Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Лисицького Олега Костянтиновича*
(ПІБ)

академічної групи *121-18-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка телеграм-бота інтернет-магазину на основі Python*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Реута О. В.*			
розділів:				
спеціальний	*доц. Реута О. В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-18-1* *Лисицького О.К.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка телеграм-бота інтернет-магазину на основі Python*

затверджена наказом ректора НТУ «ДП» від *18.05.2022* № *268-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц. Реута О. В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Лисицький О.К.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 73 с., 41 рис., 1 табл., 3 дод., 20 джерел.

Об'єкт розробки: телеграм-бот інтернет-магазин на основі Python.

Мета кваліфікаційної роботи: розробка сучасного застосунку для реалізації тематичного інтернет-магазину у звичайному месенджері.

У вступі розглядаються наступні позиції: постановка проблеми, галузь використання, а також підтвердження актуальності теми.

У першому розділі розглядаються предметна галузь та існуючі технології, встановлюється актуальність завдання та призначення розробки, визначається постановка завдання.

У другому розділі вже є розглянутими існуючі технології, обирається платформа розробки, виконується проектування та розробка застосунку, наводиться опис структури та алгоритмів програми, визначаються вхідні і вихідні дані, наводиться опис технічних засобів.

В економічному розділі визначається трудомісткість розробленого програмного продукту, підраховується вартість роботи зі створення додатку та визначається час на його написання.

Практичне значення полягає у розробці застосунку, який покращує досвід користування магазином у звичайного користувача.

Актуальність даного програмного продукту визначається стрімким розвитком індустрії бот-спілкувань, а також сучасною тенденцією до мінімізації інтерфейсу для інтуїтивно зрозумілого сприйняття інформації.

Список ключових слів: ЗАСТОСУНОК, КОМП'ЮТЕР, ІНТЕРНЕТ, МАГАЗИН, БОТ, СПІЛКУВАННЯ, ІНТЕРФЕЙС, МЕСЕНДЖЕР.

ABSTRACT

Explanatory note: 73 p., 41 figs., 1 table, 3 appx., 20 sources.

Object of development: telegram-bot online store based on Python.

The purpose of the qualification work: development of a modern application for the implementation of the thematic online store in the usual messenger.

The introduction considers the following positions: problem statement, field of application, as well as confirmation of the relevance of the topic.

The first section examines the subject area and existing technologies, establishes the relevance of the task and the purpose of development, determines the task.

The second section already discusses existing technologies, selects a development platform, performs design and development of the application, describes the structure and algorithms of the program, determines the input and output data, describes the technical means.

The economic section determines the complexity of the developed software product, calculates the cost of creating an application and determines the time to write it.

Of practical importance is the development of an application that improves the experience of using the store for the average user.

The relevance of this software product is determined by the rapid development of the bot communication industry, as well as the current trend towards minimizing the interface for intuitive perception of information.

List of keywords: APPLICATION, COMPUTER, INTERNET, SHOP, BOT, COMMUNICATION, INTERFACE, MESSAGE.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення;

ОС – операційна система;

БД – база даних;

ЧБ – чат бот;

МП – мова програмування;

ПК – персональний комп'ютер;

ЦП – центральний процесор;

ЕОМ – електронно-обчислювальна машина;

API – Application Programming Interface;

TG – Telegram;

B2B – Business to Business;

UX – User Experience;

SW – Software;

SD – Software Development;

CB – Chat Bot;

JSON – JavaScript Object Notation;

ID – Identifier;

GB – Gigabyte;

TB – Terabyte.

ЗМІСТ

ВСТУП

Завдання даної кваліфікаційної роботи є безпосередньо пов’язаним з підготовчим напрямом та відповідає узагальненій тематиці кваліфікаційних робіт, а також переліку зазначених виробничих функцій, типових задач діяльності, умінням та компетенціям, якими повинний володіти бакалавр напряму 121 «Інженерія програмного забезпечення».

Ще з давніх часів людство шукає можливості обміну власних товарів та послуг на інші речі для отримання користі, вигоди та комфорту для свого існування. З плином часу мета не змінюється, однак з розвитком людських можливостей форма обміну товарами стає все більш незвичайною, цікавішою та різноманітнішою, надаючи більший спектр послуг для учасників будь-якого процесу товаро-обміну. Раніше люди обмінювали товар на інший шляхом бартеру, пізніше це переросло у грошові стосунки, адже гроші є універсальним засобом для обміну на будь-які товари. Починають з’являтися ринки, магазини, супер-маркети, де люди отримали можливість обирати товари та витрачати власні кошти.

Для зручності закупівель людство почало переходити як на електронні гроші, так і на електронні магазини, що одразу ж вирішило проблеми звичайних маркетів, прибравши такі речі, як черги, пошуки решти та необхідність покидати дім для задоволення повсякденних потреб. З розвитком прогресу навіть звийчані розмови з друзями та сім’єю перейшли до формату онлайн, який нам надають спеціальні застосунки - месенджери. Одним з найвідоміших та найпопулярніших месенджерів сучасності є Telegram, розроблений всесвітньо відомим підприємцем та програмістом Павлом Дуровим. Цей застосунок займає позицію у топ-5 найбільш завантажуваних застосунків як на операційній системі Android, так і на IOS.

Поєднавши людські потреби та бажання проводити більше часу онлайн, постало питання розробки онлайн-магазину для закупівлі товарів через

месенджер Telegram за допомогою вбудованого API, яке надається для створення розумних ботів для вирішення будь-яких потреб.

Наразі, одними з технологічно найрозвинутішими людьми є кібер-спортсмени. Саме на них націлений асортимент, дизайн та функціонал створеного магазину. Більше не потрібно обирати необхідний магазин серед тисяч існуючих в інтернеті. Усі тематичні товари зібрані в одному, багатофункціональному чаті одночасно у смартфоні та комп'ютері.

Підсумовуючи все вище сказане було обрано завдання для кваліфікаційної роботи «Розробка телеграм-бота інтернет-магазину на основі Python». Саме ця мова програмування допоможе швидко вирішити поставлену задачу. Вона сучасна, не потребує великих обсягів коду і саме головне, мова є рідною для всесвітньо відомого месенджера Telegram.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ
1.1 Загальні відомості з предметної галузі

Інтернет-магазини прийнято вважати одним із найважливіших досягнень людства у сфері високих технологій. І це - небезпідставно, адже вони допомагають економити час і гроші, тому що полегшують людям пошук товару (його можна знайти, не виходячи з дому), а вартість товару в інтернет-магазинах, як правило, набагато нижча, ніж у звичайних, адже представники онлайн-трейдингу працюють безпосередньо з виробниками продукції, і їм не треба переплачувати за утримання торгової площі. Ще одна заслуга інтернет-магазинів у тому, що завдяки ним з'явилися електронні системи оплати та електронні гаманці.

Прообраз сучасних інтернет-магазинів з'явився набагато раніше, ніж з'явилося світове павутиння. Ще 1979 року Майкл Альдріх, який постачав до Великобританії комунікаційні мережі, винайшов «відеотекст». Ця технологія запускала пряму трансляцію каталогів з товаром та контактними даними, за якими можна зв'язатися для покупки цих товарів.

Томсон Холідейс у 1981 році вже задумався над ідеєю створення B2B (англ. Business to Business, у перекладі «бізнес для бізнесу») - цей термін визначає вид економічної та інформаційної взаємодії, що класифікується залежно від типу суб'єктів, що взаємодіють.

Перший інтернет-магазин в тому вигляді, як ми звикли бачити їх зараз, з'явився в 1992 році, коли Чарльз Стек використав всі можливості B2B і створив інтернет-магазин книжкової продукції. Чому книжки? Відповідь проста – тоді книжки купувалися за паперовими каталогами з цінами та коротким описом товару (прямо як на будь-якій сторінці з продукцією будь-якого інтернет-магазину). Ці каталоги потрібно було лише оцифрувати – перенести у формат, зрозумілий для всесвітньої павутини, що й зробив Чарльз. Ще однією незаперечною перевагою було те, що книги не мали обмеження щодо терміну

придатності і не піддавалися різким змінам у плані попиту через будь-які нові модні тенденції тощо. Багато інтернет-магазинів, які зароджувалися на початку 90-х, розпочинали свою діяльність саме з продажу друкованої продукції.

У 1994 році стартував проект «Amazon» Джеффа Безоса, який вигадав його загальну концепцію, доки добирався до Сіетлу поїздом. Сайт було запущено 16 липня 1995 року. Саме тоді через інтернет стали продавати велику кількість речей, які раніше можна було придбати лише у звичайному магазині. Пізніше він позиціонувався як інтернет-магазин побутової техніки. Хоч код інтернет-ресурсу ще не був повністю доведений до робочого стану, цей магазин отримав загальне визнання, про що свідчить і те, що його творець отримав у 1999 році звання «Людина року», яке присудив йому журнал "Time".

Величезну роль для розвитку інтернет-магазинів відіграла поява банківських карток, за допомогою яких можна робити покупки дистанційно. Якщо раніше багато хто писав номер своєї карти в інтернеті з обережністю, то зараз все більше людей довіряє цьому способу оплати і не сумнівається в його безпеці.

Але, мабуть, найважливіша причина популярності магазинів у мережі – це економія часу, яку вони нам забезпечують. Здійснюючи покупки в інтернеті, не потрібно нікуди їхати, стояти у черзі чи пробці. Весь час, який ви витрачаєте на покупку, - це вибір потрібного вам товару в магазині. Не дивно, що онлайн-шопінг дуже зручний для ділових людей, тих, хто не має можливості їздити в магазин, і просто для тих, хто цінує свій час і не любить виходити з дому, проводячи свій вільний час в онлайні.

Сучасна людина проводить дуже багато часу зі своїми електронними пристроями. Ця тенденція призвела до того, що більшість користувачів бажає отримувати велику кількість інформації з глобальної мережі інтернет, користуючись інтуїтивно зрозумілим інтерфейсом. На допомогу таким людям починають з’являтися чат-боти, які можуть не тільки підтримати розмову, а й виконати нескладні дії після користувацького запиту.

Проте, незважаючи на стрімке поширення, чат-боти – це не сучасний винахід. Перші чат-боти з'явилися понад півстоліття тому і розвивалися протягом багатьох років.

1950 року Алан Т'юрінг, піонер комп'ютерів, написав наукову статтю під назвою «Обчислювальні машини та інтелект». У статті вчений мав на увазі, що комп'ютерна програма може думати та говорити як людина. Щоб довести це, Т'юрінг запропонував експеримент під назвою «Імітаційна гра», який сьогодні відомий як тест Т'юрінга.

В експерименті Т'юрінга людина, призначена суддею, розмовляла за допомогою комп'ютера з людиною і машиною, яких не можна було побачити. Завдання судді полягало в тому, щоб відрізнити комп'ютер від реальної людини. Т'юрінг припустив, що якщо суддя не може сказати, які відповіді належать комп'ютеру, це доведе, що комп'ютер здатний імітувати людську мову. Т'юрінг вважав, що до 2020 року машини зможуть легко пройти його випробування.

У 1988 році програміст-самоук Ролло Карпентер створив Jabberwacky. То була програма, призначена для розважальної імітації людської розмови. Jabberwacky навчався на минулому досвіді та з часом розвивався. Він відображав особистість та поведінку користувачів.

З 2010 року, коли Apple запустила Siri, кількість віртуальних помічників зростала. Siri стала першим персональним помічником, доступним у всьому світі. Компанія Google пішла стопами Apple, випустивши Google Now у 2012 році. У 2014 році були випущені Microsoft Cortana та Amazon Alexa.

Вже у 2016 році Facebook відкрив свою платформу Messenger для чат-ботів. Це сприяло розвитку платформ чат-ботів. На даний момент у Messenger налічується понад 300 000 активних роботів, які кожної хвилини задовольняють потреби користувачів.

Історія Telegram розпочалася у 2011 році. За словами засновника Павла Дурова, ідея створити месенджер виникла, коли він особисто зіткнувся із цензурою та незахищеністю даних. Першою метою було створення приватної можливості для обміну повідомленнями, а згодом ціль перетворилась на

«зробити месенджер швидшим і зручнішим, ніж WhatsApp», який завжди був на передньому плані.

Вже до листопада 2013-го кількість встановлень месенджера наблизилась до мільйона. Команда почала доопрацьовувати функціональність і вигадала опції, яких ніколи не було в WhatsApp. 2015 року всередині месенджера з'явилася платформа для створення ботів. Вони використовувалися в розважальних цілях: завантажували музику та генерували меми. Того ж року з'явилися публічні канали, які виділили Telegram серед інших месенджерів. Будь-який користувач міг зробити власний майданчик для публікації контенту та залучення передплатників. Так у Telegram почав зароджуватися ринок реклами.

За кількістю опцій Telegram помітно випереджав головного конкурента. WhatsApp критикували за слабкий інтерфейс та недоопрацьований UX. Сам Дуров засуджував месенджер за обмеження кількості людей у групових чатах і відсутність функції крос-девайс (коли месенджер одночасно працює на декількох пристроях).

Один із головних аргументів Дурова проти WhatsApp – слабкий захист даних. Довгий час листування не мало взагалі жодного захисту. І навіть зараз, після впровадження наскрізного шифрування, WhatsApp має доступ до повідомлень користувачів - адже всі чати зберігаються на хмарі резервного копіювання.

«Не має значення скільки існує інших месенджерів, якщо всі вони погані», - так у 2015 році Дуров відповів на запитання журналіста про конкурентну перевагу Telegram.

1.2 Призначення розробки та галузь застосування

«Розробка телеграм-бота інтернет-магазину на основі Python» - назва програмного додатку, створеного для кваліфікаційної роботи.

Основні терміни та ключові слова:

Розробка програмного забезпечення (англ. software development) - діяльність із створення нового програмного забезпечення. Розробка програмного забезпечення як інженерна дисципліна є складовою програмної інженерії, поруч із дисциплінами, відповідальними за функціонування та супроводження програмних продуктів.

Програмне забезпечення (ПЗ) - програма або безліч програм, що використовуються для керування комп'ютером. Програмне забезпечення є одним із видів забезпечення обчислювальної системи, поруч із технічним (апаратним), математичним, інформаційним, лінгвістичним, організаційним, методичним і правовим забезпеченням. Академічні області, що вивчають програмне забезпечення, - це інформатика та інженерія програмного забезпечення. У комп'ютерному сленгу часто використовується слово "софт", що походить від англійського слова "software".

Telegram - кросплатформна система миттєвого обміну повідомленнями (месенджер) з функціями VoIP, що дозволяє обмінюватися текстовими, голосовими та відеоповідомленнями, стікерами та фотографіями. Також можна здійснювати відео- та аудіодзвінки та трансляції в каналах та групах, організовувати конференції, розраховані на багато користувачів. За допомогою ботів функціонал програми практично не обмежений. Клієнтські програми Telegram доступні для Android, iOS, Windows, macOS та GNU/Linux. Кількість щомісячних активних користувачів сервісу станом січень 2021 року становить близько 500 млн людей.

Віртуальний співрозмовник, програма-співрозмовник, чатбот - програма, яка з'ясовує потреби користувачів, а потім допомагає їх задовольнити. Автоматичне спілкування з користувачем здійснюється за допомогою тексту або голосу. Чатбот веде комунікацію від імені компанії або бренду з метою спростити онлайн-спілкування (надати актуальну інформацію в найбільш оперативні терміни), використовується як альтернатива листуванню з живим оператором або дзвінку менеджеру компанії.

Інтернет-магазин - сервіс, що торгує товарами через мережу Інтернет. Дозволяє користувачам онлайн, у своєму браузері або через мобільний додаток, сформувати замовлення на покупку, вибрати спосіб оплати та доставки замовлення, сплатити замовлення. При цьому продаж товарів здійснюється дистанційним способом.

Python - високорівнева мова програмування загального призначення з динамічною строгою типізацією та автоматичним управлінням пам'яттю, орієнтована на підвищення продуктивності розробника, читаності коду та її якості, і навіть забезпечення переносимості написаних программ. Мова є повністю об'єктно-орієнтованою. Незвичайною особливістю мови є виділення блоків коду пробільними відступами. Синтаксис ядра мови мінімалістичний, за рахунок чого рідко виникає необхідність звертатися до документації. Сама ж мова інтерпретована і використовується в тому числі для написання скриптів.

Розроблений продукт може використовуватися поціновувачами відео-ігор та кібер-спортсменами за допомогою мобільних пристроїв, персональних комп'ютерів, браузерного web-застосунку. Чат-бот вміщує у собі необхідний асортимент від всесвітньо відомих брендів, а отже він є не лише зручним, а й безпечним з гарантією якості від передових компаній. Користувач завжди зможе подивитися історію розмови з ботом. Його чат є захищеним від можливості викрадання особистих даних.

1.3 Підстави для розробки

Відповідно до освітньої програми та навчального плану, графіків навчального процесу, в кінці навчання студент (майбутній бакалавр) виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується з наказом ректора.

Таким чином підставами для виконання кваліфікаційної роботи є:

- освітня програма спеціальності 121 «Інженерія програмного забезпечення»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету “Дніпровська політехніка” № 268-с від 18.05.2022р;
- завдання на кваліфікаційну роботу на тему “Розробка телеграм-бота інтернет-магазину на основі Python”.

1.4 Постановка завдання

Метою кваліфікаційної роботи є створення телеграм-бота інтернет-магазину на основі Python для продажу товарів геймерського призначення без необхідності виходити з дому та зайвих пошуків серед звичайної комп'ютерної техніки.

Під час роботи необхідно розробити інтуїтивно зрозумілий застосунок з відповідним стилем та дизайном, який буде адаптованим під різні платформи та доступним для усіх типів користувачів.

Кінцевий продукт повинен мати наступний функціонал:

- реєстрація користувача;
- обирання необхідних категорій і товарів;
- складання товарів у кошик;
- можливість замовлення товарів з відомостями щодо оплати.

Наведений програмний додаток буде розроблено за допомогою високорівневої мови програмування Python, яка є сучасною, швидкою, гнучкою, має велику кількість вбудованих та додаткових бібліотек і модулів, поширену спільноту та незалежність від платформи.

Для роботи з TelegramBotAPI (інтерфейс на основі HTTP, створений для розробників ботів для Telegram) була обрана бібліотека telebot, яка є простою та

зрозумілою для звичайного користувача. Вона працює на використанні звичайних функцій та декораторів для відслідковування команд людини.

Для роботи з операційною системою буде використовуватися вбудований модуль os (operation system, що в перекладі означає «операційна система»), який дозволить проводити необхідні маніпуляції з файловою системою серверу.

Цей продукт дозволить користувачеві порівнювати товари за характеристиками та зовнішнім виглядом, а керуватися буде за допомогою кнопок щоб позбавити клієнта необхідності вводити текст та надавати інші довгі команди.

1.5 Вимоги до програми або програмного виробу

1.5.1 Вимоги до функціональних характеристик

Готова робота повинна мати та дотримуватися наступних функціональних вимог:

- кросплатформність;
- інтуїтивний, простий інтерфейс;
- сучасний, молодіжний дизайн;
- індивідуальна (окрема) сторінка;
- висока швидкість відгуку;
- можливість керування вбудованою клавіатурою;
- можливість порівнювати декілька товарів одночасно;
- можливість зробити онлайн замовлення.

1.5.2 Вимоги до інформаційної безпеки

Готова робота повинна дотримуватися наступних вимог до інформаційної безпеки:

- конфіденційність інформації;

– відокремленість робочого простору;
– неможливість потрапляння до окремої бази даних;
– використання користувацьких даних лише для оформлення замовлення.

Робота телеграм боту можлива лише за наявності спеціально згенерованого токену (ключа). Можливість використання програмного коду іншою особою відсутня.

1.5.3 Вимоги до складу та параметрів технічних засобів

Застосунок розроблено на базі програмного забезпечення Telegram, що є доступним на наступних платформах:

– Windows
– macOS
– Linux
– Android
– iOS

Для роботи застосунку необхідні відповідні технічні характеристики та обладнання для ПК:

– центральний процесор (ЦП): Intel Core i3;
– внутрішній накопичувач: 5 Гб;
– оперативна пам’ять: 2 Гб.

Для роботи застосунку на мобільних пристроях необхідні відповідні версії операційних системем:

– Android 6.0
– iOS 7

1.5.4 Вимоги до інформаційної та програмної сумісності

Готова робота написана використовуючи високорівневу мову програмування Python у поєднані з бібліотекою telebot для самого боту, модулем os для операційної системи та нереляційною базою даних MongoDB для збереження необхідної інформації. Для підключення до коду MongoDB була використана бібліотека pymongo.

Програмний код має відповідати усім правилам та стандартам, найвідомішим з яких є PEP8. Для редагування та тестування коду було використане спеціальне програмне забезпечення Visual Studio Code, розроблене компанією Microsoft. Саме цей редактор коду має вбудовану, сертифіковану можливість створювати програми з використанням мови Python.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1 Функціональне призначення програми

Результатом цієї кваліфікаційної роботи має бути програмний застосунок, що повінстю замінює звичайний магазин, відтворюючи усі основні функції з вибору та порівняння товару, замовлення та отримання товару, підрахунку та оплати суми замовлення.

Кінцевим продуктом є функціональне віконце чату у месенджері Telegram із супроводжуючим текстовим та ілюстраційним оформленням кроків користувача, а також багатофункціональною клавіатурою для переходу на різні стадії роботи магазину.

Основне призначення застосунку:

- зробити сучасний, інтуїтивний дизайн
- відокремити кібер-спортивну тематику від іншої електротехніки
- змінити представлення щодо складності та небажання робити покупки

Для реалізації поставленого завдання застосунок має бути швидким, лаконічним, однак у цей же час зберігати у собі усю корисну інформацію та функціонал не обмежений, порівнюючи із фізичним чи онлайн магазином.

2.2 Опис застосованих математичних методів

Під час роботи з інтернет магазином користувача в першу чергу цікавить можливість оплати та отримання власного замовлення. Саме на моменті оплати він зустрінеться із застосуванням математичних методів для підрахунку фінальної суми, яка буде додана до замовлення.

Кожний товар телеграм-бота інтернет-магазину має власну ціну, вказану у гривні, а також, при створенні замовлення, користувачеві необхідно ввести бажану кількість товару, яку він хоче придбати.

Виходячи з цього, є необхідним підахувати спочатку загальну суму у гривні по кожній позиції, доданій до кошика. Після чого поєднати усі зібрані суми до єдиного фінального чеку. Ціна на товари має помножуватися на кількість продукції, після чого, отримані суми також додаються.

Візьмемо деякий товар за t_i, а його кількість за n_j. Тоді розрахунок фінальної суми оплати матиме вигляд:

$$S = (t_1 * n_1) + (t_2 * n_2) + (t_i * n_j) + \cdots \quad (2.1)$$

Також слід зазначити, що сума кожного товару складається з урахунком звичайної оптової закупівлі, а також коштів необхідних для забезпечення працездатності системи. Інтернет-магазин чат-бот не має великої кількості персоналу, а отже ціни для користувача завжди нижчі за середні по ринку.

2.3 Опис використаної архітектури та шаблонів проектування

Для легкого для швидкого розуміння програми необхідно звертатися до логічного розбиття програмного коду, файлової системи та архітектури бази даних, щоб використовувати менше ресурсів системи, а також автоматизувати однотипні дії з даними.

Що стосується файлової частини, вона має поділятися на три окремі частини проекту:

- основна частина – з перевіркою введеної команди та реакцією на запит користувача
- клавіатурна частина – вміщує у себе набір усіх необхідних клавіатур для інтуїтивного використання програми
- частина налаштувань – відповідає за підключення бази даних, а також скретного ключа (токену) для функціонування боту

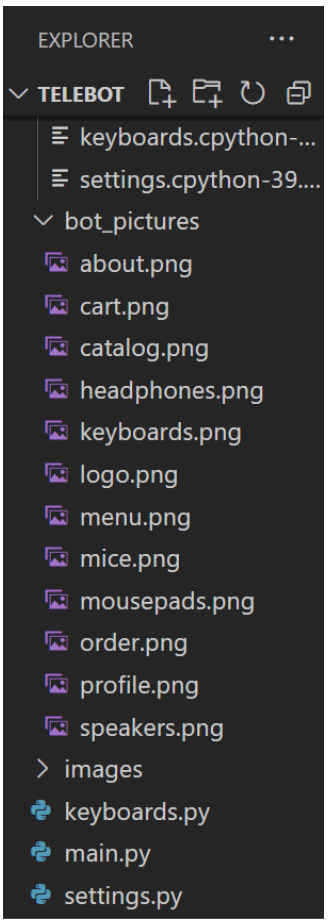

Рис. 2.1 Файлова (кодова) система проєкту

Таблиця 2.1

Опис файлів проєкту

Назва	Призначення	Об'єм
main.py	Файл. Відповідає за запуск та основну логіку програми	21 кілобайт
keyboards.py	Файл. Відповідає за маніпуляцію та збереження вбудованих клавіатур	5 кілобайт
settings.py	Файл. Відповідає за підключення до бази даних та API Telegram	1 кілобайт
bot_pictures	Директорія. Зберігає зображення самого бота	1.80 мегабайт
images	Директорія. Зберігає зображення товарів магазину.	27.8 мегабайт
__pycache__	Директорія. Зберігає байт-код кеш Python-файлів	4 кілобайт

Сам програмний код працює завдяки функціональному підходу, а отже кожна функція та перевірка введеної команди має знаходитися окремо одна від одної. Дані передаються між функціями як параметри та аргументи, щоб знову не знаходити необхідну інформацію, отриману в попередніх кроках. Програмний код має замкнену систему, а отже користувач завжди зможе повернутися назад, викликавши функцію, що відповідає за головне меню. Окрім звичайних функцій, у проекті використовуються так звані декоратори.

Декоратор - це патерн проектування в Python, а також функція другого рівня, тобто приймає інші функції як змінні і повертає їх. І в сам декоратор, і в функцію-обертку можна передати і позиційні, і іменовані аргументи - args і kwargs відповідно. Декоратори працюють не лише з функціями, а й із класами та методами. У самому проекті вони допомагають розрізнити, де використовується текстове введення, а де команди, надіслані за допомогою кнопок чи спеціальних символів, як наприклад «/».

```python
@bot.message_handler(commands = ["start"])
def start(message):

    def get_firstName(answer):
        reg_list["firstName"] = answer.text
        answer = bot.send_message(message.chat.id, "Введіть прізвище:")
        bot.register_next_step_handler(answer, get_lastName)

    def get_lastName(answer):
        reg_list["lastName"] = answer.text
        answer = bot.send_message(message.chat.id, "Введіть ім'я по-батькові:")
        bot.register_next_step_handler(answer, get_middleName)

    def get_middleName(answer):
        reg_list["middleName"] = answer.text
        answer = bot.send_message(message.chat.id, "Введіть номер мобільного телефону:")
        bot.register_next_step_handler(answer, get_phoneNumber)

    def get_phoneNumber(answer):
        reg_list["phoneNumber"] = answer.text
        answer = bot.send_message(message.chat.id, "Введіть email:")
        bot.register_next_step_handler(answer, get_email)
```

Рис. 2.2 Приклад використання декоратора

Робота з базою даних включає у себе розбиття на три пункти:

– база для користувачів
– база товарів
– база для підтверджених замовлень

Таке розбиття дозволяє обрати необхідну схему документів всередині бази для зручного подальшого пошуку, а також запобігає загубленню одних даних серед інших.

Файлова система із картинками за розбиттям схожа на схему з базою даних, адже кожний товар має власні зображення, що знаходяться у відповідних директоріях файлової системи. Окрім зображень товарів, існують зображення самого боту, які не мають заважати роботі користувачів з товарами.

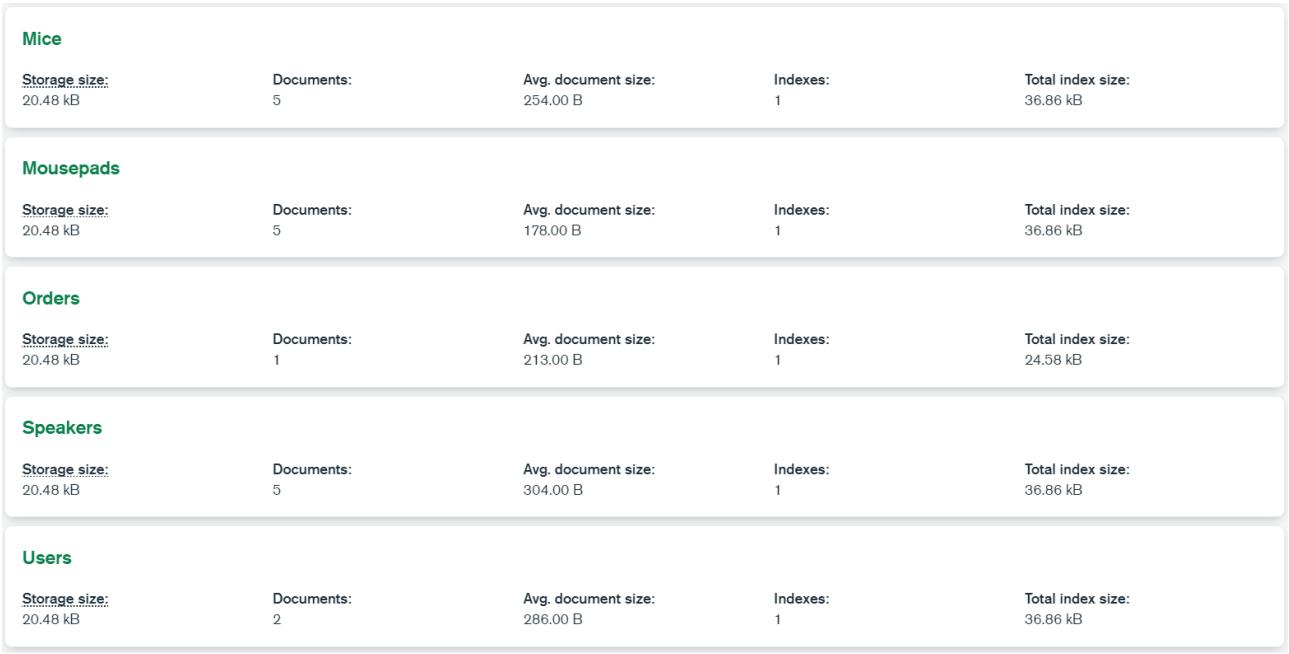

Рис. 2.3 Розподілення категорій бази даних для різних потреб

2.4 Опис використаних технологій та мов програмування

Для реалізації застосунка за темою кваліфікаційної роботи були використані наступні технології:

– мова програмування Python
– бібліотека для розробки телеграм-ботів telebot

– модуль для роботи з операційною системою os
– нереляційна база даних MongoDb
– дизайнерський онлайн-сервіс Figma

Першим завданням було обрати необхідну мову програмування для реалізації телеграм-боту. Вибір пав на Python, адже він сучасний, гнучкий та має спрощений синтаксис:

Python - високорівнева мова програмування загального призначення з автоматичним управлінням пам'яттю та динамічною строгою типізацією, орієнтована на підвищення продуктивності розробника, читання коду та його якості, а також на забезпечення переносимості написаних на ньому програм.

Мова є повністю об'єктно-орієнтованою у тому сенсі, що все усередині є об'єктами. Незвичайною особливістю мови є виділення блоків коду пробільними відступами, навідміну від стандартного вигляду з використанням фігурних та інших дужок.

Синтаксис ядра мови мінімалістичний, за рахунок чого у практиці рідко виникає необхідність звертатися до документації. Сама ж мова відома як інтерпретована і використовується в тому числі для написання скриптів.

Недоліками мови є часто нижча швидкість роботи та більш високе споживання пам'яті написаних на ньому програм порівняно з аналогічним кодом, написаним компілюваними мовами, таких як C або C++, однак Python є лідером серед написання програм для штучного інтелекту, в тому числі й для розвинених ботів. Легкість синтаксису надає можливість витрачати значно менше часу на розробку будь-якої системи.

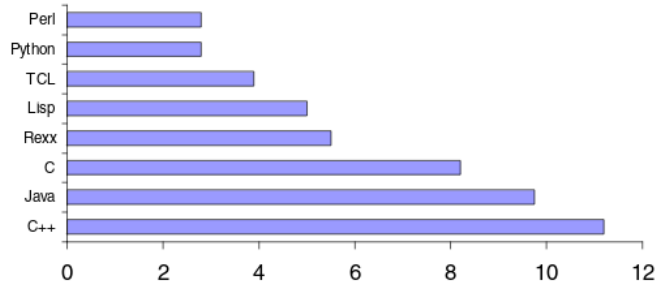

Рис. 2.4 Порівняння затраченого часу серед різних мов програмування на вирішення поставленої задачі

Python є мультипарадигмальною мовою програмування, що підтримує імперативне, процедурне, структурне, об'єктно-орієнтоване програмування, метапрограмування та функціональне програмування, яке і використовується для реалізації чат-бота інтернет-магазину. Завдання узагальненого програмування вирішуються за рахунок динамічної типізації. Аспектно-орієнтоване програмування частково підтримується через декоратори, повноцінніша підтримка забезпечується додатковими фреймворками. Такі методики як контрактне та логічне програмування можна реалізувати за допомогою бібліотек чи розширень.

Основні архітектурні риси - динамічна типізація, автоматичне управління пам'яттю, повна інтроспекція, механізм обробки винятків, підтримка багатопоточних обчислень з глобальним блокуванням інтерпретатора, високорівневі структури даних. Підтримується розбиття програм на модулі, які можуть об'єднуватися в пакети.

Стандартна бібліотека включає великий набір корисних функцій, що переносяться, починаючи з можливостей для роботи з текстом і закінчуючи засобами для написання мережевих додатків. Додаткові можливості, такі як математичне моделювання, робота з обладнанням, написання веб-додатків або розробка ігор можуть реалізовуватися за допомогою великої кількості сторонніх бібліотек, а також інтеграцією бібліотек, написаних на C або C++, при цьому і сам інтерпретатор Python може інтегруватися в проекти. Існує і спеціалізований репозиторій програмного забезпечення, написаного на Python, - PyPI. Даний репозиторій надає засоби для простого встановлення пакетів в операційну систему і став стандартом де-факто для Python. Станом на 2019 рік у ньому було понад 175 тисяч пакетів.

Python став однією з найпопулярніших мов, він використовується в аналізі даних, машинному навчанні, DevOps та веб-розробці, а також в інших сферах, включаючи розробку ігор. За рахунок читабельності, простого синтаксису та відсутності необхідності в компіляції мова добре підходить для навчання програмування, дозволяючи концентруватися на вивченні алгоритмів, концептів

та парадигм. Налагодження ж і експериментування значною мірою полегшуються тим фактом, що мова інтерпретується. Застосовується мова багатьма великими компаніями, такими як Google або Facebook. Станом на жовтень 2021 року Python посідає перше місце у рейтингу TIOBE популярності мов програмування з показником 11,27%. "Мовою року" за версією TIOBE Python оголошувався в 2007, 2010, 2018 та 2020 роках.

Dec 2021	Dec 2020	Change	Programming Language	Ratings	Change
1	3	↑	Python	12.90%	+0.69%
2	1	↓	C	11.80%	-4.69%
3	2	↓	Java	10.12%	-2.41%
4	4		C++	7.73%	+0.82%
5	5		C#	6.40%	+2.21%
6	6		Visual Basic	5.40%	+1.48%
7	7		JavaScript	2.30%	-0.06%
8	12	⇈	Assembly language	2.25%	+0.91%
9	10	↑	SQL	1.79%	+0.26%
10	13	↑	Swift	1.76%	+0.54%
11	9	↓	R	1.58%	-0.01%
12	8	⇊	PHP	1.50%	-0.62%
13	23	⇈	Classic Visual Basic	1.27%	+0.56%
14	11	↓	Groovy	1.23%	-0.30%
15	15		Ruby	1.16%	-0.01%

Рис. 2.5 Рейтинг використання різних мов програмування

Що стосується розробки самого телеграм-боту, головну роль відіграє бібліотека telebot. Це проста, але розширювана реалізація Python для Telegram Bot API. Підтримує як синхронну, так і асинхронну роботу. Telebot дійсно простий, адже для початку роботи потребує лише двох речей:

- створення об’єкту класа Telebot (що інкапсулює всі виклики API)
- отримання токену від BotFather (менеджер зі створення ботів)

BotFather - єдиний бот, який керує усіма. Використовуємо його для створення нових облікових записів ботів і керування наявними ботами. Якщо користувач є у Telegram, він може зв'язатися з BotFather відразу. Саме там створюється назва боту, а також обирається його нік-нейм (@username), за яким

у майбутньому відбуватиметься пошук бота серед усіх інших. BotFather також допомагає встановити іконку, та налаштувати опис того, що може виконувати саме цей телеграм-бот.

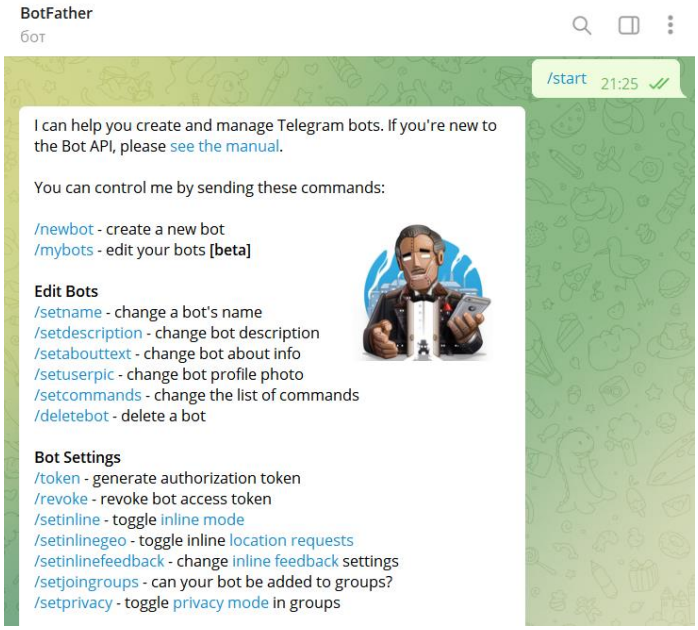

Рис. 2.6 BotFather – менеджер з керування ботами

Для роботи з операційною системою використовується модуль os. Модуль os надає безліч функцій для роботи з операційною системою, причому їх поведінка, як правило, не залежить від ОС, тому програми залишаються переносними. Цей модуль забезпечує портативний спосіб використання функцій, що залежать від операційної системи. Якщо ви просто хочете прочитати або записати файл, можна скористуватися функцією open(), якщо ви хочете маніпулювати шляхами, можна звернутися до модулю os.path.

```
1  import os.path
2
3  save_path = 'C:/example/'
4
5  name_of_file = raw_input("What is the name of the file: ")
6
7  completeName = os.path.join(save_path, name_of_file+".txt")
8
9  file1 = open(completeName, "w")
10
11 toFile = raw_input("Write what you want into the field")
12
13 file1.write(toFile)
14
15 file1.close()
```

Рис. 2.7 Використання модулю os.path для створення та запису текстового файла

Під час реєстрації у телеграм-магазині кожен користувач надає дані телеграм-боту для майбутнього оформлення та доставки замовлень. Було прийняте рішення зберігати уведені дані у вигляді json-об'єктів, адже це є доволі швидким та легким методом зберігання та передачі великої кількості даних.

JSON (JavaScript Object Notation) - текстовий формат обміну даними, що базується на JavaScript. Як і багато інших текстових форматів, JSON легко читається людьми. Формат JSON був розроблений Дугласом Крокфордом.

Незважаючи на походження від JavaScript (точніше, від підмножини мови стандарту ECMA-262 1999), формат вважається незалежним від мови і може використовуватися практично з будь-якою мовою програмування. Для багатьох мов існує готовий код для створення та обробки даних у форматі JSON. Не виключенням є і мова програмування Python.

JSON-текст в закодованому вигляді є однією з двох структур:

- Набір пар ключ-значення. У різних мовах це реалізовано як запис, структура, словник, хеш-таблиця, список із ключем або асоціативний масив. Імена ключів, що повторюються, допустимі, але не рекомендуються стандартом; обробка таких ситуацій відбувається на розсуд програмного забезпечення, можливі варіанти — враховувати лише перший такий ключ, враховувати останній такий ключ, генерувати помилку.
- Впорядкований набір значень. У багатьох мовах це реалізовано як масив, вектор, список чи послідовність.

Структури даних, що використовуються JSON, підтримуються будь-якою сучасною мовою програмування, що дозволяє застосовувати JSON для обміну даними між різними мовами програмування і програмними системами.

Рис. 2.8 Вигляд «словника» у Python, який буде збережено у форматі JSON. Під час реєстрації вільні значення заповняться даними. Ключі завжди залишаються незмінними, вони є «сталими» для користувачів у базі даних MongoDB

MongoDB - документоорієнтована система управління базами даних, яка не вимагає опису схеми таблиць. Вважається одним із класичних прикладів NoSQL-систем, використовує JSON-подібні документи та схему бази даних. Написана мовою C++. Застосовується для електронної комерції, у розробці веб-застосунків, відео-ігор та інших випадках, для яких табличний вигляд не є припустимим.

Система може бути використана як файлове сховище з балансуванням навантаження та реплікацією даних (функція Grid File System поставляється разом з драйверами MongoDB). Надаються програмні засоби для роботи з файлами та їх вмістом.

Основним інтерфейсом бази даних була командна оболонка mongo. З версії MongoDB 3.2 як графічна оболонка поставляється «MongoDB Compass». Існують

продукти та сторонні проекти, які пропонують інструменти з графічним інтерфейсом для адміністрування та перегляду даних.

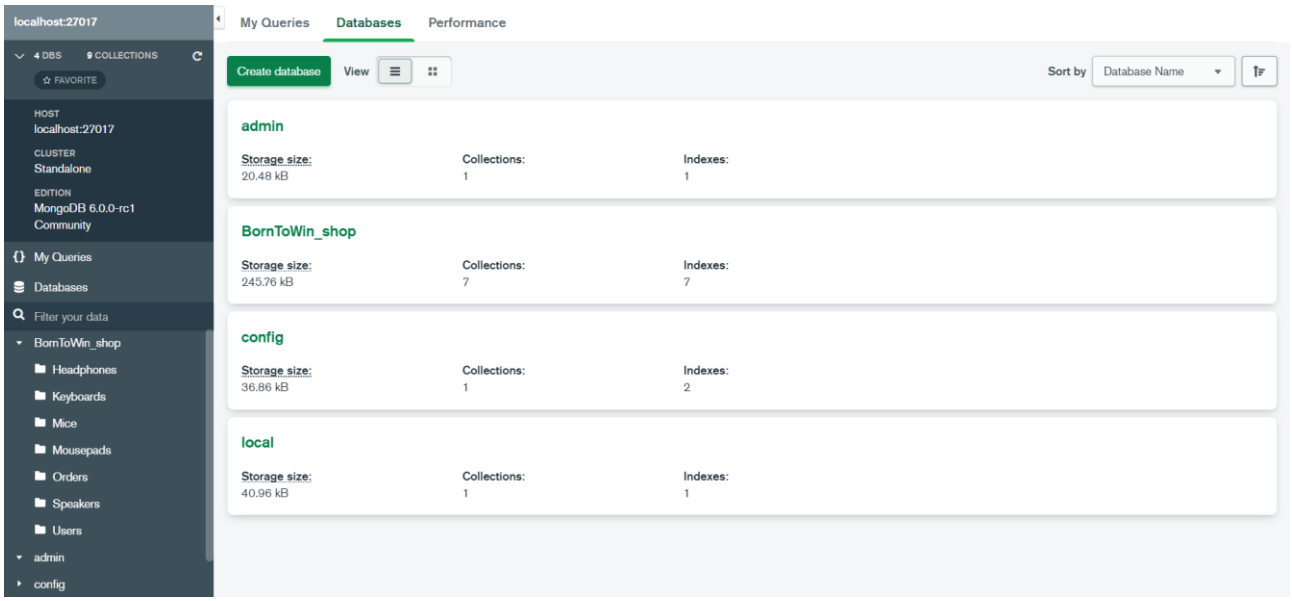

Рис. 2.9 Зовнішній вигляд «MongoDB Compass», який також використовується при створенні кваліфікаційної роботи

Для написання коду, його тестування, а також для підключення інших модулів та технологій було використано редактор коду Visual Studio Code, розроблений компанією Microsoft.

Visual Studio Code – редактор вихідного коду, розроблений Microsoft для Windows, Linux та macOS. Позиціонується як «легкий» редактор коду для кросплатформної розробки програм. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense та засоби для рефакторингу. Має широкі можливості для кастомізації: теми користувача, поєднання клавіш і файли конфігурації. Поширюється безкоштовно, розробляється як програмне забезпечення з відкритим вихідним кодом. Visual Studio Code заснований на Electron та реалізується через веб-редактор Monaco, розроблений для Visual Studio Online.

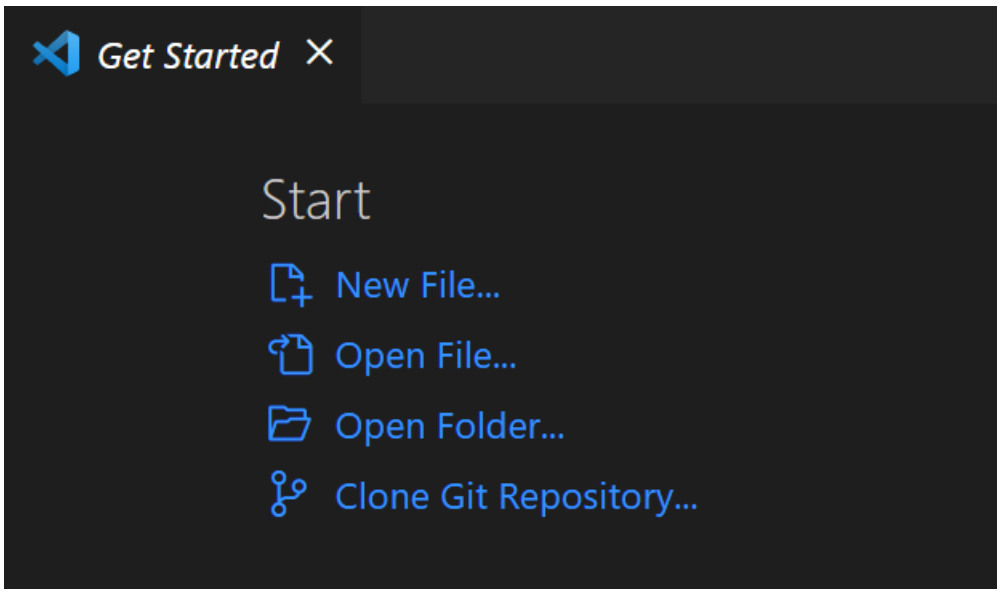

Рис. 2.10 Початкове вікно Visual Studio Code

Figma - онлайн-сервіс для розробки інтерфейсів та прототипування з можливістю організації спільної роботи у режимі реального часу. Сервіс доступний за передплатою, передбачено безкоштовний тарифний план для одного користувача. Є офлайн-версії для Windows, MacOS. Реалізовано інтеграцію з корпоративним месенджером Slack та інструментом прототипування Framer. Використовується для створення спрощених прототипів інтерфейсів, так і для детального опрацювання дизайну інтерфейсів мобільних додатків, веб-сайтів, корпоративних порталів. Саме за допомогою Figma було реалізовано усі зображення для сучасного дизайну телеграм-бота.

2.5 Опис структури системи та алгоритмів її функціювання

Готова програма поділяється на три файли – модулі, що входять до складу єдиного пакета. До модулів входять:

- main.py (відповідає за основну логіку програми, очікує команди та реагує на них в залежності від обраної дії користувачем)
- keyboards.py (відповідає за вбудовані клавіатури, завдяки яким користувач буде рухатися вздовж усього процесу співпраці з ботом)
- settings.py (відповідає за підключення самого бота з допомогою токена, а також зберігає у собі підключення до колекцій бази даних)

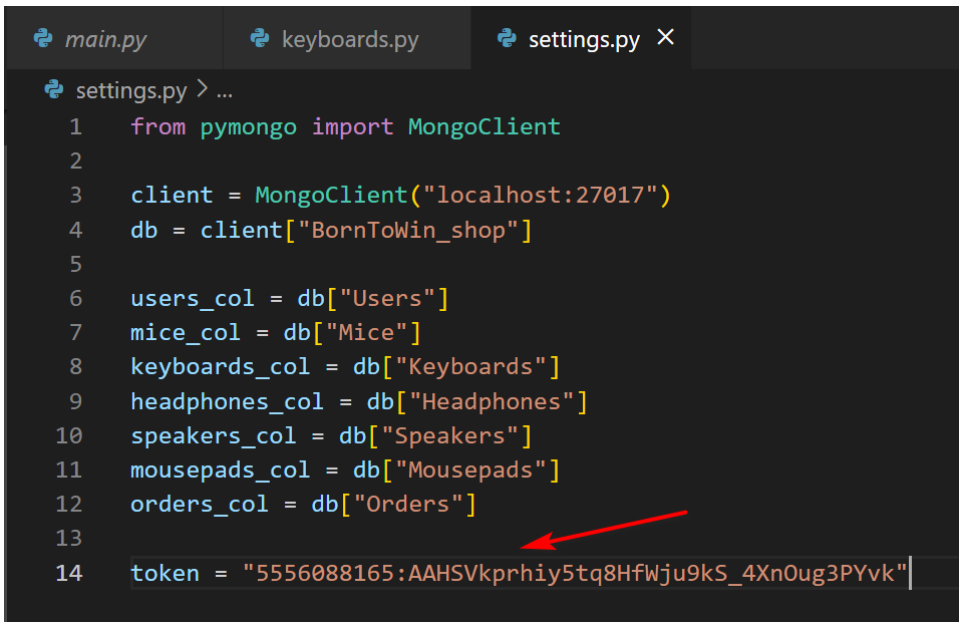

Рис. 2.11 Особистий токен для бота зберігається у модулі з налаштуваннями

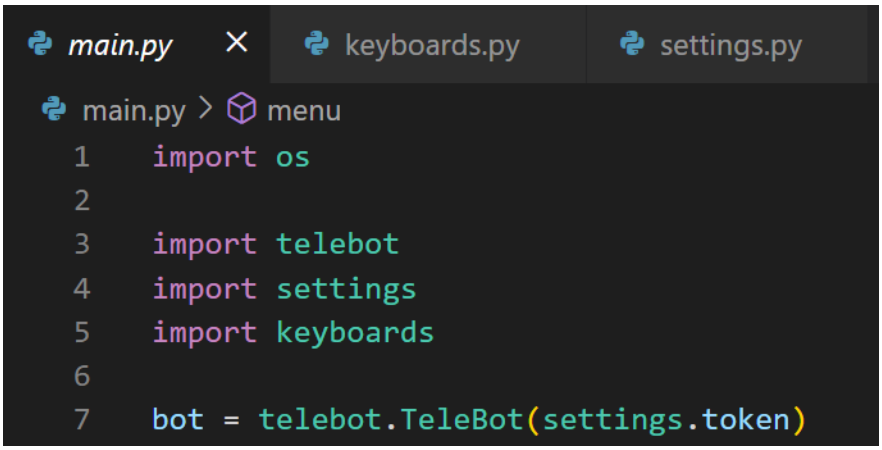

Рис. 2.12 До головного файлу програми підключені необхідні модулі, а також токен боту

Класичною командою на початку будь-якого бота є команда «/start», саме вона викликає телеграм-магазин для першої дії користувача – реєстрації. При отриманні команди за домогою відповідного декоратора «@bot.message_handler()» відбувається надсилання першого повідомлення та стартує процес послідовного отримування даних від користувача. Крок за кроком наповнюється словник з даними, який по завершенню реєстрації зберігається у базі даних MongoDB у вигляді JSON документу.

```
bot.send_message(message.chat.id, "Доброго дня, вітаємо Вас у магазині BornToWin. Будь ласка, зареєструйтеся.")

answer = bot.send_message(message.chat.id, "Введіть ім'я:")
bot.register_next_step_handler(answer, get_firstName)
```

Рис. 2.13 Перше вітальне повідомлення та перехід до покрокової реєстрації

Після того як користувач зареєструвався, він отримує можливість знаходитися в меню чат-бота, реєстрація вже не є необхідною, адже його нову сторінку прив'язано до так званого «telegram_id». Це унікальний ідентифікатор кожного користувача Telegram. Саме за цим ID відбуватиметься пошук необхідних даних клієнтів телеграм-магазину.

Майбутні дії користувача будуть переводити його до наступних сторінок інтернет-магазину. Ці дії він зможе обрати завдяки вбудованим клавіатурам. Кожна клавіатура може мати власний набір клавіш, їхнє унікальне положення, а також передавати команди основному модулю для швидкого реагування.

```
menuboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 2)
menu_btn1 = telebot.types.KeyboardButton("👤Профіль")
menu_btn2 = telebot.types.KeyboardButton("🎮Каталог товарів")
menu_btn3 = telebot.types.KeyboardButton("🛒Кошик")
menu_btn4 = telebot.types.KeyboardButton("🌐Про магазин")
menuboard.add(menu_btn1, menu_btn2, menu_btn3, menu_btn4)
```

Рис. 2.14 Приклад клавіатури для головного меню

Клавіатури у Telegram поділяються на два основних види:

– ReplyKeyboard
– InlineKeyboard

Різниця між ними полягає у тому, що ReplyKeyboard знаходиться під віконцем чату користувача та надсилає текстові повідомлення до головного файлу. Тобто для отримання такої текстової команди необхідний спеціальний декоратор «@bot.message_handler(content_type=[“text”])».

Що стосується іншої, InlineKeyboard, вона знаходиться прямо під повідомленням боту та надсилає не текстові повідомлення, а так звані «call-back», перекладаючи на українську мову – зворотні відгуки. Приклади отримання обох команд наведено нижче:

Рис. 2.15 Якщо текстове повідомлення містить деякий рядок – викликається необхідна функція (наприклад, виведення інформації профілю)

Інший декоратор «@bot.callback_query_handler()» реагує саме на зворотні відгуки, тобто напис на кнопці може відрізнятися від надісланого запиту. Користувач натискає кнопку «Додати в кошик», а необхідний відгук вилітає до декоратора:

```
if call.data == "add_to_cart":
    input_product_quantity(None, call, profile_info)
```

Рис. 2.16 Якщо людина хоче додати товар до кошику – необхідно ввести кількість бажаного товару

Окрім можливості додавання товару до кошика є функціонал щодо переглядання зображень товарів, вони також реалізовані за допомогою InlineKeyboard. Останній переглянутий товар користувачем зберігається у базу, щоб він мав можливість співпрацювати саме з актуальною позицією, що зображена на екрані. Користувач може переглядати фотографії, повертаючись до минулих, або ж обрати інший товар та порівняти характеристики обох кібер-спортивних продуктів.

Важливо відмітити, що до магазину можна додавати нові категорії товарів, нові продукти, а також будь-яку кількість зображень. Програма автоматично створює необхідні кнопки клавіатури, а також забороняє увімкнути неіснуюче зображення.

```
profile_info = settings.users_col.find_one({"tg_id": call.message.chat.id})
photo_limit = find_photo_quantity(profile_info["last_viewed"])

if call.data == "next_photo":
    if profile_info["last_photo"] < photo_limit:
        media = telebot.types.InputMediaPhoto(open(f"images/{profile_info['last_viewed']}
        bot.edit_message_media(media = media, chat_id = call.message.chat.id, message_id
        settings.users_col.update_one({"tg_id": profile_info["tg_id"]}, {"$set":{'last_ph
if call.data == "previous_photo":
    if profile_info["last_photo"] > 1:
        media = telebot.types.InputMediaPhoto(open(f"images/{profile_info['last_viewed']}
        bot.edit_message_media(media = media, chat_id = call.message.chat.id, message_id
        settings.users_col.update_one({"tg_id": profile_info["tg_id"]}, {"$set":{'last_ph
```

Рис. 2.17 Програма автоматично знаходить користувача, щоб отримати останній переглянутий товар, а також підраховує кількість зображень доступних для конкретного продукту

Після додавання товарів до кошика, користувач може або зробити замовлення, або очистити кошик, якщо він хоче внести зміни до замовлення. Увесь кошик, який клієнт обрав для замовлення та підтвердив заноситься до бази даних. Маючи усі данні щодо замовлення, представники магазину можуть зв'язатися з користувачем та довести оплату з доставленням до кінця.

```
if message.text == "✅Підтвердити замовлення":
    person_info = settings.users_col.find_one({"tg_id":message.chat.id})
    order_dict = {
        "fullName": f"{person_info['lastName']} {person_info['firstName']} {person_info['middleName']}",
        "phone_number": person_info['phoneNumber'],
        "email": person_info['email'],
        "address": person_info['address'],
        "shopping_list": person_info['shopping_list'],
        "order_sum": person_info['order_sum']
    }

    settings.orders_col.insert_one(order_dict)
    settings.users_col.update_one({"tg_id": message.chat.id}, {"$set":{"shopping_list": [], "order_sum": 0}})
    bot.send_message(message.chat.id, "Ваше замовлення прийняте! Ми зв'яжемося з вами найближчим часом. Дякуємо, що обрали наш магазин."
```

Рис. 2.18 Користувач підтверджує замовлення. Дані додаються до бази даних, а вже замовлений кошик очищується для майбутнього використання

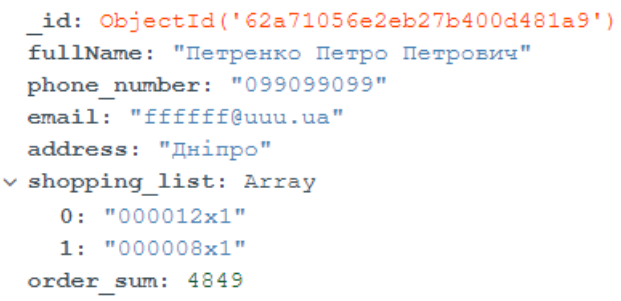

Рис. 2.19 Отримане замовлення разом із контактними даними, сумою та кількістю товарів формується у окремий документ бази даних

2.6 Обґрунтування та організація вхідних та вихідних даних програми

Під час роботи програми відбувається обмін інформацією між користувачем (клієнтом) та ботом (сервером). Користувач вводить персональні дані щоб оформити замовлення на свою електронну пошту, адресу, після чого дані переміщуються у базу даних за допомогою файлу формату JSON.

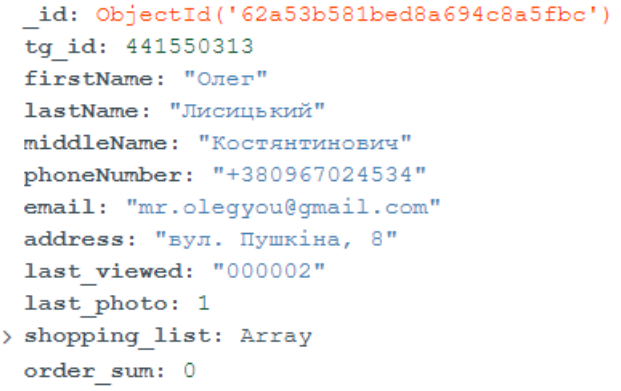

Рис. 2.20 Зовнішній вигляд користувача у формат JSON-документу в базі

Сервер (телеграм-бот), у свою чергу, надсилає користувачеві власну базу товарів, їхніх характеристик та зображень. Товари з характеристиками мають власну колекцію документів, згідно категорії товару, наприклад навушники та клавіатури знаходяться окремо один від одного.

Зображення товарів знаходяться у файловій системі самого бота. Це необхідно для якнайшвидшої взаємодії між користувачем та фотографіями, які він хоче продивитися.

Кожен товар має власний магазинний ID (ідентифікатор). Саме через нього відбувається синхронізація характеристик та зображень.

```
_id: ObjectId('62a601c924472a74b545a956')
id: "000002"
name: "Миша Razer Viper Ultimate"
sensor: "Razer Focus+"
dpi: "20000"
interface: "Радіо-канал 2.4"
power: "Акумулятор"
acceleration: "50"
weight: "74"
color: "Чорний"
price: 3999
```

Рис. 2.21 Характеристики миші з ідентифікатором «000002»

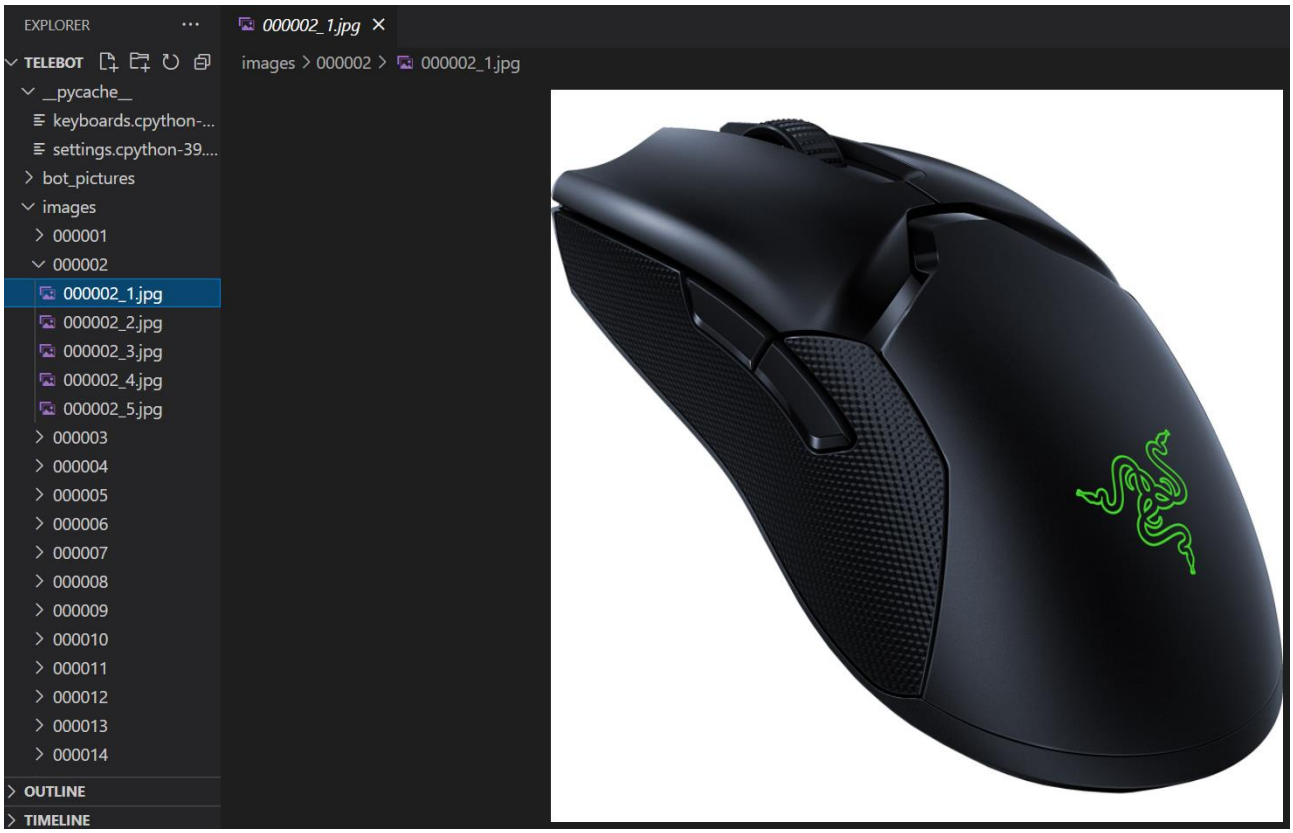

Рис. 2.22 Перелік необхідних зображень з цим самим ідентифікатором «000002»

2.7 Опис розробленого програмного продукту

2.7.1 Використані технічні засоби

Під час робзробки програмного продукту телеграм-бота інтернет-магазину на основі Python було використано наступні технічні характеристики персонального комп'ютера:

- операційна система Windows 10;
- центральний процесор Intel Core i5;
- відеоадаптер Nvidia GeForce 1050 Ti (4 GB);
- внутрішній накопичувач 1 TB;
- оперативна пам'ять 8 GB.

2.7.2 Використані програмні засоби

Для створення додатку, а також для його тестування та поєднання з іншими технологіями, було обрано спеціалізований редактор коду Visual Studio Code. Окрім програмного забезпечення для створення коду, було обрано застосунок для зручного графічного інтерфейсу бази даних MongoDB Compass.

Visual Studio Code - редактор вихідного коду, розроблений для Windows, Linux та macOS. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense та засоби для рефакторингу. Має широкі можливості для кастомізації: теми користувача, поєднання клавіш і файли конфігурації. Розповсюджується безкоштовно, розробляється як програмне забезпечення з відкритим вихідним кодом. Visual Studio Code заснований на Electron та реалізується через веб-редактор Monaco, розроблений для Visual Studio Online.

MongoDB Compass - це інтерактивний інструмент для запитів, оптимізації та аналізу даних MongoDB. Compass надає все: від аналізу схеми до оптимізації індексів і конвеєрів агрегації в єдиному централізованому інтерфейсі. Зовнішній

вигляд програми допомагає легко орієнтуватися у архітектурі бази даних, надаючи доступ до самої бази, внутрішніх колекцій (категорій), а також до документів, які складають окрему сутність всередені колекції.

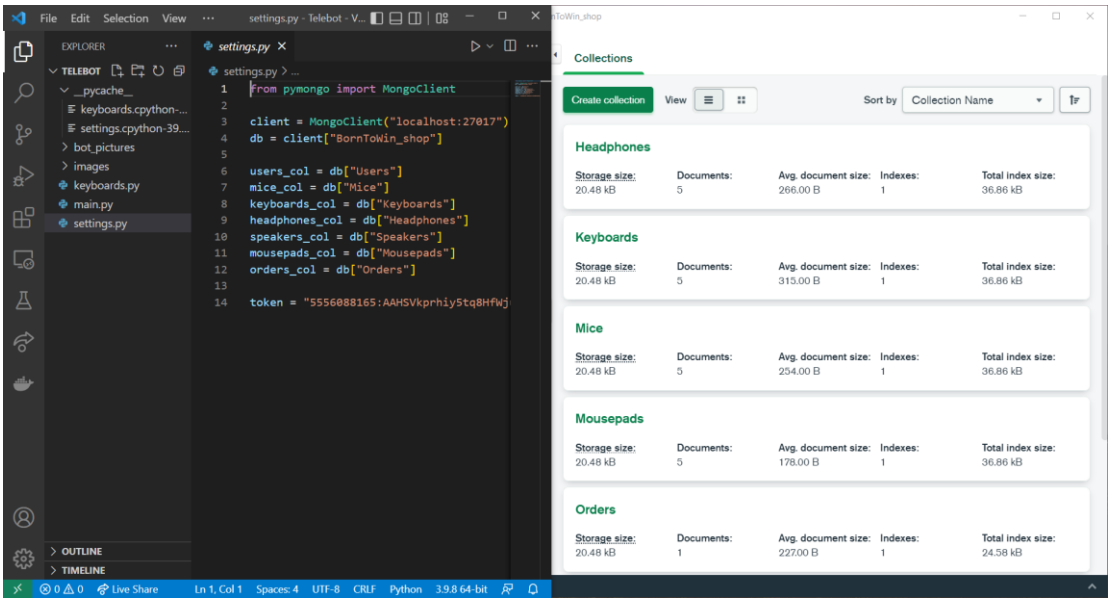

Рис. 2.23 Сумісна робота Visual Studio Code та MongoDB Compass

2.7.3 Виклик та завантаження програми

Усі необхідні файли (модулі) готового застосунку було зібрано в єдиний пакет для зручного запуску та використання на будь-якій машині-сервері. Для запуску програмного забезпечення необхідно мати:

- встановлений застосунок Telegram для можливості комунікації з ботом-магазином
- повний пакет модулів, що включає у себе головний файл, файл клавіатур та файл налаштувань, а також усі необхідні зображення для бота та товарів магазину
- встановлений Python (із налаштованими системними шляхами) мінімальної версії 3.7.4
- (бажано, але не обов'язково) встановлені застосунки для редагування коду, наприклад Visual Studio Code (для запуску програми в один клік), а також для графічного інтерефейсу бази даних MongoDB Compass

Для старту роботи бота необхідно запустити єдиний головний файл, що є мозком програми, а саме «main». Зробити це можна або ж з допомогою редактору коду, або використовуючи командний рядок cmd, якщо знаходячись у директорії проекту ввести команду «python main.py».

Запуск бота з боку користувача відбувається ще легше. Для цього необхідно зайти у месенджер Telegram та знайти бота за його @username. Натиснути кнопку «Запустити», яка надасть команду «/start», що розпочне розмову з користувачем.

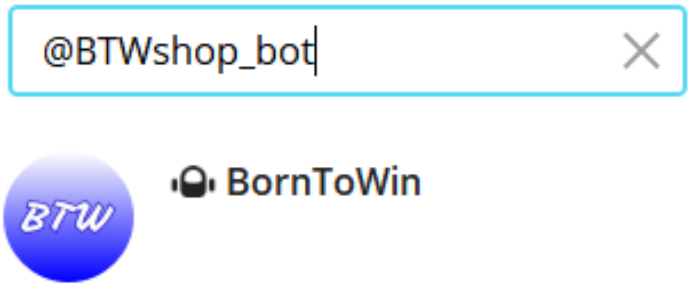

Рис. 2.24 Пошук бота серед чатів у Telegram

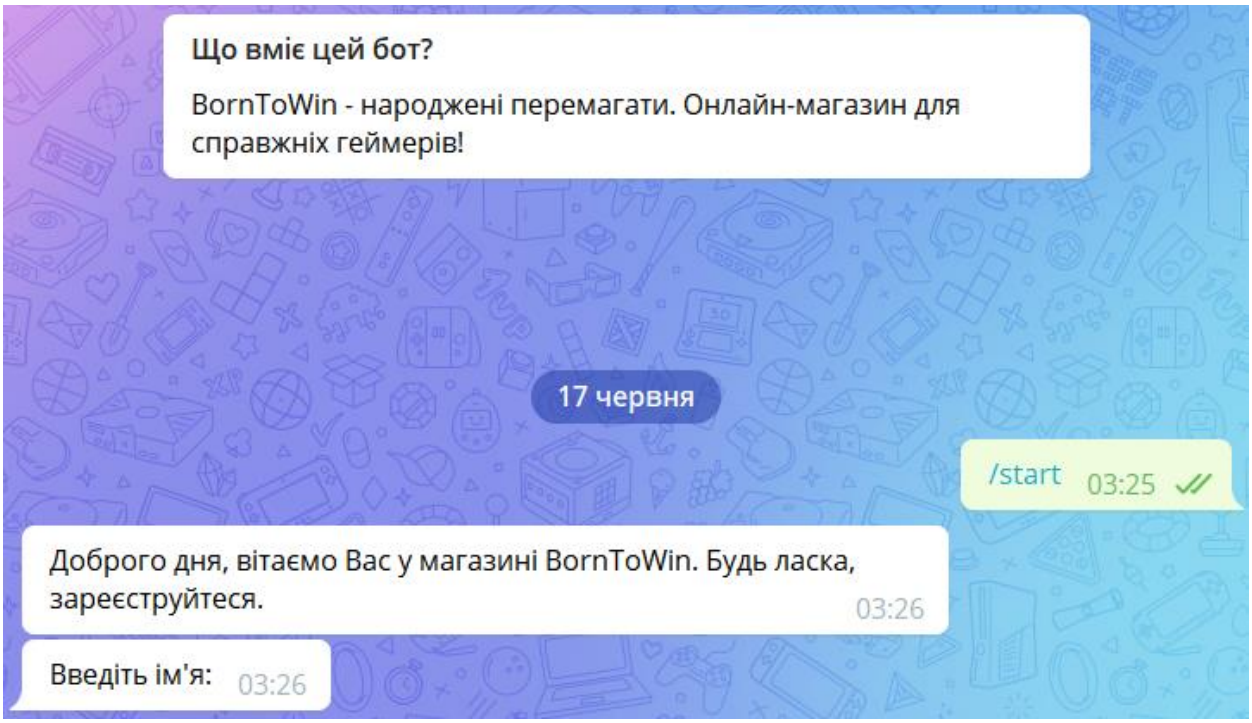

Рис. 2.25 Запуск бота з боку користувача (клієнта)

2.7.4 Опис інтерфейсу користувача

Одразу після натискання кнопки «Запустити», бот-магазин вітає користувача та надає йому можливість ввести особисті дані для реєстрації клієнта у системі. Якщо клієнт вже був зареєстрований та, відповідно, збережений у базі даних, старт бота привітає користувача, звернувшись до нього за власним ім'ям.

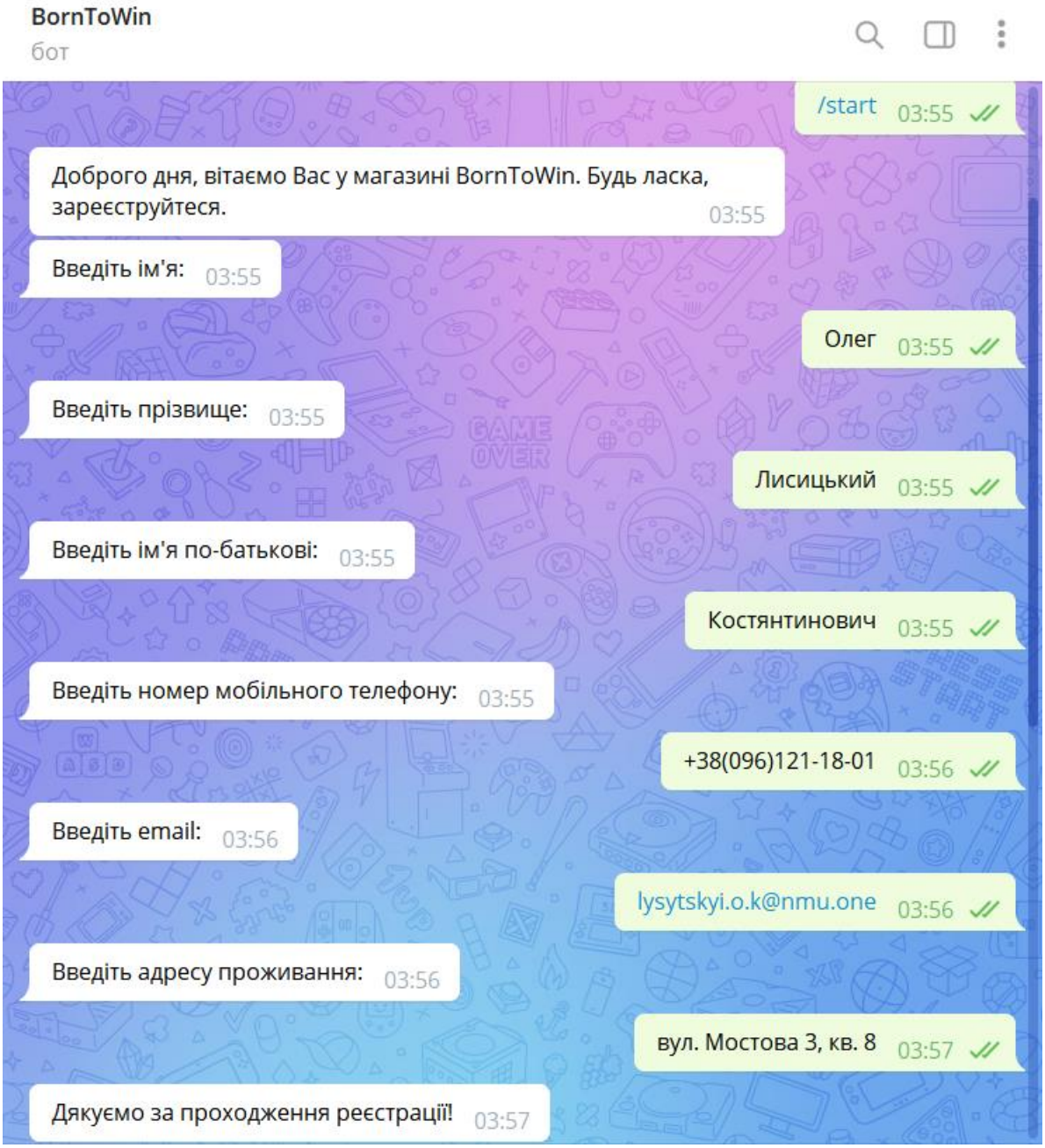

Рис. 2.26 Перший візит до бота, успішне проходження реєстрації

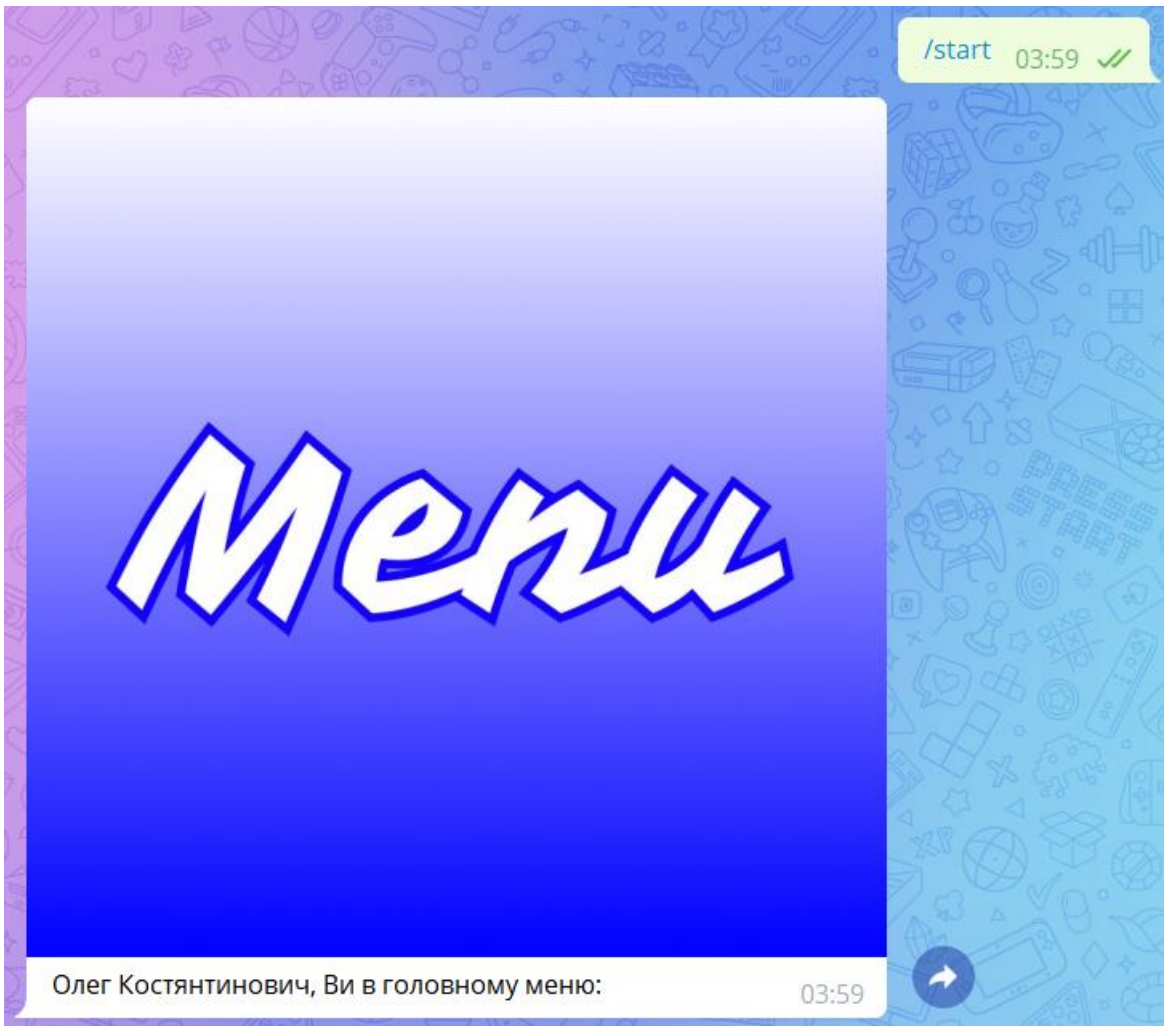

Рис. 2.27 Усі наступні візити вітають користувача без проходження повторної реєстрації

В обох випадках (не залежно від того чи реєструємося ми, чи звертаємося до бота вже не перший раз) користувачеві надається інтуїтивна та багатофункціональна клавіатура з можливостями переміщуватися по різних стадіях боту.

Рис. 2.28 Багатофункціональна Reply клавіатура

Тепер маємо змогу роздивитися увесь функціонал покроково. Почнемо з пункту «Профіль». Саме тут можна знайти інформацію щодо своєї сторінки, відредагувати дані (на вибір), або ж повернутися до головного меню.

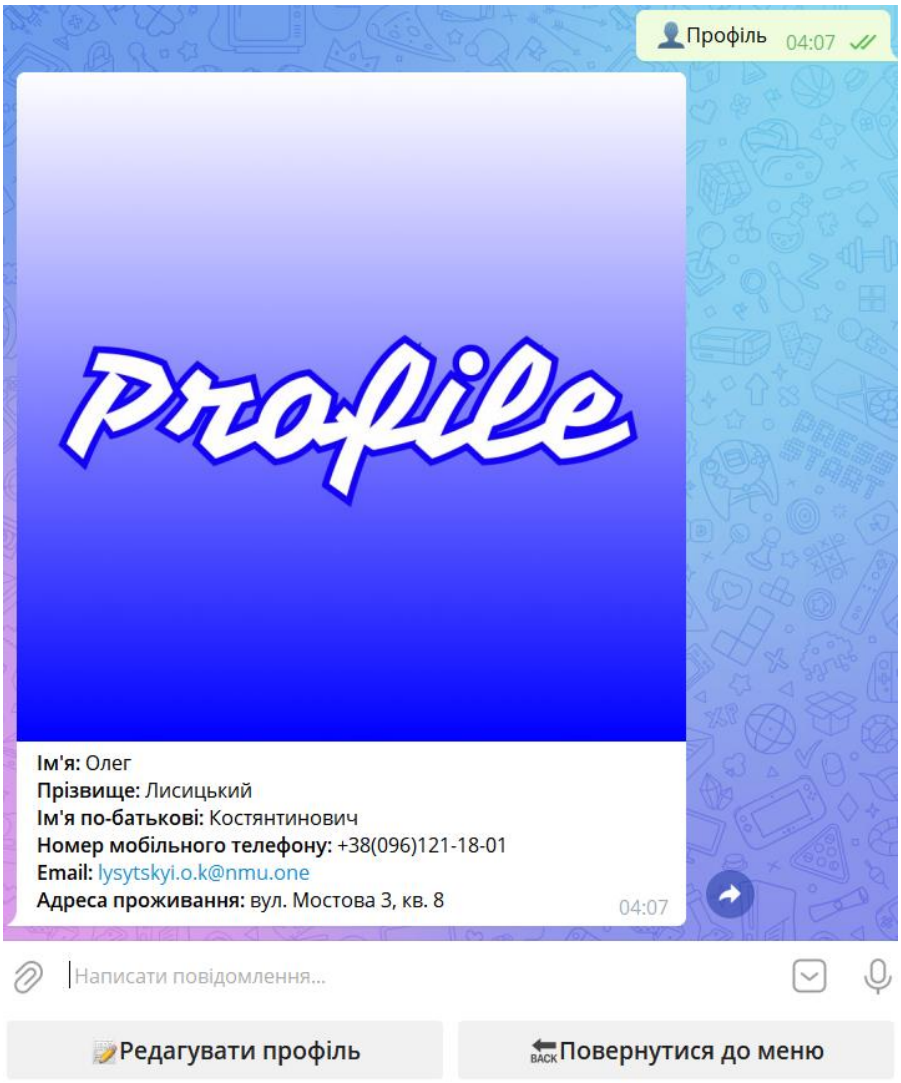

Рис. 2.29 Профіль користувача з можливістю звичайного перегляду та редагуванням

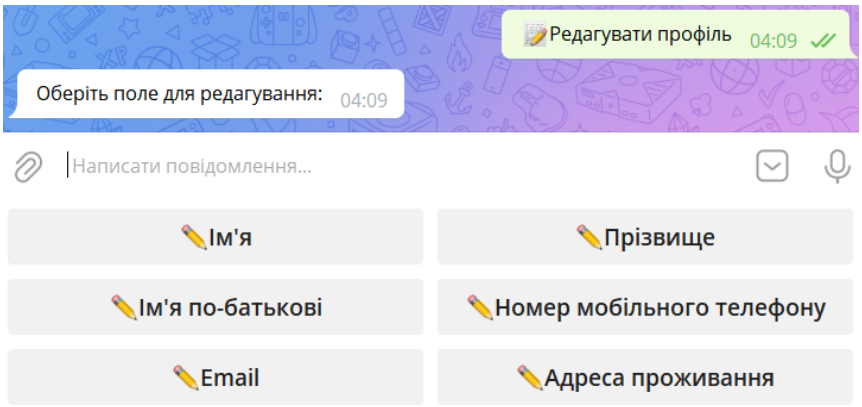

Рис. 2.30 Під час редагування користувачу надається вибір поля для редагування

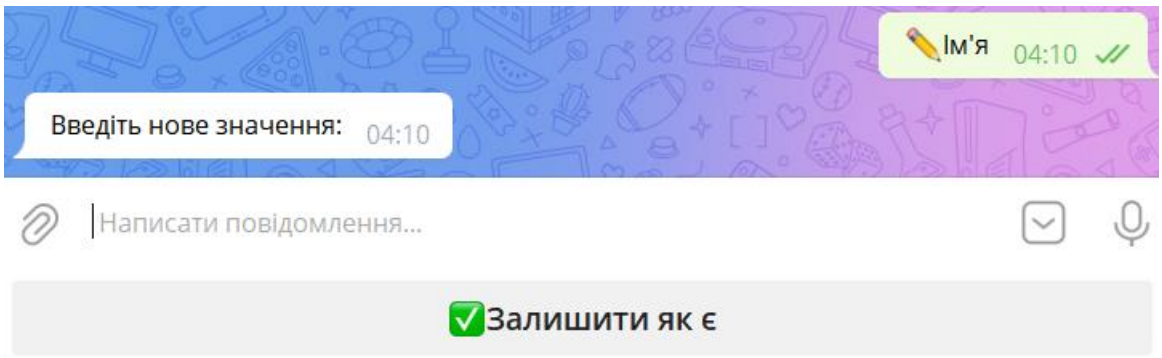

Рис. 2.31 Коли користувач редагує поле, він може ввести нове значення, або ж «залишити як є»

Після редагування повертаємося до меню, можемо переглянути інформацію про магазин. Для цього тиснемо відповідну клавішу, після чого бот надає необхідні дані. Звідси також можна повернутися до головного меню завдяки відповідній клавіатурі.

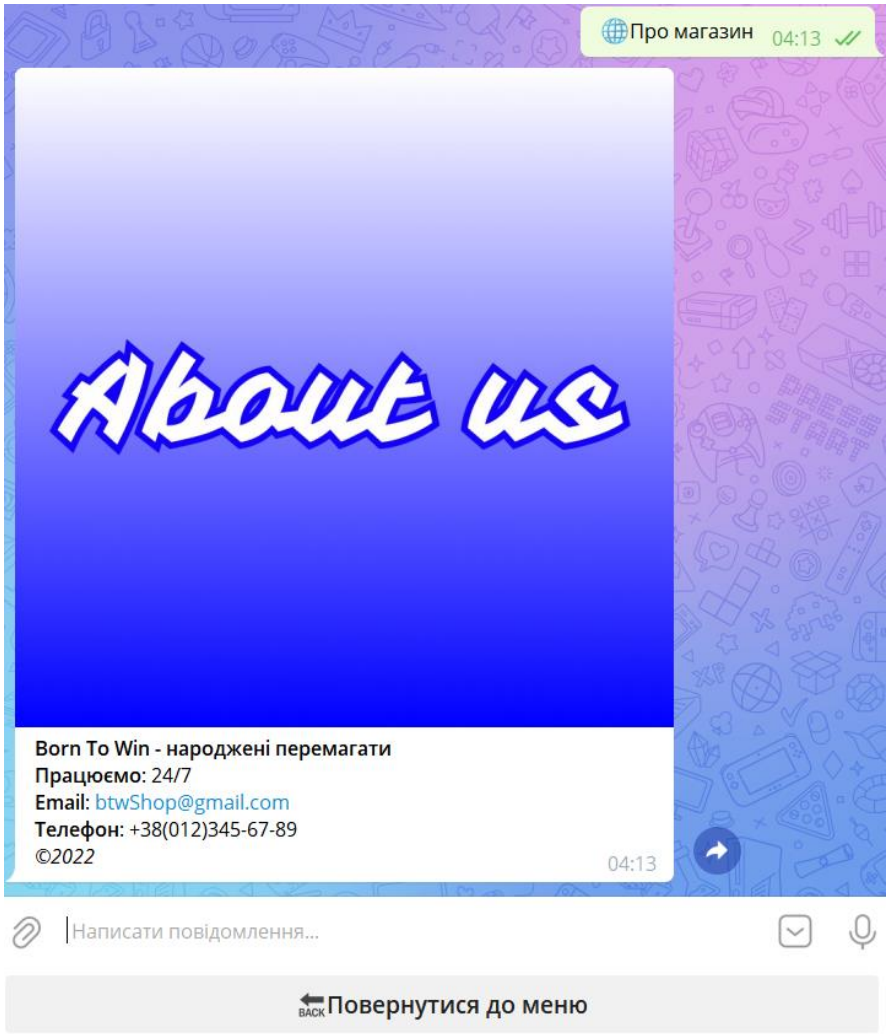

Рис. 2.32 Переглянули інформацію, можемо повертатися до меню

Якщо ж одразу відкрити користувацький кошик – бот зустріне нас повідомленням, що наразі у кошику пусто. Можемо повернутися до меню, щоб обрати товари для майбутнього замовлення.

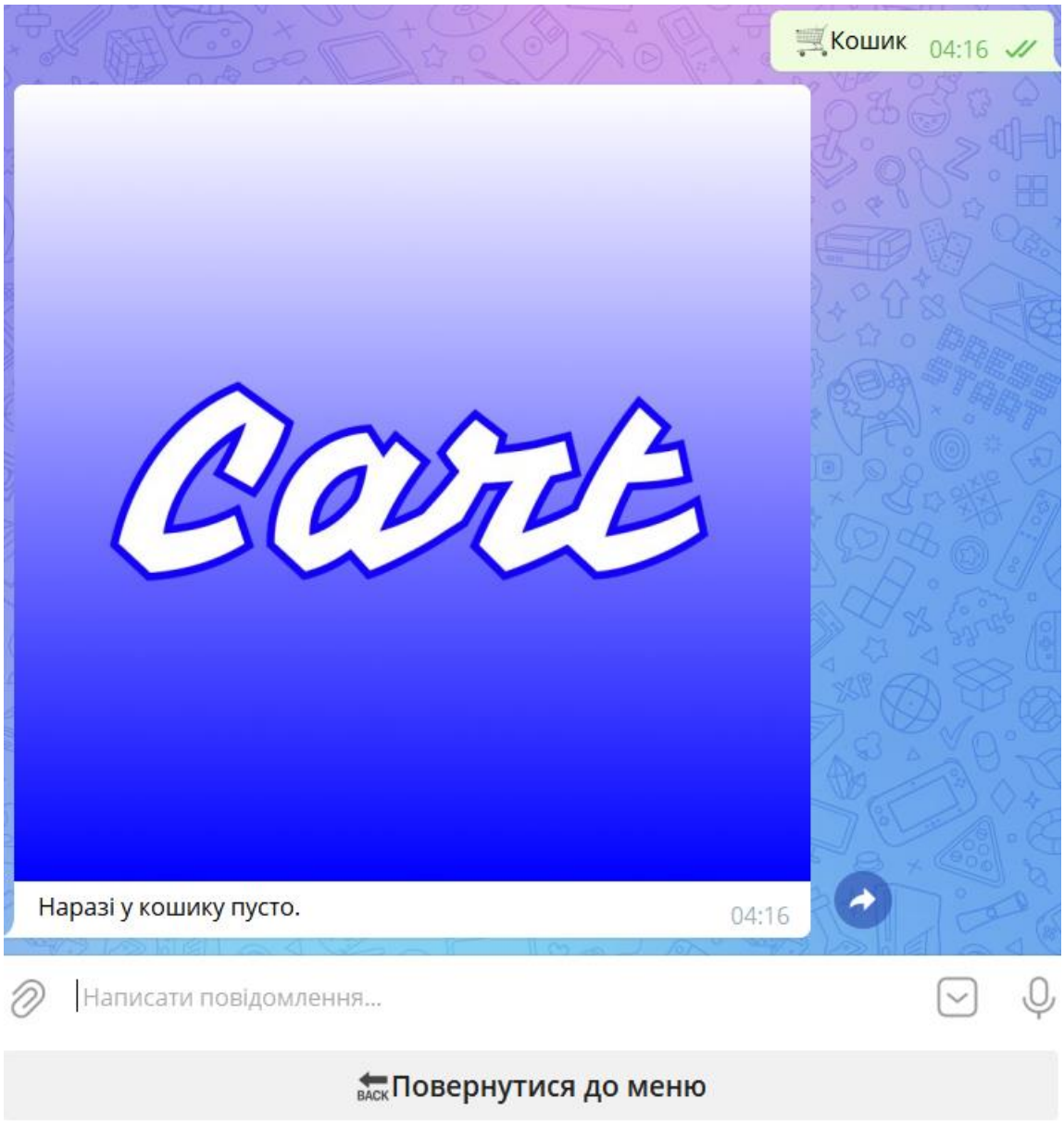

Рис. 2.33 Наразі кошик є пустим

З усіх пунктів меню залишається лише кнопка «Каталог товарів», яка вміщує у собі увесь асортимент категорій та товарів всередині. На цьому етапі користувач може обрати необхідну категорію та перейти до товару, або ж знову повернутися до головного меню.

Рис. 2.34 Можливість переглянути усі категорії товарів

Для прикладу оберемо категорію з мишами, саме там користувач зможе побачити бажані характеристики товару, переглянути зображення, що належать продукту, а також покласти товар до кошика у необхідній кількості.

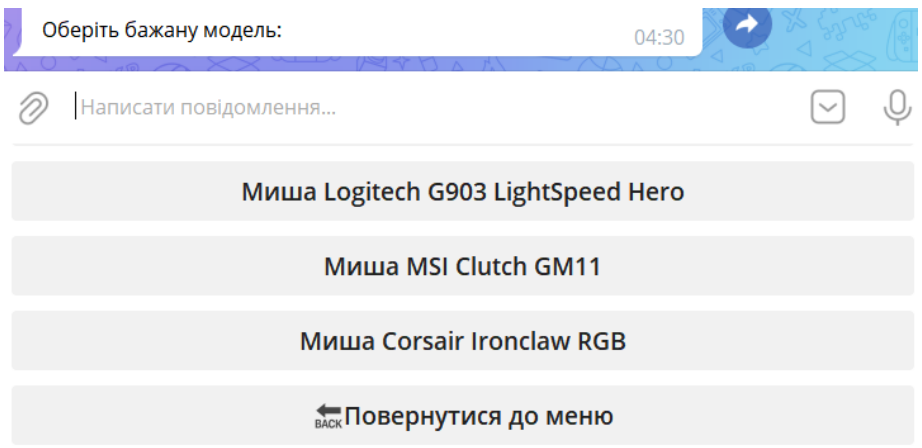

Рис. 2.35 Бачимо можливість обрати модель, або ж повернутися до меню

Оберемо ігрову мишу Logitech та побачимо повну «картку» товару.

Рис. 2.36 Миша Logitech з усіма необхідними параметрами та додатковою Inline клавіатурою

При натисканні на кнопки перегляду фото характеристики залишаються незмінними, однак з'являється інше зображення товару, що дозволяє роздивитися продукт з усіх боків. Перегляд фото є «обмеженим» у тому сенсі, що користувач не зможе відкрити неіснуюче зображення. Наприклад, якщо маємо п'ять фотографій, користувач не зможе відкрити нульове або шосте зображення.

Натискання на кнопку «Додати в кошик» надає можливість обрати бажану кількість товару та покласти продукти до кошику. Важливо наголосити, що

користувач все ще має можливість відкрити інший товар, адже Reply клавіатура все ще активна та знаходиться під полем для введення тексту.

Рис. 2.37 Той самий товар, однак з іншими додатковими зображеннями

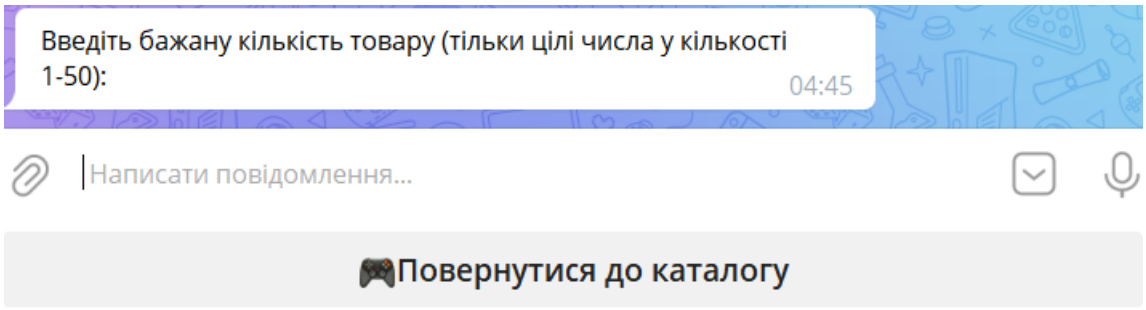

Рис. 2.38 При додаванні товару до кошика можна обрати кількість, або ж повернутися до каталогу (з відміною замовлення)

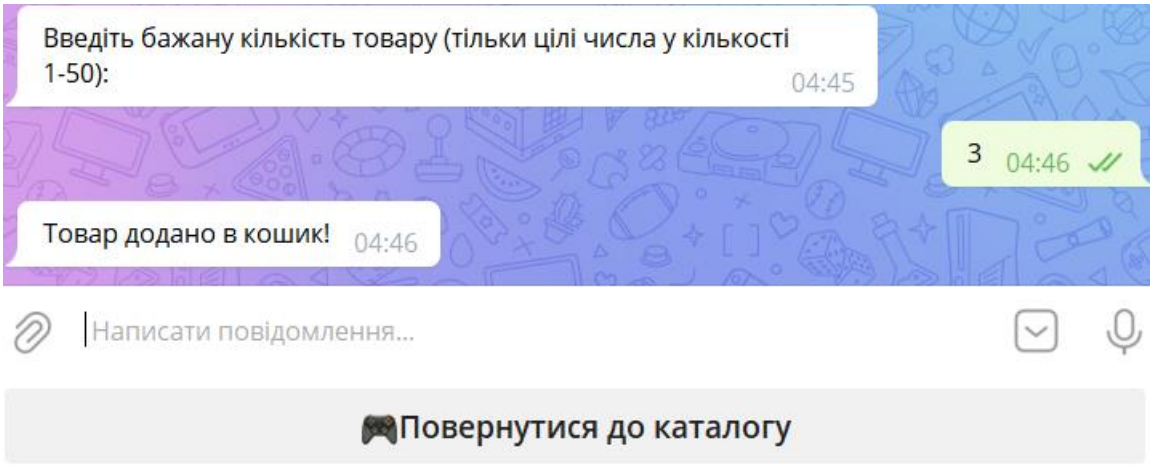

Рис. 2.39 Бот сповіщує про додавання товару, запрошує до каталогу

Після додавання товару до кошика можемо повернутися до нього, де для нас автоматично буде розрахована сума замовлення, а також надана клавіатура для майбутньої взаємодії. З цього пункту ми можемо очистити кошик, якщо користувач, наприклад, змінив рішення щодо замовлення. Можемо повернутися до меню, якщо зайшли до кошика випадково, або ж підтвердити замовлення зі збереженням даних щодо замовлення у базу даних.

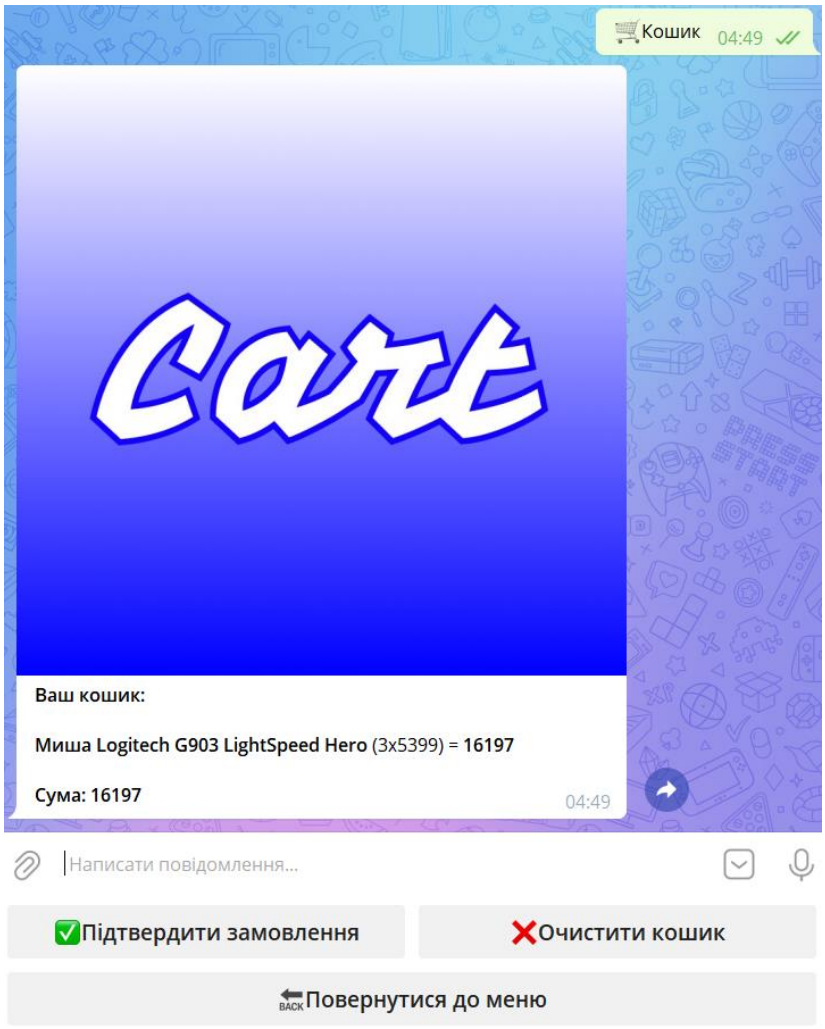

Рис. 2.40 Зовнішній вигляд кошика з доданими товарами

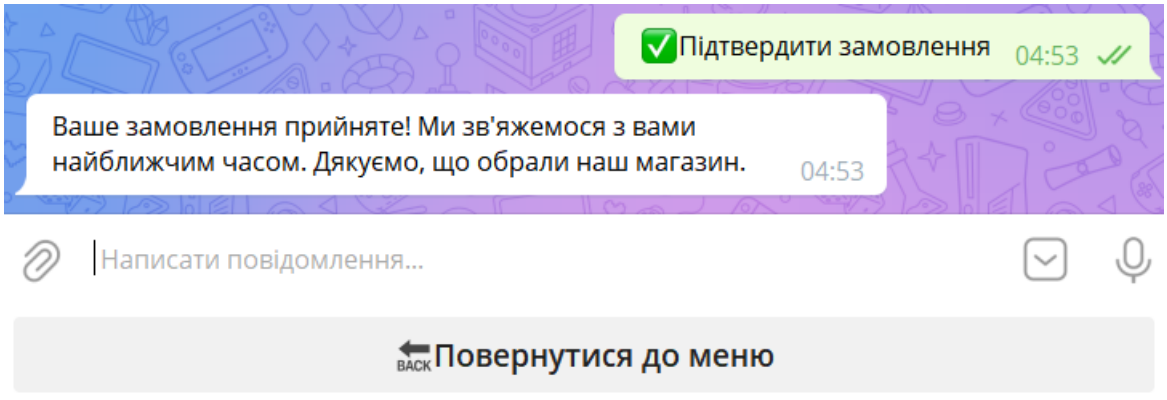

Рис 2.41 Підтвердження замовлення

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості розробки програмного забезпечення

Початкові дані:

1. передбачуване число операторів програми - 474;
2. коефіцієнт складності програми - 1,4;
3. коефіцієнт корекції програми в ході її розробки - 0,07;
4. годинна заробітна плата розробника – 146,25 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі - 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності - 0,8;
7. вартість машино-години ЕОМ - 19 грн/год.

Орієнтуючись на дані найбільшої української спільноти IT-спеціалістів Dou.ua, можна підрахувати годинну заробітну платню програміста, що пише мовою програмування Python. Середня заробітна плата Python-розробника по Україні з досвідом роботи близько одного року становить 800 доларів США на місяць. На початок червня 2022 року, НБУ оцінює один американський долар як 29,25 грн, а отже середня місячна зарплата в гривнях дорівнює 23400 грн. При восьмигодинному робочому дні (160 робочих годин) середня зарплата за годину буде становити 146,25 грн.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t=t_o+t_u+t_a+t_n+t_{отл}+t_d, \text{ людино-годин}, \quad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається за 50);

$t_и$ - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

$t_п$ - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

t_d - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q=q\cdot C\cdot(1+p), \tag{3.2}$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт кореляції програми в ході її розробки.

Розраховуючи за формулою (3.2):

$$Q=474\cdot 1,4\cdot(1+0,07)=710,05$$

Витрати праці на вивчення опису задачі $t_и$ визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u=\frac{Q\cdot B}{(75\ldots85)\cdot k}\ \text{, людино-годин,} \tag{3.3}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності;

Витрати праці на розробку алгоритму рішення задачі формулою (3.3):

$$t_u = \frac{710{,}05 \cdot 1{,}2}{80 \cdot 0{,}8} = 13{,}31 \quad \text{, людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20 \ldots 25) \cdot k} \quad \text{, людино-годин,} \qquad (3.4)$$

де Q - умовне число операторів програми;

k - коефіцієнт кваліфікації програміста.

Підставивши відповідні значення змінних у формулу (3.4):

$$t_a = \frac{710{,}05}{22 \cdot 0{,}8} = 40{,}34 \quad \text{, людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20 \ldots 25) \cdot k} \quad \text{, людино-годин,} \qquad (3.5)$$

Відповідно до формули (3.5):

$$t_n = \frac{710{,}05}{20 \cdot 0{,}8} = 44{,}37 \quad \text{, людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл}=\frac{Q}{(4\ldots5)\cdot k} \quad \text{, людино-годин,} \tag{3.6}$$

Підставивши значення до формули (3.6):

$$t_{отл}=\frac{710,05}{5\cdot0,8}=177,51 \quad \text{, людино-годин.}$$

– за умови комплексного налагодження завдання:

$$t^{k}_{отл}=1,5\cdot t_{отл} \quad \text{, людино-годин,} \tag{3.7}$$

Підставивши значення до формули (3.7):

$$t^{k}_{отл}=1,5\cdot177,51=266,27 \quad \text{, людино-годин.}$$

Витрати праці на підготовку документації:

$$t_d=t_{dp}+t_{do} \quad \text{, людино-годин,} \tag{3.8}$$

де t_{dp} - трудомісткість підготовки матеріалів і рукопису;

$$t_{dp}=\frac{Q}{(15\ldots20)\cdot k} \quad \text{, людино-годин,} \tag{3.9}$$

Використавши формулу (3.9):

$$t_{dp}=\frac{710,05}{19\cdot0,8}=46,71 \quad \text{, людино-годин.}$$

t_{do} - трудомісткість редагування, печатки й оформлення документації.

$$t_{do}=0,75\cdot t_{dp} \quad \text{, людино-годин,} \tag{3.10}$$

Відповідно до формули (3.10):

$$t_{do}=0,75\cdot 46,71=35,03 \quad \text{, людино-годин.}$$

Повертаючись до формули (3.8):

$$t_{d}=46,71+35,03=81,74 \quad \text{, людино-годин.}$$

Тоді виводимо трудомісткість розробки програмного забезпечення за формулою (3.1):

$$t=50+13,31+40,34+44,37+177,51+81,74=407,27, \text{ людино-годин.}$$

Для розробки даного програмного забезпечення в загальній складності необхідно 407,27 людино-годин.

3.2. Розрахунок витрат на створення програмного забезпечення

Витрати на створення програмного забезпечення Кпо включають у себе витрати на заробітну плату виконавця програми (Ззп) і витрати машинного часу (Змв), необхідного на налагодження програми на ЕОМ.

$$K_{ПО}=З_{ЗП}+З_{МВ} \quad \text{, грн,} \tag{3.11}$$

де $З_{ЗП}$ - заробітна плата виконавців, що визначається за формулою:

$$З_{ЗП}=t\cdot C_{ПР} \quad , грн, \tag{3.12}$$

де t - загальна трудомісткість, людино-годин;

$C_{ПР}$ - середня годинна заробітна плата програміста, грн/година

Скориставшись формулою (3.12):

$$З_{ЗП}=407,27\cdot 146,25=59563,24, грн,$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ}=t_{отл}\cdot C_{МЧ} \quad , грн, \tag{3.13}$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год.

$C_{МЧ}$ - вартість машино-години ЕОМ, грн/год.

Підставивши значення до формули (3.13):

$$З_{МВ}=177,51\cdot 19=3372,7, грн,$$

Тоді витрати на створення ПЗ знайдемо за формулою (3.11):

$$К_{ПО}=59563,24+3372,7=62935,94, грн,$$

Витрати на створення програмного забезпечення є частиною одноразових капітальних витрат.

Очікуваний період створення ПЗ:

$$Т=\frac{t}{B_k\cdot F_p} \quad , міс, \tag{3.14}$$

де B_k - число виконавців;

$F_р$ - місячний фонд робочого часу (при 40 годинному робочому тижні Fp = 160 годин).

Відповідно до формули (3.14):

$$T=\frac{407,27}{1\cdot 160}=2.55, \quad \text{міс},$$

На розробку даного програмного забезпечення необхідно 407,27 людино-годин. Тобто, ймовірна тривалість розробки для одного програміста складатиме 2,55 місяці при стандартному 8-годинному робочому дні (160-годинному робочому місяці). Очікувані витрати на створення програмного забезпечення (при курсі гривні відносно долара США, який коштує 29.25 грн на початок червня) складатимуть 62935,94 грн.

ВИСНОВКИ

Після виконання цієї кваліфікаційної роботи було розроблено застосунок телеграм бот інтернет-магазин на основі Python.

Протягом розробки використовувався сучасний підхід з можливістю подальшого розвитку інтернет-магазину з додаванням нового функціоналу та асортименту товарів.

Наразі телеграм-бот інтернет-магазин дозволяє користувачеві швидко, зручно та інтуїтивно зрозуміло замовити бажаний товар без необхідності зайвого пошуку серед мільйонів позицій електро-техніки. Продукт має сучасний дизайн та підхід до клієнта. Мінімум взаємодії та робота з ботом у будь-який час завдяки можливості ведення бесіди із «запрограмованим» консультантом.

Цей застосунок робить наступний крок уперед, дозволяючи індустрії продажів перейти на новий рівень. Мінімальна кількість персоналу, повна відсутність відділень та можливість додавання товару «на ходу» - критерії, які кожен бізнесмен має оцінити.

Програма є дуже мінімалістичною, не потребує потужного технічного обладнання. Усе, що необхідно – звичайний месенджер, яким ми користуємося кожного дня. Кібер-спортивний асортимент зібрано в одному місці, потрібно лише обрати товар та натиснути кнопку «Замовити».

Виконуючи кваліфікаційну роботу, були досягнуті наступні етапи:

- повний аналіз поставленої задачі;
- встановлення вимог та критеріїв щодо програмного застосунку;
- визначення структури продукту та підбір необхідних засобів для вирішення проблеми;
- розробка програмного коду застосунку;
- надання рекомендацій щодо взаємодії та використання застосунку.

Також був проведений економічний аналіз застосунку, завдяки якому було визначено трудомісткість, а також підраховано вартість та затрачений час.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лутц М. Вивчаємо Python. 5-е видання. 1 том. 2009 р.
2. Sumit Raj. Building Chatbots with Python. 2018 р.
3. Python Developer’s Guide. URL: https://devguide.python.org/. Дата звернення: 30.05.2022.
4. Telegram API. URL: https://core.telegram.org/api. Дата звернення: 30.05.2022.
5. PyTelegramBotAPI. URL: https://pypi.org/project/pyTelegramBotAPI/. Дата звернення: 30.05.2022.
6. Standard keyboard. URL: https://www.botpress.org/docs/telegram/reply-markup-keyboard-inline-keyboard/. Дата звернення: 02.06.2022.
7. Python decorators. URL: https://www.programiz.com/python-programming/decorator. Дата звернення: 02.06.2022.
8. JSON introduction: https://www.w3schools.com/js/js_json_intro.asp. Дата звернення: 03.06.2022.
9. Working with cmd. URL: https://expert-only.com/en/ms-dos/display-the-current-directory-path-in-cmd/. Дата звернення: 03.06.2022.
10. MongoDB Atlas. URL: https://www.mongodb.com/atlas/database. Дата звернення: 04.06.2022.
11. Generating SSH token. URL: https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent. Дата звернення: 04.06.2022.
12. What is Figma. URL: https://www.theme-junkie.com/what-is-figma/. Дата звернення: 04.06.2022.
13. Visual Studio Code: Getting started. URL: https://code.visualstudio.com/docs. Дата звернення: 05.06.2022.
14. A history of Bots. URL: https://abusix.com/resources/botnets/a-brief-history-of-bots-and-how-theyve-shaped-the-internet-today. Дата звернення: 05.06.2022.

15. Online shopping. URL: https://en.wikipedia.org/wiki/Online_shopping. Дата звернення: 05.06.2022.
16. Voice assistants history. URL: https://voicebot.ai/2017/07/14/timeline-voice-assistants-short-history-voice-revolution/. Дата звернення: 05.06.2022.
17. Comparing famous programming languages. URL: https://www.thoughtco.com/comparing-popular-programming-languages-958275. Дата звернення: 05.06.2022.
18. Tiobe index. URL: https://www.tiobe.com/tiobe-index/. Дата звернення: 05.06.2022.
19. Dou salaries. URL: https://jobs.dou.ua/salaries/?period=2021-12. Дата звернення: 07.06.2022.
20. Економіка. Розрахунок людино- та машино-годин. URL: https://docs.infor.com/ln/10.4/ru-ru/lnolh/help/ti/onlinemanual/000159.html. Дата звернення: 07.06.2022.

ДОДАТОК А

КОД ПРОГРАМИ

Лістинг main.py

```
import os

import telebot
import settings
import keyboards

bot = telebot.TeleBot(settings.token)


def check_and_buy(answer, call, profile_info):
    if answer.text.isdigit():
        if int(answer.text) > 0 and int(answer.text) < 51:
            settings.users_col.update_one({"tg_id": call.message.chat.id},
{"$addToSet":{"shopping_list":profile_info["last_viewed"] + "x" + answer.text}})
            bot.send_message(call.message.chat.id, "Товар додано в кошик!")
        else:
            input_product_quantity(answer, call, profile_info)
    elif answer.text == "🎮Повернутися до каталогу":
        bot.send_photo(call.message.chat.id, open("bot_pictures/catalog.png", "rb"), "Оберіть
бажану категорію:", reply_markup = keyboards.catalogboard)
    else:
        input_product_quantity(answer, call, profile_info)


def input_product_quantity(answer, call, profile_info):
    try:
        bot.edit_message_reply_markup(chat_id = call.message.chat.id, message_id =
call.message.id, reply_markup = None)
    except:
        pass
    answer = bot.send_message(call.message.chat.id, "Введіть бажану кількість товару (тільки
цілі числа у кількості 1-50):", reply_markup = keyboards.returncatalogboard)
    bot.register_next_step_handler(answer, check_and_buy, call, profile_info)


def find_photo_quantity(product_id):
    photo_list = os.listdir(f"images/{product_id}")
    photo_quantity = len(photo_list)
    return photo_quantity


@bot.callback_query_handler(func=lambda call:True)
def check_call(call):

    if isinstance(call, telebot.types.Message):
        profile_info = settings.users_col.find_one({"tg_id": call.chat.id})
        photo_limit = find_photo_quantity(profile_info["last_viewed"])
```

```
        try:
            bot.edit_message_reply_markup(chat_id = call.chat.id, message_id = call.id - 1,
reply_markup = None)
        except:
            pass
    else:
        profile_info = settings.users_col.find_one({"tg_id": call.message.chat.id})
        photo_limit = find_photo_quantity(profile_info["last_viewed"])

        if call.data == "next_photo":
            if profile_info["last_photo"] < photo_limit:
                media =
telebot.types.InputMediaPhoto(open(f"images/{profile_info['last_viewed']}/{profile_info['last_
viewed']}_{profile_info['last_photo'] + 1}.jpg", "rb"), caption = call.message.caption)
                bot.edit_message_media(media = media, chat_id = call.message.chat.id, message_id =
call.message.id, reply_markup = keyboards.productboard)
                settings.users_col.update_one({"tg_id": profile_info["tg_id"]},
{"$set":{'last_photo':profile_info['last_photo'] + 1}})
        if call.data == "previous_photo":
            if profile_info["last_photo"] > 1:
                media =
telebot.types.InputMediaPhoto(open(f"images/{profile_info['last_viewed']}/{profile_info['last_
viewed']}_{profile_info['last_photo'] - 1}.jpg", "rb"), caption = call.message.caption)
                bot.edit_message_media(media = media, chat_id = call.message.chat.id, message_id =
call.message.id, reply_markup = keyboards.productboard)
                settings.users_col.update_one({"tg_id": profile_info["tg_id"]},
{"$set":{'last_photo':profile_info['last_photo'] - 1}})
        if call.data == "add_to_cart":
            input_product_quantity(None, call, profile_info)


@bot.message_handler(commands = ["start"])
def start(message):

    def get_firstName(answer):
        reg_list["firstName"] = answer.text
        answer = bot.send_message(message.chat.id, "Введіть прізвище:")
        bot.register_next_step_handler(answer, get_lastName)

    def get_lastName(answer):
        reg_list["lastName"] = answer.text
        answer = bot.send_message(message.chat.id, "Введіть ім'я по-батькові:")
        bot.register_next_step_handler(answer, get_middleName)

    def get_middleName(answer):
        reg_list["middleName"] = answer.text
        answer = bot.send_message(message.chat.id, "Введіть номер мобільного телефону:")
        bot.register_next_step_handler(answer, get_phoneNumber)

    def get_phoneNumber(answer):
        reg_list["phoneNumber"] = answer.text
```

```
        answer = bot.send_message(message.chat.id, "Введіть email:")
        bot.register_next_step_handler(answer, get_email)

    def get_email(answer):
        reg_list["email"] = answer.text
        answer = bot.send_message(message.chat.id, "Введіть адресу проживання:")
        bot.register_next_step_handler(answer, get_address)

    def get_address(answer):
        reg_list["address"] = answer.text
        bot.send_message(message.chat.id, "Дякуємо за проходження реєстрації!")
        settings.users_col.insert_one(reg_list)
        show_menu(message)

    person_registered = settings.users_col.find_one({"tg_id":message.chat.id})

    if person_registered != None:
        show_menu(message)
    else:
        reg_list = {
            "tg_id" : message.chat.id,
            "firstName" : "",
            "lastName" : "",
            "middleName" : "",
            "phoneNumber" : "",
            "email" : "",
            "address" : "",
            "last_viewed": "",
            "last_photo": 1,
            "shopping_list": [],
            "order_sum": 0
        }

        bot.send_message(message.chat.id, "Доброго дня, вітаємо Вас у магазині BornToWin.
Будь ласка, зареєструйтеся.")

        answer = bot.send_message(message.chat.id, "Введіть ім'я:")
        bot.register_next_step_handler(answer, get_firstName)


def show_menu(message):
    profile_info = settings.users_col.find_one({"tg_id":message.chat.id})
    name = profile_info["firstName"]
    middle_name = profile_info["middleName"]

    bot.send_photo(message.chat.id, open("bot_pictures/menu.png", "rb"), f"{name}
{middle_name}, Ви в головному меню:", reply_markup = keyboards.menuboard)


@bot.message_handler(content_types = ["text"])
def menu(message):
```

```
    def show_profile():
        profile_info = settings.users_col.find_one({"tg_id":message.chat.id})

        firstName = "*Ім'я: *" + profile_info["firstName"]
        lastName = "*Прізвище: *" + profile_info["lastName"]
        middleName = "*Ім'я по-батькові: *" + profile_info["middleName"]
        phoneNumber = "*Номер мобільного телефону: *" + profile_info["phoneNumber"]
        email = "*Email: *" + profile_info["email"]
        address = "*Адреса проживання: *" + profile_info["address"]

        profile_text = "\n".join([firstName, lastName, middleName, phoneNumber, email,
address])
        bot.send_photo(message.chat.id, open("bot_pictures/profile.png", "rb"), profile_text,
parse_mode = 'Markdown', reply_markup = keyboards.profileboard)

    if message.text == "👤Профіль":
        show_profile()

    if message.text == "📝Редагувати профіль":
        bot.send_message(message.chat.id, "Оберіть поле для редагування:", reply_markup =
keyboards.editprofileboard)

    def profile_edit(answer, field):
        if answer.text != "✅Залишити як є":
            if field == "➖Ім'я":
                field = "firstName"
            elif field == "➖Прізвище":
                field = "lastName"
            elif field == "➖Ім'я по-батькові":
                field = "middleName"
            elif field == "➖Номер мобільного телефону":
                field = "phoneNumber"
            elif field == "➖Email":
                field = "email"
            elif field == "➖Адреса проживання":
                field = "address"
            settings.users_col.update_one({"tg_id": message.chat.id}, {"$set":{field: answer.text}})
        show_profile()

    if message.text in ["➖Ім'я", "➖Прізвище", "➖Ім'я по-батькові", "➖Номер мобільного
телефону", "➖Email", "➖Адреса проживання"]:
        answer = bot.send_message(message.chat.id, "Введіть нове значення:", reply_markup =
keyboards.noeditboard)
        bot.register_next_step_handler(answer, profile_edit, message.text)

    if message.text == "🎮Каталог товарів":
        bot.send_photo(message.chat.id, open("bot_pictures/catalog.png", "rb"), "Оберіть бажану
категорію:", reply_markup = keyboards.catalogboard)

    if message.text == "🖱Миші":
```

```
    bot.send_photo(message.chat.id, open("bot_pictures/mice.png", "rb"), "Оберіть бажану модель:", reply_markup = keyboards.mouseboard)

  if message.text.startswith("Миша"):
    mouse_info = settings.mice_col.find_one({"name": message.text})

    settings.users_col.update_one({"tg_id": message.chat.id}, {"$set":{"last_viewed": mouse_info["id"], "last_photo": 1}})

    mouseId = "ID: " + mouse_info["id"]
    mouseName = "Назва: " + mouse_info["name"]
    mouseSensor = "Сенсор: " + mouse_info["sensor"]
    mouseDpi = "Чутливість (dpi): " + mouse_info["dpi"]
    mouseInterface = "Інтерфейс: " + mouse_info["interface"]
    mousePower = "Живлення: " + mouse_info["power"]
    mouseAcceleration = "Прискорення (G): " + mouse_info["acceleration"]
    mouseWeight = "Вага (г): " + mouse_info["weight"]
    mouseColor = "Колір: " + mouse_info["color"]
    mousePrice = "Ціна (грн): " + str(mouse_info["price"])

    spec_list = [mouseId, mouseName, mouseSensor, mouseDpi, mouseInterface, mousePower, mouseAcceleration, mouseWeight, mouseColor, mousePrice]

    mouse_text = "\n".join(spec_list)
    bot.send_photo(message.chat.id, open(f"images/{mouse_info['id']}/{mouse_info['id']}_1.jpg", "rb"), mouse_text, reply_markup = keyboards.productboard)
    check_call(message)

  if message.text == "⌨Клавіатури":
    bot.send_photo(message.chat.id, open("bot_pictures/keyboards.png", "rb"), "Оберіть бажану модель:", reply_markup = keyboards.keyboardboard)

  if message.text.startswith("Клавіатура"):
    keyboard_info = settings.keyboards_col.find_one({"name": message.text})

    settings.users_col.update_one({"tg_id": message.chat.id}, {"$set":{"last_viewed": keyboard_info["id"], "last_photo": 1}})

    keyboardId = "ID: " + keyboard_info["id"]
    keyboardName = "Назва: " + keyboard_info["name"]
    keyboardLanguage = "Мова: " + keyboard_info["language"]
    keyboardInterface = "Інтерфейс: " + keyboard_info["interface"]
    keyboardMechanism = "Механізм клавіш: " + keyboard_info["mechanism"]
    keyboardKeytype = "Тип перемикачів: " + keyboard_info["key_type"]
    keyboardPower = "Живлення: " + keyboard_info["power"]
    keyboardMaterial = "Матеріал: " + keyboard_info["material"]
    keyboardWeight = "Вага (г): " + keyboard_info["weight"]
    keyboardColor = "Колір: " + keyboard_info["color"]
    keyboardPrice = "Ціна (грн): " + str(keyboard_info["price"])
```

```
        spec_list = [keyboardId, keyboardName, keyboardLanguage, keyboardInterface,
keyboardMechanism, keyboardKeytype, keyboardPower, keyboardMaterial, keyboardWeight,
keyboardColor, keyboardPrice]

        keyboard_text = "\n".join(spec_list)
        bot.send_photo(message.chat.id,
open(f"images/{keyboard_info['id']}/{keyboard_info['id']}_1.jpg", "rb"), keyboard_text,
reply_markup = keyboards.productboard)
        check_call(message)

    if message.text == "🎧Навушники":
        bot.send_photo(message.chat.id, open("bot_pictures/headphones.png", "rb"), "Оберіть
бажану модель:", reply_markup = keyboards.headphonesboard)

    if message.text.startswith("Навушники"):
        headphones_info = settings.headphones_col.find_one({"name": message.text})

        settings.users_col.update_one({"tg_id": message.chat.id}, {"$set":{"last_viewed":
headphones_info["id"], "last_photo": 1}})

        headphonesId = "ID: " + headphones_info["id"]
        headphonesName = "Назва: " + headphones_info["name"]
        headphonesMicrophone = "Мікрофон: " + headphones_info["microphone"]
        headphonesRemote = "Керування: " + headphones_info["remote"]
        headphonesInterface = "Інтерфейс: " + headphones_info["interface"]
        headphonesMaterial = "Матеріал: " + headphones_info["material"]
        headphonesWeight = "Вага: " + headphones_info["weight"]
        headphonesColor = "Колір: " + headphones_info["color"]
        headphonesPrice = "Ціна (грн): " + str(headphones_info["price"])

        spec_list = [headphonesId, headphonesName, headphonesMicrophone,
headphonesRemote, headphonesInterface, headphonesMaterial, headphonesWeight,
headphonesColor, headphonesPrice]

        headphones_text = "\n".join(spec_list)
        bot.send_photo(message.chat.id,
open(f"images/{headphones_info['id']}/{headphones_info['id']}_1.jpg", "rb"), headphones_text,
reply_markup = keyboards.productboard)
        check_call(message)

    if message.text == "🔛Акустика":
        bot.send_photo(message.chat.id, open("bot_pictures/speakers.png", "rb"), "Оберіть
бажану модель:", reply_markup = keyboards.speakerboard)

    if message.text.startswith("Акустична система"):
        speaker_info = settings.speakers_col.find_one({"name": message.text})

        settings.users_col.update_one({"tg_id": message.chat.id}, {"$set":{"last_viewed":
speaker_info["id"], "last_photo": 1}})

        speakerId = "ID: " + speaker_info["id"]
        speakerName = "Назва: " + speaker_info["name"]
```

```
        speakerFormat = "Формат: " + speaker_info["format"]
        speakerRms = "Потужність (RMS): " + speaker_info["rms"]
        speakerRemote = "Керування: " + speaker_info["remote"]
        speakerInterface = "Інтерфейс: " + speaker_info["interface"]
        speakerMaterial = "Матеріал: " + speaker_info["material"]
        speakerPower = "Живлення: " + speaker_info["power"]
        speakerWeight = "Вага (кг): " + speaker_info["weight"]
        speakerColor = "Колір: " + speaker_info["color"]
        speakerPrice = "Ціна (грн): " + str(speaker_info["price"])

        spec_list = [speakerId, speakerName, speakerFormat, speakerRms, speakerRemote,
speakerInterface, speakerMaterial, speakerPower, speakerWeight, speakerColor, speakerPrice]

        speaker_text = "\n".join(spec_list)
        bot.send_photo(message.chat.id,
open(f"images/{speaker_info['id']}/{speaker_info['id']}_1.jpg", "rb"), speaker_text,
reply_markup = keyboards.productboard)
        check_call(message)

    if message.text == "■Поверхні (килими)":
        bot.send_photo(message.chat.id, open("bot_pictures/mousepads.png", "rb"), "Оберіть
бажану модель:", reply_markup = keyboards.mousepadboard)

    if message.text.startswith("Поверхня"):
        mousepad_info = settings.mousepads_col.find_one({"name": message.text})

        settings.users_col.update_one({"tg_id": message.chat.id}, {"$set":{"last_viewed":
mousepad_info["id"], "last_photo": 1}})

        mousepadId = "ID: " + mousepad_info["id"]
        mousepadName = "Назва: " + mousepad_info["name"]
        mousepadSize = "Розмір (мм): " + mousepad_info["size"]
        mousepadMaterial = "Матеріал: " + mousepad_info["material"]
        mousepadColor = "Колір: " + mousepad_info["color"]
        mousepadPrice = "Ціна (грн): " + str(mousepad_info["price"])

        spec_list = [mousepadId, mousepadName, mousepadSize, mousepadMaterial,
mousepadColor, mousepadPrice]

        mousepad_text = "\n".join(spec_list)
        bot.send_photo(message.chat.id,
open(f"images/{mousepad_info['id']}/{mousepad_info['id']}_1.jpg", "rb"), mousepad_text,
reply_markup = keyboards.productboard)
        check_call(message)

    if message.text == "🛒Кошик":
        person_info = settings.users_col.find_one({"tg_id":message.chat.id})
        catalog_list = [settings.mice_col, settings.keyboards_col, settings.headphones_col,
settings.speakers_col, settings.mousepads_col]
        cart_list = []
        for position in person_info["shopping_list"]:
            product_list = []
```

```
            product_and_quantity = position.split("x")
            for category in catalog_list:
                result = category.find_one({"id":product_and_quantity[0]})
                if result != None:
                    product_list.append(result["name"])
                    product_list.append(product_and_quantity[1])
                    product_list.append(result["price"])
                    product_list.append(int(product_and_quantity[1])*result["price"])
            cart_list.append(product_list)

        total_sum = 0
        cart_text = "*Ваш кошик:*\n\n"
        for position in cart_list:
            total_sum += position[3]
            cart_text += f"*{position[0]}* ({position[1]}x{position[2]}) = *{position[3]}*\n"

        if total_sum != 0:
            cart_text += f"\n*Сума: {total_sum}*"
            settings.users_col.update_one({"tg_id": message.chat.id}, {"$set":{"order_sum":
total_sum}})
            bot.send_photo(message.chat.id, open("bot_pictures/cart.png", "rb"), cart_text,
parse_mode = 'Markdown', reply_markup = keyboards.cartboard)
        else:
            cart_text = "Наразі у кошику пусто."
            bot.send_photo(message.chat.id, open("bot_pictures/cart.png", "rb"), cart_text,
parse_mode = 'Markdown', reply_markup = keyboards.fromcartboard)

    if message.text == "✅Підтвердити замовлення":
        person_info = settings.users_col.find_one({"tg_id":message.chat.id})
        order_dict = {
            "fullName": f"{person_info['lastName']} {person_info['firstName']}
{person_info['middleName']}",
            "phone_number": person_info['phoneNumber'],
            "email": person_info['email'],
            "address": person_info['address'],
            "shopping_list": person_info['shopping_list'],
            "order_sum": person_info['order_sum']
        }

        settings.orders_col.insert_one(order_dict)
        settings.users_col.update_one({"tg_id": message.chat.id}, {"$set":{"shopping_list": [],
"order_sum": 0}})
        bot.send_message(message.chat.id, "Ваше замовлення прийняте! Ми зв'яжемося з вами
найближчим часом. Дякуємо, що обрали наш магазин.", reply_markup =
keyboards.fromcartboard)

    if message.text == "❌Очистити кошик":
        settings.users_col.update_one({"tg_id": message.chat.id}, {"$set":{"shopping_list": [],
"order_sum": 0}})
        bot.send_message(message.chat.id, "Кошик очищено!", reply_markup =
keyboards.fromcartboard)
```

```
    if message.text == "🌐Про магазин":

        shopSlogan = "*Born To Win - народжені перемагати*"
        shopHours = "*Працюємо*: 24/7"
        shopEmail = "*Email*: btwShop@gmail.com"
        shopPhone = "*Телефон*: +38(012)345-67-89"
        shopCopyright = "_©2022_"

        shop_text = "\n".join([shopSlogan, shopHours, shopEmail, shopPhone, shopCopyright])
        bot.send_photo(message.chat.id, open("bot_pictures/about.png", "rb"), shop_text,
parse_mode = 'Markdown', reply_markup = keyboards.aboutshopboard)

    if message.text == "🎮Повернутися до каталогу":
        bot.send_photo(message.chat.id, open("bot_pictures/catalog.png", "rb"), "Оберіть бажану
категорію:", reply_markup = keyboards.catalogboard)

    if message.text == "🔙Повернутися до меню":
        show_menu(message)

bot.polling(True)
```

Лістинг keyboards.py

```
import telebot
import settings

returnmenu_btn = telebot.types.KeyboardButton("🔙Повернутися до меню")

menuboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 2)
menu_btn1 = telebot.types.KeyboardButton("👤Профіль")
menu_btn2 = telebot.types.KeyboardButton("🎮Каталог товарів")
menu_btn3 = telebot.types.KeyboardButton("🛒Кошик")
menu_btn4 = telebot.types.KeyboardButton("🌐Про магазин")
menuboard.add(menu_btn1, menu_btn2, menu_btn3, menu_btn4)

profileboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 2)
profile_btn1 = telebot.types.KeyboardButton("📝Редагувати профіль")
profileboard.add(profile_btn1, returnmenu_btn)

editprofileboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 2)
editprofile_btn1 = telebot.types.KeyboardButton("➖Ім'я")
editprofile_btn2 = telebot.types.KeyboardButton("➖Прізвище")
editprofile_btn3 = telebot.types.KeyboardButton("➖Ім'я по-батькові")
editprofile_btn4 = telebot.types.KeyboardButton("➖Номер мобільного телефону")
editprofile_btn5 = telebot.types.KeyboardButton("➖Email")
editprofile_btn6 = telebot.types.KeyboardButton("➖Адреса проживання")
editprofileboard.add(editprofile_btn1, editprofile_btn2, editprofile_btn3, editprofile_btn4,
editprofile_btn5, editprofile_btn6)
```

```
noeditboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 1)
noedit_btn1 = telebot.types.KeyboardButton("✅Залишити як є")
noeditboard.add(noedit_btn1)

catalogboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 2)
catalog_btn1 = telebot.types.KeyboardButton("🖱Миші")
catalog_btn2 = telebot.types.KeyboardButton("⌨Клавіатури")
catalog_btn3 = telebot.types.KeyboardButton("🎧Навушники")
catalog_btn4 = telebot.types.KeyboardButton("📻Акустика")
catalog_btn5 = telebot.types.KeyboardButton("■Поверхні (килими)")
catalogboard.add(catalog_btn1, catalog_btn2, catalog_btn3, catalog_btn4, catalog_btn5,
returnmenu_btn)

mouseboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 1)
all_mice = settings.mice_col.find()
for mouse in all_mice:
   mouse_btn = telebot.types.KeyboardButton(mouse["name"])
   mouseboard.add(mouse_btn)
mouseboard.add(returnmenu_btn)

keyboardboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 1)
all_keyboards = settings.keyboards_col.find()
for keyboard in all_keyboards:
   keyboard_btn = telebot.types.KeyboardButton(keyboard["name"])
   keyboardboard.add(keyboard_btn)
keyboardboard.add(returnmenu_btn)

headphonesboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 1)
all_headphones = settings.headphones_col.find()
for headphones in all_headphones:
   headphones_btn = telebot.types.KeyboardButton(headphones["name"])
   headphonesboard.add(headphones_btn)
headphonesboard.add(returnmenu_btn)

speakerboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 1)
all_speakers = settings.speakers_col.find()
for speaker in all_speakers:
   speaker_btn = telebot.types.KeyboardButton(speaker["name"])
   speakerboard.add(speaker_btn)
speakerboard.add(returnmenu_btn)

mousepadboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 1)
all_mousepads = settings.mousepads_col.find()
for mousepad in all_mousepads:
   mousepad_btn = telebot.types.KeyboardButton(mousepad["name"])
   mousepadboard.add(mousepad_btn)
mousepadboard.add(returnmenu_btn)

productboard = telebot.types.InlineKeyboardMarkup(row_width = 2)
product_btn1 = telebot.types.InlineKeyboardButton("➡Наступне фото", callback_data =
"next_photo")
```

```
product_btn2 = telebot.types.InlineKeyboardButton("⬅Попереднє фото", callback_data =
"previous_photo")
product_btn3 = telebot.types.InlineKeyboardButton("🛒Додати в кошик", callback_data =
"add_to_cart")
productboard.add(product_btn1, product_btn2, product_btn3)

returncatalogboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 1)
returncatalog_btn = telebot.types.KeyboardButton("🎮Повернутися до каталогу")
returncatalogboard.add(returncatalog_btn)

cartboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 2)
cart_btn1 = telebot.types.KeyboardButton("✅Підтвердити замовлення")
cart_btn2 = telebot.types.KeyboardButton("❌Очистити кошик")
cartboard.add(cart_btn1, cart_btn2, returnmenu_btn)

fromcartboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 1)
fromcartboard.add(returnmenu_btn)

aboutshopboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard = True, row_width = 1)
aboutshopboard.add(returnmenu_btn)
```

Лістинг settings.py

```
from pymongo import MongoClient

client = MongoClient("localhost:27017")
db = client["BornToWin_shop"]

users_col = db["Users"]
mice_col = db["Mice"]
keyboards_col = db["Keyboards"]
headphones_col = db["Headphones"]
speakers_col = db["Speakers"]
mousepads_col = db["Mousepads"]
orders_col = db["Orders"]

token = "5556088165:AAHSVkprhiy5tq8HfWju9kS_4XnOug3PYvk"
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота 121-18-1 Лисицький О.К..doc	Пояснювальна записка до кваліфікаційної роботи в форматі Word.
Кваліфікаційна робота 121-18-1 Лисицький О.К..pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
121-18-1 Лисицький О.К..rar	Архів. Містить коди програми, json-базу даних і необхідні зображення.
Презентація	
Презентація 121-18-1 Лисицький О.К..ppt	Презентація кваліфікаційної роботи.