КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 – Комп'ютерні науки,
освітньо-наукова програма «Управління проєктами»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА
на тему:

«Дослідження методів управління проєктом розробки програмного забезпечення для процесів комерційної доставки вантажів»

Студента 2-го курсу групи УП-22

Олега ШЛАПАКА
(ім'я, прізвище)

Науковий керівник:

к.т.н., доцент
(науковий ступінь, вчене звання)

Вадим ЗЮЗЮН
(ім'я, прізвище)

_____ _____ _____
(підпис студента) *(дапа)* *(підпис)*

Попередній захист:

_____
(Висновок: "До захисту в Екзаменаційній комісії")

Завідувач кафедри
технологій управління_____ Віктор МОРОЗОВ _____
(підпис) (прізвище, ініціали) (дата)

Київ 2024

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Освітній рівень Магістр

Спеціальність 122 Комп'ютерні науки

Освітньо-наукова програма Управління проєктами

ЗАТВЕРДЖУЮ

Завідувач кафедри

професор Віктор МОРОЗОВ

_____

«08» листопада 2023 року

З А В Д А Н Н Я

НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент: Шлапак Олег Русланович

Група: УП-22

1. Тема кваліфікаційної роботи:

«Дослідження методів управління проєктом розробки програмного забезпечення для процесів комерційної доставки вантажів»

Затверджена Протоколом №6 від 06.11.2023 року.

2. Строк подання студентом готової роботи – «06» 05 2024 р.

3. Цільова установка та вихідні дані до роботи: дослідження різних методів та інструментів для управління проєктом, їх використання у плануванні проєкту, управління ризиками та управління якістю; вихідними даними є складений план проєкту

4. Зміст роботи: Розробка концепції проєкту та аналіз ринку, аналіз предметної галузі, мета, цілі та продукт проєкту, SWOT-аналіз, розробка концепції проєкту. розробка програмного забезпечення проєкту, концептуальної та математичної моделі проєкту, організаційна структура команди проєкту, опис модулів програмного забезпечення та їх функціональність, структура бази даних ІТ-проєкту, розроблення концептуальної та фізичної моделей бази даних проєкту, гнучкі технології створення продукту та планування ІТ-проєкту, визначення вимог до проєкту, календарне планування та віхи проєкту, використання методології Scrum для управління проєктом, життєвий цикл проєкту та його етапи, технології управління ІТ-проєктом, управління ризиками проєкту, запобігання негативним наслідкам та контроль за прогресом, управління якістю проєкту та забезпечення високої якості продукту, аналіз розробленого продукту.

5. Перелік графічного матеріалу: Загальний вигляд концептуальної моделі, Організаційна структура проєкту, Модулі ПЗ та їх основні функції, Концептуальна модель Бази даних, Фізична модель бази даних, задачі та віхи проєкту, дизайн розробленої програми.

6. Календарний план виконання роботи

№ з/п	Назва частин роботи	Виконання роботи
1	Вивчення літературних джерел з предмету дослідження	12.12.23-19.12.23
2	Збір і вивчення матеріалів досліджуваного підприємства	20.12.23-25.12.23
3	Складання розгорнутого плану кваліфікаційної роботи	26.12.23-27.12.23
4	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін	22.01.24
5	Підготовка розділу 1 «Розробка концепції та аналіз ринку»	05.02.24-26.02.24

6	Підготовка розділу 2 «Розробка програмного забезпечення»	27.02.24-16.03.23
7	Підготовка розділу 3 «Гнучкі технології створення продукту»	17.03.24-06.04.24
8	Підготовка розділу 4 «Технології управління ІТ-проєктом»	07.04.24-17.04.24
9	Оформлення кваліфікаційної роботи	22.04.24-02.05.24
10	Передача кваліфікаційної роботи науковому керівникові	03.05.24
11	Передача кваліфікаційної роботи рецензенту для рецензування	06.05.24
12	Захист кваліфікаційної роботи	21.05.24-23.05.24

Дата видачі завдання «09» листопада 2023 р.

Керівник роботи доцент Вадим ЗЮЗЮН
(посада, ім'я, прізвище)

_____
(підпис)

Завдання прийняв до виконання студент групи УП-22

Олег ШЛАПАК
(ім'я, прізвище)

_____
(підпис)

АНОТАЦІЯ

кваліфікаційної роботи магістра на тему
«Дослідження методів управління проєктом розробки програмного забезпечення для процесів комерційної доставки вантажів»

Студент: Шлапак Олег Русланович
Науковий керівник: Зюзюн Вадим Ігорович
Рік захисту – 2024

Метою підготовки роботи є дослідження методів управління проєктом розробки програмного забезпечення для оптимізації процесів комерційної доставки вантажів

Ціль проєкту – створити продукт для підвищення продуктивності комерційної доставки вантажів.

Наукова новизна полягає у побудові концептуальної моделі розробки програмного забезпечення для процесів комерційної доставки вантажів, особливістю якої є оптимізація процесів визначення матеріальних та програмних ресурсів.

Кваліфікаційна робота складається з анотації, вступу, основної частини, яка включає чотири розділи, висновків, переліку використаних інформаційних джерел та додатків.

У першому розділі наведено аналіз предметної галузі, розглянуто мету, цілі та продукт проєкту, проведено SWOT-аналіз, розроблено концепцію проєкту.

У другому розділі розроблено концептуальну та математичну моделі проєкту, розроблено організаційну структуру команди проєкту, описано модулі програмного забезпечення та їх функціональність, Сформовано структуру бази даних ІТ-проєкту, розроблено концептуальну та фізичну моделей бази даних проєкту.

У третьому розділі визначено вимоги до проєкту, побудовано організаційну структуру команди проєкту, розроблено календарне планування та віхи проєкту, обґрунтовано вибір методології управління проєктом, описано використання методології Scrum для управління проєктом, описано життєвий цикл проєкту та його етапи.

У четвертому розділі описано управління ризиками проєкту, розроблено план запобігання негативним наслідкам та контролю за прогресом, описано управління якістю проєкту та забезпечення високої якості продукту, проведено аналіз розробленого продукту

Робота містить 94 сторінки без додатків, 14 рисунків та 6 таблиць.

Додатки складають 9 сторінок.

Ключові слова: управління проєктами, управління якістю, управління ризиками, програмне забезпечення, доставка вантажів, фінанси.

ЗМІСТ

ВСТУП

Актуальність дослідження. Дослідження методів управління проєктом розробки програмного забезпечення для процесів комерційної доставки вантажів є вкрай актуальним у сучасному світі, де споживачі вимагають швидкості, точності та надійності у доставці своїх товарів. В контексті глобалізації економіки та зростаючої конкуренції, ефективне управління проєктами стає критично важливим для підприємств, що займаються комерційною доставкою вантажів. Завдяки впровадженню сучасного програмного забезпечення та оптимізації управління проєктами можна забезпечити підвищення швидкості доставки, зменшення витрат та поліпшення якості обслуговування. Дослідження цієї теми дозволить ідентифікувати ключові проблеми та ризики, пов'язані з розробкою та впровадженням програмного забезпечення для комерційної доставки, та розробити стратегії їх вирішення. Враховуючи тенденції до цифровізації та автоматизації у сфері логістики, розробка ефективних методів управління проєктами стає запорукою успішного впровадження інноваційних рішень у сфері комерційної доставки. Розвиток та впровадження нових технологій управління проєктами відкривають шлях до підвищення ефективності та конкурентоспроможності підприємств на ринку логістики та доставки.

Кваліфікаційна робота присвячена дослідженню методів управління проєктом розробки програмного забезпечення, спрямованого на оптимізацію процесів комерційної доставки вантажів. Це дослідження враховує важливі аспекти, такі як забезпечення якості, виконання термінів, ефективне використання ресурсів та здатність відповідати зростаючим вимогам ринку.

У контексті цієї роботи було проаналізовано різноманітні методи управління проєктами розробки програмного забезпечення та визначено їх ефективність в контексті впровадження системи для оптимізації процесів комерційної доставки вантажів. Робота також спрямована на ідентифікацію ключових викликів та ризиків, пов'язаних з розробкою та впровадженням таких систем, та розробку рекомендацій щодо їх подолання.

Вивчення цієї теми має велике значення для підвищення конкурентоспроможності підприємств у сфері комерційної доставки вантажів, підвищення якості обслуговування клієнтів та забезпечення оптимального використання ресурсів.

Метою кваліфікаційної роботи є дослідження методів управління проєктом розробки програмного забезпечення для оптимізації процесів комерційної доставки вантажів, які дозволять підвищити якість та ефективність процесів доставки вантажів, забезпечуючи вчасне виконання завдань та відповідність вимогам ринку

Для досягнення мети потрібно виконати наступні задачі дослідження, а саме:

1. Ретельно проаналізувати сучасні методи управління проєктами розробки програмного забезпечення, зокрема Scrum, та їх вплив на ефективність процесів комерційної доставки вантажів.

2. Визначити стратегії та напрями діяльності проєкту, що базуються на SWOT-аналізі, з метою максимізації сильних сторін та мінімізації слабких, а також використання можливостей та запобігання загрозам.

3. Здійснити розробку концептуальної та математичної моделей управління проєктом розробки програмного забезпечення для процесів комерційної доставки вантажів, особливістю яких є оптимізація процесів визначення матеріальних та програмних ресурсів.

4. Здійснити планування модулів, бази даних та інших компонентів програмного забезпечення з метою забезпечення якості та гнучкості у розробці.

5. Розробити чітку організаційну структуру команди, створити систему календарного планування та моніторингу прогресу проєкту, включаючи контроль за ризиками, відхиленнями та якістю, для забезпечення своєчасного та успішного завершення проєкту.

6. Провести аналіз розробленого продукту з метою ідентифікації його переваг та недоліків для подальшого вдосконалення, відповідно до потреб та

очікувань зацікавлених сторін.

Об'єктом дослідження є процеси комерційної доставки вантажів, які включають у себе управління логістикою, забезпеченням якості, маршрутизацію, взаємодію з клієнтами та інші аспекти, що впливають на ефективність та конкурентоспроможність бізнесу.

Предметом дослідження є методи управління проєктами, спрямовані на розробку та впровадження програмного забезпечення для оптимізації процесів комерційної доставки вантажів. Це включає в себе вивчення різних підходів до управління проєктами, вибір оптимальних методів та інструментів для конкретного домену застосування.

Наукова новизна полягає у побудові концептуальної моделі розробки програмного забезпечення для процесів комерційної доставки вантажів, особливістю якої є оптимізація процесів визначення матеріальних та програмних ресурсів.5

Практичне значення отриманих результатів можуть бути використані практичною галуззю для покращення управління проєктами та впровадження програмного забезпечення для оптимізації процесів комерційної доставки вантажів. Вони нададуть підприємствам інструменти та рекомендації для підвищення ефективності, зменшення ризиків та підвищення конкурентоспроможності на ринку доставки вантажів.

РОЗДІЛ 1. РОЗРОБКА КОНЦЕПЦІЇ ПРОЄКТУ ТА АНАЛІЗ РИНКУ НА ЯКИЙ ОРІЄНТОВАНО ПРОДУКТ ПРОЄКТУ

1.1 Аналіз предметної галузі

В еру неймовірного темпу технологічних змін і глобалізації, менеджмент перевезення вантажів є важливим елементом, який впливає на успішність будь-якого бізнесу. Ефективні системи доставки вантажів можуть принести значні переваги, такі як зменшення витрат, підвищення продуктивності та покращення загального задоволення клієнтів. Однак впровадження таких систем вимагає не тільки технологічних знань, але й вміння ефективно керувати проєктами.

Зараз на ринку існує багато готових рішень для автоматизації процесів комерційної доставки вантажів. Однак відсутність даних про методи управління проєктами, за допомогою яких ці рішення були розроблені, робить неможливим зрозуміти їхній потенціал і обмеження [1-4]. Методи управління проєктами, що використовуються при розробці таких рішень, можуть сильно вплинути на функціональність, надійність та загальну якість кінцевого продукту.

Окрім того, хоча CRM-системи вважаються потенційно найкращим рішенням для оптимізації процесів комерційної доставки вантажів, вони не вирішують в полній мірі викликів, пов'язаних з управлінням проєктами. CRM-системи ефективні для управління відносинами з клієнтами, але мають обмеження, що стосуються управління проєктами та процесами, особливо у складних та динамічних областях, як-от доставка вантажів.

Під час мого досвіду на науково-дослідній практиці я мав можливість пізнати, що підприємства зазвичай вдаються до використання гнучких методологій управління проєктами. Це дозволяє оперативно реагувати на зміни у вимогах та необхідностях бізнесу, а також забезпечує високу адаптивність до зовнішнього середовища. Результати таких проєктів, як правило, позитивні, проте важливо наголосити, що для кожного конкретного

випадку варто обирати методи, оптимальні саме для його умов, вимог та особливостей [5-9].

Цікавим прикладом гнучкої методології є Scrum, який дедалі частіше використовується у проєктах розробки програмного забезпечення. На основі цього підходу, проєкт розробки програмного забезпечення для комерційної доставки вантажів може бути структурований наступним чином [26].

Планування спринту: На цьому етапі команда вирішує, які вимоги або «історії користувача» (user stories) будуть реалізовані протягом наступного спринту. Наприклад, основною задачею цього спринту може бути автоматизоване відстеження місцезнаходження вантажу.

Виконання спринту: Протягом двох-чотирьох тижнів, команда працює над виконанням визначеного обсягу роботи. Програмісти можуть розробити код для нового модуля відстеження, тестувальники перевіряють його на помилки та вичерпність, а документувальники готують інструкції для кінцевих користувачів.

Огляд спринту: По завершенню спринту команда демонструє готовий продукт або його частину і отримує відгук від зацікавлених сторін, які можуть включати представників бізнесу, клієнтів тощо.

Ретроспектива спринту: Останнім етапом є аналіз успіхів та невдач спринту. Це дозволяє команді вивчити уроки та впровадити покращення для наступного спринту.

Такий циклічний підхід дозволяє поступово реалізовувати вимоги до продукту, одночасно адаптуючись до виниклих змін та відгуків від зацікавлених сторін. Це робить процес розробки більш гнучким та результативним.

Kanban – це ще одна популярна гнучка методологія, яка орієнтована на візуалізацію робочого процесу та постійне поліпшення. Наступний сценарій демонструє можливе використання Kanban у проєкті розробки програмного забезпечення для комерційної доставки вантажів:

Визначення колонок: На початку команда визначає етапи робочого процесу, які будуть представлені колонками на дошці Kanban. Наприклад, «Запланована робота», «Розробка», «Тестування», «Готово до випуску», та «Випущено».

Наповнення дошки: Команда додає карточки з задачами до колонки «Запланована робота». Кожна задача може бути окремою функціональною вимогою до програмного забезпечення, наприклад «Додати можливість слідкувати за статусом вантажу в реальному часі».

Робота над задачами: Коли розробник готовий взятися за нову задачу, він переміщає одну з карточок з «Запланована робота» до колонки «Розробка». При цьому слід притримуватися обмеження кількості активних карточок (work in progress limit) для кожної колонки [10].

Переміщення карточок: Як тільки робота над задачею завершена, карточка переміщається до наступної колонки. Це дає команді змогу бачити поточний стан проєкту, ідентифікувати «вузькі місця» та постійно вдосконалювати процеси.

Kanban створює прозорий робочий процес, допомагає команді краще розуміти, як вони працюють, та відповідає на питання, що виконується нараз, на чому буде робота в майбутньому та що вже виконано.

Таким чином, необхідно подальше дослідження методів управління проєктами, які можуть бути використані при розробці програмного забезпечення ««для комерційних вантажних перевезень. Такі дослідження повинні охоплювати не тільки традиційні методи, які переважно використовувались раніше, але й більш гнучкі та адаптивні методи, які зарекомендували себе у сучасних IT-проєктах.

SKRUMBan – це комбінація двох методологій управління проєктами: Scrum і Kanban. Він поєднує в собі принципи і практики обох методологій з метою забезпечення більшої гнучкості та ефективності управління проєктами розробки програмного забезпечення.

Scrum – це ітеративний метод управління проєктами, який розглядає розробку програмного забезпечення як серію коротких ітераційних циклів, відомих як «спринти» [10-12].

Kanban – це метод управління виробництвом, який акцентується на візуальному представленні робочого процесу та обмеженні робочого навантаження на кожному етапі.

SKRUMBan поєднує ці дві методології, щоб забезпечити більшу гнучкість та адаптивність управління проєктом. Він дозволяє команді використовувати основні принципи Scrum, такі як ітеративність та планування спринтів, разом з візуальним управлінням завданнями і обмеженням робочого навантаження з Kanban.

Управління проєктом розробки програмного забезпечення для оптимізації процесів комерційної доставки вантажів може виграти від використання SKRUMBan у наступних аспектах:

- Гнучкість та адаптивність: SKRUMBan дозволяє команді швидко реагувати на зміни вимог та умовах ринку, оскільки він поєднує гнучкість Scrum з візуальним управлінням Kanban.
- Прозорість процесу: Використання візуальних дошок у Kanban дозволяє всій команді бачити поточний стан роботи та ідентифікувати можливі перешкоди або затримки.
- Постійне вдосконалення: SKRUMBan підтримує культуру постійного вдосконалення, оскільки він спонукає до інспекції та адаптації процесів під час кожного спринту або ітерації.
- Ефективне використання ресурсів: Обмеження робочого навантаження на кожному етапі за допомогою Kanban допомагає уникнути перевантаження та зберігає продуктивність команди.

Отже, SKRUMBan може бути корисним інструментом управління проєктом для розробки програмного забезпечення для оптимізації процесів комерційної доставки вантажів, дозволяючи забезпечити ефективну розробку продукту та вчасну доставку до клієнтів.

На підставі результатів дослідження, що були проведені, можна виокремити декілька можливих підходів до розробки проєкту.

Першим можливим підходом до управління проєктом розробки програмного забезпечення для процесів комерційної доставки вантажів є застосування гнучких методологій, таких як Agile, Scrum та Kanban. Спостереження показали, що ці методи дозволяють підприємствам швидко реагувати на зміни у вимогах, що є важливим у динамічному середовищі комерційної доставки вантажів. Agile дозволяє здійснювати розробку програмного забезпечення ітеративно, реагуючи на зміни вимог і відгуку клієнтів. Scrum надає фреймворк для ефективного керування командою розробників та планування робіт на короткі терміни [13, 26]. Крім того, Kanban забезпечує візуальне управління робочим процесом, дозволяючи керівникам проєктів максимально ефективно розподіляти ресурси та контролювати прогрес.

Другим підходом є використання CRM-систем для автоматизації процесів комерційної доставки вантажів. Ці системи надають можливість упорядкувати багатофакторний процес управління вантажопотоками, спростити планування доставок і покращити якість обслуговування клієнтів. Завдяки автоматизації та централізації даних, CRM-системи допомагають збільшити продуктивність та ефективність управління проєктами.

1.2 Мета, цілі та продукт проєкту

В сучасному світі електронної комерції та логістики, ефективність та точність процесів комерційної доставки вантажів стають критичними факторами для успішної діяльності підприємств. Забезпечення швидкого, надійного та оптимального перевезення вантажів до клієнтів є важливою складовою конкурентоспроможності на ринку. У цьому контексті розробка програмного забезпечення, спрямованого на оптимізацію процесів комерційної доставки вантажів, набуває великого значення.

Мета проєкту розробки програмного забезпечення для оптимізації процесів комерційної доставки вантажів полягає в створенні інноваційного рішення, яке спростить та покращить всі аспекти логістики вантажів. Цей проєкт ставить перед собою амбіційну мету підвищення ефективності та зручності управління процесами доставки вантажів, щоб забезпечити задоволення потреб клієнтів та підвищення конкурентоспроможності підприємств.

Головні завдання проєкту включають розробку інтегрованої системи управління логістикою вантажів, яка охоплює весь цикл перевезення від прийому замовлення до його виконання та доставки. Ця система повинна бути гнучкою, масштабованою та забезпечувати високу надійність у виконанні завдань. Крім того, важливим завданням є забезпечення інтеграції з існуючими системами управління та моніторингу транспортних засобів, щоб забезпечити стабільну та безперебійну роботу всіх компонентів логістичної системи.

Розробка програмного забезпечення для оптимізації процесів комерційної доставки вантажів вимагає використання передових технологій та інноваційних підходів. Впровадження аналізу даних, автоматизації процесів та використання Big Data може значно покращити ефективність та точність логістичних операцій.

Отже, *мета проєкту* розробки програмного забезпечення для оптимізації процесів комерційної доставки вантажів полягає в покращенні якості обслуговування клієнтів, зниженні витрат та підвищенні ефективності логістичних процесів. Цей проєкт має стратегічне значення для розвитку сучасної логістики та електронної комерції і спрямований на створення нових стандартів у галузі доставки вантажів.

Однією з ключових *цілей проєкту* є підвищення ефективності логістичних процесів. Це включає в себе автоматизацію та оптимізацію усіх етапів доставки, від прийому замовлення до фінальної доставки вантажу до клієнта. Розроблене програмне забезпечення має забезпечити швидке та

безперебійне виконання операцій, зменшення затрат часу та ресурсів, а також забезпечити оптимальне використання логістичних ресурсів.

Ще одна важлива ціль проєкту - покращення точності та надійності процесів доставки. Це означає впровадження системи, яка дозволяє в реальному часі відстежувати місцезнаходження вантажів, контролювати їхній стан та уникати можливих затримок чи втрат. Розроблене програмне забезпечення повинно забезпечити максимальну точність і надійність усіх операцій, що здійснюються в рамках логістичного процесу.

Також до важливих цілей можна віднести створення зручного та інтуїтивно зрозумілого інтерфейсу для клієнтів. Розроблене програмне забезпечення має забезпечити можливість легко здійснювати замовлення, відстежувати статус доставки та отримувати інформацію про вантажі без зайвих зусиль [16].

Останньою, але не менш важливою ціллю проєкту є зниження витрат та оптимізація використання ресурсів. Це означає зменшення непотрібних витрат на логістичні операції, оптимізацію маршрутів доставки, зменшення кількості пустих пробігів та збільшення загальної ефективності використання ресурсів.

Тож, підсумовуючи сказане, можна виділити наступні цілі:

1. Підвищення ефективності логістичних процесів
2. Покращення точності та надійності процесів
3. Забезпечення зручності для клієнтів
4. Зниження витрат та оптимізація ресурсів

Ці цілі відображають стратегічне значення проєкту розробки програмного забезпечення для оптимізації процесів комерційної доставки вантажів. Реалізація цих цілей дозволить підприємствам покращити якість обслуговування, знизити витрати та підвищити конкурентоспроможність на ринку.

Продуктом цього проєкту є інтегроване програмне забезпечення для управління процесами комерційної доставки вантажів. Це програмне забезпечення включатиме в себе модулі для моніторингу вантажів,

маршрутизації, управління запасами, відстеження вантажів, а також аналізу та звітності. Воно буде забезпечувати автоматизацію багатьох процесів, що допоможе знизити час доставки, оптимізувати витрати та підвищити рівень задоволеності клієнтів. Крім того, програмне забезпечення буде здатне адаптуватися до змінних умов ринку та потреб клієнтів, завдяки своїй гнучкій архітектурі та можливостім швидкого впровадження змін.

Для цього необхідно пройти через усі етапи розробки та впровадження ПЗ, що включають:

1. Аналіз: Провести аналіз поточних процесів комерційної доставки вантажів та ідентифікувати основні проблеми та області для оптимізації.
2. Дослідження: Дослідити сучасні технології та методи управління логістичними процесами, зокрема в галузі розробки програмного забезпечення для оптимізації доставки вантажів.
3. Розробка: Розробити програмне забезпечення, яке враховуватиме виявлені проблеми та відповідатиме вимогам та потребам підприємств у сфері комерційної доставки.
4. Тестування: Провести тестування розробленого програмного забезпечення для перевірки його ефективності, надійності та відповідності вимогам.
5. Впровадження: Впровадити розроблене програмне забезпечення на пілотних підприємствах та оцінити його реальні результати та вплив на бізнес-процеси.

Програмне забезпечення для оптимізації процесів комерційної доставки вантажів включає такі ключові функції:

1. Управління замовленнями: Система дозволяє ефективно керувати замовленнями, від прийому до виконання, забезпечуючи точність та швидкість обробки.

2. Маршрутизація та логістика: Програмне забезпечення допомагає визначити оптимальні маршрути доставки та керувати ресурсами для їх виконання.
3. Відстеження вантажів: Система надає можливість відстеження місцезнаходження вантажу в реальному часі, що забезпечує велику прозорість та надійність у процесі доставки.
4. Аналітика та звітність: Програмне забезпечення забезпечує аналітичні звіти та статистику про продуктивність та ефективність процесів доставки, що допомагає виявляти потенційні покращення та оптимізувати робочі процеси.

Наш продукт використовує передові технології, такі як штучний інтелект, аналіз даних та автоматизація процесів, що дозволяє покращити ефективність та точність логістичних операцій.

Програмне забезпечення для оптимізації процесів комерційної доставки вантажів є ключовим інструментом для підвищення конкурентоспроможності та ефективності підприємств у галузі логістики та електронної комерції. Наш продукт ставить перед собою завдання покращити якість обслуговування клієнтів, зменшити витрати та збільшити ефективність логістичних процесів, щоб задовольнити вимоги сучасного ринку.

1.3 SWOT-аналіз проєкту

Для успішного впровадження проєкту розробки програмного забезпечення для ефективних процесів доставки комерційних вантажів необхідно провести SWOT-аналіз, що охопить усі аспекти діяльності підприємства, враховуючи його унікальні характеристики порівняно з конкурентами. Перш ніж розглядати можливості та загрози, важливо визначити сильні та слабкі сторони підприємства. Мета SWOT-аналізу полягає у виявленні областей для покращення та використанні переваг для забезпечення ефективних комерційних процесів доставки вантажів [23]. Аналіз допоможе розробити стратегії для використання сильних сторін

проєкту та виділення його серед конкурентів, а також визначити шляхи до успіху у впровадженні програмного забезпечення для доставки вантажів.

Детальний SWOT-аналіз дозволить ідентифікувати сильні сторони проєкту, такі як інноваційні рішення у сфері програмного забезпечення, а також ефективне управління проєктом. Унікальні характеристики, такі як технологічна перевага чи стратегічне розташування, можуть стати ключовими конкурентними перевагами. Водночас, слабкі сторони, такі як недостатня фінансова стабільність або обмежені ресурси людських чи матеріальних ресурсів, потребують уваги та вирішення для успішної реалізації проєкту.

Зростання конкуренції може стимулювати інновації та покращення, але водночас потребує уважного вивчення ринкового середовища та розробки стратегій залучення та утримання клієнтів. Пошук нових постачальників може бути вирішено шляхом розширення мережі партнерів або укладання стратегічних угод, що забезпечать стабільність та надійність ланцюжка постачання.

Загальна мета SWOT-аналізу полягає у використанні цієї стратегічної інформації для розробки дорожньої карти успіху проєкту. Це включає в себе визначення пріоритетних напрямків розвитку, розробку конкурентоспроможних стратегій та виявлення ризиків, які потребують управління. У кінцевому результаті SWOT-аналіз допоможе забезпечити ефективну реалізацію програмного забезпечення для доставки вантажів та забезпечити конкурентні переваги на ринку.

Отож, за результатами проведеного SWOT-аналізу можна зробити наступні висновки:

1) Сильні сторони:
    a) Інноваційні рішення: Розробка програмного забезпечення забезпечить впровадження сучасних технологій і процесів у сфері доставки вантажів, що зробить компанію конкурентоспроможною на ринку.

b) Ефективне управління проєктом: Команда професіоналів з управління проєктом забезпечить вчасну реалізацію завдань та контроль над бюджетом, що сприятиме успішному завершенню проєкту.
c) Технічна експертиза: Наявність кваліфікованих і досвідчених інженерів і програмістів дозволить розробити високоякісне програмне забезпечення, яке відповідає потребам клієнтів та требованням ринку.
d) Широкі можливості розширення: Програмне забезпечення може бути легко адаптоване та розширене для впровадження в інші сфери логістики та транспорту, що відкриває нові ринки та можливості для росту компанії.
e) Підтримка користувачів: Забезпечення надійної підтримки та сервісу для клієнтів підвищить їх задоволеність та лояльність до продукту, що сприятиме збільшенню клієнтської бази та прибутковості проєкту.

2) Слабкі сторони:
a) Недостатня фінансова підтримка: Розробка програмного забезпечення може вимагати значних інвестицій у дослідження, розробку та тестування. Недостатність фінансування може призвести до обмежень у розвитку та якості продукту.
b) Обмежені ресурси людських та матеріальних ресурсів: Недостатність кваліфікованого персоналу або доступу до потрібних технічних ресурсів може уповільнити процес розробки та впровадження програмного забезпечення.
c) Невизначені вимоги замовника: Недостатня чіткість у вимогах до програмного забезпечення може призвести до непослідовності у розробці та незадоволення з боку замовника.
d) Технічні складнощі: Розробка програмного забезпечення для оптимізації процесів доставки може зустрічати технічні труднощі,

особливо у зв'язку з інтеграцією з існуючими системами або апаратним забезпеченням.

e) Зміна регулятивного середовища: Зміни у регулюванні в галузі логістики та доставки можуть вплинути на вимоги до програмного забезпечення та вимагати додаткових зусиль для адаптації продукту.

f) Конкурентний тиск: Швидко зростаюча конкуренція на ринку програмного забезпечення для логістики та доставки може створювати тиск на швидкість розробки та впровадження, що може вплинути на якість продукту.

3) Можливості:

a) Ринковий попит: Зростаючий обсяг е-комерції та онлайн-торгівлі створює значний попит на програмні рішення для оптимізації процесів доставки вантажів. Проєкт може ефективно задовольнити цей попит, пропонуючи розвинуті та інноваційні рішення.

b) Технологічні інновації: Використання передових технологій, таких як штучний інтелект, машинне навчання та Інтернет речей (IoT), в проєкті може дозволити створити продукт, що відповідає потребам сучасного ринку та забезпечить конкурентні переваги.

c) Глобальний розвиток: Можливість розширення проєкту на міжнародний рівень може відкрити нові ринки та збільшити потенційну базу клієнтів. Залучення міжнародних партнерів та впровадження міжнародних стандартів може підвищити конкурентоспроможність продукту.

d) Підвищення ефективності та зниження витрат: Програмне забезпечення може допомогти компаніям знизити витрати на доставку вантажів шляхом оптимізації маршрутів, керування запасами та підвищення ефективності роботи логістичних процесів.

e) Партнерські відносини: Укладення стратегічних партнерських угод з провідними компаніями у сфері логістики та транспорту може допомогти в розширенні функціональності та вдосконаленні продукту.

4) Загрози:
   a) Зростання конкуренції: Зі зростанням популярності ринку доставки вантажів може збільшуватися конкуренція з боку інших компаній, що пропонують схожі рішення. Це може призвести до зниження частки ринку або потребу в більших інвестиціях у маркетинг і розвиток продукту для збереження конкурентоспроможності.
   b) Технологічні загрози: Швидкий темп змін у технологічній сфері може призвести до виникнення нових технологій або рішень, які можуть конкурувати з розробленим програмним забезпеченням. Важливо забезпечити постійне оновлення та вдосконалення продукту, щоб залишатися на передовій технологічного ринку.
   c) Питання щодо безпеки даних: З поширенням цифрових загроз і кібератак може збільшуватися загроза для безпеки даних клієнтів та важливої інформації про вантажі. Недостатня захищеність може призвести до втрати довіри клієнтів і порушення законодавства щодо захисту персональних даних.
   d) Потенційні ризики у зв'язку зі змінами у законодавстві: Зміни у регулюючому середовищі, такі як нові закони про безпеку та регулювання перевезень, можуть вимагати значних змін у програмному забезпеченні. Неправильне врахування цих змін може призвести до штрафів або втрати репутації.

1.4. Постановка задачі проєкту

Концепція проєкту розробки програмного забезпечення для процесів комерційної доставки вантажів передбачає створення інноваційного та ефективного рішення, яке допоможе підприємствам у сфері логістики оптимізувати свої процеси та підвищити ефективність доставки вантажів. Основними цілями проєкту є автоматизація та оптимізація ключових процесів доставки, зменшення витрат та часу, підвищення точності та надійності логістичних операцій.

Це програмне забезпечення, спрямоване на оптимізацію та автоматизацію процесів комерційної доставки вантажів. Це інноваційний продукт, який дозволить компаніям, що займаються логістикою, ефективно управляти вантажними перевезеннями, зменшуючи витрати і підвищуючи продуктивність.

Цілі проєкту:

1. Підвищити ефективність та продуктивність процесів комерційної доставки вантажів.
2. Зменшити витрати на логістичні операції та перевезення.
3. Забезпечити точність та надійність відстеження вантажів для покращення обслуговування клієнтів.
4. Створити інтегровану та легко масштабовану платформу для управління вантажними перевезеннями.
5. Підвищити конкурентоспроможність логістичних компаній та вантажних перевізників на ринку.

Завдання проєкту:

1. Розробка та впровадження інтерфейсу для управління замовленнями та маршрутизацією доставок.
2. Інтеграція системи відстеження вантажів з використанням GPS-технологій та IoT-пристроїв.
3. Реалізація модульної архітектури програмного забезпечення для легкої розширюваності функціоналу.

4. Розробка засобів аналітики та звітності для відстеження ефективності перевезень та витрат.
5. Тестування та вдосконалення продукту з урахуванням фідбеку від користувачів та партнерів.
6. Проведення навчання та підтримка користувачів під час впровадження та використання платформи.
7. Забезпечення високого рівня безпеки та конфіденційності даних користувачів та їх вантажів.

Основні функції:

- Управління замовленнями: Платформа дозволить користувачам створювати, відстежувати та керувати замовленнями на доставку вантажів.
- Маршрутизація: CargoFlow буде використовувати алгоритми маршрутизації для оптимізації маршрутів доставки, що дозволить зменшити витрати на паливо та скоротити час доставки.
- Відстеження вантажу: Клієнти зможуть в реальному часі відстежувати місцезнаходження свого вантажу через GPS-технології.
- Аналітика та звітність: CargoFlow забезпечить користувачів зручними засобами аналізу даних та формування звітів щодо ефективності перевезень та витрат.
- Інтеграція з платіжними системами: Система буде підтримувати різні методи оплати, що забезпечить зручність для клієнтів і партнерів.
- Модульність: Платформа буде розроблена з урахуванням модульної архітектури, що дозволить легко розширювати функціонал в майбутньому.

Бізнес-переваги:

- Зниження витрат: Оптимізація маршрутів та управління вантажами дозволить зменшити витрати на паливо та інші операційні витрати.
- Підвищення продуктивності: Автоматизація процесів дозволить звільнити людські ресурси та підвищити продуктивність персоналу.

– Покращення обслуговування клієнтів: Реальний час відстеження вантажу та зручне керування замовленнями підвищить задоволення клієнтів.

Цільовою аудиторією є логістичні компанії, вантажні перевізники, виробники та роздрібні компанії, які здійснюють доставку вантажів.

Технічні вимоги:

– Веб-платформа: Розробка веб-додатку для доступу через браузер.
– Мобільні додатки: Розробка мобільних додатків для Android та iOS для зручного користування на смартфонах і планшетах.
– Використання сучасних технологій: Використання технологій хмарних обчислень, GPS-відстеження, аналізу даних тощо.

Цей проєкт має на меті створити інноваційний продукт, який спростить та покращить процеси комерційної доставки вантажів, забезпечуючи ефективне управління та оптимізацію витрат.

РОЗДІЛ 2. РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ ПРОЄКТУ

2.1 Розробка концептуальної та математичної моделі проєкту

Концептуальна модель проєкту – це абстрактне відображення основних складових та взаємозв'язків проєкту, яке відображає його структуру, функції та основні аспекти [14]. Ця модель надає загальне уявлення про те, яким чином проєкт буде організований та які елементи в ньому будуть задіяні. Концептуальна модель допомагає уточнити потреби та очікування учасників проєкту, а також допомагає визначити основні напрямки та цілі розробки. Це перший етап у процесі проєктування, коли визначається загальна концепція та стратегія його реалізації.

Тож концептуальна модель для проєкту розробки програмного забезпечення для процесів комерційної доставки вантажів (рис. 2.1) включає наступні основні компоненти [20-22]:

1) Співробітники:
    a) Менеджери: вони відповідають за прийняття та обробку замовлень, взаємодію з клієнтами та внутрішнім персоналом.
    b) Логісти: вони відповідають за планування та організацію маршрутів доставки, вибір транспортних засобів тощо.
    c) Технічна підтримка: забезпечують підтримку програмного забезпечення та обладнання, вирішують технічні проблеми.
2) Офіс:
    a) Програмне забезпечення: це центральний елемент, який обробляє та зберігає дані про замовлення, клієнтів, вантажі та маршрути доставки.
    b) Інформаційна система: включає в себе бази даних, системи відстеження вантажів, аналітичні засоби та інші додаткові модулі для ефективного управління процесами.

3) Обладнання:
    a) Комп'ютери та сервери: для зберігання та обробки даних.
    b) IoT-пристрої: для відстеження руху вантажів, моніторингу температури, вологості тощо.
    c) Комерційні транспортні засоби: для перевезення вантажів.

Рис 2.1. Загальний вигляд концептуальної моделі

Представляючи ресурси, необхідні для успішної розробки та впровадження програмного забезпечення, отримаємо:

$$R=\{R^M+R^H+R^P+R^I\}, \quad (2.1)$$

Де:

R^M – це сукупність матеріальних ресурсів, задіяних у проєкті,

R^H – це сукупність людських ресурсів, що беруть участь у проєкті,

R^P – це набір програмних ресурсів, що використовуються в проєкті,

R^I – це набір інформаційних ресурсів у проєкті.

Де:

$R^M=\{r_1^M,r_2^M,...,r_i^M\}$ – сукупність матеріальних ресурсів, де i – кількість видів матеріальних ресурсів, необхідних у проєкті;

$R^H=\{r_1^H,r_2^H,...,r_k^H\}$ – сукупність людських ресурсів, де k – кількість видів людських ресурсів, залучених до проєкту;

$R^P=\{r_1^P,r_2^P,...,r_S^P\}$ – набір програмних ресурсів, де s – кількість видів програмних ресурсів, задіяних у проєкті;

$R^I=\{r1^I,r_2^I,...,r_z^I\}$ – набір інформаційних ресурсів, де z – кількість видів інформаційних ресурсів, задіяних у проєкті.

Ця математична модель узагальнює ресурси, необхідні для успішної розробки та впровадження програмного забезпечення для процесів комерційної доставки вантажів.

Для цього проєкту найбільший вплив матимуть параметри R^H (людські ресурси) та R^P (програмні ресурси). Оптимізація людських ресурсів може бути досягнута шляхом відповідного планування та розподілу завдань між командою, врахуванням навичок та досвіду кожного учасника проєкту, а також забезпеченням належного мотиваційного середовища.

Щодо програмних ресурсів, їх можна оптимізувати шляхом вибору ефективних інструментів розробки, використанням автоматизованих процесів тестування та розгортання [14], а також підтримкою відкритих стандартів та гнучкої архітектури, яка дозволить легко розширювати та модифікувати систему у майбутньому.

Параметри матеріальних та інформаційних ресурсів також важливі, проте їх вплив може бути менш відчутним у порівнянні з людськими та програмними ресурсами. Оптимізація їх може включати в себе раціоналізацію використання обладнання та інфраструктури, а також ефективне управління та забезпечення безпеки інформації в рамках проєкту.

Загальна оптимізована формула для визначення оптимальної кількості ресурсів у проєкті може бути представлена так:

$$R^{H}_{\text{опт}} = \max h\ (\Phi_1, \Phi_2, \Phi_3, \Phi_4), \qquad (2.2)$$

де $R^{H}_{\text{опт}}$ – оптимальна кількість ресурсів,

а $h()$ – функція, яка враховує Φ_n факторів $(\Phi_1,\Phi_2,...,\Phi_n)$, що впливають на використання ресурсів.

Оптимізована формула для програмних ресурсів може виглядати наступним чином:

$$\max R^{P}_{\text{опт}} = f(X_I, X_П, X_A) \qquad (2.3)$$

де $R^{P}_{\text{опт}}$ – оптимізована кількість програмних ресурсів, а $f()$ – функція, що враховує важливі аспекти:

Відповідно завершальна формула буде мати наступний вигляд:

$R_{\text{опт}}=\{R^{M}_{\text{опт}}, R^{H}, R^{P}_{\text{опт}}, R^{I}\}$,

Оскільки, у нас немає чіткого представлення достатьої кількості кожного з необхідих ресурсів і ми маємо справу із невизначеністю, доцільно буде використати підхід теорії нечітких множин, базуючись на змінних $R^{M}_{\text{опт}}$ та $R^{P}_{\text{опт}}$. Відповідно наша математична задача буде полягати у визначенні оптимального набору параметрів для цих змнних.

Нехай $R = \{ R^{M}_{\text{опт}} , R^{P}_{\text{опт}} \}$ вектор параметрів що оптимізуються, відповідно, необхідно вирішити задачу оптимізації функції F(R) тобто,

$$\max_r F(R) \qquad (2.4)$$

при обмеженнях:

$$d_i(R) \le b_i$$

де $d_i(R)$ – функції, що обмежують і-ий ресурс у векторі b, до яких належать ресурси, що використовуються проектом.

Далі для кожного параметру r треба ввести нечітку змінну разом із функцією приналежності та параметром ваги.

Тож, для $R^{M}_{\text{опт}}$ це буде змінна A, функція $\mu_A(X)$ та параметр w_{r1},

а для $R^{P}_{\text{опт}}$ - B, $\mu_B(X)$ та параметр w_{r2},

де X – множина можливих значеннь параметрів.

Отже цільова функція буде виглядати наступним чином:

$$F(R) = \sum_{i=1}^{n} \mu_A(x_i) * w_{r1} + \mu_B(x_i) * w_{r2} \quad (2.5)$$

2.2 Опис модулів програмного забезпечення та їх функціональність

Модуль – це окрема частина програмного забезпечення, яка виконує певну функцію або набір пов'язаних функцій. Розділення програмного забезпечення на окремі модулі є стратегічним підходом, який полегшує розробку, тестування та підтримку системи. Це дозволяє розробникам концентруватися на конкретних завданнях і забезпечує більшу модульність та гнучкість програми. Крім того, розділення на модулі полегшує співпрацю між різними членами команди, оскільки кожен модуль може бути розроблений та тестований незалежно від інших. Такий підхід сприяє зменшенню складності програмного забезпечення, полегшує розуміння коду та розвиток продукту в майбутньому.

Для проєкту розробки програмного забезпечення для процесів комерційної доставки вантажів будуть розроблені різні модулі, які відповідають за різні аспекти функціональності програми.

Модуль управління замовленнями:

Цей модуль є центральною складовою частиною системи програмного забезпечення для процесів комерційної доставки вантажів. Він відповідає за всі аспекти управління замовленнями, від прийняття нових замовлень до їх виконання та відстеження.

1. Створення та обробка замовлень:

Модуль дозволяє операторам вводити нові замовлення в систему. Це може включати в себе інформацію про відправника та отримувача, опис вантажу, вагу, обсяг, вартість та інші важливі деталі. Після створення замовлення воно переходить до обробки, де визначається найкращий маршрут доставки та відповідний транспорт.

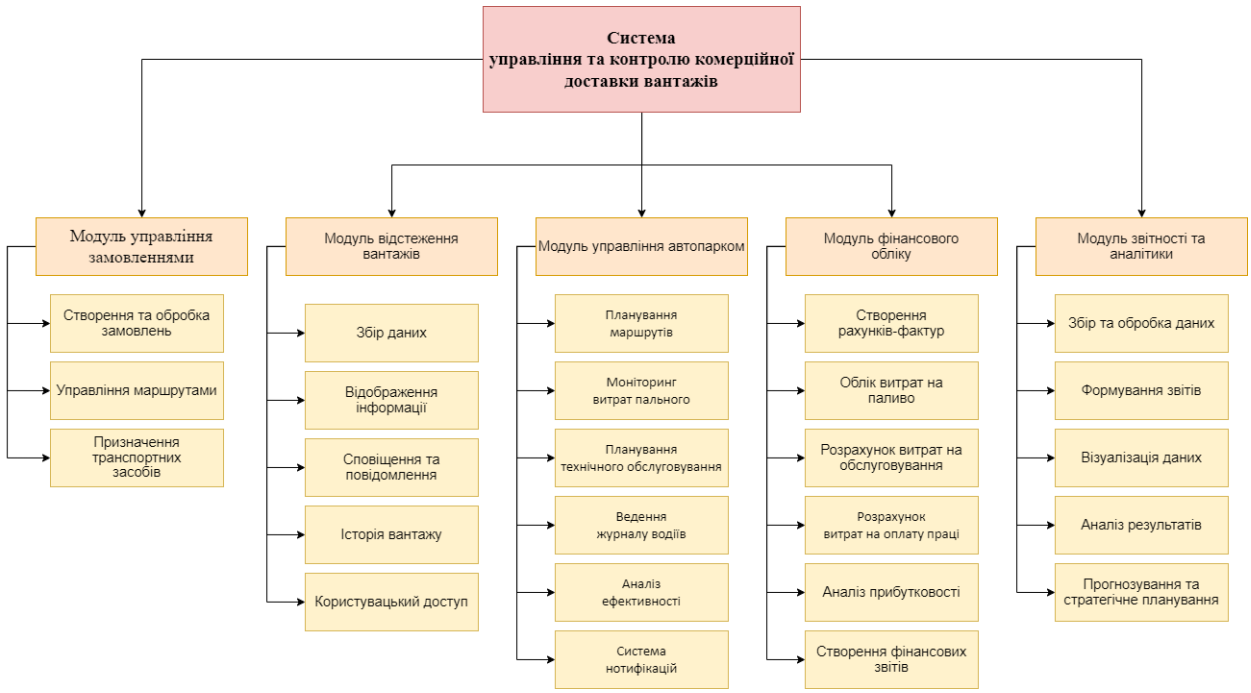


Рис. 2.2. Модулі ПЗ та їх основні функції

2. Управління маршрутами:

Модуль автоматично або за допомогою вручного втручання оператора розраховує оптимальний маршрут доставки для кожного замовлення. Враховуються різні фактори, такі як відстань, час, вага вантажу, умови дорожнього руху та інші обмеження.

3. Призначення транспортних засобів:

Після визначення маршруту модуль призначає відповідні транспортні засоби для виконання замовлення. Це може включати в себе розподіл замовлень між різними видами транспорту, такими як вантажівки, поїзди або літаки, в залежності від характеру вантажу та вимог доставки.

Модуль відстеження вантажів:

Цей модуль дозволяє відстежувати місцезнаходження вантажів під час їх транспортування. Він може включати функції отримання GPS-координат, сповіщення про статус доставки та відображення історії переміщень вантажу.

1. Збір даних:

Модуль автоматично отримує та обробляє дані щодо місцезнаходження вантажу в режимі реального часу. Це може включати в себе отримання GPS-координат з транспортних засобів або інші методи відстеження.

2. Відображення інформації:

Інформація про статус доставки та місцезнаходження вантажу відображається у вигляді графічного інтерфейсу, що дозволяє користувачам в режимі реального часу слідкувати за переміщенням їх вантажу.

3. Сповіщення та повідомлення:

Модуль може надсилати автоматичні сповіщення користувачам про зміни в статусі доставки, такі як відправлення, прибуття на склад, відправлення для доставки та успішне отримання.

4. Історія вантажу:

Історія переміщень вантажу зберігається в системі, що дозволяє користувачам переглядати інформацію про всі попередні переміщення та статуси доставки.

5. Користувацький доступ:

Користувачі можуть мати доступ до модулю через веб-портал або мобільний додаток, що дозволяє їм отримувати актуальну інформацію про свої замовлення в будь-який час та з будь-якого пристрою.

Модуль управління автопарком:

Цей модуль допомагає ефективно керувати транспортним флотом, забезпечуючи оптимальне використання ресурсів та зменшення витрат на експлуатацію.

1. Планування маршрутів:

Модуль дозволяє автоматично або з допомогою вручного втручання оператора планувати оптимальні маршрути для кожного транспортного засобу. Це враховує такі фактори, як відстань, час, умови дорожнього руху та обмеження на маршруті.

2. Моніторинг витрат пального:

Модуль відстежує споживання пального кожним транспортним засобом і генерує звіти про витрати пального. Це дозволяє ефективно керувати витратами та зменшувати витрати на пальне.

3. Планування технічного обслуговування:

Модуль відслідковує потребу в технічному обслуговуванні кожного транспортного засобу на підставі пробігу, часу експлуатації та інших факторів. Він автоматично створює розклад технічного обслуговування і нагадує про нього операторам.

4. Ведення журналу водіїв:

Модуль збирає і зберігає інформацію про роботу водіїв, включаючи час відправлення, прибуття, час відпочинку та робочий час. Це дозволяє відстежувати дотримання правил трудового режиму та безпеки дорожнього руху.

5. Аналіз ефективності:

Модуль аналізує різні показники ефективності автопарку, такі як використання палива, пробіг, час роботи транспортних засобів та інші, і генерує звіти, що дозволяють оптимізувати роботу флоту.

6. Система нотифікацій:

Модуль може автоматично надсилати повідомлення операторам про незаплановані зупинки, технічні проблеми, відставання від графіка та інші події, які потребують уваги.

Модуль фінансового обліку:

Цей модуль відповідає за фінансовий облік замовлень, оплату послуг та облік витрат. Він може включати функції створення рахунків-фактур, обліку витрат на паливо, розрахунків з водіями та клієнтами.

1. Створення рахунків-фактур:

Модуль дозволяє автоматично генерувати рахунки-фактури для клієнтів на основі інформації про замовлення, вартості послуг та інших даних. Це забезпечує швидкий та точний процес виставлення рахунків.

2. Облік витрат на паливо:

Модуль відстежує витрати на паливо для кожного транспортного засобу та генерує звіти про ці витрати. Це дозволяє контролювати та оптимізувати витрати на паливо.

3. Розрахунок витрат на обслуговування:

Модуль враховує витрати на технічне обслуговування та ремонт транспортних засобів та генерує звіти про ці витрати. Це дозволяє планувати бюджет і підтримувати транспортний флот у робочому стані.

4. Розрахунок витрат на оплату праці:

Модуль розраховує витрати на оплату праці водіїв та інших співробітників, які беруть участь у процесі доставки. Це включає в себе розрахунок зарплат, додаткових виплат та інших витрат, пов'язаних з персоналом.

5. Аналіз прибутковості:

Модуль аналізує витрати та доходи від кожного замовлення або клієнта і генерує звіти про прибутковість. Це допомагає оцінити ефективність роботи та приймати рішення щодо оптимізації бізнес-процесів.

6. Створення фінансових звітів:

Модуль створює різноманітні фінансові звіти, такі як баланс, звіт про прибутки та збитки, звіт про готовність до операцій та інші, що дозволяє керівництву компанії отримувати повну фінансову звітність.

Модуль звітності та аналітики:

Цей модуль відповідає за збір та аналіз даних про ефективність та продуктивність роботи системи доставки. Він може включати функції створення звітів про виконані замовлення, аналізу витрат та прибутковості, а також прогнозування попиту на послуги доставки.

1. Збір та обробка даних:

Модуль здійснює збір даних з різних джерел, таких як модулі управління замовленнями, відстеження вантажів, управління флотом та фінансовий облік. Після збору дані обробляються для подальшого аналізу.

2. Формування звітів:

На основі зібраних та оброблених даних модуль генерує різноманітні звіти та аналітичні дані. Це може включати в себе звіти про виконані замовлення, витрати на паливо, ефективність використання транспортних засобів, фінансові показники та багато іншого.

3. Візуалізація даних:

Модуль надає можливість візуалізації даних у вигляді графіків, діаграм, таблиць та інших форматів. Це допомагає користувачам краще розуміти дані та виявляти тенденції та патерни.

4. Аналіз результатів:

Модуль проводить аналіз зібраних даних для виявлення ключових показників продуктивності, ефективності та економічної прибутковості. Це може включати в себе оцінку виконання планів, виявлення потенційних проблемних ситуацій та виявлення можливостей для оптимізації.

5. Прогнозування та стратегічне планування:

Модуль може використовувати дані для прогнозування майбутніх тенденцій та визначення стратегічних напрямків розвитку. Це допомагає компанії приймати обґрунтовані рішення щодо розвитку та реагувати на зміни у внутрішньому та зовнішньому середовищі.

Ці модулі разом утворюють інтегровану систему управління та контролю комерційної доставки вантажів, яка допомагає оптимізувати процеси та підвищувати ефективність роботи логістичної компанії. Інтегрована система управління та контролю комерційної доставки вантажів, яка складається з розглянутих модулів, є ключовим інструментом для оптимізації логістичних процесів та підвищення ефективності діяльності логістичної компанії. Подивимося на основні переваги цієї інтегрованої системи:

1. Покращення ефективності:

Система дозволяє автоматизувати багато рутинних операцій, таких як прийняття замовлень, планування маршрутів та створення рахунків-фактур. Це допомагає збільшити швидкість обробки замовлень та зменшити кількість помилок, що може значно покращити загальну ефективність компанії.

2. Підвищення точності та контролю:

Завдяки системі відстеження вантажів та управління флотом компанія може точно відстежувати місцезнаходження вантажів, відстежувати рух транспортних засобів та контролювати їх використання. Це дозволяє забезпечити вчасну доставку та покращити рівень обслуговування клієнтів.

3. Оптимізація витрат:

Система фінансового обліку та аналітики надає компанії можливість відстежувати та аналізувати всі витрати, пов'язані з процесами доставки. Це дозволяє ідентифікувати можливості для зменшення витрат та оптимізації бізнес-процесів.

4. Покращення прийняття рішень:

Звітність та аналітика, надані системою, дозволяють керівництву компанії приймати обґрунтовані рішення щодо стратегій розвитку, вибору постачальників та оптимізації бізнес-процесів.

Загалом, інтегрована система управління та контролю комерційної доставки вантажів створює сприятливі умови для підвищення конкурентоспроможності та ефективності логістичної компанії, забезпечуючи відмінне обслуговування клієнтів та оптимальне використання ресурсів.

2.3 Структура бази даних IT-проєкту

База даних – це організована колекція даних, яка зберігається та обробляється за допомогою комп'ютерної системи. Вона представляє собою структурований набір інформації, доступ до якої може бути здійснений за

допомогою запитів, забезпечуючи ефективний пошук, оновлення та видалення даних.

Бази даних є основним елементом для зберігання та організації великого обсягу даних у різноманітних сферах діяльності, від бізнесу та науки до громадської адміністрації та освіти. Вони використовуються для зберігання інформації про клієнтів, транзакцій, товарів, виробів, наукових досліджень, студентів та багато іншого.

Бази даних не лише забезпечують зручний доступ до даних, але і дозволяють забезпечити їхню цілісність, безпеку та консистентність. Вони використовуються для виконання різноманітних завдань, таких як аналіз даних, генерація звітів, автоматизація бізнес-процесів, прийняття рішень та багато іншого.

Одним з головних переваг баз даних є їхній потенціал для організації та зберігання великого обсягу даних у структурованому та систематизованому вигляді, що сприяє покращенню ефективності та продуктивності в різних галузях діяльності.

Реляційні та нереляційні бази даних (БД) – це два основних типи систем зберігання та організації даних, які мають різні підходи до структури та обробки інформації. Нижче наведений порівняльний аналіз цих двох типів БД:

1. Структура даних:

Реляційні БД: Дані організовані у вигляді таблиць, де кожен рядок представляє запис, а кожний стовпчик - атрибут цих записів. Вони використовуються для зберігання даних у взаємозв'язаних таблицях, які можна зв'язати за допомогою ключів.

Нереляційні БД: Дані можуть бути збережені у вигляді колекцій документів, ключ-значення, графів або інших структур. Вони не вимагають строгої схеми та можуть бути більш гнучкими для зберігання різноманітних типів даних.

2. Запити:

Реляційні БД: Для отримання даних використовуються структуровані мови запитів, такі як SQL (Structured Query Language). SQL дозволяє виконувати складні запити до реляційних таблиць, включаючи вибірку, оновлення, видалення та об'єднання даних.

Нереляційні БД: Зазвичай використовуються спеціалізовані мови запитів або API для взаємодії з даними. Ці мови можуть бути менш структурованими та мають більшу свободу в обробці даних.

3. Масштабованість:

Реляційні БД: Зазвичай складніше масштабувати горизонтально (додавання нових серверів) через складність зберігання взаємозв'язаних даних.

Нереляційні БД: Мають кращу масштабованість, особливо горизонтальну, тому що дані можуть бути розподілені на багато серверів.

4. Гнучкість схеми:

Реляційні БД: Мають строгу схему даних, яка вимагає означення структури таблиць заздалегідь. Зміни в схемі можуть бути складними.

Нереляційні БД: Мають більш гнучку схему даних, де структура даних може змінюватися з часом без значних зусиль.

5. Використання:

Реляційні БД: Зазвичай використовуються для додатків, які потребують складних зв'язків між даними, таких як бухгалтерські програми, системи управління відносинами з клієнтами (CRM) та управління ресурсами підприємства (ERP).

Нереляційні БД: Широко використовуються для веб-програмування, аналізу великих обсягів даних (Big Data), інтернету речей (IoT), мобільних додатків та інших випадків використання, де потрібна висока швидкість та масштабованість.

Обидва типи БД мають свої переваги та недоліки, і вибір між ними залежить від конкретних вимог проєкту та віддає пріоритет одній або іншій характеристиці, такі як гнучкість схеми, масштабованість чи продуктивність.

У даному проєкті, який стосується процесів комерційної доставки вантажів, може бути вигідніше використовувати нереляційну базу даних. Нереляційна база даних може бути кращим варіантом за наведених нижче причин:

1. Гнучкість схеми:

У сфері логістики та транспорту можуть виникати ситуації, коли потрібно зберігати різноманітні типи даних, такі як географічні координати, зображення вантажів, статуси доставок тощо. Нереляційна БД може забезпечити більшу гнучкість у зберіганні цих даних без необхідності в строгій схемі.

2. Масштабованість:

У логістичній галузі може виникати потреба у масштабуванні системи, особливо коли мова йде про великі обсяги даних або потреби в реальному часі. Нереляційні БД часто мають кращу масштабованість, що дозволить забезпечити стабільну роботу системи при зростанні навантаження.

3. Взаємодія з сучасними технологіями:

У сфері логістики дедалі більше використовуються сучасні технології, такі як IoT, додатки з мобільним доступом, аналітика в реальному часі тощо. Нереляційна БД може бути більш сумісною з такими технологіями та забезпечити потрібну швидкодію та гнучкість.

Хоча реляційні бази даних також мають свої переваги, з урахуванням специфіки даного проєкту, де можуть зустрічатися великі обсяги різноманітних даних та потреба у гнучкості, нереляційна база даних може бути кращим вибором для забезпечення потреб проєкту з комерційної доставки вантажів.

Структура бази даних для проєкту розробки програмного забезпечення для процесів комерційної доставки вантажів може бути складною, але добре організованою та ефективною. Нижче наведено структуру модель бази даних та зв'язків між таблицями.

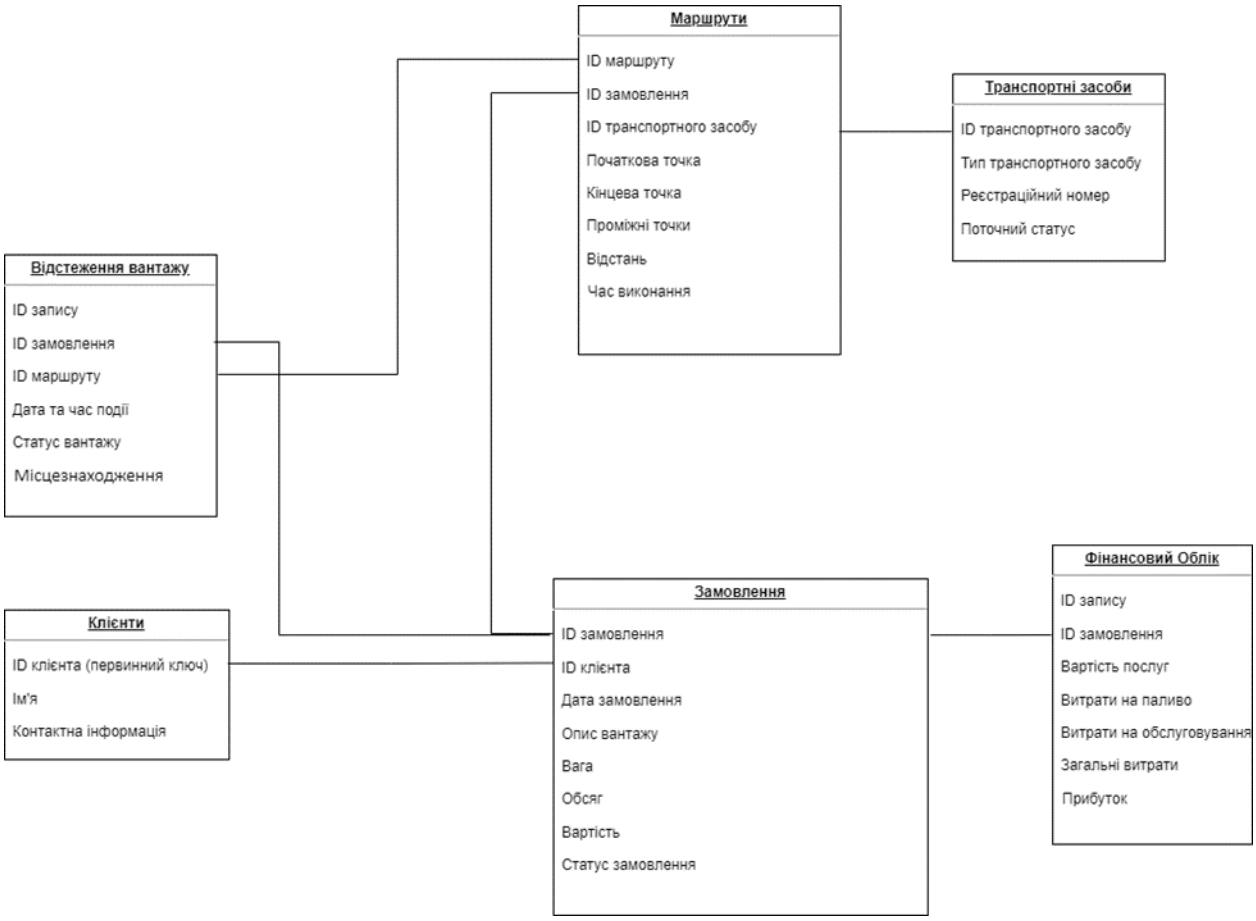


Рис. 2.3. Структура та звя'зки у базі даних

Таблиця «Клієнти» відповідає за зберігання інформації про клієнтів компанії, включаючи їхні особисті дані (ім'я, контактна інформація). Ця таблиця дозволяє вести базу клієнтів і швидко отримувати доступ до їхніх даних при обробці замовлень.

Таблиця «Замовлення» відповідає за зберігання інформації про замовлення вантажів, включаючи деталі замовлення (опис, вага, обсяг, вартість), статус замовлення та інші важливі дані. Ця таблиця дозволяє вести облік замовлень і керувати ними під час їх обробки та доставки.

Таблиця «Транспортні засоби» відповідає за зберігання інформації про транспортні засоби компанії, включаючи їхні типи, реєстраційні номери та поточний статус. Ця таблиця дозволяє вести облік транспортних засобів і керувати їхнім використанням під час виконання замовлень.

Таблиця «Маршрути» відповідає за зберігання інформації про маршрути доставки для кожного замовлення, включаючи початкову та кінцеву точки, проміжні точки, відстань та час виконання. Ця таблиця дозволяє оптимізувати маршрутизацію та планування доставки.

Таблиця «Відстеження вантажу» відповідає за зберігання інформації про статус вантажів та їх місцезнаходження під час доставки. Ця таблиця дозволяє клієнтам та компанії в режимі реального часу відстежувати рух вантажів та отримувати актуальну інформацію про їхнє місцезнаходження.

Таблиця «Фінансовий облік» відповідає за зберігання інформації про фінансові аспекти замовлень, включаючи витрати та прибуток від кожного замовлення. Ця таблиця дозволяє компанії вести фінансовий облік і оцінювати прибутковість різних замовлень.

2.4. Розроблення концептуальної та фізичної моделей бази даних проєкту

2.4.1 Розробка концептуальної моделі БД

Концептуальна модель бази даних (БД) – це абстрактна представлення структури даних та їх взаємозв'язків в рамках конкретної області бізнесу чи додатка. Вона описує сутності (або класи), їхні атрибути та зв'язки між ними без врахування технічних деталей реалізації бази даних.

Основні елементи концептуальної моделі бази даних:

- Сутності (або класи): Це об'єкти чи концепції, які важливі для даних. Наприклад, у системі управління книгами сутностями можуть бути «Книга», «Автор», «Видавництво» тощо.
- Атрибути: Це властивості сутностей, які характеризують їх. Наприклад, у сутності «Книга» атрибутами можуть бути «Назва», «Автор», «Рік видання» тощо.
- Зв'язки: Це взаємозв'язки між сутностями, які показують, як вони пов'язані одна з одною. Наприклад, у системі управління книгами

зв'язком може бути «Автор пише Книгу», що показує, що один автор може написати багато книг, а книга може мати одного або декілька авторів.

Концептуальна модель бази даних (БД) є ключовим етапом у процесі проєктування бази даних і виконує ряд важливих функцій. По-перше, вона дозволяє розібратися в структурі даних та їх зв'язках в рамках конкретної області бізнесу або додатка. Це дає можливість краще зрозуміти, які інформаційні об'єкти важливі для системи, і як вони пов'язані між собою.

По-друге, концептуальна модель допомагає у визначенні потреб користувачів та бізнес-вимог до системи. Вона дозволяє створити абстрактне представлення про те, які дані потрібно зберігати та як вони пов'язані з функціональністю системи. Це сприяє виробленню згоди між розробниками, бізнес-аналітиками та клієнтами щодо того, яким повинен бути кінцевий продукт.

По-третє, концептуальна модель дозволяє виявити потенційні проблеми або неоднозначності щодо структури даних ще на ранніх етапах проєктування. Це дозволяє виправити їх до початку фази фізичного проєктування, щозберігає час та ресурси в майбутньому.

Концептуальна модель бази даних (рис. 2.4) є важливим інструментом для розробників та бізнес-аналітиків, оскільки вона допомагає краще зрозуміти потреби та вимоги бізнесу, визначити структуру даних та їх зв'язки, і врешті-решт, забезпечити успішну реалізацію проєкту з мінімальними проблемами.

Таким чином, концептуальна модель є фундаментальним інструментом для систематичного підходу до вирішення проблем, планування і управління проектами, а також для покращення комунікації та співпраці між різними фахівцями.

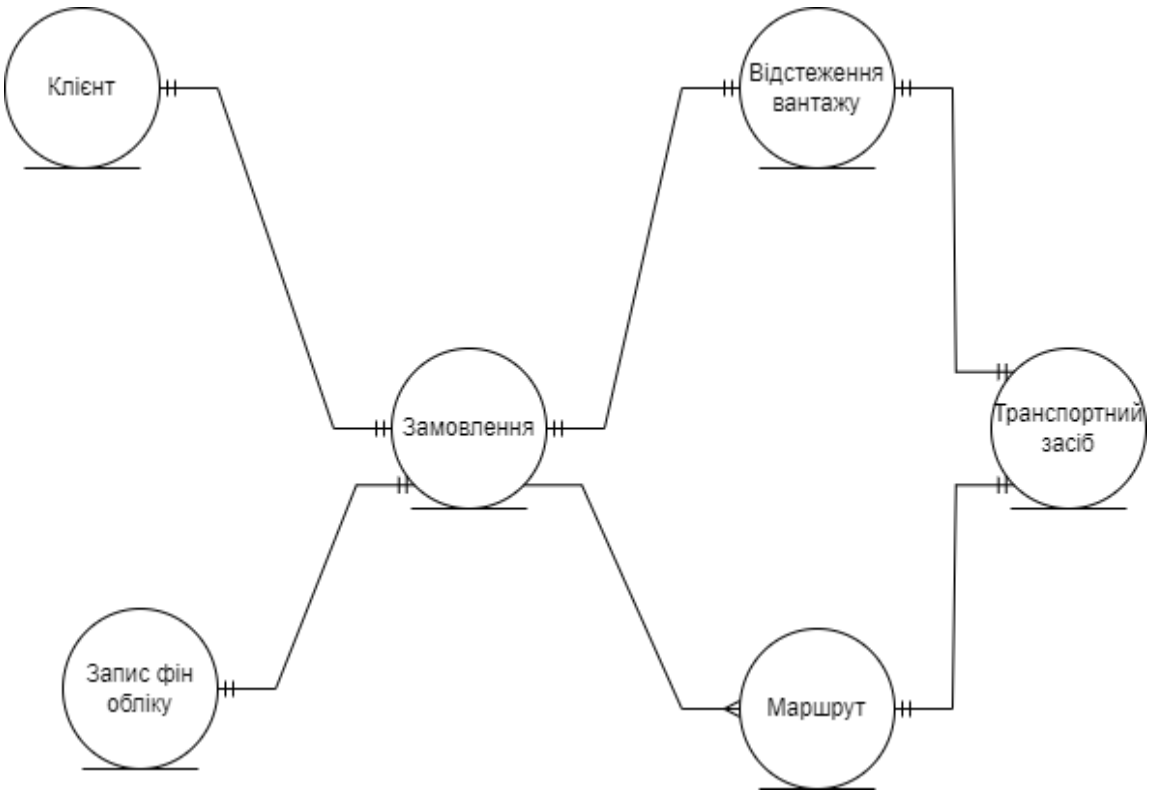


Рис. 2.4. Концептуальна модель бази даних

У даному поданні пропонується концептуальна модель бази даних для проєкту розробки програмного забезпечення, яка спрямована на автоматизацію та оптимізацію управління процесами доставки. Ця модель розроблена з урахуванням потреб бізнесу та включає сутності, атрибути та зв'язки, необхідні для забезпечення комплексного контролю за усіма аспектами процесу доставки вантажів. Детальне розглядання концептуальної моделі дозволить краще зрозуміти структуру та функціональність системи, сприяючи подальшому розвитку та вдосконаленню програмного забезпечення для логістичних потреб.

1. Зв'язок між сутностями «Клієнт» та «Замовлення»: Кожне замовлення (З) пов'язане з одним клієнтом (К), оскільки кожен клієнт може робити одне або кілька замовлень.

2. Зв'язок між сутностями «Замовлення» та «Маршрут»: Кожне замовлення (З) може мати один або декілька маршрутів (М), оскільки для кожного замовлення можуть бути встановлені різні маршрути доставки вантажу.
3. Зв'язок між сутностями «Маршрут» та «Транспортний засіб»: Кожен маршрут (М) пов'язаний з одним транспортним засобом (Т), оскільки кожному маршруту необхідно призначити транспортний засіб для виконання доставки.
4. Зв'язок між сутностями «Відстеження вантажу» та «Замовлення» та «Маршрут»: Кожне відстеження вантажу (В) пов'язане з одним замовленням (З) та одним маршрутом (М), оскільки для кожного відстеження вантажу необхідно вказати, до якого замовлення та маршруту воно відноситься.
5. Зв'язок між сутностями «Фінансовий облік» та «Замовлення»: Кожен запис фінансового обліку (Ф) пов'язаний з одним замовленням (З), оскільки кожному замовленню відповідає певний обсяг фінансових операцій.

2.4.2 Розробка фізичної моделі БД

Фізична модель бази даних – це конкретна реалізація концептуальної моделі, яка визначає структуру та параметри бази даних на рівні, зрозумілому для конкретної бази даних, такої як MySQL, PostgreSQL або MongoDB. Фізична модель включає в себе деталізовану специфікацію таблиць, стовпців, індексів, ключів, обмежень, типів даних та інших технічних аспектів.

У розрізі проєкту з розробки програмного забезпечення для процесів комерційної доставки вантажів, фізична модель бази даних є важливою для наступних цілей:

1. Визначення оптимальної структури даних: Фізична модель дозволяє визначити, як саме дані будуть організовані в базі даних для оптимального доступу, швидкості та ефективності.

2. Підготовка для реалізації: Фізична модель надає детальний план для створення бази даних на конкретній платформі. Це включає в себе створення таблиць, визначення типів даних, індексів, ключів тощо.
3. Оптимізація продуктивності: Належне проєктування фізичної моделі може сприяти підвищенню продуктивності бази даних, зменшенню часу виконання запитів та підвищенню загальної ефективності системи.
4. Забезпечення цілісності даних: Фізична модель також включає в себе визначення обмежень цілісності даних, таких як унікальність, зовнішні ключі, перевірки, що забезпечують правильність та надійність даних в базі.

Після розробки концептуальної моделі бази даних для проєкту управління та контролю процесів комерційної доставки вантажів, наступним кроком є розробка фізичної моделі, яка визначає конкретну реалізацію цієї моделі на рівні певної бази даних. Фізична модель надає детальні технічні специфікації структури даних та визначає всі необхідні атрибути, зв'язки та обмеження для ефективного зберігання та обробки інформації. Нижче наведена фізична модель бази даних для забезпечення ефективного управління та контролю процесів доставки вантажів:

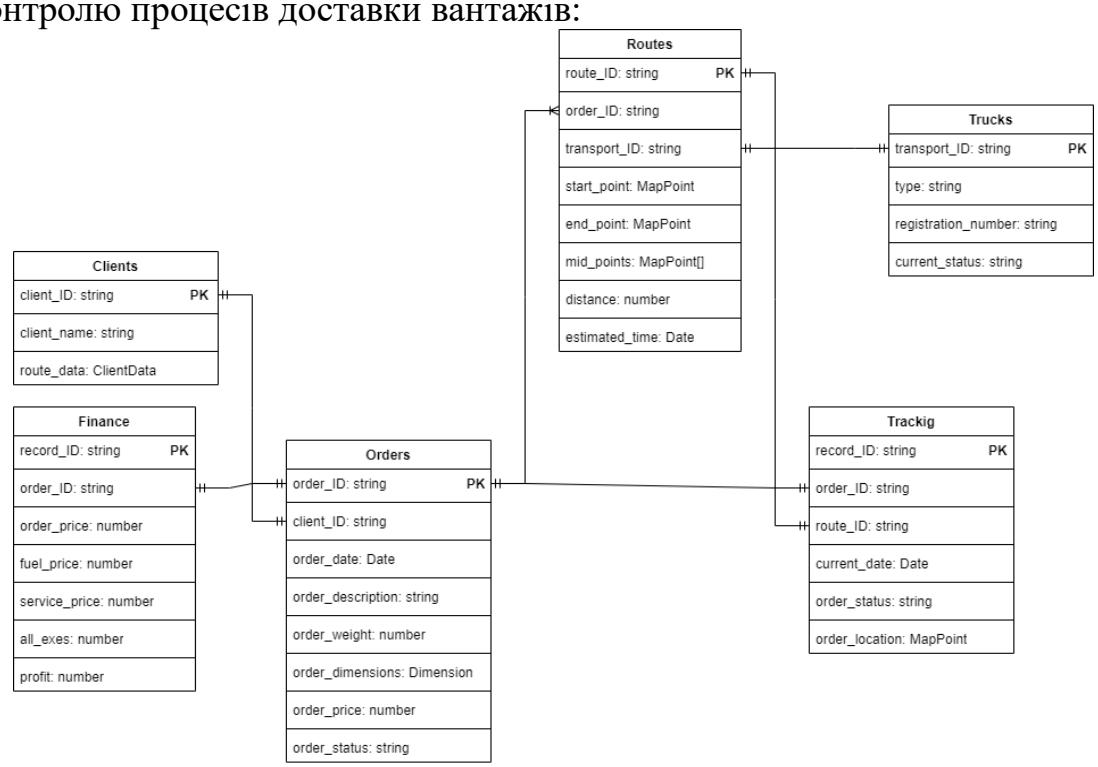


Рис. 2.5. Фізична модель бази даних

У підсумку, фізична модель бази даних для проєкту управління та контролю процесів комерційної доставки вантажів визначає конкретну реалізацію концептуальної моделі на рівні обраної бази даних. Ця модель включає таблиці з відповідними полями та індексами, які відображають сутності та їх зв'язки. Кожна таблиця має свою унікальну структуру, а також визначені індекси для оптимізації доступу до даних. Фізична модель забезпечує ефективне зберігання, обробку та управління даними, що дозволяє підвищити ефективність та продуктивність системи управління доставкою вантажів.

РОЗДІЛ 3. ГНУЧКІ ТЕХНОЛОГІЇ СТВОРЕННЯ ПРОДУКТУ ТА ПЛАНУВАННЯ ІТ-ПРОЄКТУ

3.1 Визначення функціональних та нефункціональних вимог до проєкту

Вимоги до проєкту – це формально визначені функціональні, нефункціональні та технічні характеристики, які система або програмне забезпечення повинні відповідати для задоволення потреб та очікувань користувачів і бізнесу. Вони визначаються під час аналізу, планування та розробки проєкту та є основою для його успішної реалізації. Визначення вимог дозволяє чітко зрозуміти, що очікують від системи її користувачі та бізнес-клієнти. Це допомагає уникнути непорозумінь та конфліктів між розробниками та замовниками проєкту. Вимоги служать основою для планування робіт і розробки архітектури системи. Вони допомагають розробникам усвідомити обсяг та складність проєкту, а також розподілити завдання між командами. Визначення вимог дозволяє переконатися, що програмне забезпечення буде відповідати вимогам стандартів безпеки, надійності та ефективності. Чітко сформульовані вимоги допомагають визначити критерії успіху проєкту. Це дозволяє оцінювати та порівнювати результати реалізації з очікуваними результатами. Визначення вимог на ранній стадії дозволяє виявити потенційні проблеми та ризики, пов'язані з проєктом, що дозволяє їх вчасно вирішувати та мінімізувати їх вплив на подальшу реалізацію.

Отже, визначення вимог до проєкту є критичним етапом у процесі його розробки, оскільки вони визначають напрямок та обсяг робіт, а також забезпечують високу якість та відповідність продукту очікуванням користувачів та бізнесу.

Основні вимоги до проєкту розробки програмного забезпечення для процесів комерційної доставки вантажів можна узагальнити наступним чином:

Функціональні вимоги:

1. Система повинна забезпечувати управління та контроль всіма етапами процесу доставки вантажів, включаючи прийом замовлень, маршрутизацію, відстеження вантажів, обробку оплати та створення фінансових звітів.
2. Наявність інтерфейсу для клієнтів та співробітників компанії, що дозволяє здійснювати замовлення, переглядати статус доставки та здійснювати необхідні дії.
3. Можливість автоматизації деяких процесів, таких як визначення оптимального маршруту доставки на основі географічних даних та трафіку.

Надійність та безпека:

4. Забезпечення безпеки та конфіденційності даних клієнтів та компанії.
5. Забезпечення стабільної роботи системи в умовах високого навантаження та можливих відмов обладнання.

Інтеграція:

6. Можливість інтеграції з існуючими системами управління складами, фінансовими системами та іншими програмними засобами, що використовуються в компанії.

Масштабованість:

7. Можливість розширення функціональності та масштабування системи з ростом бізнесу та збільшенням обсягу доставок.

Ефективність:

8. Швидка обробка та відповідь на запити користувачів.
9. Оптимізація витрат часу та ресурсів при виконанні різних операцій, таких як розрахунок маршрутів або відстеження вантажів.

Зручність використання:

10. Інтуїтивний та зрозумілий інтерфейс для користувачів різних рівнів навичок.

11. Наявність документації та підтримки для користувачів та адміністраторів системи.

Ці вимоги визначають основні параметри та функціональність системи, необхідні для успішної реалізації проєкту та забезпечення задоволення потреб бізнесу та користувачів. Варто зауважити, що це лише перший рівень вимог, кожна із яких може мати у собі підпункти, що більш детально визначений функціонал.

Наприклад для пункту 1, секції функціональних вимог наступні два рівні будуть виглядати наступним чином:

1. Управління замовленнями:
   1.1. Можливість створення нових замовлень користувачами.
   1.2. Можливість перегляду, редагування та видалення існуючих замовлень.
   1.3. Автоматичне призначення унікального ідентифікатора кожному замовленню.
2. Маршрутизація доставки:
   2.1. Розрахунок оптимальних маршрутів доставки на основі географічних даних, ваги та об'єму вантажу.
   2.2. Можливість додавання та редагування проміжних точок маршруту.
   2.3. Автоматичне оновлення маршрутів у випадку зміни умов або введення нової інформації.
3. Відстеження вантажів:
   3.1. Можливість перегляду статусу та місцезнаходження вантажу в режимі реального часу.
   3.2. Надання сповіщень користувачам про зміни статусу вантажу.
   3.3. Історія руху вантажу та подробиці кожного етапу доставки.
4. Фінансовий облік:
   4.1. Реєстрація витрат на доставку, таких як паливо, обслуговування, оплата праці тощо.

4.2. Розрахунок загальної вартості послуг та виручки від кожного замовлення.
4.3. Створення фінансових звітів для аналізу прибутковості та ефективності роботи.

5. Керування користувачами та доступом:
5.1. Реєстрація нових користувачів з різними ролями та рівнями доступу.
5.2. Аутентифікація користувачів та забезпечення безпеки доступу до системи.
5.3. Можливість управління правами доступу та обмеження функціональності для різних категорій користувачів.

Оскільки, вимоги визначені дуже абстрактно лише їх недостатньо для успішного запуску розробки проєкту. Потрібно визначити конкректні метрики для розуміння виконання вимог.

Тож основні метрики за якими буде відслідковуватись виконання вимог проєкту це:

1. Відсоток покриття функціональних вимог тестами: Ця метрика вимірює відсоток функціональних вимог, які було протестовано. Вона дозволяє визначити, наскільки повно вимоги функціональності системи було покрито тестами.
2. Час відповіді системи: Метрика вимірює час, який потрібно системі для обробки запитів користувачів або виконання певних операцій. Це дозволяє переконатися, що система відповідає вимогам до швидкості реакції.
3. Кількість виявлених та виправлених дефектів: Ця метрика вказує на ефективність тестування та якість програмного забезпечення. Вона вимірює кількість дефектів, виявлених під час тестування, та кількість дефектів, які було виправлено.
4. Відсоток виконаних вимог за планом: Ця метрика вказує на те, яка частина вимог була виконана за запланований час. Вона дозволяє

визначити рівень виконання проєкту в порівнянні з відведеним на нього часом та ресурсами.

5. Стабільність системи: Метрика вимірює частоту виникнення помилок чи збоїв у системі після випуску. Вона дозволяє визначити стабільність та надійність програмного забезпечення.
6. Рівень задоволення користувачів: Ця метрика вимірюється через опитування користувачів щодо їх задоволеності системою та її функціональністю. Вона дозволяє отримати зворотний зв'язок від користувачів та враховувати їхні побажання та потреби в подальшому розвитку системи.

Ці метрики дозволять нам більш детально аналізувати прогрес та виконання проєкту. Але й цього не достатньо, ще потрібно визначити ціль, яку проєкт має досягти та до якої ми ідемо. Вона необхідна для розуміння чи виконали ми початковий задум та чи можна вважати проєкт виконаним. Тож для проєкту розробки програмного забезпечення для процесів комерційної доставки вантажів буде поставлено наступні цілі:

1. Створення системи, яка забезпечить повний цикл управління та контролю процесів доставки вантажів протягом 12 місяців з моменту початку розробки.
2. Зниження часу доставки вантажів на 20% протягом перших 6 місяців експлуатації програмного забезпечення.
3. Забезпечення 98.5% доступності системи та захисту даних користувачів та компанії протягом усього періоду експлуатації.
4. Досягнення показника задоволення користувачів на рівні не менше 85% протягом перших 3 місяців використання системи.
5. Успішна інтеграція з існуючими системами управління складами та фінансовими програмами протягом перших 9 місяців після випуску програмного забезпечення.

6. Забезпечення можливості масштабування системи для обробки не менше ніж подвоєного обсягу замовлень протягом перших 12 місяців після введення в експлуатацію.

3.2 Організаційна структура команди проєкту

Організаційна структура – це схема, що визначає структурні підрозділи, ланцюжки командування, систему взаємодії між ними та розподіл обов'язків і відповідальності в організації [28]. Ця структура допомагає визначити, як організація буде функціонувати, як взаємодіяти зі своїми структурними підрозділами, як управляти ресурсами та як приймати рішення.

Організаційна структура важлива з кількох причин:

- Координація та співпраця: Вона визначає, як будуть взаємодіяти різні відділи та працівники для досягнення загальних цілей організації. Це дозволяє забезпечити ефективну координацію дій та спільну роботу.
- Розподіл відповідальності: Вона визначає, хто відповідає за які області діяльності та які функції відповідальності виконує кожен працівник. Це робить управління більш ефективним та допомагає уникнути дублювання завдань або пропусків.
- Ієрархія управління: Вона визначає ступені влади та ланцюжки командування в організації. Це спрощує процеси управління та забезпечує відповідність між вищим та нижчим рівнями управління.
- Ефективність ресурсів: Організаційна структура допомагає оптимізувати використання ресурсів, включаючи людські, фінансові та матеріальні ресурси. Це дозволяє забезпечити оптимальне функціонування організації.
- Контроль та нагляд: Вона визначає механізми контролю за діяльністю різних структурних підрозділів та працівників. Це допомагає забезпечити виконання поставлених завдань та досягнення цілей організації.

Отже, організаційна структура визначає спосіб організації та управління діяльністю організації з метою досягнення її цілей та забезпечення успішного функціонування.

Організаційна структура проєкту розробки програмного забезпечення для процесів комерційної доставки вантажів може включати наступні основні компоненти:

Керівництво проєкту:

- Проєктний менеджер: відповідає за управління всім процесом розробки програмного забезпечення, визначає завдання, розподіляє ресурси та контролює виконання графіку робіт.
- Технічний керівник: відповідає за технічну архітектуру та технічні рішення в проєкті.

Розробницький та тестувальний персонал:

- Команда програмістів: розробляє програмний код відповідно до вимог проєкту.
- QA-інженери: відповідають за тестування програмного забезпечення та виявлення помилок.

Аналітичний відділ:

- Бізнес-аналітики: займаються вивченням вимог бізнесу та клієнтів, формулюванням вимог до програмного забезпечення.

Дизайн-відділ:

- UX/UI дизайнери: відповідають за розробку інтерфейсу користувача та забезпечення його зручності та привабливості.

Інфраструктурна команда:

- Системний адміністратор: Забезпечує налаштування, підтримку і безпеку серверів та інших компонентів IT-інфраструктури.
- Девопс інженер: Відповідає за інтеграцію та доставку коду, автоматизацію процесів розгортання та підтримку безперервного циклу розробки.

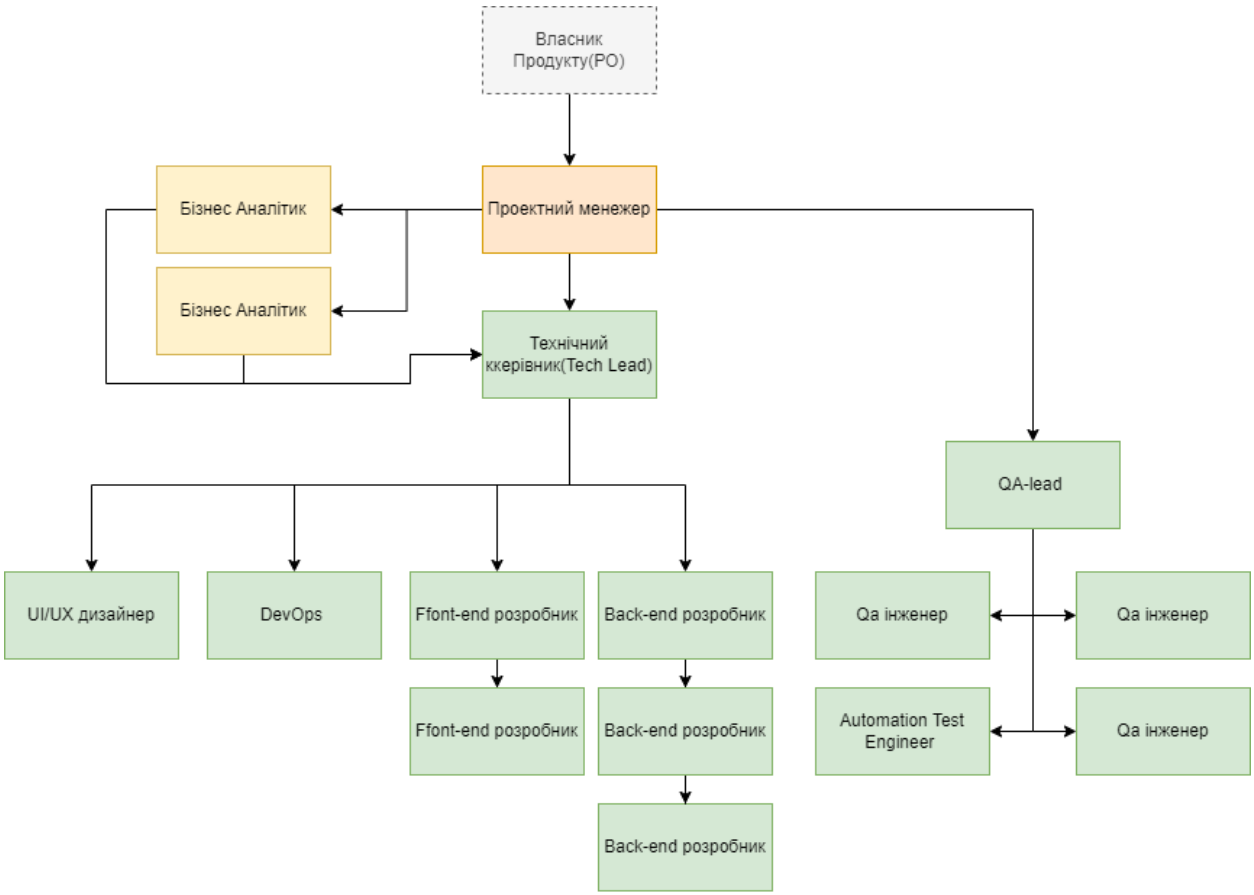


Рис. 3.1. Організаційна структура проєкту

Керівництво проєкту включає в себе кілька ключових ролей, кожна з яких відповідає за певні аспекти проєкту та забезпечує його успішну реалізацію:

Проєктний менеджер:

Відповідальності:

- Управління всіма етапами проєкту, від визначення вимог до випуску готового продукту.
- Розподіл ресурсів та завдань між учасниками команди.
- Контроль за дотриманням графіку робіт та бюджету.
- Забезпечення комунікації між учасниками проєкту та зацікавленими сторонами.
- Вирішення конфліктів та ризиків, які можуть виникнути під час реалізації проєкту.

Технічний керівник:

Відповідальності:

- Розробка стратегії технічного розвитку проєкту.
- Визначення технічних вимог та архітектури програмного забезпечення.
- Вибір технологій та інструментів розробки, які використовуються в проєкті.
- Забезпечення відповідності технічних рішень вимогам бізнесу та користувачів.

Ці дві ролі виконують ключові функції у керівництві проєктом, забезпечуючи координацію, контроль та технічну експертизу на різних етапах розробки програмного забезпечення для процесів комерційної доставки вантажів.

Розробницький та тестувальний персонал включає в себе різні ролі, кожна з яких відповідає за конкретні завдання у процесі розробки програмного забезпечення. Нижче розпишемо ці ролі та їх функції:

Команда програмістів:

- *Senior Developer* (Старший розробник): Ця роль виконується досвідченим програмістом, який має глибокі знання технологій та архітектурних рішень. Він відповідає за розробку складних модулів та архітектурних компонентів програмного забезпечення, а також надає консультації менш досвідченим членам команди.
- *Software Developer* (Розробник програмного забезпечення): Ці спеціалісти відповідають за реалізацію функціональності програмного забезпечення на основі вимог та дизайну. Вони пишуть код, тестують його та вносять необхідні зміни під час розробки.

QA-інженери:

- *QA Engineer* (Інженер з забезпечення якості): Ця роль відповідає за планування, розробку та виконання тестових сценаріїв для перевірки функціональності та якості програмного забезпечення. Вони також виявляють та документують помилки, співпрацюючи з розробницькими командами для їх виправлення.

– *Automation Test Engineer* (Інженер з автоматизованого тестування): Ці спеціалісти відповідають за розробку та підтримку автоматизованих тестів, які допомагають автоматизувати тестування функціональності програмного забезпечення. Вони розробляють скрипти тестування, виконують тестові сценарії та аналізують результати.

Зв'язок між розробницьким та тестувальним персоналом, а також іншими членами команди забезпечується за допомогою інших важливих ролей у проєкті, таких як DevOps інженер, UI/UX дизайнер та бізнес-аналітик.

DevOps інженер відповідає за підтримку інфраструктури розробки та впровадження програмного забезпечення. Вони розробляють автоматизовані процеси для збирання, тестування та розгортання програмного забезпечення. DevOps інженер також забезпечує моніторинг, логування та безпеку системи.

UI/UX дизайнер відповідає за розробку інтерфейсу користувача (UI) та користувацького досвіду (UX) програмного забезпечення. Вони створюють дизайн екранів, компонентів і взаємодії, щоб забезпечити зручність, привабливість та ефективність використання продукту користувачами.

Бізнес-аналітик відповідає за збір, аналіз та формулювання вимог до програмного забезпечення з точки зору бізнесу та клієнтів. Вони співпрацюють зі зацікавленими сторонами для розуміння бізнес-процесів, визначення потреб користувачів та формулювання стратегій впровадження програмних рішень.

Кожна з цих ролей відіграє важливу роль у процесі розробки програмного забезпечення для процесів комерційної доставки вантажів. Співпраця та координація між цими ролями дозволяє забезпечити успішне впровадження та оптимальне функціонування продукту.

Загалом, організаційна структура проєкту допомагає забезпечити систематизацію та контроль над роботою команди, що в свою чергу сприяє успішному виконанню проєкту. Нижче наведена структура команди в аспекті кількісного складу та можливих витрат (табл. 3.1).

Таблиця 3.1

Структура команди в аспекті кількісного складу та можливих витрат

№	Позиція	Кількість людей	Тривалість (місяці)	Ціна за місяць роботи ($)
1	**2**	**3**	**4**	**5**
1	Проєктний менеджер	1	12	2500
2	Бізнес Аналітик	2	12	2400
3	Технічний керівник	1	11	4500
4	UI/UX дизайнер	1	12	2000
5	DevOps інженер	1	11	2300
6	Front-end розробник	2	10	2000
7	Back-end розробник	3	10	2200
8	QA-lead інженер	1	10	2500
9	QA інженер	3	10	1700
10	AQA інженер	1	10	2000
11	HR спеціаліст	1	2	1100

Також було визначено вимоги до кожної з позицій. Чітке визначення вимог до кандидатів є важливим для успішного підбору та збереження кваліфікованого персоналу. Вимоги допомагають забезпечити, що кандидати відповідають потребам та очікуванням компанії, а також мають необхідні навички та досвід для виконання поставлених завдань. Це зменшує ризик неправильного відбору кандидатів і підвищує шанси на успішну роботу та розвиток персоналу в організації. Крім того, чіткі вимоги сприяють ефективному плануванню та управлінню процесом набору персоналу, а також покращують взаєморозуміння між керівництвом та кадровим відділом.

Тож нижче наведено табл. 3.2 із основними вимогами та функціями членів команди проєкту.

Таблиця 3.2

Основні вимоги та функції членів команди

Позиція	Вимоги
1	**2**
Проєктний менеджер	– Досвід управління IT-проєктами, бажано у сфері логістики або транспорту. – Вміння планувати, організовувати та контролювати весь процес розробки ПЗ. – Вміння ефективно комунікувати з усіма учасниками проєкту та замовниками. – Знання методологій управління проєктами, таких як Agile або Scrum.
Бізнес аналітик	– Досвід роботи у сфері аналізу бізнес-процесів, перевагою буде досвід у логістиці або електронній комерції. – Вміння визначати та формулювати вимоги користувачів до програмного забезпечення. – Знання методів та інструментів аналізу вимог, таких як Use Case, User Stories тощо.
Технічний керівник	– Глибокі технічні знання у сфері розробки програмного забезпечення. – Досвід роботи у ролі технічного лідера команди розробників. – Вміння приймати стратегічні та технічні рішення, спрямовані на досягнення цілей проєкту.
UI/UX дизайнер	– Досвід роботи у сфері дизайну користувацького інтерфейсу та користувацьких досліджень. – Креативність та здатність створювати естетичні та зручні дизайн-концепції. – Знання програм для дизайну, таких як Adobe Photoshop, Adobe Illustrator, Sketch тощо.
DevOps інженер	– Досвід у розгортанні та підтримці інфраструктури проєктів на основі хмарних сервісів. – Вміння автоматизувати процеси розробки, тестування та розгортання за допомогою інструментів, таких як Docker, Kubernetes, Jenkins, Ansible тощо. – Знання практик Continuous Integration та Continuous Deployment (CI/CD).
Front-end розробник	– Досвід у розробці клієнтської частини веб-додатків з використанням HTML, CSS та JavaScript. – Знання фреймворків та бібліотек, таких як React.js, Angular, Vue.js тощо. – Вміння оптимізувати та адаптувати інтерфейс для різних пристроїв та браузерів.

Продовження табл. 3.2

1	2
Back-end розробник	– Досвід у розробці серверної частини веб-додатків з використанням мов програмування, таких як Java, Python, Node.js тощо. – Знання фреймворків та технологій, таких як Spring, Django, Express.js тощо. – Розуміння принципів роботи та розробки RESTful API.
QA-lead інженер	– Досвід у проведенні тестування програмного забезпечення та керівництва QA-командою. – Знання методологій тестування, таких як ручне, автоматизоване та відладка програмного забезпечення. – Вміння створювати тест-плани, виконувати тестування та аналізувати результати.
QA інженер	– Досвід у проведенні ручного програмного забезпечення. – Знання технік тестування, таких як модульне, інтеграційне та системне. – Вміння створювати та виконувати тест-кейси, вести журнали дефектів та співпрацювати з розробницькою командою.
AQA інженер	– Досвід у розробці автоматизованих тестів з використанням фреймворків, таких як Selenium, Appium, TestNG тощо. – Знання мов програмування для автоматизації тестування, таких як Java, Python, JavaScript тощо. – Вміння аналізувати та вдосконалювати процес автоматизації тестування.
HR спеціаліст	– Досвід у наборі та утриманні персоналу в IT-сфері, бажано у сфері розробки програмного забезпечення. – Вміння проводити співбесіди, визначати критерії відбору та розробляти програми навчання та розвитку персоналу. – Знання законодавства у сфері трудових відносин та уміння працювати з HR-системами.

Врахування всіх ключових аспектів при формуванні команди проєкту є запорукою успішного його виконання, відповідно даний аргумент став основою при формуванні даного підрозділу кваліфікаційної роботи.

3.3 Календарне планування та віхи проєкту

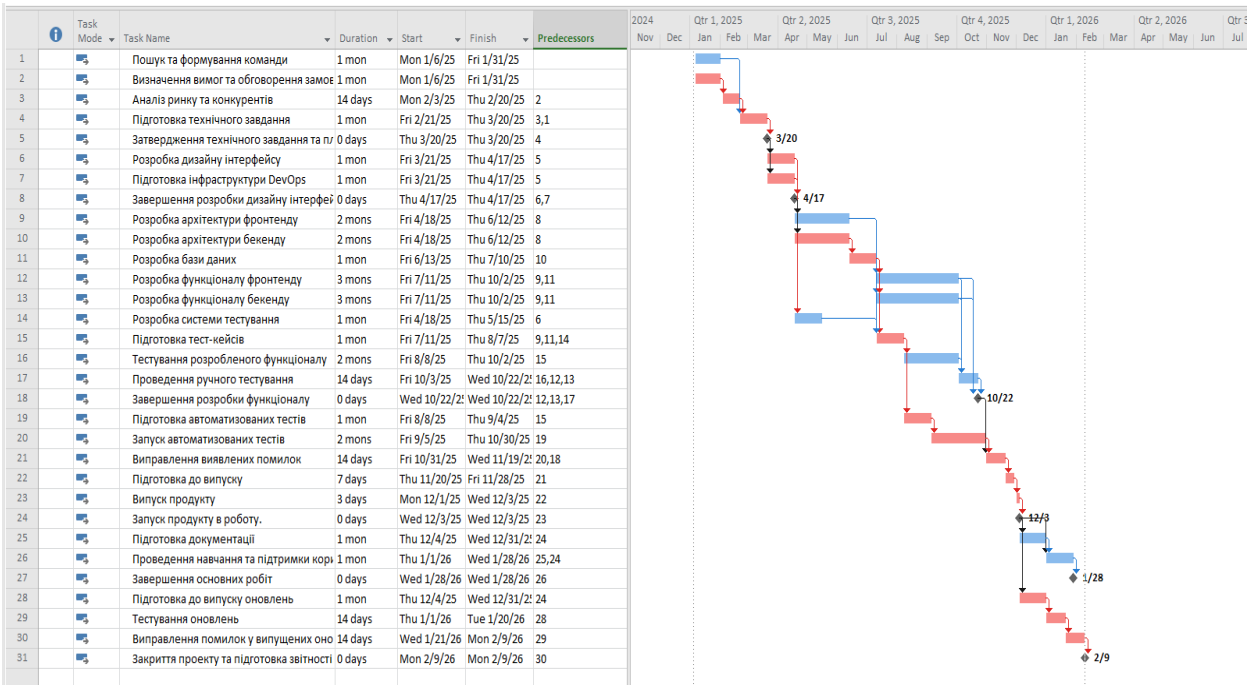
Календарне планування проєкту – це процес створення та управління графіком робіт, що визначає послідовність та тривалість кожної задачі чи етапу проєкту у відповідності до календаря. Основна мета календарного планування - це побудова реалістичного графіка виконання проєкту, який дозволяє планувати та контролювати часові рамки та ресурси, необхідні для успішного завершення проєкту.

Календарне планування важливе з декількох причин:

1. Дозволяє структурувати та організувати роботу над проєктом за конкретними термінами, що сприяє ефективному використанню часу та уникненню затримок у виконанні завдань.
2. Надає засоби для контролю та відстеження прогресу виконання робіт, що дозволяє оперативно реагувати на зміни та ризики та при необхідності коригувати плани.
3. Допомагає виявити критичні шляхи у проєкті - послідовність завдань, яка визначає мінімальний час, необхідний для завершення проєкту.
4. Дозволяє здійснити аналіз доступних ресурсів та визначити їхнє раціональне розподілення для досягнення максимально ефективних результатів.
5. Створює базу для комунікації між учасниками проєкту, яка базується на конкретних термінах та часових рамках, що сприяє взаєморозумінню та уникненню непорозумінь.

Таким чином, календарне планування є критичним інструментом для успішного управління проєктами, оскільки забезпечує чітку структуру, контроль і координацію всіх аспектів роботи, необхідних для досягнення поставлених цілей у визначені терміни.

Нижче наведено основні задачі та віхи проєкту (рис. 3.2).



	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		Пошук та формування команди	1 mon	Mon 1/6/25	Fri 1/31/25	
2		Визначення вимог та обговорення замов	1 mon	Mon 1/6/25	Fri 1/31/25	
3		Аналіз ринку та конкурентів	14 days	Mon 2/3/25	Thu 2/20/25	2
4		Підготовка технічного завдання	1 mon	Fri 2/21/25	Thu 3/20/25	3,1
5		Затвердження технічного завдання та пл	0 days	Thu 3/20/25	Thu 3/20/25	4
6		Розробка дизайну інтерфейсу	1 mon	Fri 3/21/25	Thu 4/17/25	5
7		Підготовка інфраструктури DevOps	1 mon	Fri 3/21/25	Thu 4/17/25	5
8		Завершення розробки дизайну інтерфей	0 days	Thu 4/17/25	Thu 4/17/25	6,7
9		Розробка архітектури фронтенду	2 mons	Fri 4/18/25	Thu 6/12/25	8
10		Розробка архітектури бекенду	2 mons	Fri 4/18/25	Thu 6/12/25	8
11		Розробка бази даних	1 mon	Fri 6/13/25	Thu 7/10/25	10
12		Розробка функціоналу фронтенду	3 mons	Fri 7/11/25	Thu 10/2/25	9,11
13		Розробка функціоналу бекенду	3 mons	Fri 7/11/25	Thu 10/2/25	9,11
14		Розробка системи тестування	1 mon	Fri 4/18/25	Thu 5/15/25	6
15		Підготовка тест-кейсів	1 mon	Fri 7/11/25	Thu 8/7/25	9,11,14
16		Тестування розробленого функціоналу	2 mons	Fri 8/8/25	Thu 10/2/25	15
17		Проведення ручного тестування	14 days	Fri 10/3/25	Wed 10/22/2	16,12,13
18		Завершення розробки функціоналу	0 days	Wed 10/22/2	Wed 10/22/2	12,13,17
19		Підготовка автоматизованих тестів	1 mon	Fri 8/8/25	Thu 9/4/25	15
20		Запуск автоматизованих тестів	2 mons	Fri 9/5/25	Thu 10/30/25	19
21		Виправлення виявлених помилок	14 days	Fri 10/31/25	Wed 11/19/2	20,18
22		Підготовка до випуску	7 days	Thu 11/20/25	Fri 11/28/25	21
23		Випуск продукту	3 days	Mon 12/1/25	Wed 12/3/25	22
24		Запуск продукту в роботу.	0 days	Wed 12/3/25	Wed 12/3/25	23
25		Підготовка документації	1 mon	Thu 12/4/25	Wed 12/31/2	24
26		Проведення навчання та підтримки кор	1 mon	Thu 1/1/26	Wed 1/28/26	25,24
27		Завершення основних робіт	0 days	Wed 1/28/26	Wed 1/28/26	26
28		Підготовка до випуску оновлень	1 mon	Thu 12/4/25	Wed 12/31/2	24
29		Тестування оновлень	14 days	Thu 1/1/26	Tue 1/20/26	28
30		Виправлення помилок у випущених оно	14 days	Wed 1/21/26	Mon 2/9/26	29
31		Закриття проекту та підготовка звітності	0 days	Mon 2/9/26	Mon 2/9/26	30

Рис. 3.2. Задачі та віхи проєкту

Ці задачі є основними, й визначають майбутній список робіт, що буде виконуватись. Нижче наведено короткий опис до головних задач:

1. *Визначення вимог та обговорення замовника*. Встановлення і уточнення потреб та очікувань замовника, а також визначення основних вимог до проєкту.
2. *Аналіз ринку та конкурентів*. Проведення дослідження ринку та аналіз конкурентів для з'ясування потенційних можливостей та визначення конкурентних переваг проєкту.
3. *Підготовка технічного завдання*. Формулювання детальних вимог до функціональності, архітектури та інші технічні параметри проєкту.
4. *Розробка дизайну інтерфейсу*. Створення візуальної концепції та дизайну користувацького інтерфейсу продукту.
5. *Підготовка інфраструктури DevOps*. Налаштування середовища розробки, тестування та розгортання продукту.
6. *Розробка архітектури фронтенду*. Створення структури та основних компонентів клієнтської частини програмного забезпечення.

7. *Розробка архітектури бекенду*. Проєктування та реалізація структури та основних компонентів серверної частини програмного забезпечення.
8. *Розробка бази даних*. Створення та оптимізація бази даних для зберігання і обробки інформації проєкту.
9. *Розробка функціоналу фронтенду*. Програмування функціональності та інтерфейсних елементів клієнтської частини програмного забезпечення.
10. *Розробка функціоналу бекенду*. Реалізація серверної логіки та взаємодії з базою даних.

Основні віхи проєкту – це ключові точки у його розвитку, що позначають досягнення певних етапів або результатів, які відображаються на прогрес проєкту. Для поточного прроекту основними віхами будуть:

1. Затвердження технічного завдання та плану робіт: Ця віха вказує на завершення процесу встановлення вимог до проєкту та розробку плану робіт, що є основою для подальшої розробки.
2. Завершення розробки дизайну інтерфейсу та архітектури DevOps: Ця віха відзначає завершення проєктування користувацького інтерфейсу та налаштування інфраструктури DevOps, що є підготовкою до початку розробки програмного забезпечення.
3. Завершення розробки функціоналу: На цій вісі відзначається завершення розробки функціональності програмного забезпечення, що включає реалізацію всіх основних функцій та можливостей.
4. Запуск продукту в роботу: Ця віха позначає момент, коли програмне забезпечення вперше встановлюється на робочих середовищах та починає активно використовуватися користувачами.
5. Завершення основних робіт: Ця віха вказує на завершення всіх основних етапів проєкту, включаючи розробку, тестування та впровадження програмного забезпечення.
6. Закриття проєкту та підготовка звітності: Остання віха позначає завершення проєкту та підготовку звітності перед стейкхолдерами, включаючи оцінку прогресу, досягнення та витрати.

3.4 Обґрунтування вибору методології управління проєктом

Стандарт методології управління проєктом є необхідним і корисним інструментом для ефективного керування проєктом з розробки програмного забезпечення для процесів комерційної доставки вантажів. Він надає наступні переваги:

1. Стандарт методології управління проєктом надає рамки, процеси та інструменти для організації та структурування усіх етапів проєкту, від визначення вимог до закриття проєкту.
2. Він допомагає в плануванні ресурсів, визначенні графіку та керуванні ризиками, що дозволяє ефективно контролювати прогрес та вчасно реагувати на зміни у ході проєкту.
3. Використання стандарту методології управління проєктом дозволяє всім учасникам проєкту працювати з однаковим розумінням процесів та стандартів, що сприяє уніфікації та зниженню помилок.
4. Чіткі процеси та методології дозволяють ідентифікувати та управляти ризиками проєкту, що допомагає уникнути проблем та забезпечує успішне завершення проєкту в обумовлені строки та бюджет.
5. Стандарт методології управління проєктом визначає ролі, відповідальності та комунікаційні канали, що сприяє ефективній взаємодії між учасниками проєкту та стейкхолдерами.

Отже, використання стандарту методології управління проєктом є ключовим для забезпечення успішного виконання проєкту з розробки програмного забезпечення для процесів комерційної доставки вантажів.

Але, оскільки на даний момент часу є безліч різноманітних стандартів, вибрати найкращий іноді буває досить складно. Тому необхідно провести аналіз найпопулярніших та визначити їх переваги та недоліки.

Серед методологій, що будуть розглядатись було вибрано наступні:

1. Стандарт PMBOK (Project Management Body of Knowledge), який включає набір знань, процесів та практик управління проєктами [24].

2. PRINCE2 (Projects IN Controlled Environments) - це методологія управління проєктами, розроблена у Великобританії. Вона забезпечує структурований підхід до управління проєктами та включає в себе принципи, процеси та ролі.
3. Agile - це гнучкий підхід до управління проєктами, який ставить акцент на ітеративному та інкрементальному розробці, співпраці замовника та здатності змінюватися.
4. Kanban - це методологія управління, яка базується на візуальному представленні завдань та обмеженні робочого процесу для підвищення продуктивності та управління потоками роботи.

Нижче наведено порівняльний аналіз для методологій управління проєктом: PMBOK, PRINCE2, Agile (Scrum), та Kanban, у розрізі їхньої відповідності різним типам проєктів.

Таблиця 3.3

Порівняльний аналіз методологій

Характеристика / Методологія	**PMBOK**	**PRINCE2**	**Agile (Scrum)**	**Kanban**
1	**2**	**3**	**4**	**5**
Сфера застосування	Широкий спектр проєктів у різних галузях та індустріях	Великі, складні проєкти, зокрема у IT, будівництві, державних проєктах тощо	Гнучкі проєкти, які вимагають швидких змін та взаємодії з клієнтом	Проєкти, де важлива візуалізація робочого процесу та управління потоками роботи
Основні принципи	Знання, процеси та практики управління проєктами	Процеси, ролі та принципи управління контрольованими середовищами	Ітеративний та інкрементальний розвиток, співпраця з клієнтом, здатність до змін	Візуалізація робочого процесу, обмеження робочого потоку, постійне вдосконалення

Продовження табл. 3.3

1	2	3	4	5
Структура ролей	Менеджер проєкту, Спеціаліст з контролю, Спонсор проєкту, Спеціалісти з відділів	Проєктний керівник, Керівник проєкту, Виконавець, Замовник, Спонсор, Директор проєкту	Продуктовий власник, Scrum Master, Розробники, QA інженери, Команда користувачів	Лідер команди, Розробники, QA інженери, Команда підтримки
Змінність	Низька	Низька	Висока	Висока
Підходить для маленьких проєктів	Так	Ні	Так	Так
Підходить для великих проєктів	Так	Так	Так	Так

Після проведення аналізу існуючих методологій було обрано гібридний підхід, що складається із двох методологій PMBOK та Agile. Цей підхід використовує найкращі практики обох методологій для досягнення успіху в проєкті. PMBOK надає широкий набір знань, процесів та практик управління проєктами, що може бути корисним для визначення загальної структури проєкту, управління ризиками, контролю витрат тощо. В той же час, Agile забезпечує гнучкий підхід до розробки, який спрямований на швидке реагування на зміни, впровадження нових функцій та взаємодію з клієнтом.

Наприклад ми можемо використовувати PMBOK для створення основного плану проєкту, визначення ризиків та контролю витрат, а Agile для розробки програмного забезпечення та спілкування з клієнтом для отримання фідбеку.

Також використання стандартів PMBOK оптимальне для управління проєктними процесами та ресурсами, а Agile для організації ітеративного розвитку та впровадження нових функцій.

Гібридний підхід дає можливість комбінувати переваги обох методологій та створювати більш ефективну стратегію управління проєктами, яка відповідає унікальним потребам та характеристикам кожного проєкту.

Об'єднання PMBOK та Agile може бути доцільним для проєкту розробки програмного забезпечення для процесів комерційної доставки вантажів з наступних причин:

1. PMBOK надає широкий спектр знань та процесів управління проєктом, включаючи планування, контроль витрат, ризиків та стейкхолдерів. Це допоможе забезпечити структурований підхід до управління проєктом та ресурсами.
2. Agile (зокрема Scrum) дозволить команді швидко розробляти та впроваджувати нові функції програмного забезпечення в ітеративному режимі. Це особливо важливо для проєктів, де потреби клієнта можуть змінюватися або вимагати швидкої реакції.
3. Agile сприяє активній співпраці з клієнтом та постійному залученню його до процесу розробки. Це допоможе забезпечити високу якість продукту та відповідність його вимогам.
4. PMBOK надає структуру для ефективного управління ризиками та змінами у проєкті. Поєднання PMBOK та Agile дозволить забезпечити контроль ризиків і змін, зберігаючи при цьому гнучкість та швидкість реагування.
5. Комбінація PMBOK та Agile дозволить забезпечити високу якість продукту через систематичність управління проєктом та спрямованість Agile на вдосконалення робочих процесів.

Це принесе розробці проєкту багато переваг, таких як:

1. Гнучкість та адаптивність: Поєднання PMBOK та Agile дозволяє команді адаптуватися до змінних умов проєкту та вимог клієнта, забезпечуючи гнучкий підхід до управління проєктом.
2. Ефективне управління ресурсами та ризиками: PMBOK надає структурований підхід до управління ресурсами та ризиками, що

доповнює Agile, який забезпечує швидке реагування на виявлені проблеми та зміни.

3. Прозорість та комунікація: Agile сприяє активній комунікації з клієнтом та учасниками проєкту, а PMBOK надає структуру для забезпечення прозорості та ефективної комунікації між учасниками.
4. Швидкий розвиток та впровадження нових функцій: Agile дозволяє швидко розробляти та впроваджувати нові функції в програмне забезпечення, тоді як PMBOK забезпечує структурованість та контроль впровадження цих змін.
5. Оптимізація ресурсів: Поєднання PMBOK та Agile дозволяє оптимізувати використання ресурсів проєкту, забезпечуючи ефективне використання часу, коштів та людських ресурсів.

Ці плюси дозволяють створити ефективну стратегію управління проєктом, яка враховує якість, вартість та швидкість виконання проєкту, та відповідає унікальним вимогам та потребам для проєкту розробки програмного забезпечення для процесів комерційної доставки вантажів.

3.5 Використання методології Scrum для управління проєктом

Scrum – це одна з найпопулярніших методологій розробки програмного забезпечення в рамках Agile. Вона базується на принципах ітеративного та інкрементального розвитку, а також на співпраці команди та її здатності до самоорганізації.

Спочатку для розробки ПЗ потрібно було визначити функціональні та нефукціональні вимоги. Нижче наведено список основних вимог: Функціональні вимоги:

1. Реєстрація користувачів: Система повинна забезпечити можливість реєстрації користувачів, що включає в себе зберігання основних даних користувача та створення унікального облікового запису.

2. Керування вантажами: Користувачі повинні мати можливість створювати, змінювати та видаляти інформацію про вантажі, включаючи опис, вагу, розмір та інші характеристики.
3. Відстеження доставки: Система повинна забезпечити функціональність для відстеження статусу доставки вантажів, щоб користувачі могли перевірити місцезнаходження своїх вантажів у режимі реального часу.
4. Оповіщення користувачів: Система повинна автоматично надсилати сповіщення користувачам про зміни статусу їхніх вантажів або про плановані доставки за допомогою SMS, електронної пошти або пуш-сповіщень.
5. Адміністрування системи: Адміністратор повинен мати можливість керувати користувачами, вантажами, статусами доставок та іншими аспектами системи через адміністративний інтерфейс.

Нефункціональні вимоги:

1. Безпека даних: Система повинна забезпечувати високий рівень захисту конфіденційності та цілісності даних користувачів, вантажів та іншої конфіденційної інформації.
2. Швидкодія: Система повинна бути швидкою та масштабованою, забезпечуючи швидку відповідь на запити користувачів навіть при великому обсязі даних та навантаженні.
3. Надійність: Система повинна бути надійною та стійкою до відмов, забезпечуючи доступність сервісу протягом усього часу роботи.
4. Користувацький інтерфейс: Користувацький інтерфейс повинен бути інтуїтивно зрозумілим та легким у використанні для забезпечення зручності та задоволення від використання системи.
5. Сумісність: Система повинна бути сумісною з різними пристроями та платформами, такими як комп'ютери, смартфони та планшети, та працювати на різних операційних системах та веб-браузерах.

На основі цих вимог було складено UserStory, для розуміння вимог користувачів системи, нижче наведено деякі із них:

1. Я, як клієнт, хочу мати можливість реєстрації, щоб мати особистий обліковий запис та отримувати персоналізовані послуги.
2. Я, як клієнт, хочу додавати нові вантажі, щоб замовляти та відстежувати їх доставку.
3. Я, як користувач, хочу відстежувати статус вантажу, щоб мати інформацію про його поточне місцезнаходження та стан доставки.
4. Я, як адміністратор, хочу мати можливість редагування інформації про вантажі, щоб коригувати дані, якщо необхідно.
5. Я, як користувач, хочу отримувати сповіщення про зміни статусу вантажу, щоб бути завчасно проінформованим про його доставку.
6. Я, як користувач, хочу мати можливість управління моїм обліковим записом, щоб змінювати особисті дані та налаштовувати установки приватності.
7. Я, як користувач, хочу здійснювати пошук вантажів за різними параметрами, щоб швидко знаходити потрібні вантажі.
8. Я, як користувач, хочу отримувати звіт про доставку, щоб мати документальне підтвердження процесу доставки.
9. Я, як користувач, хочу мати можливість звернутися до служби підтримки, щоб отримати допомогу або вирішити технічні питання.
10. Я, як користувач, хочу мати можливість відміняти замовлення, щоб скасувати вантаж, якщо це необхідно.

Грунтуючись на наведених вище сторі було розроблено початковий беклог проєкту. Нижче наведено назву задач та їх ідентифікатор:
PRJDS_01: Реєстрація користувачів.

PRJDS_02: Створення форми додавання нового вантажу.

PRJDS_03: Налаштування системи відстеження статусу вантажів.

PRJDS_04: Розробка панелі адміністратора для редагування даних вантажів.

PRJDS_05: Реалізація системи сповіщень про зміни статусу вантажів.

PRJDS_06: Розробка сторінки управління особистим обліковим записом.
PRJDS_07: Реалізація функції пошуку вантажів за різними параметрами.
PRJDS_08: Розробка модуля для формування звіту про доставку вантажів.
PRJDS_09: Створення сторінки контактів для звернень до служби підтримки.
PRJDS_10: Розробка функції відміни замовлення вантажу.
PRJDS_11: Оптимізація реєстраційного процесу для зручності користувачів.
PRJDS_12: Вдосконалення інтерфейсу форми додавання нового вантажу.
PRJDS_13: Покращення системи відстеження статусу вантажів для швидкого доступу до інформації.
PRJDS_14: Оптимізація процесу редагування даних вантажів для адміністраторів.
PRJDS_15: Реалізація механізму автоматичних сповіщень користувачам про зміни в статусах вантажів.

3.6 Життєвий цикл проєкту та його етапи

Життєвий цикл проєкту – це послідовність фаз або етапів, через які проєкт пройде від його ініціації до завершення. Кожен етап включає в себе певні дії, завдання та виробляє певні результати, що допомагають досягти цілей проєкту. В основному у життєвий цикл проєкту входять наступні фази:

1. Ініціація: Це початкова фаза, під час якої визначаються бізнес-потреби та цілі проєкту. Тут також формується команда, визначаються важливі стейкхолдери та робиться оцінка приблизного обсягу робіт.
2. Планування: На цьому етапі деталізується стратегія реалізації проєкту. Формується робочий план, в якому визначаються обсяг, бюджет, графік та ресурси. Також на цьому етапі розробляються плани управління якістю, комунікаціями, ризиками тощо.
3. Виконання: На цьому етапі команда здійснює роботу, яка визначена у плані проєкту. Тут відбувається розробка, тестування та впровадження

результатів роботи. Стейкхолдери можуть отримувати регулярні звіти про прогрес.

4. Моніторинг та контроль: На цьому етапі виконується постійний моніторинг прогресу проєкту, вимірюються витрати, порівнюються реальні результати з планованими. Якщо необхідно, вносяться корективи в план проєкту.
5. Завершення або закриття проєкту: На цьому етапі виконується фіналізація всіх процесів та завдань проєкту. Перевіряються всі вимоги, виконані роботи, оцінюється якість та досягнення цілей. Після цього проєкт офіційно закривається, можливо, залишаючи певні витяги для майбутнього використання.

Тож і проєкт розробки програмного забезпечення для процесів комерційної доставки вантажів також буде включати ці фази. Проте, зважаючи на специфіку проєкту, поний життєвий цикл буде виглядати наступним чином:

1. Ініціація проєкту
2. Планування
3. Виконання
4. Перевірка та оцінка
5. Доставка та впровадження
6. Експлуатація та підтримка
7. Закриття проєкту

Ці етапи утворюють повний життєвий цикл проєкту, який забезпечує ефективний та систематичний підхід до розробки та впровадження програмного забезпечення для процесів комерційної доставки вантажів.

Розглядаючи процес розробки програмного забезпечення для комерційної доставки вантажів, на першому етапі, ініціації проєкту, визначаються основні цілі та бізнес-потреби. На цьому етапі формується проєктна команда і розподіляються ролі, такі як скрам-майстер, власник

продукту і розробник. Крім того, встановлюється початкове технічне завдання для проєкту та оцінюються ризики, які можуть вплинути на реалізацію.

На другому етапі, плануванні, розробляється детальний план проєкту. Команда визначає обсяг робіт і створює бэклог продукту, включаючи всі необхідні функції та вимоги. На цьому етапі також визначаються ресурси, бюджет і графік проєкту.

Третій етап – виконання – це активний процес розробки програмного забезпечення. Команда працює над реалізацією функції, виконуючи завдання з продуктового беклогу. Щоденні Scrum-зустрічі допомагають підтримувати постійну комунікацію та вирішувати проблеми.

На четвертому етапі, верифікації та оцінці, виконана робота перевіряється на відповідність вимогам та оцінюється якість продукту. Наприкінці кожного спринту проводиться огляд спринту, щоб продемонструвати результати роботи власнику продукту та іншим зацікавленим сторонам.

На етапі 5, «доставка та впровадження», завершене програмне забезпечення випускається і впроваджується в робоче середовище. Це включає тестування, затвердження та підтримку після випуску.

Етап 6 «Експлуатація та підтримка» охоплює обслуговування та підтримку продукту після випуску. Команда вирішує будь-які виявлені помилки та впроваджує вдосконалення.

Сьомий етап, закриття проєкту, передбачає оцінку успіху проєкту та досягнення його цілей. Після завершення проєкту створюється звіт і зберігається проєктна документація.

РОЗДІЛ 4. ТЕХНОЛОГІЇ УПРАВЛІННЯ ІТ-ПРОЄКТОМ

4.1 Управління ризиками проєкту

У сучасному світі управління ризиками проєкту стає надзвичайно важливою складовою успішності будь-якого проєкту, особливо в галузі розробки програмного забезпечення. Ризики можуть виникнути на будь-якому етапі проєкту та мають потенційно великий вплив на його результат. Тому ефективне управління цими аспектами є важливим завданням для керівників проєктів.

Враховуючи швидкі темпи технологічного розвитку та постійні зміни в умовах ринку, управління ризиками стає складним завданням, яке потребує системного підходу та гнучкості. Навички виявлення, аналізу та ефективного управління ризиками дозволяють зменшити ймовірність негативних наслідків та забезпечити успішне завершення проєкту.

У цьому розділі розглянуто основні аспекти управління ризиками проєкту в контексті розробки програмного забезпечення для процесів комерційної доставки вантажів. Буде проведено аналіз потенційних ризиків, їх класифікацію та оцінку впливу на проєкт, а також розглянуто стратегії та методи управління цими ризиками.

Для успішного управління ризиками потрібно провести ряд ключових подій, метою яких буде ідентифікувати, аналізувати, відстежувати та керувати ризиками протягом усього проєкту. Тож після ідентифікації було сформовано наступний список можливих ризиків, розділених на сегменти:

a) Програмні ризики:
    1) Несумісність програмних компонентів з ОС.
    2) Припинення підтримки зовнішніх бібліотек, які використовуються в проєкті.
    3) Нестача обчислювальної спроможності серверів/місця.
    4) У команди зникли доступи до інструментів розробки.

b) Апаратні ризики:

1) Вимкнення фізичних серверів через відсутність електрики.
2) Втрата даних через несправність.
3) Перевантаження мережі.
4) Несумісність обладнання.

c) Внутрішні ризики проєкту:
1) Мобілізація членів команди.
2) Погана комунікація між членами команди.
3) Недостатні технічні навички членів команди.
4) Відставання від графіку розробки.

d) Зовнішні:
1) Зміни державного устрою.
2) Зміни податкового законодавства.
3) Затримки платежів за реалізовану розробку.
4) Кримінальні та економічні злочини.

e) Форс мажори:
1) Військові дії на території розробки та застосування.
2) Введення необхідних соціальних обмежень.
3) Втрата всіх напрацювань впродовж розробки.
4) Природні стихійні лиха.

Після чого для кожної з можливих подій було присвоєно критерій сили впливу та керованість. Що є важливою частиною процесу управління ризиками. Визначення сили впливу допомагає відібрати найбільш важливі ризики для управління. Ризики з великим потенціальним впливом на проєкт мають вищий пріоритет і вимагають більшої уваги. Знаючи силу впливу ризиків, команда може краще розподілити свої ресурси, зосереджуючи зусилля на управлінні найбільш важливими або найбільш ймовірними ризиками. Сила впливу також впливає на те, як будуть розроблятися стратегії керування ризиками. Для ризиків із високим потенціальним впливом можуть бути розроблені більш складні стратегії або запасні плани. Результати наведені у таблиці нижче (табл. 4.1).

Таблиця 4.1

Ризики проєкту

№	Тип ризику	Ризикова подія	Сила впливу	Керованість
1	Програмні ризики	Несумісність програмних компонентів з ОС	Низька	Висока
2		Припинення підтримки зовнішніх бібліотек, які використовуються в проєкті	Середня	Висока
3		Нестача обчислювальної спроможності серверів/місця	Середня	Висока
4		У команди зникли доступи до інструментів розробки	Низька	Середня
5	Апаратні ризики	Вимкнення фізичних серверів через відсутність електрики	Висока	Середня
6		Втрата даних через несправність	Висока	Середня
7		Перевантаження мережі	Середня	Висока
8		Несумісність обладнання	Середня	Висока
9	Внутрішні ризики проєкту (команда)	Мобілізація членів команди	Середня	Низька
10		Погана комунікація між членами команди	Середня	Висока
11		Недостатні технічні навички членів команди	Середня	Висока
12		Відставання від графіку розробки	Висока	Середня
13	Зовнішні (оточення)	Зміни державного устрою (обмеження законами)	Середня	Низька
14		Зміни податкового законодавства	Висока	Низька
15		Затримки платежів за реалізовану розробку	Висока	Низька
16		Кримінальні та економічні злочини (перевірки та затримки роботи проєкту)	Висока	Низька
17	Форс мажорні	Військові дії на території розробки та застосування	Висока	Низька
18		Введення необхідних соціальних обмежень (пандемія)	Середня	Середня
19		Втрата всіх напрацювань впродовж розробки	Висока	Середня
20		Природні стихійні лиха	Середня	Низька

Після ідентифікації ризиків було проведено їх оцінку та ранжування відповідно до цієї оцінки. Це потрібно для того, щоб визначити найпріоритетніші ризикові події та основні заходи щодо їх превентизації. Нижче наведена таблиця із зведеними оцінками від групи експертів що визначали ризик за наступними критеріями: затримки у часі, фінансові витрати, ймовірність, частота:

Таблиця 4.2

Оцінка ризиків проєкту

№	Ризикова подія	Затримки у часі		Фінансові втрати		Ймовірність		Частота (за проєкт)		Важливість ризику (компл. показн.)
		Якіс. оц	Кільк. оц.	Якіс. оц	Кільк. оц.	Якіс. оц	Кільк. оц.	Якіс. оц	Кільк. оц.	
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	Несумісність програмних компонентів з ОС	нв	3	сн	4	сс	5	нс	2	14
2	Припинення підтримки зовнішніх бібліотек, які використовуються в проєкті	сн	4	нв	3	нс	2	нн	1	10
3	Нестача обчислювальної спроможності серверів/місця	нс	2	сс	5	нв	3	нн	1	11
4	У команди зникли доступи до інструментів розробки	нс	2	нс	2	св	6	нс	2	12
5	Вимкнення фізичних серверів через відсутність електрики	св	6	сн	4	вс	8	сс	5	23
6	Втрата даних через несправність	вн	7	св	6	нв	3	нс	2	18

Продовження табл.4.2

1	2	3	4	5	6	7	8	9	10	11
7	Перевантаження мережі	сс	5	нв	3	вн	7	сн	4	19
8	Несумісність обладнання	сн	4	сс	5	св	6	нв	3	18
9	Мобілізація членів команди	вв	9	св	6	вс	8	нс	2	25
10	Погана комунікація між членами команди	св	6	св	6	св	6	сс	5	23
11	Недостатні технічні навички членів команди	вн	7	вн	7	сс	5	нс	2	21
12	Відставання від графіку розробки	вв	9	вс	8	сс	5	сн	4	26
13	Зміни державного устрою (обмеження законами)	св	6	вн	7	св	6	сн	4	23
14	Зміни податкового законодавства	сс	5	вн	7	св	6	сс	5	23
15	Затримки платежів за реалізовану розробку	вн	7	вс	8	св	6	сн	4	25
16	Кримінальні та економічні злочини(перевірки та затримки роботи проєкту)	вн	7	нв	3	сс	5	нс	2	17
17	Військові дії на території розробки та застосування	вв	4	вс	5	вв	4	вв	1	14
18	Втрата всіх напрацювань впродовж розробки	вв	9	вс	8	сс	5	сн	1	23
20	Природні стихійні лиха	сн	4	нс	2	нс	2	нн	1	9

Після проведення оцінки та визначення важливості, можна ранжувати ризикові події в залежності від їхньої важливості. Тож було сформовано наступний список:

Найбільш впливові (20-26):

- Відставання від графіку розробки.
- Затримки платежів за реалізовану розробку.
- Втрата всіх напрацювань впродовж розробки.
- Недостатні технічні навички членів команди.
- Погана комунікація між членами команди.
- Мобілізація членів команди.
- Вимкнення фізичних серверів через відсутність електрики.

Середнього впливу (13-19):

- Несумісність програмних компонентів з ОС.
- Втрата даних через несправність.
- Перевантаження мережі.
- Несумісність обладнання.
- Кримінальні та економічні злочини.
- Військові дії на території розробки та застосування.

Найменш впливові (9-12):

- Природні стихійні лиха.
- Введення необхідних соціальних обмежень.
- Нестача обчислювальної спроможності серверів/місця.
- Припинення підтримки зовнішніх бібліотек, які використовуються в проєкті.

4.2 Запобігання негативним наслідкам та контроль за прогресом

Попередження негативних наслідків дозволяє уникнути витрачання часу, грошей та інших ресурсів на вирішення проблем під час проєкту. Виявлення та управління ризиками на ранній стадії може заощадити значні ресурси в майбутньому. Успішне запобігання негативним наслідкам може зберегти репутацію проєктної команди та організації в цілому. Проактивне

управління ризиками також позитивно впливає на сприйняття проєкту зовнішніми зацікавленими сторонами та клієнтами.

Запобігання негативним результатам допомагає забезпечити стійкість проєкту. Це важливо для дотримання планів, бюджетів і графіків проєкту. Попередження проблем під час процесу розробки допомагає забезпечити якість продукту. Відстеження та усунення ризиків на ранніх стадіях процесу розробки може запобігти дефектам і підвищити задоволеність кінцевим продуктом.

Успішне управління ризиками дає зацікавленим сторонам впевненість у здатності команди проєкту ефективно вирішувати потенційні проблеми. Це може підвищити довіру і полегшити співпрацю зі сторонніми зацікавленими сторонами.

Відповідно до попереднього пункту, де були визначені найбільш ризикові події було обговорено і складено перелік протиризикових заходів (Додаток А) для них, фрагмент якого наведено в табл. 4.3.

Таблиця 4.3

Список протиризикових заходів

№	Ризикова подія	ПРЗ 1	Симптом (рання ознака)	ПРЗ 2	ПРЗ 3
		профілактика		при симптомі	при проблемі
1	**2**	**3**	**4**	**5**	**6**
1	Припинення підтримки зовнішніх бібліотек, які використовуються в проєкті	Моніторинг за бібліотеками, що використовуються у проєкті	Бібліотека стає deprecated, що свідчить про те, що бібліотека більше не оновлюється та скоро вона перестане підтримуватись	Аналіз та пошук аналогів	Сформувати список заходів, для переходу на іншу бібліотеку

Продовження табл. 4.3

1	2	3	4	5	6
2	Нестача обчислювальної спроможності серверів/місця	Систематичний перегляд статистики використання та залишку ресурсів	Лист, щодо ліміту використання	Сформувати рішення, щодо переходу на інші сервери із більшим лімітом, або збільшення ліміту споживання ресурсів поточного серверу	Реалізувати перехід на інший сервер або збільшити ліміт на поточному
4	Втрата даних через несправність	Регулярні резервні копії даних, Системний моніторинг	Втрата доступу до даних, Поява помилок	Ізолювання проблеми, Діагностика та ремонт	Забезпечення надійності апаратури, Постійний моніторинг
6	Відставання від графіку розробки	Регулярні мітинги на яких буде продемонстровано прогрес в розробці проєкту, регулярні ретроспективи для обговорення існуючих/потенційних проблем на проєкті	Учасники команду працюють понаднормово, щоб закінчити поставлені задачі.	Провести мітинг на якому спробувати знайти причини відставання від графіку і усунути їх.	Запропонувати оплачувані овертайми, Найняти додаткових спеціалістів

Таким чином, наявність деталізованого переліку протиризикових заходів сприятиме оптимізації процесу управління ризиками проєкту і відповідно загальну ефективність управління проєктом.

4.3 Управління якістю проєкту та забезпечення високої якості продукту

4.3.1 Управління якістю проєкту

Управління якістю проєкту та забезпечення високої якості продукту - це ключовий аспект успішного виконання будь-якого проєкту. Це включає в себе систематичний підхід до планування, контролю та забезпечення якості на всіх етапах проєкту. Забезпечення високої якості продукту передбачає не лише виявлення та виправлення дефектів, але й активну участь у всьому процесі

розробки, включаючи аналіз вимог, проєктування, розробку, тестування та впровадження. Управління якістю є необхідною складовою будь-якого успішного проєкту, оскільки воно дозволяє забезпечити відповідність продукту вимогам та очікуванням користувачів, зберігаючи відповідність з бюджетом та графіком проєкту.

На теперішній момент найбільш популярні підходи наступні:

1. Методологія Total Quality Management (TQM): Цей підхід ставить на перший план управління якістю на всіх рівнях організації. TQM орієнтований на постійне вдосконалення процесів, залучення всіх працівників до управління якістю та виробництво продукції, що відповідає вимогам споживачів.
2. Методологія Six Sigma: Six Sigma спрямована на виявлення та усунення дефектів у процесі виробництва чи послуг, що може призвести до покращення якості продукту та оптимізації бізнес-процесів.
3. Методологія Lean Management: Lean Management спрямована на мінімізацію витрат та оптимізацію продуктивності шляхом усунення зайвих операцій та втрат, що допомагає підвищити ефективність та якість продукту.
4. Керування якістю за стандартом ISO: Стандарти серії ISO (наприклад, ISO 9001) надають систематичний підхід до управління якістю, що включає процедури, вимоги та практики для забезпечення високої якості продукту або послуги.
5. Agile підходи до розробки програмного забезпечення: Agile методології (наприклад, Scrum, Kanban) спрямовані на невеликі ітеративні цикли розробки, тестування та звітування, що дозволяє швидко реагувати на зміни та підвищує якість продукту.
6. Continuous Integration/Continuous Deployment (CI/CD): Цей підхід передбачає автоматизовану процесу розгортання та тестування змін у коді, що дозволяє забезпечити стабільність та якість продукту на кожному етапі розробки.

Вибір конкретного підходу залежить від потреб проєкту, його характеристик, вимог клієнта та умов виробництва. Комбінація різних методів може бути також ефективним рішенням для забезпечення високої якості продукту.

Для проєкту розробки програмного забезпечення для процесів комерційної доставки вантажів може бути найбільш доцільним комбінований підхід, який об'єднує Agile методології з Lean Management та Continuous Integration/Continuous Deployment (CI/CD).

Основні причини вибору наступні:

1. Agile методології: Вони дозволять ефективно управляти змінами у вимогах та швидко адаптуватися до нових умов або вимог з боку клієнта.
2. Lean Management: Методологія Lean спрямована на ефективне використання ресурсів та мінімізацію витрат, що є важливим для оптимізації процесів у логістичній компанії.
3. CI/CD: Continuous Integration та Continuous Deployment допоможуть забезпечити стабільність та якість продукту шляхом автоматизації процесу розгортання та тестування змін у коді.

Комбінація цих підходів дозволить забезпечити гнучкість, ефективність та високу якість продукту, що відповідає вимогам у сфері логістики та комерційної доставки вантажів, оскільки кожен із них відповідає за власну сферу діяльності.

1. Agile методології (Scrum):

- Застосування ітеративного підходу до розробки, що дозволяє швидко реагувати на зміни вимог та вносити корективи.
- Формування крос-функціональних команд, що сприяє комунікації та спільній роботі з різними аспектами проєкту.

2. Lean Management:

- Впровадження принципів Lean для оптимізації процесів розробки та управління проєктом.

– Акцент на видаленні зайвих операцій та мінімізації втрат, що дозволяє ефективно використовувати ресурси та прискорювати розробку.

3. Continuous Integration/Continuous Deployment (CI/CD):

– Впровадження автоматизованого процесу розгортання та тестування змін у коді, що забезпечує стабільність та надійність продукту.

– Швидка ітерація та постійне включення нового функціоналу в продукт за допомогою автоматизованих процесів.

Ця комбінація підходів дозволяє створити гнучке, ефективне та високоякісне програмне забезпечення, що відповідає потребам у сфері комерційної доставки вантажів. Вона підтримує постійну адаптацію до змін, мінімізацію втрат та швидке впровадження нового функціоналу.

4.3.2 Забезпечення якості продукту

Юніт-тести – це метод тестування програмного забезпечення, який спрямований на перевірку окремих компонентів (юнітів) програми. Вони зазвичай проводяться на ранніх етапах розробки, коли код ще не об'єднаний в одну програму. Ці тести перевіряють правильність роботи окремих функцій, методів чи класів без урахування їх взаємодії з іншими частинами програми.

Важливою перевагою юніт-тестів є їх можливість виявлення помилок на ранніх етапах розробки, коли виправлення виявлених проблем є менш витратним. Вони також сприяють покращенню структури програмного коду, оскільки змушують розробників писати більш модульний код. Юніт-тести забезпечують незалежність від середовища виконання програми, оскільки вони тестують окремі функції без їх фактичного запуску. Завдяки їм можна автоматизувати процес тестування, що дозволяє швидше виявляти та виправляти помилки під час розробки. У кінцевому результаті, використання юніт-тестів сприяє підвищенню якості програмного продукту шляхом покращення стійкості до помилок та забезпечення коректності функціонування окремих компонентів.

Відповідно під час розробки було також впроваджено покриття юніт-тестами програми, та проведено тестування всієї системи. Результати наведено нижче (рис. 4.1):

File		Statements		Branches		Functions		Lines	
src		100%	3/3	100%	0/0	100%	0/0	100%	3/3
src/app		100%	36/36	100%	23/23	100%	10/10	100%	31/31
src/app/Services		100%	76/76	100%	8/8	100%	27/27	100%	69/69
src/app/home-page		100%	84/84	100%	12/12	100%	42/42	100%	81/81
src/app/load-dialog		53.33%	8/15	76.67%	23/30	100%	7/7	50%	7/14
src/app/login-and-registration		100%	17/17	100%	5/5	100%	8/8	100%	51/51
src/app/update-truck		47.37%	9/19	25%	1/4	25%	2/8	44.44%	8/18
src/environments		100%	1/1	100%	0/0	100%	0/0	100%	1/1

Рис. 4.1. Покриття коду юніт тестами

Як можна побачити на рисунку вище компонент login-and-registration повністю покритий юніт тестами, а саме:

- 51/51 ліній коду
- 17/17 виразів
- 5/5 розгалужень
- 8/8 функцій

Цей компонент відповідає за форми логіну та реєстрації, що реалізовувася у першу чергу а також є дуже важливим для системи, оскільки працює з особистими даними користувачів, і їх витік значно погіршить репутацію проєкту.

4.4. Аналіз розробленого продукту

Розроблена веб-система допомагає налагодити комунікацію між клієнтом та особою, яка займається перевезеннями, та автоматизувати цей процес.

Існує два основних сценарії поведінки користувача:

1. замовлення перевезення вантажу.

2. виконання вантажного перевезення.

Для оформлення замовлення необхідно виконати наступні кроки:

1. Ввійти у особистий кабінет як замовник.

Для цього необхідно скористатись однією із наступних форм, для нових користувачів необхідно у полі «Your role» вибрати «Shipper»

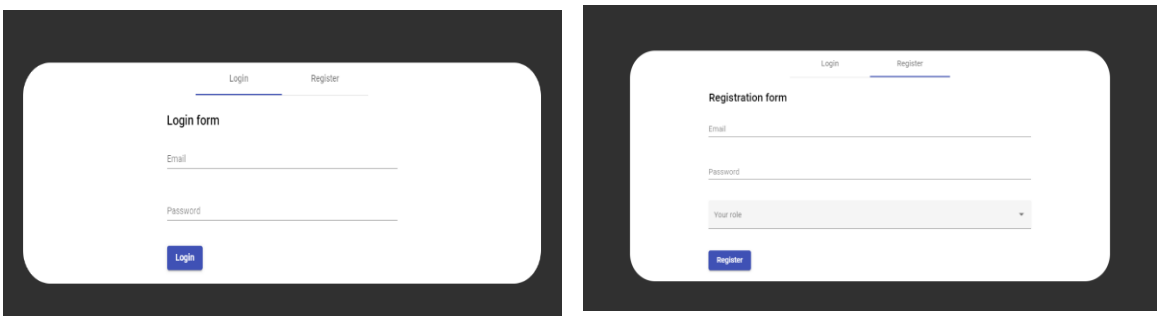


Рис. 4.2. Форми логіну та реєстрації

2. Перейдіть на сторінку створення замовлення, введіть всі необхідні дані і натисніть кнопку «Save the load», щоб зберегти його як чернетку для подальшої публікації.

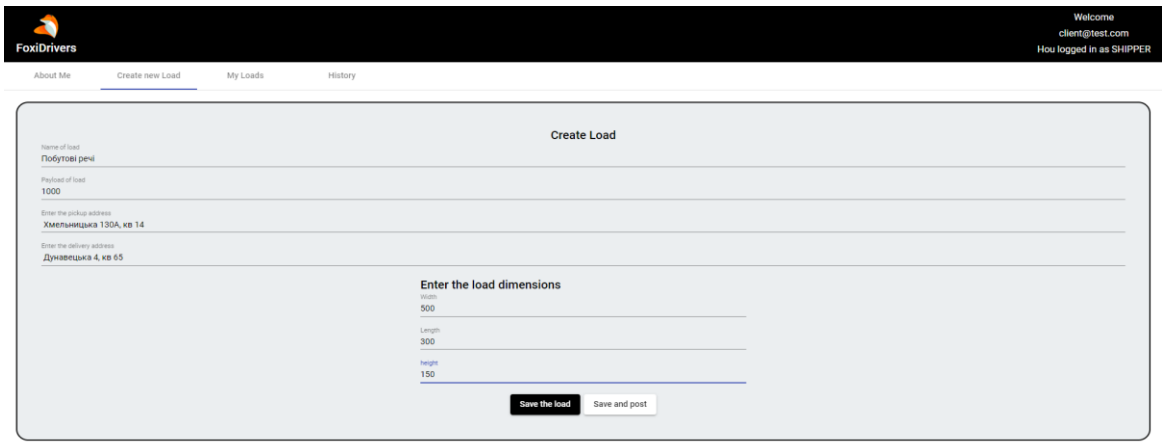


Рис.4.3. Приклад заповненої форми завмовлення

3. У разі успішного призначення водія відображається наступне повідомлення, а статус замовлення вважається «в роботі».

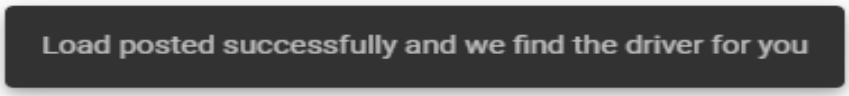


Рис. 4.4. Повідомлення про успішне назначення водія

4. Після завершення перевезення, замовлення автоматично зберігається в архіві і закривається.

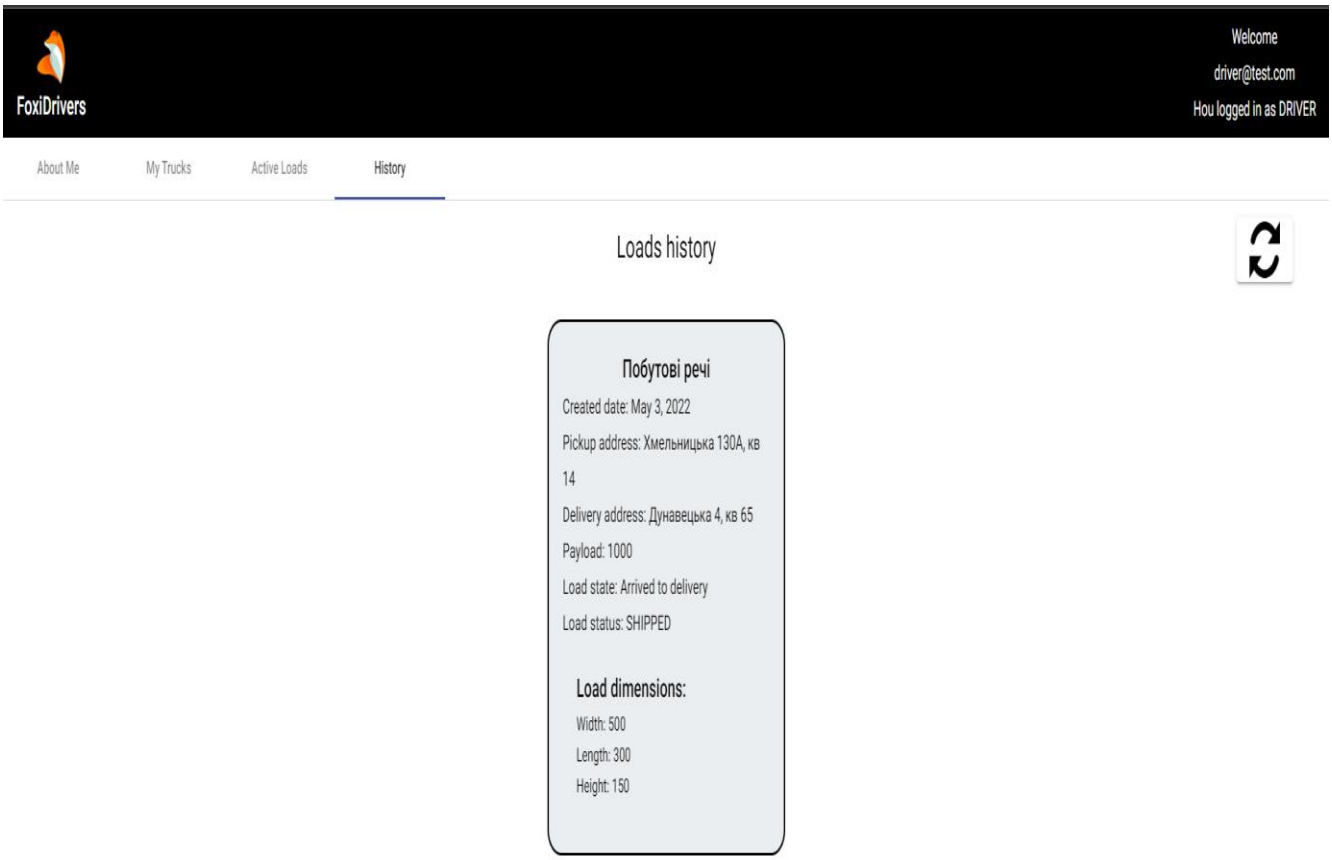


Рис. 4.5. Сторінка архіву

Це лише один із можливих сценаріїв взаємодії корисувачів системи, нижче таведено схему із можливих дій, що були імплементовані у початкову версію системи:

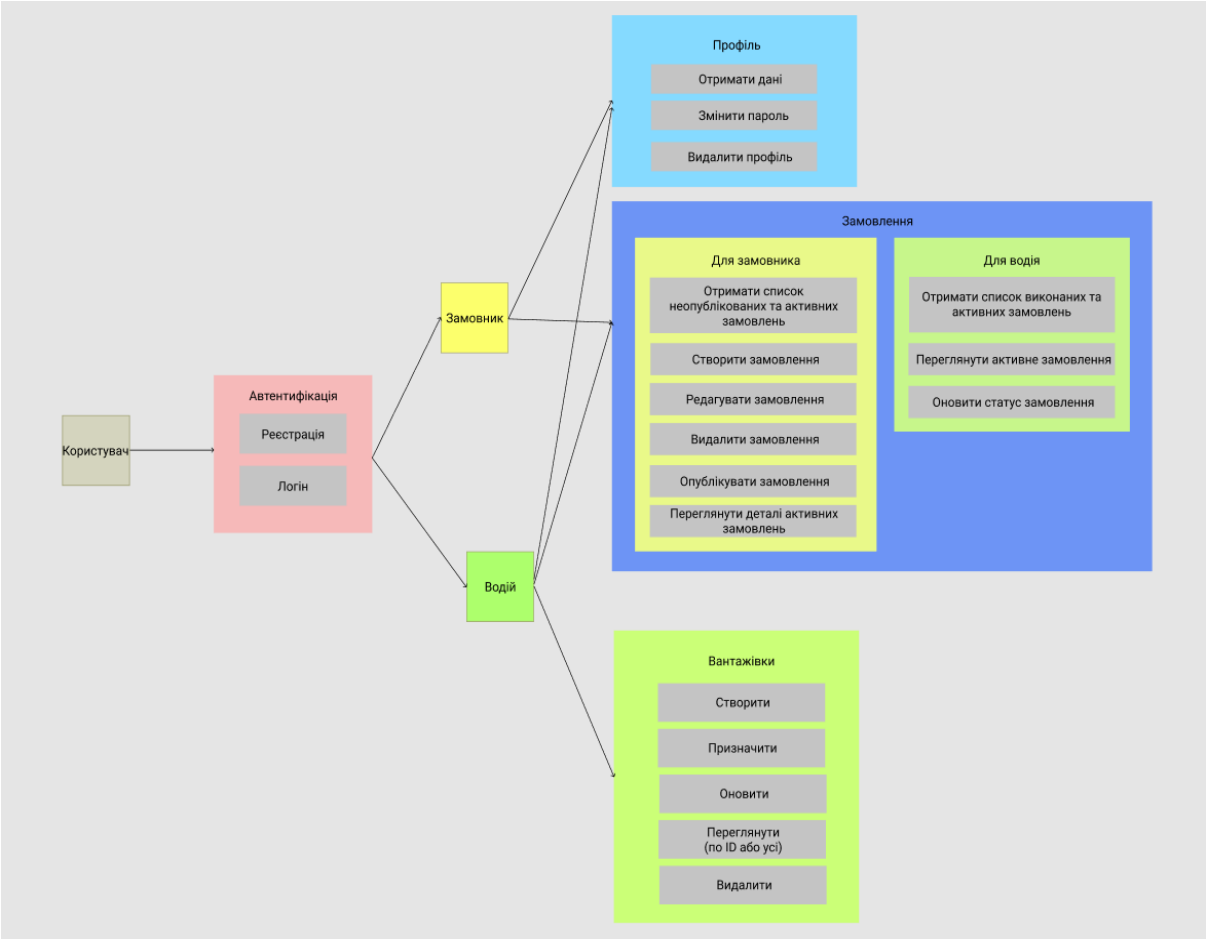


Рис. 4.6. Схема взаємодії із користувачами системи

Таким чином, на основі (Рис. 4.6) ми можемо побачити основні дії та можливості користувачів системи, що були імплементовані під час розробки початкової версії системи. Зокрема, цей рисунок відображає ключові функціональні області, такі як управління замовленнями, відстеження вантажівок, процес автентифікації та інші критичні операції. Крім того, на рисунку представлена поверхнева архітектура програми, що демонструє взаємодію між основними модулями системи. Це дозволяє отримати цілісне уявлення про структуру програмного забезпечення і взаємозв'язки між його компонентами, що є важливим для подальшого розвитку та вдосконалення системи.

ВИСНОВКИ

Проведене дослідження в рамках написання кваліфікаційної роботи, яке присвячене методам управління проєктом розробки програмного забезпечення для оптимізації процесів комерційної доставки вантажів, має надзвичайно велике значення в контексті сучасного світу, де швидкість, точність та надійність у доставці товарів є невід'ємними вимогами споживачів. Зростаюча конкуренція та глобалізація економіки роблять ефективне управління проєктами важливим для підприємств, що займаються комерційною доставкою.

Проведений аналіз предметної галузі дозволив глибше зрозуміти потреби та вимоги користувачів, а також ідентифікувати можливі ризики та перешкоди, які можуть виникнути протягом реалізації проєкту. Визначення мети, цілей та продукту проєкту дало чітке розуміння того, що потрібно досягти і які результати має надати проєкт на завершення. SWOT-аналіз сприяв ідентифікації сильних та слабких сторін, а також можливостей та загроз, що виникають в контексті реалізації проєкту.

Розробка концепції проєкту стала фундаментом для подальшої роботи, визначаючи загальну стратегію та напрямки діяльності. Велике значення має розробка програмного забезпечення, яка вимагає чіткої організаційної структури команди, а також ретельного планування модулів, бази даних та інших компонентів. Гнучкі технології створення продукту, такі як Scrum, дозволяють ефективно керувати процесами розробки, забезпечуючи гнучкість та адаптивність у відповіді на зміни.

Розроблені концептуальна та математична моделі управління проєктом розробки програмного забезпечення для процесів комерційної доставки вантажів, особливістю яких є оптимізація процесів визначення матеріальних та програмних ресурсів сприятимуть якісному підвищенню ефективності організації даних процесів. Дані моделі мають як наукову та практичну цінність.

Організоване календарне планування, контроль за прогресом та відхиленнями, а також управління ризиками та якістю є необхідними компонентами успішного завершення проєкту. Аналіз розробленого продукту відображає його поточний стан, а також можливості для подальшого вдосконалення та розвитку. Висновки роботи підкреслюють, що комплексний підхід до управління проєктом та його технологічною складовою є ключем до успіху та задоволення потреб зацікавлених сторін.

У завершальному аналізі дипломної роботи можна підсумувати, що важливість правильного планування та управління проєктом розробки програмного забезпечення для процесів комерційної доставки вантажів є дуже важливою в успішності реалізації проєкту. Досліджено, що використання аналітичних методів та гнучких технологій, таких як Scrum, виявляється ключовим для досягнення успіху та створення високоякісного продукту. Проведений аналіз предметної галузі, визначення мети та цілей проєкту, а також SWOT-аналіз показалися незамінними кроками для забезпечення ефективності та успішного завершення проєкту.

Подальше управління ризиками, відхиленнями від графіків та контроль за прогресом виявилися ключовими аспектами успішного завершення проєкту. Аналіз розробленого продукту дозволяє виявити його переваги та недоліки для подальшого вдосконалення. В цілому, грамотне впровадження зазначених методів та стратегій управління дозволяє не лише ефективно керувати процесом, але і забезпечує успішний результат, що відповідає потребам та очікуванням зацікавлених сторін.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Прогресивні веб застосунки [Електронний ресурс] // – 2024. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
2. Створення AMP сторінок [Електронний ресурс] // – 2024. – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/amp-technology.html
3. 15 провідних тенденцій веб розробки у 2020 році [Електронний ресурс] // – 2020р. - Режим доступу до ресурсу: https://web4u.in.ua/blog/15-prov-dnih-tendenc-y-veb-rozrobki-u-2020-roc-31
4. Офіційний сайт компанії JetBrains [Електронний ресурс] // – 2024. – Режим доступу до ресурсу: https://www.jetbrains.com/
5. Графічний редактор Figma [Електронний ресурс] // – 2024. – Режим доступу до ресурсу: https://www.figma.com/
6. Що таке NPM ? [Електронний ресурс] // – 2024. – Режим доступу до ресурсу: https://ivanmalaniak.pp.ua/webdev-blog/nodejs/shcho-take-npm/
7. Бібліотека Karma [Електронний ресурс] // - 2024. – Режим доступу до ресурсу: https://karma-runner.github.io/latest/index.html \
8. Клієнт-серверна Архітектура [Електронний ресурс] // – 2024. – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/
9. Офіційний сайт фреймворку Angular [Електронний ресурс] // – 2024. – Режим доступу до ресурсу: https://angular.io/
10. Adam Friman. Pro Angular 9: Build Powerful and Dynamic Web Apps – 2020. – 809 p.
11. Anton Moiseev & Yakov Fain Angular Development with TypeScript – 2018. – 1288 p.
12. Nate Murray, Ari Lerner, Felipe Coury, Carlos Taborda ng-book 2: The Complete Book on Angular 2 (Volume 2) – 2016. – 626 p.
13. Яновицька А.В. Договір міжнародного перевезення вантажів автомобільним транспортом – 2019. – 180 с.

14. Розробка математичної моделі [Електронний ресурс] // – 2023. – Режим доступу до ресурсу: https://stud.com.ua/162464/prirodoznavstvo/rozrobka_matematichnoyi_modeli
15. М.С. Юхимчук і А.І. Деркач, Розробка математичних моделей аналізу впливу параметричних збурень на стійкість автоматичних систем з логічними управляючими пристроями», вісник впі, вип. 2, с. 145–151, трав. 2016.
16. «Introduction to Mathematical Modeling» by Edward A. Bender and Suzanne Feldman – 2000. – 272 p.
17. «Mathematical Modeling: Models, Analysis and Applications» by Sandip Banerjee – 2021. – 434 p.
18. «Mathematical Modeling and Simulation: Introduction for Scientists and Engineers» by Kai Velten – 2009. – 362 p.
19. «Principles of Mathematical Modeling» by Clive Dym and Elizabeth Dieter – 2004. – 303 p.
20. «Mathematical Modeling: A Chemical Engineer's Perspective» by Rutherford Aris – 1999. – 479 p.
21. The Ultimate Guide to Business Strategy: Leveraging SWOT Analysis for Optimal Success by How2Become – 2023. – 102 p.
22. Problem Solver: Maximizing Your Strengths to Make Better Decisions by Cheryl Strauss Einhorn – 2023. – 192 p.
23. The SWOT Analysis Kindle Edition by Lawrence Fine – 2011p. – 80 p.
24. «A Guide to the Project Management Body of Knowledge (PMBOK® Guide)» by Project Management Institute (PMI) – 2017. – 756 p.
25. «Project Management for Dummies» by Stanley E. Portny – 2017p. – 448 p.
26. «Scrum: The Art of Doing Twice the Work in Half the Time» by Jeff Sutherland – 2014. – 256 p.
27. «The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses» by Eric Ries – 2011. – 336 p.

28.«The Agile Samurai: How Agile Masters Deliver Great Software» by Jonathan Rasmusson – 2014. – 384 p.
29.«The Mythical Man-Month: Essays on Software Engineering» by Frederick P. Brooks Jr. – 1995. – 322 p.
30.«The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail» by Clayton M. Christensen – 2016. – 336 p.
31.«The Lean Six Sigma Pocket Toolbook: A Quick Reference Guide to 100 Tools for Improving Quality and Speed» by Michael L. George – 2005. – 282 p.
32.«Leadership: Theory and Practice» by Peter G. Northouse – 2015. – 512 p.
33.«Agile Estimating and Planning» by Mike Cohn – 2005. – 368 p.
34.«The Art of Project Management» by Scott Berkun – 2005. – 392 p.
35.«Managing Successful Projects with PRINCE2» by Axelos – 2017. – 398 p.
36.«Project Management: A Systems Approach to Planning, Scheduling, and Controlling» by Harold Kerzner – 2017. – 1104 p.
37.«The Project Manager's Guide to Mastering Agile: Principles and Practices for an Adaptive Approach» by Charles G. Cobb – 2015. – 352 p.

ДОДАТКИ

Додаток А

Таблиця А.1

Перелік протиризикових заходів

№	Ризикова подія	ПРЗ 1 профілактика	Симптом (рання ознака)	ПРЗ 2 при симптомі	ПРЗ 3 при проблемі
1	2	3	4	5	6
1	Припинення підтримки зовнішніх бібліотек, які використовуються в проєкті	Моніторинг за бібліотеками, що використовуються у проєкті	Бібліотека стає deprecated, що свідчить про те, що бібліотека більше не оновлюється та скоро вона перестане підтримуватись	Аналіз та пошук аналогів	Сформувати список заходів, для переходу на іншу бібліотеку
2	Нестача обчислювальної спроможності серверів/місця	Систематичний перегляд статистики використання та залишку ресурсів	Лист, щодо ліміту використання	Сформувати рішення, щодо переходу на інші сервери із більшим лімітом, або збільшення ліміту споживання ресурсів поточного серверу	Реалізувати перехід на інший сервер або збільшити ліміт на поточному
3	Вимкнення фізичних серверів через відсутність електрики	Встановлення безперебійного живлення, Регулярне створення резервних копій даних	Відсутність доступу до серверів, Переривання роботи сервісів (додатку)	Перевірка статусу живлення, Часткове використання резервного живлення	Розгортання додаткового резервного живлення
4	Втрата даних через несправність	Регулярні резервні копії даних, Системний моніторинг	Втрата доступу до даних, Поява помилок	Ізолювання проблеми, Діагностика та ремонт	Забезпечення надійності апаратури, Постійний моніторинг
5	Погана комунікація між членами команди	Проводити ретроспективу раз на спринт, Проводити регулярно F2F мітинги	Негативні відгуки членів команди одне про одного, відставання графіку розробки	Провести мітинги між учасниками конфлікти. Спробувати вирішити конфлікт	Вивести учасників команди через яких виникали конфлікти

6	Відставання від графіку розробки	Регулярні мітинги на яких буде продемонстровано прогрес в розробці проєкту, регулярні ретроспективи для обговорення існуючих/потенційних проблем на проєкті	Учасники команду працюють понаднормово, щоб закінчити поставлені задачі.	Провести мітинг на якому спробувати знайти причини відставання від графіку і усунути їх.	Запропонувати оплачувані овертайми, Найняти додаткових спеціалістів
7	Затримки платежів за реалізовану розробку	На початку проєкту юридично затвердити пункти, які стосуються оплати. Моніторити стан проєкту, його розвиток	Мінімальна затримка виплат з обіцянками наступного разу	Підготувати документи/ команду до можливого ходу дій. Дати їм можливість вибору	Затримка проєкту Зсилання на юридичну домовленість. Припинення роботи працівників
8	Кримінальні та економічні злочини(перевірки та затримки роботи проєкту)	Мати прозору та чисту документацію, впевнитись, що немає проблем з усіма зацікавленими сторонами	Починаються обшуки офісів схожих локацій, попередження від замовника	Провести бесіду з командою, мати всі необхідні документи, які свідчать про чисту діяльність	Мати резервні копії напрацювань, створити графік при якому працівник можуть працювати дистанційно. Підготувати документи
9	Введення необхідних соціальних обмежень(пандемія)	Підготувати людей до дистанційної роботи, моніторити ситуації на інших проєктах	Поступовий перехід усіх діяльностей на карантин	Попередити всю команду проєкту, оцінити стан команди та можливості	Сформувати новий графік, перевести команду в дістанційний стан, моніторити ситуацію
10	Втрата всіх напрацювань впродовж розробки	Створити необхідні резервні копії та налаштувати зберігання в захищеному місці	Недостатньо даних про минулі розробки на проєкті	Оцінити чи є можливість повернути дані чи хоча б частину, провести розмову з командою, підготувати до плотного графіку	Затримка проєкта, можлива зміна працівників/проєктного менеджера. Відбудова по ранім розробкам

Додаток Б

Лістинг Back-end частини

index.js

```
const express = require(«express»);
const mongoose = require(«mongoose»);
const cors = require(«cors»);
const app = express();

const authRouter = require(«./routers/authRouter»);
const userRouter = require(«./routers/userRouter»);
const truckRouter = require('./routers/truckRouter');
const loadRouter = require('./routers/loadRouter');

mongoose.connect(`mongodb+srv://user:12345@cluster0.hnyca.mongodb.net/NODEJS_HW3?retryWrites=true&w=majority`, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useFindAndModify: false,
    useCreateIndex: true
});
app.use(cors());
app.use(express.json());

app.use(«/api»,authRouter);
app.use(«/api»,userRouter);
app.use(«/api»,truckRouter);
app.use(«/api»,loadRouter);
```

```
app.use(express.static('dist/client'));
app.get('/*', (req, res) => {
  res.sendFile(`index.html`, {root: `dist/client/`});
});

app.listen(8080||process.env.PORT , () => console.log(«listening»));
controllers => authController

const jwt = require('jsonwebtoken');
const User = require('../models/user');
const RegistrationCredentials = require('../models/registrationCredentials');
const {secret} = require('../configs/auth');


module.exports.register = (request, response) => {
    const {email, password, role} = request.body;
    const createdDate = new Date();
    const user = new User({email, createdDate});
    const regCredential = new RegistrationCredentials({email, password,role});
    regCredential.save()
        .then(() => {
            if (!email || !password || !role) {
                return  response.status(400).json({massage: «Bad request»});
            }
            user.save()
                .catch((err) => {
                    return  response.status(500).json({massage: «Server 2 error», err});
                });
```

```
            return  response.json({massage: 'Success'});
        })
        .catch((e) => {
            return  response.status(500).json({massage: «Server 1 error», e});
        });
    }

    module.exports.login = async (request, response) => {

        const {email, password} = request.body;
        let user;
         await User.findOne({email}).exec()
            .then(selectedUser =>{
                user = selectedUser;
            })
            .catch((err) => response.status(500).json({massage: err} ));

        await RegistrationCredentials.findOne({email, password}).exec()
            .then(selectedUser => {
                if (!selectedUser) {
                    return response.status(400).json({massage: «Wrong email or password»});
                }

                let jwtObj = {
                    regCred: {
                        _id : selectedUser._id,
                        role: selectedUser.role,
                        password: selectedUser.password,
                        email: selectedUser.email
```

```
                },
                user:{
                    _id: user._id,
                    createdDate: user.createdDate,
                    email: user.email
                }
            }
            return response.status(200).json({massage: 'success', token: jwt.sign(JSON.stringify(jwtObj), secret)});
        })
        .catch((err) => {
            return  response.status(500).json({massage: err + «SSS»});
        });
    }
    middleware => authMiddleware
    const jwt = require('jsonwebtoken');
    const { secret } = require('../configs/auth');

    module.exports = (request, response, next) => {
      const authHeader = request.headers['authorization'];

        if(!authHeader) {
            return response.status(400).json({status: 'No authorization header found'});
        }
        const [, jwtToken] = authHeader.split(' ');
        try {
            request.token = jwt.verify(jwtToken, secret);
            next();
        } catch (err) {
```

```
        return response.status(400).json({status: 'Invalid JWT'});
    }
};
models => registrationCredentials
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
module.exports = mongoose.model('registrationCredentials', new Schema({
    email: {
        required: true,
        type: String,
        unique: true
    },
    password: {
        required: true,
        type: String
    },
    role: {
        required: true,
        type: String,
        enum: [ «SHIPPER», «DRIVER» ]
    }
}));
routers=>authRouter
const express = require('express');
const router = express.Router();
const { register, login } = require(«../controllers/authController»);
router.post('/auth/register', register);
router.post('/auth/login', login);
```

Додаток В

Лістинг Front-end частини(app.module.ts)

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { LoginAndRegistrationComponent } from './login-and-registration/login-and-registration.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import {MatTabsModule} from '@angular/material/tabs';
import {FormsModule, ReactiveFormsModule} from '@angular/forms';
import {MatFormFieldModule} from '@angular/material/form-field';
import {MatInputModule} from '@angular/material/input';
import {MatButtonModule} from '@angular/material/button';
import {MatSelectModule} from '@angular/material/select';
import { HttpClientModule } from '@angular/common/http';
import { HomePageComponent } from './home-page/home-page.component';
import {MatTooltipModule} from '@angular/material/tooltip';
import {MatSnackBarModule} from '@angular/material/snack-bar';
import {MatDialogModule} from '@angular/material/dialog';
import { LoadDialogComponent } from './load-dialog/load-dialog.component';
import { TruckStatusPipe } from './truck-status.pipe';
import { UpdateTruckComponent } from './update-truck/update-truck.component';
import { LoadPipe } from './load.pipe';
@NgModule({
  declarations: [
    AppComponent,
    LoginAndRegistrationComponent,
```

```
    HomePageComponent,
    LoadDialogComponent,
    TruckStatusPipe,
    UpdateTruckComponent,
    LoadPipe
  ],
    imports: [
        BrowserModule,
        BrowserAnimationsModule,
        MatTabsModule,
        ReactiveFormsModule,
        MatFormFieldModule,
        MatInputModule,
        MatButtonModule,
        MatSelectModule,
        HttpClientModule,
        MatTooltipModule,
        MatSnackBarModule,
        MatDialogModule,
        FormsModule
    ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```