МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

_____

(підпис)

_____

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційне та програмне забезпечення месенджерового боту для ведення бізнесу.»
здобувача групи ІН - 03 Костенка Олексія Володимировича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Олексій КОСТЕНКО

(підпис)

Керівник, кандидат технічних наук
доцент кафедри комп'ютерних наук Ігор ШЕЛЕХОВ _____

(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-03 Костенка Олексія Володимировича

1. Тема роботи: «Інформаційне та програмне забезпечення месенджерового боту для ведення бізнесу»
затверджую наказом по СумДУ від *«» червня 2024 р.*
2. Термін здачі здобувачем кваліфікаційної роботи *до червня 2024 р.*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Розробка ТЗ. 2) Аналіз варіантів використання та вимог до системи . 3) Аналіз можливих тестів системи Тестові сценарії системи.4) Реалізація функціоналу застосування Застосування, яке має основний функціонал та може бути протестоване. 5) Тестування застосунку, виправлення помилок, відладка*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «___» _____ ____ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз варіантів використання та вимог до системи Діаграма варіантів використання, таблиця функціональних вимог*	06-08.05.2024	
2	*Аналіз можливих тестів системи Тестові сценарії системи*	09-11.05.2024	
3	*Реалізація функціоналу застосування Застосування, яке має основний функціонал та може бути протестоване*	12-14.05.2024	
4	*Тестування застосунку, виправлення помилок, відладка Готове до використання застосування*	15-17.05.2024	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	18-20.05.2024	

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 56 сторінок., 12 рисунки, 1 таблиця, 2 додаток, 57 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є дуже актуальною, адже сучасний світ дедалі більше залежить від інтернету. Чат-боти стають невід'ємною частиною розвитку B2C-бізнесу, відкриваючи широкі можливості для розширення сфери онлайн-продажів.

Об’єкт дослідження – Телеграм бот для онлайн продажів

Мета роботи – розробка боту на платформі Телеграм для онлайн продажів автомобілів.

Методи дослідження – ознайомлення з програмними засобами та інструментами для розробки Телеграм боту. Розробка дизайну та функціоналу, що буде оптимально працювати, і бути зрозумілим усім клієнтам.

Результати – розроблено Телеграм бот, який задовольняє потреби користувачів, надаючи унікальний користувацький досвід.

TELEGRAM БОТ, PYTHON

ЗМІСТ

ВСТУП

Актуальність. У сучасному світі швидкого розвитку цифрових технологій і посилення інтеграції інтернет-сервісів у повсякденне життя людини, значно зросла потреба у зручних, ефективних та інноваційних цифрових інструментах. Одним із напрямків, де ці потреби є особливо актуальними, є сфера продажу автомобілів. Використання мессенджерів, зокрема Telegram, для комерційних цілей відкриває нові можливості для бізнесу, надаючи їм інструмент для швидкого та зручного зв'язку з клієнтами. Сьогодні ринок автомобілів в Україні характеризується високою конкуренцією та різноманітністю пропозицій. В таких умовах важливо не тільки надати потенційному покупцеві широкий вибір можливостей, але й забезпечити високий рівень сервісу та зручності при виборі та придбанні автомобіля. У цьому контексті розробка телеграм-бота для автомобільної платформи стає вагомим кроком у напрямку цифрової трансформації та підвищення якості обслуговування клієнтів.

Об’єкт дослідження. Процес розробки інформаційного та програмного забезпечення телеграм-бота, який не тільки дозволить користувачам зручно переглядати каталог автомобілів, отримувати інформацію про специфікації та ціни, але й забезпечить можливість замовлення автомобіля та отримання консультацій від представників компанії.

Предмет дослідження. Новітні інструменти для реалізації чат-боту.

Гіпотеза. Створення суто рекламних чат-ботів вже не є актуальним рішенням, яке принесе бренду розвиток. Навпаки, на власному досвіді було доведено, що чат-боти тісно пов’язані з маркетингом. Саме цей взаємозв’язок допомагає побудувати якісну та унікальну комунікацію із споживачами ботів. Якісна комунікація досягається шляхом розуміння стратегій проєкту, його основної цілі та бажаного результату.

Новизна. Цей проект включає в себе аналіз існуючого ринку, визначення ключових вимог до функціональності бота, розробку програмного забезпечення

на основі сучасних технологій, а також тестування та впровадження готового рішення. Цілісний підхід до вирішення задачі, використання інноваційних технологій і орієнтація на потреби користувачів дозволять створити ефективний інструмент, який зможе стати невід'ємною частиною сучасного автомобільного ринку України.

Структура. Дана робота складається зі вступу, аналізу предметної області, постановки задачі, проєктування бази даних, розробки програмного продукту, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальний огляд предметної області

Автосалон або автодилер — це підприємство, яке продає нові або вживані автомобілі на роздрібному рівні на основі дилерського контракту з автовиробником або його дочірньою компанією з продажу. Автосалони також часто продають запчастини та послуги з технічного обслуговування автомобілів.

У Сполучених Штатах автосалони історично були важливим джерелом державних і місцевих податків з продажу. Вони мають значний політичний вплив і лобіюють нормативні акти, які гарантують їх виживання та прибутковість. До 2010 року в усіх штатах США діяли закони, які забороняли виробникам обходити незалежні автосалони та продавати автомобілі безпосередньо споживачам. До 2009 року більшість штатів наклали обмеження на створення нових дилерських центрів, щоб конкурувати з діючими дилерськими центрами.

Економісти характеризують ці правила як форму пошуку ренти, яка витягує ренту з виробників автомобілів, збільшує витрати для споживачів і обмежує входження нових автосалонів, одночасно збільшуючи прибутки діючих автодилерів. Дослідження показують, що в результаті цих законів роздрібні ціни на автомобілі вищі, ніж були б за інших обставин.[1]

Перші автомобілі продавали автовиробники клієнтам безпосередньо або через різні канали, включаючи замовлення поштою, універмаги та подорожуючих представників.[1] Наприклад, компанія Sears зробила свою першу спробу продати висококолісну машину з ланцюговим приводом з бензиновим двигуном у 1908 році через свій каталог поштою, а починаючи з 1951 року Allstate через окремі магазини та каталог.[2][3]

Фред Коллер відкрив перший спеціальний автосалон у 1889 році. Відомий як Reading Automobile Company, Коллер продавав автомобілі, вироблені в Клівленді, Огайо, у своєму дилерському центрі в Редінгу, Пенсільванія.

Вважається, що це перший дилерський центр, присвячений виключно продажу автомобілів, тобто спочатку він не був створений для продажу кінних екіпажів.[4]

Сьогодні прямі продажі автовиробників споживачам обмежені більшістю штатів США через закони про франшизу, які вимагають, щоб нові автомобілі продавалися лише ліцензованими та незалежними дилерськими центрами. Першою жінкою-дилером автомобілів у Сполучених Штатах була Рейчел «Мама» Краус, яка в 1903 році відкрила свій бізнес, Krouse Motor Car Company, у Філадельфії, штат Пенсільванія.

Кількість автосалонів у США досягла піку в 1927 році в 53 125 і неухильно зменшувалася протягом наступних десятиліть. До 1960 року було 33 658 дилерських центрів; до 1980 року 23 379; а до 2001 року – 22 007.[1]

Автосалони зазвичай мають франшизу для продажу та обслуговування транспортних засобів певними компаніями. Вони часто розташовані на об’єктах, де є достатньо місця для розміщення виставкового залу, механічного обслуговування та ремонту кузовів, а також для зберігання вживаних і нових транспортних засобів. Багато дилерських центрів розташовані за містом або на околиці міських центрів. Прикладом традиційного автосалону, який є одним власником, є Collier Motors у Північній Каролініі.[7] Багато сучасних дилерських центрів тепер є частиною корпоративних мереж із сотнями місць.[8] Прибутки дилерів у США в основному надходять від обслуговування, частково від вживаних автомобілів і мало від нових автомобілів.[9]

Більшість виробників автомобілів перемістили фокус своїх франчайзингових роздрібних торговців на брендинг і технології. Нові або відремонтовані об’єкти повинні мати стандартний вигляд для своїх дилерських центрів і мати експертів з продукту для зв’язку з клієнтами.[10][11] Audi поекспериментувала з високотехнологічним салоном, який дозволяє клієнтам налаштовувати та переглядати автомобілі на цифрових екранах у масштабі

1:1.[12][13] На ринках, де це дозволено, Mercedes-Benz відкрив фірмові магазини в центрі міста.[14]

Tesla Motors відмовилася від дилерської моделі продажів на основі ідеї, що дилерські центри не пояснюють належним чином переваги своїх автомобілів, і вони не можуть покладатися на дилерські центри третіх сторін, щоб керувати своїми продажами.[15] Однак у Сполучених Штатах прямі продажі автомобілів виробником заборонені майже в кожному штаті законами про франшизу, які вимагають, щоб нові автомобілі продавалися лише дилерами.[16] У відповідь Tesla відкрила галереї в центрі міста, де потенційні клієнти можуть побачити автомобілі, які можна замовити лише онлайн.[17][18] Ці магазини були створені за мотивами Apple Store.[19] Модель Tesla була першою у своєму роді та дала їм унікальні переваги як новій автомобільній компанії.[20]

В економічній теорії автосалони можна охарактеризувати як франчайзі, а виробників автомобілів як франчайзерів. Франчайзингові відносини можуть бути вигідними для обох сторін, оскільки франчайзі може продавати якісний і привабливий продукт, а франчайзер може покладатися на франчайзі, щоб нести витрати на подальшому виробництві та використовувати свої місцеві відносини для продажу більше продуктів і послуг.[1]

Франчайзер може діяти оппортуністично, накладаючи обмеження та тягар на франчайзі після того, як останній поніс безповоротні витрати, такі як інвестування у фізичні активи та створення репутації серед клієнтів. Франчайзер може, наприклад, вимагати, щоб автомобілі продавалися за низькими цінами, а послуги надавалися за невелику компенсацію. З іншого боку, франчайзі може діяти оппортуністично, використовуючи свою місцеву монополію для надання неякісного обслуговування клієнтів, стягувати з клієнтів більше та перекладати ці невиправдано високі витрати на франчайзера.[1][21]

1.2 Огляд існуючих рішень

1. AutoRia Telegram Bot

Опис: AutoRia Telegram Bot — це розширення популярного українського онлайн-авторинку. Бот дозволяє шукати автомобілі, використовуючи різні фільтри, і надає детальну інформацію про кожен автомобіль. Переваги:

- Велика база автомобілів.
- Інтуїтивно зрозумілі фільтри для пошуку.
- Доступ до детальної інформації про кожне авто. Недоліки:
- Обмежена інтерактивність порівняно з повноцінними мобільними додатками або веб-сайтами.
- Може бути недостатньо зручним для користувачів, які не звикли до Telegram-інтерфейсу.

2. OLX.ua Bot

Опис: OLX.ua Bot дозволяє користувачам шукати автомобілі на популярному українському сайті оголошень OLX. Користувачі можуть налаштовувати фільтри пошуку та переглядати оголошення безпосередньо в Telegram. Переваги:

- Широкий вибір автомобілів з усієї України.
- Простота використання та зручний інтерфейс.
- Можливість відразу перейти на повне оголошення на сайті OLX. Недоліки:
- Відсутність додаткових сервісів, наприклад, перевірки історії автомобіля або оцінки його стану.
- Обмежені можливості для комунікації з продавцем безпосередньо через бота.

3. AutoSale Telegram Bot

Опис: AutoSale — це спеціалізований бот для Telegram, створений для пошуку та купівлі автомобілів в Україні. Бот пропонує кілька категорій авто та включає функції для порівняння цін і моделей. Переваги:

- Прямий фокус на продаж автомобілів.
- Налаштування сповіщень про нові оголошення.

- Можливість порівняння моделей і цін. Недоліки:
- Менша база автомобілів порівняно з великими майданчиками, такими як OLX або AutoRia.
- Інтерфейс може бути не таким зручним для деяких користувачів.

Ці рішення представляють широкий спектр можливостей, від загальних платформ онлайн-оголошень до спеціалізованих автомобільних ботів, кожне з яких має свої унікальні переваги та обмеження.

1.3 Постановка задачі

Мета проекту

Створити інтерактивний телеграм-бот для автомобільної платформи, який забезпечить користувачам зручний доступ до каталогу автомобілів, інформації про компанію, персонального кабінету з інформацією про статус замовлень, а також можливість зробити замовлення і запросити зворотний дзвінок.

Основні завдання

1. Аналіз ринку та існуючих рішень: Вивчення існуючих телеграм-ботів та онлайн-платформ у сфері автомобільних продажів.
2. Визначення функціональних вимог: Розробка переліку функціональних можливостей бота, включаючи каталог автомобілів, інформацію про компанію, персональний кабінет користувача та замовлення.
3. Технічна реалізація: Вибір технологій для розробки (наприклад, PyTelegramBotAPI для Telegram і SQLite для бази даних), написання коду.
4. Тестування: Перевірка роботи бота на помилки і недоліки.
5. Впровадження і підтримка: Розгортання бота, моніторинг його роботи і внесення необхідних змін або оновлень.

Очікувані результати

Готовий до використання телеграм-бот, який забезпечує ефективний і зручний інтерфейс для користувачів, які шукають автомобілі в Україні. Бот має

включати в себе повний набір визначених функцій і бути легким у використанні та навігації.

Стадії та етапи розробки застосування наведено у табл.1.1

Таблиця 1.1 – Стадії та етапи розробки

Етап	**Зміст робіт**	**Результат робіт**	**Термін виконання**
1	Розробка ТЗ	ТЗ	Березень 2024
2	Аналіз варіантів використання та вимог до системи	Діаграма варіантів використання, таблиця функціональних вимог	Березень 2024
3	Аналіз можливих тестів системи	Тестові сценарії системи	Березень 2024
4	Реалізація функціоналу застосування	Застосування, яке має основний функціонал та може бути протестоване	Травень 2024
5	Тестування застосунку, виправлення помилок, відладка	Готове до використання застосування	Травень 2024

2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ

2.1 Проектування функціоналу системи

Діаграма варіантів використання UML (Unified Modeling Language) є одним з ключових інструментів у процесі аналізу та проектування систем, який дозволяє візуалізувати функціональні вимоги та взаємодії між користувачами та системою. Ця діаграма є невід'ємною частиною UML, яка є стандартною нотацією для моделювання об'єктно-орієнтованих систем.

Значення та використання діаграми варіантів використання

1. Визначення вимог: Діаграма варіантів використання допомагає аналітикам та розробникам визначити функціональні вимоги до системи. Це спрощує процес визначення того, що система повинна робити, і як вона повинна взаємодіяти з користувачами.
2. Комунікація з зацікавленими сторонами: Через свою наглядність, діаграма варіантів використання є відмінним інструментом для спілкування з нетехнічними зацікавленими сторонами. Вона дозволяє всім учасникам проекту мати спільне розуміння функціональності системи.
3. Планування розробки: Використовуючи діаграму варіантів використання, команди розробників можуть ефективніше планувати процес розробки, визначаючи пріоритетність функцій та вимог.

Основні компоненти діаграми варіантів використання UML

1. Актори: Актори представляють користувачів системи або зовнішні системи, які взаємодіють з системою. Вони можуть бути людьми, іншими програмами або організаціями. В діаграмі актори зазвичай зображуються у вигляді фігурок людини.
2. Варіанти використання (Use Cases): Варіанти використання описують дії або послідовності дій, які система може виконувати у відповідь на вимоги акторів. Вони представляють основну функціональність системи і зображуються у вигляді овалів.

3. Зв'язки: Зв'язки на діаграмі варіантів використання визначають взаємозв'язки між акторами та варіантами використання. Вони можуть бути представлені лініями або стрілками, показуючи взаємодію.
4. Системні межі: Системні межі визначають контекст або обсяг системи, до якої належать варіанти використання. Це допомагає уточнити, які функції належать системі.

Типи зв'язків у діаграмі варіантів використання

1. Асоціації: Прості зв'язки, які показують, що актор бере участь у варіанті використання.
2. Включення (Include): Використовується для представлення випадків, коли один варіант використання завжди включає інший.
3. Розширення (Extend): Вказує на випадки, коли варіант використання може бути розширений або доповнений іншим варіантом використання.
4. Узагальнення: Використовується для групування подібних варіантів використання або акторів у загальні категорії.

Розробка діаграми варіантів використання

Розробка діаграми варіантів використання вимагає глибокого розуміння вимог до системи. Процес включає ідентифікацію акторів, збір варіантів використання, визначення взаємодій та моделювання цих елементів на діаграмі. Ключовим моментом є встановлення чітких та зрозумілих вимог для кожного варіанту використання.

Застосування діаграми варіантів використання

1. Планування проекту: Діаграма варіантів використання використовується для планування розробки проекту, визначення етапів роботи та розподілу ресурсів.
2. Аналіз та дизайн системи: Вона допомагає аналітикам та дизайнерам системи визначити потрібну функціональність і структуру системи.

3. Тестування та валідація: Використовується як основа для розробки тестових сценаріїв і валідації вимог до системи.

Діаграма варіантів використання UML є універсальним інструментом, що допомагає командам розробників визначити, планувати та реалізувати вимоги до системи. Вона відіграє ключову роль у спрощенні комунікації між різними учасниками проекту та забезпечує ефективне розуміння функціональності системи.

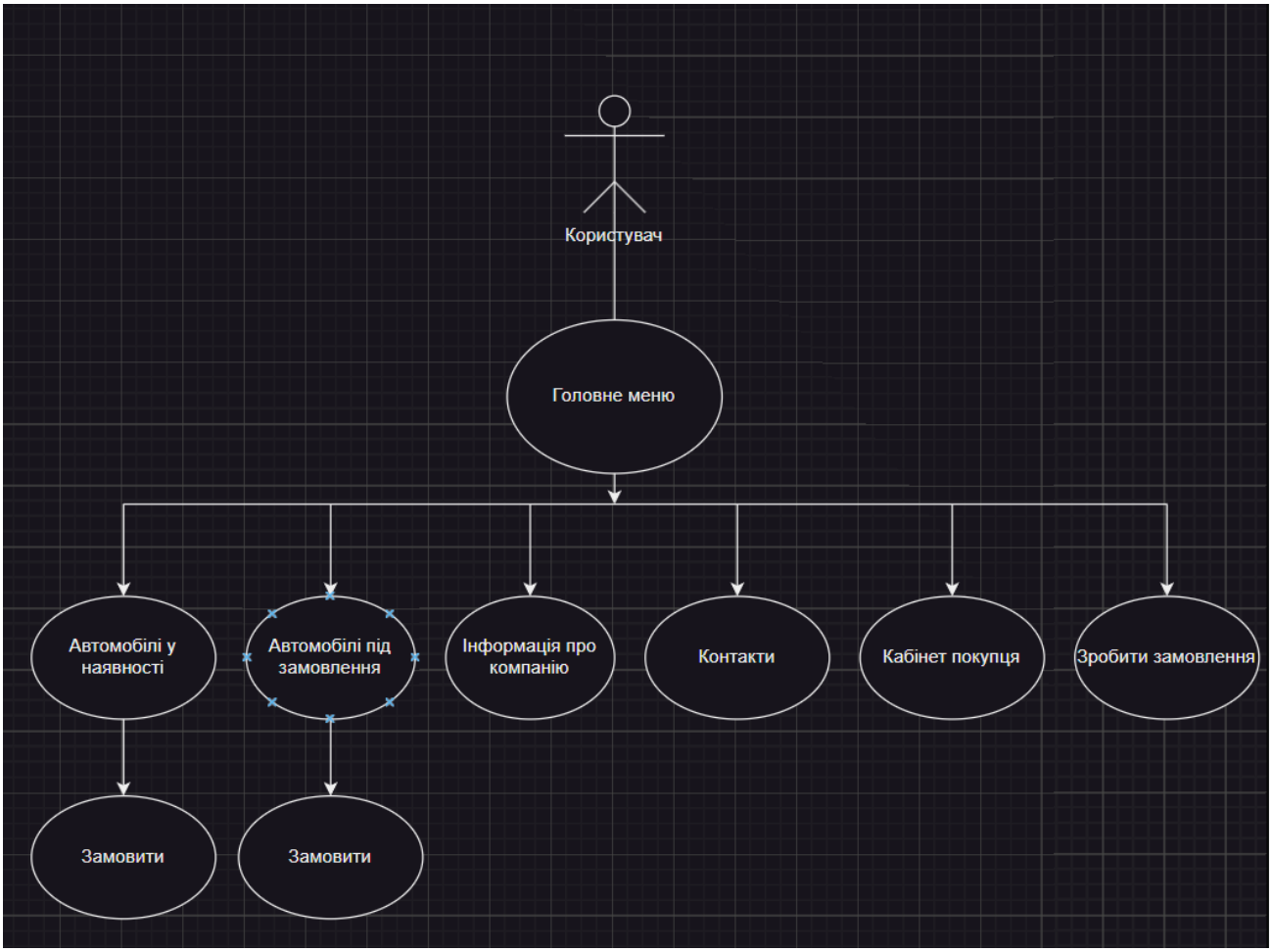

Рисунок 2.1 — Діаграма варіантів використання для користувача

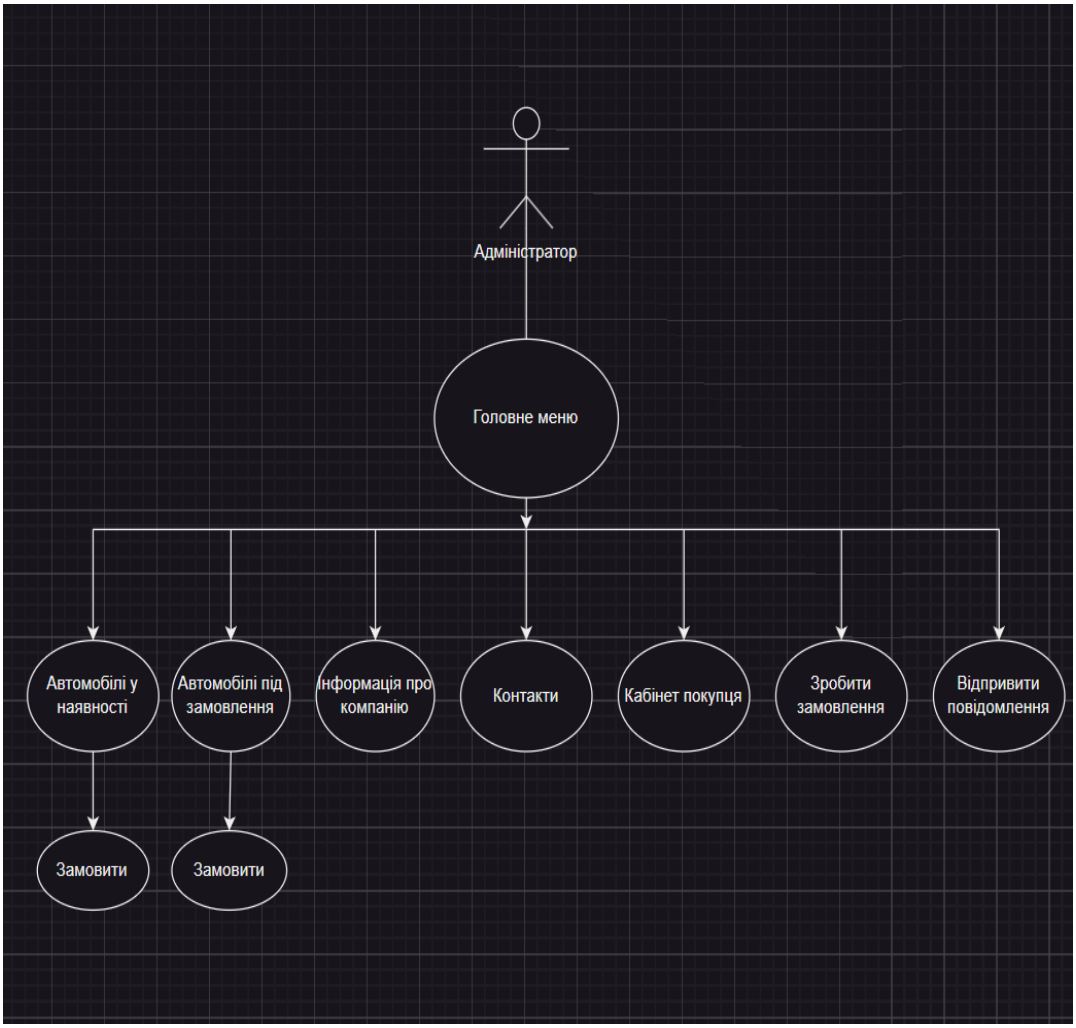

Рисунок 2.2 — Діаграма варіантів використання для адміністратора

2.2 Проектування внутрішньої будови

Під час розробки програмного забезпечення, зазвичай, зображення внутрішньої структури системи подається у формі діаграми груп та курсів, яка демонструє склад груп та модулів проекту та їх взаємодію.

У сфері програмної інженерії, діаграма груп курсів, згідно з уніфікованою мовою моделювання (UML), є статичною структурною діаграмою, що надає огляд структури системи, включаючи системні класи, їх атрибути, операції (або методи) та зв'язки між об'єктами.

На рисунку 2.3 зображено діаграму класів, яка відображає внутрішню будову проекту.

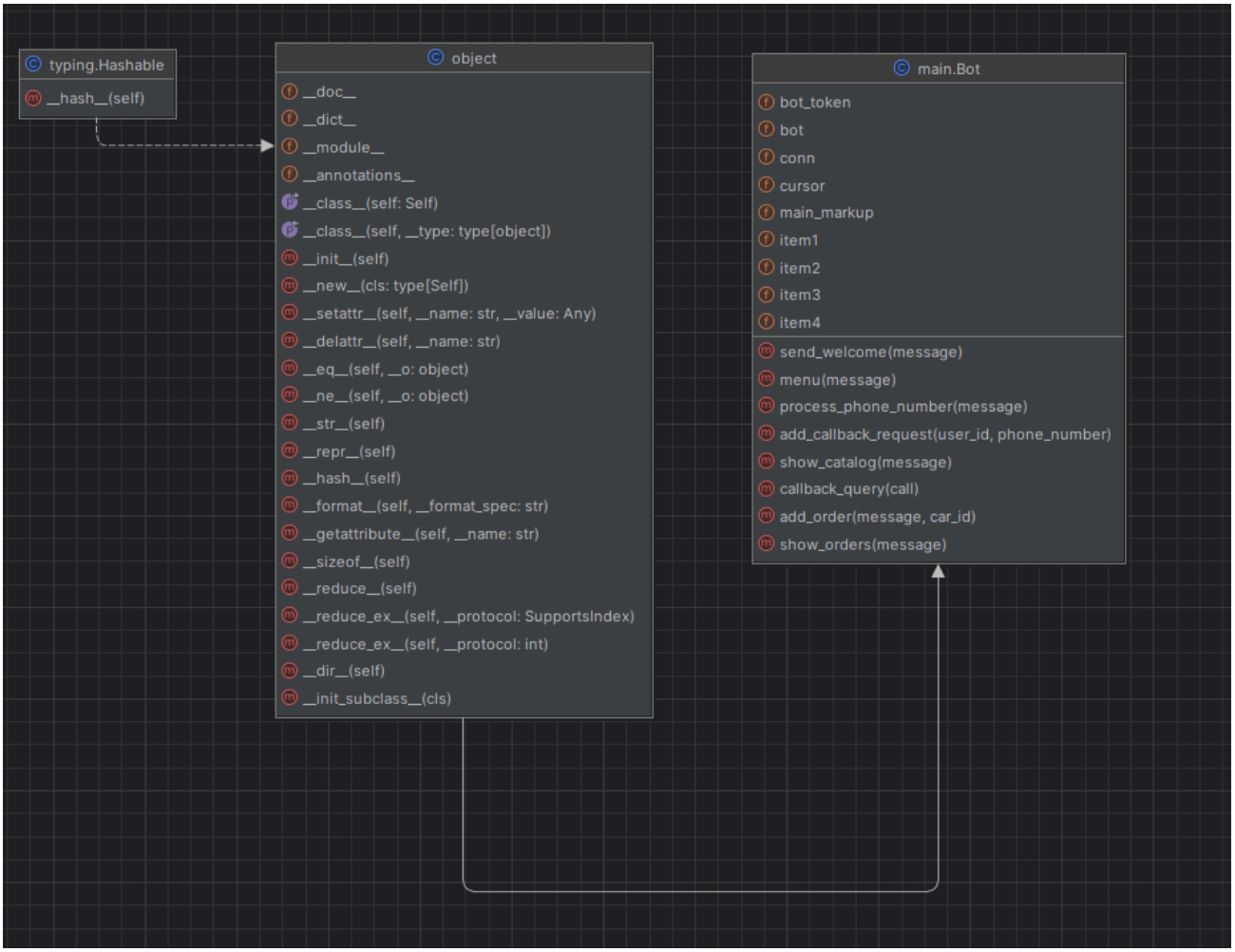

Рисунок 2.3 — Діаграма класів системи

2.3 Проектування бази даних

Проектування бази даних є одним з первинних етапів проектування системи, оскільки всі подальші етапи розробки тим чи іншим чином будуть залежати від бази даних.

На рисунку 2.4 представлено схему розробленої бази даних.

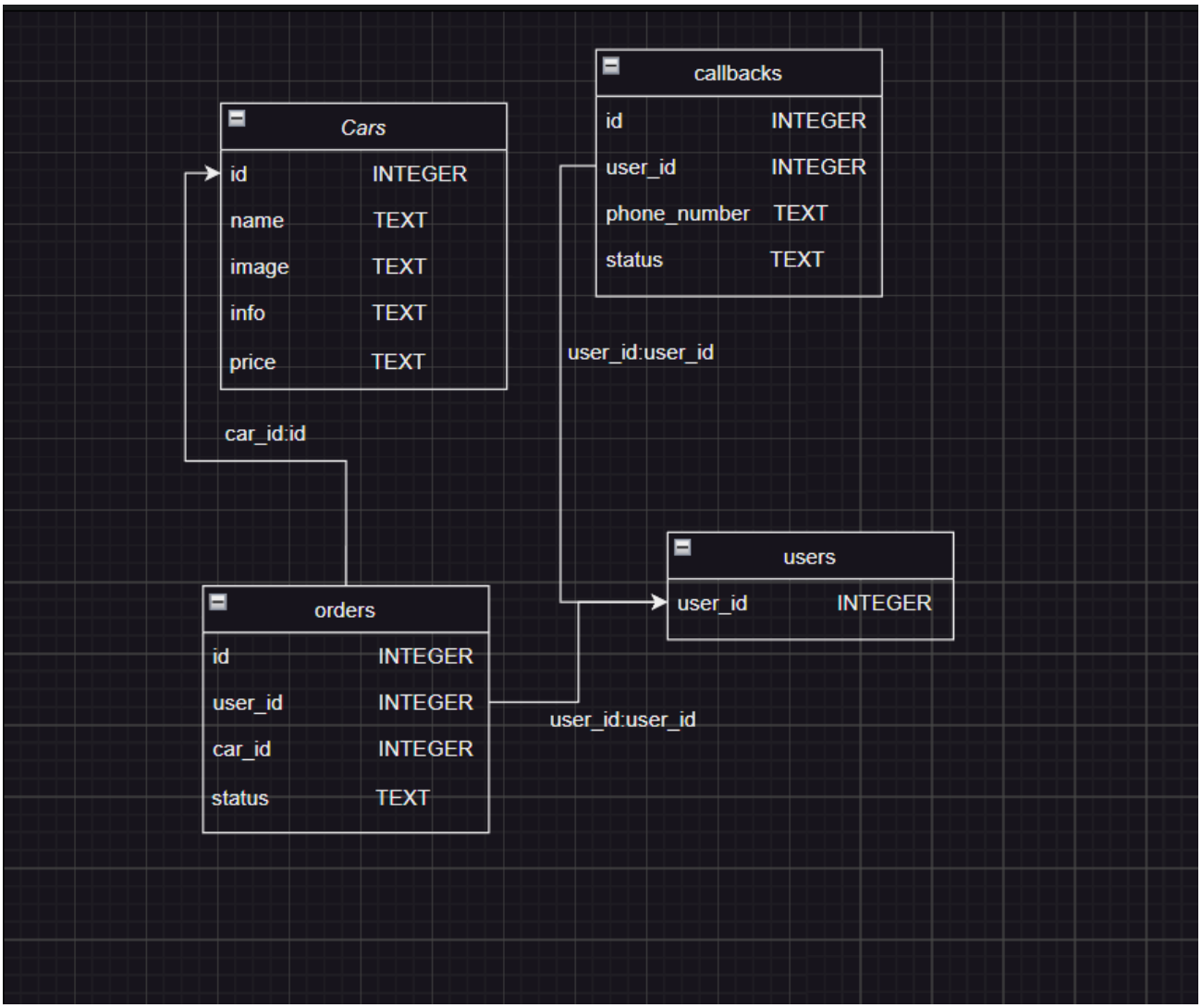

Рисунок 2.4 — Схема бази даних

3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Інструментальні засоби розробки

Python

Python - це мова програмування загального призначення високого рівня, яка працює на принципах, спрямованих на забезпечення читабельності коду через використання значних відступів. Її мовна структура та об'єктно-орієнтований підхід призначені для допомоги розробникам у написанні послідовних та логічних кодів як для невеликих, так і для великих проектів.

Python володіє динамічним збором сміття та підтримує кілька парадигм програмування, включаючи структурне (включаючи процедурне), об'єктно-орієнтоване та функціональне програмування. Велику користь приносить розширена стандартна бібліотека Python, яку часто називають "мовою з вбудованими батареями".

Як наступник мови програмування ABC, Гвідо ван Россум розпочав роботу над Python наприкінці 1980-х років і випустив першу версію під назвою Python 0.9.0 в 1991 році. У 2000 році був представлений Python 2.0, який вніс нові функції, такі як розширене керування списками та систему збору сміття на основі рахунку посилань. Python 2 був офіційно припинений в 2020 році з випуском версії 2.7.18. Python 3.0, випущений у 2008 році, є головною версією мови, яка, хоч і не є повністю сумісною з попередньою версією, стала стандартом.

Python завжди володів великою популярністю серед мов програмування. Гвідо ван Россум розпочав роботу над ним наприкінці 1980-х років в Нідерландах, розвиваючи його як наступник мови програмування ABC, яка була натхненною SETC та спроможною опрацьовувати винятки та взаємодіяти з операційною системою Amoeba. Перша реалізація Python розпочалася у грудні 1989 року. Він вів розробку проекту, бувши провідним розробником, до 12 липня 2018 року, коли оголосив про свою "постійну відпустку" і отримав титул "Диктатора, дружнього до життя" Python. Ван Россум був завжди відомий своєю

відданістю спільноті Python та його тривалими внесками у проект. Він продовжує грати важливу роль, беручи участь в роботі наглядової ради Python разом з іншими розробниками. У січні 2019 року активні розробники ядра Python обрали Бретта Кеннона, Ніка Коглана, Баррі Варшава, Керол Віллінг та самого Гвідо ван Россума до складу ради директорів з п'яти членів. Відтоді Гвідо ван Россум зняв свою кандидатуру з розгляду на посаду в раді директорів в 2020 році.

Python є багатопарадигмовою мовою програмування, повністю підтримує об'єктно-орієнтоване і структурне програмування, а також надає можливості функціонального і аспектно-орієнтованого програмування, включаючи метапрограмування та метаоб'єктний підхід. Python також включає динамічний збір сміття і підтримує динамічне розділення імен для зв'язування методів і змінних під час виконання програми.

Зокрема, Python розроблений з метою підтримки функціонального програмування, запозичуючи функції, такі як фільтрація, відображення, скорочення і ітератори. Стандартна бібліотека містить модулі itertools і functools, які реалізують функціональні інструменти, вдосконалені за запозиченням з мов Haskell і Standard ML.

Python має вражаючу розширюваність завдяки своєму модульному підходу, хоча не всі його функції вбудовані в ядро мови. Ця модульність дозволяє легко додавати програмовані інтерфейси до існуючих програм, роблячи його популярним серед розробників для розширення функціональності.

Гвідо ван Россум визначив фундаментальні принципи дизайну Python, зокрема, зосередженість на читабельності коду, змістовній та зрозумілій синтаксису та простоті у виборі методів кодування. Як відмінність від філософії Perl "Є більше одного способу зробити це", Python спирається на девіз "Повинен бути один - бажано лише один - очевидний спосіб зробити це."

Розробники Python докладають зусиль, щоб запобігти передчасній оптимізації і не вносять змін до некритичних частин CPython, які можуть

покращити швидкість за рахунок складності. Там, де швидкість має велике значення, програмісти можуть перевести критичні фрагменти коду на розширення, написані на C, або використовувати інші інтерпретатори, такі як PyPy, які пропонують компіляцію "точно вчасно". Також є варіанти, такі як Cython, які можуть конвертувати Python-код в C і забезпечити прямий доступ до API рівня C для інтерпретатора Python.

Python не підтримує оптимізацію хвостових викликів або першокласні розширення, і, за словами Гвідо ван Россума, ніколи не буде. Однак, розширивши генератор Python, у версії 2.5 надається краща підтримка функцій, подібних до корутина. До 2,5 генераторів є ледачими ітераторами, інформація передається від генераторів в одному напрямку. З Python 2.5 ви можете передавати інформацію назад до функції генератора, а з Python 3.3 ви можете передавати інформацію через кілька рівнів стека.

Python чітко розрізняє вирази та вирази, що відрізняється від мов, таких як Common Lisp, Scheme або Ruby. Це може призвести до дублювання певних функцій.

Метод для об’єкта — це функція, приєднана до класу об’єкта; синтаксис instance.method (аргумент) — це синтаксичний цукор Class.method (екземпляр, аргумент) для поширених методів і функцій. Методи Python мають явний параметр self для доступу до даних екземпляра, що є протилежністю неявного self (або цього) в деяких інших об’єктно-орієнтованих мовах програмування

Бібліотеки, такі як NumPy, SciPy і Matplotlib, роблять Python корисним для наукових обчислень, тоді як спеціалізовані бібліотеки, наприклад Biopython і Astropy, надають можливості, орієнтовані на конкретні галузі. SageMath-Math є програмним продуктом з інтерфейсом ноутбука, який може бути програмований на Python і охоплює широкий спектр математичних галузей, включаючи алгебру, комбінаторику, числову математику, теорію чисел і обчислення. OpenCV має розширені можливості в галузі комп'ютерного зору та обробки зображень.

Python також широко використовується у проектах зі штучного інтелекту та машинного навчання, включаючи бібліотеки, такі як TensorFlow, Keras, Pytorch і Scikit-learn. Завдяки своїй модульній архітектурі, простому синтаксису та багатими інструментами для обробки тексту, Python також є популярним для роботи з обробкою природних мов. Крім того, Python використовується як мова сценаріїв у багатьох програмних продуктах, включаючи Abaqus, FreeCAD, програми для 3D-анімації, програми для обробки зображень та програми для роботи з графічними зображеннями.

Python є складовою багатьох операційних систем, включаючи Linux, AmigaOS 4, FreeBSD, NetBSD, OpenBSD та macOS. Він широко використовується в інсталяторах для дистрибутивів Linux, таких як Ubuntu та Red Hat Linux, і в системі керування пакетами Portage для Gentoo Linux.

Мова Python також використовується для розробки експлойтів та інформаційної безпеки. Багато програм для навчання, такі як Sugar Laptop XO і проект Raspberry Pi, використовують Python для програмування користувачів.

LibreOffice включає Python і планує замінити Java на Python в якості мови для сценаріїв. Його постачальник сценаріїв Python став важливою особливістю версії 4.0, яка вийшла 7 лютого 2013 року.

З урахуванням широкого спектру застосувань та великих можливостей мови Python, яку було описано вище, можна визначити, що ця мова програмування відзначається універсальністю. Саме через це вона обрана для виконання цієї роботи.

PyCharm

PyCharm представляє собою інтегроване середовище розробки (IDE), спеціально призначене для програмування мовою Python. Це програмне забезпечення було розроблене чеською компанією JetBrains і володіє широким функціоналом, включаючи аналіз коду, графічний налагоджувач, вбудований модуль тестування та можливість інтеграції з системами контролю версій (VCS).

PyCharm також підтримує використання фреймворку Django для розробки веб-додатків і платформи Anaconda для аналізу даних.

Ця інтегрована середовище розробки доступна для користувачів Windows, macOS та Linux, завдяки чому вона є кросплатформовою. Крім безкоштовної версії PyCharm Community Edition з відкритим вихідним кодом, існує також професійна версія, яка надає додатковий функціонал і розповсюджується за власною ліцензією.

Зокрема, PyCharm Community Edition, яка є вільно доступною для користувачів з відкритим кодом, була представлена 22 жовтня 2013 року

3.2 Архітектура програмного додатку

Модель водоспаду - це розподіл процесу розробки на послідовні фази, де кожна фаза передбачає певний набір завдань та ґрунтується на результатах попередньої фази. Цей підхід типовий для певних галузей інженерного проектування. У виправленні програмного забезпечення він зазвичай використовується як менш ітеративний та більш жорсткий метод, оскільки процес рухається в одному напрямку (зверху вниз), через послідовні фази, які включають в себе концепцію, ініціацію, аналіз, дизайн, розробку, тестування, впровадження та підтримку.

Модель водоспаду мала своє походження в промисловості та будівництві, де структуроване фізичне оточення призводило до високих витрат на зміни в проектуванні на ранніх стадіях розробки. Коли цей підхід був застосований до розробки програмного забезпечення, він був єдиним підходом для творчої роботи, що базується на знаннях.

Перша детальна презентація, яка описує використання таких етапів у розробці програмного забезпечення, була проведена Феліксом Торресом та Гербертом Д. Бенінгтоном на симпозіумі з передових методів програмування для цифрових комп'ютерів у 1956 році.

Хоча термін "водоспад" не був використаний в цій статті, перша офіційна детальна діаграма процесу, пізніше відома як "модель водоспаду", зазвичай приписується статті Вінстона В. Ройса 1970 року. У 1985 році Міністерство оборони Сполучених Штатів відобразило цей підхід у своєму стандарті DOD-STD-2167A для роботи з підрядниками щодо розробки програмного забезпечення, вимагаючи виконання шести етапів: аналізу вимог до програмного забезпечення, попереднього проекту, детального дизайну, кодування та модульного тестування, інтеграції та тестування.

1. Вимоги до системи та програмного забезпечення перераховані в документі з вимогами до продукту.
2. Під час аналізу формуються моделі, схеми та бізнес-правила.
3. Під час дизайну створюється архітектура програмного забезпечення.
4. Кодування включає в себе розробку, перевірку та інтеграцію програмного забезпечення.
5. Тестування полягає в систематичному виявленні та виправленні дефектів.
6. Операції включають встановлення, міграцію, підтримку та технічне обслуговування повних систем.

Отже, модель водоспаду наголошує на тому, що перехід до наступної фази повинен відбуватися лише після того, як попередню фазу перевірено і переглянуто.

Існують різні модифіковані версії моделі водоспаду, включаючи остаточну модель Ройса, які можуть включати незначні або значні зміни в цьому процесі. Ці варіації можуть включати повернення до попередньої фази після виявлення недоліків або повний повтор фази проектування, якщо наступні фази виявилися недостатньо успішними.

Важливим аргументом на користь моделі водоспаду є те, що вона підкреслює значення документації, такої як вимоги та проектні документи, а також вихідний код. У менш структурованих та докладно задокументованих

методологіях втрачена інформація, якщо члени команди покидають проект до його завершення. Модель водоспаду забезпечує можливість легкої ознайомленості з проектом для нових членів команди або для випадків зміни команди, оскільки всі необхідні документи доступні.

Модель водоспаду забезпечує систематизований підхід, оскільки вона лінійно проходить через окремі, легкі для розуміння і пояснення фази, надаючи чіткі пункти в розробці. Саме ця структура робить її зрозумілою. Модель водоспаду також надає чіткі точки відсічі в процесі розробки, що полегшує відслідковування прогресу. Через це, вона часто використовується як вихідна точка для розуміння моделей розробки в багатьох підручниках і курсах з програмної інженерії.

Критика

1. Клієнти можуть не мати точно сформульованих вимог до продукту до того, як побачать його в дії, і це може викликати зміни у вимогах. Це може призвести до необхідності редизайну, перерозробки та повторного тестування, що призводить до додаткових витрат.
2. Розробники можуть не передбачити майбутніх труднощів під час створення нового програмного продукту або функцій. У таких випадках, було б краще переглянути проект, а не настоювати на виконанні проекту, який не враховує виявлені обмеження, вимоги або проблеми.
3. Деякі організації намагаються вирішити відсутність конкретних вимог від клієнтів шляхом залучення системних аналітиків до вивчення існуючих ручних систем і аналізу їх функціоналу. Проте в практиці може бути важко дотримуватися жорсткого розділення між системним аналізом і програмуванням, оскільки впровадження складної системи майже завжди призводить до виявлення проблем та крайових випадків, які системний аналітик не передбачив.

У відповідь на ці проблеми були запропоновані модифіковані моделі водоспаду, такі як "модель Сашимі", "модель з фазами, що перекриваються", "модель з підпроектами" та "модель з зменшенням ризику". Модифіковані моделі водоспаду відповідають на критику "чистої" моделі водоспаду, шляхом впровадження змін у процес розробки програмного забезпечення для зменшення витрат і полегшення управління змінами.

Монолітна архітектура програмного забезпечення (ПЗ) - це традиційний підхід до розробки програмного забезпечення, який базується на створенні одного компактного, нерозділеного програмного модуля, який включає в себе всі компоненти системи. Цей підхід використовується протягом багатьох років і продовжує бути популярним в сучасній програмно-інженерній практиці. В цій статті ми розглянемо основні характеристики та переваги монолітної архітектури ПЗ, а також деякі недоліки та виклики, з якими вона може зіткнутися.

1. Основні характеристики монолітної архітектури ПЗ
    - Один великий модуль: Монолітна архітектура передбачає розробку програмного забезпечення у вигляді одного великого модуля, де всі компоненти (наприклад, логіка бізнесу, інтерфейс користувача, база даних) розташовані в межах цього модуля.
    - Всеїдність: Усі компоненти системи в монолітній архітектурі обмінюються даними безпосередньо між собою, оскільки вони знаходяться в одному контексті виконання.
    - Лінійний процес розробки: Розробка та розгортання монолітної системи зазвичай відбуваються в лінійному порядку, коли всі компоненти розробляються та інтегруються разом, а після цього відбувається розгортання.
    - Загальна база коду: У монолітній архітектурі весь код програмного забезпечення знаходиться в одній кодовій базі, що спрощує розробку та управління проектом.

2. Переваги монолітної архітектури ПЗ
    - Простота: Монолітна архітектура проста в розумінні та розробці, оскільки всі компоненти знаходяться в одному місці.
    - Швидкість розгортання: Розгортання монолітних систем зазвичай є швидким і простим, оскільки всі компоненти розробляються разом та інтегруються в єдиний образ.
    - Продуктивність: У монолітній архітектурі немає витрат на мережеву комунікацію між компонентами, що може поліпшити продуктивність системи.
    - Легке масштабування: Монолітна архітектура дозволяє легко масштабувати систему, додавши додаткові ресурси до єдиного модуля.
3. Недоліки та виклики монолітної архітектури ПЗ
    - Велика кодова база: У монолітних системах кодова база може стати великою та складною для управління, особливо при зростанні розміру проекту.
    - Обмежена гнучкість: Монолітна архітектура може бути менш гнучкою, оскільки зміни в одному компоненті можуть мати вплив на всю систему.
    - Відсутність модульності: Монолітна архітектура утримує всі компоненти в одному модулі, що робить складним тестування та розгортання окремих частин системи.
    - Складність масштабування: Великі монолітні системи можуть бути складними для масштабування, оскільки потрібно масштабувати всі компоненти разом, навіть якщо потреба в масштабуванні є лише для деяких частин системи.
4. Сучасні тренди і підходи

Незважаючи на те, що монолітна архітектура є класичним підходом до розробки програмного забезпечення, сучасна практика демонструє широкий спектр альтернативних підходів, таких як мікросервісна архітектура, контейнеризація та серверне забезпечення функцій. Ці підходи надають більшу гнучкість, модульність та швидкість розгортання системи, але також вимагають від розробників додаткових знань та навичок.

У підсумку, монолітна архітектура програмного забезпечення є традиційним підходом, який все ще залишається популярним. Вона має свої переваги, такі як простота та швидкість розгортання. Однак, вона також має свої недоліки, такі як обмежена гнучкість та складність масштабування. З розвитком технологій та появою нових вимог до програмного забезпечення, розробники активно досліджують та використовують інші підходи, які надають більшу гнучкість та модульність системи.

3.3 Програмна реалізація

Імпорт бібліотек:

```
import telebot
from telebot import types
from bot_logger import BotLogger  # імпорт логера
from dotenv import load_dotenv  # імпорт файла в якому є секретний токен та id адміна бота
import sqlite3
import re
import os  # імпорт os для знаходження файлу env
```

Тут імпортуються необхідні бібліотеки: telebot для роботи з Telegram API, types з telebot для створення клавіатур, sqlite3 для роботи з базою даних SQLite та re для роботи з регулярними виразами.

Створюємо экземпляр логеру

```
logger = BotLogger('bot.log')
```

Завантаження змінних з .env файлу

```
load_dotenv()
```

Токен вашого бота

```
bot_token = os.getenv('BOT_TOKEN')
bot_admin = os.getenv('BOT_ADMIN')
bot = telebot.TeleBot(bot_token)
```

Підключення до бази даних

```
conn = sqlite3.connect('cars.db', check_same_thread=False)
cursor = conn.cursor()
```

Створення таблиць у базі даних (якщо вони ще не існують)

```
cursor.execute('''CREATE TABLE IF NOT EXISTS cars
(id INTEGER PRIMARY KEY, name TEXT, image TEXT, info TEXT, price TEXT)''')
cursor.execute('''CREATE TABLE IF NOT EXISTS orders
(id INTEGER PRIMARY KEY, user_id INTEGER, car_id INTEGER, status TEXT)''')
cursor.execute('''CREATE TABLE IF NOT EXISTS callbacks
(id INTEGER PRIMARY KEY, user_id INTEGER, phone_number TEXT, status TEXT)''')
cursor.execute('''CREATE TABLE IF NOT EXISTS users
(user_id INTEGER PRIMARY KEY)''')
```

Функція для додавання користувача до бази даних

```
def add_user(user_id):
cursor.execute("INSERT OR IGNORE INTO users (user_id) VALUES (?)", (user_id,))
```

```
    conn.commit()
```

Функція для видалення користувача з бази даних

```
    def remove_user(user_id):
    cursor.execute("DELETE FROM users WHERE user_id=?",
(user_id,))
    conn.commit()
```

Створення головного меню

```
    def create_main_markup(is_admin=False):
    main_markup =
types.ReplyKeyboardMarkup(resize_keyboard=True)
    item_stock = types.KeyboardButton("□ У наявності 🚗")
    item_not_stock = types.KeyboardButton("🚤 Під замовлення
🚘")
    item_info = types.KeyboardButton("Інформація про
компанію")
    item_contact = types.KeyboardButton("Контакти")
    item_orders = types.KeyboardButton("📋 Кабінет покупця")
    item_order = types.KeyboardButton("✍ Зробити замовлення")
    main_markup.add(item_stock, item_not_stock)
    main_markup.add(item_info, item_contact)
    main_markup.add(item_orders, item_order)
    if is_admin:
    item_broadcast = types.KeyboardButton("Відправити
повідомлення")
    main_markup.add(item_broadcast)
    return main_markup
```

Головне меню

```
    @bot.message_handler(commands=['start'])
```

```
def send_welcome(message):
user_id = message.from_user.id
is_admin = str(user_id) == bot_admin
bot.send_message(message.chat.id, "Вітаємо в нашому автоботі!",
reply_markup=create_main_markup(is_admin))
add_user(user_id) # Додавання користувача до бази даних
logger.log_info(f"Користувач {user_id} запустив бота з командою /start")
```

Обробка кнопок головного меню

```
@bot.message_handler(func=lambda message: True)
def menu(message):
user_id = message.from_user.id
if message.text == '□ У наявності 🚗':
show_catalog(message)
elif message.text == '🛤 Під замовлення 🚘':
show_catalog(message)
elif message.text == 'Інформація про компанію':
bot.send_message(message.chat.id, "Замовляйте новинки електромобілів в GetMaximum!"
" - Ви отримаєте консультацію від наших експертів та зможете замовити автомобіль," "
який відповідає вашим потребам і бажанням. Повністю перевіряємо авто перед
відправкою" " та видачею клієнтам Прямі поставки електрокарів із Китаю та Європи
Можливість замовлення" " додаткових опцій чи аксесуарів Надаємо розстрочку, кредит чи
лізинг",
```

```
    reply_markup=create_main_markup(str(user_id)          ==
bot_admin))
    elif message.text == '📋 Кабінет покупця':
    show_orders(message)
    elif message.text == '✅ Зробити замовлення':
    msg  =  bot.send_message(message.chat.id,  "Надішліть  ваш
номер телефону, і наш
    менеджер зв'яжеться з вами.")
    bot.register_next_step_handler(msg, process_phone_number)
    elif message.text == 'Контакти':
    text_contact = """🏢 Київ
    🏢 Крещатик 10
    📲  +380112233445"""  bot.send_message(message.chat.id,
text_contact,
    parse_mode='Markdown',
reply_markup=create_main_markup(str(user_id) == bot_admin))
    elif  message.text  ==  'Відправити  повідомлення'  and
str(user_id) == bot_admin:
    msg   =   bot.send_message(message.chat.id,   "Введіть
повідомлення для розсилки:")
    bot.register_next_step_handler(msg, broadcast_message)
```

Функція для відправки повідомлення всім

Користувачам

```
    def broadcast_message(message):
    if str(message.chat.id) == bot_admin:
    broadcast_text = message.text
    cursor.execute("SELECT user_id FROM users")
    users = cursor.fetchall()
    for user in users:
```

```
    try:
    bot.send_message(user[0], broadcast_text)
    except telebot.apihelper.ApiTelegramException as e:
    if e.error_code == 403: # User blocked the bot
    remove_user(user[0])
    logger.log_info(f"Користувач {user[0]} заблокував бота і був видалений з бази даних")
    else:
    logger.log_error(f"Не вдалося відправити повідомлення користувачу {user[0]}: {e}")
```

Обробка введеного номера телефону

```
    def process_phone_number(message):
    phone_number = message.text
    # Перевірка на валідність номера телефону
    if not re.match(r"^+?\d{10,15}$", phone_number):
    bot.send_message(message.chat.id, "Будь ласка, введіть валідний номер телефону.",
    reply_markup=create_main_markup(str(message.chat.id) == bot_admin))
    return
    add_callback_request(message.chat.id, phone_number)
    bot.send_message(message.chat.id, "Ваш запит зафіксовано!",
    reply_markup=create_main_markup(str(message.chat.id) == bot_admin))
```

Функція для додавання запиту на зворотний дзвінок

```
    def add_callback_request(user_id, phone_number):
    cursor.execute("INSERT INTO callbacks (user_id, phone_number, status) VALUES (?, ?, ?)",
```

```
    (user_id, phone_number, 'Очікування'))
    conn.commit()
```

Показ каталогу

```
    def showcatalog(message):
    cursor.execute("SELECT * FROM cars")
    for car in cursor.fetchall():
    markup = types.InlineKeyboardMarkup()
    order_button = types.InlineKeyboardButton(text="Замовити",
callback_data=f'order{car[0]}')
    markup.add(order_button)
    bot.send_photo(message.chat.id, car[2], f"{car[1]}\nЦіна:
{car[4]}\n{car[3]}",
    reply_markup=markup)
```

Обробка callback-запитів від інлайн-кнопок

```
    @bot.callbackquery_handler(func=lambda call: True)
    def callback_query(call):
    if call.data.startswith('order'):
    carid = int(call.data.split('')[1])
    add_order(call.message, car_id)
    bot.answer_callback_query(call.id, "Замовлення прийнято")
```

Функція для додавання замовлення

```
    def add_order(message, car_id):
    user_id = message.chat.id
    cursor.execute("INSERT INTO orders (user_id, car_id,
status) VALUES (?, ?, ?)", (user_id,
    car_id, 'Оформлено'))
    conn.commit()
    bot.send_message(message.chat.id, "Ваше замовлення було
успішно додано!")
```

Показ замовлень користувача

```
    def show_orders(message):
    user_id = message.chat.id
    cursor.execute("SELECT FROM orders WHERE user_id=?",
(user_id,))
    orders = cursor.fetchall()
    if orders:
    ifpass = True
    for order in orders:
    cursor.execute("SELECT FROM cars WHERE id=?", (order[2],))
    car = cursor.fetchone()
    if car is None:
    pass
    else:
    bot.send_message(message.chat.id, f"Замовлення {car[1]}:
{order[3]}")
    ifpass = False
    if ifpass:
    bot.send_message(message.chat.id, "У вас немає активних
замовлень.")
    else:
    bot.send_message(message.chat.id, "У вас немає активних
замовлень.")
```

Запуск бота

```
    bot.polling(none_stop=True)
    ```
```
```

3.4 Використання програмного додатку

Програмний додаток представляє собою телеграм-бота для замовлення автомобілів. Скріншот роботи представлено на рисунку 3.1.

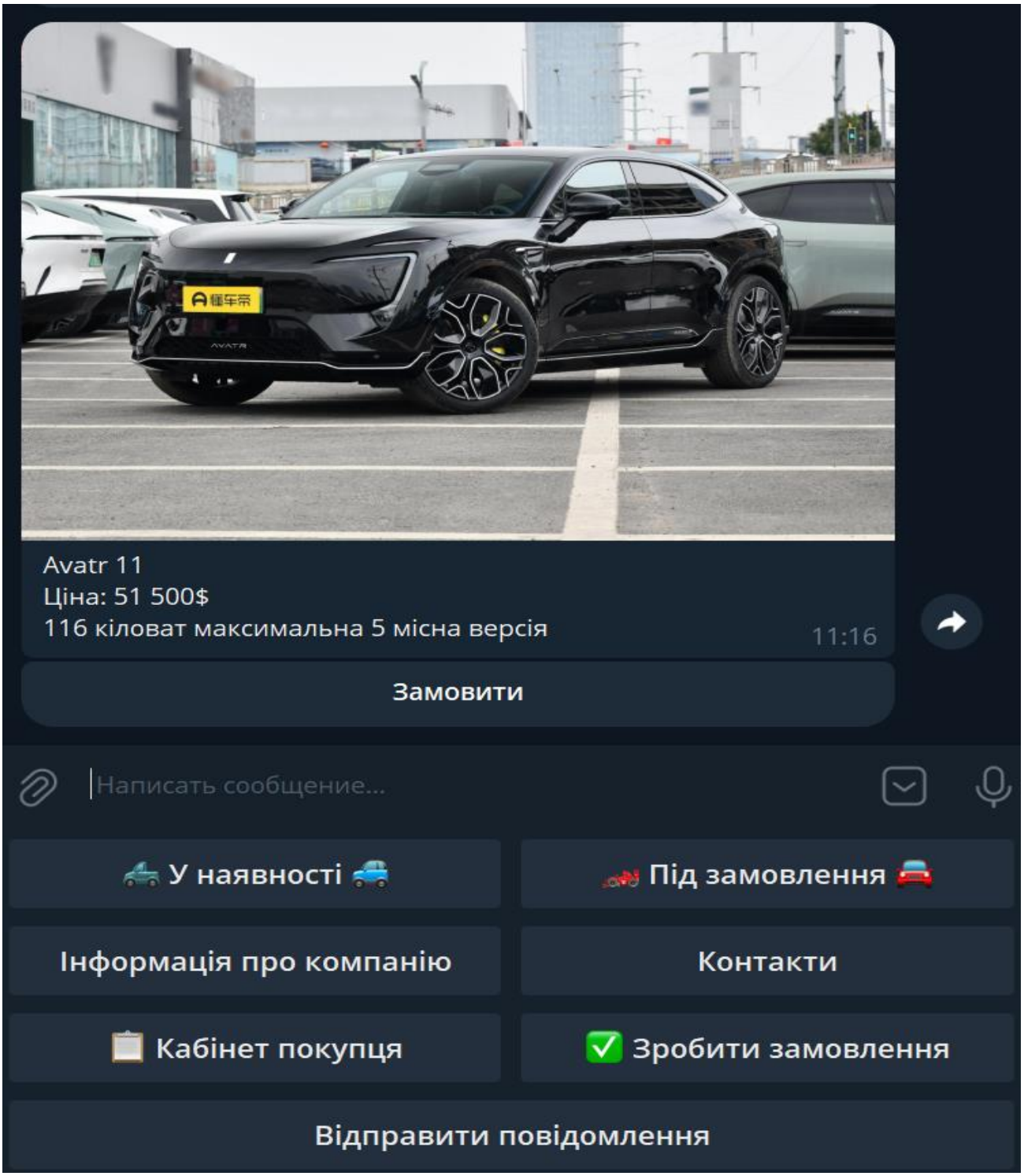

Рисунок 3.1 — Скріншот роботи

3.5 Перевірка роботи кнопок

Перевірка роботи функціоналу бота є важливою частиною його розробки. Зараз ми будемо перевіряти кожен елемент інтерфейсу окремо, і подивимося на результат.

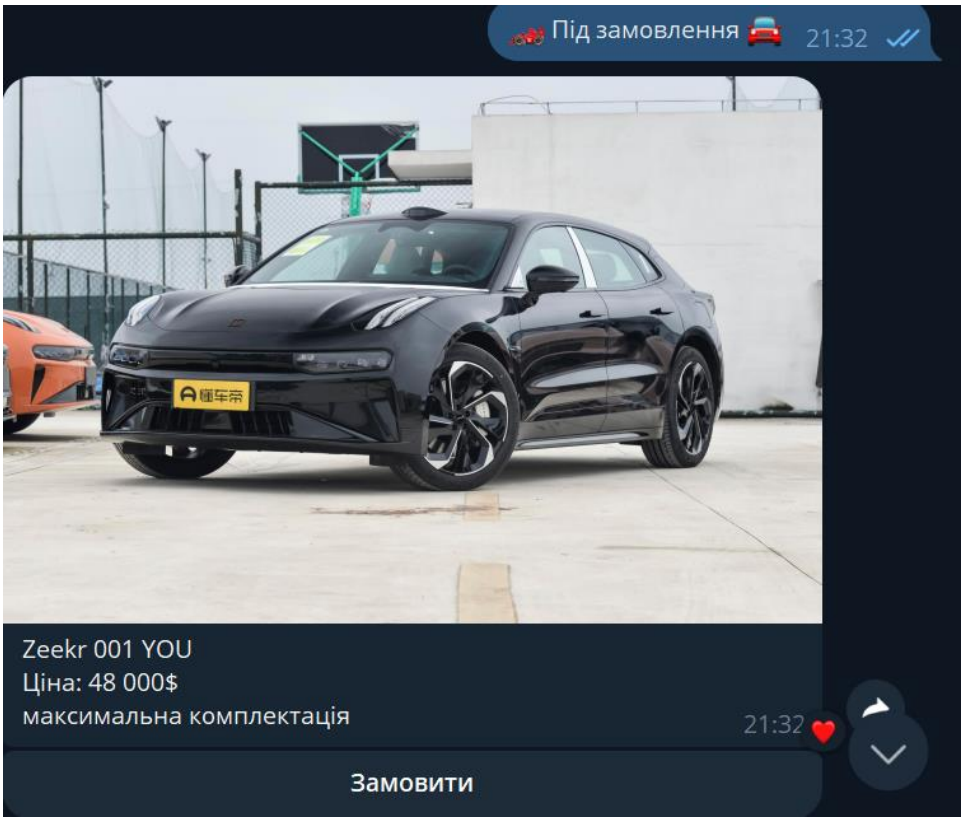

Рисунок 3.2 — Скріншот роботи кнопки “Під замовлення”

Кнопка “Під замовлення” повинна була видати перелік автомобілів з їх характеристиками,ціною, фото, взявши цю інформацію із бази даних. Ми отримали очікуваний результат

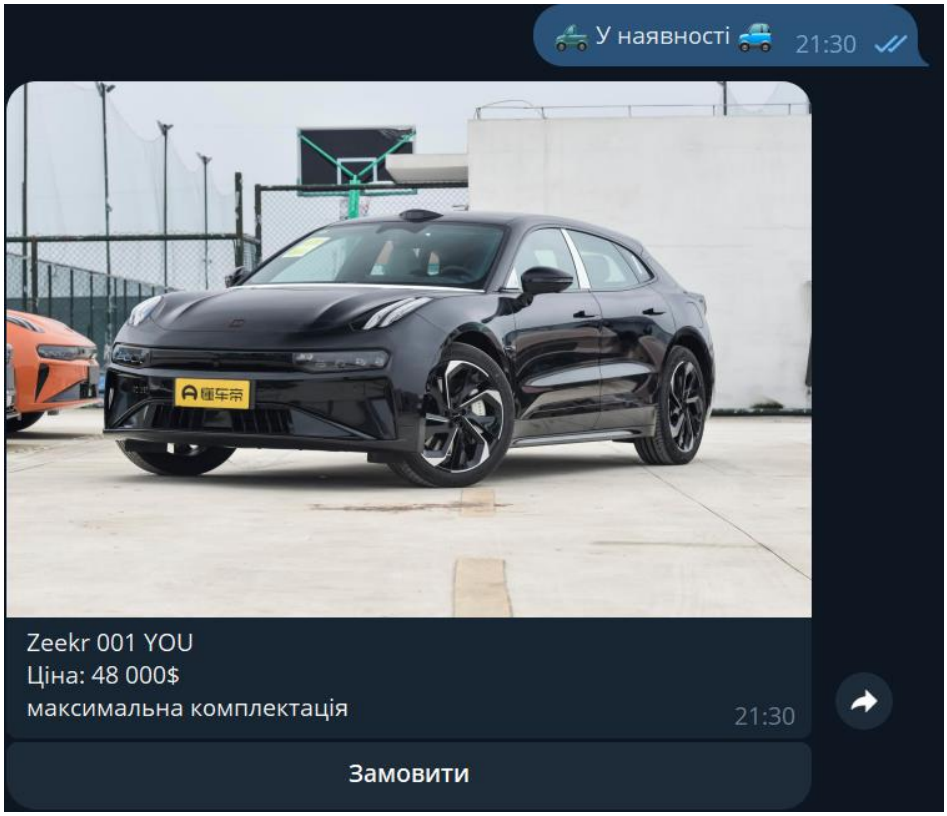

Рисунок 3.3 — Скріншот роботи кнопки “У наявності”

Кнопка “У наявності” мала мивести список автомобілів доступних для купівля з наявності. Весь перелік доступних автомобілів був виведений.

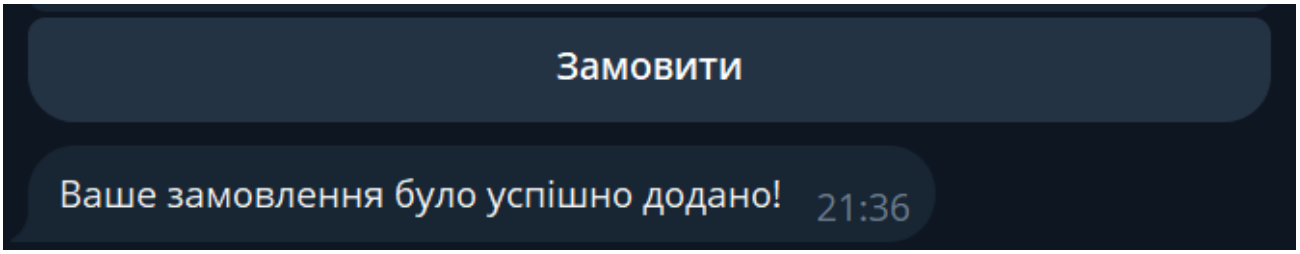

Рисунок 3.4 — Скріншот роботи кнопки “Замовити”

Кнопка “Замовити” наявна під оголошенням на кожен автомобіль в наявноності та під замовлення, результат відображається у кабінеті покупця. Задача повнітю виповнена.

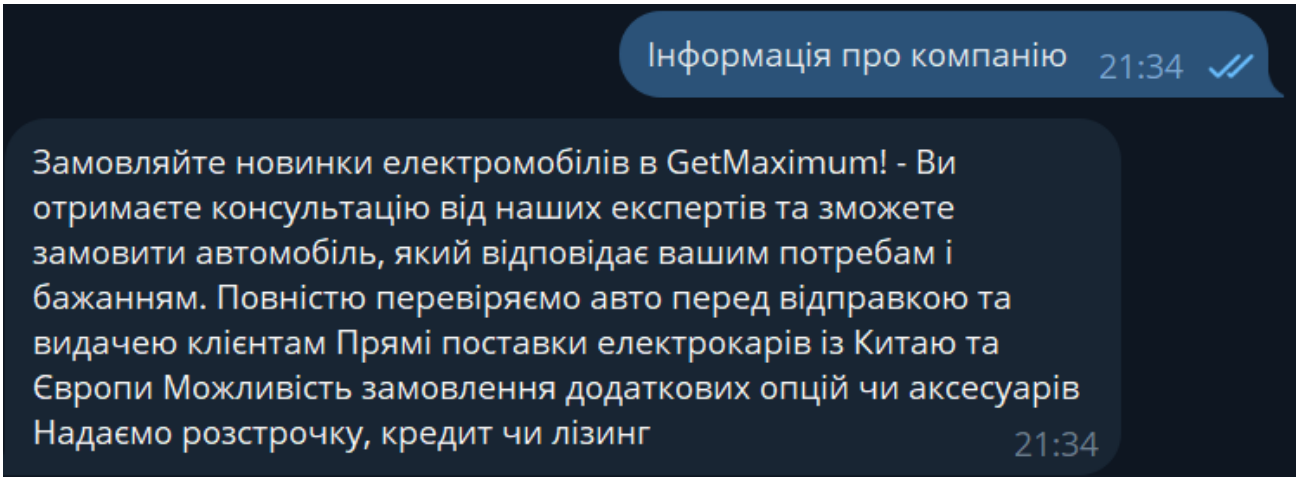

Рисунок 3.5 — Скріншот роботи кнопки “Інформація про компанію”

Кнопка “ Інформація про компанію ” лише виводить потрібний текст із описом нашої компанії.

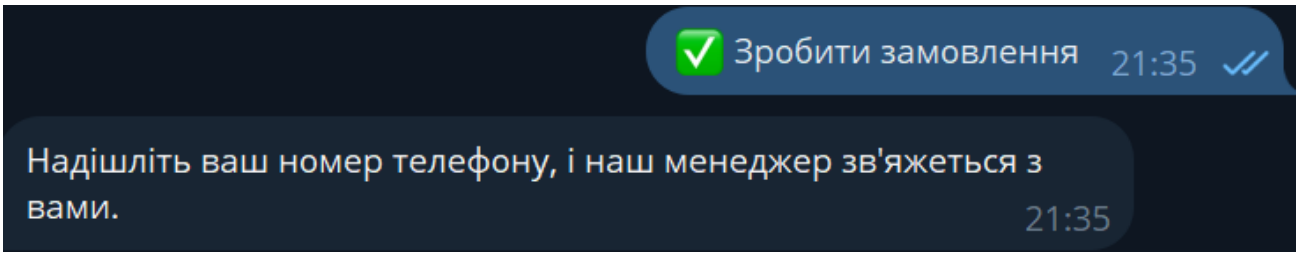

Рисунок 3.6 — Скріншот роботи кнопки “Зробити замовлення”

Кнопка “ Зробити замовлення ” видає клієнту текст, та залишає запит у нашій базі даних.

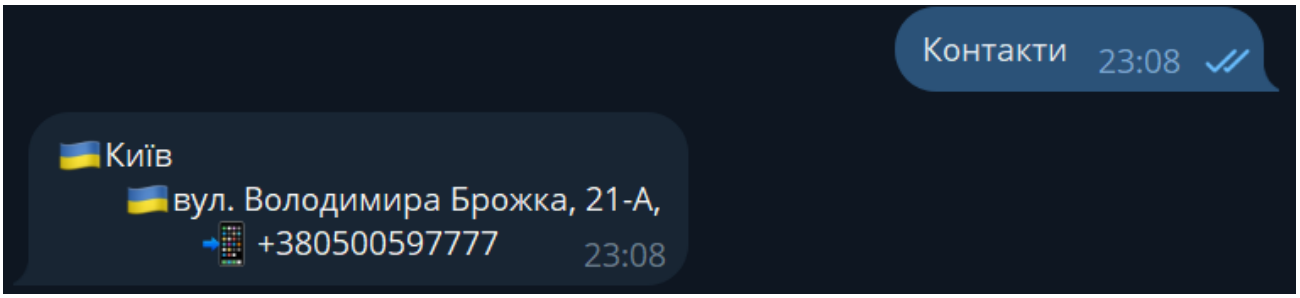

Рисунок 3.6 — Скріншот роботи кнопки “Контакти”

Кнопка “Контакти” видаж клієнту заздалегіть прописаний текст із актуальною адресою та номером телефону.

Рисунок 3.7 — Скріншот роботи кнопки “Кабінет покупця”

Кнопка “Кабінет покупця” виводить весь перелік автомобілів замовлених клієнтом.

3.6 Розгортання бота на сервері

Вхід на сервер через SSH

Копіювання бота на сервер

Для копіювання файлів бота на сервер ви можете використовувати команду scp (Secure Copy Protocol).

Запуск бота у Docker контейнері

1. Відкрийте командний рядок на вашому комп'ютері (Command Prompt на Windows

або Terminal на Mac/Linux).

2. Введіть команду для підключення до сервера через SSH. Вам знадобляться IPадреса сервера та ваш SSH логін (наприклад, user@192.168.1.1):

ssh user@192.168.1.1

3. Введіть ваш пароль, коли вас попросять. Ви успішно увійдете на сервер, якщо все зроблено правильно.

1. Відкрийте командний рядок на вашому комп'ютері.

2. Введіть команду для копіювання файлів на сервер:

scp -r C:\Users\Desktop\Студент\2. Бот\bot_cars

user@192.168.1.1:/home/user/bot_cars

-r означає рекурсивне копіювання, що необхідно для копіювання всієї директорії.

user@192.168.1.1 – ваш SSH логін і IP-адреса сервера.

/home/user/bot_cars – шлях на сервері, куди будуть скопійовані файли.

ВИСНОВКИ

У процесі виконання цього дипломного проекту ми досягли поставлених цілей і вирішили ключові завдання, пов'язані з розробкою телеграм-бота для автомобільної платформи. Результати цієї роботи можуть бути підсумовані наступним чином:

1. Реалізація функціональності: Телеграм-бот був успішно розроблений і включає в себе основні функціональні можливості: перегляд каталогу автомобілів, отримання інформації про автомобілі та компанію, здійснення замовлень, а також функцію зворотного дзвінка.
2. Використання сучасних технологій: У проекті були застосовані сучасні технології, зокрема PyTelegramBotAPI для розробки бота та SQLite для створення і управління базою даних.
3. Тестування та оптимізація: Проведене тестування підтвердило стабільність і надійність роботи бота. Виявлені помилки були виправлені, що забезпечило високий рівень ефективності інтерфейсу та зручності для користувачів.
4. Аналіз ринку: Аналіз існуючих рішень на ринку дозволив краще зрозуміти потреби цільової аудиторії та адаптувати бота до специфіки українського ринку автомобілів.
5. Перспективи розвитку: Проект демонструє значний потенціал для подальшого розвитку та вдосконалення, включаючи інтеграцію з додатковими сервісами, розширення функціоналу бота та поліпшення користувацького інтерфейсу.

У цілому, розробка телеграм-бота для автомобільної платформи виявилася успішним проектом, який відповідає сучасним вимогам ринку та потребам користувачів. Цей проект не тільки сприяє цифровізації процесів купівлі автомобілів, але й відкриває нові можливості для підвищення ефективності бізнесу в цій галузі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Translations". Archived from the original on 6 May 2018. Retrieved 18 September 2022.
2. "Q: Can I translate Telegram?". Telegram Messenger LLP. Retrieved 4 February 2023.
3. "List of Telegram applications". 6 February 2014. Archived from the original on 22 May 2016. Retrieved 6 February 2014.
4. "Telegram FAQ". Telegram. Archived from the original on 9 February 2014. Retrieved 29 March 2021.
5. "Members ordered by country code". RIPE Network Coordination Centre.
6. "Telegram introduces end-to-end encrypted video calls". The Next Web. 14 August 2020. Archived from the original on 1 March 2021. Retrieved 29 March 2021.
7. Peters, Jay (16 February 2023). "Meta is copying Telegram channels in Instagram". The Verge. Retrieved 17 October 2023.
8. Russell, Jon (18 May 2017). "Telegram now lets users buy things from chatbots in its messaging app". TechCrunch. Retrieved 17 October 2023.
9. EWDN (30 August 2013). "Russia's Zuckerberg launches Telegram, a new instant messenger service". Reuters. Archived from the original on 28 January 2021. Retrieved 8 November 2020.
10. "Meet Telegram, A Secure Messaging App From The Founders Of VK, Russia's Largest Social Network". TechCrunch. 28 October 2013. Archived from the original on 26 November 2015. Retrieved 8 November 2020.
11. "'Nobody can block it': how the Telegram app fuels global protest". The Guardian. 7 November 2020. Archived from the original on 7 November 2020. Retrieved 7 November 2020.

12."The Evolution of Telegram". Telegram. Archived from the original on 26 January 2021. Retrieved 4 January 2021.
13."Most Popular Messaging Apps: Top Messaging Apps 2021". respond.io. Retrieved 17 October 2023.
14.Cheh, Samantha (11 August 2017). "Cambodia: Govt officials favor Telegram to protect communications". Tech Wire Asia. Retrieved 17 October 2023.
15.Singh, Manish (18 July 2023). "Telegram raises $210 million through bond sales". TechCrunch. Retrieved 2 August 2023.
16.Singh, Manish (30 August 2021). "Telegram surpasses 1 billion downloads". TechCrunch. Retrieved 17 October 2023.
17."Durov Telegram". Telegram. 8 February 2021. Archived from the original on 9 February 2021. Retrieved 10 February 2021.
18."Telegram Tops The List Of Most Downloaded Apps In The World For January 2021: Report". Mashable India. 9 February 2021.
19."Pavel Durov left Russia after being pushed out". Agence France-Presse. 22 April 2014. Archived from the original on 24 April 2014. Retrieved 23 April 2014 – via Economic Times.
20.Hakim, Danny (2 December 2014). "Once Celebrated in Russia, the Programmer Pavel Durov Chooses Exile". The New York Times. Archived from the original on 6 September 2015. Retrieved 19 November 2015.
21.Shu, Catherine (27 October 2013). "Meet Telegram, A Secure Messaging App From The Founders Of VK, Russia's Largest Social Network". TechCrunch. Archived from the original on 26 November 2015. Retrieved 18 March 2016.
22."Telegram F.A.Q". Archived from the original on 9 February 2014. ...making profits will never be an end-goal for Telegram.
23."Why Telegram has become the hottest messaging app in the world". The Verge. Archived from the original on 13 March 2016. Retrieved 25 February 2014.

Telegram operates as a non-profit organization, and doesn't plan to charge for its services.

24. Dewey, Caitlin (23 November 2015). "The secret American origins of Telegram, the encrypted messaging app favored by the Islamic State". The Washington Post. Archived from the original on 12 May 2019. Retrieved 31 March 2018.
25. "Telegram Messenger on the App Store". App Store. Archived from the original on 19 May 2019. Retrieved 19 November 2015.
26. Telegram FZ LLC – Dun & Bradstreet
27. Thornhill, John (3 July 2015). "Lunch with the FT: Pavel Durov". Financial Times. Archived from the original on 19 September 2016. Retrieved 19 November 2015.
28. Auchard, Eric (23 February 2016). "Telegram app free-speech advocate no stranger to Apple-FBI woes". Reuters. Archived from the original on 12 May 2019. Retrieved 12 August 2019 – via www.reuters.com.
29. Turton, William (29 September 2017). "What isn't Telegram saying about its connections to the Kremlin?". The Outline. Archived from the original on 18 May 2019. Retrieved 11 October 2017.
30. Brandom, Russell (6 October 2014). "Surveillance drives South Koreans to encrypted messaging apps". The Verge. Archived from the original on 6 November 2015. Retrieved 19 November 2015.
31. Descalsota, Marielle (28 March 2022). "Meet Pavel Durov, the tech billionaire who founded Telegram, fled from Moscow 15 years ago after defying the Kremlin, and has a penchant for posting half-naked selfies on Instagram". Business Insider. Retrieved 1 May 2022.
32. Hakim, Danny (2 December 2014). "Once Celebrated in Russia, the Programmer Pavel Durov Chooses Exile". The New York Times. ISSN 0362-4331. Retrieved 19 August 2022.

33."Telegram Hits 35M Monthly Users, 15M Daily With 8B Messages Received Over 30 Days". TechCrunch. 24 March 2014. Archived from the original on 24 November 2015. Retrieved 25 June 2017.

34."Telegram Reaches 1 Billion Daily Messages". Telegram. 8 December 2014. Archived from the original on 24 July 2019. Retrieved 8 December 2014.

35."Telegram Hits 2 Billion Messages Sent Daily". Telegram. 13 May 2015. Archived from the original on 12 May 2019. Retrieved 31 July 2015.

36.Lomas, Natasha (21 September 2015). "Telegram Now Seeing 12BN Daily Messages, up From 1BN in February". Techcrunch. Archived from the original on 27 November 2015. Retrieved 19 November 2015.

37.Burns, Matt (23 February 2016). "Encrypted Messaging App Telegram Hits 100M Monthly Active Users, 350k New Users Each Day". TechCrunch. Archived from the original on 9 May 2019. Retrieved 23 February 2016.

38."This $5 Billion Encrypted App Isn't for Sale at Any Price". Bloomberg. 12 December 2017. Archived from the original on 20 May 2019. Retrieved 22 December 2017.

39."200,000,000 Monthly Active Users". Telegram. 22 March 2018. Archived from the original on 23 September 2018. Retrieved 22 March 2018.

40.Lomas, Natasha (14 March 2019). "Telegram gets 3M new signups during Facebook apps' outage". TechCrunch. Archived from the original on 2 May 2019. Retrieved 14 March 2019.

41.Shieber, Jonathan (13 March 2019). "Update: Facebook, Instagram and Messenger were down for many users". TechCrunch. Archived from the original on 2 May 2019. Retrieved 14 March 2019.

42."SECURITIES AND EXCHANGE COMMISSION, ::Plaintiff, :19 Civ. 9439(PKC):-against -:ECF Case:TELEGRAM GROUPINC.andTON ISSUERINC" (PDF). The US Securities and Exchange Commission. 11 October

2019. Archived (PDF) from the original on 12 November 2020. Retrieved 17 October 2019.

43. "Telegram hits 500M monthly active users". Telegram. Archived from the original on 11 June 2021. Retrieved 12 January 2021.
44. ""Respect your users": Telegram founder Pavel Durov slams Facebook". The Hindu Business Line. 9 January 2021. Archived from the original on 9 January 2021. Retrieved 10 January 2021.
45. "Telegram tops 1 billion downloads". TechCrunch. 30 August 2021. Retrieved 1 May 2022.
46. Porter, Jon (6 October 2021). "Telegram gains 70M new users in just one day after Facebook outage". The Verge. Archived from the original on 6 October 2021. Retrieved 6 October 2021.
47. "Telegram surpasses WhatsApp to become Russia's top messenger". Business Recorder. Reuters. 21 March 2022. Retrieved 21 March 2022.
48. Roth, Emma (20 June 2022). "Telegram's Premium subscription is here and it costs $4.99 / month". The Verge. Retrieved 20 June 2022.
49. "700 Million Users and Telegram Premium". Telegram. 21 June 2022. Retrieved 8 July 2022.
50. Lopez, Miguel (30 January 2014). "Configurando Telegram en el iPhone, en la web y en el Mac" [Configuring Telegram in the Apple iPhone, the Web and the Mac] (in Spanish). Applesfera. Archived from the original on 7 August 2019. Retrieved 4 December 2014.
51. Mehta, Ivan (7 December 2022). "Telegram is auctioning phone numbers to let users sign up to the service without any SIM". TechCrunch. Retrieved 9 December 2022.
52. Witman, Emma (22 January 2021). "How to make a Telegram account and start using the popular group chatting app". Business Insider. Archived from the original on 9 February 2021. Retrieved 28 May 2021.

53."no login by sms code in desktop version". GitHub. Archived from the original on 28 October 2021. Retrieved 10 July 2021.

54.Munizaga, Jonathan (1 December 2014). "Telegram ya permite migrar conversaciones y contactos a una línea nueva" [Telegram already allows migrating conversations and contacts to a new line] (in Spanish). Wayerless. Archived from the original on 19 December 2014. Retrieved 2 December 2014.

55.Mateo, David G (1 December 2014). "Telegram ahora permite traspasar mensajes al cambiar de número" (in Spanish). TuExperto. Archived from the original on 12 May 2019. Retrieved 2 December 2014.

56."Secure Messaging App Telegram Adds Usernames And Snapchat-Like Hold-To-View For Media". Techcrunch. 23 October 2014. Archived from the original on 12 May 2019. Retrieved 23 October 2014.

57."Hiding Last Seen Time - Done Right". 19 November 2014. Archived from the original on 9 May 2019. Retrieved 18 May 2017.

ДОДАТОК А
ЛІСТИНГ ПРОГРАМНОГО КОДУ ТЕЛЕГРАМ БОТУ

```
import telebot  # імпорт телебота
from telebot import types
from bot_logger import BotLogger  # імпорт логера
from dotenv import load_dotenv  # імпорт файла в якому є секретний токен та id адміна бота
import sqlite3
import re
import os  # імпорт os для знаходження файлу env

# Создаем экземпляр логера
logger = BotLogger('bot.log')

# Завантаження змінних з .env файлу
load_dotenv()

# Токен вашого бота
bot_token = os.getenv('7132540095:AAH1DyMqSf4BIp4WwCwSPD16Fa-hio_9oec')
bot_admin = os.getenv('416548168')
bot = telebot.TeleBot(bot_token)

# Підключення до бази даних
conn = sqlite3.connect('cars.db', check_same_thread=False)
cursor = conn.cursor()
```

```
    # Створення таблиць у базі даних (якщо вони ще не існують)
    cursor.execute('''CREATE TABLE IF NOT EXISTS cars
                      (id INTEGER PRIMARY KEY, name TEXT, image TEXT, info TEXT, price TEXT)''')
    cursor.execute('''CREATE TABLE IF NOT EXISTS orders
                      (id INTEGER PRIMARY KEY, user_id INTEGER, car_id INTEGER, status TEXT)''')
    cursor.execute('''CREATE TABLE IF NOT EXISTS callbacks
                      (id INTEGER PRIMARY KEY, user_id INTEGER, phone_number TEXT, status TEXT)''')
    cursor.execute('''CREATE TABLE IF NOT EXISTS users
                      (user_id INTEGER PRIMARY KEY)''')


    # Функція для додавання користувача до бази даних
    def add_user(user_id):
        cursor.execute("INSERT OR IGNORE INTO users (user_id) VALUES (?)", (user_id,))
        conn.commit()


    # Функція для видалення користувача з бази даних
    def remove_user(user_id):
        cursor.execute("DELETE FROM users WHERE user_id=?", (user_id,))
        conn.commit()


    # Створення головного меню
```

```
def create_main_markup(is_admin=False):
    main_markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    item_stock = types.KeyboardButton("□ У наявності 🚗")
    item_not_stock = types.KeyboardButton("🚤 Під замовлення 🚘")
    item_info = types.KeyboardButton("Інформація про компанію")
    item_contact = types.KeyboardButton("Контакти")
    item_orders = types.KeyboardButton("💼 Кабінет покупця")
    item_order = types.KeyboardButton("✅ Зробити замовлення")
    main_markup.add(item_stock, item_not_stock)
    main_markup.add(item_info, item_contact)
    main_markup.add(item_orders, item_order)
    if is_admin:
        item_broadcast = types.KeyboardButton("Відправити повідомлення")
        main_markup.add(item_broadcast)
    return main_markup

# Головне меню
@bot.message_handler(commands=['start'])
def send_welcome(message):
    user_id = message.from_user.id
    is_admin = str(user_id) == bot_admin
    bot.send_message(message.chat.id, "Вітаємо в нашому автоботі!",
reply_markup=create_main_markup(is_admin))
    add_user(user_id)  # Додавання користувача до бази даних
    logger.log_info(f"Пользователь {user_id} запустил бота с командой
/start")
```

```
# Обробка кнопок головного меню
@bot.message_handler(func=lambda message: True)
def menu(message):
    user_id = message.from_user.id
    if message.text == '□ У наявності 🚗':
        show_catalog(message)
    elif message.text == '🛠️ Під замовлення 🚘':
        show_catalog(message)
    elif message.text == 'Інформація про компанію':
        bot.send_message(message.chat.id, "Замовляйте новинки електромобілів в GetMaximum!"
                                          " - Ви отримаєте консультацію від наших експертів та зможете замовити автомобіль,"
                                          " який відповідає вашим потребам і бажанням. Повністю перевіряємо авто перед відправкою"
                                          " та видачею клієнтам Прямі поставки електрокарів із Китаю та Європи Можливість замовлення"
                                          " додаткових опцій чи аксесуарів Надаємо розстрочку, кредит чи лізинг",
                         reply_markup=create_main_markup(str(user_id) == bot_admin))
    elif message.text == '🛍️ Кабінет покупця':
        show_orders(message)
    elif message.text == '✅ Зробити замовлення':
        msg = bot.send_message(message.chat.id, "Надішліть ваш номер телефону, і наш менеджер зв'яжеться з вами.")
```

```
            bot.register_next_step_handler(msg, process_phone_number)
        elif message.text == 'Контакти':
            text_contact = """UAКиїв
            UAвул. Володимира Брожка, 21-А,
                📞 [+380500597777](tel:+380500597777)"""
            bot.send_message(message.chat.id, text_contact, parse_mode='Markdown',
reply_markup=create_main_markup(str(user_id) == bot_admin))
        elif message.text == 'Відправити повідомлення' and str(user_id) ==
bot_admin:
            msg = bot.send_message(message.chat.id, "Введіть повідомлення для
розсилки:")
            bot.register_next_step_handler(msg, broadcast_message)


    # Функція для відправки повідомлення всім користувачам
    def broadcast_message(message):
        if str(message.chat.id) == bot_admin:
            broadcast_text = message.text
            cursor.execute("SELECT user_id FROM users")
            users = cursor.fetchall()
            for user in users:
                try:
                    bot.send_message(user[0], broadcast_text)
                except telebot.apihelper.ApiTelegramException as e:
                    if e.error_code == 403:  # User blocked the bot
                        remove_user(user[0])
```

```
                logger.log_info(f"Користувач {user[0]} заблокував бота і був видалений з бази даних")
            else:
                logger.log_error(f"Не вдалося відправити повідомлення користувачу {user[0]}: {e}")


    # Обробка введеного номера телефону
    def process_phone_number(message):
        phone_number = message.text
        # Перевірка на валідність номера телефону
        if not re.match(r"^\+?\d{10,15}$", phone_number):
            bot.send_message(message.chat.id, "Будь ласка, введіть валідний номер телефону.", reply_markup=create_main_markup(str(message.chat.id) == bot_admin))
            return
        add_callback_request(message.chat.id, phone_number)
        bot.send_message(message.chat.id, "Ваш запит зафіксовано!", reply_markup=create_main_markup(str(message.chat.id) == bot_admin))


    # Функція для додавання запиту на зворотний дзвінок
    def add_callback_request(user_id, phone_number):
        cursor.execute("INSERT INTO callbacks (user_id, phone_number, status) VALUES (?, ?, ?)", (user_id, phone_number, 'Очікування'))
        conn.commit()


    # Показ каталогу
```

```
def show_catalog(message):
    cursor.execute("SELECT * FROM cars")
    for car in cursor.fetchall():
        markup = types.InlineKeyboardMarkup()
        order_button = types.InlineKeyboardButton(text="Замовити", callback_data=f'order_{car[0]}')
        markup.add(order_button)
        bot.send_photo(message.chat.id, car[2], f"{car[1]}\nЦіна: {car[4]}\n{car[3]}", reply_markup=markup)


# Обробка callback-запитів від інлайн-кнопок
@bot.callback_query_handler(func=lambda call: True)
def callback_query(call):
    if call.data.startswith('order_'):
        car_id = int(call.data.split('_')[1])
        add_order(call.message, car_id)
        bot.answer_callback_query(call.id, "Замовлення прийнято")


# Функція для додавання замовлення
def add_order(message, car_id):
    user_id = message.chat.id
    cursor.execute("INSERT INTO orders (user_id, car_id, status) VALUES (?, ?, ?)", (user_id, car_id, 'Оформлено'))
    conn.commit()
    bot.send_message(message.chat.id, "Ваше замовлення було успішно додано!")
```

```
# Показ замовлень користувача
def show_orders(message):
    user_id = message.chat.id
    cursor.execute("SELECT * FROM orders WHERE user_id=?", (user_id,))
    orders = cursor.fetchall()
    if orders:
        ifpass = True
        for order in orders:
            cursor.execute("SELECT * FROM cars WHERE id=?", (order[2],))
            car = cursor.fetchone()
            if car is None:
                pass
            else:
                bot.send_message(message.chat.id, f"Замовлення {car[1]}: {order[3]}")
                ifpass = False
        if ifpass:
            bot.send_message(message.chat.id, "У вас немає активних замовлень.")
    else:
        bot.send_message(message.chat.id, "У вас немає активних замовлень.")


# Запуск бота
bot.polling(none_stop=True)
```