробництва, механікам теж доводилось постійно вдосконалювати свої вміння та навички і вивчати обробку нових матеріалів, використання нових запчастин та обладнання. За рахунок зміни матеріалів у виробництві, як наприклад заміщення деревини металом та зменшення кількості тканини у салоні (тканина замінялась на шкіру для покращення вигляду та строку експлуатації, деревину ж використовували для фронтальної панелі, або в інших елементах салону, де це

було доречним) диктувало нові правила для механіків. Ці новації вимагали покращення кваліфікації і навичок, в свою чергу, діагностичне обладнання кардинально відрізнялось від сучасного, легкого в використанні. Одні лише акумуляторні батареї мали вагу що перевищувало 20кг, та змушувало використовувати спеціальні вози, при цьому ця вага жодним чином не впливала на потужність обладнання.

Механіки в західних країнах мали якісне спорядження необхідне для роботи: вони мали малярну фарбу, захисні комбінезони, спеціальні рукавички, фартухи, та гумові чоботи. Для поціновувачів автомобільної культури випускалися журнали-каталоги для де окрім авто, були присутні діагностичні прилади, а для кожного найменування спорядження виділялось декілька сторінок.

До радянського союзу автосервіси дісталися набагато пізніше - автомобілів було мало, але ситуація змінилася буквально за 4 роки: в період з 1928 до 1932 рік автомобілі були абсолютно всюди, в результаті чого виник дефіцит кадрів - автослюсарів, механіків, автоінженерів, що вплинуло на активну появу училищ, які готували автоспеціалістів різної спрямованості. Автосервіси були величезними комплексами, що могли прийняти до сотні автомобілів одночасно й надавали весь спектр послуг від шиномонтажу до ремонту агрегатів та крупних вузлів. До середини 80-х таких сервісів по було 2,5 тисячі. Автомеханіки, що працювали на цих величезних СТО, були висококваліфікованими, та міняли не тільки деталі автомобіля, але й ремонтували агрегати та цілі вузли. Сьогодні водії можуть вибирати, до якого автосервісу надати свою автівку. Хтось вибирає СТО за цінами, хтось за кваліфікацією автомеханіків, хтось шукає якісний сервіс та швидкі терміни виконання робіт [5].

Інформаційна система для визначення потреби в комплектуючих для ремонту автомобілів має передбачити наступні можливості:

- Наявність особистого кабінету та можливість редагування особистих даних користувача;

- Можливість реєстрації автомобіля в системі для кожного користувача;
- Оформлення заявки на ремонт авто;
- Перевірку статусу заявки.

Для розуміння процесів з якими ми працюємо полягає потреба у виборі способу зображення процесів-дій, об’єктів, що є невід’ємними під час цих процесів, та акторів, які безпосередньо беруть участь у виконанні чи виклику тих, чи інших процесів на виконання.

З метою представлення процесів у зрозумілій формі, буде використано методологію OPM (Object Process Methodology) (ISO/PAS 19450) [6] для побудови відповідної діаграми з використанням мови OPL (Object-Process Language). За допомогою цієї методології, можливо показати взаємодію об’єктів з процесами системи, та їх послідовність виконання, зміну станів. В якості середовища для розробки діаграм обрано середовище OPCAT[7].

Системна діаграма верхнього рівня (див рисунок 1.1) ілюструє систему як «чорний ящик», об’єктами що взаємодіють з цією системою є: клієнт – особа, що отримує послугу з ремонту автомобіля, автомобіль, що належить клієнту, адміністратор – один з працівників, що оперує системою з метою надання послуг з ремонту, механік – один чи декілька працівників, що виконують сам процес ремонту автомобіля.

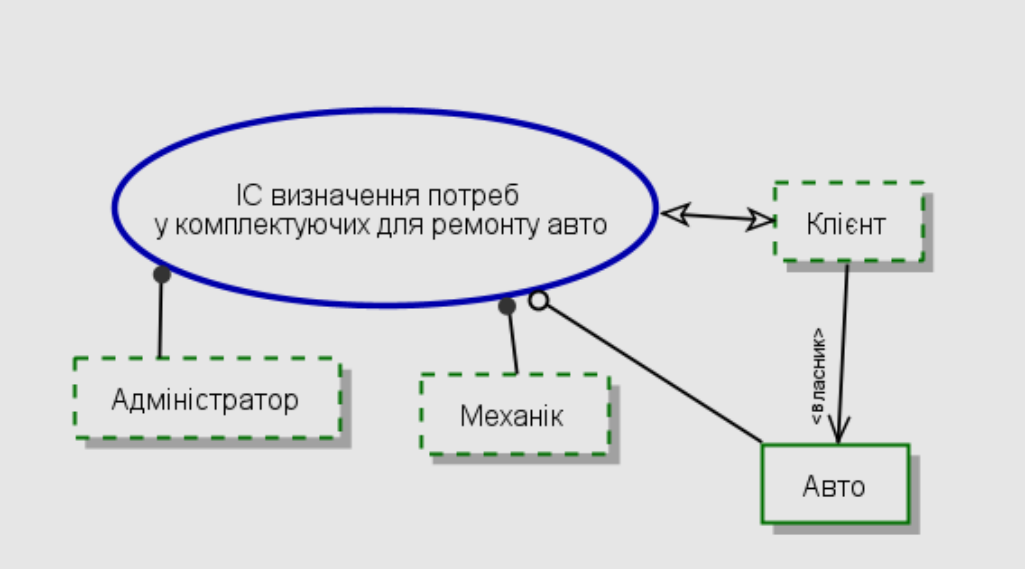


Рисунок 1.1 – Діаграма OPD верхнього рівня

Джерело: розроблено автором самостійно

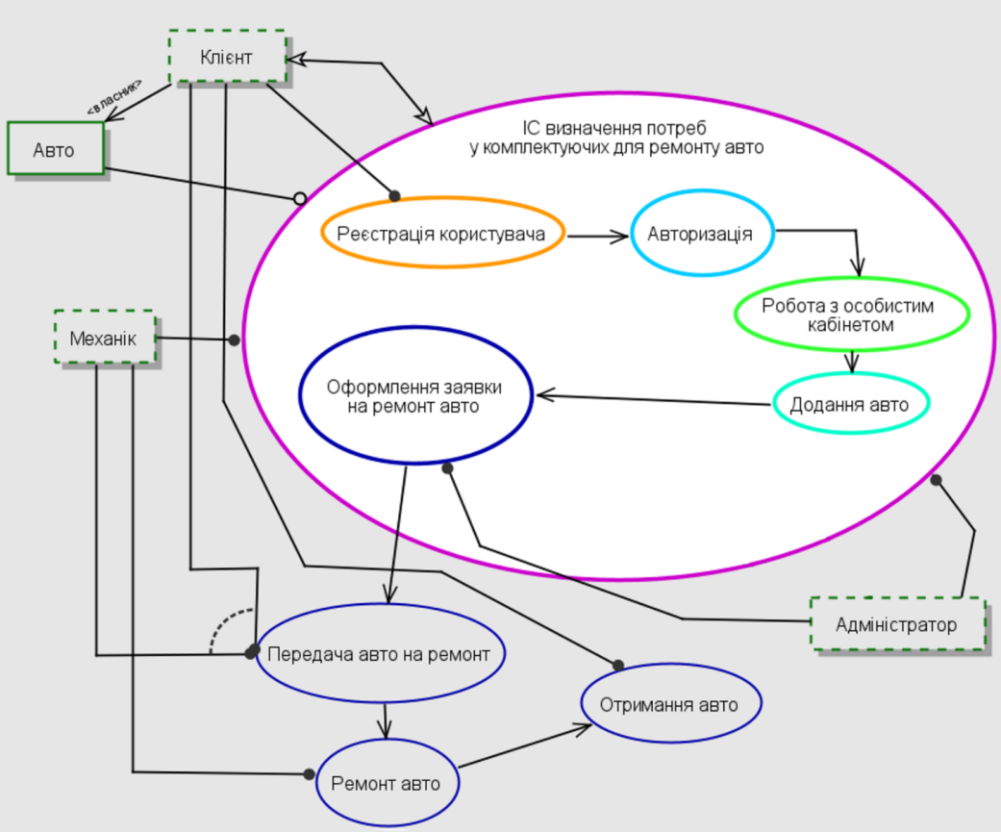


Рисунок 1.2 – Діаграма OPD нижнього рівня, детальна схема системи

Джерело: розроблено автором самостійно

На діаграмі нижнього рівня (рисунок 1.2) зображено ключові процеси, що виконуються в системі, деякі процеси, пов’язані з фізичними об’єктами виконуються за межами системи.

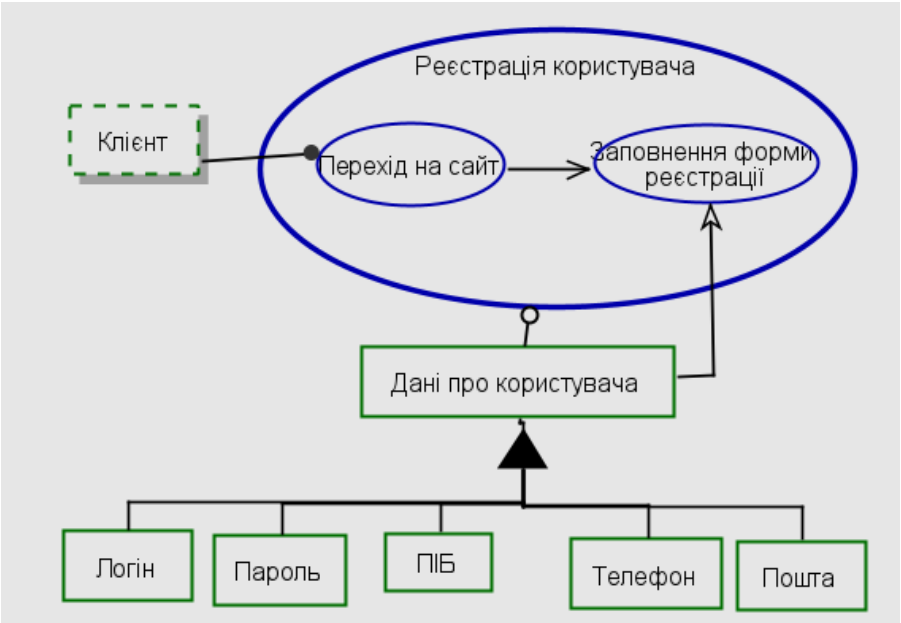


Рисунок 1.3 – Діаграма OPD нижнього рівня, процес реєстрації

Джерело: розроблено автором самостійно

На діаграмі (рисунок 1.3) зображено процес реєстрації користувача, цей процес виконує клієнт, що бажає отримати послуги з ремонту, але не має облікового запису.

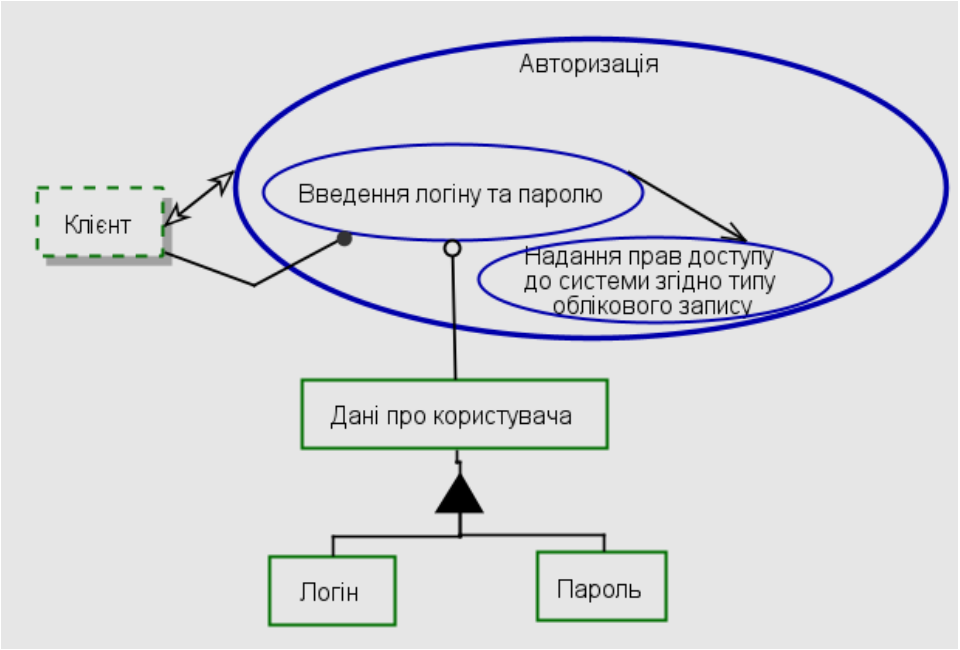


Рисунок 1.4 – Діаграма OPD нижнього рівня, процес авторизації

Джерело: розроблено автором самостійно

На діаграмі (рисунок 1.4) зображено процес авторизації користувача. Після реєстрації, створюється обліковий запис, на який можна авторизуватись маючи логін та пароль.

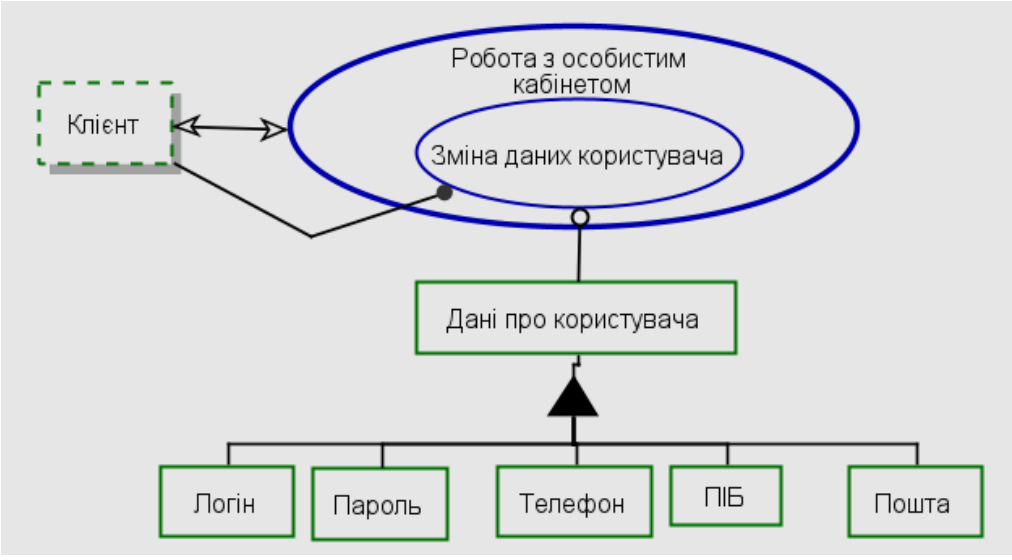


Рисунок 1.5 – Діаграма OPD нижнього рівня, процес роботи з кабінетом користувача

Джерело: розроблено автором самостійно

На діаграмі (рисунок 1.5) зображено процес зміни даних облікового запису користувача при роботі з особистим кабінетом, користувач може змінити ПІБ, логін, пароль, телефон, чи пошту.

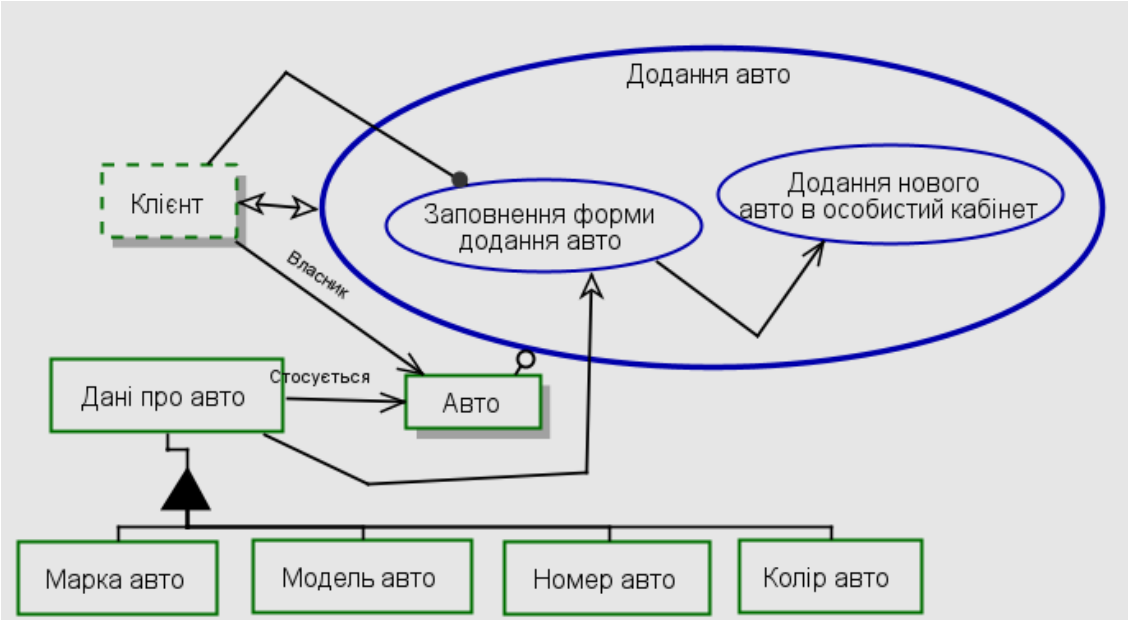


Рисунок 1.6 – Діаграма OPD нижнього рівня, процес додання автівки користувача

Джерело: розроблено автором самостійно

На діаграмі (рисунок 1.6) зображено процес додання авто. Клієнт заповнює форму на якій вказує марку, модель, номер та колір своєї автівки, після чого дана інформація зберігається в БД, і транспортний засіб асоціюється з цим користувачем.

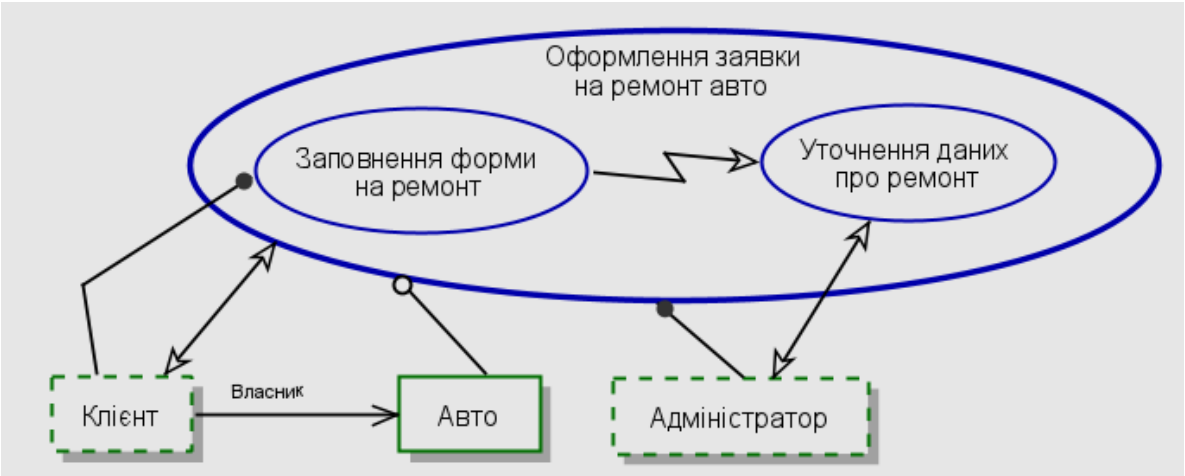


Рисунок 1.7 – Діаграма OPD нижнього рівня, процес оформлення заявки на ремонт авто

Джерело: розроблено автором самостійно

На діаграмі (рисунок 1.7) зображено процес оформлення заявки на ремонт авто. Клієнт заповнює форму на ремонт, після чого очікує на зворотній зв’язок з адміністратором, що уточнює дані про ремонт, та підтверджує, або скасовує заявку на ремонт авто.

1.2 Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

Логічним кроком при проєктуванні системи є проведення дослідження вже існуючих системних рішень, що виконують запланований функціонал, або є наближеними до цього. Цільовою системою є така ІС, що дозволяє подавати заявки на ремонт транспортних засобів зберігати дані клієнта/користувача, та його заявки, визначення потреби у комплектуючих відбувається після оформлення заявки. В Україні дуже багато сервісів що пропонують обслуговування чи ремонт транспортних засобів, для ознайомлення ми знайшли 4 Українські системи, що стосуються нашої тематики: ATL, Elcars, 1a-autoservice, Gepard. Загальні характеристики обраних систем:

Загальна характеристика системи Atl. Atl це одна з відомих мереж автомагазинів та сервісів обслуговування автомобілів в Україні якою користується велика кількість людей. Компанія працює з 2000 року та має свою систему у вигляді веб ресурсу, завдяки якому надає свої послуги [8]. На веб ресурсі компанії передбачено:

- Особистий кабінет. Завдяки наявності особистого кабінету користувача, клієнти можуть в будь-який момент подивитись історію замовлень комплектуючих, чи звернень до сервісу обслуговування автомобіля;
- Власний магазин. Компанія має власний магазин комплектуючих та запчастин для автомобілів, тож питання в постачанні комплектуючих для ремонту власним сервісом не є проблемою;
- Дисконтна система. За використання послуг що надає компанія, клієнту нараховується знижка на подальші замовлення комплектуючих чи звернень за ремонтом авто.

Загальна характеристика системи Elcars. Elcars це компанія автодилер з Харкова, що займається продажом електроавтомобілів, наданням послуг з ремонту та обслуговування як електроавтомобілів так і транспорту з двигунами внутрішнього згорання, детейлінг центр, та автомийку. Компанія відкрита до франчайзингу і надає власну систему для керування економічною складовою автосервісу при співпраці [9]. Для клієнтів автосервісу, доступний веб-ресурс з особистим кабінетом та можливістю реєстрації власного автомобіля. В цілому, система Elcars для клієнтів пропонує:

- Особистий кабінет. Подібно до АТЛ, наявний особистий кабінет користувача, клієнти можуть в будь-який момент подивитись історію замовлень комплектуючих, чи звернень до сервісу обслуговування автомобіля;
- Розподіл системи на підрозділи. Головна сторінка компанії дозволяє обрати які послуги цікавлять клієнта – покупка електроавтомобіля, ремонт чи обслуговування, детейлінг чи мийка автомобіля;
- Мобільний додаток. Підрозділ автосервісу має власний додаток на IOS та Android пристрої завдяки якому, можна отримати функціонал веб-ресурсу в форматі мобільного додатку.

Загальна характеристика системи 1a-autoservice. 1a-autoservice [10] це компанія що надає послуги автосервісу, в якості інформаційної системи доступною для клієнтів використовується веб-ресурс, на якому можна переглянути графік роботи, місцезнаходження, контакти автосервісу, та оформити заявку на отримання ремонтних послуг. На відміну від попередніх двох систем, особистий кабінет користувача відсутній, тому клієнт не зможе переглянути історію замовлень. В формі заявки вказується ПІБ клієнта, номер телефону, та вибирається одна з категорій дефектів автомобіля. Особливих функцій даного ресурсу виокремити, на жаль не вдається.

Загальна характеристика системи Gepard. Gepard [11] це компанія що надає послуги автосервісу, аналогічно до попередньої системи (1a-autoservice) в якості інформаційної системи доступною для клієнтів використовується веб-

ресурс, функціонал даного веб ресурсу має абсолютно ідентичний функціонал для клієнта, тому детальний опис на нашу думку не є потрібним.

На основі порівняльної таблиці 1.1, можемо побачити, що найбільш ближчою до цільової задумки є сервіси Atl та Elcars. Визначити яка з них краще важко, оскільки Elcars пропонує мобільний додаток, а сервіс Atl має власний магазин окрім послуг з ремонту транспортних засобів.

Перевагою цих двох ресурсів над іншими двома є наявність особистого кабінету користувача, де клієнт може переглянути історію звернень, деталі стосовно цих звернень, зареєструвати транспортний засіб. Припускається, що сервіси Gepard та 1a-autoservice теж ведуть історію звернень по клієнтам, але для власної звітності з подальшою аналітикою, без надання доступу до цієї інформації клієнтам.

Таблиця 1.1 – Таблиця порівняння існуючих інформаційних систем

№	Можливості	Atl	Elcars	1a-autoservice	Gepard
Функціонал особистого кабінету					
1	Особистий кабінет клієнта	Так	Так	Ні	Ні
2	Перегляд історії ремонтів клієнтом	Так	Так	Ні	Ні
3	Додання транспортного засобу в кабінет	Так	Так	Ні	Ні
4	Моніторинг статусу ремонту	Так	Так	Ні	Ні
Додатковий функціонал системи					
5	Запис на діагностику через сайт	Так	Так	Так	Так
6	Перегляд місцезнаходження сервісу	Так	Так	Так	Так
7	Мобільний додаток	Ні	Так	Ні	Ні

Джерело: розроблено автором самостійно

Тобто, порівнявши дані продукти ми можемо виділити певні характеристики що нас абсолютно влаштовують, та теж мають бути в нашій інформаційній системи також, а саме:

- Особистий кабінет користувача. Додання даного функціоналу до ІС спростить життя клієнтам оскільки вони будуть в базі, та зможуть вручну додавати інформацію;
- Перегляд історії заявок. Завдяки імплементації особистого кабінету, є можливість зручно показувати історію звернень клієнта за ремонтними послугами;
- Можливість додання користувачем транспортного засобу в кабінеті. Реєстрація клієнтом транспортного засобу через кабінет полегшує життя як і собі, так і сервісу що надає послуги з сервісу чи ремонту транспортного засобу.

Проаналізувавши позитивні характеристики, варто виокремити й недоліки існуючих рішень.

Gepard та 1a по суті є ідентичними по функціоналу, тому недоліки однієї системи характерні й іншій, а саме – відсутність особистого кабінету, за рахунок якого втрачається велика кількість функцій, які клієнт може виконувати самостійно. Замість цього використовується проста форма запису по телефону, коли заповнюється форма, заявка надходить до системи, та оператор зв’язується для уточнення деталей стосовно проблеми.

Основні можливості які мають бути присутніми в нашій ІС: зрозумілий та нескладний інтерфейс, розподіл кабінету, реєстрації транспорту та заявки на окремі блоки, можливість редагування контактної інформації та інформації про транспортний засіб користувачем. В перспективі розробити мобільну версію ІС, подібну до сервісу Elcars, що дублює функціонал веб-ресурсу.

В першому розділі бакалаврського проєкту було проведено огляд предметної області згідно теми проєкту. За допомогою методології ОРМ було описано процеси, що використовуються в інформаційній системі з використанням діаграм та текстовим описом. Було проведено огляд існуючих рішень та побудовано порівняльну таблицю на основі цих рішень.

РОЗДІЛ 2 РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

2.1 Аналіз і специфікація вимог до інформаційної підсистеми

2.1.1 Бізнес-вимоги до системи

Під час проєктування інформаційних систем виконується безліч різних дій, починаючи від концепції до реалізації та вдосконалення. В даному розділі, планується встановити вимоги до системи, основною задачею яких, є визначення певних рамок, функціоналу, особливостей, обмежень, тощо.

Для подальшої роботи над проєктуванням системи, необхідно визначитись з вимогами до системи, що вона має виконувати і як. З метою створення вимог, було використано мову моделювання SysML [12]. Середовище для побудови даних діаграм «Enterprise Architect 13.5» [13]. Наведені вимоги орієнтуються одночасно і на клієнта, і на адміністратора даної системи, в подальшій перспективі, з метою розширення проєкту передбачається додання нових вимог завдяки зручності роботи з даним середовищем. Вимоги поділяються на 3 основних види: бізнес вимоги, функціональні вимоги, та нефункціональні вимоги.

Бізнес-вимоги представляють собою загальні твердження, що визначають цілі, завдання та потреби організації, та являються основою проєкту. Джерелом бізнес вимог слугують зацікавлені сторони, тобто як і власники підприємства, так і користувачі.

Даний вид вимог поверхнево охоплюють цілі підприємства та те, що вимагається від продукту. Оскільки вони не вдаються в деталі функціонування, їх розглядають як опорну точку для розробки функціональних та нефункціональних вимог, які в свою чергу вже детально і чітко описують ті, чи інші задачі, які виконуватиме проєкт.

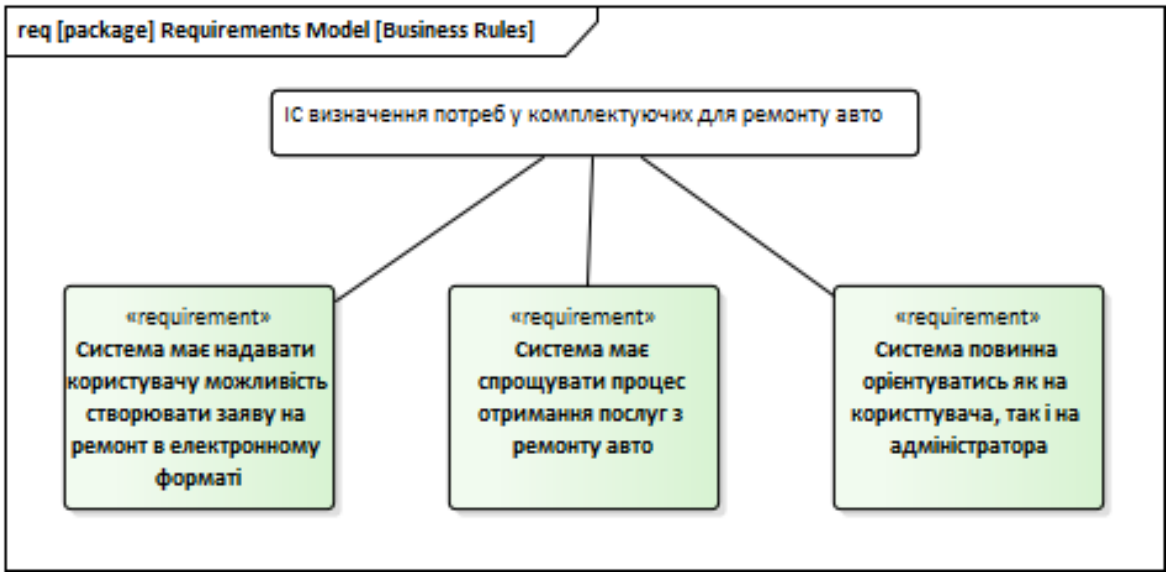


Рисунок 2.1 – Діаграма бізнес вимог до системи для визначення потреби в комплектуючих для ремонту автомобілів

Джерело: розроблено автором самостійно

На діаграмі (рисунок 2.1) наведено 3 початкові ключові бізнес вимоги, на основі яких, сформовані функціональні та нефункціональні вимоги до системи.

- Система має надавати користувачу можливість створювати заяву на ремонт в електронному форматі;
- Система має спрощувати процес отримання послуг з ремонту авто;
- Система повинна орієнтуватись як на користувача, так і на адміністратора.

2.1.2 Функціональні вимоги до системи

Функціональні вимоги показують, які завдання повинна виконувати система, зосереджуючись на конкретних функціях, поведінці та операціях системи Даний вид вимог визначають функціонал системи. Функціональні вимоги до системи для визначення потреби в комплектуючих для ремонту автомобілів наведені на діаграмі на рисунку 2.2.

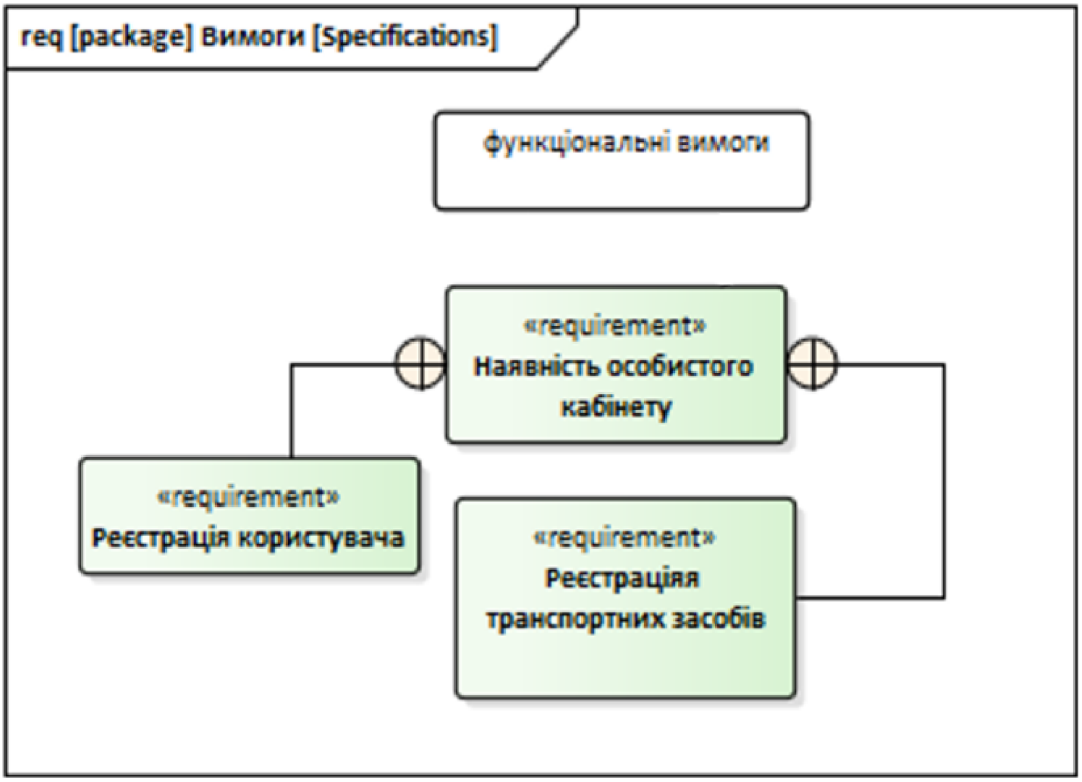


Рисунок 2.2 – Діаграма функціональних вимог до системи для визначення потреби в комплектуючих для ремонту автомобілів

Джерело: розроблено автором самостійно

На рисунку 2.2 показано які функціональні вимоги до системи були встановлені на стартовому етапі, а саме:

- Наявність особистого кабінету з можливістю його реєстрації користувачами та можливістю додання транспортних засобів. Користувач зможе зареєструвати транспортні засоби на ресурсі, та в подальшому переглядати цей транспорт в своєму кабінеті;

2.1.3 Нефункціональні вимоги до системи

Нефункціональні вимоги показують як система працює, на відміну від функціональних, які визначають можливості та функції, ці вимоги описують наскільки якісно працює система. Тобто, описувати якість функціональних вимог, наприклад система мати зручний інтерфейс, або швидко працювати. Нефункціональні вимоги до системи для визначення потреби у комплектуючих для ремонту автомобілів наведені на діаграмі на рисунку 2.3.

- Швидкодія, система має відображати сторінки з задовільною швидкістю;
- Безпека даних, користувачі можуть вводити свої контактні дані без переймань, що ці дані можуть продаватись як товар;
- Зручний інтерфейс, мінімальний інтерфейс, що дозволяє пересуватись по сайту без проблем;
- Доступність, завдяки зрозумілому інтерфейсу, так використанні піктограм, можна зменшити кількість тексту на сайті, що полегшить роботу з сервісом.

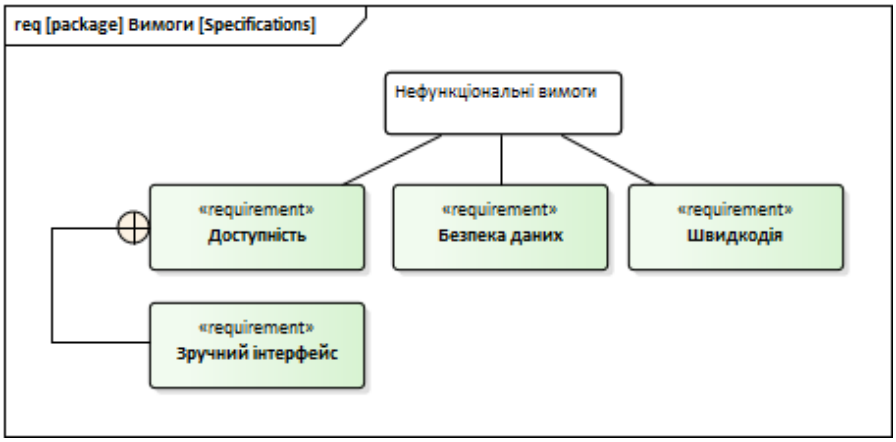


Рисунок 2.3 – Діаграма нефункціональних вимог до системи для визначення потреби в комплектуючих для ремонту автомобілів

Джерело: розроблено автором самостійно

З метою ведення обліку вимог, використовується функціонал середовища Enterprise Architect, який був використаний для створення вимоги, за своєю логікою, вимоги мають статуси виконання чи імплементації, рівень складності, та пріоритет. З використанням зазначеного середовища, з'являється можливість візуалізувати дані параметри у вигляді діаграм, та графіків. Діаграми та графіки що стосуються вимог наведені на рисунках 2.4 – 2.7.

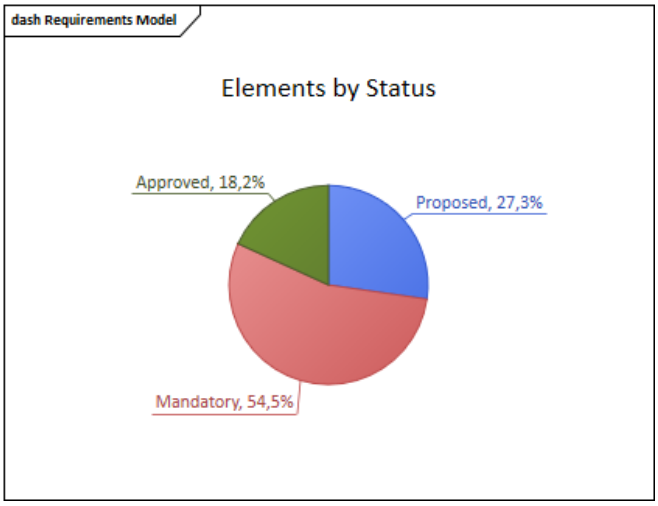


Рисунок 2.4 – Діаграма розподілу вимог до системи для визначення потреби в комплектуючих для ремонту автомобілів за статусом виконання

Джерело: розроблено автором самостійно

На графіку (рисунок 2.4) показано вимоги за їх статусом виконання, 54,5% від усіх вимог мають статус обов'язкових, 27,3% мають статус запропонованих, підтверджених 18,2% від загальної кількості.

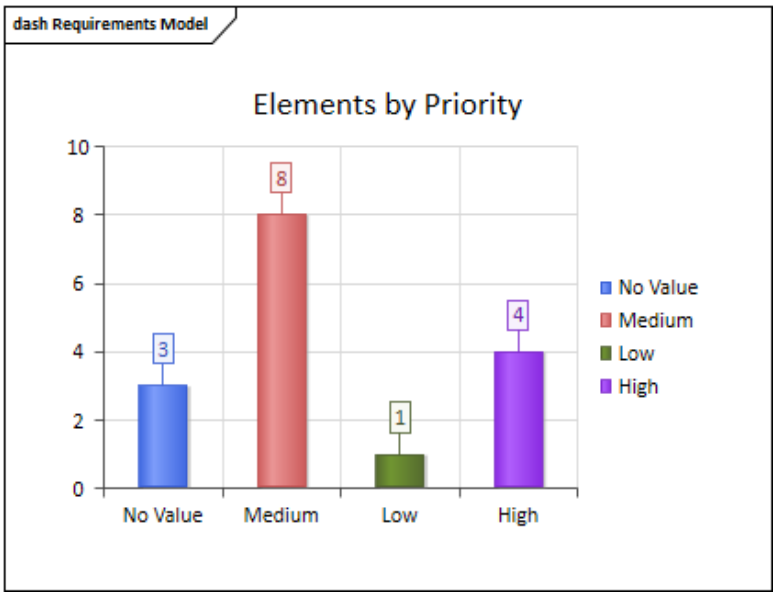


Рисунок 2.5 – Діаграма розподілу вимог до системи для визначення потреби в комплектуючих для ремонту автомобілів за пріоритетом виконання

Джерело: розроблено автором самостійно

На графіку (рисунок 2.5) показано кількість вимог за їх пріоритетом, 8 вимог 1 вимога має низький пріоритет, 8 середній, 4 високих. На графіку

фігурує елемент з назвою «No Value», це особливість роботи середовища, оскільки для нього, діаграми теж підраховуються.

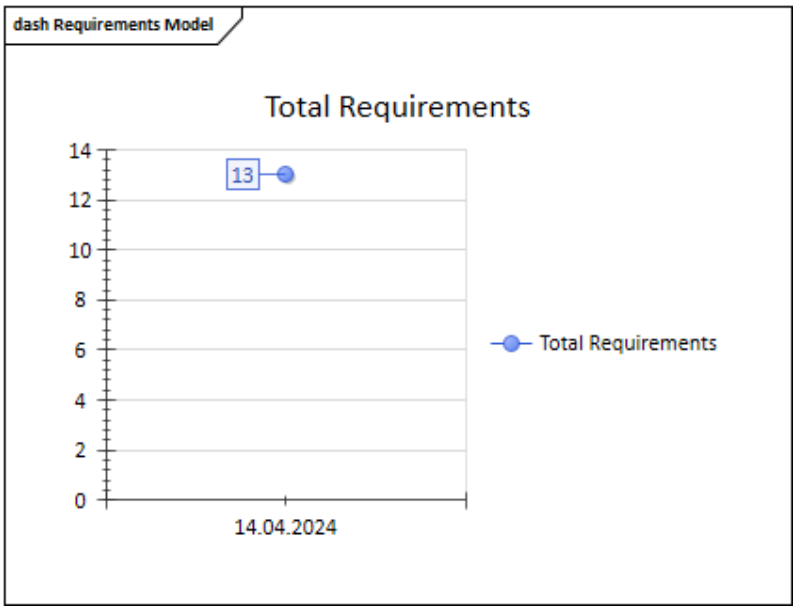


Рисунок 2.6 – Діаграма загальної кількості вимог до системи для визначення потреби в комплектуючих для ремонту автомобілів

Джерело: розроблено автором самостійно

На графіку (рисунок 2.6) показано загальну кількість вимог створених в середовищі, подібно до інших діаграм, враховуються і сторонні об’єкти, тому цифра не є точною.

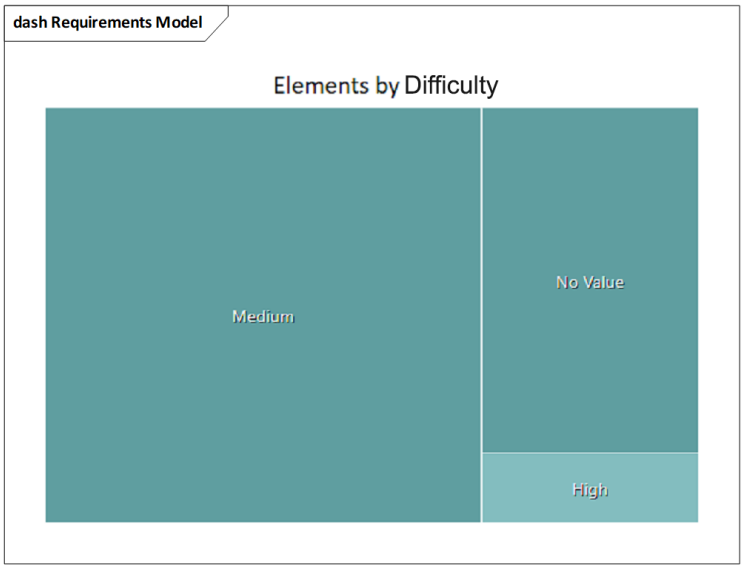


Рисунок 2.7 – Діаграма-карта вимог до системи для визначення потреби в комплектуючих для ремонту автомобілів по рівню складності

Джерело: розроблено автором самостійно

На діаграмі-карті (рисунок 2.7) складність реалізації вимог відображається у вигляді прямокутників в середині іншого прямокутника, чим більше розмір, тим більше значення цього типу.

На діаграмах іноді фігурує елемент з позначкою «No Value», це є особливістю програми, оскільки вона рахує кількість елементів у директорії, незалежно від типу файлу, в той час, діаграма на рисунку 2.6 показує правдиву сумарну кількість вимог [14].

2.2 Постановка задачі та алгоритм розв’язання задачі

Постановка задачі до системи визначення для потреби в комплектуючих для ремонту автомобілів частково посилається до функціональних вимог та можливостей системи. Інформаційна система для визначення потреб в комплектуючих для ремонту автомобілів проєктується з метою надання клієнтам певної компанії можливість використовувати дану інформаційну систему для запису на діагностику, технічне обслуговування чи ремонт автомобіля. Інформаційна система представляє собою веб-ресурс, який передбачає реєстрацію нового клієнта, автомобілів клієнта, та оформлення заявок для отримання послуг автосервісу. Задача може зупинятись в разі планового обслуговування бази даних чи всієї системи, в якості профілактичних мір, або у разі доповнення функціоналу в майбутньому, якщо зміни були погоджені.

Вхідна інформація включає в себе вхідні дані користувача для аутентифікації, інформацію про клієнта та інформація про авто клієнта для особистого кабінету користувача з метою оформлення заявок на ремонт, зображення авто для клієнта та заявки на ремонт, інформацію про автомобіль клієнта, інформація про діагностику авто, список деталей після діагностики.

Вихідна інформація включає в себе загальну інформацію про замовлення, яку може використовувати користувач, поширену інформацію може побачити адміністратор, інформація про клієнта та автомобіль.

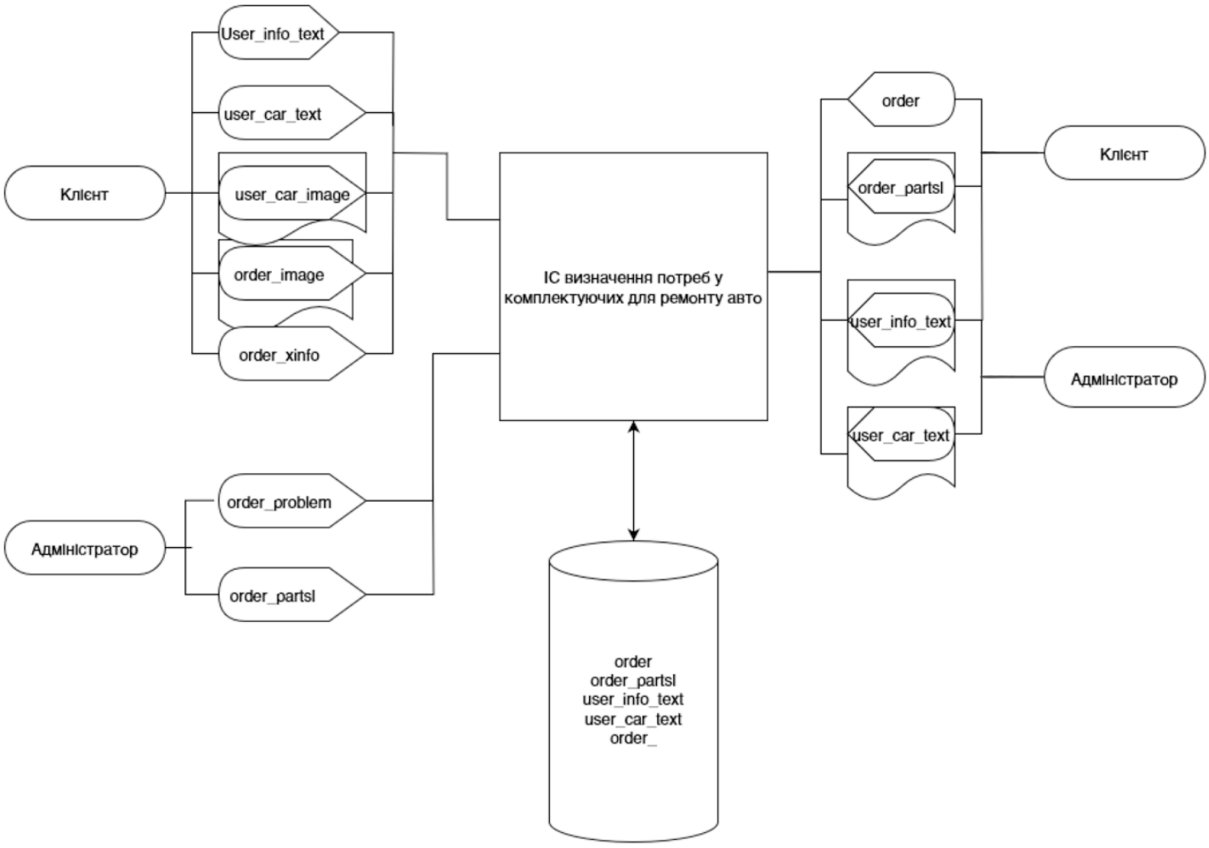


Рисунок 2.8 – Інформаційна модель задачі

Джерело: розроблено автором самостійно

З метою детального огляду інформації що буде використовуватись в системі необхідно провести розподіл даних на вхідні та вихідні. База даних буде містити таблиці в яких буде зберігатись інформація про користувача для аутентифікації, тобто логін та пароль, інформацію про клієнта, інформація про авто, зображення авто клієнта, дана інформація буде використана для особистого кабінету користувача, для оформлення заявки буде використовуватись зображення дефекту авто, та текст-опис від клієнта, інформація про діагностику автомобіля та список необхідних комплектуючих буде надавати адміністратор після вердикту автомеханіка, що виконує сам процес діагностики авто.

2.2.1 Вхідна інформація

В основному, вхідна інформація включає в себе ключові дані про користувача. «user_credentials» містять вхідні дані користувача, для авторизації на веб-ресурсі, «user_info_text» містить контактну інформацію про

користувача, «user_car_image» для зберігання зображення авто користувача. «order_problem» стосується дефектів авто та заповнюється після діагностики, «order_partsl» передбачається для зберігання переліку деталей у вигляді списку, «order_image» з метою зберігання фотографії авто з дефектом і є не обов'язковим для заповнення.

Таблиця 2.1 – Перелік вхідних повідомлень

№	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
1	Інформація про клієнта	user_info_text	Екранна форма	При реєстрації клієнта, або зміні існуючої інформації	Користувач
2	Інформація про авто	order_car_text	Екранна форма	При новій заявці на ремонт	Користувач
3	Діагноз автомобіля	order_problem	Екранна форма	При завершенні діагностики	Адміністратор
4	Зображення авто клієнта	user_car_image	Файл	При доданні нового авто	Користувач
5	Зображення авто для діагностики	order_image	Файл	Коли клієнт додає фотографію авто на діагностику	Користувач
6	Текст додаток до заявки на ремонт	order_xinfo	Екранна форма	При новій заявці на ремонт	Користувач
7	Список деталей для ремонту	order_partsl	Екранна форма	При завершенні діагностики	Адміністратор
8	Дані для входу користувача	user_credentials	Екранна форма	При авторизації клієнта	Користувач

Джерело: розроблено автором самостійно

2.2.2 Вихідна інформація

Вихідна інформація містить частково подібні дані до вхідних, але доповнених чи змінених. Так наприклад дані про ремонт «order_problem», користувач зможе дізнатись що за поломка в авто, а з «order_partsl» дізнатись

про деталі які необхідні для ремонту. Адміністратором використовується «order_car_text» щоб отримати попередню інформацію про дефект від користувача, «order» містить повну інформацію про замовлення, авто, клієнта/користувача.

Таблиця 2.2 – Перелік вихідних повідомлень

№	Назва вихідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Споживач
1	Інформація про звернення за ремонтом тз	order	Екранна форма	Миттєво	Адміністратор, Клієнт
2	Перелік деталей необхідних для ремонту тз	order_partsl	Екранна форма, файл	Миттєво	Клієнт, Адміністратор
3	Інформація про клієнта	user_info_text	Екранна форма, файл	Миттєво	Адміністратор, Клієнт
4	Інформація про авто	order_car_text	Екранна форма, файл	Миттєво	Адміністратор, Клієнт

Джерело: розроблено автором самостійно

2.2.3 Використовувана та результатна інформація

У таблиці 2.3 позначено масиви використовуваної інформації.

Таблиця 2.3 – Перелік використовуваної інформації

№	Назва масиву	Ідентифікатор масиву	Максимальна кількість записів
1	Загальна кількість заявок	orders_total	500000
2	Загальна кількість користувачів	users_total	100000
3	Загальна кількість авто	cars_total	300000
4	Кількість виконаних ремонтів	orders_c_total	500000

Джерело: розроблено автором самостійно

У таблиці 2.4 позначено масиви результативної інформації.

Таблиця 2.4 – Перелік масивів результативної повідомлень

№	Назва масиву	Ідентифікатор масиву	Максимальна кількість записів
1	Автівки	CAR	300000
2	Заявки	ORN	500000
3	Користувачі	CLN	100000
4	Успішні ремонти	ORR	500000
5	Повторні клієнти	CLR	100000

Джерело: розроблено автором самостійно

2.2.4 Математичне забезпечення та алгоритм розв'язання задачі

1. Визначення відсотків успішних заяв

$$O_{srt} = \frac{N_{\text{виконаних}}}{N_{\text{заяв}}} * 100\%, \quad (2.1)$$

де, O_{srt} – відсоток успішних заяв, співвідношення виконаних замовлень до сумарної кількості замовлень,

$N_{\text{виконаних}}$ – кількість успішних та виконаних заявок на ремонт/діагностику,

$N_{\text{заяв}}$ – кількість заяв(замовлень) в системі.

2. Відношення наявних запчастин до потрібних

$$P_{ratio} = \frac{P_{\text{склад}}}{P_{\text{потрібно}}}, \quad (2.2)$$

де, P_{ratio} – співвідношення деталей що є на складі, до потрібних деталей для ремонту авто,

$P_{\text{склад}}$ – деталі для ремонту авто що є на складі,

$P_{\text{потрібно}}$ – потрібні деталі для ремонту авто.

3. Середня кількість повторних звернень

$$O_{repeating\_avg} = \frac{N_{\text{повт\_замовлень}}}{N_{\text{повт\_клієнт}}}, \quad (2.3)$$

де, $O_{repeating\_avg}$ – середня кількість повторних заявок в сервіс,

$N_{\text{повт\_замовлень}}$ – кількість успішних заяв у клієнта у якого кількість замовлень >1,

$N_{\text{повт\_клієнт}}$ – кількість клієнтів за ідентифікатором якого наявне >1 успішної заявки.

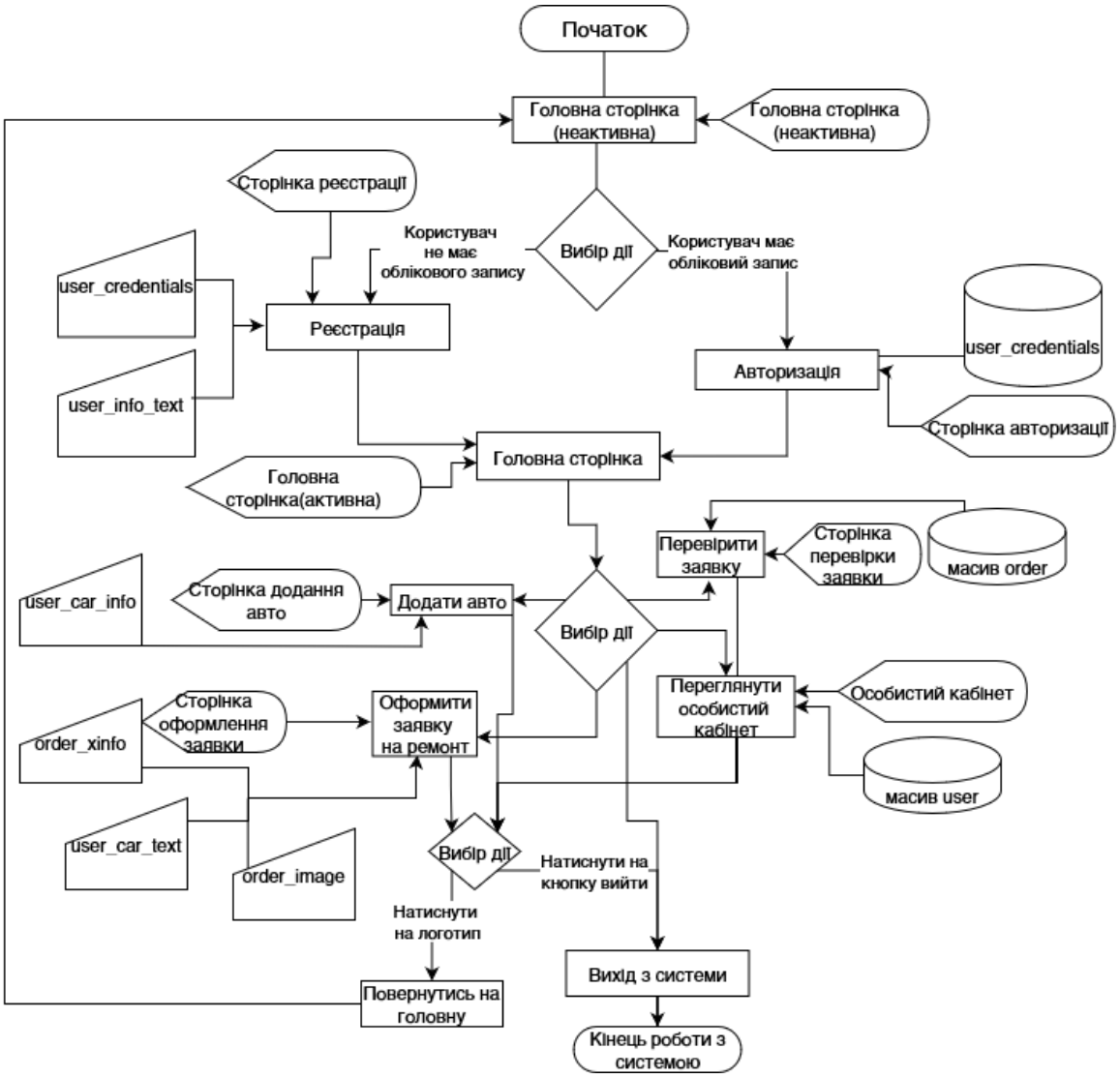


Рисунок 2.9 – Алгоритм функціонування системи з даними

Джерело: розроблено автором самостійно

2.3 Моделювання інформаційної підсистеми

2.3.1 Моделювання поведінки системи

З метою моделювання поведінки системи, було застосовано середовище «Enterprise Architect». Поведінка системи та її учасників, була змодельована у вигляді Use-Case діаграми з використанням мови моделювання. На даній діаграмі (рисунок 2.1) зображено акторів (учасників), та функції (прецеденти). Діаграма подібна до діаграми на рисунку 1.2, але більш детально показує взаємодію акторів з системою, та між собою.

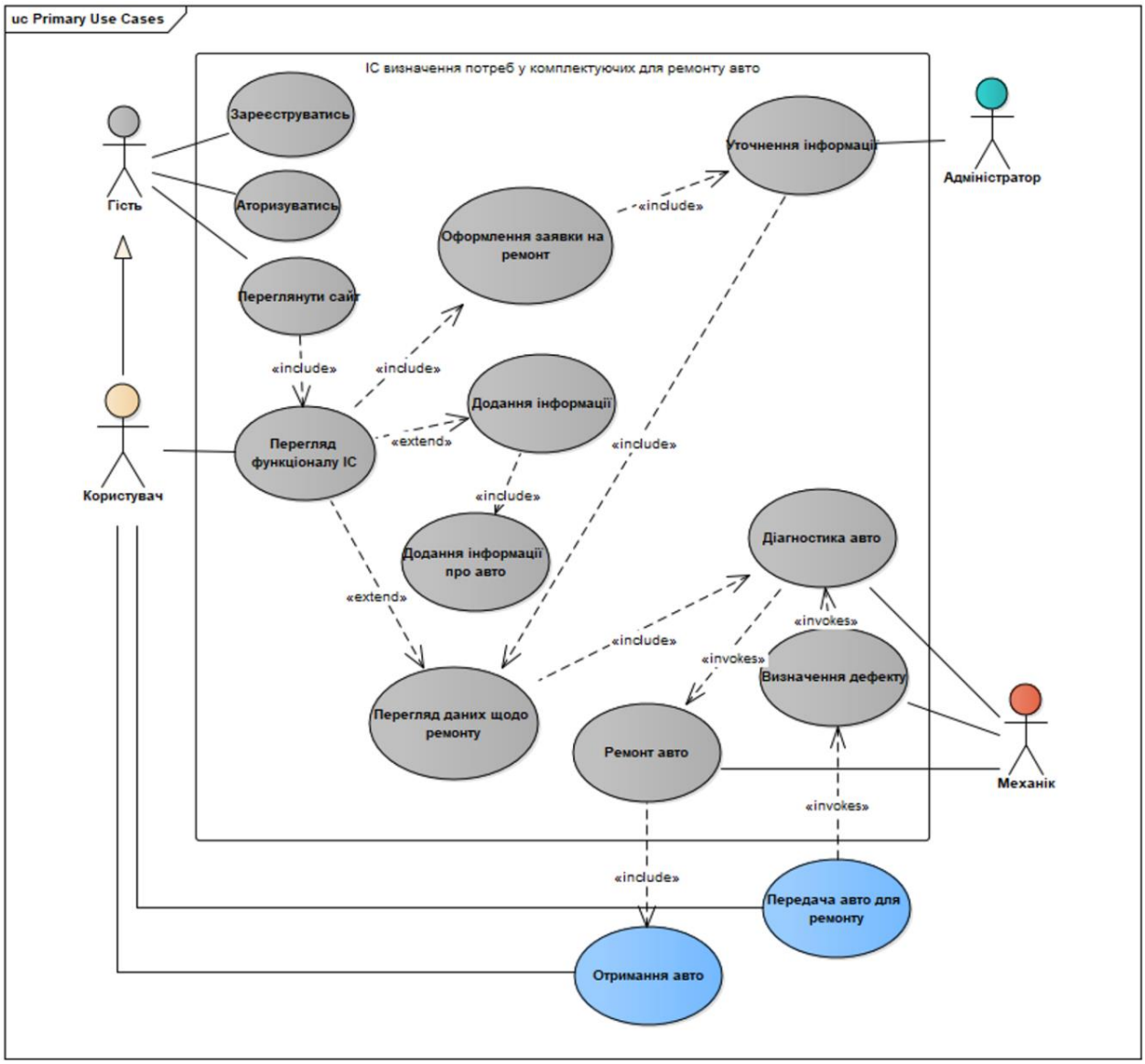


Рисунок 2.10 – Use Case діаграма

Джерело: розроблено автором самостійно

Актор Гість може стати актором Користувач після проходження реєстрації та авторизації на веб ресурсі. Після чого, отримує доступ до функціоналу веб ресурсу та може оформлювати заявку на ремонт. Актори Адміністратор та Механік взаємодіють з актором Користувач за допомогою процесів, результатом виконання яких є отримання користувачем відремонтованого авто.

Для певних прецедентів було побудовано сценарії, на рисунках 2.11-2.16, що передбачають основний хід подій по даним прецедентам, альтернативний хід подій, та виключення.

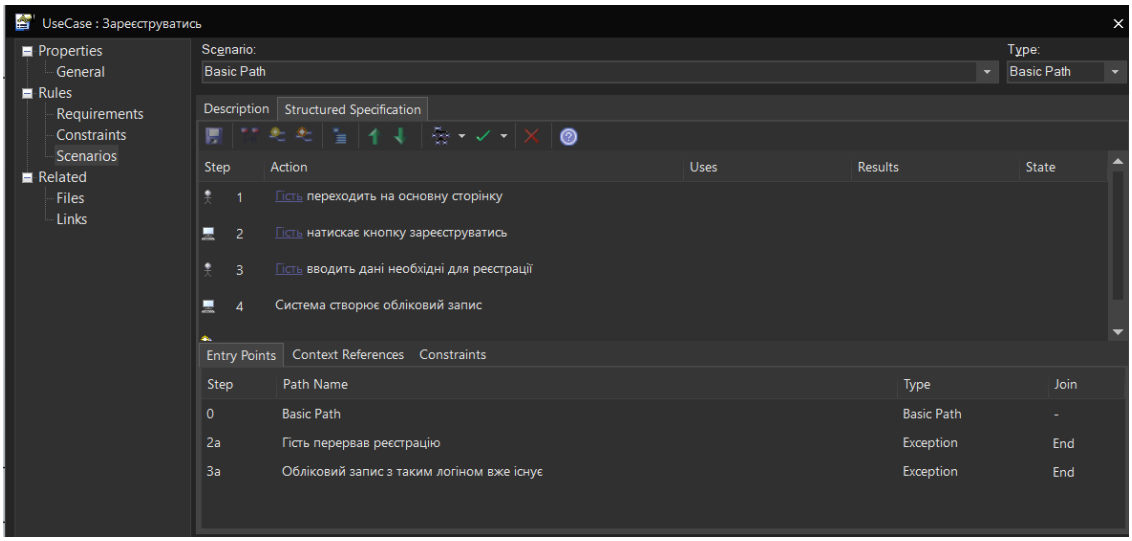


Рисунок 2.11 – Сценарії прецеденту «Зареєструватись»

Джерело: розроблено автором самостійно

На рисунку 2.11 показано сценарій прецеденту «Зареєструватись». Основний сценарій - гість проходить реєстрацію на веб-ресурсі та система створює обліковий запис. Виключеннями є зупинка реєстрації користувачем, або якщо пошта користувача вже закріплена за іншим обліковим записом.

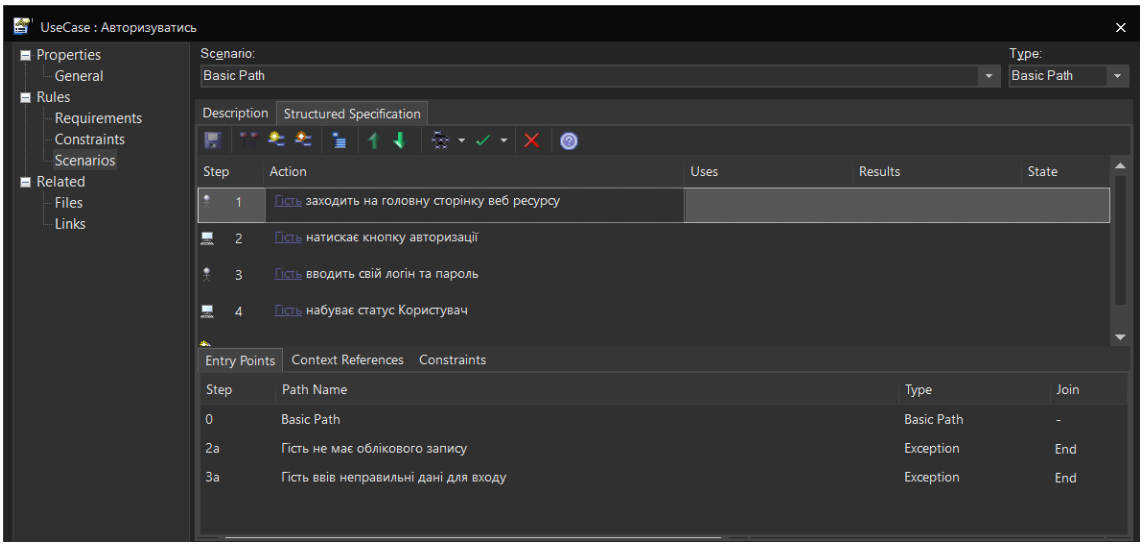


Рисунок 2.12 – Сценарії прецеденту «Авторизуватись»

Джерело: розроблено автором самостійно

На рисунку 2.12 показано сценарій прецеденту «Авторизуватись». Основний сценарій - гість проходить авторизацію на веб-ресурсі, вводить дані для входу та набуває статусу користувача, що дає доступ до функціоналу веб-

ресурсу. Виключеннями є відсутність облікового запису з такими даними, або введені некоректні пошта чи пароль.

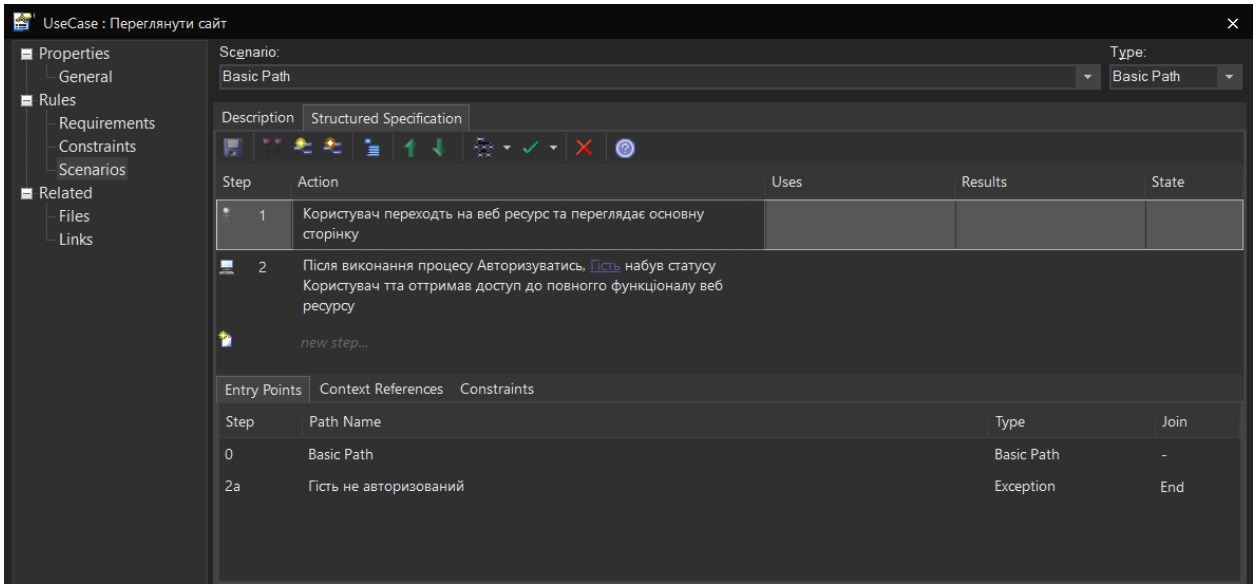


Рисунок 2.13 – Сценарії прецеденту «Переглянути сайт»

Джерело: розроблено автором самостійно

На рисунку 2.13 показано сценарій прецеденту «Переглянути сайт». Основний сценарій – неавторизований гість переходить на головну сторінку та переглядає веб-ресурс Якщо користувач не проходив авторизацію, в такому разі, він не має доступу до функціоналу веб-ресурсу.

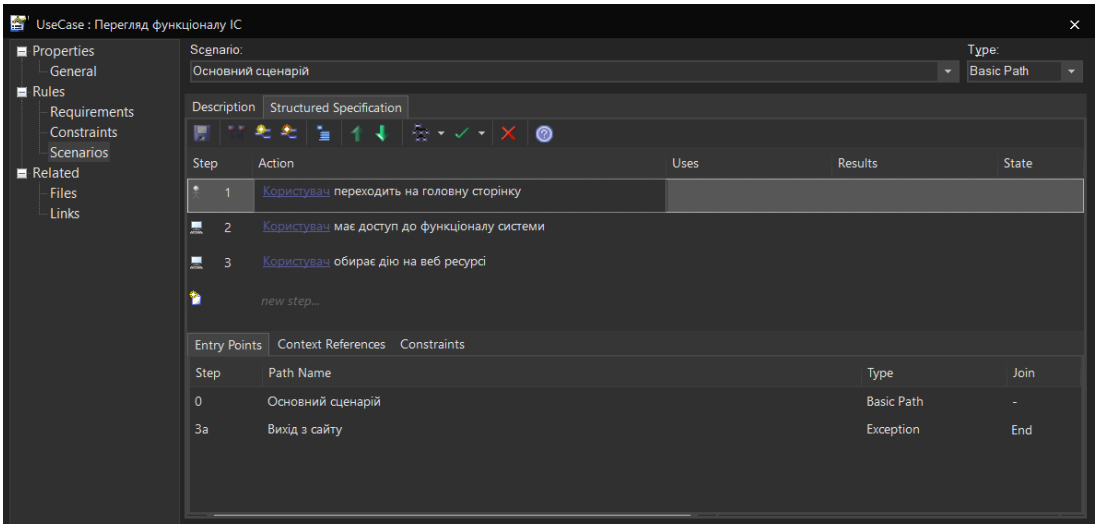


Рисунок 2.14 – Сценарії прецеденту «Перегляд функціоналу ІС»

Джерело: розроблено автором самостійно

На рисунку 2.14 показано сценарій прецеденту «Перегляд функціоналу ІС». Основний сценарій – користувач переходить на головну сторінку та

переглядає доступні йому кнопки. Виключенням є якщо користувач вийшов з системи, або закрив сторінку веб-ресурсу.

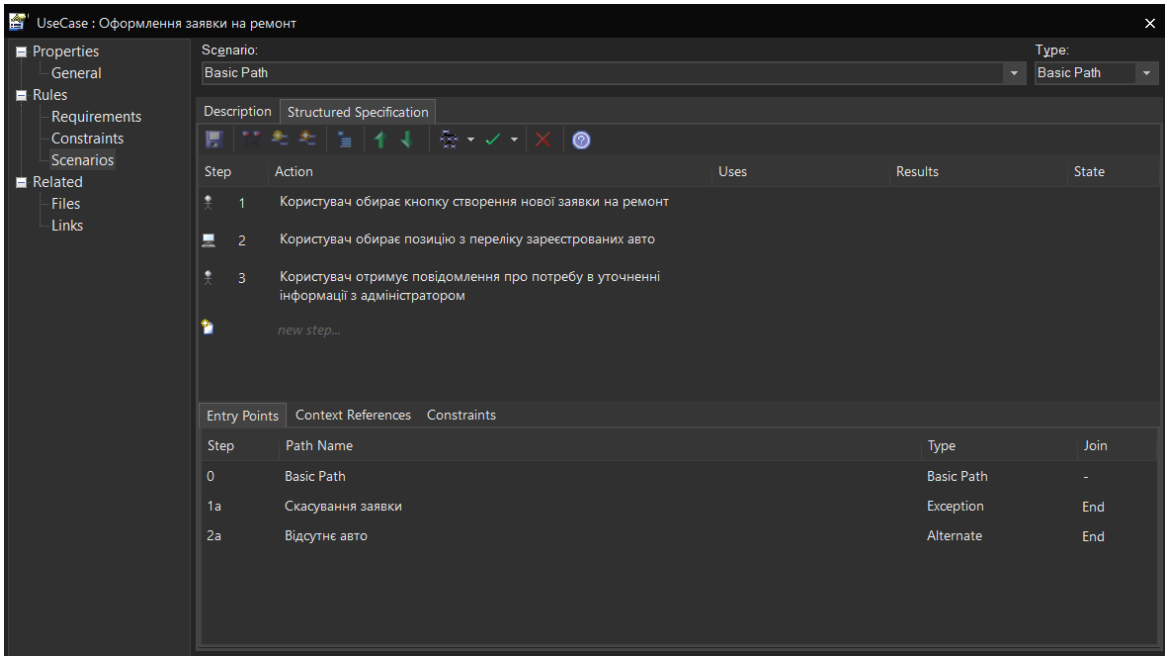


Рисунок 2.15 – Сценарії прецеденту «Оформлення заявки на ремонт»

Джерело: розроблено автором самостійно

На рисунку 2.15 показано сценарій прецеденту «Оформлення заявки на ремонт». Основний сценарій – користувач обирає кнопку створення заявки на ремонт, обирає зі списку одну з зафіксованих за ним авто та заповнює форму. Виключеннями є скасування користувачем заявки під час заповнення, або відсутність зареєстрованих авто за користувачем.

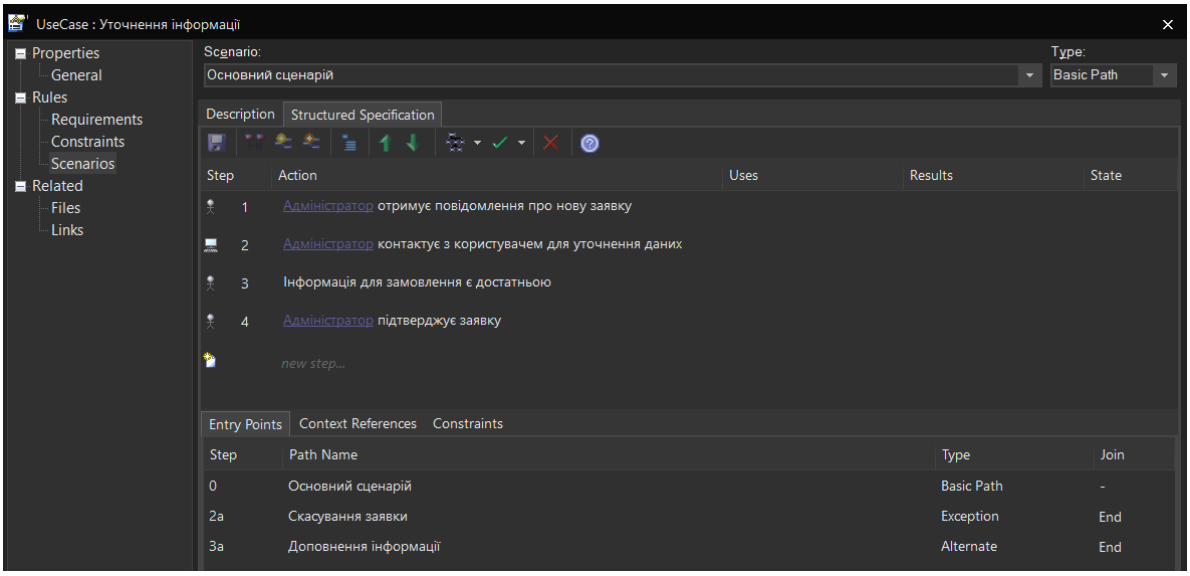


Рисунок 2.16 – Сценарії прецеденту «Уточнення інформації»

Джерело: розроблено автором самостійно

Для певних прецедентів також було побудовано діаграми послідовності [15], щоб детально показати як актори взаємодіють з системою чи між собою, які дії виконуються, та що вони отримують після певних дій. На рисунках 3.8-3.12 зображено прецеденти: «Реєстрація», «Додання інформації в кабінет», «Оформлення заявки на ремонт», «Діагностування авто», «Ремонт авто».

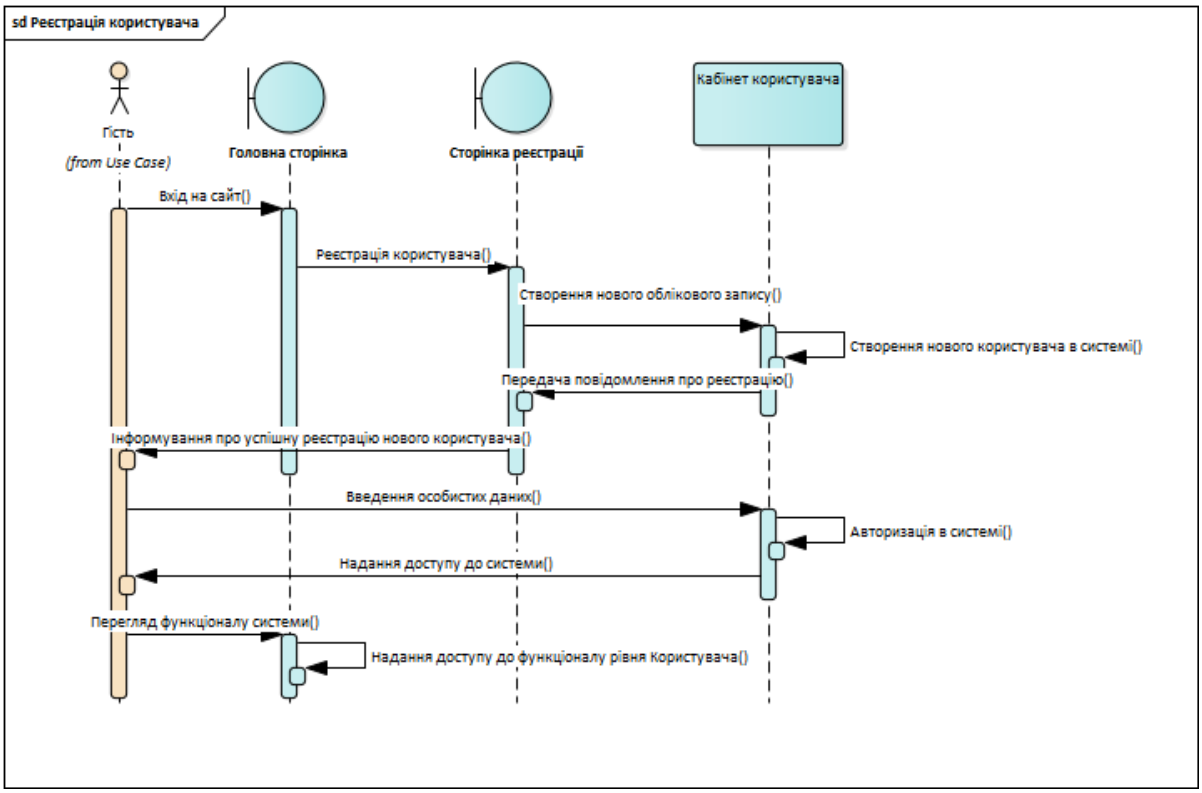


Рисунок 2.17 – Діаграма послідовності процесу «Реєстрація користувача»

Джерело: розроблено автором самостійно

На рисунку 2.17 зображена діаграма послідовності процесу «Реєстрація користувача». Гість потрапляє на веб-ресурс та не має облікового запису. Він переходить на сторінку реєстрації та реєструє свій обліковий запис. Система створює нового користувача в базі та інформує про це особу. Користувач вводить свої дані для входу, система перевіряє ці дані та при введенні коректних даних надає доступ до системи згідно рівня прав користувача.

На рисунку 2.18 зображена діаграма послідовності процесу «Додавання інформації». Користувач переходить на головну сторінку веб-ресурсу і звідти переходить в свій кабінет, де може редагувати свої дані. В разі вибору редагування та оновлення даних, система оновлює ці дані. Якщо користувач

хоче додати авто, то операція подібна, обирається пункт додання нового авто, заповнюється інформація, зберігається системою.

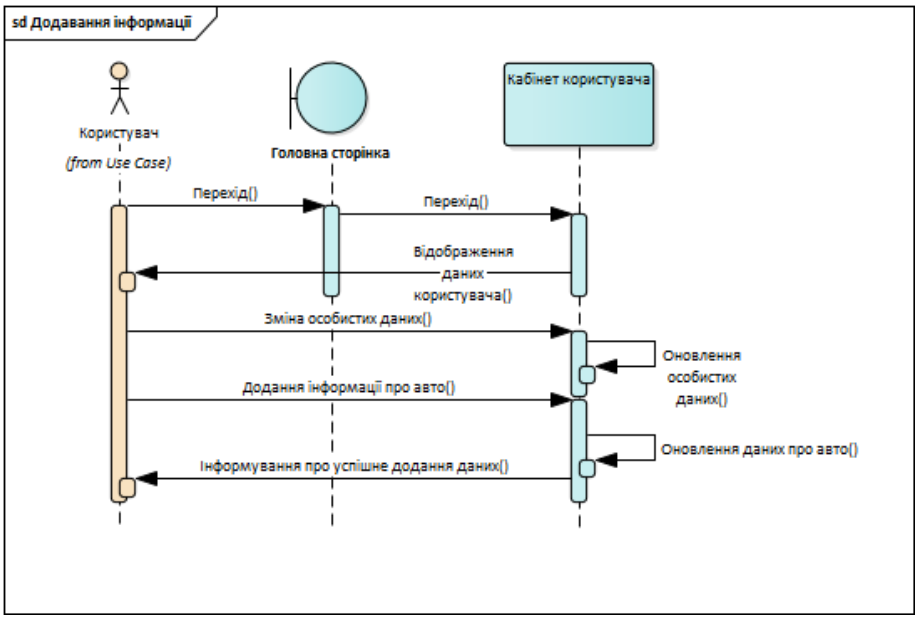


Рисунок 2.18 – Діаграма послідовності процесу «Додавання інформації»

Джерело: розроблено автором самостійно

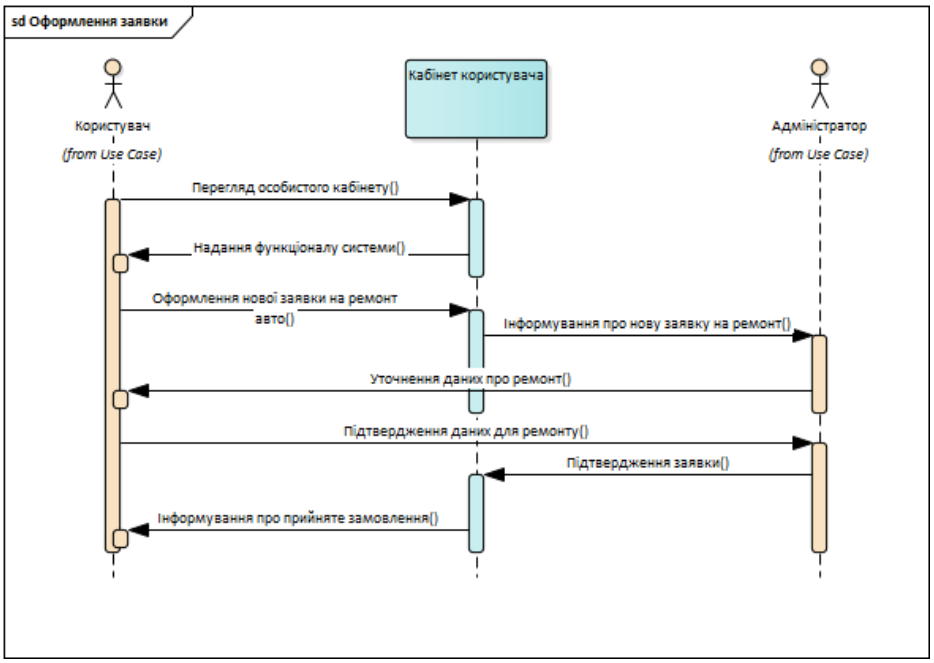


Рисунок 2.19 – Діаграма послідовності процесу «Оформлення заявки»

Джерело: розроблено автором самостійно

На рисунку 2.19 зображена діаграма послідовності процесу «Оформлення заявки». Користувач оформлює заявку на ремонт авто і це фіксується системою. Система інформує адміністратора про наявність нової

заявки на ремонт, після чого відбувається контакт адміністратора чи менеджера з клієнтом/користувачем, якщо всі умови влаштовують – заявка отримує статус прийнятої.

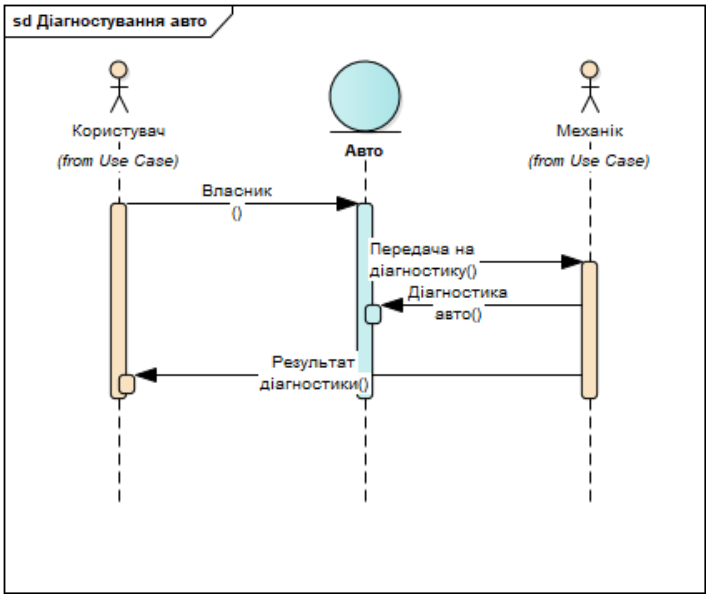


Рисунок 2.20 – Діаграма послідовності процесу «Діагностування авто»

Джерело: розроблено автором самостійно

На рисунку 2.20 зображена діаграма послідовності процесу «Діагностування авто». Дана діаграма поверхнево описує процес як авто потрапляє на діагностику, та користувач отримує інформацію про дефект.

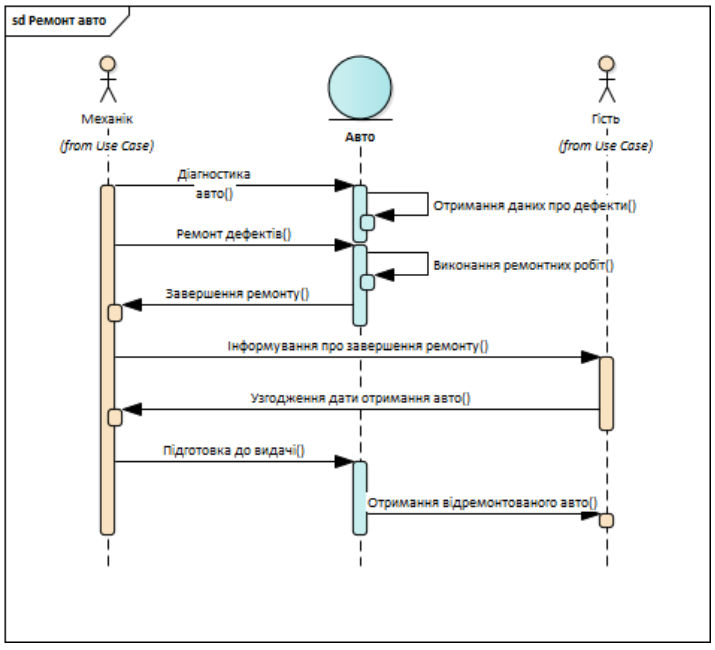


Рисунок 2.21 – Діаграма послідовності процесу «Ремонт авто»

Джерело: розроблено автором самостійно

На рисунку 2.21 зображена діаграма послідовності процесу «Ремонт авто». Дана діаграма поєднує частину діаграми на рисунку 2.20, але показує, що після діагностики користувачу дають можливість погодитись чи відхилити ремонт, якщо користувач погоджується, виконуються ремонтні роботи, по завершенню яких, вуідбуваєтсья інформування користувача, назначається час отримання авто та безпосереднє отримання користувачем/клієнтом відремонтованого авто

2.3.2 Моделювання структури системи

Використовуючи методологію SysML, були створені діаграми: визначення блоків (рисунок 2.22), внутрішніх блоків (рисунок 2.23), класів (рисунок 2.24), граничних класів (рисунок 2.25), класів керувань (рисунок 2.26). Діаграма визначення блоків (рисунок 2.22) показує, як система умовно поділена на декілька модулів, що виконують свої функції, та є складовою однієї системи.

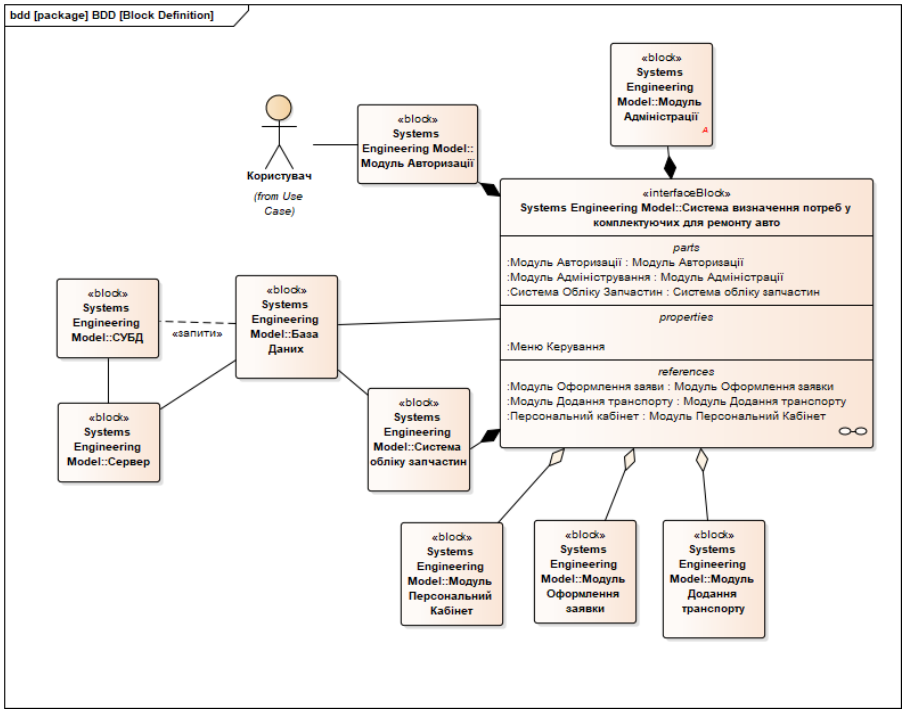


Рисунок 2.22 – Діаграма визначення блоків системи для визначення потреби в комплектуючих для ремонту автомобілів

Джерело: розроблено автором самостійно

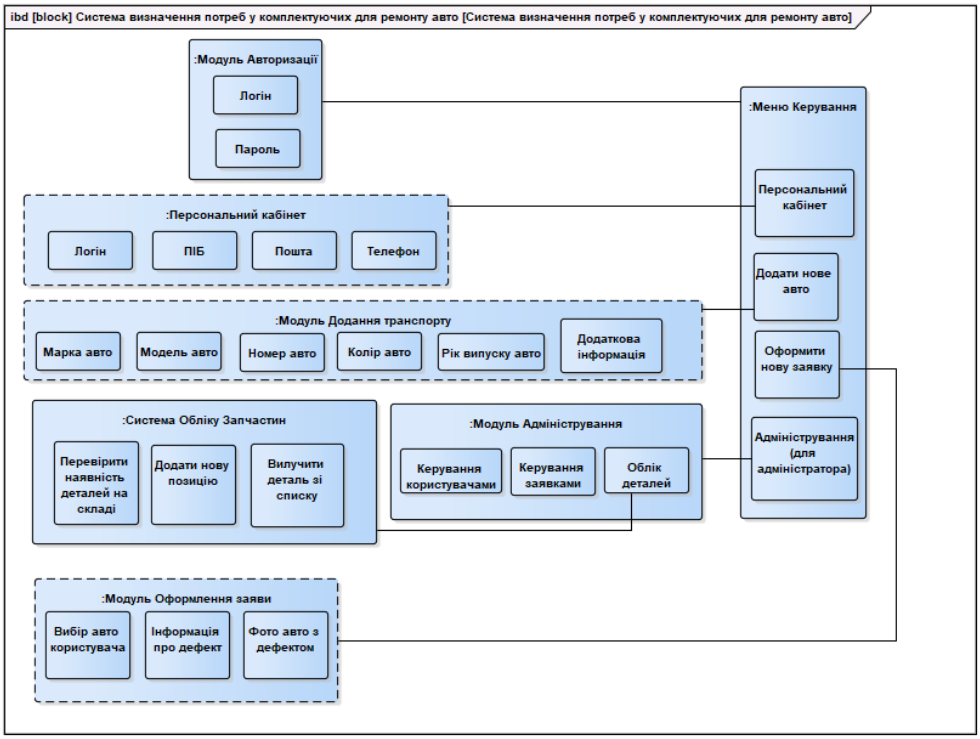


Рисунок 2.23 – Діаграма внутрішніх блоків системи для визначення потреби в комплектуючих для ремонту автомобілів

Джерело: розроблено автором самостійно

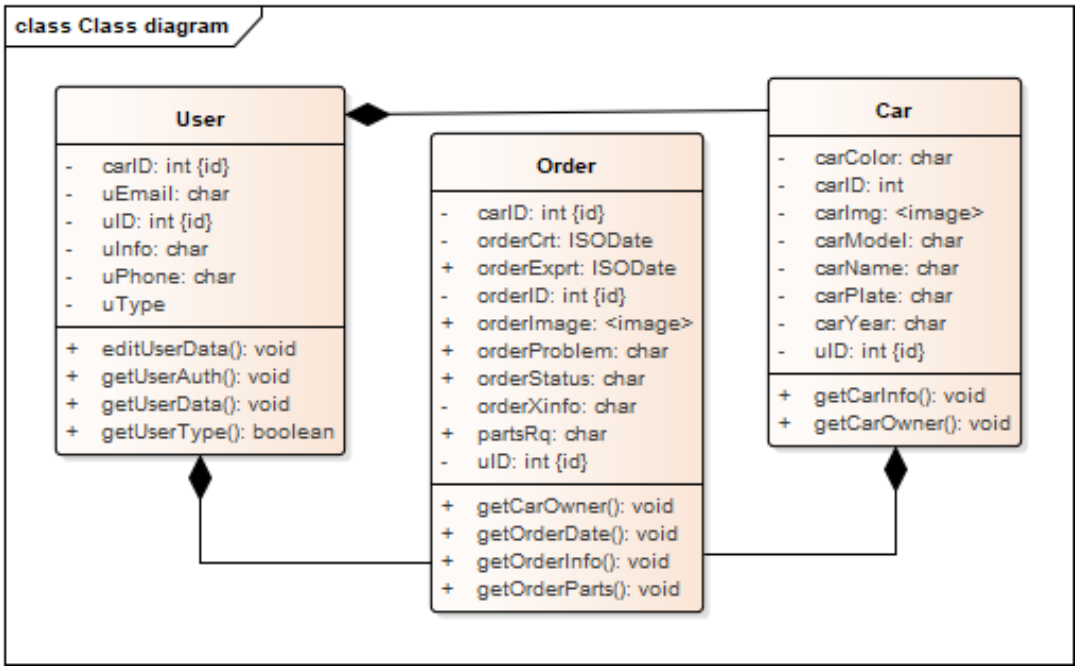


Рисунок 2.24 – Діаграма класів системи для визначення потреби в комплектуючих для ремонту автомобілів

Джерело: розроблено автором самостійно

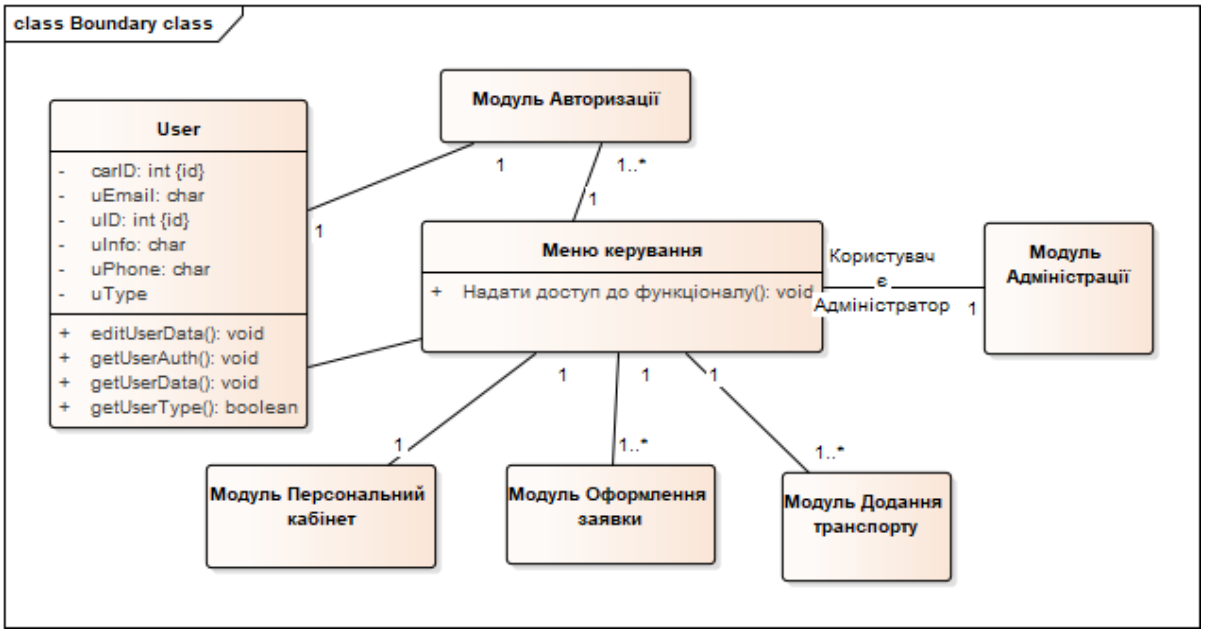


Рисунок 2.24 – Діаграма граничних класів системи для визначення потреби в комплектуючих для ремонту автомобілів

Джерело: розроблено автором самостійно

Діаграма граничних класів (рисунок 2.24) показує зв’язки умовно поділених модулів та тип зв’язків між ними.

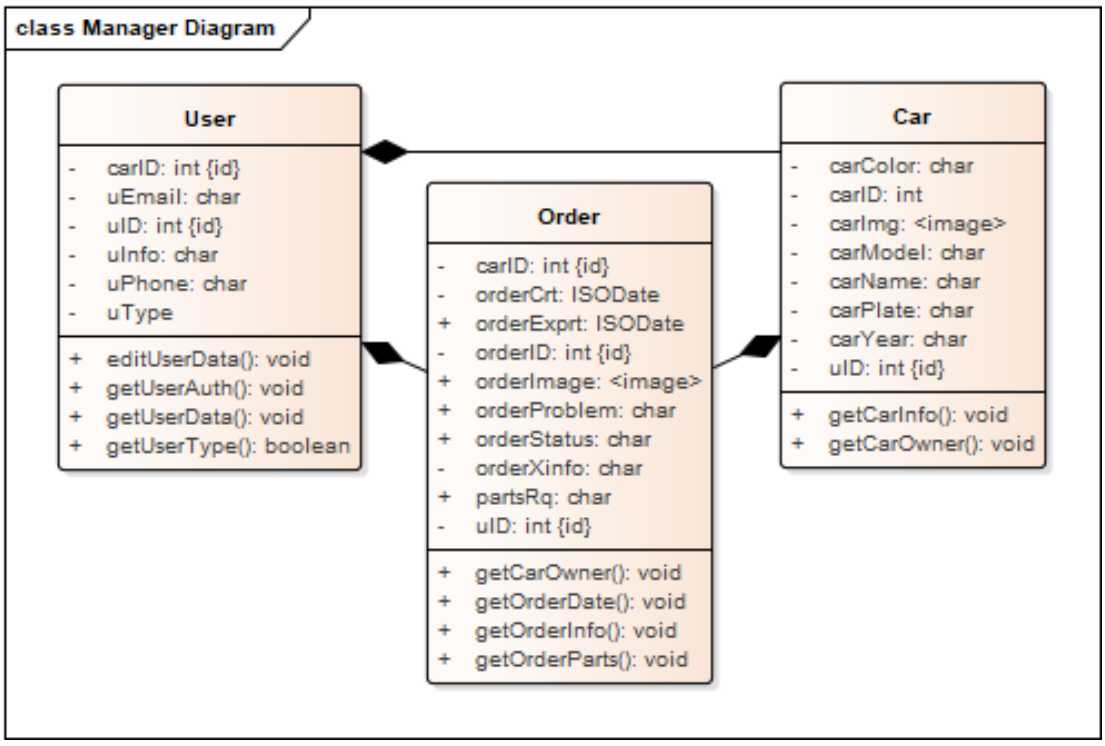


Рисунок 2.25 – Діаграма класів керування системи для визначення потреби в комплектуючих для ремонту автомобілів

Джерело: розроблено автором самостійно

Діаграма класів (рисунок 2.24) ілюструє класи та певні залежності. Атрибути в даних класах можуть бути 3-х видів: public, що є доступним в будь-якій точці системи та має префікс +, private, доступний лише в межах класу, префікс -, restricted, доступний в межах класу, та підкласів, префікс # [16]. Більшість атрибутів є закритими (private), з метою інкапсуляції та забезпечення безпеки. Методи в свою чергу є публічними (public), та надають достатню інформацію в потрібний клас. Клас Order існує на основі даних класів User та Car, оскільки заявки на ремонт не можуть існувати, якщо не існує особи, що створює цю заявку, та транспортного засобу, що підлягає ремонтним процедурам, який в свою чергу, не може існувати в системі без власника (користувача).

2.3.3 Розподіл вимог за компонентами

Діаграма трасування на рисунку 2.26 демонструє співвідношення зазначених вимог до прецедентів присутніх на діаграмі Use Case (див рисунок 2.10), та зв’язок з діаграмою класів керування (див рисунок 2.25). Дана діаграма створена з метою трасування зв’язків між елементами.

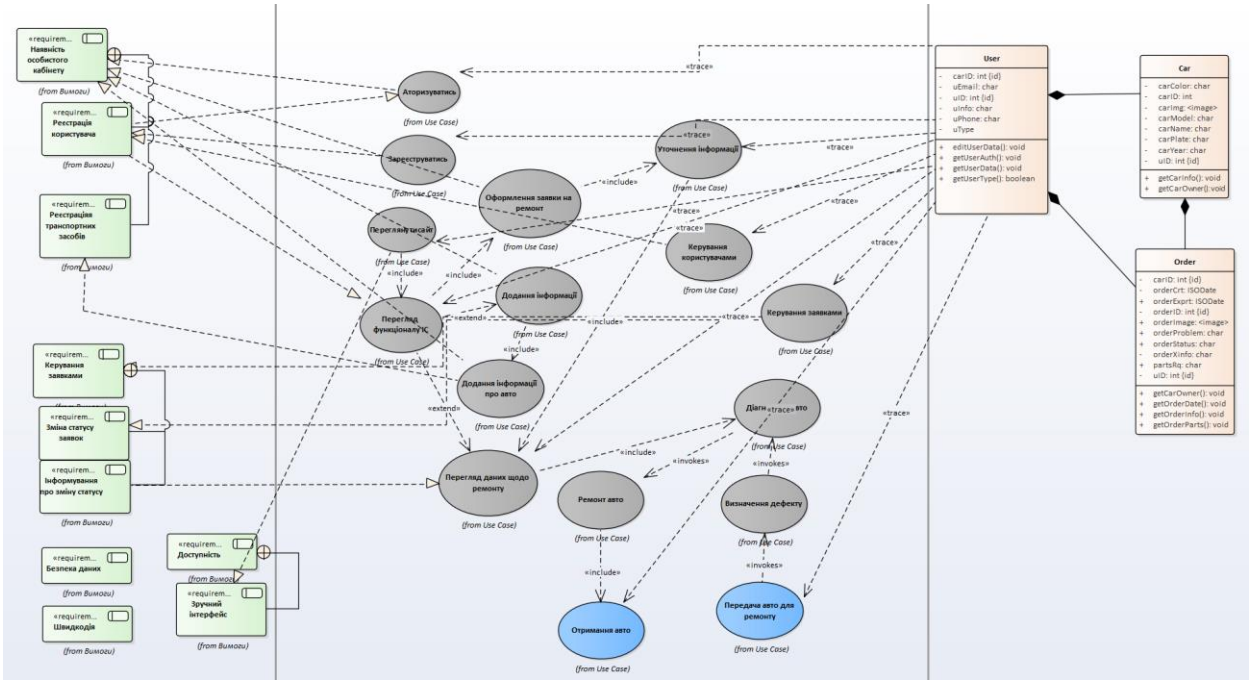


Рисунок 2.26 – Діаграма трасування вимог до системи для визначення потреби в комплектуючих для ремонту автомобілів

Джерело: розроблено автором самостійно

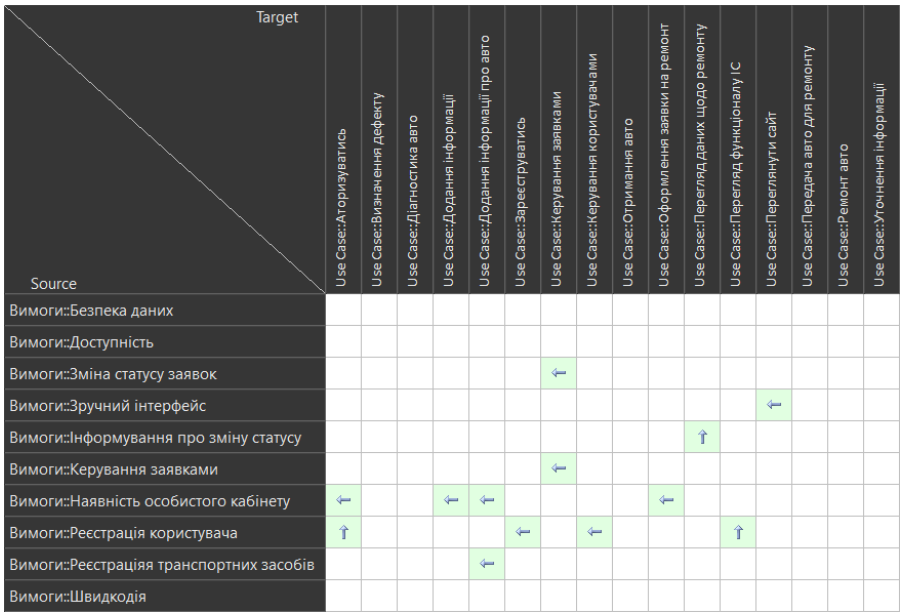


Рисунок 2.27 – Матриця взаємозв’язків прецедентів та вимог

Джерело: розроблено автором самостійно

З метою доповнення діаграми трасування, та подання зв’язків у більш зрозумілому вигляді, було побудовано матрицю взаємозв’язків прецедентів (рисунок 2.27) [17]. Вертикальні стрілки вказують на зв'язок вимог до прецеденту Source-Target, горизонтальні на Target- Source, це дає зрозуміти – вимога надає можливість виконання прецеденту, чи прецедент потребує дану вимогу.

В другому розділі бакалаврського проєкту було встановлено: бізнес вимоги, функціональні вимоги, нефункціональні вимоги, на основі яких було побудовано однойменні діаграми. Було проведено опис задачі, та створена інформаційна модель, побудовано таблиці вхідних та вихідних даних, створено математичні формули, діаграма-алгоритм функціонування задачі. Було виконано моделювання поведінки системи з використанням Use case діаграми, для певних прецедентів були сплановані короткі сценарії, на основі яких були побудовані діаграми послідовностей. Було зображено внутрішню структуру системи у вигляді ієрархії блоків за допомогою діаграми визначення блоків, відображення внутрішньої будови блоків за допомогою діаграми внутрішніх блоків, побудовано діаграми класів, класів керування, граничних класів, побудовано діаграму трасування та матрицю.

РОЗДІЛ 3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

3.1 Інформаційне забезпечення

3.1.1 Загальна характеристика інформаційного забезпечення

Інформаційне забезпечення являється сукупністю різних нормативних документів, інформаційних документів даних, довідників. В склад інформаційного забезпечення системи визначення потреби в комплектуючих для ремонту автомобілів входять: класифікація та кодування, нормативно-довідкові та методичні документи, інформаційна база. Схема інформаційного забезпечення системи наведена на рисунку 3.1.

Кодування виконується для об'єктів: код замовлення, код авто, код користувача. Класифікація виконується для дефектів.

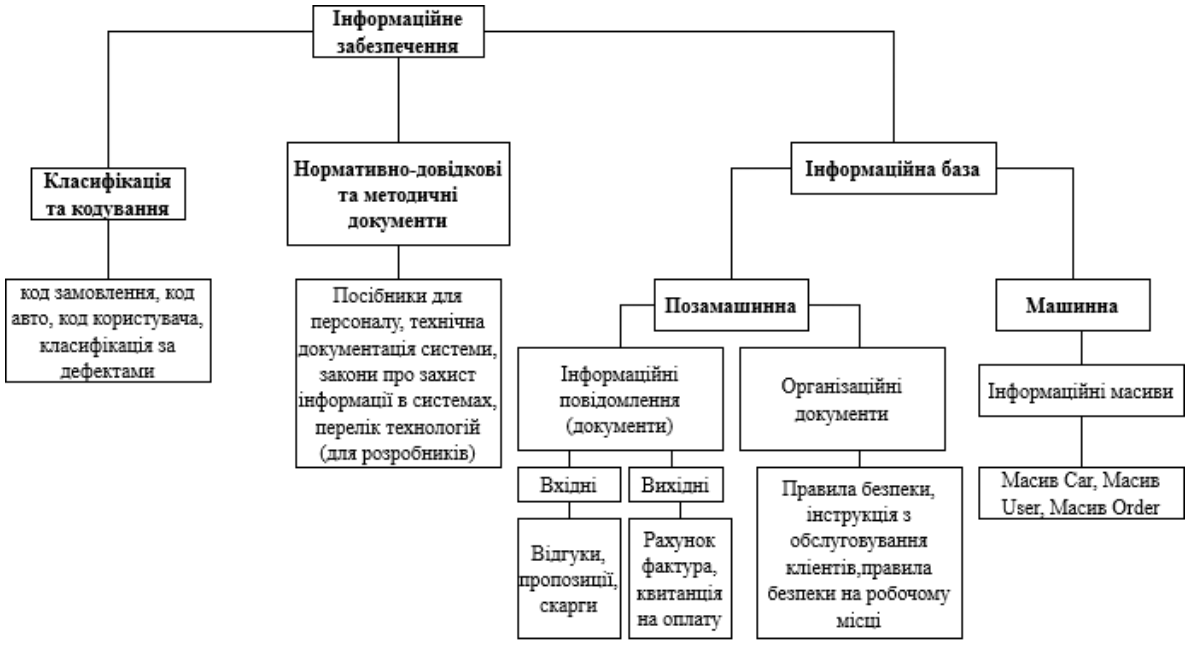


Рисунок 3.1 – Схема інформаційного забезпечення системи

Джерело: розроблено автором самостійно

Нормативно-довідкові та методичні документи можуть включати в себе посібники користувача, для розуміння принципу роботи з системою та базою даних, документацію що стосується будь-якої з технологій, які були

використані під час розробки системи, закони про захист інформації в інформаційно-комунікаційних системах, тощо.

Інформаційна база ділиться на два типи: позамашинну та машинну. До позамашинної інформаційної бази входять інформаційні повідомлення та організаційні документи. Позамашинні інформаційні повідомлення можуть включати в себе будь-яку інформацію вхідну чи вихідну, що передається без використання інформаційної системи. Машинна база включає в себе інформаційні масиви що проходять через систему, включають в себе всю інформацію що можна отримати з вхідних та вихідних даних в інформаційних масивах системи [18].

3.1.2 Організація збору і передавання первинної інформації

Джерелом контактних даних про користувачів та їх відомості про їх авто в систему поступають від самих користувачів. Особистий кабінет слугує місцем отримання цієї інформації. При реєстрації, користувачі надають системі контактну інформацію. Коли користувачі мають облікові записи, вони матимуть можливість реєструвати власні автомобілі, в наслідок чого, джерелом інформації про авто знову виступає користувач. Система повинна дотримуватись певних вимог стосовно передавання та організації інформації і законів щодо захисту інформації в інформаційно-комунікаційних системах [19]. Одним зі способів дотримання цих вимог є передбачити впровадження шифрування з метою захисту даних користувачів. Носії інформації включають: екранні форми, базу даних, певні паперові документи.

3.1.3 Побудова системи класифікації та кодування

Система кодування [20] та класифікації інформаційної системи для визначення потреби в комплектуючих для ремонту автомобілів згідно схеми на рисунку 3.1 включає в себе:

Код Замовлення. Метод кодування: Комбінована система (дата-унікальний ідентифікатор);

Формат: ORD-YYYYMMDD-XXXX;

ORD: Префікс для замовлення;

YYYYMMDD: Дата замовлення;

XXXX: Унікальний ідентифікатор (наприклад, порядковий номер).

Код Авто. Метод кодування: Комбінована система (префікс-код марки-код моделі-рік випуску)

Формат: CAR-MM-YY-RRRR;

CAR: Префікс для авто;

MM: Код марки авто (2 літери);

YY: Код моделі авто (2 літери);

RRRR: Рік випуску.

Код Користувача. Метод кодування: Комбінована система (префікс-ініціали-унікальний ідентифікатор);

Формат: USR-II-XXXX;

USR: Префікс для користувача;

II: Ініціали користувача;

XXXX: Унікальний ідентифікатор.

Класифікація за Дефектами. Метод кодування: Комбінована система (префікс-категорія-підкатегорія-унікальний номер);

Формат: DEF-CC-PP-XXXX;

DEF: Префікс для дефекту;

CC: Категорія дефекту (2 літери);

PP: Підкатегорія дефекту (2 літери);

XXXX: Унікальний номер.

3.1.4 Проєктування форм первинних документів, машинограм та відеокадрів

В якості первинних документів планується використання оповіщень на пошту користувача про успішне оформлення заявки на ремонт та рахунок-фактуру про факт надання послуг з діагностики, обслуговування та ремонту авто користувача. На рисунку 3.2 зображено оповіщення про нову заявку на ремонт, яку планується відправляти на пошту користувачу після заповнення

форми. На рисунку 3.3 зображено вигляд рахунка-фактури про факт надання послуг з діагностики та ремонту авто.

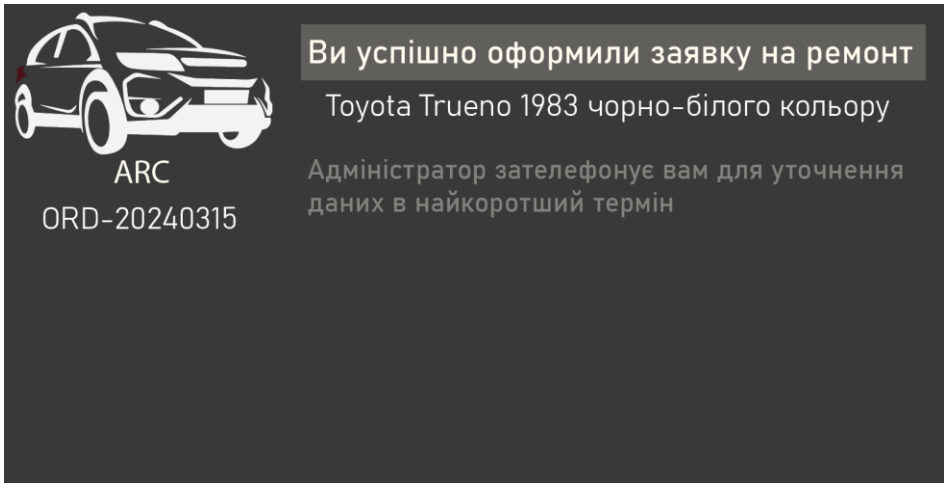
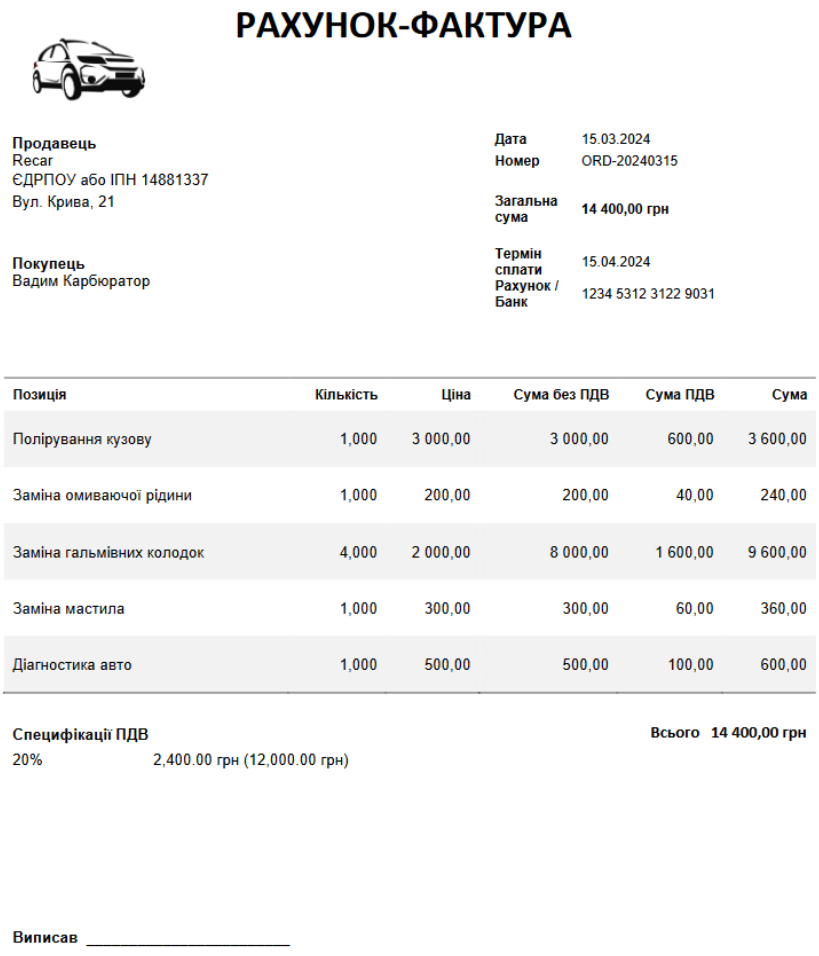


Рисунок 3.2 – Оповіщення про нову заявку на ремонт

Джерело: розроблено автором самостійно



РАХУНОК-ФАКТУРА

Продавець
Recar
ЄДРПОУ або ІПН 14881337
Вул. Крива, 21

Покупець
Вадим Карбюратор

Дата 15.03.2024
Номер ORD-20240315
Загальна сума **14 400,00 грн**
Термін сплати 15.04.2024
Рахунок / Банк 1234 5312 3122 9031

Позиція	Кількість	Ціна	Сума без ПДВ	Сума ПДВ	Сума
Полірування кузову	1,000	3 000,00	3 000,00	600,00	3 600,00
Заміна омиваючої рідини	1,000	200,00	200,00	40,00	240,00
Заміна гальмівних колодок	4,000	2 000,00	8 000,00	1 600,00	9 600,00
Заміна мастила	1,000	300,00	300,00	60,00	360,00
Діагностика авто	1,000	500,00	500,00	100,00	600,00

Специфікації ПДВ
20% 2,400.00 грн (12,000.00 грн)

Всього 14 400,00 грн

Виписав _____

Рисунок 3.3 – Оповіщення про нову заявку на ремонт

Джерело: розроблено автором самостійно

3.1.5 Структура інформаційних масивів

Структура інформаційних масивів буде подана у вигляді таблиць з назвами масивів, ідентифікаторами, типом даних, ключем, умовою значення.

Таблиця 3.1 – Інформаційний масив User

Інформаційний масив User:

Найменування масиву — User;

Ідентифікатор масиву — user_id;

Найменування носія інформації — Жорсткий диск;

Максимальний об’єм масиву — 100,000 записів ;

Довжина запису — 151 символ;

Метод організації — База даних;

Ідентифікатор індексного масиву — user_id.

Назва масиву	Ідентифікатор	Тип даних	Ключ	Умова значення	Обов’язкове поле	Індексне поле
Код користувача	user_id	9(5)	PK	>0	Так	+
Пошта користувача	user_mail	X(45)	-	-	Так	-
Телефон користувача	user_phone	9(10)	-	-	Так	-
Тип користувача	user_type	bool	-	-	Так	-
ПІБ користувача	user_info	X(45)	-	-	Так	-
Пароль користувача	user_pwd	X(45)	-	-	Так	-

Джерело: розроблено автором самостійно

Таблиця 3.2 – Інформаційний масив Car

Інформаційний масив Car

Найменування масиву — Car;

Ідентифікатор масиву — car_id;

Найменування носія інформації — Жорсткий диск;

Максимальний об’єм масиву — 300,000 записів ;

Довжина запису — 194 символів;

Метод організації — База даних;

Ідентифікатор індексного масиву — car_id.

Назва масиву	Ідентифікатор	Тип даних	Ключ	Умова значення	Обов’язкове поле	Індексне поле
Код авто	car_id	9(5)	PK	>0	Так	+
Колір авто	car_color	X(45)	-	-	Так	-
Назва авто	car_name	X(45)	-	-	Так	-
Модель авто	car_model	X(45)	-	-	Так	-
Рік випуску	car_year	9(4)	-	>0	Так	-
Номер	car_plate	X(45)	-	>0	Так	-
Фото авто	car_image	BLOB	-	-	Ні	-
Код користувача	user_id	9(5)	FK	>0	Так	-

Джерело: розроблено автором самостійно

Таблиця 3.3 – Інформаційний масив Order

Інформаційний масив Order

Найменування масиву — Order;

Ідентифікатор масиву — order_id;

Найменування носія інформації — Жорсткий диск;

Максимальний об’єм масиву — 500,000 записів ;

Довжина запису — 352 символів;

Метод організації — База даних;

Ідентифікатор індексного масиву — order_id.

Назва масиву	Ідентифікатор	Тип даних	Ключ	Умова значення	Обов’язкове поле	Індексне поле
Код заявки	Order_id	9(7)	PK	>0	Так	+
Дата створення	Order_date	DATE TIME	-	>0	Так	-
Очікуване завершення робіт	Order_exp_date	DATE TIME	-	-	Ні	-
Проблема	Order_problem	X(45)	-	-	Так	-
Статус ремонту	Order_status	X(45)	-	-	Так	-
Додаткова інформація	Order_extra_info	X(45)	-	-	Ні	-

Назва масиву	Ідентифікатор	Тип даних	Ключ	Умова значення	Обов’язкове поле	Індексне поле
Список деталей	Order_parts	JSON	-	-	Так	-
Фото дефекту	Order_photo	BLOB	-	-	Ні	-
Код користувача	user_id	9(5)	FK	>0	Так	-
Код авто	car_id	9(5)	FK	>0	Так	-

Джерело: розроблено автором самостійно

3.1.6 Вибір СКБД

В ході вибору СКБД, було прийнято рішення обрати реляційну систему MySQL версії 8.0.32 зі стандартним пакетом для розробки. MySQL - це клієнт-серверна реляційна система управління базами даних з відкритим вихідним кодом, яка використовує мову структурованих запитів (SQL) для управління даними [21]. Широко використовується завдячуючи своїй швидкості, надійності та простоті використання. MySQL була розроблена шведською компанією MySQL AB у 1995 році. У 2008 році її придбала компанія Sun Microsystems, яку згодом у 2010 році, придбала компанія Oracle. На 2024 MySQL досі являється програмним продуктом компанії Oracle [22].

Оскільки MySQL є клієнт-серверною СКБД, система передбачає наявність серверного демону (mysqld), що керує файлами бази даних, приймає та обробляє SQL-запити і повертає результати клієнтським програмам.

MySQL реляційна система, тому, дані зберігаються в таблицях, що складаються з рядків і стовпців. Кожна таблиця представляє окрему сутність, а зв'язки між цими таблицями встановлюються за допомогою первинних і зовнішніх ключів, підтримує широкий спектр типів даних, включаючи числові, дати і часу, а також рядкові типи даних. Крім того, вона включає такі функції, як транзакції, які дозволяють виконувати кілька SQL-операцій в якості однієї. За рахунок чого, забезпечується цілісність і узгодженість даних, особливо в сервісах, що вимагають синхронного внесення декількох змін. Така

реляційна структура робить MySQL придатною для обробки структурованих даних і складних запитів. Ключовими властивостями СКБД MySQL є:

- Масштабованість. MySQL може працювати як з невеликими додатками з кількома таблицями та малою кількістю записів так і з габаритними системами з мільйонами записів та складними запитами до бази даних. Досягається це, завдяки такій функції як, реплікація, коли дані з одного сервера MySQL копіюються на інший, полегшується розподіл навантаження та передбачається аварійне відновлення. Функція MySQL Cluster дозволяє розподіляти бази даних між кількома вузлами, забезпечуючи високу доступність і масштабованість для великих і складних систем.
- Безпека. MySQL надає надійні механізми для захисту даних. Вона підтримує аутентифікацію користувачів, дозволяючи адміністраторам баз даних створювати облікові записи користувачів і керувати ними. Тільки авторизовані користувачі можуть мати доступ до даних, або вносити зміни. Є підтримка протоколу SSL (Secure Sockets Layer) для шифрування даних, що передаються між сервером MySQL і клієнтськими програмами.
- Продуктивність. Досягається дана властивість, за допомогою різних параметрів конфігурації та інструментів. Сервер MySQL можна налаштувати для оптимізації продуктивності залежно від навантаження, наприклад, змінити розмір кешу, параметри буфера та способи оптимізації запитів. Такі інструменти, як MySQL Performance Schema та MySQL Enterprise Monitor, дають уявлення про продуктивність бази даних і допомагають виявити слабкі точки та області для покращення.
- Інтеграція. MySQL досить добре взаємодіє з різними мовами програмування та платформами, що робить її популярним вибором для веб-додатків, особливо тих, що створені за допомогою стеку LAMP (Linux, Apache, MySQL, PHP/Perl/Python). Більшість сервісів розробки передбачають наявність спеціальних коннекторів, які

з’єднують сервіс з БД, що дозволяє розробникам створювати додатки, які взаємодіють з базами даних MySQL.

- Спільнота. Популярність сервісу дуже часто гарантує те, що спільнота буде великою. Розробниками передбачається документація, але підтримка користувачів дозволяє вирішувати питання чи проблеми інших користувачів, працювати сумісно над проєктами. Одним з прикладів плодів роботи спільноти є сервіс адміністрування phpMyAdmin, що дозволяє керувати сервером, створювати запити, редагувати чи переглядати записи баз даних MySQL та MariaDB [23].

3.1.7 Інфологічна модель бази даних

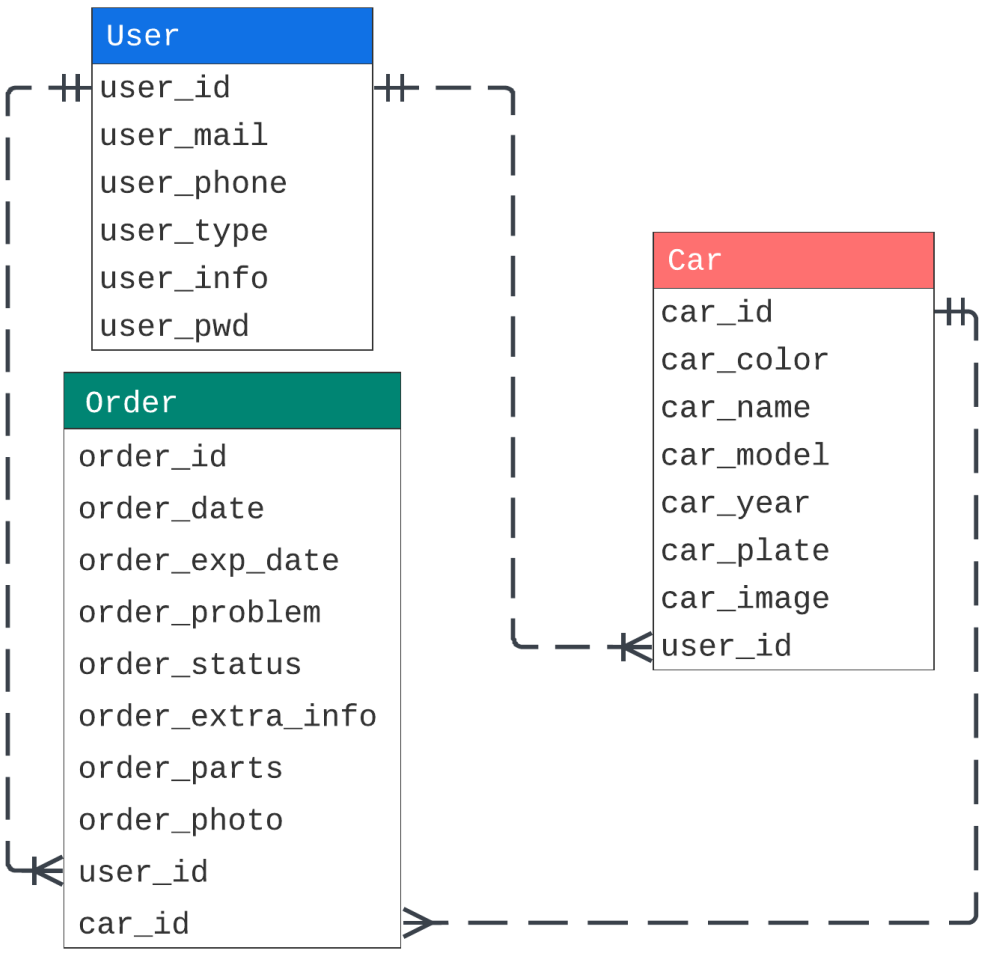


Рисунок 3.4 – Інфологічна модель бази даних

Джерело: розроблено автором самостійно

Інфологічна модель бази даних зображена на рисунку 3.4 була побудована в середовищі Lucidchart [24], модель складається з 3-х таблиць та відображає таблиці 6-8 в загальному вигляді, для того, щоб показати як виглядає логіка зберігання інформації та які таблиці пов’язані між собою.

3.1.8 Даталогічна модель бази даних

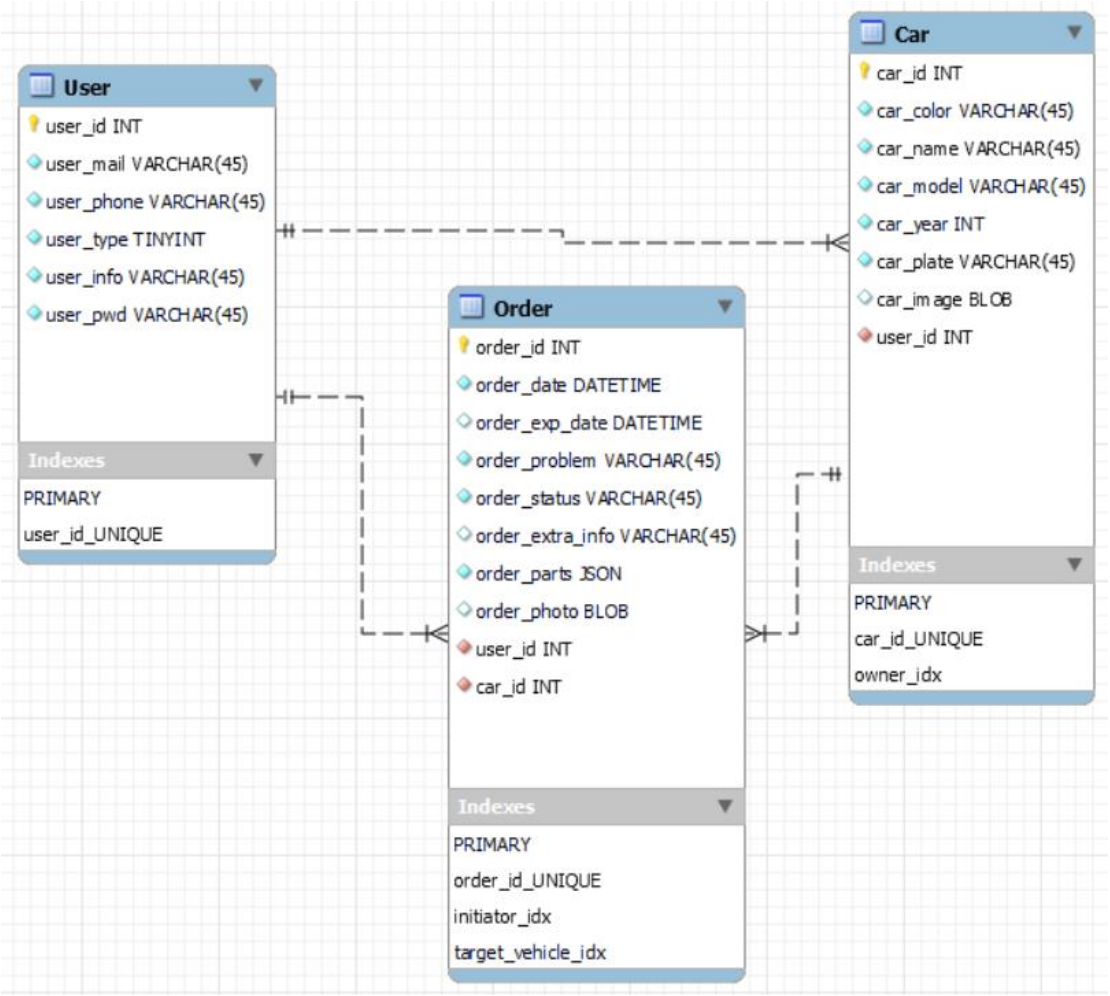


Рисунок 3.5 – Даталогічна модель бази даних

Джерело: розроблено автором самостійно

Даталогічна модель бази даних зображена на рисунку 3.5 була створена в середовищі MySQL Workbench [25] та відображає таблиці 6-8 вже в робочому вигляді, зі зв’язками та первинними і вторинними ключами.

З діаграми можна побачити залежність таблиці Order від Car та User, логіка полягає в тому, що автомобіль в системі не може існувати без власника, відповідно замовлення не може існувати без предмету над яким виконується

операція діагностики і ремонту, а також без ініціатора цього замовлення, тобто користувача.

3.2 Технічне забезпечення

Як зазначалось раніше, інформаційна система для визначення потреби в комплектуючих для ремонту автомобілів виступає в ролі веб-ресурсу що розміщується в мережі інтернет на обладнанні яке одночасно слугує і носієм модулів інформаційної системи, і сервером бази даних цієї системи. Оскільки система не вимагає високої обчислювальної потужності, можливо зберігати все на одній машині.

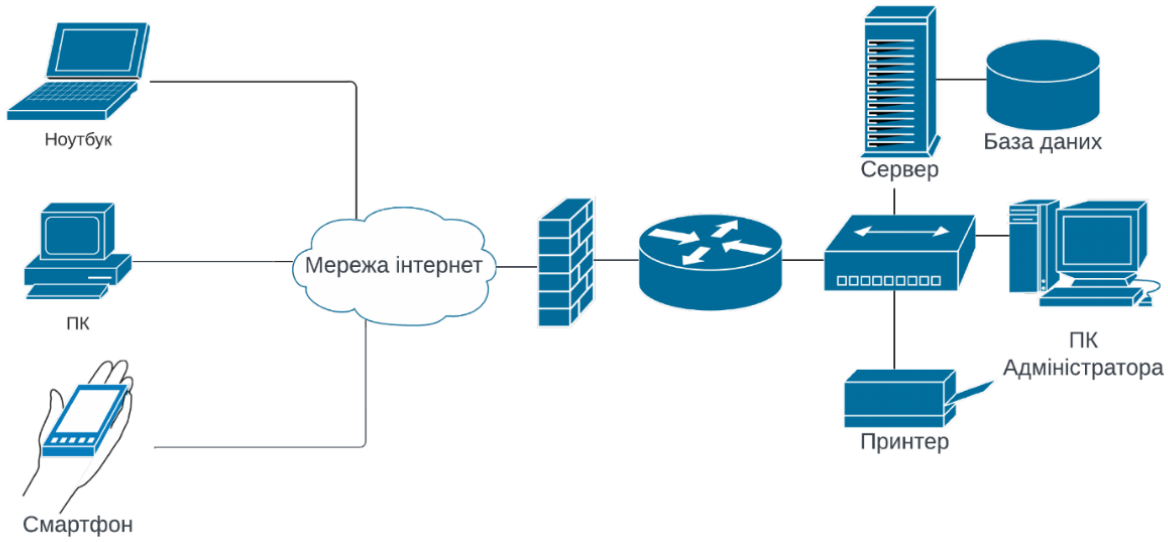


Рисунок 3.6 – Схема взаємодії технічних засобів в системі

Джерело: розроблено автором самостійно

Користувачі можуть потрапити на веб-ресурс використовуючи ноутбук, пк, чи телефон через мережу інтернет. Інформаційна система зберігається на сервері, що одночасно зберігає як і самі модулі, так і базу даних. Сервер є частиною локальної мережі, що об’єднується за допомогою комутатора, в мережу також входять пк адміністратора чи адміністраторів, та принтер який за потреби друкує документи, накладні, тощо. Обладнання локальної мережі підключається через обладнання провайдера до мережі інтернет, та надається доступ лише до сервісу, що розміщується на сервері.

Структура комплексу технічних засобів

Структура комплексу технічних засобів складається з:

Користувачів, або клієнтів, людей що бажають отримати послуги що надає інформаційна система для визначення потреби в комплектуючих для ремонту автомобілів. Ці користувачі використовують ноутбуки, персональні комп'ютери чи смартфони що можуть бути підключені до мережі інтернет бездротовим, або дротовим зв'язком.

Серверу, спеціального комп'ютера на якому розміщуються модулі системи та база даних, де зберігається інформація про користувачів, їх авто та заявки на ремонт.

Схема мережі передачі даних

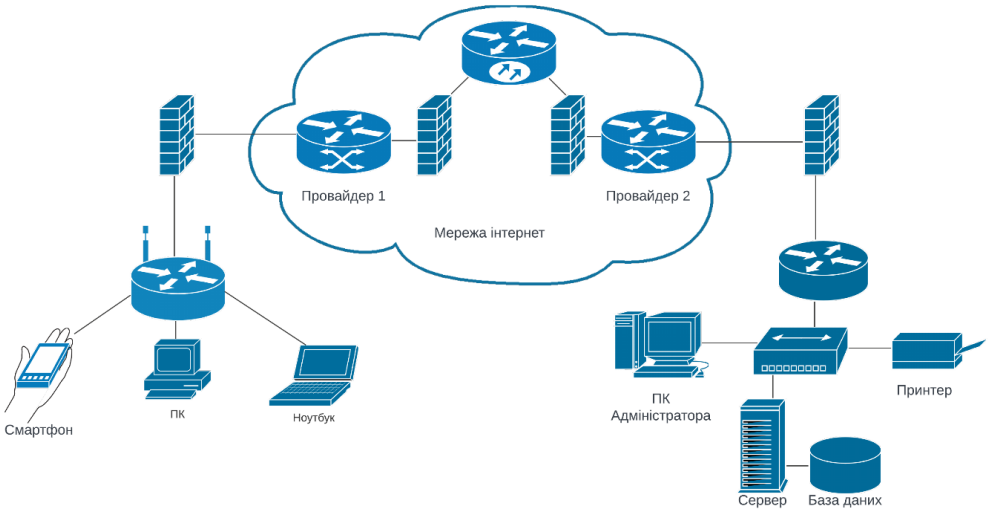


Рисунок 3.7 – Схема мережі передачі даних

Джерело: розроблено автором самостійно

Схема на рисунку 3.7 демонструє більш детально принцип передачі даних в системі. Доступ користувачів та системи для визначення потреби в комплектуючих для ремонту автомобілів здійснюється за допомогою технічного обладнання провайдерів.

Локальна мережа користувача/клієнта може складатись з декількох пристроїв з доступом до мережі інтернет та поєднаних між собою дротовим чи бездротовим каналом зв'язку який організовую маршрутизатор, він може бути особистим чи надаватись провайдером в рамках контракту про надання

послуг. Клієнту надається доступ до мережі інтернет за допомогою одного з безлічі провайдерів, провайдери в свою чергу взаємодіють між собою та передають дані. Клієнтська лінія зв’язку не має встановлених вимог, оскільки всі вимоги до якості надання послуг суб’єктивні для кожного користувача, але мінімальний рекомендований стандарт лінії передачі зв’язку для клієнта це ethernet 100Base-T4, тобто підключення до постачальника послуг через кабель типу вита пара з пороговою швидкістю до 1 Гбіт/с.

Інформаційна система, як було зазначено раніше, розміщується разом з базою даних на одному сервері, що підключається до локальної мережі за допомогою мережевого комутатора. Локальна мережа може включати в себе принтер та комп’ютер адміністратора для керування системою та базою даних. Поширена методика стосовно вибору обладнання – це проєктування системи «на виріст», тобто передбачити, що система та обсяг даних що в ній рухаються буде більше. Для локальної мережі передбачається встановлення комутатор з гігабітними портами, тобто підтримка швидкості прийому та передачі 1 Гбіт/с, швидкість обумовлена можливим розширенням системи. Обладнання для доступу до провайдеру має бути зі швидкістю від 1 Гбіт/с, підключенням ethernet 1000Base для забезпечення швидкого доступу до сервісу при великій кількості одночасних звернень. При цьому, необхідно мати сервіс від провайдера що надає фіксовану ip-адресу, тобто адресу що по контракту з провайдером не змінюється в кінці місяця [26].

Обладнання провайдеру залежить від самого провайдеру зв’язку і схема їх зв’язку проєктується професіоналами у своїй справі і частіше за все є конфіденційною інформацією, тому надати рекомендації неможливо.

3.3 Програмне забезпечення

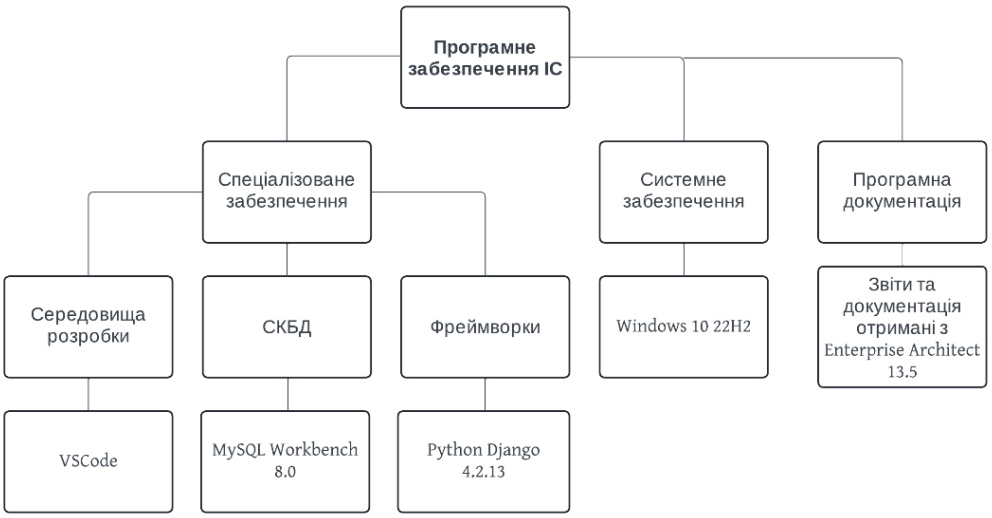


Рисунок 3.8 – Схема структури програмного забезпечення системи

Джерело: розроблено автором самостійно

Структура програмного забезпечення системи зображена у вигляді схеми на рисунку 3.8. Програмне забезпечення включає в себе спеціалізоване забезпечення, системне забезпечення та програмну документацію.

Спеціалізоване забезпечення

Середовищем розробки коду є програмне забезпечення Microsoft Visual Studio Code версії 1.89.1, обрано дане середовище розробки через модульність, що дозволяє встановити лише необхідні додатки для роботи з певними мовами програмування [27].

Для побудови базових діаграм та алгоритмів були використані веб-сервіси drawio [28] та Lucidchart. Для побудови діаграм методології OPM було використано середовище OPCAT.

Для побудови розширених діаграм, вимог, сценаріїв, графіків було використано середовище візуального моделювання Sparx Systems Enterprise Architect версії 13.5. Перевагою цього засобу над онлайн редакторами є тим, що цей продукт був створений як виділений супровідний інструмент під час проєктування систем, за допомогою якого можна відслідковувати процес

створення продукту від стадії проєктування, до подальшого циклу розробки, включаючи розширення продукту.

В якості бази даних, було обрано базу даних MySQL версії 8.0.32. MySQL являється реляційною базою даних, що полегшує роботу з нею у зв’язку з більшою кількістю документації та підтримки. Для перегляду та керування таблицями бази даних обрано графічну оболонку що постачається зі встановленням повного пакету розробки MySQL - MySQL Workbench 8.0. Ключовою перевагою використання графічної оболонки є зрозумілість і легкість в використанні. Workbench дозволяє створювати бази даних, або редагувати вже існуючі, створювати запити, переглядати дані в базах, видаляти чи додавати нові записи, відображати це в графічному форматі, або у вигляді SQL запитів.

Для створення веб-ресурсу було обрано фреймворк Django, що базується на мові програмування Python. Django це відкритий фреймворк для веб-розробки. Ціль створення даного фреймворку полягала у спрощенні роботи для початківців, надаючи при цьому досить розширений функціонал.

Наявність деяких компонентів при встановленні з однієї сторони є одночасно і зручним для тих, хто тільки починає, і трохи дратуючим для тих, хто хоче повний контроль над своїм проєктом. За потреби, завжди є можливість розширити застосунок і Django справляється з цією задачею без надскладних операцій. Ключовими перевагами для вибору саме цього фреймворку стали: система об'єктно-реляційного відображення (ORM) та вбудований захист від базових атак. ORM надає розробникам можливість взаємодіяти з базою даних за допомогою моделей на мові Python замість написання SQL запитів, при цьому підтримує складні запити та зв’язки між різними моделями даних. Захист від базових атак полягає у наявності вбудованого захисту від атак типу: SQL-ін'єкцій, міжсайтового скриптингу (XSS), підробки міжсайтових запитів (CSRF) [29].

Системне забезпечення

Операційною системою для проєктування системи визначення у комплектуючих для ремонту авто є Microsoft Windows 10 Pro версії 22H2

збірка 19045, обрана саме ця система через легкість використання системи типу Windows NT.

Програмна документація

Програмна документація включає в себе всі матеріали отримані в результаті побудови схем, діаграм та сценаріїв в середовищі Enterprise Architect 13.5. Ця документація складається з:

- Керівництва персоналу обслуговування. Що містить інформацію про систему, порядок її обслуговування, налаштування та створення копій;
- контрольний приклад роботи. Містить інформацію про коректну роботу інформаційної системи.

Прикладне програмне забезпечення

Прикладне програмне забезпечення створюється для виконання конкретних задач поставлених певною групою людей, складність задач може варіюватись, проєктована інформаційна система включає в себе декілька категорій програмного забезпечення:

- Текстові редактори. Для роботи з текстовими даними, написанням звітів планується використовувати редактор Microsoft Office Word;
- табличний редактор. Для роботи з таблицями у вигляді кошторисів, списку запчастин чи інших бухгалтерських документів використовується редактор Microsoft Office Excel;
- графічні процесори. Для відображення користувацьких фото планується використовувати редактор растрової графіки Adobe Photoshop;
- редактор презентацій. Для зручної та швидкої подачі певної звітної інформації можливе використання презентацій, для створення яких використовується редактор Microsoft Office Powerpoint;

- системи управління базами даних. Як зазначалось раніше, планується використовувати базу даних MySQL з графічною оболонкою MySQL Workbench.

3.4 Результат реалізації інформаційної підсистеми

Процес роботи з системою наступний – користувач/ клієнт заходить на ІС у вигляді веб-ресурсу та авторизується чи реєструється для отримання послуг з визначенням неполадок транспортного засобу та в подальшому ремонті цього дефекту при узгодженні з клієнтом.

Початкова версія ресурсу має надавати можливість клієнтам заходити в свій особистий кабінет на сайті, мати контактні дані, додавати транспортний засіб, дивитись історію звернень та їх статус, можливість створювати звернення та перевіряти їх статус. Функціональний прототип у вигляді зв’язаних екранних форм ІС був розроблений в середовищі Figma [30].

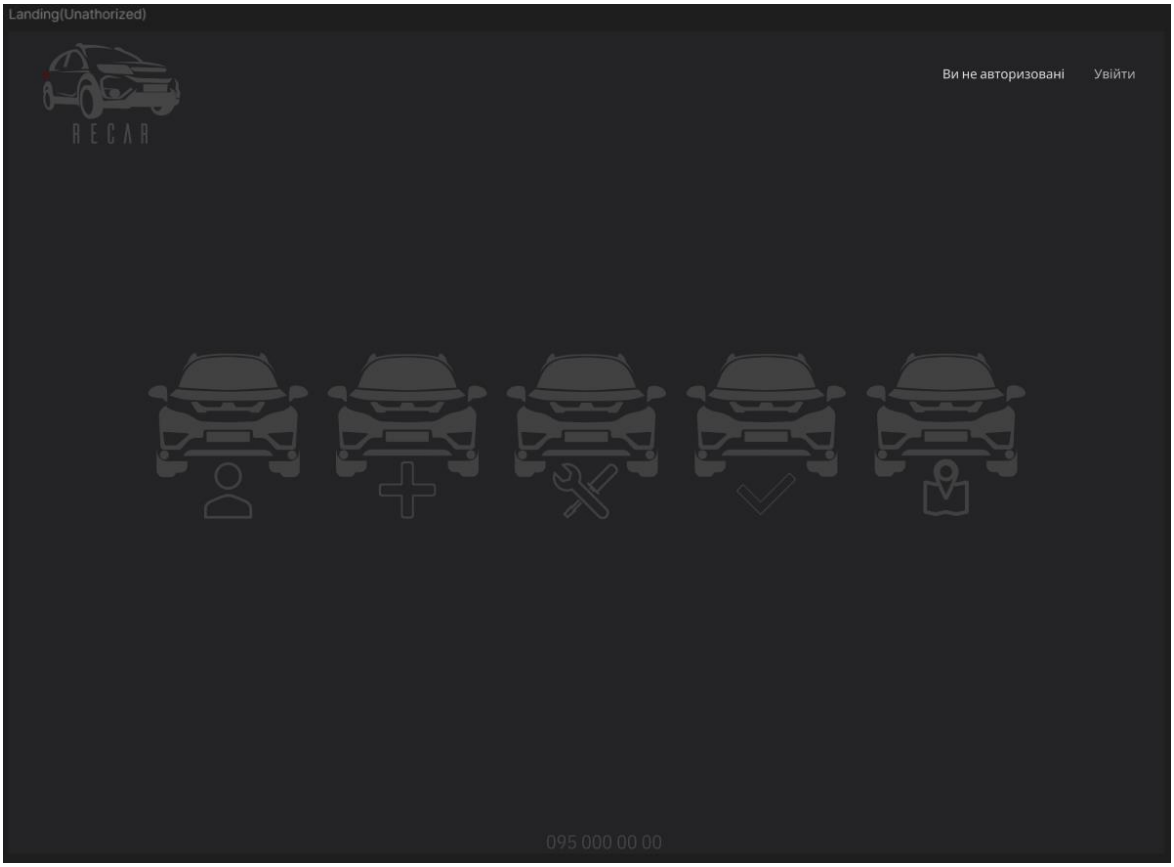


Рисунок 3.9 – Головна сторінка (користувач неавторизований)

Джерело: розроблено автором самостійно

Головна сторінка, або лендінг (рисунок 3.9) містить кнопку (клікабельний логотип сайту) з переходом на домашню сторінку, містить 5 неактивних піктограм, кнопку авторизації.

Рисунок 3.10 – Сторінка авторизації користувача

Джерело: розроблено автором самостійно

Сторінка авторизації (рисунок 3.10) містить поле введення пошти користувача та паролю, кнопку авторизації на сайті, та кнопку реєстрації нового користувача.

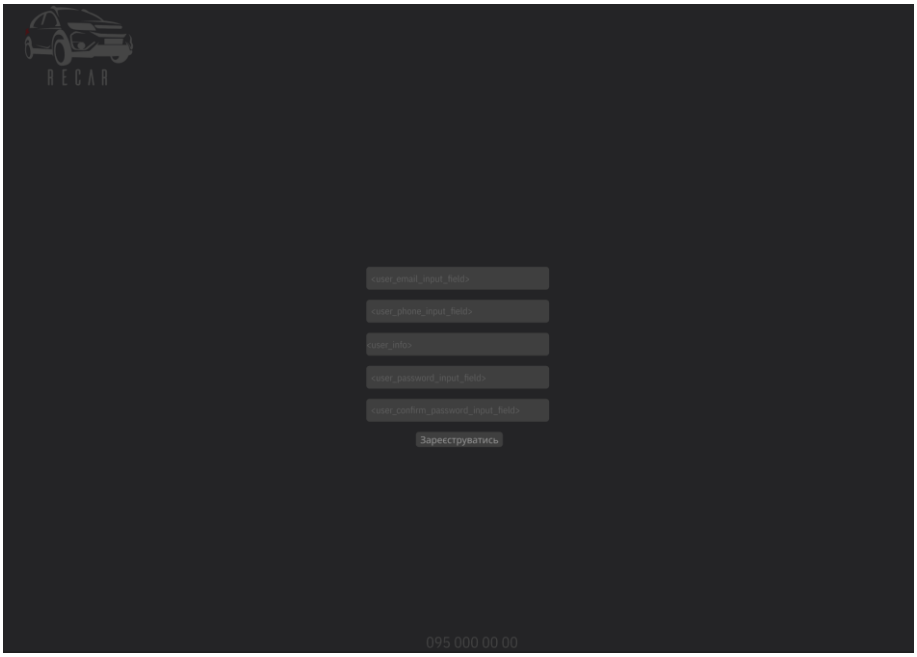


Рисунок 3.11 – Сторінка форми реєстрації нового користувача

Джерело: розроблено автором самостійно

Сторінка реєстрації нового користувача (рисунок 3.11) містить поля введення піб користувача, пошти користувача, паролю та телефону. Після натискання на кнопку реєстрації, та успішній реєстрації користувача відбувається перехід на головну сторінку ІС.

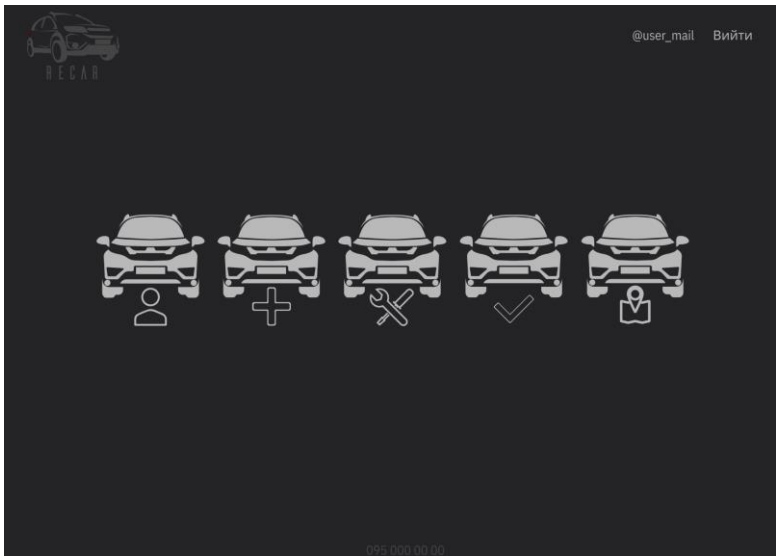


Рисунок 3.12 – Головна сторінка

Джерело: розроблено автором самостійно

Головна сторінка (рисунок 3.12) після авторизації активує 5 піктограм та змінює кнопку входу на пошту користувача, яка слугує кнопкою переходу в особистий кабінет. Роль кнопок-піктограм зліва-направо: Особистий кабінет користувача, Реєстрація нового транспортного засобу, Нове звернення на ремонт, Перевірка статусу звернення, Місцезнаходження сервісу.

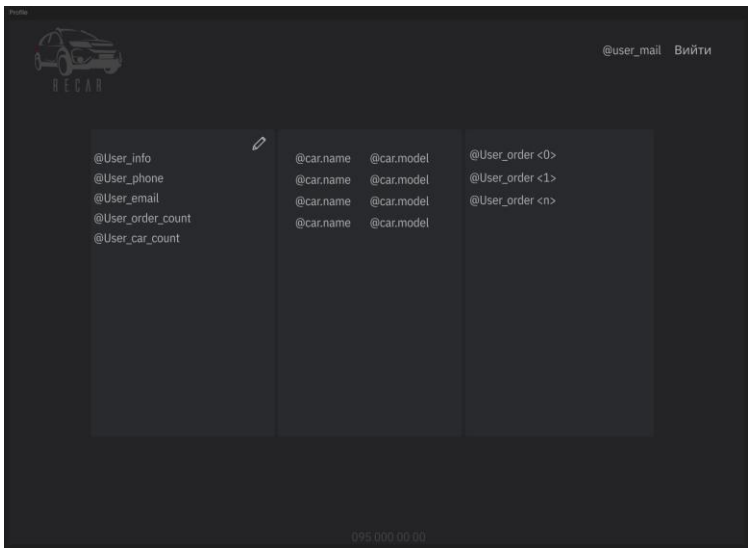


Рисунок 3.13 – Сторінка особистого кабінету

Джерело: розроблено автором самостійно

Сторінка особистого кабінету (рисунок 3.13) містить контактні дані користувача, список транспортних засобів (який при реєстрації є порожнім) та історія замовлень (теж є порожніми при реєстрації).

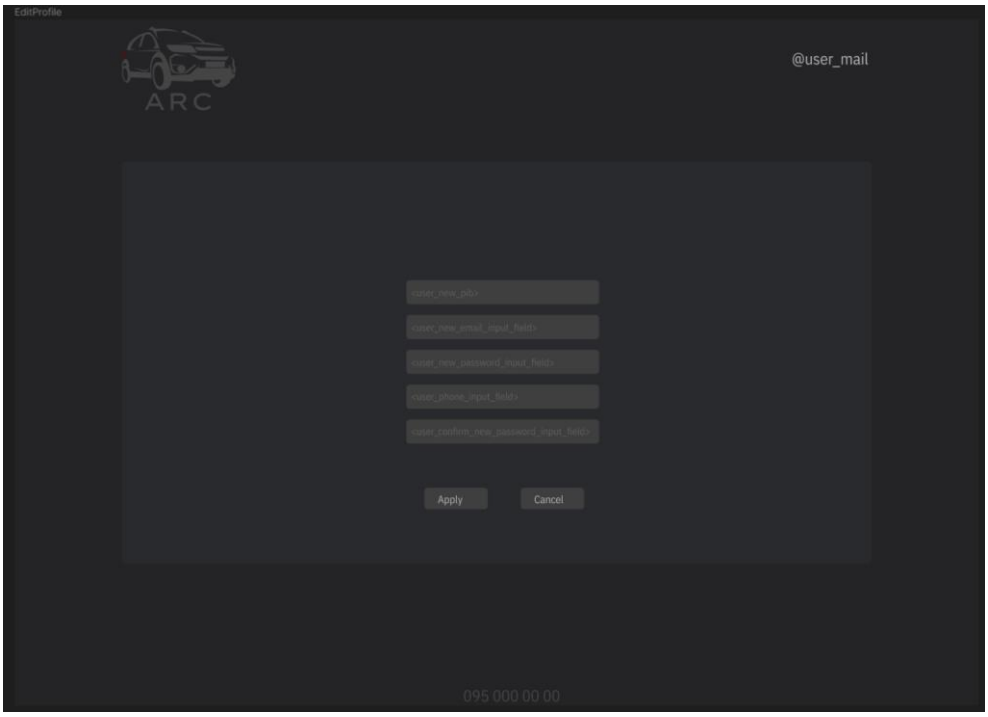


Рисунок 3.14 – Сторінка зміни даних

Джерело: розроблено автором самостійно

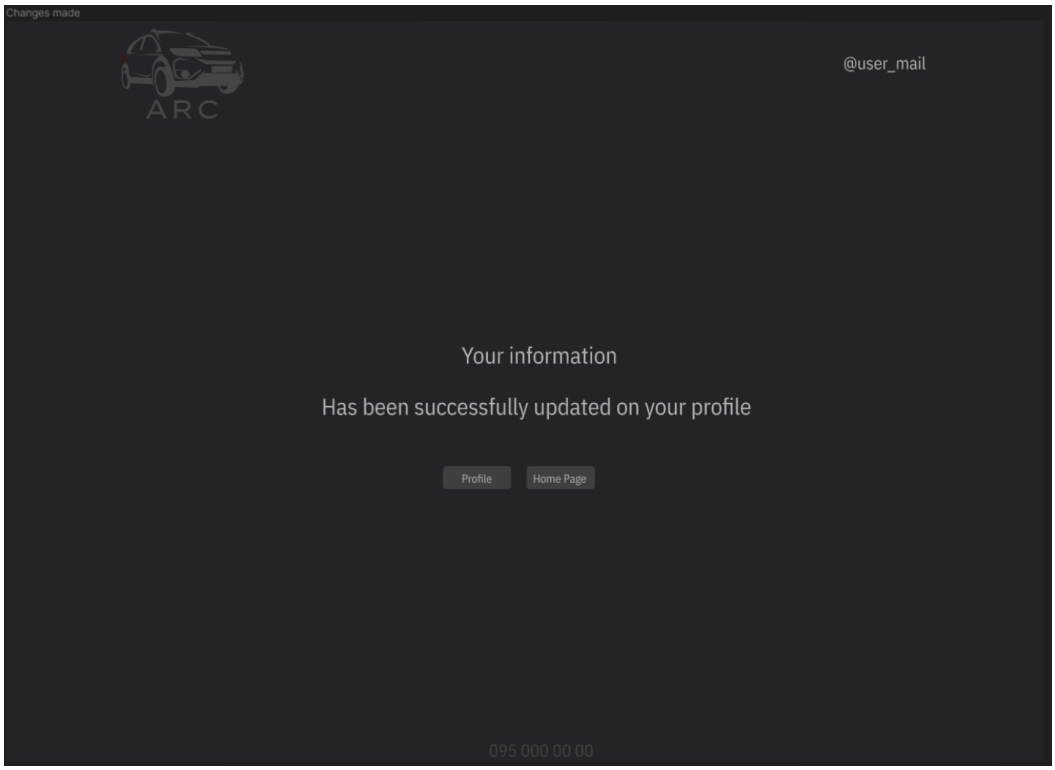


Рисунок 3.15 – Успішна зміна даних

Джерело: розроблено автором самостійно

Сторінка зміни особистих даних (рисунок 3.14) трохи ідентична формі реєстрації та дозволяє змінити все за одну операцію. При успішній зміні, відбувається інформування користувача про дану операцію (рисунок 3.15)

Рисунок 3.16 – Сторінка форми реєстрації нового транспортного засобу

Джерело: розроблено автором самостійно

Сторінка реєстрації нового транспорту (рисунок 3.16) містить поля введення марки автомобіля, моделі автомобіля, колір, номер, рік автомобіля, додаткову інформацію та зображення транспортного засобу.

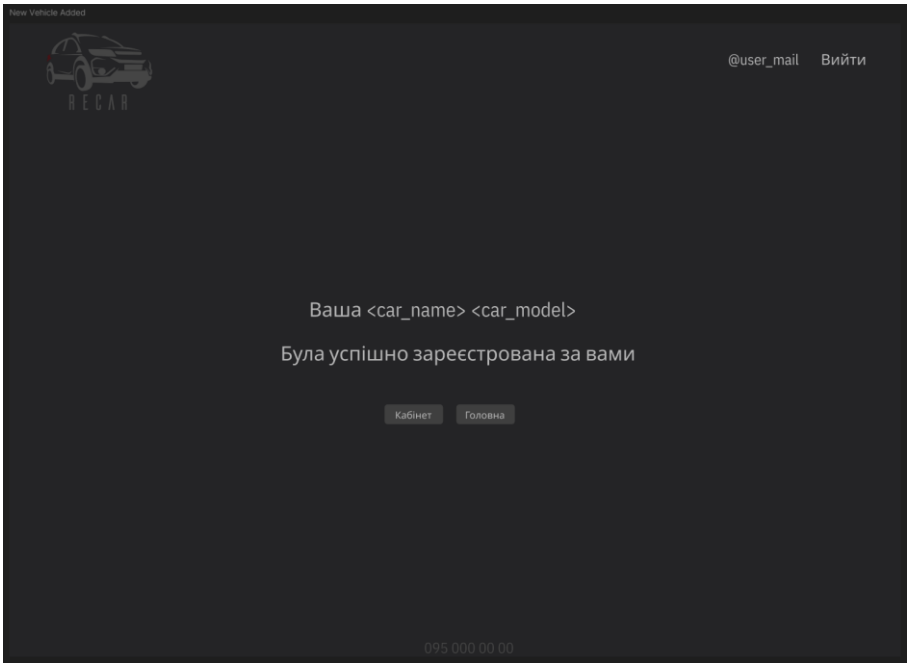


Рисунок 3.17 – Сторінка про успішне додання авто

Джерело: розроблено автором самостійно

Сторінка підтвердження реєстрації нового транспорту (рисунок 3.17) показує інформацію про успішно доданий автомобіль, присутні кнопки переходу на профіль чи на домашню сторінку.

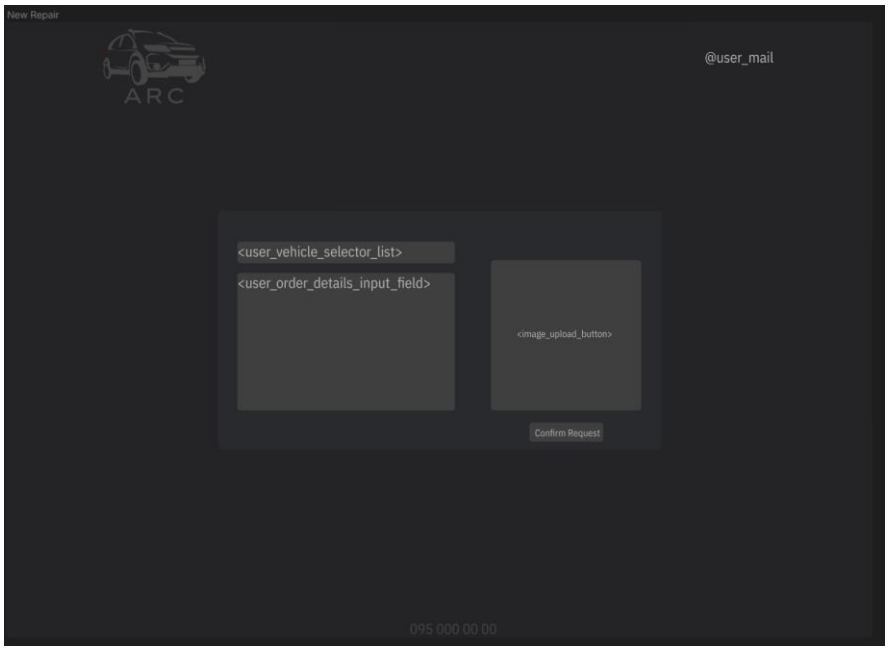


Рисунок 3.18 – Сторінка форми нового звернення за ремонтом

Джерело: розроблено автором самостійно

Сторінка форми звернення за ремонтом нового транспорту (рисунок 3.18) має список з можливістю вибору зареєстрованого автомобіля користувача, допис тексту користувачем про симптоматику, та можливим прикріпленням фотографії дефективної зони.

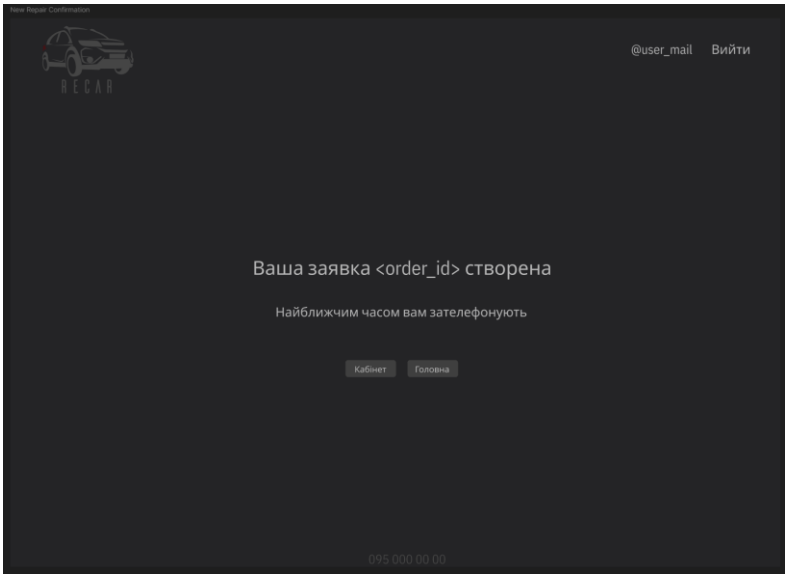


Рисунок 3.19 – Сторінка підтвердження звернення

Джерело: розроблено автором самостійно

Сторінка підтвердження звернення (рисунок 3.19) показує номер щойно створеного звернення та має кнопки виходу на головну сторінку чи на профіль.

Рисунок 3.20 – Сторінка вибору заявки для перевірки статусу звернення

Джерело: розроблено автором самостійно

Сторінка перевірки заявки (рисунок 3.20) має поле зі списком заявок-звернень користувача за послугами з ремонту.

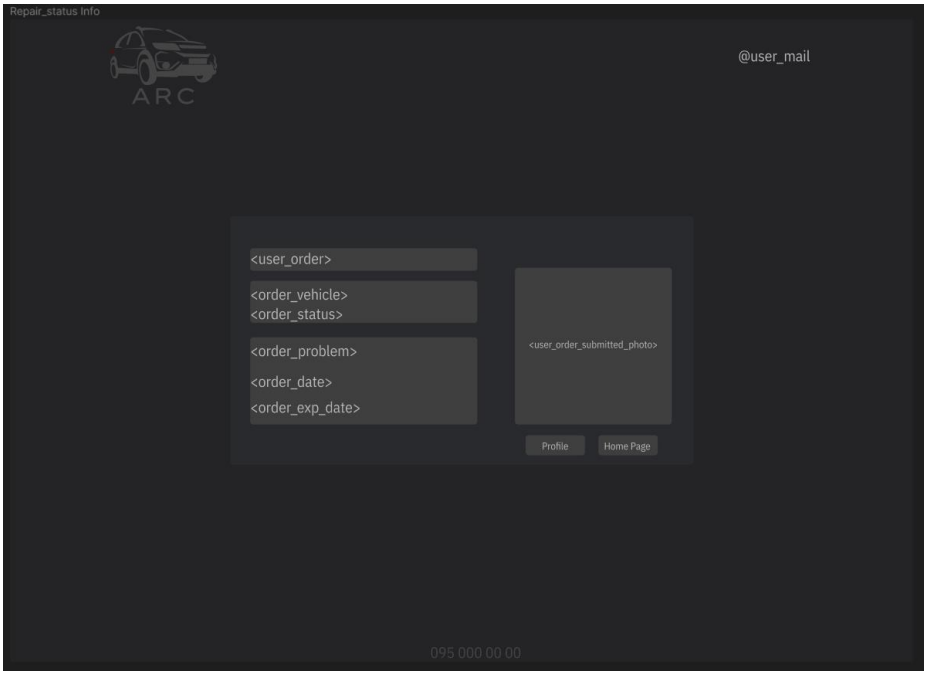


Рисунок 3.21 – Сторінка інформації стосовно звернення

Джерело: розроблено автором самостійно

Сторінка перевірки заявки (рисунок 3.21) містить інформацію про заявку, транспортний засіб, статус заявки, проблему визначену експертами, очікуваний час ремонту та зображення що міг прикріпити користувач.

Рисунок 3.22 – Схема переходів між сторінками

Джерело: розроблено автором самостійно

Схема переходів (рисунок 3.22) показує як сторінки прототипу пов’язані між собою, так як з однієї сторінки можна потрапити на іншу.

Реалізація системи на основі прототипу включає в себе наступні функціональні сторінки: неактивну головну сторінку, форму авторизації, форму реєстрації, активну головну сторінку, особистий кабінет, сторінку-форму реєстрації нового автомобіля в системі.

Рисунок 3.23 – Неактивна головна сторінка

Джерело: розроблено автором самостійно

Реалізація лендінгу (рисунок 3.23) містить кнопку (клікабельний логотип сайту) з переходом на домашню сторінку, містить 5 неактивних піктограм, кнопку авторизації.

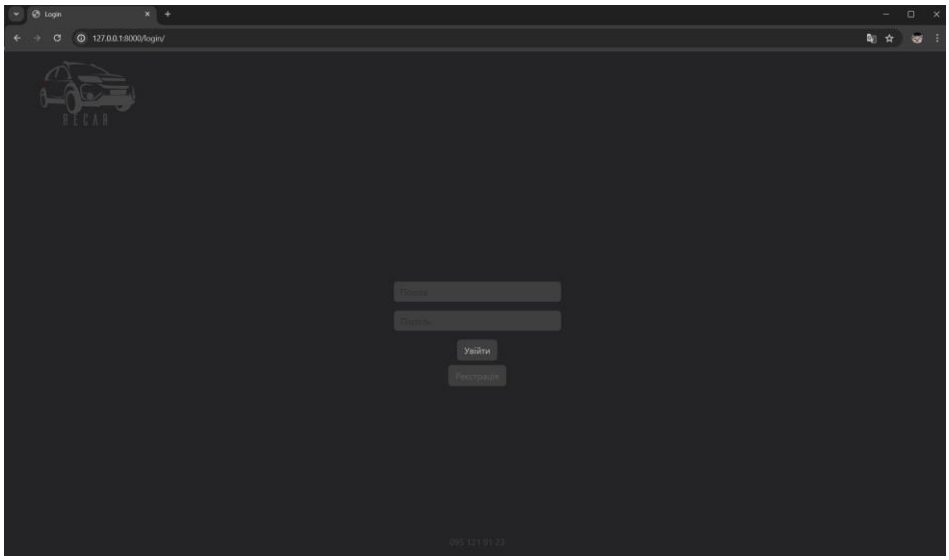


Рисунок 3.24 – Авторизація

Джерело: розроблено автором самостійно

Сторінка авторизації (рисунок 3.24) містить поле введення пошти користувача та паролю, кнопку авторизації на сайті, та кнопку реєстрації нового користувача.

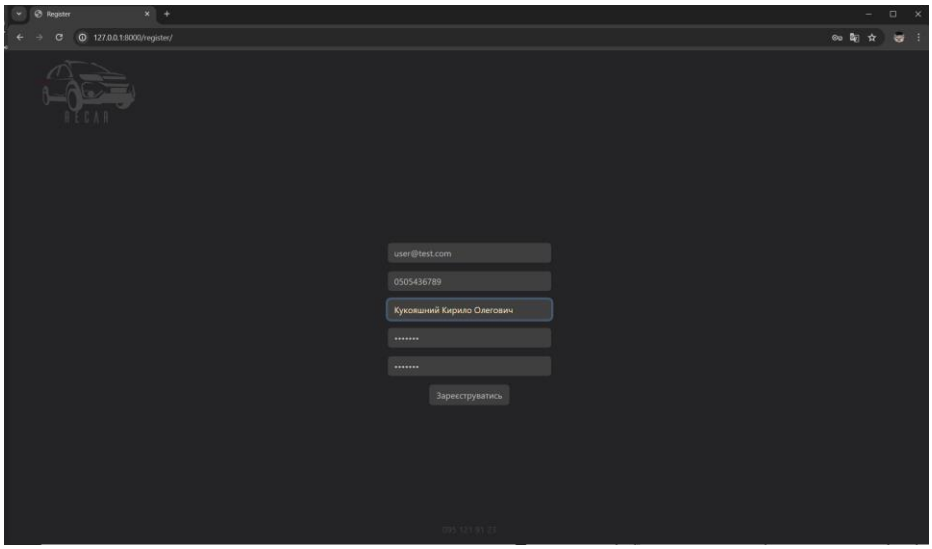


Рисунок 3.25 – Реєстрація користувача

Джерело: розроблено автором самостійно

Сторінка реєстрації нового користувача (рисунок 3.25) містить поля введення піб користувача, пошти користувача, паролю та телефону. Після

натискання на кнопку реєстрації, та успішній реєстрації користувача відбувається перехід на головну сторінку ІС.

Рисунок 3.26 – Головна сторінка авторизованого користувача

Джерело: розроблено автором самостійно

Головна сторінка (рисунок 3.26) після авторизації активує 5 піктограм та змінює кнопку входу на пошту користувача, яка слугує кнопкою переходу в особистий кабінет. Роль кнопок-піктограм зліва-направо: Особистий кабінет користувача, Реєстрація нового транспортного засобу, Нове звернення на ремонт, Перевірка статусу звернення, Місцезнаходження сервісу

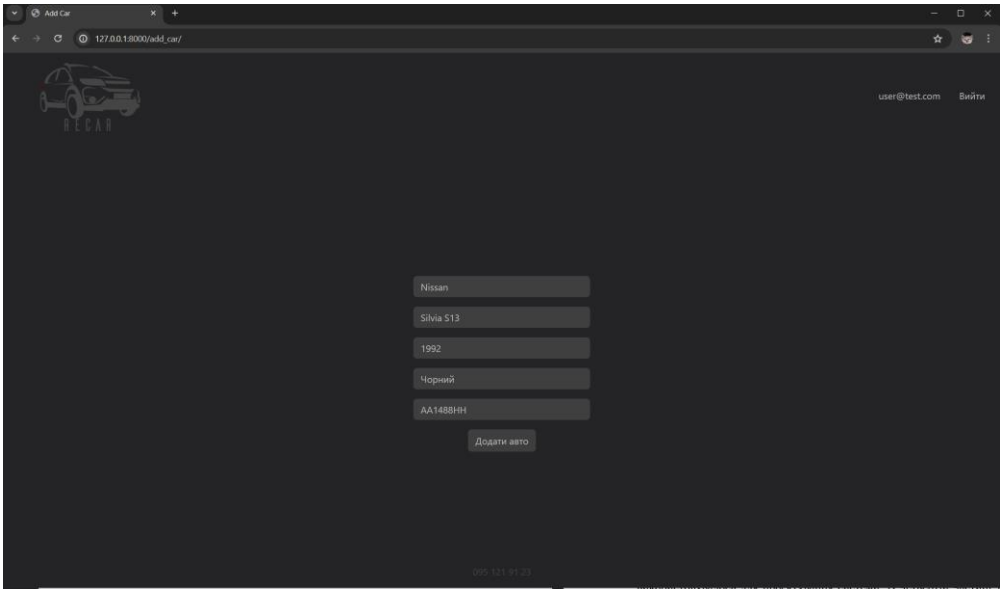


Рисунок 3.27 – Створення авто в системі

Джерело: розроблено автором самостійно

Сторінка реєстрації нового транспорту (рисунок 3.16) містить поля введення марки транспортного засобу автомобіля, моделі, колір, номер, рік випуску автомобіля.

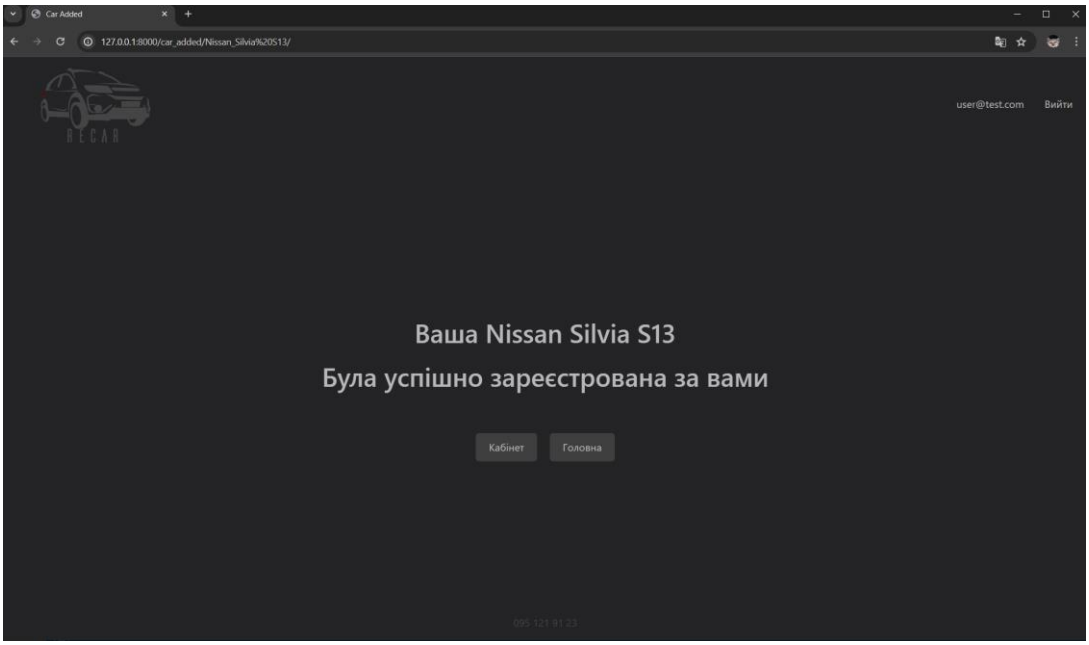


Рисунок 3.28 – Успішна реєстрація авто

Джерело: розроблено автором самостійно

Сторінка підтвердження реєстрації нового транспорту (рисунок 3.17) показує інформацію про успішно доданий автомобіль, присутні кнопки переходу на профіль чи на домашню сторінку.

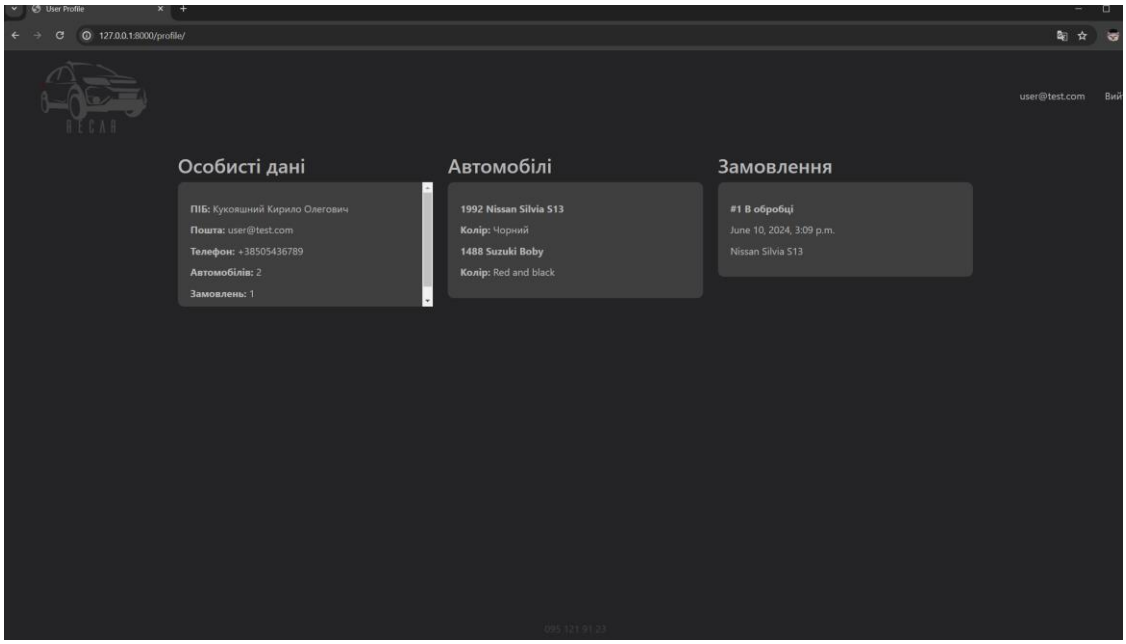


Рисунок 3.29 – Кабінет користувача

Джерело: розроблено автором самостійно

Сторінка особистого кабінету (рисунок 3.29) містить контактні дані користувача, список транспортних засобів (який при реєстрації є порожнім) та історія замовлень, вид кабінету відрізняється від прототипу, оскільки дизайн на прототипі не вписувався в дизайн веб-ресурсу.

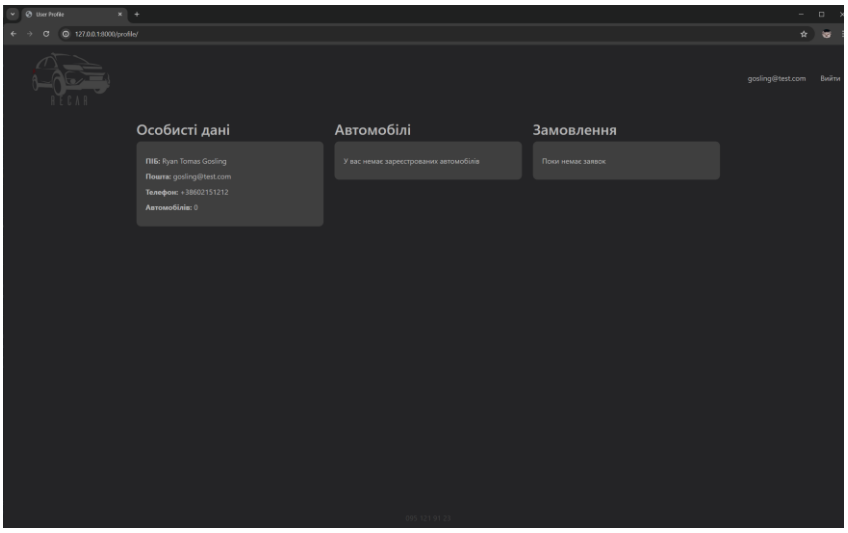


Рисунок 3.30 – Кабінет іншого користувача

Джерело: розроблено автором самостійно

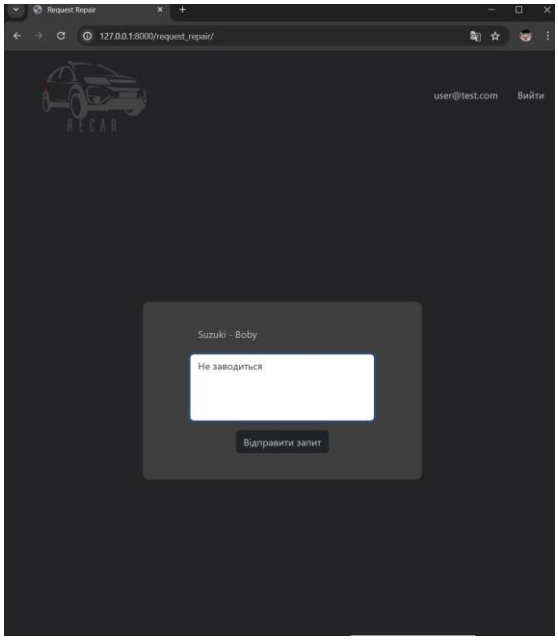


Рисунок 3.31 – Заповнення заявки

Джерело: розроблено автором самостійно

Заповнення заявки на ремонт (рисунок 3.31) дозволяє вибрати авто зі списку доступних, та опціонально написати додаткову інформацію про дефект.

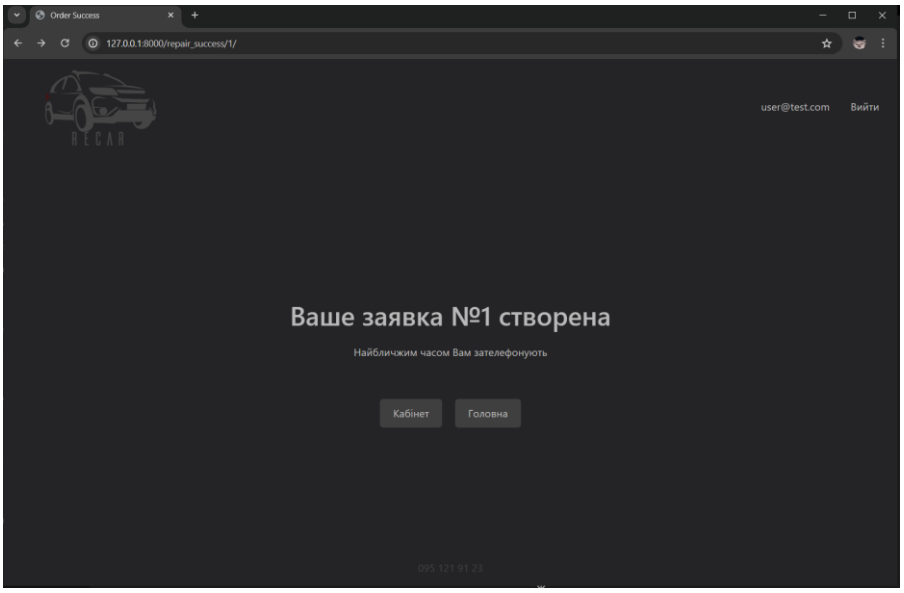


Рисунок 3.32 – Успішно створена заявка

Джерело: розроблено автором самостійно

Після створення заявки пишеться про те, що заявка створена та те, що скоро зателефонують (див рисунок 3.32).

В третьому розділі було описано вибір програмного забезпечення для реалізації проєктування компонентів або модулів системи та створення прототипу системи, на основі якого було реалізовано компоненти інформаційної системи.

ВИСНОВКИ

В ході роботи над кваліфікаційним бакалаврським проєктом на тему проєктування інформаційної системи для визначення потреби в комплектуючих для ремонту автомобілів, було проведено дослідження інформації, роботу з програмними засобами для побудови діаграм, таблиць масивів, схем, та створення прототипу системи.

В першому розділі проведено історичний огляд предметної області згідно тематики бакалаврського проєкту. За допомогою методології OPM, в середовищі OPCAT, було створено OPD діаграми верхнього рівня та нижніх рівнів, з метою описати загальний принцип функціонування інформаційної системи, було проведено аналіз існуючих систем та сформовано порівняльну таблицю, на основі яких були виокремлені найголовніші вимоги.

В другому розділі було встановлено вимоги до проєктування системи визначення потреби в комплектуючих для ремонту автомобілів, цими вимогами є: бізнес вимоги, функціональні вимоги, нефункціональні вимоги, що були сформовані згідно побажань клієнтів та адміністратора на основі порівняльної таблиці в першому розділі проєкту. Також, було створено допоміжні діаграми, з метою демонстрації кількості вимог, їх розподіл за статусом виконання, складності, та пріоритету. Було створено модель поведінки системи у вигляді Use-Case діаграми з прецедентами, на основі яких були побудовані сценарії та діаграми послідовності з метою чіткого відображення ходу подій, за допомогою діаграм класів та визначення блоків було проведено моделювання структури системи.

В третьому розділі було детально описано вибір забезпечення для реалізації проєктування компонентів системи та створення функціонального прототипу системи і в подальшому реалізації компонентів системи. Було створено діаграми планового технічного забезпечення системи, або підприємства на якому розміщуються модулі системи та база даних, як користувачі взаємодіють з цією системою використовуючи мережу інтернет. Надано загальну характеристику інформаційного забезпечення інформаційної

системи, та що входить в категорії, як організовується збір та передача інформації в системі та хто слугує її джерелом. Створено систему кодування для автомобілів, користувачів та замовлень, наведено структуру інформаційних масивів інформації для використання в інформаційній системі. Приведено причини вибору СКБД MySQL, та за допомогою графічної оболонки побудовано даталогічну модель бази даних на основі інформаційних масивів.

В другій частині третього розділу було описано можливе технічне забезпечення користувача та оптимальне забезпечення для інформаційної системи. В третій частині третього розділу перелічено системне програмне забезпечення, прикладне та спеціалізоване програмне забезпечення що використовувалось для проєктування системи. В четвертій частині третього розділу продемонстровано функціональний прототип інформаційної системи в середовищі Figma та реалізовані компоненти системи.

Проєкт у певній мірі є подібним до двох існуючих систем що були взяті за основу, але спрощений для більш зручного використання користувачами. В якості покращення системи, пропонується приділити увагу штучному інтелекту, а саме розглядати технології комп’ютерного бачення розпізнавання та класифікації об’єктів, дізнатись про тренування моделей комп’ютерного бачення на дефектах авто які можна визначити візуально та які деталі є подібними до тих, що пошкоджені, з метою автоматичної ідентифікації дефективної зони та підбору комплектуючих для ремонту. Система орієнтована в першу чергу на цивільний ринок, але при деяких змінах, можлива її адаптація під військову сферу для обліку транспорту на ремонті та полегшення ведення звітності про ремонті роботи, закупку деталей та обслуговування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Сірий А. І. Т РІШЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ АВТОСЕРВІСУ [Електронний ресурс] / А. І. Сірий. – 2013. – Режим доступу до ресурсу: http://repository.hneu.edu.ua/bitstream/123456789/8937/1/Збірник наукових праць студентів спеціальностей 2013 - 1.pdf#page=130 (дата звернення 14.04.2024).

2. Шип Д. В. Програмне забезпечення для автоматизації роботи автосервісу [Електронний ресурс] / Д. В. Шип, Н. В. Швець. – 2023. – Режим доступу до ресурсу: https://card-file.ontu.edu.ua/items/0622e400-bdda-4802-a09e-d4a6493e4c34 (дата звернення 14.04.2024).

3. Устенко І. А. ОРГАНІЗАЦІЙНІ ПРОЦЕДУРИ МАЛОГО БІЗНЕСУ В АВТОСЕРВІС [Електронний ресурс] / І. А. Устенко. – 2018. – Режим доступу до ресурсу: https://conf.ztu.edu.ua/wp-content/uploads/2018/12/383.pdf (дата звернення 14.04.2024).

4. Ситник Н. В. Проектування баз і сховищ даних. Навчальний посібник [Електронний ресурс] / Н. В. Ситник. – 2004. – Режим доступу до ресурсу: //repository.hneu.edu.ua/bitstream/123456789/8937/1/Збірник наукових праць студенhttps://www.scribd.com/document/H-B-Ситник-Навчальний-посібник-pdf (дата звернення 14.04.2024).

5. Історія розвитку сфери автосервісу. MasterService [Електронний ресурс] – Режим доступу до ресурсу: https://master.shop/ (дата звернення 10.04.2024) (дата звернення 14.04.2024).

6. ISO 19450 [Електронний ресурс] – Режим доступу до ресурсу: https://www.iso.org/standard/84612.html (дата звернення 14.04.2024).

7. OPCAT [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/220842657_OPCAT_-_Object-process_case_tool_An_integrated_system_engineering_environment_ISEE (дата звернення 14.04.2024).

8. ATL [Електронний ресурс] – Режим доступу до ресурсу: https://atl.ua/ua/ (дата звернення 14.04.2024).

9. ElCars [Електронний ресурс] – Режим доступу до ресурсу: https://elcars.ua/ua/ (дата звернення 14.04.2024).

10. 1a-autoservice [Електронний ресурс] – Режим доступу до ресурсу: https://1a-autoservice.com.ua/ (дата звернення 14.04.2024).

11. Gepard [Електронний ресурс] – Режим доступу до ресурсу: https://gepard.org.ua/ua// (дата звернення 14.04.2024).

12. SysML [Електронний ресурс] – Режим доступу до ресурсу: https://sysml.org/ (дата звернення 17.04.2024).

13. Enterprise Architect [Електронний ресурс] – Режим доступу до ресурсу: https://sparxsystems.com/ (дата звернення 17.04.2024).

14. Dashboards [Електронний ресурс] – Режим доступу до ресурсу: https://sparxsystems.com/enterprise_architect_user_guide/14.0/guidebooks/tools_ea_dashboard_diagrams.html (дата звернення 17.04.2024).

15. Sequence diagrams [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram// (дата звернення 17.04.2024).

16. Class diagrams. Attribute types [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/ (дата звернення 17.04.2024).

17. Relationship matrix and trace diagrams in Enterprise Architect [Електронний ресурс] – Режим доступу до ресурсу: https://sparxsystems.com/enterprise_architect_user_guide/14.0/guidebooks/tools_ba_relationship_matrix.html (дата звернення 17.04.2024).

18. Інформаційне забезпечення інформаційних систем. Моделі і методи проектування [Електронний ресурс] – Режим доступу до ресурсу: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/260038/index.html (дата звернення 17.04.2024).

19. Про захист інформації в інформаційних системах [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/80/94-%D0%B2%D1%80 (дата звернення 25.04.2024).

20. Кодування економічної інформації [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/5118185/page:6/ (дата звернення 25.04.2024).

21. Oracle MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://dev.mysql.com/doc/ (дата звернення 25.04.2024).

22. Oracle corporation [Електронний ресурс] – Режим доступу до ресурсу: https://www.oracle.com/ua// (дата звернення 25.04.2024).

23. PHPmyAdmin [Електронний ресурс] – Режим доступу до ресурсу: https://www.phpmyadmin.net/(дата звернення 3.05.2024).

24. Lucidchart [Електронний ресурс] – Режим доступу до ресурсу: https://www.lucidchart.com/ (дата звернення 26.04.2024).

25. MySQL Workbench [Електронний ресурс] – Режим доступу до ресурсу: https://www.mysql.com/products/workbench// (дата звернення 3.05.2024).

26. Web service hosting. Static IP address [Електронний ресурс] – Режим доступу до ресурсу: https://ipstack.com/what-is-a-fixed-ip/ (дата звернення 3.05.2024).

27. Microsoft Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/ (дата звернення 3.05.2024).

28. Drawio (diagrams.net) [Електронний ресурс] – Режим доступу до ресурсу: https://app.diagrams.net/ (дата звернення 17.05.2024).

29. Python django [Електронний ресурс] – Режим доступу до ресурсу: https://www.djangoproject.com/start/overview/ (дата звернення 17.05.2024).

30. Figma [Електронний ресурс] – Режим доступу до ресурсу: https://www.figma.com/ (дата звернення 17.05.2024).

ДОДАТКИ

Додаток А

Скрипт діаграми OPD верхнього рівня

Клієнт is environmental and physical.
Клієнт власник> Авто.
Механік is environmental and physical.
Механік handles ІС визначення потреб у комплектуючих для ремонту авто.
Адміністратор is environmental and physical.
Адміністратор handles ІС визначення потреб у комплектуючих для ремонту авто.
Авто is physical.
ІС визначення потреб у комплектуючих для ремонту авто requires Авто.
ІС визначення потреб у комплектуючих для ремонту авто affects Клієнт.

Скрипт діаграми OPD верхнього рівня (Механізм збільшення)

Клієнт is environmental and physical.
Клієнт власник> Авто.
Клієнт handles Отримання авто, Передача авто на ремонт, and Реєстрація користувача.
Механік is environmental and physical.
Механік handles Передача авто на ремонт, Ремонт авто, and ІС визначення потреб у комплектуючих для ремонту авто.
Адміністратор is environmental and physical.
Адміністратор handles Оформлення заявки на ремонт авто, and ІС визначення потреб у комплектуючих для ремонту авто.
Авто is physical.
Передача авто на ремонт Ремонт авто.
Ремонт авто Отримання авто.
ІС визначення потреб у комплектуючих для ремонту авто requires Авто.
ІС визначення потреб у комплектуючих для ремонту авто affects Клієнт.
ІС визначення потреб у комплектуючих для ремонту авто zooms into Авторизація, Реєстрація користувача, Робота з особистим кабінетом, Оформлення заявки на ремонт авто, Додання авто.

Авторизація Робота з особистим кабінетом.
Реєстрація користувача Авторизація.
Робота з особистим кабінетом Додання авто.
Оформлення заявки на ремонт авто Передача авто на ремонт.
Додання авто Оформлення заявки на ремонт авто.

Скрипт діаграми OPD нижнього рівня. Процес авторизації

Клієнт is environmental and physical.
Клієнт handles Введення логіну та паролю.
Дані про користувача consists of Логін and Пароль.
Авторизація affects Клієнт.
Авторизація zooms into Введення логіну та паролю and Надання прав доступу до системи згідно типу облікового запису.

Введення логіну та паролю Надання прав доступу до системи згідно типу облікового запису.

Введення логіну та паролю requires Дані про користувача.

Скрипт діаграми OPD нижнього рівня. Процес реєстрації

Клієнт is environmental and physical.
Клієнт handles Перехід на сайт.
Дані про користувача consists of ПІБ, Телефон, Пошта, Логін, and Пароль.
Реєстрація користувача requires Дані про користувача.
Реєстрація користувача zooms into Перехід на сайт and Заповнення форми реєстрації.
Перехід на сайт Заповнення форми реєстрації.
Заповнення форми реєстрації consumes Дані про користувача.

Скрипт діаграми OPD нижнього рівня. Процес додання авто

Клієнт is environmental and physical.
Клієнт Власник Авто.
Клієнт handles Заповнення форми додання авто.
Авто is physical.
Дані про авто consists of Марка авто, Модель авто, Номер авто, and Колір авто.
Дані про авто Стосується Авто.
Додання авто requires Авто.
Додання авто affects Клієнт.
Додання авто zooms into Додання нового авто в особистий кабінет and Заповнення форми додання авто.
Заповнення форми додання авто Додання нового авто в особистий кабінет.
Заповнення форми додання авто consumes Дані про авто.

Скрипт діаграми OPD нижнього рівня. Процес роботи з особистим кабінетом

Клієнт is environmental and physical.
Клієнт handles Зміна даних користувача.
Дані про користувача consists of ПІБ, Телефон, Пошта, Логін, and Пароль.
Робота з особистим кабінетом affects Клієнт.
Робота з особистим кабінетом zooms into Зміна даних користувача.
Зміна даних користувача requires Дані про користувача.

Скрипт діаграми OPD нижнього рівня. Процес оформлення заявки на ремонт

Клієнт is environmental and physical.
Клієнт Власник Авто.
Клієнт handles Заповнення форми на ремонт.
Адміністратор is environmental and physical.
Адміністратор handles Оформлення заявки на ремонт авто.
Авто is physical.
Оформлення заявки на ремонт авто requires Авто.
Оформлення заявки на ремонт авто affects Клієнт.
Оформлення заявки на ремонт авто zooms into Уточнення даних про ремонт and Заповнення форми на ремонт.
Уточнення даних про ремонт affects Адміністратор.
Заповнення форми на ремонт invokes Уточнення даних про ремонт.

Додаток Б

Код програми

Models.py

```
from django.db import models
from django.utils.timezone import now
# Create your models here.
from django.contrib.auth.models import AbstractBaseUser
class User(AbstractBaseUser):
    user_id = models.AutoField(primary_key=True, unique=True)
    user_mail = models.CharField(max_length=255, unique=True)
    user_phone = models.BigIntegerField()
    user_type = models.BooleanField(default=False)
    user_info = models.CharField(max_length=255)
    user_pwd = models.CharField(max_length=255)
    REQUIRED_FIELDS = ['user_phone', 'user_info', 'user_pwd']
    USERNAME_FIELD='user_mail'
    def __str__(self):
        return self.user_mail

class Car(models.Model):
    car_id = models.AutoField(primary_key=True)
    car_color = models.CharField(max_length=45)
    car_name = models.CharField(max_length=45)
    car_model = models.CharField(max_length=45)
    car_year = models.IntegerField()
    car_plate = models.CharField(max_length=10)
    car_image = models.ImageField(upload_to='media/user_car_images/')
    user = models.ForeignKey(User,on_delete=models.CASCADE, blank=False)

class Order(models.Model):
    order_id = models.AutoField(primary_key=True)
    order_date = models.DateTimeField(default=now, editable=False)
    order_exp_date = models.DateField(default=now)
    order_problem = models.CharField(max_length=255, default='Не вказано')
    order_status = models.CharField(max_length=45, default='В обробці')
    order_extra_info = models.TextField(max_length=512, blank=True, null=True)
    order_parts = models.JSONField(default='Не вказано')
    order_photo = models.ImageField(upload_to='media/order_car_images/', blank=True,
null=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE, blank=False)
    car = models.ForeignKey(Car, on_delete=models.CASCADE, blank=False)
```

forms.py

```
from django import forms
from .models import User, Car, Order

class LoginForm(forms.Form):
```

```
    user_mail = forms.EmailField()
    password = forms.CharField(widget=forms.PasswordInput)

class RegistrationForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput)
    confpwd = forms.CharField(widget=forms.PasswordInput, label='Підтвердження паролю')

    class Meta:
        model = User
        fields = ['user_mail', 'user_phone', 'user_info', 'password']

    def clean(self):
        cleaned_data = super().clean()
        password = cleaned_data.get('password')
        confpwd = cleaned_data.get('confpwd')

        if password and confpwd and password != confpwd:
            self.add_error('confpwd', 'Паролі не співпадають')

class CarForm(forms.ModelForm):
    class Meta:
        model = Car
        fields = ['car_name', 'car_model', 'car_year', 'car_color', 'car_plate']

class RepairRequestForm(forms.ModelForm):
    class Meta:
        model = Order
        fields = ['car', 'order_extra_info']
        widgets = {
            'order_extra_info': forms.Textarea(attrs={'placeholder': 'Опишіть проблему'}),
        }
```

Views.py

```
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from .forms import LoginForm, RegistrationForm, CarForm, RepairRequestForm
from .models import User, Car, Order
from django.urls import reverse
from django.utils.timezone import now

def inactive_landing(request):
    return render(request, 'inactive_landing.html')

def login_view(request):
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            user_mail = form.cleaned_data.get('user_mail')
            password = form.cleaned_data.get('password')
            try:
                user = User.objects.get(user_mail=user_mail, password=password)
                request.session['user_id'] = user.user_id
                request.session['user_mail'] = user.user_mail
```

```
            return redirect('active_landing')
        except User.DoesNotExist:
            form.add_error(None, 'Неправильні дані для входу')
    else:
        form = LoginForm()
    return render(request, 'login.html', {'form': form})

def register_view(request):
    if request.method == 'POST':
        form = RegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
    else:
        form = RegistrationForm()
    return render(request, 'register.html', {'form': form})

def active_landing(request):
    if 'user_id' not in request.session:
        return redirect('login')
    user_id = request.session['user_id']
    try:
        user = User.objects.get(user_id=user_id)
        user_mail = user.user_mail
        return render(request, 'active_landing.html', {'user': user})
    except User.DoesNotExist:
        return redirect('login')

def profile_view(request):
    if 'user_id' not in request.session:
        return redirect('login')
    user_id = request.session['user_id']
    try:
        user = User.objects.get(user_id=user_id)
        cars = Car.objects.filter(user=user)
        orders = Order.objects.filter(user=user).select_related('car')
        car_counter = cars.count()
        order_counter = orders.count()
        return render(request, 'profile.html', {'user': user, 'cars': cars, 'orders': orders, 'car_count':
car_counter, 'order_count': order_counter})
    except User.DoesNotExist:
        return redirect('login')

def add_car_view(request):
    if 'user_id' not in request.session:
        return redirect('login')
    user_id = request.session['user_id']
    #user_mail = request.session['user_mail']
    user = User.objects.get(user_id=user_id)

    if request.method == 'POST':
        form = CarForm(request.POST, request.FILES)
        if form.is_valid():
            car = form.save(commit=False)
```

```
            car.user = user
            car.save()
            return redirect(reverse('car_added', kwargs={'car_name': car.car_name, 'car_model': car.car_model}))
    else:
        form = CarForm()

    return render(request, 'add_car.html', {'form': form, 'user': user})

def car_added_view(request, car_name, car_model):
    if 'user_id' not in request.session:
        return redirect('login')
    user_id = request.session['user_id']
    user = User.objects.get(user_id=user_id)
    return render(request, 'car_added.html', {'car_name': car_name, 'car_model': car_model, 'user': user})

def request_repair_view(request):
    if 'user_id' not in request.session:
        return redirect('login')
    user_id = request.session['user_id']
    user = User.objects.get(user_id=user_id)

    if request.method == 'POST':
        form = RepairRequestForm(request.POST)
        if form.is_valid():
            order = form.save(commit=False)
            order.user = user
            if not order.order_exp_date:
                order.order_exp_date = now().date()
            if not order.order_status:
                order.order_status = 'В обробці'
            order.save()
            return redirect(reverse('repair_success', kwargs={'order_id': order.order_id}))
    else:
        form = RepairRequestForm()
        form.fields['car'].queryset = Car.objects.filter(user=user)

    return render(request, 'request_repair.html', {'form': form, 'user': user})

def repair_success_view(request, order_id):
    if 'user_id' not in request.session:
        return redirect('login')
    user_id = request.session['user_id']
    user = User.objects.get(user_id=user_id)
    return render(request, 'order_success.html', {'order_id': order_id, 'user': user})

def logout_view(request):
    logout(request)
    return redirect(reverse('inactive_landing'))

def repair_success_view(request, order_id):
    if 'user_id' not in request.session:
        return redirect('login')
```

```
    user_id = request.session['user_id']
    user = User.objects.get(user_id=user_id)
    return render(request, 'order_success.html', {'order_id': order_id, 'user': user})

def logout_view(request):
    logout(request)
    return redirect(reverse('inactive_landing'))
```

register.html

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
    <title>Register</title>
    <script src="{% static 'recar/js/bootstrap.js' %}"></script>
    <link rel="stylesheet" href="{% static 'recar/css/bootstrap.css' %}">
    <style>
      html, body {
          height: 100%;
          margin: 0;
      }
      body {
          display: flex;
          flex-direction: column;
      }
      .container {
          flex: 1;
          display: flex;
          justify-content: center;
          align-items: center;
      }
      footer {
        color: #B9B9B9;
      }
      p {
        color: #B9B9B9;
      }
      .form-signin {
        width: 100%;
        max-width: 330px;
        padding: 15px;
        margin: auto;
        display: flex;
        flex-direction: column;
        align-items: center;
      }
      .form-control {
        max-width: 300px;
        margin: auto;
        background-color: #3F3F3F;
        color: #B9B9B9;
      }
```

```
    .navbar-brand {
      display: flex;
      align-items: center;
      margin-left: 50px;
    }
  </style>
</head>
<body>
  <nav class="navbar navbar">
    <div class="container-fluid">
      <a class="navbar-brand" href="http://127.0.0.1:8000">
        <img src="{% static 'recar/images/logo_n.png' %}" alt="" width="173" height="auto" class="d-inline-block"></a>
    </div>
  </nav>
  <div class="container">
    <form class="form-signin" method="post">
      {% csrf_token %}

      <input id="inputEmail" class="form-control mb-3 border-0" type="email" placeholder="Пошта" name="user_mail">

      <input id="inputPhone" class="form-control mb-3 border-0" type="text" placeholder="(0xx)xxxxxxx" name="user_phone" maxlength="10">

      <input id="inputInfo" class="form-control mb-3 border-0" type="text" placeholder="Ваш ПІБ" name="user_info">

      <input id="inputPassword" class="form-control mb-3 border-0" type="password" placeholder="Пароль" name="password">

      <input id="inputPasswordConfirm" class="form-control mb-3 border-0" type="password" placeholder="Підтвердіть пароль" name="confpwd">

      <button type="submit" class="btn btn-dark mb-2" style="color: #B9B9B9;">Зареєструватись</button>
    </form>
  </div>
  <footer class="mastfoot mt-auto">
    <div class="inner">
      <p class="text-dark text-center">095 121 91 23</p>
    </div>
  </footer>
</body>
</html>
```

Login.html

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
  <title>Login</title>
  <script src="{% static 'recar/js/bootstrap.js' %}"></script>
```

```
    <link rel="stylesheet" href="{% static 'recar/css/bootstrap.css' %}">
    <style>
      html, body {
          height: 100%;
          margin: 0;
      }
      body {
          display: flex;
          flex-direction: column;
      }
      .cover-container {
          flex: 1;
          display: flex;
          flex-direction: column;
          justify-content: center;
          margin-top: 50px;
      }
      .container {
          flex: 1;
          display: flex;
          justify-content: center;
          align-items: center;
      }
      footer {
        color: #B9B9B9;
      }
      p {
        color: #B9B9B9;
      }
      .form-signin {
        width: 100%;
        max-width: 330px;
        padding: 15px;
        margin: auto;
      }
      .form-control {
        max-width: 300px;
        margin: auto;
        background-color: #3F3F3F;
      }
      .navbar-brand {
        display: flex;
        align-items: center;
        margin-left: 50px;
      }
    </style>
  </head>
  <body class="text-center">

    <nav class="navbar navbar">
        <div class="container-fluid">
          <a class="navbar-brand" href="http://127.0.0.1:8000">
            <img src="{% static 'recar/images/logo_n.png' %}" alt="" width="173" height="auto"
  class="d-inline-block"></a>
```

```
      </div>
    </nav>

    <div class="container">
      <div class="alert alert-danger mt-3 d-none" role="alert">
        Неправильний пароль чи пошта.
       </div>
      <form class="form-signin" method="post">
        {% csrf_token %}
        <input id="inputEmail" class="form-control mb-3 border-0" type="email" placeholder="Пошта" name="user_mail" style="background-color: #3F3F3F; color: #B9B9B9;">
        <input id="inputPassword" class="form-control mb-3 border-0" type="password" placeholder="Пароль" name="password" style="background-color: #3F3F3F; color: #B9B9B9;">
        <button type="submit" class="btn btn-dark mb-2" style="color: #B9B9B9;">Увійти</button>
        <br>
        <a class="btn btn-dark" href="{% url 'register' %}" role="button" style="color: #616161;">Реєстрація</a>
      </form>
    </div>
    <footer class="mastfoot mt-auto">
      <div class="inner">
        <p class="text-dark text-center">095 121 91 23</p>
      </div>
    </footer>
</body>
</html>
```

Active_landing.html

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
    <title>Recar</title>
    <script src="{% static 'recar/js/bootstrap.js' %}"></script>
    <link rel="stylesheet" href="{% static 'recar/css/bootstrap.css' %}">
    <style>
     html, body {
        height: 100%;
        margin: 0;
     }
     body {
        display: flex;
        flex-direction: column;
     }
     .cover-container {
        flex: 1;
        display: flex;
        flex-direction: column;
        justify-content: center;
     }
```

```
    footer {
       color: #B9B9B9;
    }
    p{
      color: #B9B9B9;
    }
    .navbar-brand {
      display: flex;
      align-items: center;
      margin-left: 50px;
    }

  </style>
</head>
<body>
   <nav class="navbar navbar">
      <div class="container-fluid">
        <a class="navbar-brand" href="{% url 'active_landing' %}">
          <img src="{% static 'recar/images/logo_n.png' %}" alt="" width="173" height="auto"
class="d-inline-block"></a>
          <ul class="nav justify-content-end">
            <li class="nav-item">
            <a class="nav-link active" href="{% url 'profile' %}">{{ user.user_mail }}</a>
            </li>
            <li class="nav-item">
              <a class="nav-link active" href="{% url 'logout' %}">Вийти</a>
            </li>
          </ul>
        </div>
   </nav>

        <div class="cover-container container-fluid flex-grow-1 d-flex flex-column">
          <div class="row flex-grow-1 align-items-center">
            <div class="col-1"></div>
            <div class="col-2">
              <div class="p-3">
                <a href="{% url 'profile' %}">
                  <img src="{% static 'recar/images/cab_a.png' %}" alt="" width="290"
height="auto">
                </a>
              </div>
            </div>
            <div class="col-2">
              <div class="p-3">
                <a href="{% url 'add_car' %}">
                  <img src="{% static 'recar/images/carnew_a.png' %}" alt="" width="290"
height="auto">
                </a>
              </div>
            </div>
            <div class="col-2">
              <div class="p-3">
                <a href="{% url 'request_repair' %}">
```

```
                <img src="{% static 'recar/images/rep_a.png' %}" alt="" width="290" height="auto">
              </a>
            </div>
          </div>
          <div class="col-2">
            <div class="p-3">
              <img src="{% static 'recar/images/check_a.png' %}" alt="" width="290" height="auto">
            </div>
          </div>
          <div class="col-2">
            <div class="p-3">
              <a href="https://www.google.com/maps/@-35.6875626,139.3412607,3a,43y,74.95h,72.87t/data=!3m6!1e1!3m4!1s1UlqG6LfO5I1nmQyaWS-5Q!2e0!7i16384!8i8192?coh=205409&entry=ttu" target="_blank">
                <img src="{% static 'recar/images/map_a.png' %}" alt="Google Maps" width="290" height="auto">
              </a>
            </div>
          </div>
          <div class="col-1"></div>
        </div>
      </div>
      <footer class="mastfoot mt-auto">
        <div class="inner">
          <p class="text-dark text-center">095 121 91 23</p>
        </div>
      </footer>
    </body>
    </html>
```

Add_car.html

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
  <title>Add Car</title>
  <script src="{% static 'recar/js/bootstrap.js' %}"></script>
  <link rel="stylesheet" href="{% static 'recar/css/bootstrap.css' %}">
  <style>
    html, body {
      height: 100%;
      margin: 0;
    }
    body {
      display: flex;
      flex-direction: column;
    }
    .cover-container {
      flex: 1;
      display: flex;
      flex-direction: column;
```

```
        justify-content: center;
        margin-top: 50px;
      }
      footer {
        color: #B9B9B9;
      }
      p {
        color: #B9B9B9;
      }
      .navbar-brand {
        display: flex;
        align-items: center;
        margin-left: 50px;
      }
      .form-container {
        padding: 20px;
        background-color: #242426;
        border-radius: 10px;
        color: #B9B9B9;
      }
      .form-signin {
        width: 100%;
        max-width: 330px;
        padding: 15px;
        margin: auto;
        display: flex;
        flex-direction: column;
        align-items: center;
      }
      .form-control {
        max-width: 300px;
        margin: auto;
        background-color: #3F3F3F;
        color: #B9B9B9;
      }
      #inputCarYear::-webkit-outer-spin-button,
      #inputCarYear::-webkit-inner-spin-button {
      opacity: 0;
      pointer-events: none;
    }

  </style>
</head>
<body>
  <nav class="navbar navbar">
    <div class="container-fluid">
      <a class="navbar-brand" href="{% url 'active_landing' %}">
        <img src="{% static 'recar/images/logo_n.png' %}" alt="" width="173" height="auto"
class="d-inline-block"></a>
          <ul class="nav justify-content-end">
            <li class="nav-item">
            <a class="nav-link active" href="{% url 'profile' %}">{{ user.user_mail }}</a>
            </li>
            <li class="nav-item">
```

```
            <a class="nav-link active" href="{% url 'logout' %}">Вийти</a>
          </li>
        </ul>
      </div>
</nav>

   <div class="container cover-container">
      <div class="row">
         <div class="col-md-2"></div>
         <div class="col-md-8">
            <div class="form-container">
               <form class="form-signin" method="POST" enctype="multipart/form-data">
                  {% csrf_token %}

                  <input id="inputCarName" class="form-control mb-3 border-0" type="text" placeholder="Марка" name="car_name">

                  <input id="inputCarModel" class="form-control mb-3 border-0" type="text" placeholder="Модель" name="car_model">

                  <input id="inputCarYear" class="form-control mb-3 border-0" type="number" placeholder="Рік випуску" name="car_year" maxlength="4" onwheel="preventScroll(event)">

                  <input id="inputCarColor" class="form-control mb-3 border-0" type="text" placeholder="Колір" name="car_color">

                  <input id="inputCarPlate" class="form-control mb-3 border-0" type="text" placeholder="Номерний знак" name="car_plate">

                  <button type="submit" class="btn btn-dark mb-2" style="color: #B9B9B9;">Додати авто</button>
               </form>
            </div>
         </div>
         <div class="col-md-2"></div>
      </div>
   </div>

   <footer class="mastfoot mt-auto">
      <div class="inner">
         <p class="text-dark text-center">095 121 91 23</p>
      </div>
   </footer>
   <script>
      function preventScroll(event) {
        event.preventDefault();
      }
      </script>
</body>
</html>
```

Profile.html

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
    <title>User Profile</title>
    <script src="{% static 'recar/js/bootstrap.js' %}"></script>
    <link rel="stylesheet" href="{% static 'recar/css/bootstrap.css' %}">
    <style>
      html, body {
          height: 100%;
          margin: 0;
      }
      body {
          display: flex;
          flex-direction: column;
          background-color: #242426;
          color: #B9B9B9;
      }
      .cover-container {
          flex: 1;
          display: flex;
          flex-direction: column;
          justify-content: center;
      }
      footer {
          color: #B9B9B9;
      }
      p {
          color: #B9B9B9;
      }
      .navbar-brand {
          display: flex;
          align-items: center;
          margin-left: 50px;
      }
      .container {
          padding: 20px;
      }
      .info-box {
        background-color: #3F3F3F;
        padding: 20px;
        border-radius: 10px;
        color: #B9B9B9;
        margin-bottom: 20px;
        max-height: 200px;
        overflow-y: auto;
      }
      .info-box h2, .info-box p {
          margin: 10px 0;
      }
      .info-box a {
          color: #B9B9B9;
          text-decoration: none;
      }
```

```
    .info-box a:hover {
      text-decoration: underline;
    }
    .col-md-4 {
     display: flex;
     flex-direction: column;
    }
    .user-info-box {
     max-height: none;
    }
    .car-info-box, .order-info-box {
     max-height: 200px;
     overflow-y: auto;
    }
   </style>
</head>
<body>
   <nav class="navbar navbar">
     <div class="container-fluid">
      <a class="navbar-brand" href="{% url 'active_landing' %}">
       <img src="{% static 'recar/images/logo_n.png' %}" alt="" width="173" height="auto"
class="d-inline-block"></a>
        <ul class="nav justify-content-end">
          <li class="nav-item">
          <a class="nav-link active" href="{% url 'profile' %}">{{ user.user_mail }}</a>
          </li>
          <li class="nav-item">
           <a class="nav-link active" href="{% url 'logout' %}">Вийти</a>
          </li>
        </ul>
       </div>
   </nav>
   <div class="container">
     <div class="row">
       <div class="col-md-4">
         <h2>Особисті дані</h2>
         <div class="info-box user-info-box">
           <p><strong>ПІБ:</strong> {{ user.user_info }}</p>
           <p><strong>Пошта:</strong> {{ user.user_mail }}</p>
           <p><strong>Телефон:</strong> +38{{ user.user_phone }}</p>
           <p><strong>Автомобілів:</strong> {{ car_count }}</p>
           <p><strong>Замовлень:</strong> {{ order_count }}</p>
         </div>
       </div>
       <div class="col-md-4">
         <h2>Автомобілі</h2>
         <div class="info-box car-info-box">
           {% if cars %}
             {% for car in cars %}
               <p><strong>{{ car.car_year }} {{ car.car_name }} {{ car.car_model
}}</strong></p>
               <p><strong>Колір:</strong> {{ car.car_color }}</p>
             {% endfor %}
           {% else %}
```

```
                <p>У вас немає зареєстрованих автомобілів</p>
            {% endif %}
          </div>
        </div>
        <div class="col-md-4">
          <h2>Замовлення</h2>
          <div class="info-box">
            {% if orders %}
              {% for order in orders %}
                <p><strong>#{{ order.order_id }} {{ order.order_status}}</strong></p>
                <p>{{ order.order_date }}</p>
                <p>{{ order.car.car_name }} {{ order.car.car_model }}</p>
              {% endfor %}
            {% else %}
              <p>Поки немає заявок</p>
            {% endif %}
          </div>
        </div>
      </div>
    </div>

    <footer class="mastfoot mt-auto">
      <div class="inner">
        <p class="text-dark text-center">095 121 91 23</p>
      </div>
    </footer>
  </body>
</html>
```