МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ПОЛТАВСЬКИЙ ПОЛІТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Циклова комісія дисциплін програмної інженерії

Затверджую:

Голова циклової комісії

_____ / Олександр БАБИЧ

“___” _____ 2024 р.

КУРСОВИЙ ПРОЄКТ

з дисципліни: “Проєктування автоматизованих інформаційних систем”

на тему Розробка автоматизованої інформаційної системи для продажу автомобілів (доска оголошень).

КП 015.41/2021.045

Виконав: здобувач освіти 4 курсу,
групи 45
спеціальність 121 «Інженерія програмного забезпечення»
спеціалізація «Розробка програмного забезпечення»
Роман МЕДВЕДЬ
(прізвище та ініціали)

Керівник _____ Світлана ГРИЦЕНКО
(підпис) (прізвище та ініціали)

Полтава – 2024

ЗМІСТ

ВСТУП

В сучасному світі інформаційні технології все більше інтегруються у різні аспекти життя, забезпечуючи швидкість і зручність у виконанні повсякденних завдань. Серед таких сфер особливе місце займає торгівля, яка стрімко переходить у формат онлайн-платформ. Важливою частиною цієї тенденції є спеціалізовані дошки оголошень, які полегшують комунікацію між покупцями та продавцями, дозволяючи швидко знайти потрібний товар або продати власний.

Мета роботи – створення автоматизованої інформаційної системи для продажу автомобілів у вигляді Telegram-бота, який спростить процес розміщення оголошень, перегляду наявних пропозицій та комунікації між користувачами. Додаток у вигляді бота обраний через його доступність, зручність у користуванні та відсутність необхідності встановлювати додаткове програмне забезпечення.

Сутність вирішуваної задачі полягає в розробці та реалізації функціонального Telegram-бота, який дозволить:

— Розміщувати оголошення про продаж автомобілів із зазначенням основних параметрів (марка, модель, рік, ціна, опис).
— Додавати фото та контактний номер для зв’язку.
— Переглядати список оголошень із можливістю швидкої навігації.
— Видаляти оголошення, які вже неактуальні або більше не потрібні.

Для реалізації проєкту обрано мову програмування Python через її популярність, простоту та велику кількість готових бібліотек для роботи з Telegram API. Основою для взаємодії із користувачами є бібліотека python-telegram-bot, яка дозволяє ефективно реалізувати чат-боти. Дані оголошень зберігаються у локальній базі даних SQLite, що забезпечує простоту роботи з інформацією без необхідності використання складних серверних рішень.

Методи розв'язання задачі включають:

1. Аналіз вимог до інформаційної системи та визначення функціональних можливостей.

2. Проєктування структури бази даних для зберігання інформації про автомобілі.
3. Розробка логіки взаємодії користувачів із ботом та реалізація діалогових сценаріїв.
4. Тестування працездатності Telegram-бота та усунення можливих помилок.
5. Документування роботи та складання інструкцій з експлуатації бота.

Результатом виконання курсового проєкту є Telegram-бот для продажу автомобілів, який надає користувачам можливість розміщувати та переглядати оголошення, а також зручний інструмент для видалення оголошень. Цей бот може стати корисним рішенням для власників автомобілів, які прагнуть продати свій транспортний засіб, а також для потенційних покупців, які шукають автомобіль за заданими критеріями.

1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАВДАННЯ

1.1. Постановка задачі

Проєкт передбачає створення Telegram-бота, який слугує інтерактивною дошкою оголошень для продажу автомобілів. Бот дає можливість користувачам публікувати оголошення, переглядати пропозиції інших користувачів та видаляти свої оголошення. Бот забезпечує зручну взаємодію із системою через популярний месенджер, що робить процес продажу автомобіля більш доступним для користувачів, не потребуючи від них додаткового програмного забезпечення або платформи.

Система призначена для приватних продавців автомобілів, а також потенційних покупців, які можуть легко отримати доступ до бази оголошень через месенджер. Система надає простий та зручний спосіб розміщення і перегляду оголошень про продаж автомобілів.

Користувачі системи:

— Продавці автомобілів – користувачі, які бажають розмістити оголошення про продаж автомобіля, додавши фотографії, опис, основні технічні параметри.

— Покупці автомобілів – користувачі, які шукають пропозиції про продаж автомобілів та можуть переглядати доступні оголошення.

Опис функціональності:

— Створення оголошення: користувачі можуть розмістити оголошення з інформацією про автомобіль (марка, модель, рік випуску, ціна, опис, контактний номер і фотографія).

— Перегляд оголошень: бот надає можливість переглядати список усіх оголошень.

— Видалення оголошення: користувачі можуть видалити свої оголошення з бази даних після продажу автомобіля чи за необхідності.

1.2. Розробка прототипу інтерфейсу

Інтерфейс Telegram-бота зосереджений на текстових повідомленнях та базових командах для управління оголошеннями. Основні команди, доступні користувачам:

— /start – стартове повідомлення з інформацією про бота та доступні команди.
— /add – початок процесу створення оголошення про продаж автомобіля. Бот послідовно запитує дані (марку, модель, рік, ціну, опис, контактний номер та фотографію).
— /list – виводить список оголошень, доступних для перегляду.
— /delete – надає можливість видалити оголошення за допомогою ідентифікатора.

1.3. Опис та побудова діаграми прецедентів роботи з системою

Опис і діаграма прецедентів (use-case diagram) є важливою частиною документації курсового проєкту, оскільки вони показують, як користувачі взаємодіють із системою та які функціональні можливості забезпечує система (Рисунок 1.1).

Актори:

— Користувач (потенційний покупець або власник автомобіля)

Прецеденти:

— Перегляд оголошень
— Видалення оголошень
— Реєстрація нового оголошення
— Перевірка оголошень

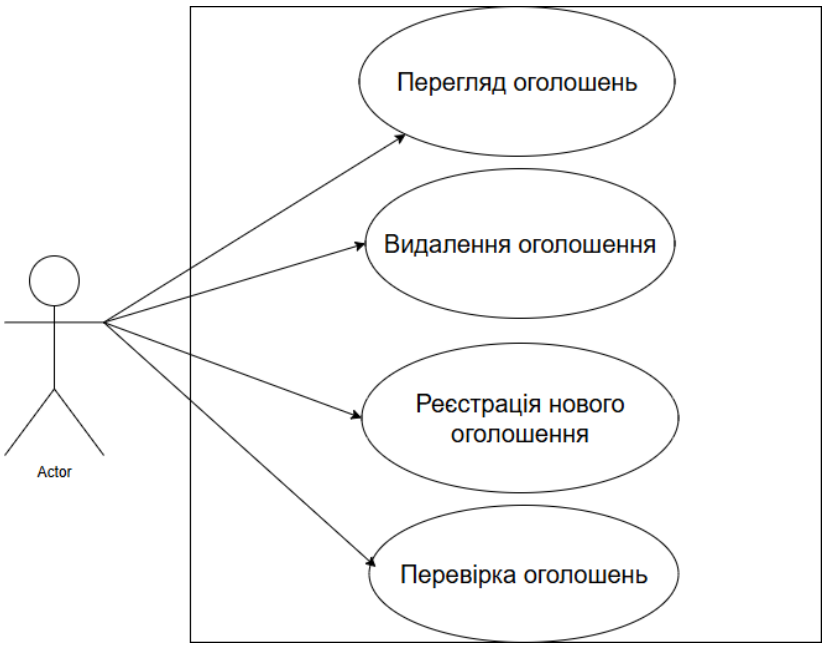

Рисунок 1.1 - Діаграма прецедентів

1.4. Інфологічне проєктування – схема бази даних

Інфологічне проєктування бази даних включає створення таблиці для зберігання оголошень (Рисунок 1.2).

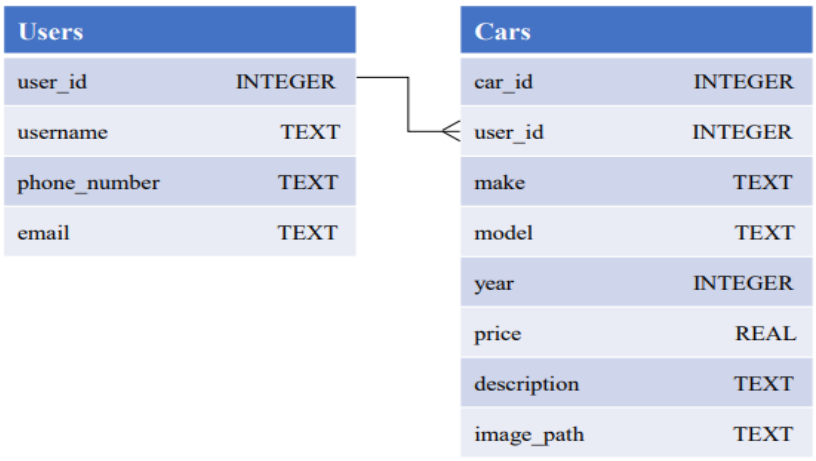

Рисунок 1.2 - Схема бази даних

Схема забезпечує оптимізоване зберігання та доступ до даних, необхідних для роботи бота.

Цей розділ надає повну інформацію про сутність системи, вхідні та вихідні дані, а також підхід до проєктування бази даних, що забезпечує функціональність Telegram-бота для управління оголошеннями.

2. ОПИС ПРОГРАМНИХ ЗАСОБІВ

2.1. Мова програмування Python

Основною мовою програмування для реалізації проєкту є Python. Python є широко використовуваною мовою, що відома своєю простотою та зрозумілим синтаксисом, а також великою кількістю бібліотек, які полегшують розробку різноманітних застосунків, зокрема для створення чат-ботів. Крім того, Python має велику кількість відкритих ресурсів, що сприяють швидкому навчанню та вирішенню завдань будь-якої складності (1).

2.2. Середовище розробки Visual Studio Code

Для написання та відлагодження коду використовувалося середовище розробки Visual Studio Code**.** Це кросплатформене середовище з великою кількістю розширень, що полегшує роботу з Python, управління проєктами та інтеграцію з системами контролю версій. Visual Studio Code надає зручні засоби для написання коду, швидкої навігації, відлагодження, а також автоматичної перевірки синтаксису.

2.3. Бібліотека python-telegram-bot

Для реалізації функціональності Telegram-бота було використано бібліотеку python-telegram-bot. Це одна з найпопулярніших бібліотек для Python, що забезпечує зручний доступ до Telegram Bot API та дозволяє будувати ботів із багатим функціоналом. Серед основних можливостей бібліотеки – підтримка асинхронних викликів, обробка команд та повідомлень користувачів, використання обробників станів для реалізації діалогів та завантаження файлів (наприклад, зображень) (2).

Основні компоненти бібліотеки, використані у проєкті:

— Application – створює основний об’єкт бота та забезпечує взаємодію з Telegram API.

— CommandHandler – обробляє введення користувачами певних команд (наприклад, /start, /add, /list).

— ConversationHandler – дозволяє створювати діалоги з користувачами для поступового збору даних для оголошень.
— MessageHandler – обробляє текстові та мультимедійні повідомлення (фотографії, документи).

2.4. Система управління базами даних SQLite

Для зберігання даних про оголошення було обрано легку реляційну базу даних SQLite. Її основними перевагами є простота налаштування та використання, висока продуктивність для невеликих проєктів і відсутність потреби у встановленні окремого серверу. SQLite інтегрується з Python за допомогою стандартного модуля sqlite3, який дозволяє легко виконувати SQL-запити, здійснювати пошук, вставку та видалення записів (3,6).

2.5. Взаємодія компонентів системи

— Користувач взаємодіє з ботом у Telegram, надсилаючи команди або дані.
— Python-telegram-bot обробляє запити та взаємодіє з Telegram API, забезпечуючи зв’язок між користувачем і ботом.
— SQLite використовується для зберігання та доступу до оголошень, що дає можливість переглядати, додавати та видаляти інформацію про автомобілі.
— Visual Studio Code використовується для написання та тестування коду, що забезпечує стабільну роботу бота і надійність його функцій.

У результаті вибір цих програмних засобів дозволив забезпечити ефективне та гнучке вирішення завдання, досягти високої продуктивності й простоти роботи системи (7).

3. РЕАЛІЗАЦІЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ ПРОЕКТУВАННЯ

3.1. Алгоритм програми, архітектура програми

Telegram-бот для продажу автомобілів реалізовано на основі асинхронної архітектури з використанням python-telegram-bot. Бот взаємодіє з користувачем через набір команд, які ініціюють певні функціональні процеси.

Основний алгоритм роботи програми передбачає початок взаємодії користувача з ботом через введення команди /start. Після цього бот відправляє привітальне повідомлення та пропонує перелік основних команд: створення оголошення та видалення оголошення. Коли користувач надсилає команду для створення нового оголошення (/add), бот ініціює діалог, під час якого поетапно запитує необхідні деталі про автомобіль. Після того, як користувач надає всі потрібні дані, такі як марка, модель, рік випуску, ціна, опис, контактний номер телефону та фото, бот зберігає оголошення в базі даних SQLite. Користувач може переглянути всі створені оголошення за допомогою команди /list. Для видалення оголошення використовується команда /delete з вказанням відповідного ідентифікатора оголошення (4,5).

3.2. Структура програми з функцій складових частин і зв'язків між ними

Програма побудована з використанням модульного підходу, де кожна функція відповідає за виконання окремої частини логіки. Основні компоненти:
1. Основний модуль (bot.py): містить функції для ініціалізації бота та обробки команд користувача (5).
1.1 Функції-обробники команд:
— start(): вітає користувача та надає список доступних команд.
— add_car(): починає діалог для створення оголошення.
— list_cars(): виводить список збережених оголошень.
— delete_car(): видаляє оголошення за його ідентифікатором.
1.2 Функції обробки діалогу:

— receive_brand(), receive_model(), receive_year(), receive_price(), receive_description(), receive_phone(), receive_photo(): поетапно приймають дані від користувача.

— save_car_to_db(): зберігає зібрані дані в базу даних.

2. База даних SQLite (cars.db): містить таблицю cars, де зберігаються всі оголошення з полями для кожної характеристики автомобіля.

3.3. Виклик та завантаження

Для запуску бота необхідно:

1. Налаштувати токен бота в Telegram і додати його у файл bot.py.

2. Встановити всі залежності, включаючи python-telegram-bot та sqlite3.

3. Запустити файл bot.py у середовищі Python.

3.4. Інструкція користувачу

Інструкція по роботі з програмою

1. Запуск програми (Рисунок 1.3):

— Запустіть bot.py у середовищі Python для активації бота.

— Відкрийте Telegram, знайдіть MEDVEDAUTOBOT та надішліть команду /start.

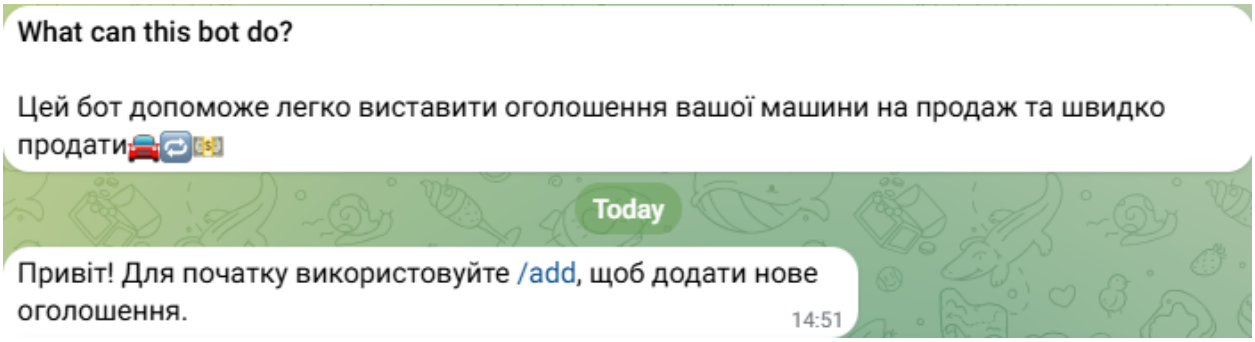

Рисунок 1.3 - Запуск програми

2. Створення оголошення (Рисунок 1.4):

— Введіть команду /add для початку створення оголошення.

— Відповідайте на послідовні запити бота, надаючи інформацію про автомобіль (марку, модель, рік, ціну, опис, контактний номер).

— Завантажте фото автомобіля, якщо бажаєте.

Рисунок 1.4 - Створення оголошення

3. Перегляд оголошень (Рисунок 1.5):

— Введіть команду /list, щоб отримати список усіх доступних оголошень із детальною інформацією про кожен автомобіль.

Рисунок 1.5 - Перегляд оголошення

4. Видалення оголошення (Рисунок 1.6):

— Щоб видалити оголошення, введіть команду /delete і вкажіть ідентифікатор оголошення.

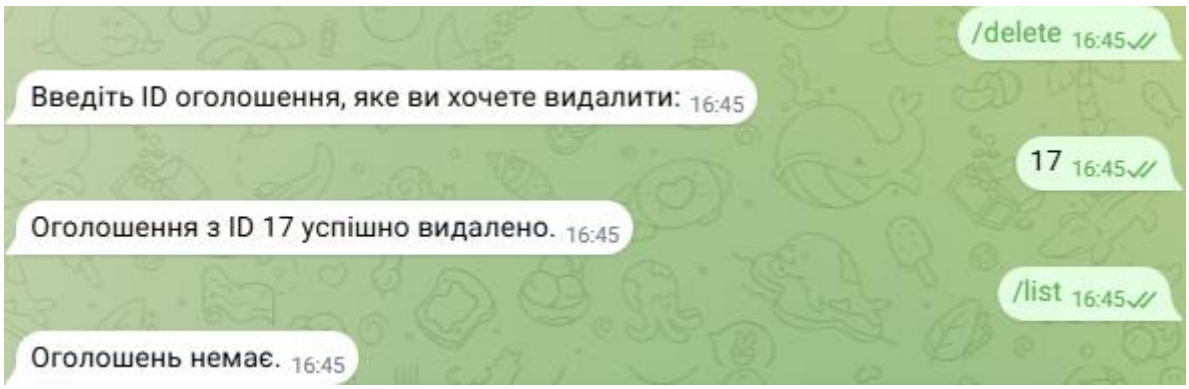

Рисунок 1.6 - Видалення оголошення

5. Завершення роботи:

— Просто закрийте сесію або вийдіть із Telegram, щоб припинити взаємодію з ботом.

ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

У ході виконання курсового проєкту було поставлено завдання розробити автоматизовану інформаційну систему у вигляді Telegram-бота для продажу автомобілів. Метою було створення платформи для зручної взаємодії між продавцями та покупцями автомобілів, де користувачі можуть легко публікувати оголошення, переглядати їх та знаходити необхідну інформацію. В рамках проєкту вдалося повністю виконати поставлені завдання:

— Створення Telegram-бота**:** реалізовано бот із функціями додавання, перегляду та видалення оголошень. Користувач може додати автомобіль із такими параметрами: марка, модель, рік випуску, ціна, опис, контактний номер, фото.

— Зберігання даних**:** бот використовує базу даних SQLite для надійного зберігання інформації про оголошення.

— Практичність**:** система дозволяє користувачам ефективно використовувати платформу Telegram, що є зручним для швидкого розміщення та пошуку автомобілів.

Розроблений бот має актуальність та практичну цінність для сучасного ринку, де попит на онлайн-платформи для продажу автомобілів залишається високим. Простота використання Telegram-бота значно спрощує процес публікації оголошень, а також є зручною альтернативою класичним веб-дошкам оголошень.

Реалізація проєкту потребувала застосування знань із наступних навчальних дисциплін:

— Програмування на Python**:** знання мови програмування, зокрема бібліотеки для роботи з Telegram API та SQL-запитами.

— Бази даних**:** навички проєктування та роботи з базами даних, у цьому випадку з SQLite, для зберігання та організації оголошень.

— Об'єктно-орієнтоване програмування**:** принципи побудови модульного коду, використання функцій та обробників для обробки запитів.

Під час роботи над проєктом вдалося поглибити знання в області розробки чат-ботів, отримати практичний досвід у роботі з бібліотекою python-telegram-bot, налаштуванні баз даних SQLite та підключенні їх до Python-проєктів. Розширено навички у сфері проєктування користувацького інтерфейсу для Telegram-ботів та вдосконалено підхід до роботи з асинхронними обробниками повідомлень.

Можливі наступні кроки для покращення та розвитку Telegram-бота:

— Додавання фільтрів для пошуку: можливість пошуку автомобілів за різними критеріями, такими як ціна, рік випуску, модель.
— Інтеграція з іншими сервісами: наприклад, з картографічними сервісами для додавання локації автомобіля.
— Розширення бази даних: додавання підтримки різних типів файлів для оголошень, таких як відео або додаткові фотографії.

Загалом, створення Telegram-бота стало корисним практичним проєктом, який дав змогу освоїти інструменти, що можуть бути використані для створення комерційних та інформаційних чат-ботів, спрямованих на різноманітні потреби ринку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алонсо, П. "Програмування на Python для початківців". Київ: Видавництво "Основа", 2020.

2. "Python Telegram Bot Documentation". Python Telegram Bot Library, https://python-telegram-bot.readthedocs.io/.

3. Чалкер, Б., Томпсон, М. "Бази даних для програмістів". Львів: Видавництво "Літера", 2019.

4. Мартін, Р. К. "Чиста архітектура: структура та розробка програмного забезпечення". Харків: Фоліо, 2021.

5. Еріксон, М. "Інтерактивні системи: проектування користувацького інтерфейсу". Одеса: Наукова думка, 2020.

6. "SQLite Documentation". SQLite Official Website, https://www.sqlite.org/docs.html.

7. Абрамс, Д. "Програмування для Telegram: бот у дії". Міжнародний журнал комп'ютерних наук, випуск № 5, 2019.

Ці джерела забезпечили теоретичну основу та практичні інструменти для розробки Telegram-бота для продажу автомобілів, сприяючи вдосконаленню знань у сфері програмування, архітектури програмного забезпечення та роботи з базами даних.

ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ

1. ПРИЗНАЧЕННЯ І МЕТА СТВОРЮВАНОЇ СИСТЕМИ

1.1. Загальні відомості

Цей проєкт розробляє систему для автоматизації процесу купівлі-продажу автомобілів шляхом створення спеціалізованого бота на платформі Telegram, який функціонує як інтерактивна дошка оголошень. Основними об’єктами автоматизації є приватні особи, які бажають придбати або продати автомобіль, а також адміністратори, які можуть забезпечувати підтримку та моніторинг оголошень.

Система розробляється в межах курсового проєкту з дисципліни «Проєктування автоматизованих інформаційних систем». Термін виконання роботи: 1 жовтня 2024 року – 1 січня 2025 року. Завдання буде виконуватися поетапно згідно з календарним планом.

1.2. Призначення системи

Бот для продажу автомобілів у Telegram розроблено для підтримки функцій оголошень про продаж та забезпечення простого й зручного користування. Основні функціональні можливості включають:

— додавання оголошень (додавання інформації про автомобіль, його характеристики, завантаження фото);
— редагування й видалення оголошень;
— пошук оголошень за певними критеріями (марка, модель, рік, ціновий діапазон тощо);
— можливість перегляду доступних оголошень;
— надання контактної інформації для зв’язку з продавцем.

1.3. Мета створення

Система створюється для забезпечення користувачів можливістю легко додавати й переглядати оголошення, а також оперативно знаходити потрібний автомобіль. Основна мета полягає в поліпшенні процесу взаємодії між

продавцями та покупцями автомобілів у межах платформи Telegram та оптимізації процесу пошуку.

2. ХАРАКТЕРИСТИКА ОБ’ЄКТІВ АВТОМАТИЗАЦІЇ ТА ВИМОГИ ДО СИСТЕМИ

2.1. Відомості про об’єкт автоматизації

Основними об'єктами автоматизації є:

— процес додавання та редагування оголошень користувачами;

— перегляд та пошук оголошень за різними параметрами;

— процес автентифікації адміністратора для управління оголошеннями та перегляду статистики.

2.2. Вимоги до системи вцілому

Система має бути децентралізованою, але з можливістю адміністрування, щоб забезпечити належний рівень підтримки користувачів. Вона повинна включати наступні підсистеми:

— збирання та обробка даних — для введення та зберігання інформації про автомобілі;

— підсистема зберігання — бази даних для збереження всіх активних оголошень;

— підсистема виводу — відображення й пошук оголошень відповідно до запитів користувачів.

2.3. Вимоги до функцій (задач), що виконуються системою

Бот має працювати в режимі 24/7, забезпечуючи доступ користувачів до оголошень, підтримувати функції додавання, пошуку та перегляду оголошень.

2.4. Вимоги до надійності

Програмний продукт повинен забезпечувати безперервну роботу та збереження даних. У випадку збоїв все повинно зберігатися автоматично.

2.5. Вимоги до інформаційної і програмної сумісності

Для розробки системи використовуються бібліотеки Python, такі як python-telegram-bot, а також SQLite як легка вбудована база даних. Платформа розробки – Visual Studio Code.

2.6. Спеціальні вимоги

Жодних специфічних вимог до апаратного чи програмного забезпечення немає.

3. СКЛАД І ЗМІСТ РОБІТ ПО СТВОРЕННЮ СИСТЕМИ

Розробка системи проводиться у три основні етапи:

— Технічне завдання – складання й затвердження всіх вимог до системи, підготовка плану роботи;
— Проєктування та розробка – програмування та налагодження, підготовка документації;
— Захист проєкту – підготовка презентації та захист перед комісією.

4. ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ СИСТЕМИ

Курсовий проєкт приймається після перевірки на повноту й правильність реалізації всіх функцій. Оцінювання проводиться за такими критеріями:

— відповідність технічним вимогам;
— повнота реалізованої функціональності;
— зручність користувацького інтерфейсу;
— якість виконання діаграм та схем бази даних;
— надійність та стабільність роботи програми.

Захист проєкту проводиться перед комісією, де оцінюється теоретична та практична частини проєкту, підготовлена презентація й відповіді на запитання комісії.

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМИ

```
import logging
import sqlite3
from telegram import Update, InputFile
from telegram.ext import Updater, CommandHandler, MessageHandler, Filters,
ConversationHandler, CallbackContext

# Налаштування логування
logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
level=logging.INFO)
logger = logging.getLogger(__name__)

# Стан діалогу
CAR_BRAND, CAR_MODEL, CAR_PRICE, CAR_YEAR, CAR_DESCRIPTION,
CAR_PHONE, CAR_PHOTO, DELETE_ID = range(8)

def start(update: Update, context: CallbackContext) -> int:
    update.message.reply_text("Вітаємо! Використайте команду /add для додавання
оголошення або /delete для видалення оголошення.")
    return ConversationHandler.END

def add_car(update: Update, context: CallbackContext) -> int:
    update.message.reply_text("Введіть марку автомобіля:")
    return CAR_BRAND

def receive_brand(update: Update, context: CallbackContext) -> int:
    context.user_data['brand'] = update.message.text
    update.message.reply_text("Введіть модель автомобіля:")
    return CAR_MODEL

def receive_model(update: Update, context: CallbackContext) -> int:
    context.user_data['model'] = update.message.text
    update.message.reply_text("Введіть ціну автомобіля:")
    return CAR_PRICE

def receive_price(update: Update, context: CallbackContext) -> int:
    context.user_data['price'] = update.message.text
    update.message.reply_text("Введіть рік випуску автомобіля:")
    return CAR_YEAR

def receive_year(update: Update, context: CallbackContext) -> int:
    context.user_data['year'] = update.message.text
    update.message.reply_text("Додайте короткий опис:")
    return CAR_DESCRIPTION

def receive_description(update: Update, context: CallbackContext) -> int:
    context.user_data['description'] = update.message.text
    update.message.reply_text("Введіть номер телефону:")
    return CAR_PHONE

def receive_phone(update: Update, context: CallbackContext) -> int:
```

```
            context.user_data['phone'] = update.message.text
            update.message.reply_text("Надішліть фото автомобіля або надішліть /skip, щоб
пропустити.")
            return CAR_PHOTO

        def receive_photo(update: Update, context: CallbackContext) -> int:
            context.user_data['photo'] = update.message.photo[-1].file_id  # Отримуємо ID фото
            save_car_to_db(context)
            update.message.reply_text("Оголошення успішно створено!")
            return ConversationHandler.END

        def skip_photo(update: Update, context: CallbackContext) -> int:
            save_car_to_db(context)
            update.message.reply_text("Оголошення успішно створено!")
            return ConversationHandler.END

        def save_car_to_db(context: CallbackContext):
            conn = sqlite3.connect('cars.db')
            cursor = conn.cursor()

            # Збереження даних автомобіля
            cursor.execute("INSERT INTO cars (brand, model, price, year, description, phone, photo)
VALUES (?, ?, ?, ?, ?, ?, ?)",
                           (context.user_data['brand'],
                            context.user_data['model'],
                            context.user_data['price'],
                            context.user_data['year'],
                            context.user_data['description'],
                            context.user_data['phone'],
                            context.user_data.get('photo')))  # Використовуємо get, щоб уникнути KeyError
            conn.commit()
            conn.close()

        def list_cars(update: Update, context: CallbackContext) -> None:
            logger.info("Отримую список автомобілів...")
            conn = sqlite3.connect('cars.db')
            cursor = conn.cursor()
            cursor.execute("SELECT rowid, brand, model, price, year, description, phone, photo FROM
cars")  # Додаємо поле photo

            cars = cursor.fetchall()
            conn.close()

            if not cars:
                update.message.reply_text("Оголошень немає.")
                logger.info("Оголошень немає.")
                return

            for car in cars:
                rowid, brand, model, price, year, description, phone, photo_id = car
                message = (f"ID: {rowid}\nМарка: {brand}, Модель: {model}, Ціна: {price}, "
                           f"Рік: {year}, Опис: {description}, Телефон: {phone}\n")
```

```
                update.message.reply_text(message)
                if photo_id:  # Якщо є фото, відправляємо його
                    update.message.reply_photo(photo_id)

            logger.info("Список автомобілів успішно надіслано.")

        def delete_car(update: Update, context: CallbackContext) -> int:
            update.message.reply_text("Введіть ID оголошення, яке ви хочете видалити:")
            return DELETE_ID

        def confirm_delete(update: Update, context: CallbackContext) -> int:
            car_id = update.message.text
            conn = sqlite3.connect('cars.db')
            cursor = conn.cursor()

            cursor.execute("DELETE FROM cars WHERE rowid=?", (car_id,))
            conn.commit()

            if cursor.rowcount > 0:
                update.message.reply_text(f"Оголошення з ID {car_id} успішно видалено.")
            else:
                update.message.reply_text(f"Оголошення з ID {car_id} не знайдено.")

            conn.close()
            return ConversationHandler.END

        def main() -> None:
            # Запускаємо бота
            updater = Updater("7920334981:AAHZJONiHGhn5rpa9eicwEufq13NYOc5D0I")

            conv_handler = ConversationHandler(
                entry_points=[CommandHandler('add', add_car), CommandHandler('delete', delete_car)],
                states={
                    CAR_BRAND: [MessageHandler(Filters.text & ~Filters.command, receive_brand)],
                    CAR_MODEL: [MessageHandler(Filters.text & ~Filters.command, receive_model)],
                    CAR_PRICE: [MessageHandler(Filters.text & ~Filters.command, receive_price)],
                    CAR_YEAR: [MessageHandler(Filters.text & ~Filters.command, receive_year)],
                    CAR_DESCRIPTION: [MessageHandler(Filters.text & ~Filters.command,
receive_description)],
                    CAR_PHONE: [MessageHandler(Filters.text & ~Filters.command, receive_phone)],
                    CAR_PHOTO: [
                        MessageHandler(Filters.photo, receive_photo),
                        CommandHandler('skip', skip_photo)  # Можливість пропустити фото
                    ],
                    DELETE_ID: [MessageHandler(Filters.text & ~Filters.command, confirm_delete)],
                },
                fallbacks=[CommandHandler('list', list_cars)],  # Додайте обробник для команди /list
            )

            updater.dispatcher.add_handler(conv_handler)
            updater.dispatcher.add_handler(CommandHandler('start', start))
            updater.dispatcher.add_handler(CommandHandler('list', list_cars))
```

```
    updater.start_polling()
    updater.idle()

if __name__ == '__main__':
    main()
```

QR-код посилання на github з файлами курсового проєкту