Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему «Розроблення мобільного додатку для здійснення замовлень засобами мови програмування Python»

Виконав: студент групи ІПЗ19-1
Спеціальність 121 Інженерія програмного забезпечення
Гетьман Олександр Геннадійович
(прізвище та ініціали)
Керівник к.т.н., доц. Чупілко Т. А.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент Університет митної справи та фінансів
(місце роботи)
В.о. завідувача кафедри кібербезпеки та інформаційних технологій
(посада)
к.т.н., доц. Прокопович-Ткаченко Д.І.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2023

АНОТАЦІЯ

Гетьман О.Г. Розроблення мобільного додатку для здійснення замовлень засобами мови програмування Python.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2023.

Кваліфікаційна робота описує процес розробки мобільного додатка для здійснення замовлень з використанням мови програмування Python. Додаток пройшов кроки аналізу вимог, проектування інтерфейсу користувача, розробки функціональності, забезпечення безпеки, тестування та вдосконалення, а також оптимізації та покращення продуктивності.

Після аналізу вимог було проведено проектування інтерфейсу користувача. З метою забезпечення зручності та інтуїтивно зрозумілого використання, був розроблений екран, створена навігаційна структура, вибрана кольорова схема. Проектування інтерфейсу спрямоване на полегшення взаємодії користувача з додатком та на ефективне відображення інформації.

Після проектування інтерфейсу розпочалась розробка функціональності додатка. Цей етап включав реалізацію можливостей замовлення товарів або послуг, обробку платежів, управління користувачами, збереження замовлень у базі даних та інші функції. Для реалізації функціональності була використана бібліотека Kivy.

Після розробки додатка важливим етапом було тестування. Воно допомогло виявити можливі помилки та недоліки, які були виправлені.

Ключові слова: проектування інтерфейсу користувача, розробка функціональності, безпека, тестування, мобільний додаток, замовлення, мова програмування Python, функціональні вимоги, нефункціональні вимоги, користувачі, конкурентне середовище, навігаційна структура, кольорова схема, Kivy, SQLite.

ABSTRACT

Hetman O. G. Development of a mobile application for ordering using the Python programming language.

Qualification work for a bachelor's degree in specialty 121 «Software Engineering». – University of Customs and Finance, Dnipro, 2023.

The qualification work describes the process of developing a mobile application for placing orders using the Python programming language. The application went through the steps of requirements analysis, user interface design, functionality development, security, testing and improvement, as well as performance optimization and improvement.

After the requirements analysis, the user interface was designed. In order to ensure convenient and intuitive use, a screen was designed, a navigation structure was created, and a color scheme was chosen. UI design is aimed at facilitating user interaction with the application and efficient display of information.

After designing the interface, the development of the application's functionality began. This stage included the implementation of the ability to order goods or services, process payments, manage users, save orders in the database, and other functions. Various Python libraries and frameworks, such as Kivy, PyQt, or Django, were used to implement the functionality.

After developing the application, testing was an important stage. It helped to identify possible errors and shortcomings, which were corrected.

Keywords: user interface design, functionality development, security, testing, mobile application, ordering, Python programming language, functional requirements, non-functional requirements, users, competitive environment, navigation structure, color scheme, Kivy, SQLite.

ЗМІСТ

ВСТУП

У сучасному світі мобільні додатки стали невід'ємною частиною нашого повсякденного життя. Вони забезпечують нам широкий спектр можливостей, спрощують наші щоденні завдання і покращують комунікацію між користувачами. У зв'язку з цим розробка мобільних додатків стала актуальною і важливою галуззю програмування.

Актуальність роботи полягає в наступних аспектах:

1) зростання популярності мобільних додатків. За останні десятиліття мобільні пристрої, такі як смартфони та планшети, стали необхідними аксесуарами для багатьох людей. За даними статистики, кількість користувачів мобільних додатків постійно зростає, що створює великі можливості для розробників нових та інноваційних додатків;

2) зручність та ефективність. Розробка мобільного додатка для замовлень засобами мови програмування Python має потенціал спростити процес замовлення товарів або послуг для користувачів. Вони зможуть здійснювати замовлення швидко та зручно, використовуючи свої мобільні пристрої з будь-якого місця та в будь-який час;

3) зростання електронної комерції. Електронна комерція постійно зростає, а з нею зростає потреба у зручних способах здійснення замовлень. Мобільні додатки стали популярними серед покупців, оскільки вони дозволяють здійснювати покупки швидко та зручно, без необхідності відвідувати фізичні магазини. Розробка мобільного додатка для замовлень є актуальним завданням в контексті розширення електронної комерції;

4) розширення можливостей мови програмування Python. Python є однією з найпопулярніших мов програмування, яка використовується в багатьох сферах, включаючи розробку мобільних додатків. Розробка мобільного додатка з використанням Python дозволяє використовувати всі переваги цієї мови, такі як простота, читабельність та великий вибір бібліотек.

Отже, розробка мобільного додатка для здійснення замовлень засобами мови програмування Python є актуальною і важливою темою в сучасному світі. Вона відповідає зростаючим потребам користувачів мобільних додатків та розвитку електронної комерції. Крім того, розробка мобільного додатка з використанням Python дозволить розширити можливості цієї мови та застосувати її в новій сфері.

Метою роботи є розроблення мобільного додатка для здійснення замовлень засобами мови програмування Python.

Для досягнення поставленої мети розробки мобільного додатка для здійснення замовлень засобами мови програмування Python ставляться наступні задачі:

1) аналіз вимог. Спочатку необхідно провести детальний аналіз вимог до мобільного додатка. Це включає вивчення функціональних та нефункціональних вимог, розуміння потреб користувачів, аналіз конкурентного середовища та визначення основних функцій та можливостей, які має мати додаток;

2) проектування інтерфейсу користувача. Важливим кроком є проектування інтерфейсу, який буде зручним та інтуїтивно зрозумілим для користувачів. Це включає розробку екранів, створення навігаційної структури, вибір кольорової схеми та розміщення елементів у додатку. Проектування інтерфейсу має забезпечити зручність взаємодії користувача з додатком та відображення інформації;

3) розробка функціональності. Після проектування інтерфейсу необхідно розробити функціональність додатка. Це включає реалізацію можливостей замовлення товарів або послуг, обробку платежів, управління користувачами, збереження замовлень у базі даних тощо. Для цього можуть використовуватися різні бібліотеки та фреймворки Python, такі як Kivy, PyQt або Django;

4) забезпечення безпеки. Безпека є важливим аспектом будь-якого мобільного додатка, особливо тоді, коли мова йде про замовлення товарів або

послуг. Для забезпечення безпеки необхідно розглянути питання аутентифікації користувачів, захисту особистих даних, безпечної передачі інформації та захисту від можливих кібератак;

5) тестування та вдосконалення. Після розробки мобільного додатка важливо провести тестування, яке допоможе виявити та виправити можливі помилки або недоліки. Також може бути потрібно збирати фідбек від користувачів, щоб покращити функціональність та досконалість додатка;

6) оптимізація та покращення продуктивності. Для забезпечення швидкої та ефективної роботи додатка можуть бути необхідні оптимізаційні заходи, такі як покращення швидкості завантаження, оптимізація запитів до бази даних або кешування даних. Це допоможе забезпечити задоволення користувачів та зменшити можливі проблеми з продуктивністю.

Виконання цих задач допоможе досягти поставленої мети розробки мобільного додатка для здійснення замовлень засобами мови програмування Python та забезпечити користувачам зручний та безпечний спосіб здійснення замовлень через мобільні пристрої.

Об’єктом дослідження є процеси роботи мобільного додатка для здійснення замовлень засобами мови програмування Python.

Предметом дослідження є апаратно-програмне забезпечення для розроблення мобільного додатка для здійснення замовлень засобами мови програмування Python.

Практичне значення одержаних результатів полягає у підвищенні якості здійснення замовлень засобами мови програмування Python.

Результатами роботи мобільний додаток для здійснення замовлень засобами мови програмування Python.

Кваліфікаційна робота складається зі вступу, 3-х розділів, висновків, використаних джерел з 20 найменувань, 1-го додатка.

Обсяг роботи складається з 50 сторінок основного тексту з 68 сторінок кваліфікаційної роботи та 19 рисунків.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Опис функціональних та нефункціональних вимог

Вивчення функціональних та нефункціональних вимог до додатка є важливим етапом у розробці мобільного додатка для здійснення замовлень засобами мови програмування Python. Воно дозволяє зрозуміти потреби користувачів та визначити основні функції та можливості, які має мати додаток.

Функціональні вимоги описують, які задачі та функції повинен виконувати додаток. Вони визначають основні функції, які потрібні користувачам для здійснення замовлень.

Деякі приклади функціональних вимог можуть включати:

1) реєстрація та аутентифікація користувачів. Додаток повинен мати можливість реєстрації нових користувачів та аутентифікації вже зареєстрованих користувачів;

2) перегляд каталогу товарів або послуг. Користувач повинен мати можливість переглядати доступний асортимент товарів або послуг, які можна замовити;

3) додавання товарів до кошика. Користувач повинен мати можливість додавати обрані товари до кошика, який відображається на екрані;

4) замовлення товарів. Користувач повинен мати можливість здійснювати замовлення товарів або послуг, вказуючи необхідну кількість, розмір, адресу доставки тощо;

5) оплата замовлення. Додаток повинен надавати можливість здійснити оплату за замовлення, використовуючи різні методи оплати, такі як кредитна карта, електронний гаманець тощо;

6) відстеження статусу замовлення. Користувач повинен мати можливість переглядати статус своїх замовлень, таких як «в обробці», «відправлено», «доставлено» тощо.

Нефункціональні вимоги визначають характеристики, які повинен мати додаток, але не стосуються конкретних функцій. Вони описують вимоги до продуктивності, безпеки, надійності та інших аспектів додатка.

Деякі приклади нефункціональних вимог можуть включати:

1) продуктивність. Додаток повинен працювати швидко та ефективно, з незначною затримкою при завантаженні сторінок чи обробці замовлень;

2) безпека. Додаток повинен забезпечувати захист персональних даних користувачів, таких як інформація про платіжні картки або особисту інформацію;

3) сумісність. Додаток повинен бути сумісним з різними мобільними пристроями та операційними системами, такими як Android та iOS;

4) інтуїтивний інтерфейс користувача. Додаток повинен мати простий та зрозумілий інтерфейс, що дозволяє легко навігувати та здійснювати замовлення без складнощів;

5) надійність. Додаток повинен бути стабільним та надійним, з мінімальною кількістю збоїв або падінь.

Вивчення функціональних та нефункціональних вимог допомагає уточнити обсяг та характеристики додатка, враховуючи потреби користувачів та вимоги предметної області. Це дозволяє розробникам належним чином спланувати та реалізувати функціонал, що задовольняє очікування користувачів та вимоги проекту.

1.2 Аналіз потреб користувачів при розробці мобільного додатка для здійснення замовлень

Детальне вивчення потреб користувачів є ключовим етапом при розробці мобільного додатка для здійснення замовлень засобами мови

програмування Python. Розуміння потреб та вимог користувачів дозволяє забезпечити функціонал, який задовольняє їх очікування та полегшує використання додатка.

Основні аспекти потреб користувачів включають:

1) зручність використання. Користувачі хочуть мати інтуїтивно зрозумілий та легкий у використанні інтерфейс, що дозволяє легко здійснювати замовлення. Вони оцінюють зручність навігації, простоту процесу замовлення та доступність основних функцій;

2) персоналізація. Користувачі цінують можливість налаштовувати додаток під свої індивідуальні потреби. Наприклад, вони можуть бажати зберігати свої адреси доставки, обирати фільтри для пошуку товарів або налаштовувати сповіщення;

3) швидкість та ефективність. Користувачі очікують, що додаток буде працювати швидко та без затримок. Вони бажають мінімізувати час очікування при завантаженні сторінок, обробці замовлень та взаємодії з додатком;

4) надійність та безпека. Користувачі хочуть мати впевненість в тому, що їхні дані та фінансова інформація захищені. Вони очікують, що додаток буде стійким до помилок та збоїв, а також надаватиме безпечні методи оплати та передачі даних;

5) інформаційна достовірність. Користувачі вірять важливості отримання достовірної та актуальної інформації про товари або послуги, таку як опис, ціна, наявність тощо. Вони хочуть мати можливість переглядати відгуки, рейтинги та іншу інформацію, що допомагає їм приймати розумні рішення щодо замовлення;

6) підтримка та зворотний зв'язок. Користувачі цінують наявність каналів підтримки та можливість звертатися до команди розробників з питаннями, проблемами чи пропозиціями. Вони бажають, щоб їхні питання чи скарги були вирішені оперативно та ефективно.

Детальне дослідження потреб користувачів допомагає розробникам забезпечити функціонал та особливості додатка, які відповідають їхнім очікуванням та вимогам. Врахування цих потреб у процесі розробки допомагає створити додаток, який забезпечує задоволення та високу користувальницьку придатність.

1.3 Визначення основних функцій та можливостей мобільного додатка для здійснення замовлень

Визначення основних функцій та можливостей додатка є важливим етапом у розробці мобільного додатка для здійснення замовлень засобами мови програмування Python. Цей етап вимагає ретельного аналізу потреб користувачів та врахування особливостей предметної області.

Реєстрація та аутентифікація користувачів:

– можливість створення нового облікового запису користувача;

– введення особистих даних, таких як ім'я, адреса, контактна інформація;

– механізм підтвердження облікового запису (наприклад, електронна пошта або SMS-повідомлення);

– аутентифікація користувача для доступу до функціоналу додатка.

Перегляд каталогу товарів або послуг:

– відображення списку доступних товарів або послуг;

– сортування та фільтрація за різними параметрами (ціна, категорія, рейтинг тощо);

– відображення основної інформації про кожен товар або послугу, такої як назва, опис, фотографія.

Додавання товарів до кошика:

– можливість додавання товарів або послуг до кошика під час перегляду каталогу;

– керування кількістю одиниць товару у кошику;

– перегляд загальної суми та деталей замовлення у кошику.

Замовлення товарів:

– вибір товарів з кошика для замовлення;

– введення необхідних даних для доставки, таких як адреса, контактна інформація;

– вибір методу оплати та введення відповідних даних;

– підтвердження замовлення та отримання підтвердження про успішну обробку.

Оплата замовлень:

– інтеграція з платіжними системами для забезпечення безпечних та зручних методів оплати;

– обробка платіжних транзакцій та отримання підтвердження про успішну оплату.

Відстеження статусу замовлень:

– надання користувачеві можливості відстежувати статус своїх замовлень;

– відображення оновлення статусу, таких як «прийнято», «виконується», «відправлено», «доставлено».

Відгуки та рейтинги:

– можливість користувачам залишати відгуки та оцінювати товари або послуги;

– відображення рейтингу та відгуків інших користувачів для допомоги прийняття рішення про покупку.

Повідомлення та сповіщення:

– можливість отримання повідомлень про статус замовлення, акції або спеціальні пропозиції;

– відображення сповіщень про нові повідомлення або оновлення.

Підтримка клієнтів:

– надання каналів зв’язку для клієнтів, таких як чат або телефонна підтримка;

– обробка запитів, скарг та запитів на допомогу від користувачів.

Визначення основних функцій та можливостей додатка базується на вивченні потреб користувачів, аналізі конкурентного середовища та вимог предметної області. Детальне розуміння цих функцій допомагає розробникам створити додаток, який забезпечує бажаний функціонал та надає зручний та задовольняючий досвід користувачам.

1.4 Огляд бібліотек та фреймворків мови програмування Python для розробки мобільних додатків

Існує кілька бібліотек та фреймворків Python, які забезпечують можливості розробки мобільних додатків:

1) Kivy (рис. 1.1). Kivy є відкритим крос-платформовим фреймворком для розробки мобільних додатків з графічним інтерфейсом користувача. Він підтримує різні платформи, включаючи Android, iOS, Windows, macOS і Linux. Kivy використовує спеціальну мову розмітки, що називається Kivy Language (KV), для опису інтерфейсу користувача. Цей фреймворк має вбудовану підтримку для мультимедіа, анімації та багатошарових додатків;

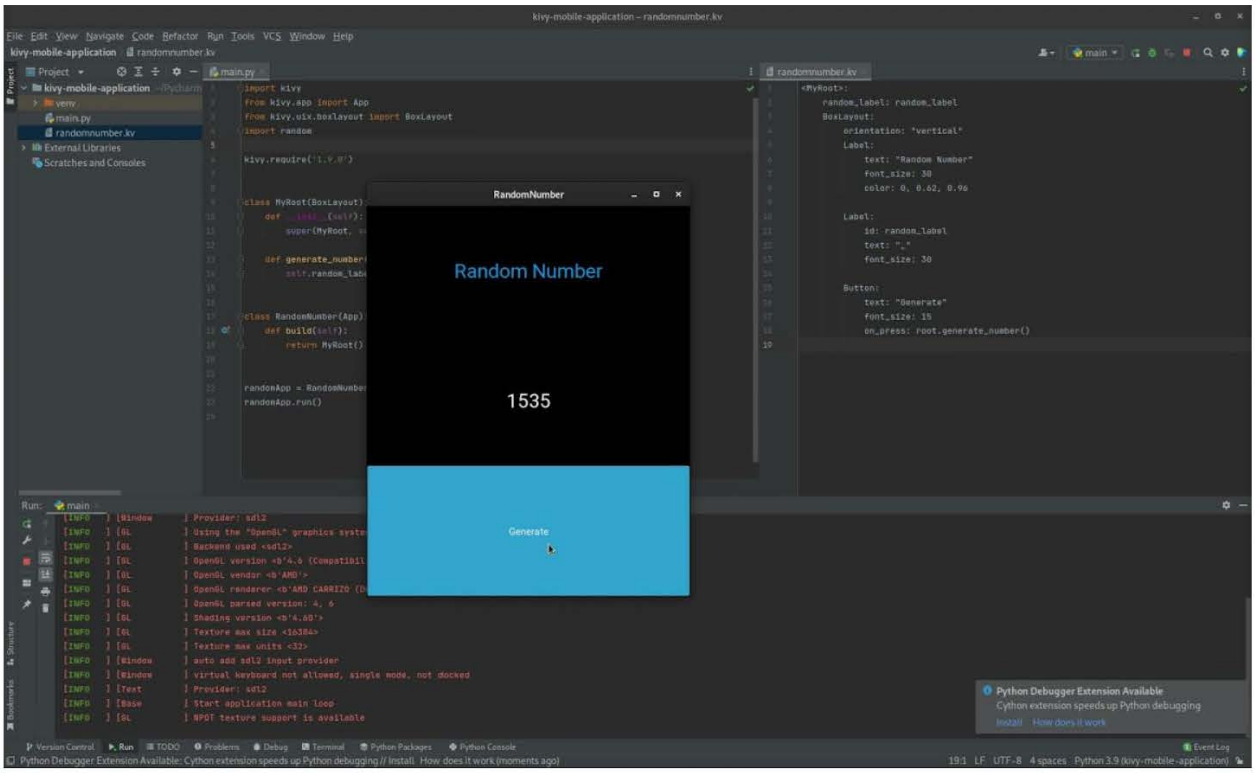

Рисунок 1.1 – Приклад використання Kivy для створення програмних застосунків

2) BeeWare (рис. 1.2). BeeWare є набором інструментів для розробки крос-платформових мобільних додатків, включаючи Android, iOS, Windows, macOS і Linux. Він дозволяє розробникам використовувати Python для створення нативних додатків, що мають повний доступ до можливостей операційної системи. BeeWare включає фреймворк Toga для створення графічних інтерфейсів та ряд інших інструментів для розробки мобільних додатків;

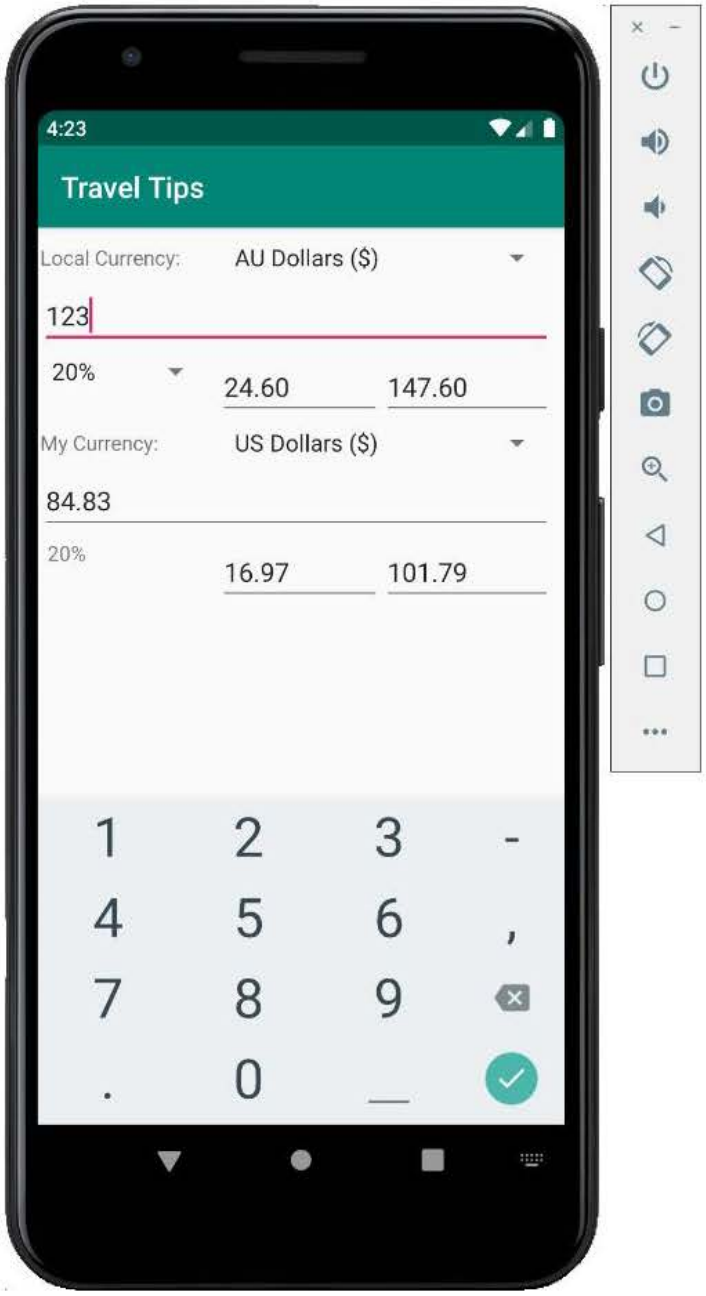

Рисунок 1.2 – Програмний застосунок, розроблений з використанням BeeWare

3) Pygame. Pygame це бібліотека Python для розробки ігор, яка також може бути використана для створення мобільних додатків з графічним інтерфейсом. Вона надає різні функції для роботи з графікою, звуком, анімацією та управлінням подіями. Pygame дозволяє розробникам створювати прості та серйозні ігри для різних мобільних платформ;

4) Flask та Django. Flask та Django є популярними веб-фреймворками Python, які можуть бути використані для створення мобільних додатків з веб-серверною частиною. Flask є легким та гнучким фреймворком, в той час як Django надає більш повні функціональність для розробки складних додатків. Обидва фреймворки підтримують розробку API, що дозволяє мобільним додаткам взаємодіяти з сервером для отримання та обробки даних;

5) Plyer. Plyer є крос-платформовою бібліотекою, яка надає простий інтерфейс для доступу до функціональності мобільних пристроїв, таких як камера, геолокація, повідомлення тощо. Вона підтримує різні мобільні платформи і дозволяє розробникам легко інтегрувати такі функції в свої мобільні додатки.

Це лише кілька прикладів бібліотек та фреймворків Python для розробки мобільних додатків. Залежно від потреб і вимог проекту, може бути вибраний певний інструмент для реалізації мобільного додатка на базі Python.

1.5 Висновки до першого розділу. Постановка задач дослідження

Розділ 1 «Дослідження предметної області. Постановка завдань дослідження» містить наступні підрозділи:

У підрозділі 1.1 наводиться детальний опис функціональних вимог до мобільного додатка для здійснення замовлень. Це можуть бути такі функції, як реєстрація користувачів, перегляд доступних продуктів, створення та оплата замовлень тощо. Також розглядаються нефункціональні вимоги, такі як продуктивність, надійність, безпека тощо. Описуються основні вимоги, які впливають на розробку додатка.

У підрозділі 1.2 проводиться аналіз потреб та вимог користувачів, що стосуються мобільного додатка. Розглядаються основні сценарії використання додатка, вимоги до зручності використання, інтерфейсу та функціональності. Також враховуються фактори, які можуть впливати на задоволення потреб користувачів і покращення їх досвіду використання додатка.

У підрозділі 1.3 конкретизується перелік основних функцій та можливостей, які будуть доступні в розроблюваному мобільному додатку. Розглядаються такі аспекти, як авторизація користувачів, перегляд каталогу товарів, додавання товарів у кошик, розміщення та оплата замовлення тощо. Визначаються основні функціональні можливості, які забезпечують задоволення потреб користувачів.

У підрозділі 1.4 проводиться огляд різних бібліотек та фреймворків мови програмування Python, які можуть бути використані для розробки мобільних додатків. Зазначаються їх особливості, переваги та недоліки, можливості і інтеграція з мобільними платформами. Зокрема, описується фреймворк Kivy, який є одним з варіантів для розробки мобільних додатків з використанням Python.

У підрозділі 1.5 формулюються висновки до першого розділу роботи. Підсумовуються отримані результати, виділяються ключові вимоги та функціональні можливості мобільного додатка, а також обговорюється вибір фреймворку Kivy для розробки додатка. Формулюються постановка задач дослідження, які будуть розглянуті у наступних розділах роботи.

Метою роботи є розроблення мобільного додатка для здійснення замовлень засобами мови програмування Python.

Для досягнення поставленої мети розробки мобільного додатка для здійснення замовлень засобами мови програмування Python ставляться наступні задачі:

1) аналіз вимог. Спочатку необхідно провести детальний аналіз вимог до мобільного додатка. Це включає вивчення функціональних та нефункціональних вимог, розуміння потреб користувачів, аналіз конкурентного середовища та визначення основних функцій та можливостей, які має мати додаток;

2) проектування інтерфейсу користувача. Важливим кроком є проектування інтерфейсу, який буде зручним та інтуїтивно зрозумілим для користувачів. Це включає розробку екранів, створення навігаційної структури,

вибір кольорової схеми та розміщення елементів у додатку. Проектування інтерфейсу має забезпечити зручність взаємодії користувача з додатком та відображення інформації;

3) розробка функціональності. Після проектування інтерфейсу необхідно розробити функціональність додатка. Це включає реалізацію можливостей замовлення товарів або послуг, обробку платежів, управління користувачами, збереження замовлень у базі даних тощо. Для цього можуть використовуватися різні бібліотеки та фреймворки Python, такі як Kivy, PyQt або Django;

4) забезпечення безпеки. Безпека є важливим аспектом будь-якого мобільного додатка, особливо тоді, коли мова йде про замовлення товарів або послуг. Для забезпечення безпеки необхідно розглянути питання аутентифікації користувачів, захисту особистих даних, безпечної передачі інформації та захисту від можливих кібератак;

5) тестування та вдосконалення. Після розробки мобільного додатка важливо провести тестування, яке допоможе виявити та виправити можливі помилки або недоліки. Також може бути потрібно збирати фідбек від користувачів, щоб покращити функціональність та досконалість додатка;

6) оптимізація та покращення продуктивності. Для забезпечення швидкої та ефективної роботи додатка можуть бути необхідні оптимізаційні заходи, такі як покращення швидкості завантаження, оптимізація запитів до бази даних або кешування даних. Це допоможе забезпечити задоволення користувачів та зменшити можливі проблеми з продуктивністю.

РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ РОЗРОБЛЕННЯ МОБІЛЬНОГО ДОДАТКА ДЛЯ ЗДІЙСНЕННЯ ЗАМОВЛЕНЬ ЗАСОБАМИ МОВИ ПРОГРАМУВАННЯ PYTHON

2.1 Вибір програмних засобів для розроблення мобільного додатка для здійснення замовлень засобами мови програмування Python

Для розробки мобільного додатка для здійснення замовлень мовою програмування Python знадобляться наступні технології:

1) Kivy. Kivy це відкрита бібліотека Python, яка дає змогу створювати мультимедійні додатки з використанням різноманітних користувацьких інтерфейсів, включно з мобільними додатками. Він надає інструменти для створення графічного інтерфейсу користувача (GUI) та керування подіями (рис. 2.1);

2) SQLite. SQLite це вбудована реляційна база даних, яка є частиною стандартної бібліотеки Python. Вона надає легкий і зручний спосіб збереження та вилучення даних у додатку;

3) KivyMD. KivyMD це надбудова над Kivy, яка надає набір готових віджетів і стилів, щоб спростити розробку додатків у стилі Material Design. Вона надає більш сучасний та естетично приємний зовнішній вигляд для мобільних додатків;

4) Python. Python це мова програмування, якою буде написано додаток. Вона має простий синтаксис і безліч бібліотек, що робить її зручним вибором для розроблення додатків.

Таким чином, основні технології, які знадобляться для розробки цього додатка, включають Kivy, SQLite, KivyMD і мову програмування Python.

```
import kivy
kivy.require('1.11.1')


import sqlite3
from kivy.app import App
from kivy.uix.label import Label
from kivy.uix.button import Button
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.textinput import TextInput
from kivy.uix.popup import Popup
from kivy.uix.checkbox import CheckBox
from kivy.uix.spinner import Spinner
from kivy.uix.scrollview import ScrollView


class OrderApp(App):
    def __init__(self):
        super().__init__()
        self.connection = None
        self.cursor = None

    def build(self):
        layout = BoxLayout(
            orientation='vertical',
            padding='10sp',
            spacing='10sp'
        )

        label = Label(
            text='Ласкаво просимо до програми замовлень!',
            font_size='24sp'
        )
        layout.add_widget(label)

        name_input = TextInput(
            hint_text='Введіть ваше ім\'я',
```

Рисунок 2.1 – Приклад розробки застосунку за допомогою Kivy

2.2 Огляд мови програмування Python

2.2.1 Основні особливості

Python – це високорівнева, інтерпретована мова програмування, яка відома своєю простотою та зрозумілістю синтаксису.

Основні особливості мови Python включають:

1) читабельний та зрозумілий синтаксис. Синтаксис Python дуже зрозумілий та легко читається, що робить його придатним для початківців і досвідчених програмістів. Він використовує простий блок-структурований стиль кодування, що полегшує розуміння програми;

2) динамічна типізація. Python є динамічно типізованою мовою, що означає, що типи змінних визначаються автоматично в процесі виконання програми. Розробникам не потрібно вказувати тип змінних явно, що спрощує процес програмування;

3) об'єктно-орієнтоване програмування. Python підтримує об'єктно-орієнтоване програмування (ООП). Він дозволяє створювати класи, об'єкти, спадкування, поліморфізм та інші принципи ООП, що сприяє структуруванню та повторному використанню коду;

4) багатий набір стандартних бібліотек. Python поставляється з великою кількістю стандартних бібліотек, які містять готові модулі для виконання різноманітних завдань, таких як робота з файлами, мережеве програмування, обробка рядків, математичні обчислення та багато іншого. Це дозволяє розробникам швидко виконувати різноманітні завдання без необхідності писати код з нуля;

5) розширені можливості. Python має багатий екосистему зовнішніх бібліотек та модулів, які дозволяють розширювати його функціональність. Наприклад, бібліотека NumPy дозволяє виконувати обчислення на великих масивах даних, бібліотека Pandas надає зручні засоби для обробки та аналізу даних, а бібліотека TensorFlow дозволяє виконувати машинне навчання та глибинне навчання;

6) переносимість. Python є переносимою мовою, що означає, що програми, написані на Python, можуть запускатися на різних операційних системах, таких як Windows, macOS, Linux, без необхідності вносити зміни в код;

7) простота інтеграції. Python легко інтегрується з іншими мовами програмування, такими як C, C++, Java, що дозволяє розробникам

використовувати різні інструменти та бібліотеки, щоб отримати найкращі результати.

Ці основні особливості мови Python роблять її популярною серед розробників. Вона використовується в різних областях, таких як веб-розробка, наукові дослідження, аналіз даних, штучний інтелект та багато іншого.

2.2.2 Переваги використання мови програмування Python для мобільної розробки

Переваги використання мови програмування Python для мобільної розробки полягають у таких ключових аспектах:

1) простота та зрозумілість. Python має простий та лаконічний синтаксис, який легко читається та розуміється. Це полегшує розробку, тестування та обслуговування мобільних додатків. Розробники можуть швидко освоїти мову та розпочати роботу над проектом;

2) широкий вибір фреймворків та бібліотек. Python має багатий екосистему фреймворків та бібліотек для мобільної розробки. Наприклад, фреймворк Kivy дозволяє створювати крос-платформові мобільні додатки з графічним інтерфейсом користувача. Фреймворк Django розширює можливості Python для створення мобільних додатків з веб-серверною частиною. Бібліотека Pygame надає інструменти для розробки ігор для мобільних платформ. Ці фреймворки та бібліотеки забезпечують розробникам гнучкість та продуктивність у розробці мобільних додатків;

3) крос-платформовість. Python підтримує крос-платформову розробку, що означає, що один і той самий код може бути використаний для створення додатків для різних мобільних платформ, таких як Android та iOS. Це дозволяє розробникам ефективно використовувати свій час та ресурси, не потребуючи окремого написання коду для кожної платформи;

4) велика спільнота та підтримка. Python має активну та розширену спільноту розробників, яка постійно вносить внесок у розвиток мови. Існує

багато ресурсів, таких як документація, форуми, статті та блоги, які допомагають розробникам вирішувати проблеми та знаходити відповіді на свої запитання. Це забезпечує підтримку та доступ до потужних інструментів для мобільної розробки;

5) інтеграція з існуючими системами та сервісами. Python легко інтегрується з різними системами та сервісами, що полегшує розробку мобільних додатків. Наприклад, Python може взаємодіяти з базами даних, веб-серверами, API сторонніх сервісів та іншими компонентами. Це дозволяє розробникам створювати додатки, які взаємодіють з різними джерелами даних та функціонують у комплексних середовищах;

6) масштабованість. Python дозволяє розробникам створювати масштабовані мобільні додатки. Завдяки підтримці асинхронного програмування та багатопотоковості, Python може обробляти великі обсяги даних та виконувати складні операції без блокування інтерфейсу користувача. Це дозволяє розробникам створювати потужні та швидкодіючі мобільні додатки.

Всі ці переваги роблять мову програмування Python привабливою для мобільної розробки, допомагають розробникам створювати ефективні та функціональні мобільні додатки з меншими зусиллями та витратами часу.

2.3 Фреймворк Kivy

Kivy – це відкритий фреймворк для розробки мультимедійних додатків з графічним інтерфейсом користувача (GUI) на різних платформах, включаючи мобільні пристрої, комп'ютери та вбудовані системи. Основними особливостями Kivy є кросплатформенність, мультитач-підтримка, швидкість та простота використання (рис. 2.2).

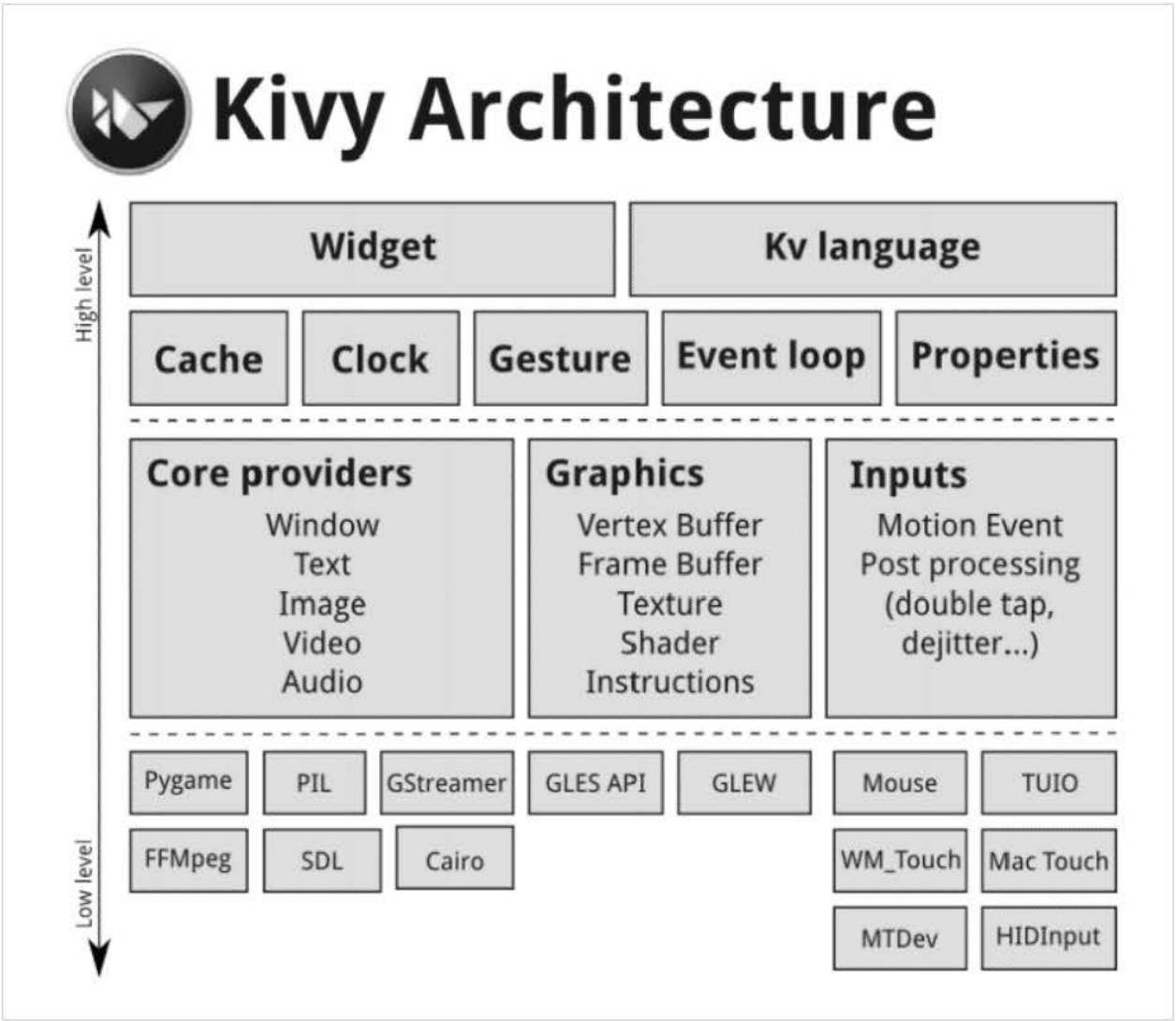

Рисунок 2.2 – Архітектура Kivy

Основні компоненти Kivy:

1) графічний движок. Kivy має вбудований графічний движок, який використовує різні методи для відображення графічних об'єктів. Цей движок може працювати зі звичайними вікнами та елементами керування, а також підтримує анімацію, рухи жестів та інші візуальні ефекти;

2) мова розмітки Kivy (KV). Kivy використовує спеціальну мову розмітки, відому як Kivy Language або просто KV. Це дозволяє відокремити опис інтерфейсу від вихідного коду, що полегшує розробку та обслуговування. Мова KV є простою і зрозумілою, і вона дозволяє створювати складні інтерфейси з меншим обсягом коду;

3) мультимедіа підтримка. Kivy має вбудовану підтримку мультимедіа, що дозволяє легко працювати зі звуком, відео та зображеннями. Можна відтворювати аудіо та відео файли, використовувати різні ефекти звуку, обробляти та відображати зображення. Це особливо корисно для розробки додатків, пов'язаних з медіа контентом;

4) мультитач-підтримка. Kivy надає підтримку для розпізнавання та обробки багатодотикових жестів на пристроях з сенсорним екраном. Це дозволяє створювати інтерактивні додатки, які реагують на рухи пальців, жести та інші взаємодії користувача;

5) підтримка анімації. Kivy має потужні можливості анімації, що дозволяє створювати рухомі ефекти та переходи між елементами інтерфейсу. Можна використовувати різні типи анімації, включаючи переміщення, масштабування, обертання та зміну прозорості;

6) розширені можливості. Kivy надає доступ до різних функцій, які допомагають у розробці мобільних додатків. Це включає підтримку роботи з базами даних, мережеві можливості, обробку введення користувача, роботу зі зображеннями та багато іншого;

7) компоненти і керування. Kivy надає широкий набір готових компонентів, таких як кнопки, поля введення, списки, меню, слайдери та інші. Вони розроблені з урахуванням стандартних вимог для мобільних додатків, що дозволяє швидко будувати інтерфейси користувача. Крім того, Kivy надає можливість створювати власні компоненти та візуально налаштовувати їх з використанням мови KV;

8) розширюваність. Kivy побудований на модульній архітектурі, що дозволяє легко розширювати його функціональність. Можна використовувати сторонні розширення, плагіни та інші інструменти для додаткового функціоналу. Крім того, Kivy має активну спільноту розробників, яка постійно робить внески до фреймворку та надає підтримку;

9) документація та ресурси. Kivy має добре оформлену документацію, яка включає пояснення основних концепцій, приклади коду та розширені

посібники. Крім цього, існує багато відеоуроків, блогів та форумів, де можна знайти додаткову інформацію, поради та приклади використання Kivy;

10) ліцензування та вартість. Kivy розповсюджується під ліцензією MIT, що означає, що можна використовувати його безкоштовно навіть у комерційних проектах. Це робить Kivy привабливим вибором для розробки мобільних додатків, особливо для стартапів та незалежних розробників з обмеженим бюджетом.

Отже, Kivy є потужним фреймворком для розробки мобільних додатків з використанням Python. Він надає широкий набір можливостей для створення кросплатформенних додатків з графічним інтерфейсом. Завдяки своїй кросплатформенності, підтримці мультитач-взаємодії, мультимедіа та іншим функціям, Kivy стає привабливим вибором для розробників, які хочуть швидко розробити мобільний додаток з використанням Python. З великою спільнотою розробників та багатим набором документації, Kivy надає підтримку та ресурси для успішної розробки мобільних додатків.

2.4 База даних SQLite

SQLite – це легка вбудована база даних, яка зберігає дані у локальному файловому форматі. Вона пропонує реляційну модель даних і використовує стандартну мову запитів SQL для маніпуляції та керування даними. Особливістю SQLite є те, що вона не потребує окремого сервера баз даних, і її можна використовувати безпосередньо в додатках (рис. 2.3).

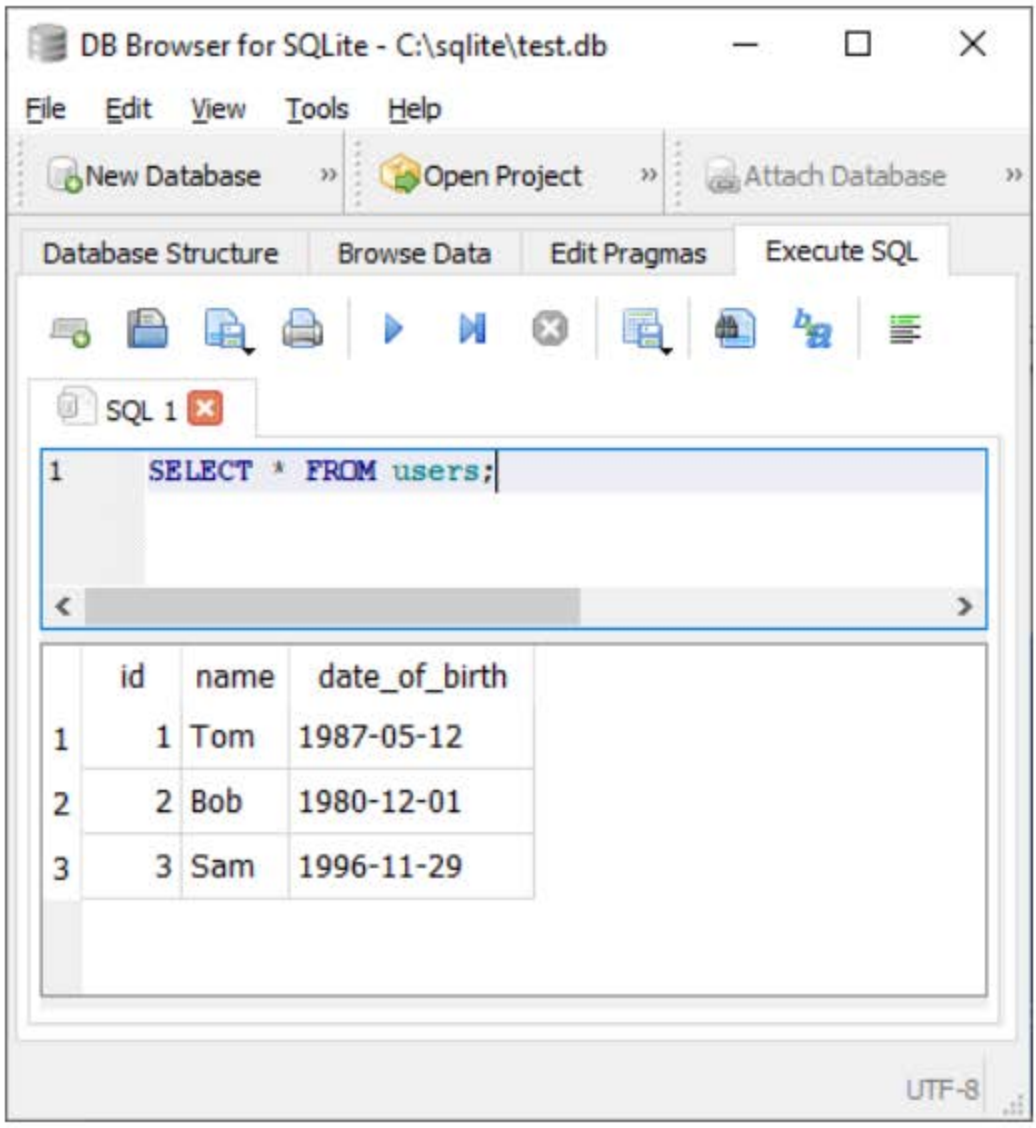

Рисунок 2.3 – Інтерфейс DB Browser for SQLite

Основні особливості SQLite:

1) легкість використання. SQLite проста у встановленні та налаштуванні. Вона не потребує складного процесу налаштування сервера баз даних або конфігурації. Можна просто включити бібліотеку SQLite у свій проект і почати використовувати базу даних;

2) вбудованість. SQLite побудована з урахуванням вбудованості, що означає, що вона може бути використана безпосередньо в додатках. Не потрібно окремо встановлювати та налаштовувати сервер баз даних. Всі дані зберігаються в одному файлі, що робить їх легкими для переміщення та резервного копіювання;

3) кросплатформеність. SQLite підтримується на багатьох платформах, включаючи Windows, macOS, Linux, iOS та Android. Це означає, що можна використовувати SQLite у своїх проектах на різних пристроях та операційних системах без зміни коду;

4) підтримка SQL. SQLite використовує мову запитів SQL для створення, зчитування, оновлення та видалення даних. Можна використовувати стандартні SQL-запити для отримання потрібної інформації з бази даних. SQLite підтримує багато SQL-операцій, включаючи створення таблиць, індексів, обмежень, з'єднань та багато іншого;

5) транзакційна безпека. SQLite має вбудовану підтримку транзакцій, що забезпечує безпеку даних. Можна використовувати транзакції для забезпечення атомарності, консистентності, ізольованості та довіреності (ACID) бази даних. Це особливо важливо при операціях, які вимагають збереження цілості даних, наприклад, при додаванні або оновленні записів;

6) підтримка широкого спектру типів даних. SQLite підтримує різноманітні типи даних, такі як цілі числа, дійсні числа, рядки, дата та час, бінарні дані та інші. Можна використовувати відповідні типи даних для збереження різних видів інформації в базі даних;

7) швидкість та ефективність. SQLite працює швидко і ефективно навіть з великими обсягами даних. Вона має оптимізовану архітектуру, що дозволяє проводити швидкі операції зчитування та запису даних;

8) розширення та плагіни. SQLite надає можливість розширення функціональності через використання власних функцій, агрегатних функцій та інших розширень. Можна створювати свої власні розширення для роботи зі специфічними типами даних або функціями.

SQLite є популярним вибором для розробки мобільних додатків, особливо для простих проектів, які потребують локального збереження даних. Вона має багатий набір функцій, ефективна та проста у використанні. Будь-який розробник, який працює з Python та має потребу у вбудованій базі даних,

може використовувати SQLite для зручного та надійного збереження даних у своїх мобільних додатках.

2.5 Вбудований модуль sqlite3 для взаємодії з базою даних SQLite

SQLite3 – це вбудовувана бібліотека Python, яка надає інтерфейс для взаємодії з базою даних SQLite. Вона дозволяє розробникам створювати, зчитувати, оновлювати та видаляти дані в базі даних SQLite з використанням мови програмування Python (рис. 2.4).

```
def open_database(self):
    self.connection = sqlite3.connect('orders.db')
    self.cursor = self.connection.cursor()

    # Створюємо таблицю orders, якщо вона не існує
    self.cursor.execute('''
        CREATE TABLE IF NOT EXISTS orders (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT,
            address TEXT,
            products TEXT,
            delivery TEXT,
            total_cost INTEGER
        )
    ''')
    self.connection.commit()

def close_database(self):
    if self.cursor:
        self.cursor.close()

    if self.connection:
        self.connection.close()
```

Рисунок 2.4 – Приклад коду для використання sqlite3

Основні особливості бібліотеки SQLite3:

1) підтримка бази даних SQLite. SQLite3 надає повний доступ до функцій SQLite, включаючи створення таблиць, індексів, транзакцій, виконання SQL-запитів та багато іншого. Вона підтримує усі можливості, які надає SQLite, і дозволяє розробникам повною мірою використовувати їх у своїх програмах;

2) простота використання. SQLite3 має простий та інтуїтивно зрозумілий інтерфейс, що дозволяє швидко освоїти його для взаємодії з базою даних. Вона надає легкі методи для створення та з'єднання з базою даних, виконання запитів, отримання результатів і роботи з транзакціями. Завдяки простоті використання, розробникам не потрібно витрачати багато часу на налаштування або вивчення складних інтерфейсів;

3) параметризовані запити. SQLite3 дозволяє використовувати параметризовані запити, що забезпечує безпеку та запобігає SQL-ін'єкціям. Розробники можуть використовувати знаки питання або іменовані параметри у своїх запитах і передавати значення параметрів окремо. Це дозволяє уникнути вразливостей, пов'язаних з некоректним введенням користувача та дозволяє більш безпечно працювати з базою даних;

4) підтримка транзакцій. SQLite3 дозволяє працювати з транзакціями, що дозволяє забезпечити атомарність, консистентність, ізольованість та довіреність (ACID) операцій з базою даних. Розробники можуть розпочати транзакцію, виконати кілька запитів і закрити транзакцію, забезпечуючи цілісність даних та уникнення проблем з несанкціонованим доступом до даних;

5) розширення та плагіни. SQLite3 дозволяє розширювати його функціональність за допомогою розширень та плагінів. Розробники можуть створювати власні функції, агрегатні функції, власні типи даних та інші розширення, що дозволяють пристосовувати SQLite3 до конкретних потреб проекту.

SQLite3 є потужним інструментом для роботи з базою даних SQLite у мові програмування Python. Вона надає простоту використання, широкий набір функцій та підтримку розширень, що дозволяють розробникам ефективно працювати з даними в мобільних додатках та інших проектах.

Окрім основних особливостей, які були згадані раніше, розглянемо додаткові деталі про бібліотеку sqlite3 у Python:

1) вбудована бібліотека. sqlite3 є частиною стандартної бібліотеки Python, що означає, що вона встановлюється разом з самою мовою. Не потрібно встановлювати додаткові пакети або залежності для початку використання sqlite3. Вона готова до використання після встановлення Python;

2) підтримка стандарту SQL. sqlite3 використовує мову запитів SQL для взаємодії з базою даних SQLite. Вона підтримує стандартні оператори SQL, такі як SELECT, INSERT, UPDATE і DELETE, а також функції агрегування, з'єднання таблиць, підзапити та багато іншого. Це дає гнучкість у роботі з даними та виконанні потрібних операцій;

3) параметризовані запити і запобігання SQL-ін'єкціям. sqlite3 дозволяє використовувати параметризовані запити, що допомагає уникнути проблем з SQL-ін'єкціями. Можна передавати значення параметрів у запиті як окремі аргументи, що забезпечує безпеку та захист від несанкціонованого доступу до даних;

4) результати запитів. sqlite3 надає різні способи отримання результатів запитів до бази даних. Можна отримувати результати у вигляді списків, кортежів або використовувати об'єкт курсора для поетапного отримання даних. Крім того, можна використовувати методи для отримання одного рядка або всіх рядків результатів, виконання агрегатних функцій та інших операцій над даними;

5) транзакції і керування. sqlite3 підтримує роботу з транзакціями, що дозволяє групувати декілька операцій разом і забезпечує цілісність даних у разі помилок або виключень. Можна розпочати транзакцію, виконати декілька запитів і потім зберегти або скасувати їх. Крім того, можна контролювати

рівень ізоляції транзакцій та режим блокування для керування конкурентним доступом до даних;

6) підтримка вбудованих типів даних. sqlite3 підтримує різноманітні вбудовані типи даних, такі як INTEGER, REAL, TEXT, BLOB та NULL. Можна використовувати ці типи для опису структури таблиць і збереження даних в базі даних;

7) підтримка транзакційних операцій. sqlite3 надає можливість виконувати різні операції з базою даних, такі як створення, перейменування та видалення таблиць, індексів та інших об'єктів бази даних. Також можна виконувати запити на отримання інформації про структуру бази даних, такі як перелік таблиць і їх схеми.

Бібліотека sqlite3 є сучасним інструментом для роботи з базою даних SQLite у Python. Вона надає простий та зручний інтерфейс, підтримку стандарту SQL, захист від SQL-ін'єкцій та багато інших функцій, які допомагають ефективно працювати з даними в програмах на Python.

2.6 Фреймворк KivyMD

KivyMD – це фреймворк для розробки користувацьких інтерфейсів (UI) заснований на Kivy і Material Design. Він надає розширення до Kivy, що дозволяє створювати естетично привабливі та функціональні мобільні додатки згідно з дизайном Material Design від Google (рис. 2.5-2.6).

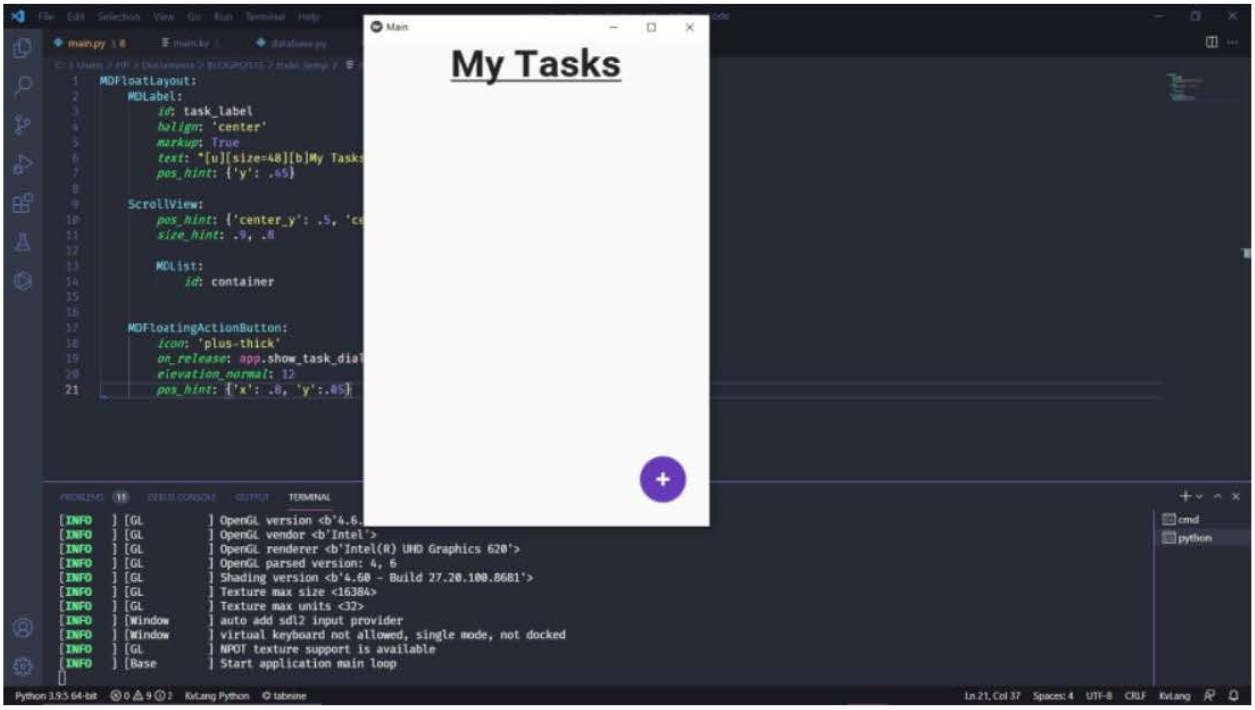

Рисунок 2.5 – Приклад використання KivyMD

Ось детальний огляд основних особливостей KivyMD:

1) Material Design. KivyMD дотримується принципів Material Design, що дає можливість створювати додатки з сучасним та привабливим дизайном. Він надає широкий спектр компонентів та ефектів, таких як кнопки, меню, картки, списки, перехресні списки, тексти, іконки та багато іншого, що дозволяє легко створювати інтуїтивний та привабливий інтерфейс користувача;

2) компоненти та функціональні можливості. KivyMD надає широкий спектр готових компонентів та функціональностей, які полегшують розробку мобільних додатків. Можна використовувати готові елементи, такі як плаваючі кнопки, панелі навігації, випадаючі списки, діалогові вікна, вкладки, перехресні списки та багато інших. Крім того, KivyMD підтримує анімацію, валідацію введення, розмітку, стилізацію та інші функції, що полегшують розробку інтерактивних додатків;

3) підтримка різних платформ. KivyMD є крос-платформеним фреймворком, що означає, що можна розробляти додатки для різних платформ, включаючи Android, iOS, Windows, macOS та Linux. Це дозволяє

створювати універсальні додатки, які можуть працювати на різних пристроях та операційних системах без необхідності переписування коду;

```
def build(self):
    layout = BoxLayout(
        orientation='vertical',
        padding='10sp',
        spacing='10sp'
    )

    label = Label(
        text='Ласкаво просимо до програми замовлень!',
        font_size='24sp'
    )
    layout.add_widget(label)

    name_input = TextInput(
        hint_text='Введіть ваше ім\'я',
        font_size='18sp'
    )

    layout.add_widget(name_input)

    address_input = TextInput(
        hint_text='Введіть вашу адресу',
        font_size='18sp'
    )
    layout.add_widget(address_input)

    product_layout = BoxLayout(
        orientation='vertical',
        size_hint=(1, None),
        spacing='5sp'
    )
    products_label = Label(
        text='Виберіть товари:',
        font_size='18sp'
    )
    product_layout.add_widget(products_label)
```

Рисунок 2.6 – Приклад використання KivyMD

4) простота використання. KivyMD має простий та зрозумілий синтаксис, що полегшує розробку та налагодження додатків. Можна використовувати Python для опису інтерфейсу та логіки додатку, а також

використовувати стандартні елементи KivyMD для створення багатофункціональних додатків;

5) розширюваність. KivyMD заснований на Kivy, що дозволяє розширювати функціональність за допомогою Kivy API. Можна використовувати можливості Kivy для створення спеціалізованих компонентів та функцій, які відповідають унікальним потребам;

6) активна спільнота. KivyMD має активну та дружню спільноту розробників, яка надає підтримку, документацію та приклади коду. Можна звертатися до спільноти за допомогою форумів;

7) крім основних особливостей, які були згадані вище, ось ще деякі деталі про KivyMD;

8) матеріальні ефекти та анімація. KivyMD надає можливість використовувати матеріальні ефекти, такі як тіні, плавні переходи та анімацію, для створення більш привабливих інтерфейсів. Це додає реалістичність та покращує взаємодію користувача з додатком;

9) підтримка іконок та шрифтів. KivyMD надає вбудовану підтримку для широкого спектру іконок та шрифтів Material Design, що дозволяє легко використовувати їх у своїх додатках. Можна використовувати іконки для кнопок, меню, табуляції та інших компонентів, а також встановлювати різні шрифти для текстових елементів;

10) підтримка мультиязичності. KivyMD дозволяє локалізувати додатки, що означає, що можна створювати додатки з підтримкою різних мов. Він надає можливість перекладу тексту в додатку, а також підтримує різні мовні ресурси, що дозволяє зручно керувати локалізацією;

11) інтеграція з Python. KivyMD побудований на основі Kivy, що означає, що можна використовувати всі можливості Python для створення потужних додатків. Можна використовувати Python-бібліотеки, взаємодіяти з базами даних, виконувати обчислення та багато іншого;

12) розширені можливості. KivyMD надає можливість розширювати функціональність за допомогою сторонніх плагінів та розширень. Є багато

розширень, які додають нові компоненти, функції та можливості до фреймворку KivyMD;

13) документація та приклади. KivyMD має детальну документацію, яка пояснює використання різних компонентів та функціональностей. Крім того, можна зайти багато прикладів коду, які допоможуть швидко розібратися з розробкою додатків на KivyMD.

Отже загалом, KivyMD є потужним інструментом для розробки мобільних додатків з використанням Material Design. Він надає розширення до фреймворку Kivy, що дозволяє створювати привабливі та функціональні додатки зі широким спектром компонентів та функціональних можливостей.

2.7 Висновок до другого розділу

У розділі 2 кваліфікаційної роботи проведено детальний аналіз засобів розробки мобільного додатка для здійснення замовлень з використанням мови програмування Python. Вибір програмних засобів є критичним етапом розробки, і у цьому розділі було розглянуто основні засоби, які можна використовувати для розробки мобільних додатків з використанням Python.

Огляд мови програмування Python показав, що це потужна та популярна мова, яка має багато переваг для мобільної розробки. Зрозумілий синтаксис, широкий вибір бібліотек та фреймворків, підтримка різних операційних систем та простота використання роблять Python привабливим вибором для розробки мобільних додатків.

Одним з розглянутих фреймворків був Kivy, який забезпечує можливість розробки кросплатформових мобільних додатків з використанням Python. Його основні особливості включають підтримку багатофункціональних інтерфейсів, анімацію, взаємодію зі вбудованими сенсорами та багато іншого.

Також була розглянута база даних SQLite, яка є легкою та швидкою вбудованою базою даних, підтримуваною Python. Модуль sqlite3 надає

зручний інтерфейс для взаємодії з базою даних SQLite, що дозволяє зберігати та отримувати дані в додатку.

Одним з ключових елементів розглянутої технологічної стеку був фреймворк KivyMD, який надає розширення до Kivy та дозволяє розробляти додатки з використанням Material Design. KivyMD забезпечує багатий вибір компонентів, стилізацію, анімацію та інші можливості для створення привабливих та функціональних мобільних додатків.

Загальною висновок з розділу 2 є те, що засоби розробки, які були розглянуті, є потужними та ефективними для розробки мобільного додатка для здійснення замовлень з використанням Python. Обраний технологічний стек, що складається з Python, Kivy, SQLite та KivyMD, надає достатній набір інструментів для реалізації поставлених функціональних та нефункціональних вимог.

РОЗДІЛ 3 РОЗРОБЛЕННЯ МОБІЛЬНОГО ДОДАТКА ДЛЯ ЗДІЙСНЕННЯ ЗАМОВЛЕНЬ ЗАСОБАМИ МОВИ ПРОГРАМУВАННЯ PYTHON

3.1 Архітектура проекту

Детальна архітектура проекту може бути такою:

Користувацький інтерфейс (UI) шар:

– main.py. Головний файл додатка, який містить клас OrderApp і слугує точкою входу для додатка. Він створює і запускає екземпляр графічного інтерфейсу користувача (GUI);

– ui/screens/. Директорія, що містить модулі, які стосуються різних екранів програми;

– home_screen.py. Модуль, що містить клас HomeScreen, який відображає головний екран програми, де користувач може вводити дані та розміщувати замовлення;

– order_history_screen.py. Модуль, що містить клас OrderHistoryScreen, який відображає екран історії замовлень, де відображається список попередніх замовлень;

– popup_screen.py. Модуль, що містить клас PopupScreen, який відображає спливаючі вікна з повідомленнями та підтвердженнями;

– ui/widgets/. Директорія, що містить модулі, які відносяться до віджетів, що налаштовуються;

– checkbox_label.py. Модуль, що містить клас CheckboxLabel, який визначає настроюваний віджет прапорця з текстом;

– spinner_label.py. Модуль, що містить клас SpinnerLabel, який визначає настроюваний віджет списку, що випадає;

– text_input_label.py. Модуль, що містить клас TextInputLabel, який визначає настроюваний віджет текстового поля введення.

Бізнес-логіка (Logic) шар:

– main.py. Файл main.py також слугує контролером, що пов'язує користувацький інтерфейс із бізнес-логікою;

– database/. Директорія, що містить базу даних SQLite orders.db, де зберігаються дані про замовлення;

– database_handler.py. Модуль, що містить клас DatabaseHandler, який надає методи для взаємодії з базою даних, такі як створення таблиці замовлень, додавання нових замовлень і отримання історії замовлень.

Модель (Model) шар: order.py. Модуль, що містить клас Order, який представляє модель замовлення. Він визначає властивості замовлення, як-от ім'я, адреса, обрані товари, спосіб доставки та загальна вартість.

Така архітектура додатка відокремлює користувацький інтерфейс (UI) від бізнес-логіки та моделі даних. Це дає змогу легко додавати нові екрани і функціональності, а також робить код більш організованим і підтримуваним. Клас OrderApp у main.py є сполучним елементом, який ініціалізує графічний інтерфейс і обробляє події користувальницької взаємодії, передаючи необхідні дані до бізнес-логіки та моделі для обробки й відображення.

3.2 Проектування бази даних

Детальний процес проектування бази даних для мобільного додатка для здійснення замовлень мовою програмування Python може бути таким:

1) визначення сутностей. Спочатку потрібно визначити основні сутності, з якими взаємодіятиме додаток. У цьому випадку, основною сутністю буде «Замовлення». Додаткові сутності можуть включати «Товар», «Клієнт», «Адреса доставки» тощо, залежно від вимог додатка;

2) визначення атрибутів сутностей. Для кожної сутності можна визначити її атрибути, тобто інформацію, яку можна зберігати в базі даних. Наприклад, для сутності «Замовлення» атрибутами можуть бути. ідентифікатор замовлення, дата і час замовлення, список обраних товарів, статус замовлення тощо;

3) визначення зв'язків між сутностями. Якщо є пов'язані сутності, необхідно визначити їхні зв'язки. Наприклад, замовлення може бути пов'язане з клієнтом, адресою доставки і товарами. Визначити типи зв'язків (один-до-одного, один-до-багатьох, багато-до-багатьох) та їхні кардинальності (1.1, 1.N, N.M);

4) створення схеми бази даних. На основі визначених сутностей, їхніх атрибутів і зв'язків створити схему бази даних. Це включає в себе визначення таблиць для кожної сутності та їхніх стовпців, а також визначення обмежень і зв'язків між таблицями (за необхідності);

5) нормалізація бази даних. Необхідно провести процес нормалізації, щоб усунути надмірність даних і забезпечити структурну цілісність. Необхідно застосувати правила нормалізації, такі як перша нормальна форма (1NF), друга нормальна форма (2NF) і третя нормальна форма (3NF), щоб розділити дані на логічно пов'язані таблиці й усунути повторення даних;

6) визначення первинних і зовнішніх ключів. Визначити первинні ключі для кожної таблиці, щоб унікально ідентифікувати записи. Також необхідно визначити зовнішні ключі для зв'язків між таблицями, щоб забезпечити цілісність посилальної цілісності;

7) визначення типів даних і обмежень. Для кожного стовпця можна визначити відповідні типи даних, такі як цілі числа, рядки, дати тощо. Також необхідно визначити обмеження, наприклад, обмеження на унікальність, обмеження на значення та інші правила для забезпечення цілісності даних;

8) оптимізація продуктивності. За необхідності можна провести оптимізацію продуктивності бази даних, наприклад, створити індекси на стовпці, що часто використовуються, або розбити таблиці на різні файли для ефективнішого зберігання та доступу до даних;

9) реалізація бази даних. На основі визначеної схеми бази даних необхідно створити таблиці, визначити зв'язки та додати необхідні індекси й обмеження з використанням мови SQL;

10) тестування та налагодження. Після реалізації бази даних необхідно провести тестування і налагодження для перевірки її працездатності та відповідності вимогам додатка.

Це загальний процес проектування бази даних для мобільного додатка на основі мови програмування Python. Він може бути доповнений або адаптований відповідно до вимог та особливостей проекту.

Розглянемо детально поля в таблицях бази даних, їх зв'язки та інші важливі аспекти для мобільного додатка для здійснення замовлень на основі мови програмування Python.

Таблиця «Користувачі»:

– id (первинний ключ). Унікальний ідентифікатор користувача;

– ім’я. Ім’я користувача;

– прізвище. Прізвище користувача;

– email. Електронна пошта користувача;

– пароль. Хешований пароль користувача для аутентифікації.

Таблиця «Товари»:

– id (первинний ключ). Унікальний ідентифікатор товару;

– назва. Назва товару;

– опис. Опис товару;

– ціна. Ціна товару;

– кількість. Кількість товару на складі.

Таблиця «Замовлення»:

– id (первинний ключ). Унікальний ідентифікатор замовлення;

– id_користувача (зовнішній ключ). Ідентифікатор користувача, який зробив замовлення;

– дата_замовлення. Дата і час замовлення;

– статус. Статус замовлення (наприклад, «в очікуванні», «виконується», «відправлено»);

– загальна_сума. Загальна сума замовлення;

– адреса_доставки. Адреса доставки замовлення;

– спосіб_оплати. Спосіб оплати (наприклад, «готівкою», «карткою»).

Таблиця «Деталі замовлення»:

– id (первинний ключ). Унікальний ідентифікатор деталі замовлення;

– id_замовлення (зовнішній ключ). Ідентифікатор замовлення, до якого належить деталь;

– id_товару (зовнішній ключ). Ідентифікатор товару, що включений у замовлення;

– кількість. Кількість товару, замовленого користувачем.

Зв'язки між таблицями:

– Таблиця «Замовлення» має зв'язок багато-до-одного з таблицею «Користувачі» за допомогою поля id_користувача;

– таблиця «Деталі замовлення» має зв'язок багато-до-одного з таблицею «Замовлення» за допомогою поля id_замовлення;

– таблиця «Деталі замовлення» має зв'язок багато-до-одного з таблицею «Товари» за допомогою поля id_товару.

Інші важливі аспекти:

– додаткові поля можуть бути додані для збереження інших відомостей, наприклад, адреси, контактних даних, додаткових опцій товарів тощо;

– можна встановити обмеження на поля, наприклад, обов'язковість заповнення певних полів або обмеження допустимих значень;

– використання індексів може поліпшити швидкодію операцій з базою даних, особливо для часто використовуваних полів або при пошуку і фільтрації даних.

Це загальна структура бази даних для мобільного додатка для здійснення замовлень на основі мови програмування Python. Можна додати або змінити поля відповідно до вимог і особливостей проекту.

3.3 Проектування інтерфейсу та опис роботи

Детальний опис роботи мобільного додатка для здійснення замовлень на основі мови програмування Python може бути таким:

1) вхід у додаток. Під час запуску додатка користувачеві пропонується увійти в систему або зареєструватися. Якщо користувач уже має обліковий запис, він може ввести свої облікові дані (наприклад, ім'я користувача та пароль) для входу. Якщо в користувача немає облікового запису, він може вибрати опцію реєстрації та ввести необхідну інформацію для створення нового облікового запису;

2) головний екран. Після успішного входу в застосунок користувач потрапляє на головний екран, де він може переглянути доступні товари або послуги, вибрати потрібні позиції та розмістити замовлення. На цьому екрані можуть бути різні елементи інтерфейсу, такі як списки товарів, кнопки додавання в кошик, поле пошуку тощо;

3) добавлення товарів до кошика. Користувач може переглядати список доступних товарів або послуг і вибирати ті, які він хоче замовити. При виборі товару користувач може вказати кількість або інші додаткові параметри. Обрані товари додаються до кошика;

4) управління кошиком. На головному екрані є можливість переглянути вміст кошика і керувати товарами в ньому. Користувач може змінювати кількість товарів, видаляти товари з кошика або очищати кошик повністю;

5) оформлення замовлення. Після того, як користувач додав усі необхідні товари в кошик, він може перейти до оформлення замовлення. На цьому етапі користувач має надати додаткову інформацію, як-от адресу доставки, контактні дані та вибрати бажаний спосіб оплати;

6) підтвердження замовлення. Після заповнення всіх необхідних даних користувач може підтвердити замовлення. Додаток перевіряє правильність заповнення даних і відправляє замовлення на обробку;

7) історія замовлень. Користувач може переглядати історію своїх попередніх замовлень. Це дає йому змогу відстежувати статус замовлень, переглядати подробиці та повторювати попередні замовлення за потреби;

8) повідомлення. Додаток може надсилати сповіщення користувачеві про статус його замовлень, нові акції або спеціальні пропозиції;

9) автентифікація та безпека. Додаток має забезпечувати безпеку користувацьких даних і аутентифікацію. Це може включати зберігання паролів у зашифрованому вигляді, використання токенів для автентифікації та авторизації користувачів;

10) інтеграція із сервером або базою даних. Додаток повинен мати можливість спілкуватися із сервером або базою даних для отримання списку товарів, збереження замовлень і оновлення інформації про користувача.

Це загальна структура та опис роботи мобільного додатка для здійснення замовлень на основі мови програмування Python. Однак, конкретні деталі та функціональність можуть варіюватися залежно від вимог та особливостей проєкту.

Вигляд графічного інтерфейсу зображено на рисунку 3.1.

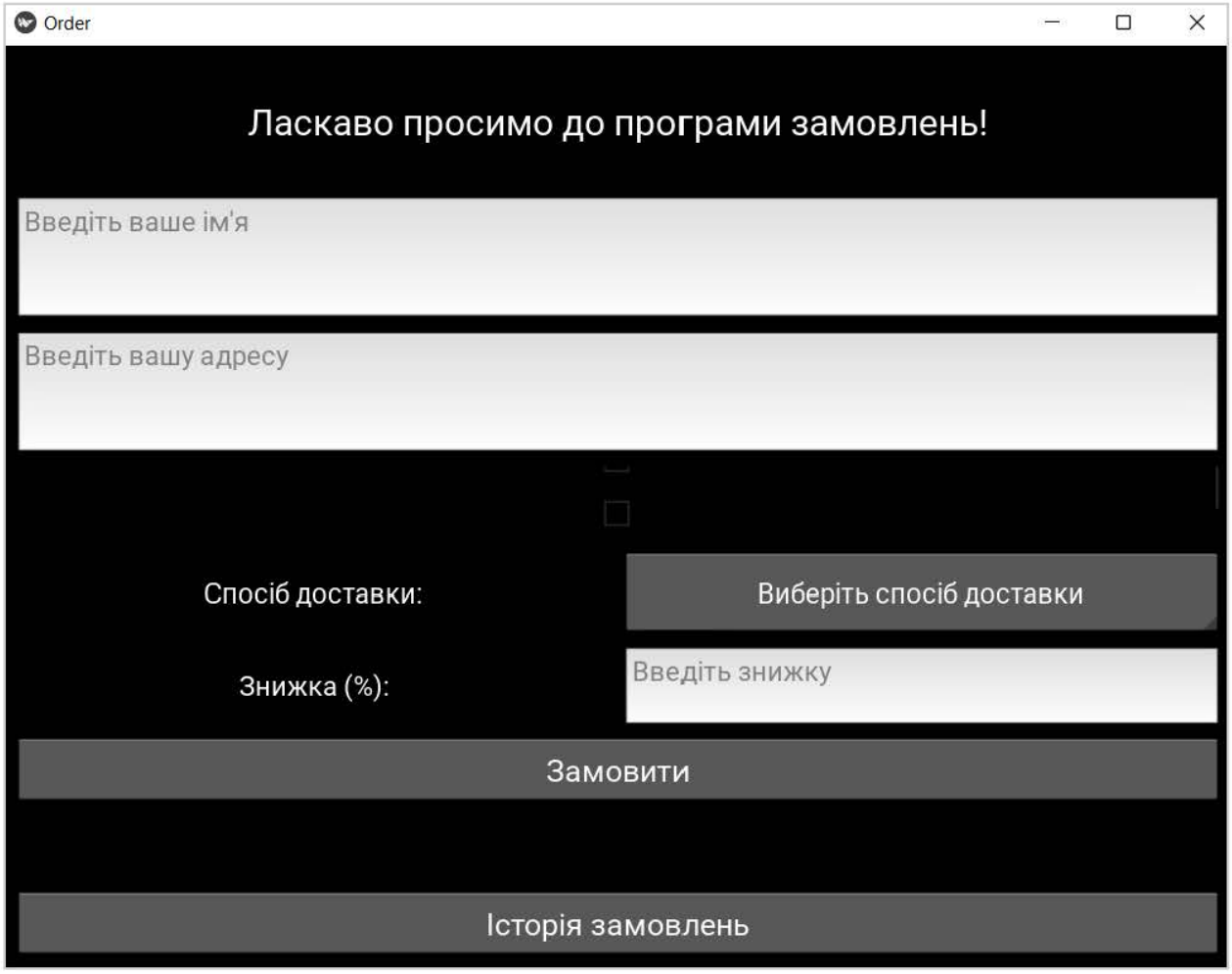

Рисунок 3.1 – Інтерфейс мобільного додатка

3.4 Тестування мобільного додатка для здійснення замовлень засобами мови програмування Python

Рисунок 3.2 відображає головне вікно мобільного додатка, яке може містити список доступних товарів або послуг, кнопки для навігації та інші елементи інтерфейсу.

Рисунок 3.3 демонструє ситуацію, коли користувач намагається зробити замовлення, але не заповнив необхідну інформацію, наприклад, адресу доставки або спосіб оплати. Може бути показана попереджувальна помилка або сповіщення, яке нагадує користувачу про необхідність заповнити всі поля.

На рисунку 3.4 можна побачити інтерфейс, де користувач може обрати потрібні товари зі списку доступних. Це може бути список зображень або назв товарів, які користувач може вибрати або додати до свого замовлення.

Рисунок 3.5 може відображати інтерфейс, де користувач вводить свої дані, такі як адреса доставки, контактні дані або додаткові опції. Наприклад, це може бути форма з полями для введення тексту, списками вибору або іншими елементами для заповнення даних.

Рисунок 3.6 може показувати інтерфейс, де користувачу надається вказівка або можливість ввести промо-код або знижку для застосування до замовлення. Може бути показано поле для введення коду або інший спосіб активувати знижку.

На рисунку 3.7 може бути показано інтерфейс, де користувач може вибрати спосіб доставки, наприклад, кур'єрську службу або самовивіз. Це може бути список варіантів, кнопки вибору або інші елементи для вибору способу доставки.

На рисунку 3.8 можна побачити інтерфейс, де користувач може переглянути готове замовлення перед підтвердженням. Це може бути список товарів, вибраних параметрів, загальної суми та іншої відомостей, що стосуються замовлення.

На рисунку 3.9 може бути показано процес здійснення замовлення, коли користувач підтверджує свої вибори та надсилає замовлення. Це може бути кнопка «Замовити» або інший спосіб підтвердження дії.

Рисунок 3.10 показує ситуацію, коли користувач намагається здійснити замовлення, але не обрав жодного товару. Може бути показана попереджувальна помилка або сповіщення, яке нагадує користувачу про необхідність вибрати товари перед здійсненням замовлення.

На рисунку 3.11 можна побачити інтерфейс, де користувач може переглянути свою історію замовлень, включаючи деталі кожного замовлення, дати, статуси тощо. Це може бути список замовлень або спеціальна сторінка для перегляду історії.

Ці рисунки допомагають зрозуміти вигляд та функціональність мобільного додатка для здійснення замовлень.

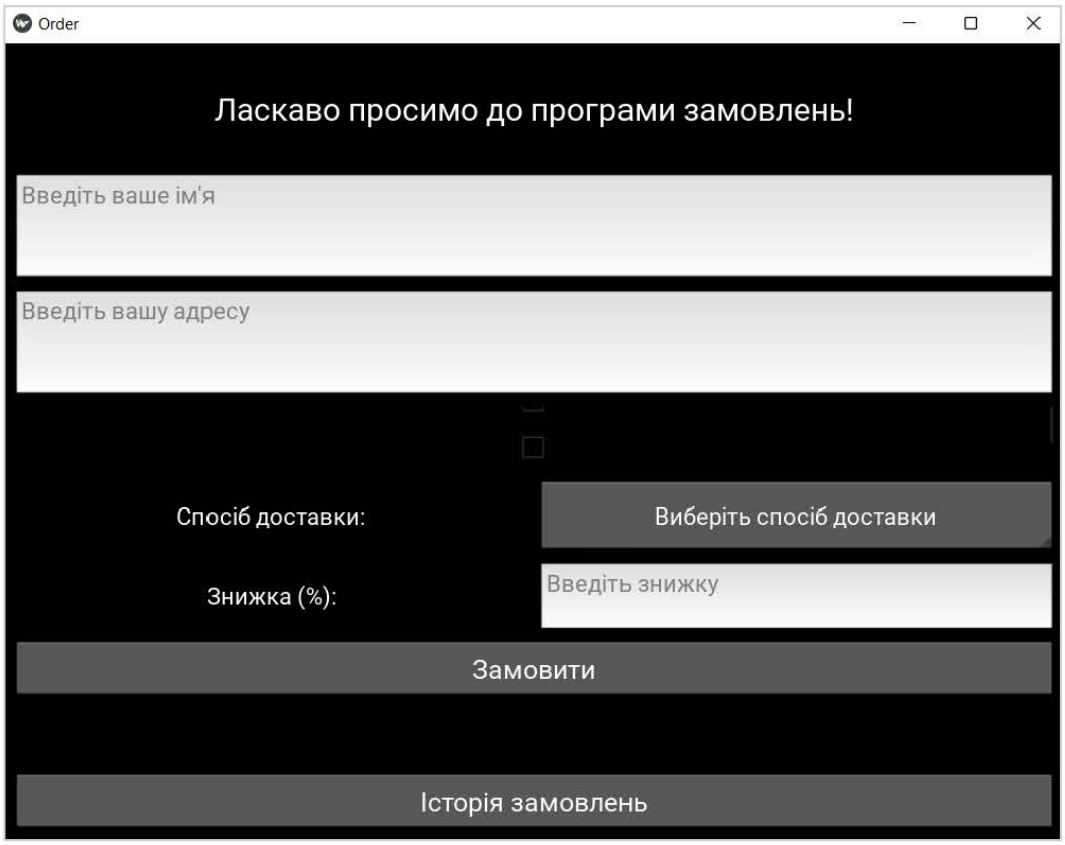

Рисунок 3.2 – Головне вікно мобільного додатка

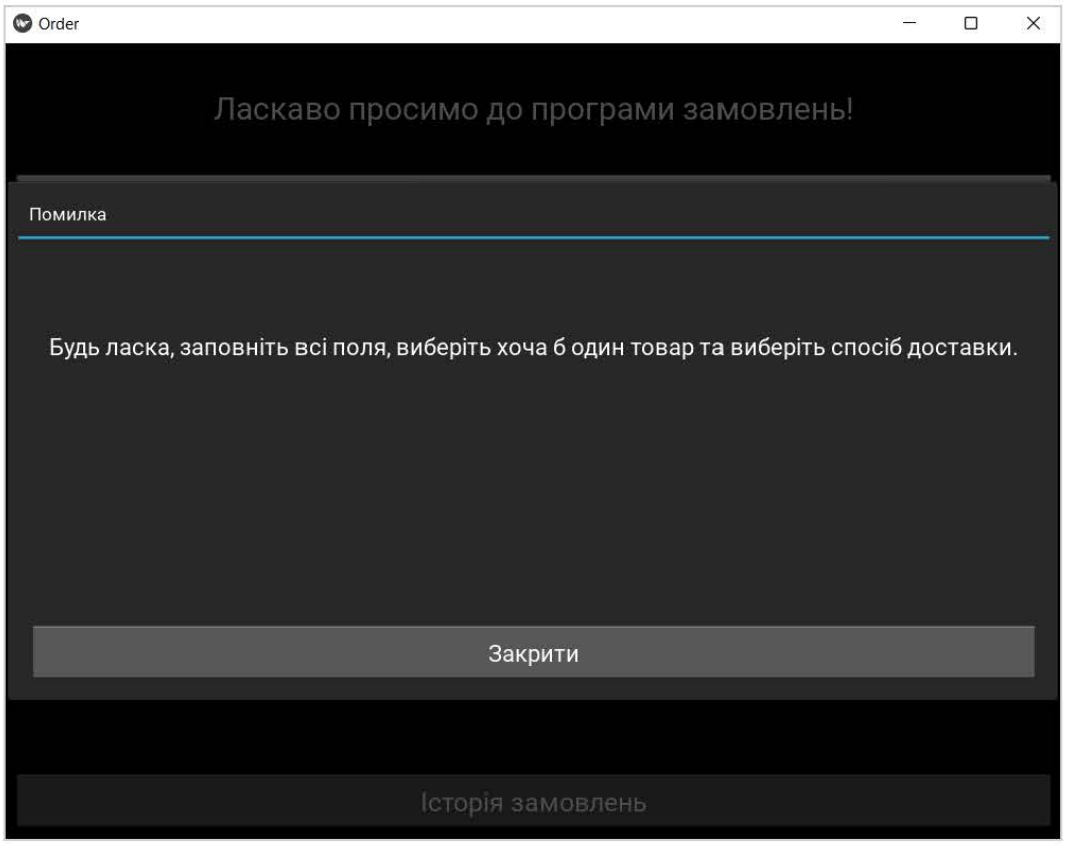

Рисунок 3.3 – Спроба зробити замовлення без заповнених даних

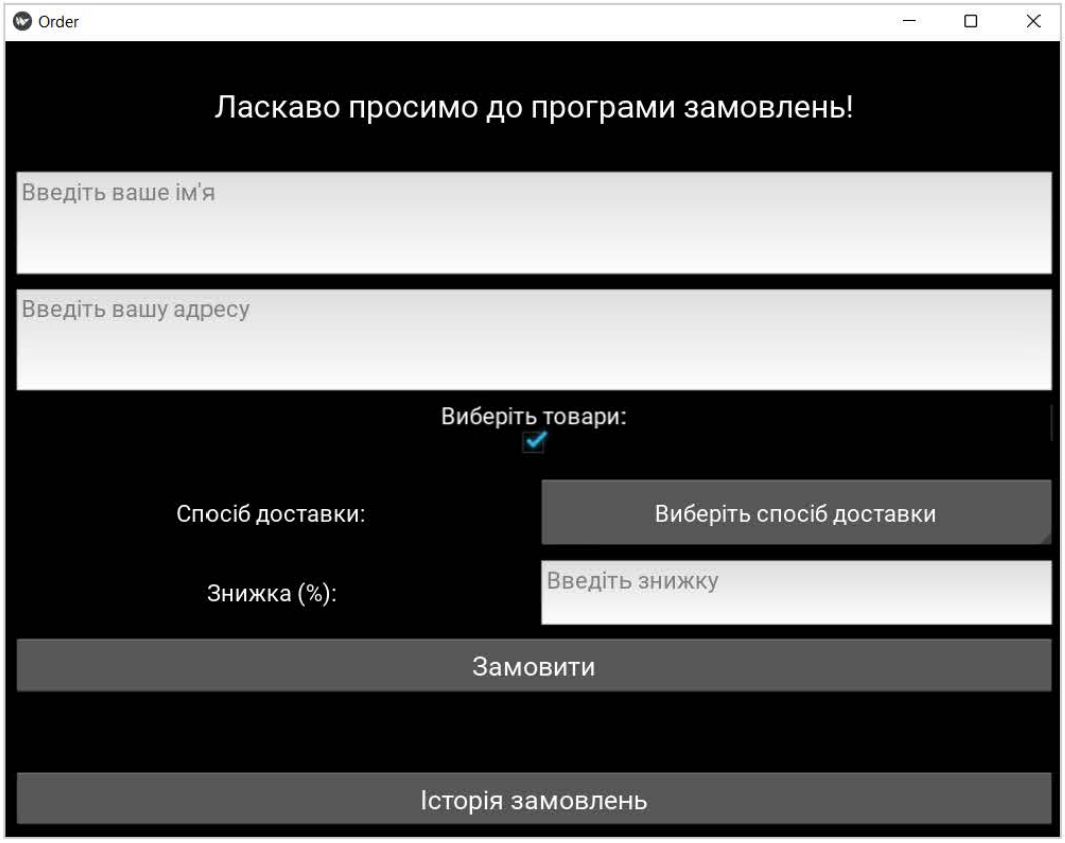

Рисунок 3.4 – Приклад обирання товарів

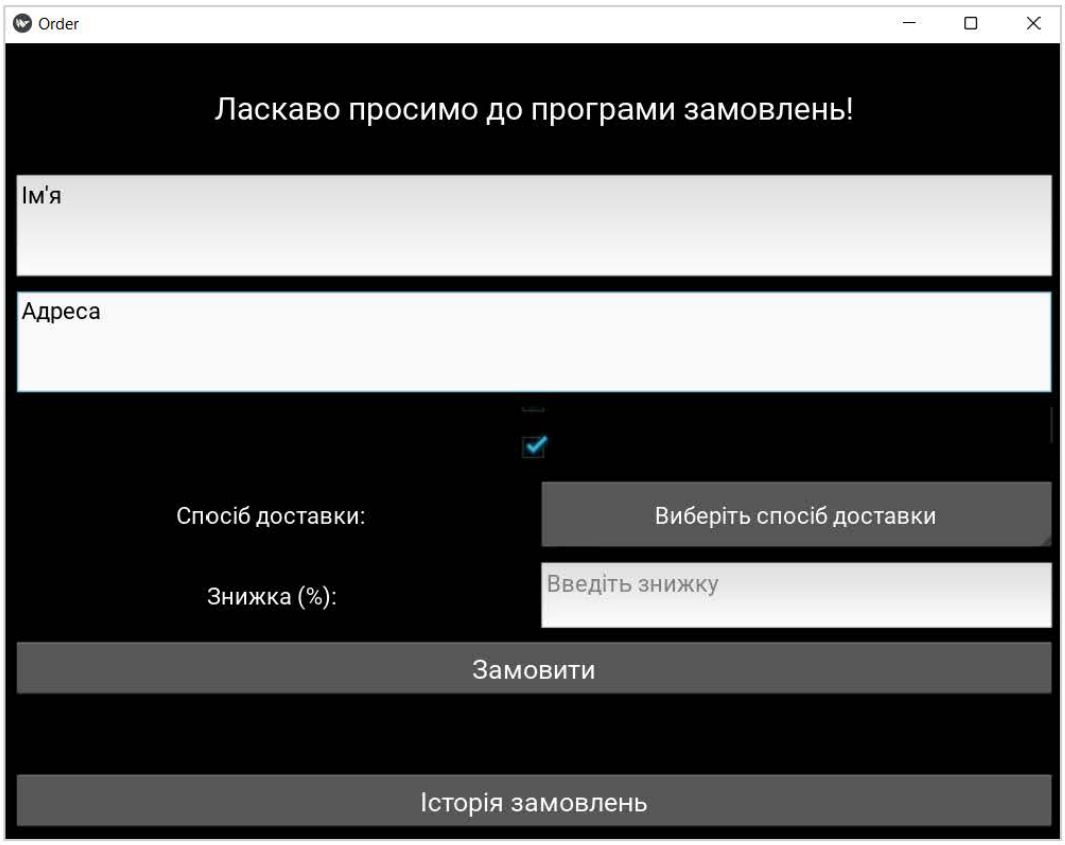

Рисунок 3.5 – Приклад заповнення даних

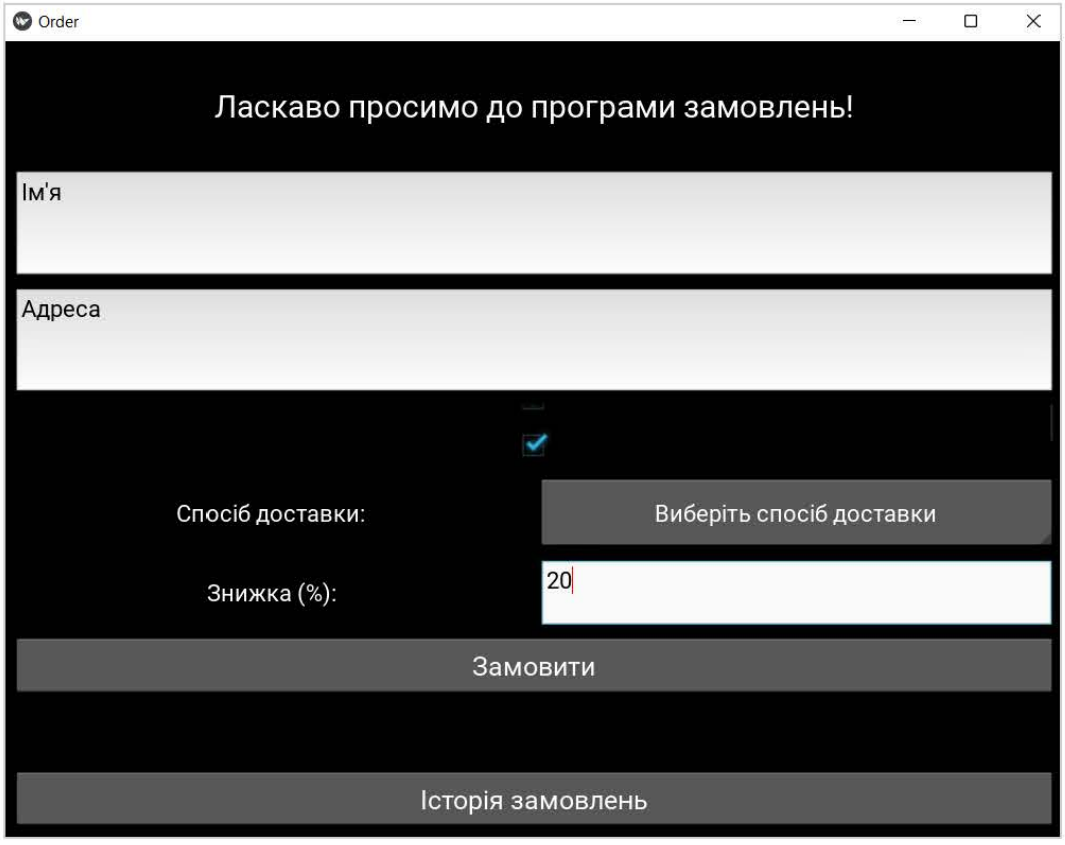

Рисунок 3.6 – Вказівка знижки

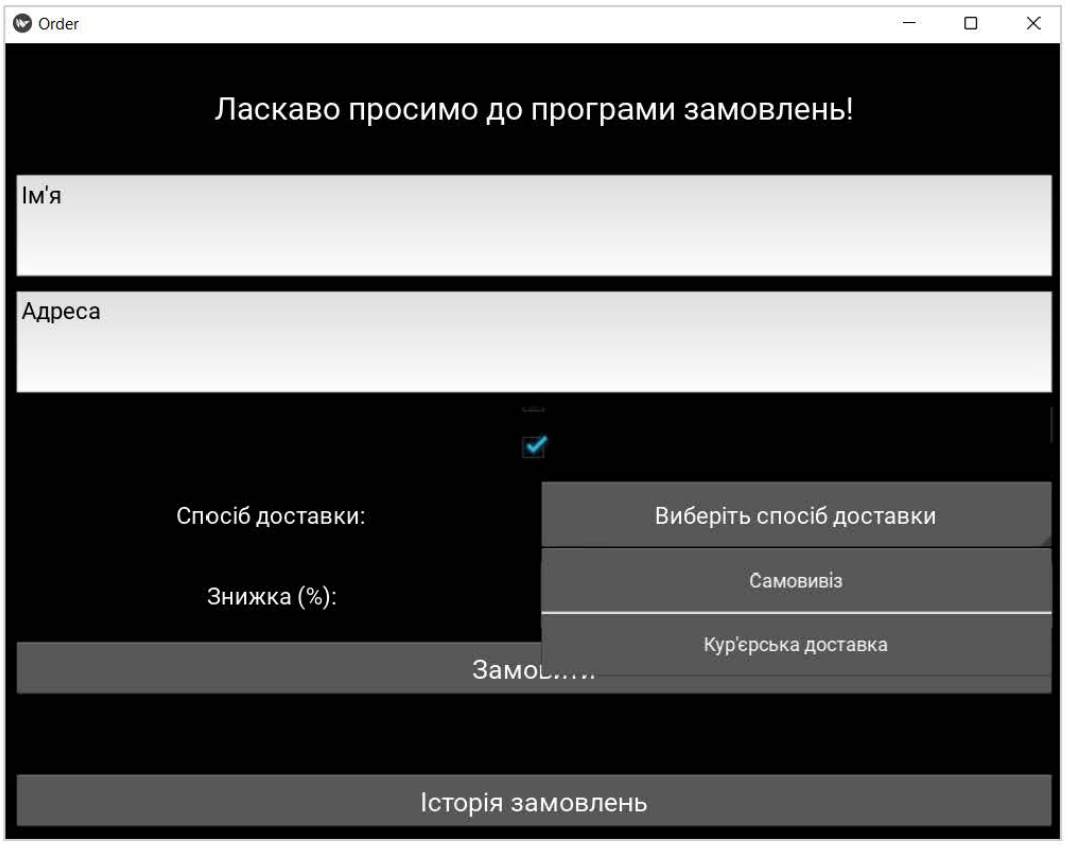

Рисунок 3.7 – Обрання способу доставки

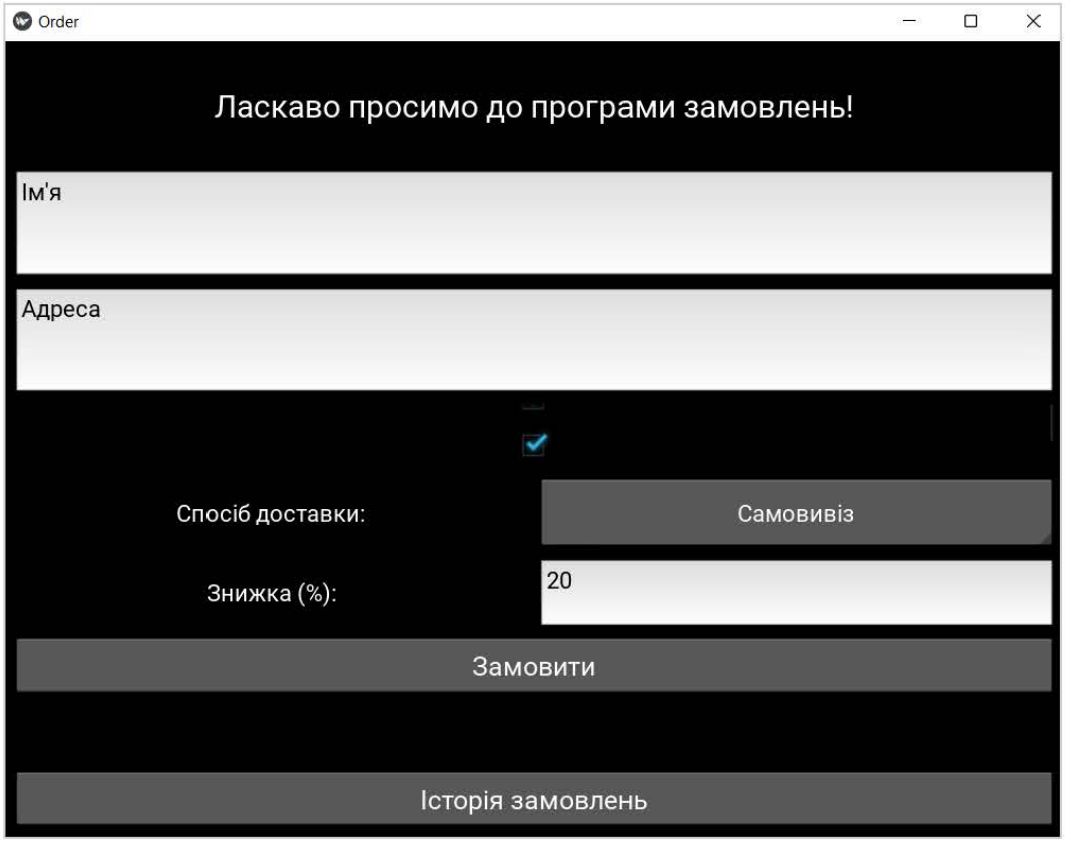

Рисунок 3.8 – Приклад готового замовлення

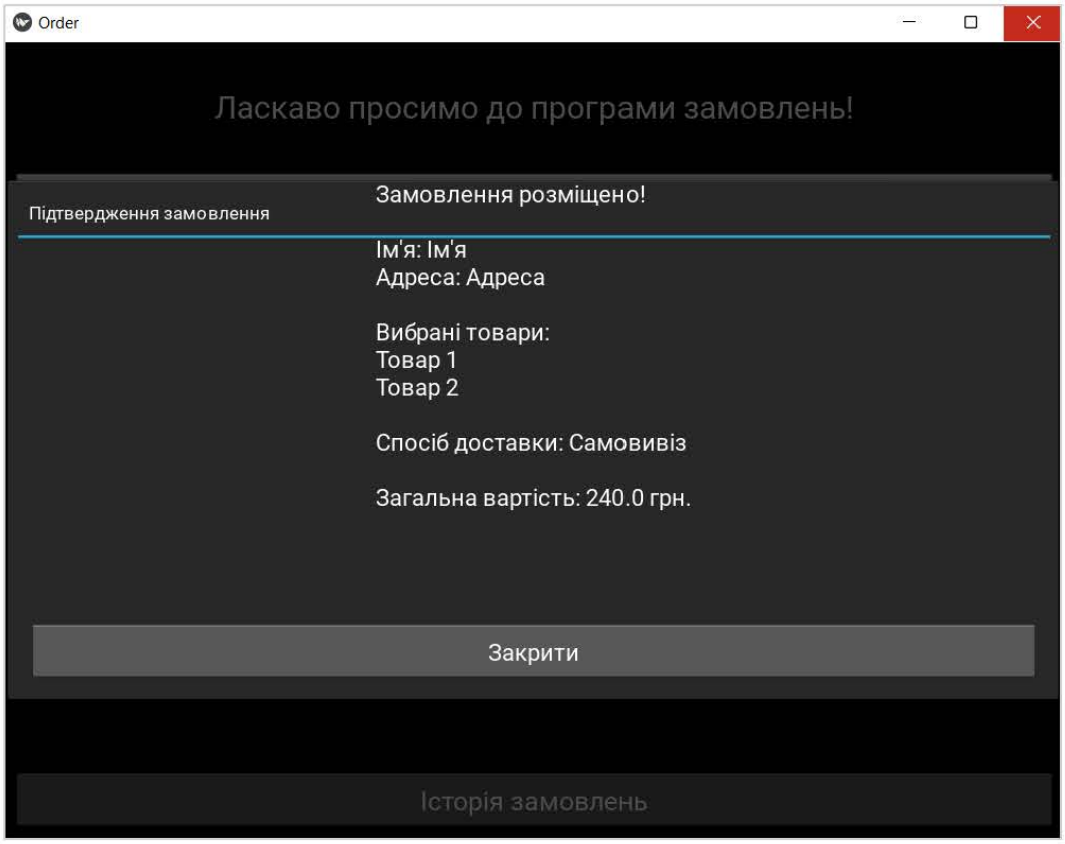

Рисунок 3.9 – Здійснення замовлення

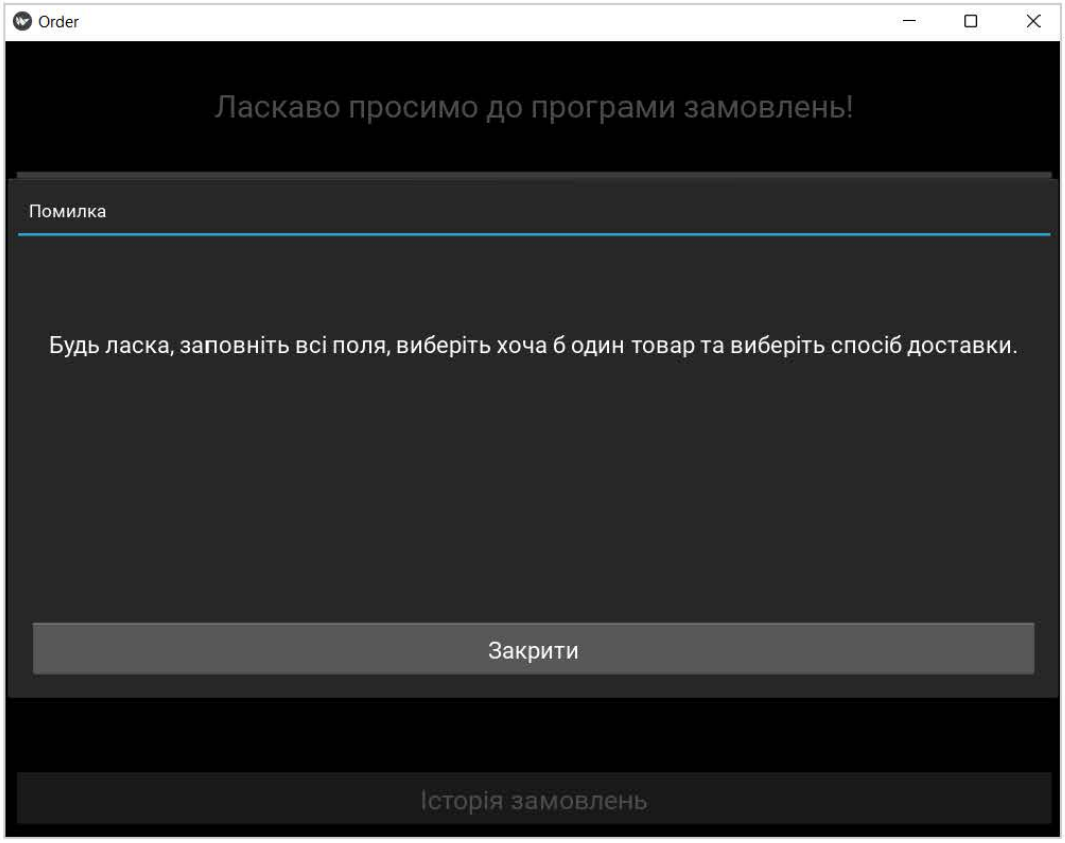

Рисунок 3.10 – Спроба здійснити замовлення без вказаних товарів

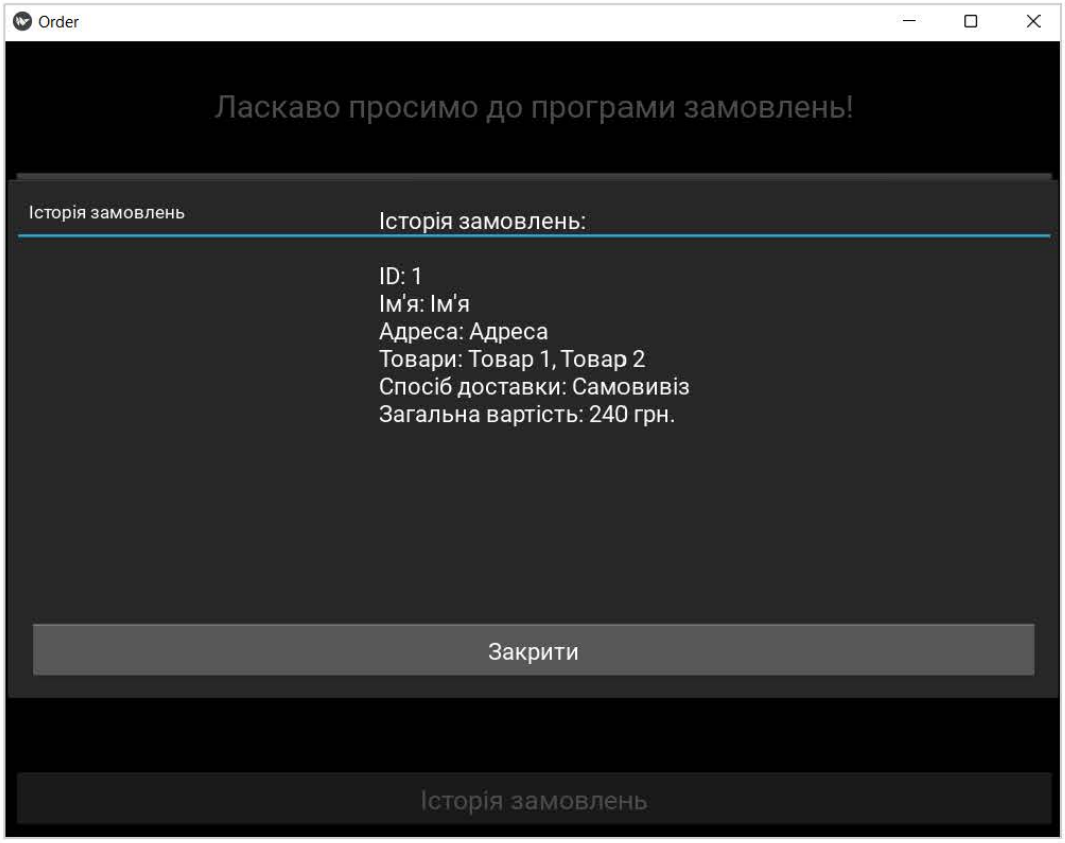

Рисунок 3.11 – Перегляд історії замовлень

3.5 Висновки до третього розділу

У третьому розділі «Розроблення мобільного додатка для здійснення замовлень засобами мови програмування Python» були представлені наступні підрозділи, які наведені по тексту нижче.

У підрозділі 3.1 було описано загальну архітектуру проекту мобільного додатка для здійснення замовлень.

В підрозділі 3.2 було описано процес проектування бази даних для мобільного додатка. На основі вимог до функціональності були запропоновані таблиці та їхні поля. Вказано, що база даних буде зберігати інформацію про користувачів, замовлення, товари тощо.

У підрозділі 3.3 було описано процес проектування інтерфейсу мобільного додатка. Наведено опис роботи додатка, такий як вибір товарів,

введення даних користувача, вказівка знижки, обрання способу доставки та інші етапи здійснення замовлення.

У підрозділі 3.4 було згадано процес тестування мобільного додатка для перевірки його функціональності, коректності роботи та виявлення можливих помилок.

В цьому підрозділі містяться висновки, які робляться на підставі представленої інформації у третьому розділі.

ВИСНОВКИ

Метою роботи було розроблення мобільного додатка для здійснення замовлень засобами мови програмування Python.

Для досягнення поставленої мети розробки мобільного додатка для здійснення замовлень засобами мови програмування Python були заздалегідь визначені наступні завдання:

1) аналіз вимог. Спочатку необхідно було провести детальний аналіз вимог до мобільного додатка. Це включало вивчення функціональних та нефункціональних вимог, розуміння потреб користувачів, аналіз конкурентного середовища та визначення основних функцій та можливостей, які має мати додаток;

2) проектування інтерфейсу користувача. Важливим кроком є проектування інтерфейсу, який буде зручним та інтуїтивно зрозумілим для користувачів. Це включає розробку екранів, створення навігаційної структури, вибір кольорової схеми та розміщення елементів у додатку. Проектування інтерфейсу має забезпечити зручність взаємодії користувача з додатком та відображення інформації;

3) розробка функціональності. Після проектування інтерфейсу необхідно розробити функціональність додатка. Це включає реалізацію можливостей замовлення товарів або послуг, обробку платежів, управління користувачами, збереження замовлень у базі даних тощо. Для цього можуть використовуватися різні бібліотеки та фреймворки Python, такі як Kivy, PyQt або Django;

4) забезпечення безпеки. Безпека є важливим аспектом будь-якого мобільного додатка, особливо тоді, коли мова йде про замовлення товарів або послуг. Для забезпечення безпеки необхідно розглянути питання аутентифікації користувачів, захисту особистих даних, безпечної передачі інформації та захисту від можливих кібератак;

5) тестування та вдосконалення. Після розробки мобільного додатка важливо провести тестування, яке допоможе виявити та виправити можливі помилки або недоліки. Також може бути потрібно збирати фідбек від користувачів, щоб покращити функціональність та досконаліть додатка;

6) оптимізація та покращення продуктивності. Для забезпечення швидкої та ефективної роботи додатка можуть бути необхідні оптимізаційні заходи, такі як покращення швидкості завантаження, оптимізація запитів до бази даних або кешування даних. Це допоможе забезпечити задоволення користувачів та зменшити можливі проблеми з продуктивністю.

Виконання цих задач допомогло досягти поставленої мети розробки мобільного додатка для здійснення замовлень засобами мови програмування Python та забезпечити користувачам зручний та безпечний спосіб здійснення замовлень через мобільні пристрої.

Об’єктом дослідження були процеси роботи мобільного додатка для здійснення замовлень засобами мови програмування Python.

Предметом дослідження було апаратно-програмне забезпечення для розроблення мобільного додатка для здійснення замовлень засобами мови програмування Python.

Практичне значення одержаних результатів полягає у підвищенні якості здійснення замовлень засобами мови програмування Python.

Результатами роботи мобільний додаток для здійснення замовлень засобами мови програмування Python.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Briggs J. R. Python for Kids: A Playful Introduction to Programming, 2013. 344 p.

2. Matthes E. Python Crash Course: A Hands-On, Project-Based Introduction to Programming, 2019. 544 p.

3. Sweigart A. Automate the Boring Stuff with Python: Practical Programming for Total Beginners, 2015. 504 p.

4. Zelle J. Python Programming: An Introduction to Computer Science, 2016. 512 p.

5. Ramalho L. Fluent Python: Clear, Concise, and Effective Programming, 2015. 792 p.

6. Beazley D., Jones B. Python Cookbook, 2013. 706 p.

7. Lutz M. Learning Python, 2013. 1648 p. (4th edition)

8. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2017. 544 p.

9. Slatkin B. Effective Python: 59 Specific Ways to Write Better Python, 2015. 384 p.

10. Forcier J., Bissex P., Chun W. Python Web Development with Django, 2016. 480 p.

11. Lutz M. Python Pocket Reference, 2020. 266 p.

12. McKinney W. Python for Data Science For Dummies, 2019. 528 p.

13. VanderPlas J. Python Data Science Handbook: Essential Tools for Working with Data, 2016. 548 p.

14. Ramalho L. Python Fluente, 2019. 792 p. (Brazilian Portuguese edition)

15. Sweigart A. Invent Your Own Computer Games with Python, 2015. 376 p.

16. Martelli A., Ravenscroft A., Holden S. Python in a Nutshell, 2017. 772 p.

17. Barry P. Python Programming for the Absolute Beginner, 2010. 464 p.

18. Summerfield M. Programming in Python 3: A Complete Introduction to the Python Language, 2009. 648 p.

19. Pilgrim M. Dive Into Python 3, 2009. 360 p.

20. Grayson M. Python Cookbook, 4th Edition, 2020. 706 p.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
import kivy
kivy.require('1.11.1')


import sqlite3
from kivy.app import App
from kivy.uix.label import Label
from kivy.uix.button import Button
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.textinput import TextInput
from kivy.uix.popup import Popup
from kivy.uix.checkbox import CheckBox
from kivy.uix.spinner import Spinner
from kivy.uix.scrollview import ScrollView


class OrderApp(App):
    def __init__(self):
        super().__init__()
        self.connection = None
        self.cursor = None

    def build(self):
        layout = BoxLayout(
            orientation='vertical',
            padding='10sp',
            spacing='10sp'
```

```
)

label = Label(
    text='Ласкаво просимо до програми замовлень!',
    font_size='24sp'
)
layout.add_widget(label)

name_input = TextInput(
    hint_text='Введіть ваше ім\'я',
    font_size='18sp'
)

layout.add_widget(name_input)

address_input = TextInput(
    hint_text='Введіть вашу адресу',
    font_size='18sp'
)
layout.add_widget(address_input)

product_layout = BoxLayout(
    orientation='vertical',
    size_hint=(1, None),
    spacing='5sp'
)
products_label = Label(
    text='Виберіть товари:',
    font_size='18sp'
)
```

```
product_layout.add_widget(products_label)

self.products = {
    'Товар 1': 100,
    'Товар 2': 200,
    'Товар 3': 300
}  # Словник з товарами та їх цінами

self.checkboxes = []  # Список для збереження прапорців

for product, price in self.products.items():
    checkbox = CheckBox(
        active=False,
        size_hint=(1, None),
        height='30sp'
    )

    checkbox.label = Label(
        text=f"{product} - {price} грн.",
        size_hint=(1, None),
        height='30sp'
    )

    checkbox.add_widget(checkbox.label)
    product_layout.add_widget(checkbox)
    self.checkboxes.append(checkbox)

scrollview = ScrollView(size_hint=(1, 0.6))
scrollview.add_widget(product_layout)
layout.add_widget(scrollview)
```

```
delivery_layout = BoxLayout(
    orientation='horizontal',
    size_hint=(1, None),
    height='50sp',
    spacing='10sp'
)

delivery_label = Label(
    text='Спосіб доставки:',
    font_size='18sp'
)
delivery_layout.add_widget(delivery_label)

self.delivery_options = [
    'Самовивіз',
    'Кур\'єрська доставка'
]  # Список доступних способів доставки

self.delivery_spinner = Spinner(
    text='Виберіть спосіб доставки',
    values=self.delivery_options,
    font_size='18sp'
)
delivery_layout.add_widget(self.delivery_spinner)

layout.add_widget(delivery_layout)

discount_layout = BoxLayout(
    orientation='horizontal',
```

```
            size_hint=(1, None),
            height='50sp',
            spacing='10sp'
        )

        discount_label = Label(
            text='Знижка (%):',
            font_size='18sp'
        )
        discount_layout.add_widget(discount_label)

        self.discount_input = TextInput(
            hint_text='Введіть знижку',
            font_size='18sp'
        )
        discount_layout.add_widget(self.discount_input)

        layout.add_widget(discount_layout)

        order_button = Button(
            text='Замовити',
            font_size='20sp',
            size_hint=(1, None),
            height='40sp'
        )
        order_button.bind(on_press=lambda x: self.place_order(name_input.text,
address_input.text))
        layout.add_widget(order_button)

        self.total_label = Label(
```

```
            text='',
            font_size='18sp',
            size_hint=(1, None),
            height='40sp'
        )
        layout.add_widget(self.total_label)

        # Кнопка для перегляду історії замовлень
        view_orders_button = Button(
            text='Історія замовлень',
            font_size='20sp',
            size_hint=(1, None),
            height='40sp'
        )
        view_orders_button.bind(on_press=lambda x: self.view_orders())
        layout.add_widget(view_orders_button)

        self.open_database()  # Відкриваємо з'єднання з базою даних

        return layout

    def open_database(self):
        self.connection = sqlite3.connect('orders.db')
        self.cursor = self.connection.cursor()

        # Створюємо таблицю orders, якщо вона не існує
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS orders (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                name TEXT,
```

```
            address TEXT,
            products TEXT,
            delivery TEXT,
            total_cost INTEGER
        )
    ''')
    self.connection.commit()

def close_database(self):
    if self.cursor:
        self.cursor.close()

    if self.connection:
        self.connection.close()

def place_order(self, name, address):
    selected_products = []
    total_cost = 0

    for checkbox in self.checkboxes:
        if checkbox.active:
            selected_products.append(checkbox.label.text.split('-')[0].strip())
            total_cost += int(checkbox.label.text.split('-')[1].strip().replace('грн.', ''))

    delivery_option = self.delivery_spinner.text
    discount = self.discount_input.text

    if not name or not address or not selected_products or not delivery_option:
        error_text = "Будь ласка, заповніть всі поля, виберіть хоча б один товар та виберіть спосіб доставки."
```

```
            error_popup = self.create_popup(
                'Помилка',
                error_text
            )
            error_popup.open()
            return

        if discount:
            try:
                discount = int(discount)
                total_cost -= (total_cost * discount) / 100
            except ValueError:
                error_text = "Будь ласка, введіть знижку у форматі цілого числа."
                error_popup = self.create_popup('Помилка', error_text)
                error_popup.open()
                return

        self.cursor.execute('''
            INSERT INTO orders (name, address, products, delivery, total_cost)
            VALUES (?, ?, ?, ?, ?)
        ''', (name, address, ', '.join(selected_products), delivery_option, total_cost))
        self.connection.commit()

        confirmation_text = f"Замовлення розміщено!\n\nІм'я: {name}\nАдреса:
{address}\n\nВибрані товари:\n"
        confirmation_text += '\n'.join(selected_products)
        confirmation_text += f"\n\nСпосіб доставки: {delivery_option}"
        confirmation_text += f"\n\nЗагальна вартість: {total_cost} грн."
        confirmation_popup = self.create_popup('Підтвердження замовлення',
confirmation_text)
```

```
        confirmation_popup.open()

    def view_orders(self):
        self.cursor.execute('SELECT * FROM orders')
        orders = self.cursor.fetchall()

        if orders:
            order_text = "Історія замовлень:\n\n"
            for order in orders:
                order_text += f"ID: {order[0]}\n"
                order_text += f"Ім'я: {order[1]}\n"
                order_text += f"Адреса: {order[2]}\n"
                order_text += f"Товари: {order[3]}\n"
                order_text += f"Спосіб доставки: {order[4]}\n"
                order_text += f"Загальна вартість: {order[5]} грн.\n"
                order_text += "\n"

            popup = self.create_popup('Історія замовлень', order_text)
            popup.open()
        else:
            popup = self.create_popup('Історія замовлень', 'Немає доступних
замовлень.')
            popup.open()

    def create_popup(self, title, content):
        popup_layout = BoxLayout(orientation='vertical', padding='10sp',
spacing='10sp')
        popup_label = Label(text=content, font_size='18sp', size_hint=(1, None),
height='400sp')
        popup_layout.add_widget(popup_label)
```

```
        close_button = Button(text='Закрити', font_size='18sp', size_hint=(1, None), height='40sp')
        close_button.bind(on_press=lambda x: popup.dismiss())
        popup_layout.add_widget(close_button)

        popup = Popup(title=title, content=popup_layout, size_hint=(None, None), size=(1000, 500))
        return popup

    def on_stop(self):
        self.close_database()


if __name__ == '__main__':
    OrderApp().run()
```