Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Проектування та розробка вебплатформи для автоматизованого створення телеграм-ботів за допомогою інструментів конструювання сценаріїв»

Виконав: студент IV курсу
групи 2ПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Брильянт І. А.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Чехместрук Р.Ю.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ОТ Войцеховська О.В.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ Романюк О.Н.
« 10 » червня 2024 р.

ВНТУ– 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 12 » березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Брильянту Івану Андрійовичу

1. Тема роботи – проектування та розробка вебплатформи для автоматизованого створення телеграм-ботів за допомогою інструментів конструювання сценаріїв.

Керівник роботи: Чехместрук Роман Юрійович, к.т.н. доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: середовище розробки sublime text, система управління базами даних – SQLite; мова запитів – SQL; інструменти для розробки веб-інтерфейсу: HTML, CSS, JavaScript; мова програмування: Python, бібліотеки: requests, flask, aiogram.

4. Зміст текстової частини: аналіз розвитку технологій створення телеграм-ботів та постановка задач дослідження, включаючи огляд сучасного стану технологій, порівняльний аналіз існуючих веб-платформ, дослідження методів програмування та інструментів для створення ботів, визначення цілей та завдань для розробки веб-платформи; розробка структури та алгоритмів програмного продукту, включаючи аналіз вимог та вхідних даних для системи, проектування користувацького інтерфейсу, моделі системи, алгоритмів роботи системи; розробка програмних засобів веб-платформи для створення телеграм-ботів з використанням конструктора сценаріїв та інтеграція готових шаблонів, включаючи варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення, розробку функцій перевірки токену та додання його до бази даних, перезапуску всіх ботів з оновленою базою даних, розробку інтерфейсу веб-платформи.

5. Перелік ілюстративного матеріалу: актуальність; мета, об'єкт та предмет дослідження; задачі дослідження; практичне значення; порівняльний аналіз

блок-схеми алгоритмів роботи веб-платформи, тестування веб-платформи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Чехместрук Р. Ю., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024р.

КАЛЕНДАРНИЙ ПЛАН

Ч.ч.	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання і вибір методу вирішення поставленої задачі дослідження	14.03.24 - 29.03.24	Вик.
2	Вибір архітектури програмного компоненту	01.04.24 - 09.04.24	Вик.
3	Розробка структури інтерфейсу	10.04.24 - 19.04.24	Вик.
4	Вибір середовища та мови розробки	20.04.24 - 24.04.24	Вик.
5	Розробка модулів програми	27.04.24 - 11.05.24	Вик.
6	Тестування програми	11.05.24 - 15.05.24	Вик.
7	Оформлення матеріалів до захисту БДР	16.05.24 - 30.05.24	Вик.

Студент ________ **І. А. Брильянт**
(підпис) (ініціали та прізвище)

Керівник бакалаврської кваліфікаційної роботи ________ **Р. Ю. Чехместрук**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

УДК 004.912.032.26

Брильянт І.А. Розробка вебплатформи для автоматизованого створення телеграм-ботів за допомогою інструменту конструювання сценаріїв: бакалаврська дипломна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 59 с.

На укр. мові. Бібліогр. : 25 назв ; рис. : 39 ; табл. 2.

У бакалаврській дипломній роботі здійснено всебічний аналіз існуючих технологій та методів створення телеграм-ботів, визначено мету дослідження – спрощення процесу створення телеграм-ботів за допомогою вебплатформи з конструктором сценаріїв.

Розроблено алгоритми та програмні засоби для вебплатформи, що дозволяє користувачам без навичок програмування легко створювати та налаштовувати телеграм-ботів, що може бути застосовано у різних сферах діяльності для автоматизації бізнес-процесів та ефективного спілкування. Запропоновано підхід до створення телеграм-ботів з використанням бібліотеки aiogram3 для Python, яка оптимізує асинхронність та взаємодію з Telegram API.

Отримані в бакалаврській дипломній роботі результати можуть бути використані для побудови інтуїтивно зрозумілих та легко налаштовуваних систем створення ботів.

Ключові слова: веб-платформа, телеграм-бот, конструктор сценаріїв, автоматизація.

ABSTRACT

Bryliant I.A. Development of a web platform for automated creation of Telegram bots using a scripting tool. Vinnytsia: VNTU, 2024. 59 p.

In Ukrainian language. Bibliography: 25 titles; fig. : 39; table 2.

In the bachelor's thesis, a comprehensive analysis of existing technologies and methods of creating Telegram bots was carried out, the purpose of the research was determined - to simplify the process of creating Telegram bots using a web platform with a script designer.

Algorithms and software tools have been developed for the web platform, which allows users without programming skills to easily create and configure Telegram bots, which can be applied in various fields of activity to automate business processes and effective communication. An approach to creating Telegram bots using the aiogram3 library for Python is proposed, which optimizes asynchrony and interaction with the Telegram API.

The results obtained in the bachelor thesis can be used to build intuitive and easily configurable bot creation systems.

Keywords: web platform, Telegram bot, script designer, automation.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сучасний етап розвитку цифрових комунікацій характеризується активним впровадженням автоматизованих систем взаємодії, зокрема телеграм-ботів. Вони стають невід’ємною частиною бізнес-процесів, забезпечуючи швидке та ефективне спілкування з клієнтами. Проте, створення та налаштування телеграм-ботів часто вимагає спеціалізованих знань у галузі програмування, що обмежує доступність цього інструменту для широкого кола користувачів [1]. Вебплатформи для автоматизованого створення телеграм-ботів відкривають нові можливості для підприємств та індивідуальних користувачів, дозволяючи їм легко створювати ботів без необхідності глибоких технічних знань. Це сприяє оптимізації робочих процесів та підвищенню ефективності взаємодії з клієнтами [2].

Актуальність. У сучасному етапі розвитку цифрових комунікацій високу ефективність взаємодії з користувачами забезпечують методи та засоби, що оптимізують використання обчислювальних ресурсів для створення інтуїтивно зрозумілих інтерфейсів [3]. Одним із таких інструментів є конструктори сценаріїв для телеграм-ботів, які дозволяють врахувати потреби користувачів без ускладнення процесу розробки [3]. Процедури створення та налаштування ботів успішно застосовують для автоматизації комунікацій, управління запитами клієнтів, маркетингових кампаній, забезпечуючи ефективне та швидке спілкування. Проте, процедурне створення телеграм-ботів часто обмежується стандартними шаблонами та не дозволяє глибокої кастомізації, що може не задовольняти всі вимоги користувачів [3]. Використання веб-платформ з конструкторами сценаріїв дозволяє користувачам без технічного досвіду створювати унікальні та функціонально багаті боти, адаптовані до специфічних потреб бізнесу та особистих запитів.

Доступність - розробка інтегрованої вебплатформи для створення телеграм-ботів дозволить забезпечити швидкий доступ до необхідної інформації та ресурсів.

Простота - платформа буде розроблена таким чином, щоб бути інтуїтивно зрозумілою та простою у використанні для всіх користувачів, незалежно від їхнього технічного рівня.

Інформативність - платформа буде надавати корисну та актуальну інформацію, яка допоможе користувачам у процесі створення та налаштування ботів.

Зручність - завдяки інтеграції різних модулів, платформа стане цінним інструментом для користувачів, дозволяючи їм легко створювати та налаштовувати телеграм-ботів, що відповідають їхнім потребам та вимогам.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є автоматизація створення телеграм-ботів, що спрощує процес їх розробки та налаштування, забезпечуючи високу доступність та адаптивність до індивідуальних потреб користувачів.

Основними задачами дослідження є:

- провести аналіз існуючих веб-платформ, методів і засобів автоматизованого створення телеграм-ботів;
- запропонувати нові:
  - методи спрощення взаємодії користувачів з платформою;
  - методи процесу налаштування ботів;
- розробити інтуїтивно зрозумілу веб-платформу, яка дозволяє створювати телеграм-ботів на основі запропонованих методів;
- провести тестування вебплатформи згідно поставлених задач.

Об'єкт дослідження – процес створення та налаштування телеграм-ботів за допомогою вебплатформи.

Предмет дослідження – методи та засоби, які використовуються для автоматизації створення телеграм-ботів.

Методи дослідження. У процесі досліджень використовувалися: методи розробки програмного забезпечення для створення та налаштування телеграм-ботів; веб-технології та фреймворки для розробки користувацького інтерфейсу та інтеграції з Telegram API; тестування для аналізу та перевірки функціональності вебплатформи та ботів.

Новизна отриманих результатів: Запропоновано новий метод автоматизованого створення телеграм-ботів, який забезпечує миттєве додавання токенів до бази даних і автоматичний перезапуск ботів з актуалізацією конфігурації. Розроблено унікальний інтерфейс та логіку взаємодії з веб-платформою, що дозволяє користувачам створювати телеграм-ботів без глибоких технічних знань. Також розроблено алгоритм аналітики, який збирає та аналізує дані про взаємодію користувачів з ботами для підвищення їхньої ефективності.

Практичне значення отриманих результатів полягає в тому, що на основі розроблених у курсовому проєкті теоретичних положень та алгоритмів створено програмні засоби для автоматизації процесу створення телеграм-ботів, що сприяє підвищенню ефективності комунікаційних процесів у цифровому середовищі.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, був зроблений особисто автором, який включає в себе ідею проекту, його проектування, розробку, тестування, аналіз потреб користувачів та ринку, щоб забезпечити максимальну ефективність та корисність продукту.

Структура та обсяг роботи. Пояснювальна записка до бакалаврської дипломної роботи складається з чотирьох розділів. У першому розділі розглянуто поточний стан веб-платформ для автоматизованого створення телеграм ботів, а також проведено порівняння запропонованого рішення з існуючими платформами та сформовано задачі. У другому розділі надано детальний опис реалізації проєкту, включаючи використані мови програмування, технології та інструменти. Також описано алгоритм та графічний інтерфейс, які використовуються для реалізації програмного забезпечення. Третій розділ присвячений розробці програмних модулів, необхідних для програмного застосунку. У розділі описані алгоритми та моделі, використані для розробки модулів, а також технології та інструменти, використані в процесі розробки. У четвертому розділі проведено тестування функціональних та графічних компонентів програмного забезпечення, розробленого в рамках бакалаврської дипломної роботи а також надано керівництво користувача для системи.

Апробація результатів здійснена на конференції XXIV Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів (2024) і опубліковані в тезах доповіді цієї конференції.

Пояснювальна записка містить 52 сторінок тексту, 39 рисунків, 2 таблиці, чотири додатки, 24 найменувань в списку використаних джерел.

1 АНАЛІЗ РОЗВИТКУ ТЕХНОЛОГІЙ СТВОРЕННЯ ТЕЛЕГРАМ БОТІВ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану технологій створення телеграм ботів

Сучасний етап розвитку технологій вимагає постійного вдосконалення методів аналізу та взаємодії, особливо в контексті розробки телеграм ботів. Традиційні методи створення ботів, які часто обмежуються статичними сценаріями, вимагають більшої гнучкості для адаптації до швидко змінюваних потреб користувачів. Вебплатформи з конструкторами сценаріїв та готовими шаблонами відкривають нові можливості для налаштування ботів, забезпечуючи широкий спектр функціональності та індивідуалізації.

Зростання швидкості та потужності обчислювальних систем є одним із ключових аспектів у розвитку технологій створення телеграм ботів. Сучасні комп'ютери та обчислювальні методи дозволяють швидко обробляти великі обсяги даних, що є критично важливим для розробки, оптимізації та налаштування ботів. Це також сприяє глибшому аналізу взаємодії ботів з користувачами, дозволяючи вдосконалювати їх поведінку та ефективність.

Важливим аспектом є інтеграція штучного інтелекту та машинного навчання, які дозволяють ботам самостійно вчитися та адаптуватися до нових умов, покращуючи якість взаємодії з користувачами. Використання передових алгоритмів обробки природної мови (NLP) та розуміння контексту розширює можливості ботів, роблячи їх більш інтуїтивно зрозумілими та зручними для користувачів.

Розвиток API та SDK для телеграм ботів відкриває двері для розробників, надаючи їм інструменти для створення ботів з розширеними функціями, такими як інтеграція з іншими сервісами, автоматизація завдань, збір та аналіз даних.

З огляду на ці тенденції, можна стверджувати, що майбутнє телеграм ботів буде зосереджене на створенні більш розумних, автономних та інтерактивних систем, які зможуть ефективно взаємодіяти з користувачами та вирішувати складні завдання. Розробка програмних засобів для створення телеграм ботів стає важливою частиною інноваційного процесу, що вимагає постійного оновлення знань та навичок, а також використання новітніх технологій для підвищення ефективності та конкурентоспроможності.

1.2 Порівняльний аналіз аналогів

Використання технологій для створення телеграм ботів набуває все більшої популярності, і ці технології інтегруються в різні вебплатформи та сервіси. Розробка програмних засобів для створення телеграм ботів з використанням конструктора сценаріїв та готових шаблонів забезпечує ефективність та гнучкість управління ботами, а також допомагає швидше реагувати на запити користувачів та оптимізувати взаємодію. До найбільш відомих рішень у цій області відносять:

- PuzzleBot;
- SendPulse;
- Gerabot;
- BotMother;
- BotHelp.

Одним із прикладів використання вебплатформи для створення телеграм ботів є сервіс Puzzlebot (рис. 1.1) - це багатофункціональний сервіс для роботи з групами і каналами в Telegram, а також для створення ботів. За допомогою цієї платформи можна створювати складних ботів, які будуть ефективно працювати в групах і каналах месенджера Telegram - здійснювати постинг, вести статистику, модерувати контент і багато іншого [5].

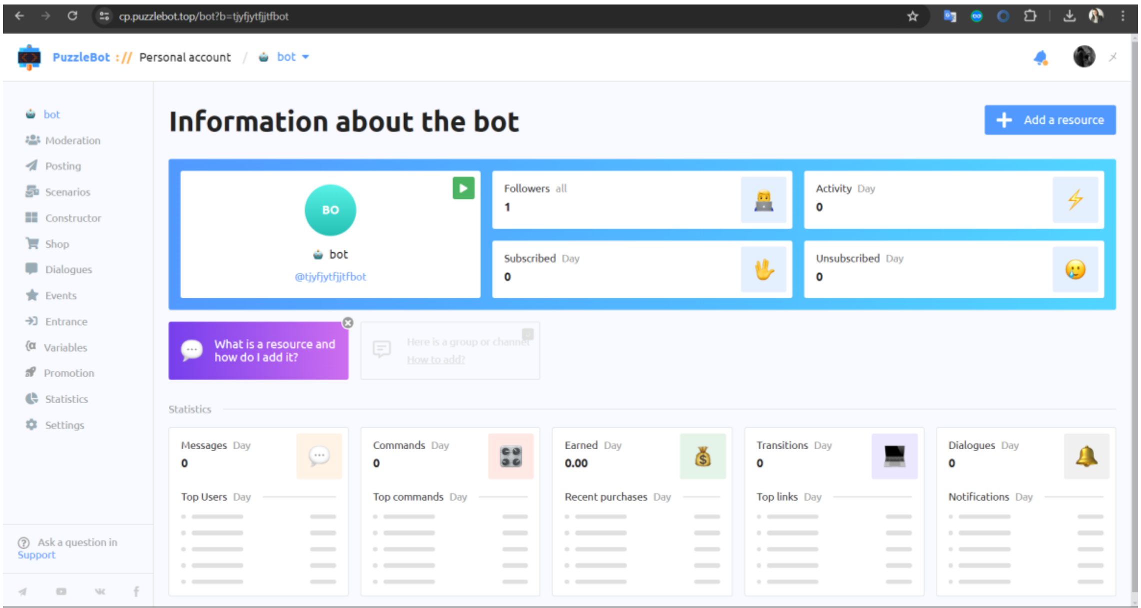

Рисунок 1.1 – Приклад роботи вебплатформи PuzzleBot

Інший приклад – SendPulse (рис. 1.2), єдина платформа автоматизації маркетингу для створення email та SMS розсилок, web push сповіщень та чат-ботів у месенджерах [6]. Він надає користувачам можливість створювати телеграм ботів безкоштовно, тарифи надають можливість використовувати великий функціонал у вигляді чат-ботів – до 1000 передплатників та web push – до 10 000. Цей інструмент часто використовується великими компаніями для зручної взаємодії з користувачами.

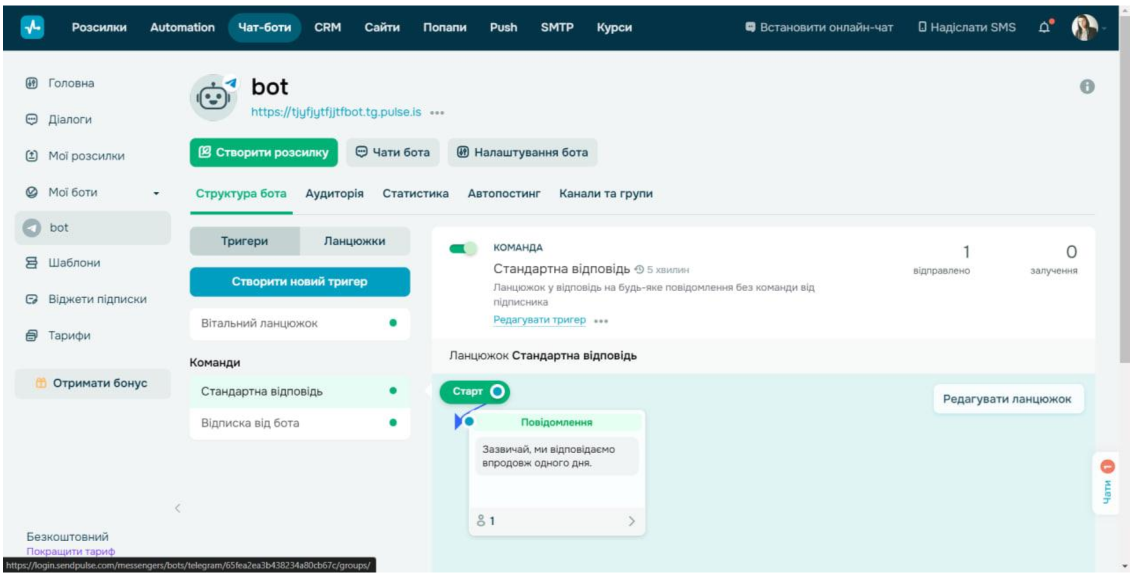

Рисунок 1.2 – Приклад роботи платформи sendpulse

Gerabot (рис. 1.3) є іншим програмним засобом, конструктор чат-ботів дозволяє розробку логіки чат-бота та подальше її впровадження в різні канали зв’язку: чат-бот у Telegram, Viber та онлайн-ассистент на сайті компанії. Вбудована в конструктор CRM система зберігає користувацькі дані в одному місці та використовує їх для всіх комунікаційних засобів, формуючи лояльність клієнта [7]. Цей інструмент використовується для розробки чат-ботів, які можуть точно відповідати на запити користувачів, забезпечуючи високу точність і швидкість обробки інформації. Варто зазначити, що дане програмне забезпечення може включати візуалізацію даних, що полегшує розуміння статистики та аналітики взаємодії з користувачами.

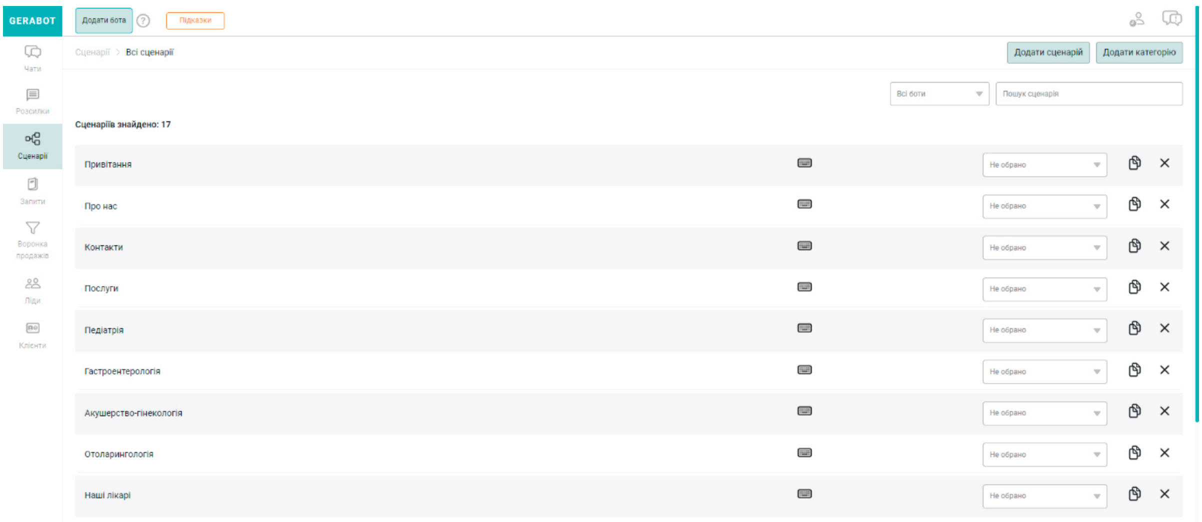

Рисунок 1.3 – Приклад роботи платформи gerabot

BotMother (рис. 1.4) веб платформа, яка дозволяє користувачам створювати чат-ботів для різних месенджерів, включаючи Telegram, без необхідності програмування. Вона пропонує візуальний редактор, де боти собираються з готових блоків простим перетягуванням [8]. Особливістю BotMother є можливість інтеграції з новітніми штучним інтелектом, включаючи ChatGPT. Це означає, що користувачі можуть створювати ботів, які використовують потужні можливості ChatGPT для обробки природної мови та взаємодії з користувачами на більш глибокому рівні.

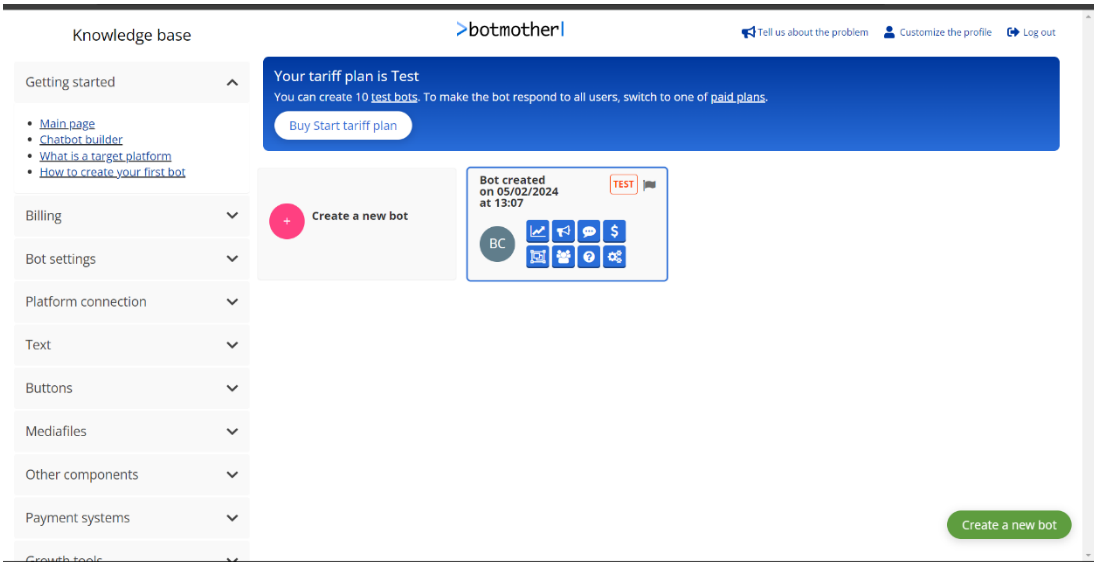

Рисунок 1.4 – Приклад роботи платформи BotMother

BotHelp (рис. 1.5) веб платформа, яка дозволяє створювати чат-ботів для Telegram без необхідності програмування. Цей інструмент використовується для автоматизації комунікації з клієнтами, збільшення бази контактів та відправлення розсилок [9]. Завдяки простому налаштуванню та інтуїтивному інтерфейсу, користувачі можуть швидко створити чат-бота, який буде відповідати на запити, збирати дані та надавати підтримку 24/7.

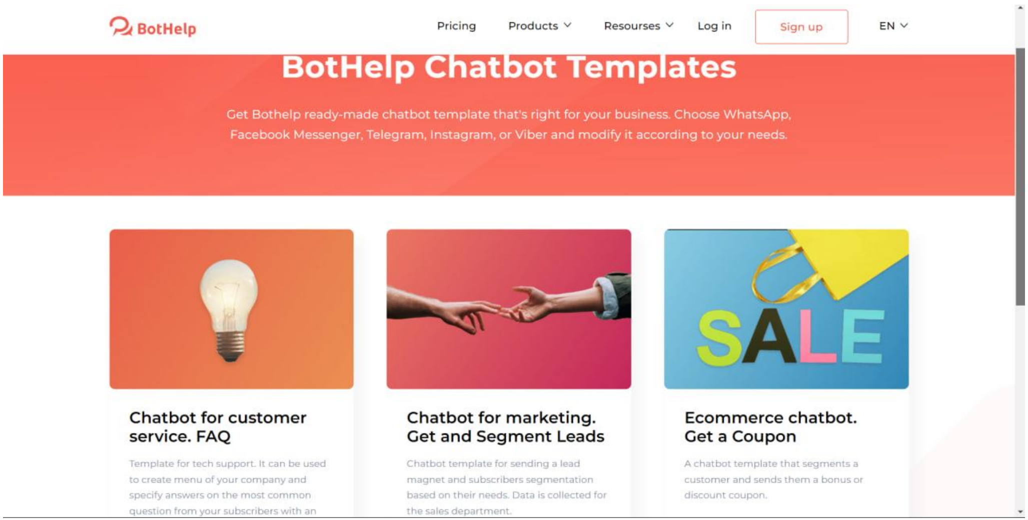

Рисунок 1.5 – Приклад роботи платформи BotHelp

Розробка програмних засобів вебплатформи для створення телеграм ботів з використанням конструктора сценаріїв та готових шаблонів вимагає досвіду роботи з веб розробкою, програмуванням чат-ботів та дизайном користувацьких інтерфейсів. Це передбачає створення внутрішньої системи, яка буде здатна обробляти команди та повідомлення від користувачів, а також виконувати автоматизовані дії згідно з налаштованими сценаріями. Складова інтерфейсу користувача передбачає розробку інтуїтивно зрозумілих інтерфейсів, які дозволять користувачу легко взаємодіяти з розробленими функціями бота.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	Puzzle Bot	SendPulse	Gerabot	BotMother	BotHelp	Власний модуль
Інтуїтивність інтерфейсу	+	+	-	-	+	+
Гнучкість налаштувань	-	-	+	+	+	+
Підтримка конструктора сценаріїв	+	+	-	+	+	+
Інтеграція з іншими сервісами	+	+	+	+	-	+
Підтримка шаблонів	+	+	-	+	-	+
Загальна оцінка	80%	80%	40%	80%	60%	100%

Відповідно до аналізу порівняльних характеристик, розробка власних програмних засобів вебплатформи для створення телеграм ботів з використанням конструктора сценаріїв та готових шаблонів є доцільною. Отриманий продукт зможе покрити недоліки існуючих рішень, та забезпечити розширені можливості для створення та налаштування телеграм ботів, включаючи більш гнучкі налаштування, інтеграцію з іншими сервісами, та автоматизацію взаємодії з користувачами. Платформа може пропонувати унікальні функції, які відповідають специфічним потребам користувачів, та надавати високу якість обслуговування та користувацького досвіду.

Отже, розробка програмних засобів вебплатформи для створення телеграм ботів з використанням конструктора сценаріїв та готових шаблонів відкриває перед бізнесом та розробниками широкі можливості для покращення комунікації, маркетингу та обслуговування клієнтів. Завдяки гнучким та інтуїтивно зрозумілим інструментам для створення ботів, фахівці можуть здійснювати більш точне та ефективне налаштування взаємодії з користувачами, що є критичним для підвищення задоволеності клієнтів та оптимізації бізнес-процесів.

1.3 Аналіз методів розв’язання задачі

Розробка програмного забезпечення для автоматизованого створення телеграм ботів з використанням конструктора сценаріїв та готових шаблонів вимагає уваги до різних методів та підходів для досягнення бажаного результату.

Одним із засобів, що широко використовується, є веб хуки. Вони дозволяють ботам отримувати та обробляти повідомлення в режимі реального часу, забезпечуючи швидку та ефективну взаємодію з користувачами. Використання веб хуків дозволяє ботам реагувати на події та взаємодіяти зі своєю аудиторією у режимі реального часу, що робить їх більш привабливими для користувачів.

Окрім цього, API (інтерфейс програмування додатків) є ще одним потужним інструментом для розробки телеграм ботів. Використання API дозволяє ботам взаємодіяти з іншими сервісами та додатками, розширюючи їхні можливості та функціональність. Цей підхід дозволяє ботам інтегруватися з різноманітними сервісами та джерелами інформації, що забезпечує користувачів більш широким спектром послуг.

Поєднання веб хуків із конструктором сценаріїв та готовими шаблонами дозволяє розробникам швидко створювати та налаштовувати різноманітні функції ботів без необхідності писати код з нуля. Такий підхід спрощує процес розробки та дозволяє швидко впроваджувати новий функціонал, що робить розробку телеграм ботів більш ефективною та привабливою для розробників.

Після уважного аналізу переваг і недоліків кожного з підходів, ми прийшли до висновку, що оптимальним рішенням буде поєднання методів веб хуків з використанням API та конструктора сценаріїв та готових шаблонів. Ця комбінація забезпечить більшу гнучкість та функціональність у розвитку телеграм ботів. Додавши до цього елементи персоналізації та адаптації до потреб користувачів, ми зможемо створити платформу, яка буде найбільш ефективною та зручною для використання.

1.4 Постановка задач курсового проєкту

Після уважного аналізу існуючих програмних засобів для автоматизованого створення телеграм ботів з використанням конструктора сценаріїв та готових шаблонів, були визначені наступні завдання, які необхідно виконати для успішної розробки програмного забезпечення:

- розробити алгоритм для автоматичного створення телеграм ботів з використанням конструктора сценаріїв та готових шаблонів;
- реалізувати інтерфейс користувача для зручного управління ботами та їх параметрами;
- провести тестування програмного забезпечення для перевірки його функціональності та надійності;
- забезпечити можливість масштабування та розширення функціоналу програмного забезпечення в майбутньому;

- здійснити інтеграцію з іншими сервісами та платформами для розширення можливостей ботів;
- автоматизація процесу створення телеграм ботів;
- провести тестування програмного забезпечення згідно поставлених задач.

1.5 Висновки

Проведений аналіз стану технологій створення телеграм ботів свідчить про постійне вдосконалення методів у цій області. Сучасні вебплатформи з конструкторами сценаріїв та готовими шаблонами відкривають нові можливості для налаштування ботів, забезпечуючи широкий спектр функціональності та індивідуалізації, що відповідає зростаючим потребам ринку. Порівняльний аналіз аналогів дозволив ідентифікувати переваги та недоліки різних рішень у цій сфері.

Виявлено, що розробка власних програмних засобів для створення телеграм ботів з використанням конструктора сценаріїв та готових шаблонів є доцільною стратегією. Такий підхід дозволить покрити недоліки існуючих рішень, забезпечуючи розширені можливості для створення та налаштування ботів. Власна платформа зможе пропонувати унікальні функції, які відповідають специфічним потребам користувачів, та забезпечити високу якість обслуговування та користувацький досвід. Такий підхід відкриває перспективи для подальшого розвитку та впровадження інноваційних рішень у сфері створення телеграм ботів.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

Для розробки програмного забезпечення для автоматизованого створення телеграм ботів з використанням конструктора сценаріїв та готових шаблонів буде використано мову програмування Python для реалізації бекенду, а також HTML, CSS та JavaScript для розробки фронтенду. Для взаємодії з Telegram API та створення ботів буде використана фреймворк aiogram3. Python – [10] це інтерпретована, високорівнева мова програмування з простим синтаксисом, що використовується для розробки всього, від штучного інтелекту до телеграм ботів. Aiogram – [11] це сучасний та повністю асинхронний фреймворк для розробки чат-ботів Telegram Bot API на Python 3.8 з використанням asyncio та aiohttp. Поєднання Python для бекенду, HTML, CSS і JavaScript для фронтенду, а також бібліотеки aiogram3 для створення ботів дозволить реалізувати програмне забезпечення для автоматизованого створення телеграм ботів з використанням конструктора сценаріїв та готових шаблонів.

Розробка програмного забезпечення для автоматизованого створення телеграм ботів - це складний та багатоетапний процес, який вимагає уваги до кожної деталі. Перший етап цього процесу - розробка модуля для зручної та інтуїтивно зрозумілої взаємодії з системою. Цей модуль має забезпечувати зручний і простий доступ до функціоналу програмного забезпечення, а також ефективну навігацію для користувачів.

На другому етапі передбачено розробку модуля для імпорту, аналізу та обробки даних. Використовуючи мову програмування Python та бібліотеку aiogram3, цей модуль буде відповідати за взаємодію з телеграм ботами, а також за обробку інформації, необхідної для їх функціонування.

Наступним етапом буде розробка модуля для автоматичного створення телеграм ботів за допомогою конструктора сценаріїв та готових шаблонів. Цей модуль спростить процес розробки ботів, надаючи користувачам можливість швидко та ефективно створювати своїх телеграм ботів без необхідності програмування з нуля.

Останнім, але не менш важливим етапом є розробка модуля для візуалізації створених ботів та їх функціоналу. Цей модуль дозволить користувачам налаштовувати та переглядати можливості своїх ботів у зручному інтерфейсі.

Для успішної реалізації програмного забезпечення будуть використані різні інструменти та технології, такі як мова програмування Python для бекенду, HTML, CSS і JavaScript для фронтенду, а також бібліотека aiogram3 для роботи з телеграм ботами. Важливим аспектом є також тестування кожного модулю для забезпечення якості та надійності програмного забезпечення.

Отже, розробка програмного забезпечення для автоматизованого створення телеграм ботів - це складний та відповідальний процес, який вимагає не лише технічних знань, але й уваги до потреб користувачів та деталей на кожному етапі розробки.

2.2 Розробка інтерфейсу програмного додатку

Розробка інтерфейсу вебплатформи для створення телеграм ботів включала ретельне планування та увагу до його зрозумілості та зручності для користувачів.

Основним кольором інтерфейсу було обрано синій, оскільки цей колір асоціюється з телеграм. Допоміжними кольорами були використані білий та чорний, для створення контрасту та поліпшення зручності використання. Окрім основних кольорів, особлива увага була приділена анімації заднього фону, щоб зробити інтерфейс більш живим та цікавим. Плавна анімація рухомого заднього фону, з використанням CSS та JavaScript, додає динаміки та привабливості веб платформі.

При першому відкритті вебплатформи, користувача вітає екран з запрошенням ввести токен їх боту телеграм, щоб розпочати процес створення бота (рис. 2.1).

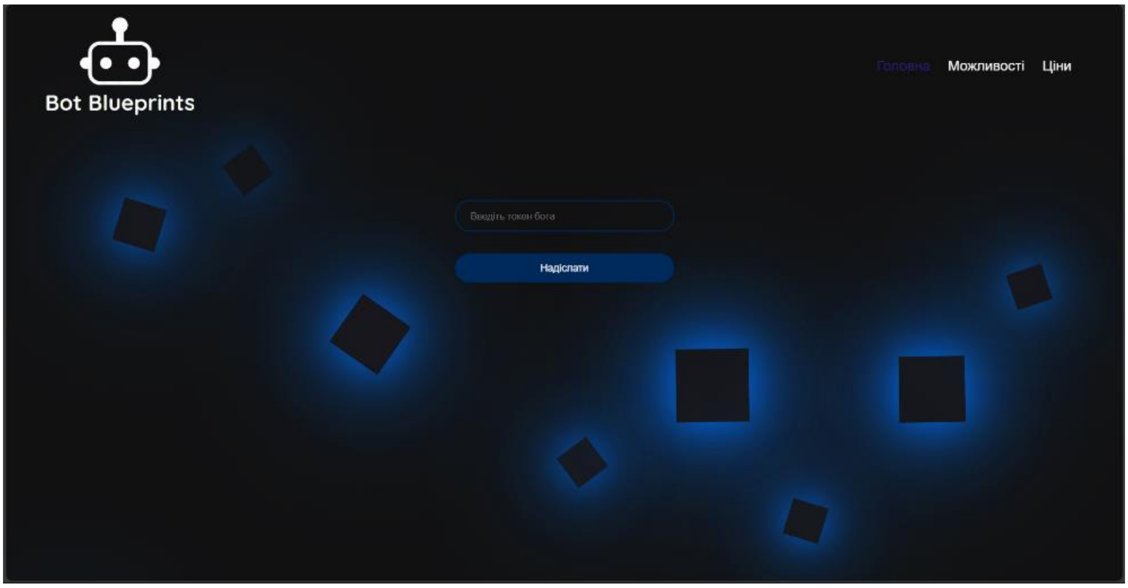

Рисунок 2.1 – Головний екран вебплатформи

Оскільки цільовим завданням програмного забезпечення є автоматизоване створення телеграм ботів, вирішено розробити віртуальне робоче середовище з чітким розмежуванням між конструктором сценаріїв та готовими шаблонами. Меню програмного застосунку буде оптимізоване для зручності взаємодії з розробленим функціоналом (рис. 2.2). На початковому етапі, після введення токену та натискання кнопки "Надіслати", користувач буде перенаправлений на сторінку інформації про бота (рис. 2.3).

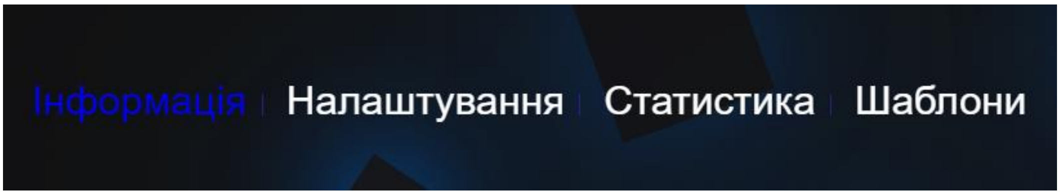

Рисунок 2.2 – Меню вебплатформи

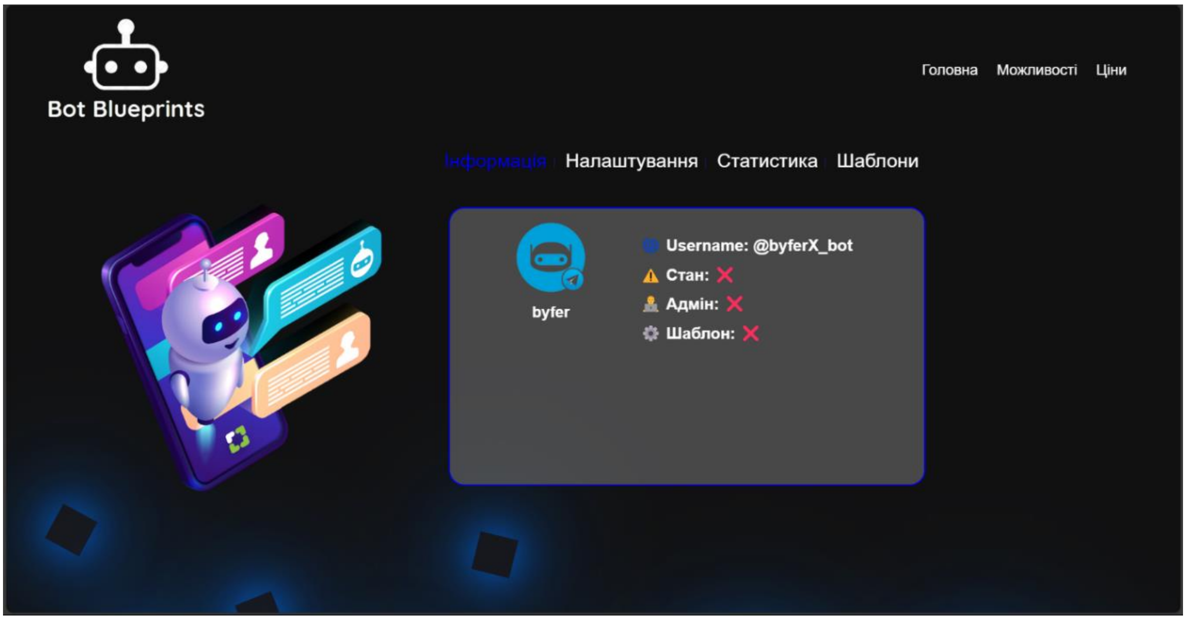

Рисунок 2.3 – Сторінка інформації про бота

Після ознайомлення з інформацією про бота, користувач може перейти на сторінку налаштування свого боту, натиснувши на відповідний пункт меню. Після цього стан віртуального робочого середовища зміниться (рис. 2.4), і на екрані з'явиться сторінка для налаштування бота з можливістю зміни параметрів і налаштувань.

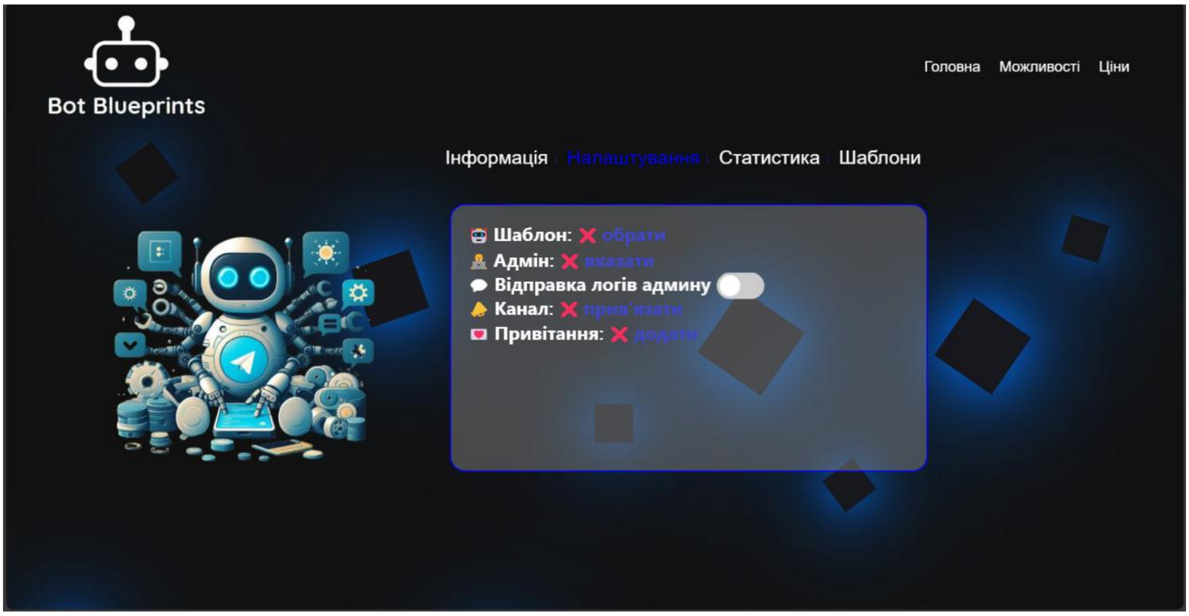

Рисунок 2.4 – Сторінка налаштування бота

Після усіх налаштувань боту користувач може перейти до свого телеграм боту, натиснувши на юзернейм на сторінці інформації про бота. Та вдосконалитись що бот працює натиснувши на кнопку “start” у нижній частині екрану (рис. 2.5).

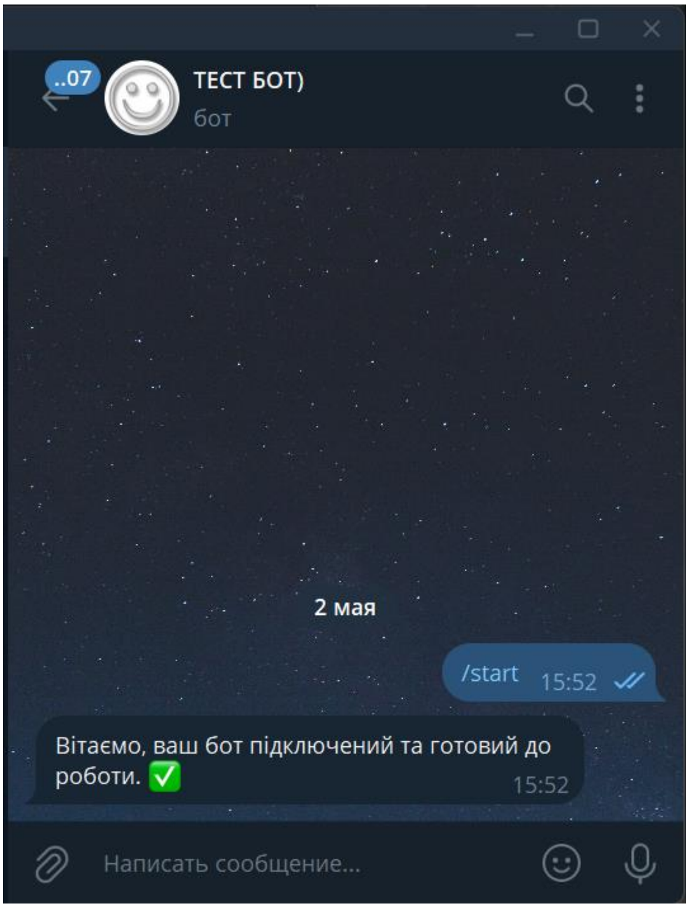

Рисунок 2.5 – Перевірки працездатності боту

Перейшовши в меню на можливості (рис.2.6), користувач може ознайомитися із основним функціоналом платформи. Це включає в себе різноманітні опції для налаштування та управління телеграм-ботами, такі як створення нових ботів, редагування існуючих, та перегляд їх статистики. Крім того, користувач може переглянути додаткову інформацію про програмний застосунок, що включає детальні інструкції з користування платформою, відповіді на часто задавані питання, а також технічну документацію, яка допоможе розібратися в нюансах роботи з ботами. У цьому розділі у майбутньому будуть представлені відео-уроки та інтерактивні підказки, які покликані полегшити користувачам роботу з платформою. Завдяки цьому, користувачі зможуть максимально ефективно використовувати всі можливості платформи для досягнення своїх цілей.

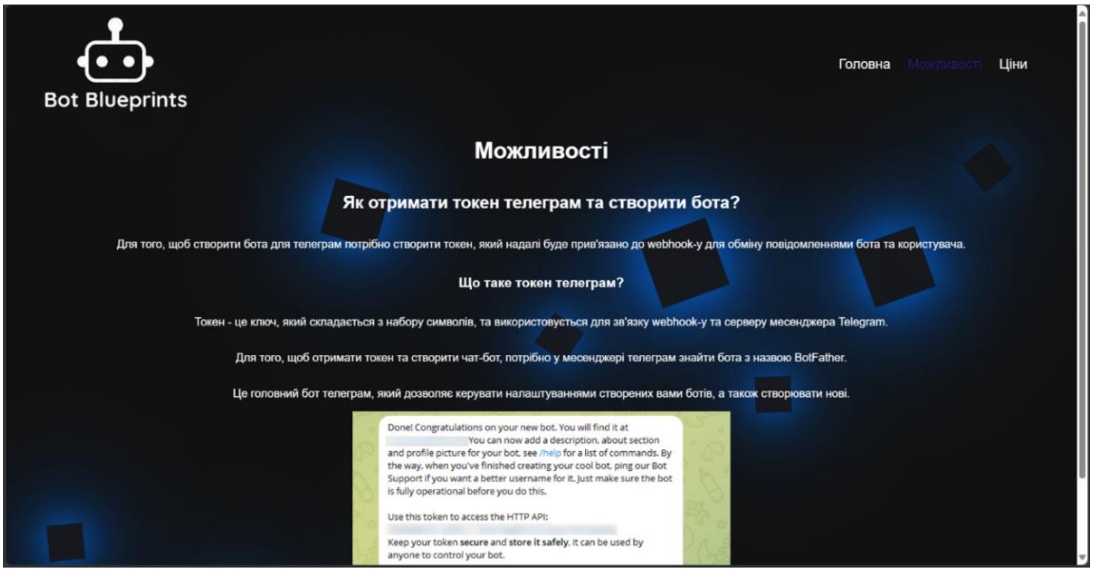

Рисунок 2.6 – Сторінка «Можливості»

На сторінці «Ціни» (рис. 2.7) користувач може побачити інформацію про тарифи платформи, ліміти, та контакти технічної підтримки.

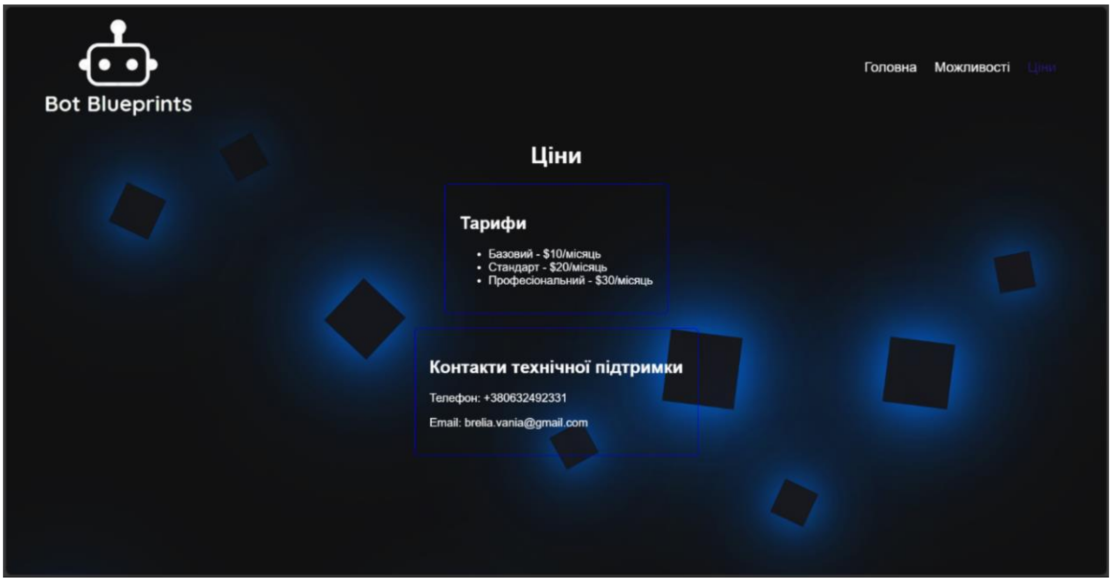

Рисунок 2.7 – Сторінка «Ціни»

Також веб платформа оптимізована для мобільних пристроїв (рис. 2.8).

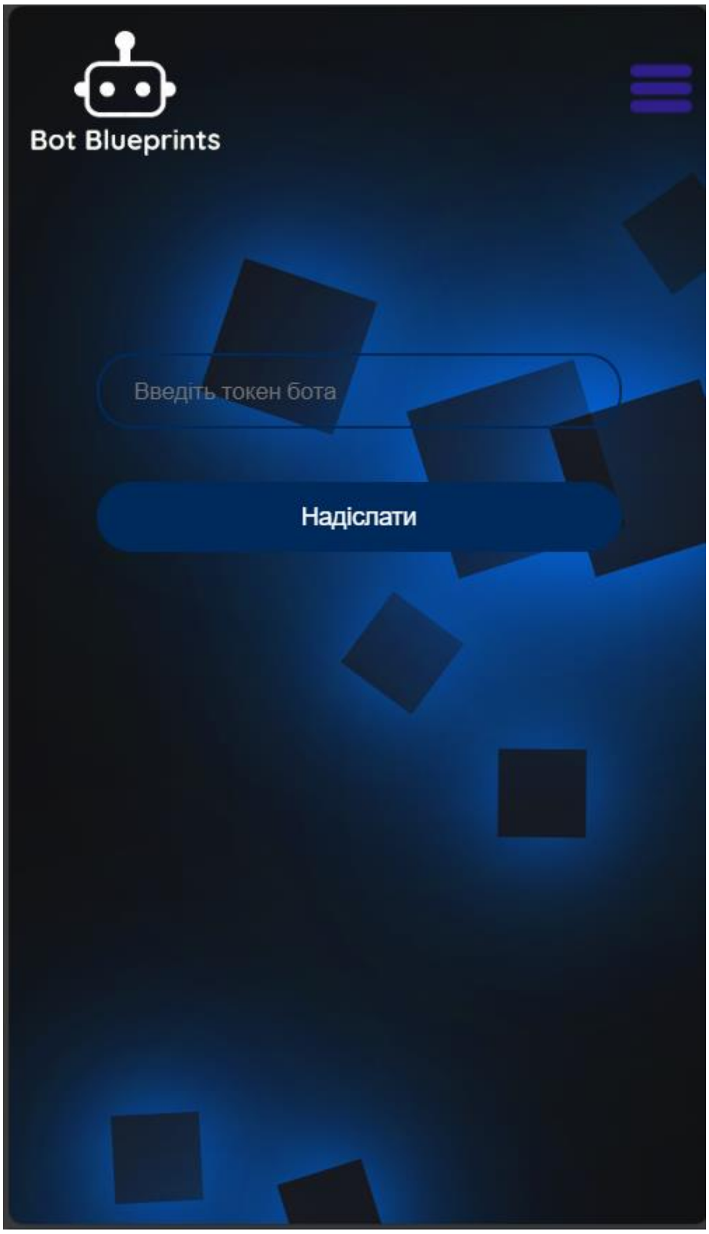

Рисунок 2.8 – Головна сторінка, вигляд з мобільного телефону

Розробка графічного інтерфейсу програмного застосунку проводилась у середовищі розробки Sublime Text з використанням HTML, CSS та JavaScript. Sublime Text обрано за його зручність і функціональність, що дозволяє ефективно організувати робочий процес і прискорити розробку. HTML використовувався для створення структури веб-сторінок, забезпечуючи чітку і логічну організацію контенту. CSS було застосовано для стилізації елементів, надаючи інтерфейсу естетично привабливий вигляд і забезпечуючи його адаптивність для різних пристроїв та екранів. JavaScript, у свою чергу, використовувався для додавання інтерактивності та динамічності інтерфейсу, дозволяючи реалізувати складні функціональні можливості, такі як валідація форм, асинхронне завантаження даних та взаємодія з сервером через API.

2.3 Розробка моделі системи

Для кращого розуміння роботи модулів програмного забезпечення для автоматизованого створення телеграм-ботів з використанням конструктора сценаріїв та готових шаблонів було вирішено розробити модель роботи системи на прикладі UML [12] діаграми діяльності (рис. 2.9). Ця діаграма дозволяє детально відобразити послідовність дій, необхідних для створення та налаштування телеграм-ботів, що сприяє кращому розумінню процесу як для розробників, так і для користувачів.

Згідно з вказаною діаграмою, процес починається з того, що користувач спочатку повинен надіслати токен свого бота до платформи. Якщо токен неправильний або невірний, система видасть відповідне повідомлення про помилку, інформуючи користувача про необхідність перевірки правильності введених даних. Це допомагає уникнути помилок на подальших етапах і забезпечує коректну роботу бота.

На наступному етапі передбачається перевірка токену в базі даних. Якщо токен виявлено в базі даних, програма надасть інформацію про бота, взяту з бази даних. Це може включати такі деталі, як назва бота, його статус, конфігурації та інші налаштування, що були збережені раніше. Якщо токен не знайдено, платформа виведе порожню інформацію про бота та автоматично додасть його до бази даних для подальшої обробки. Це забезпечує, щоб усі боти були належним чином зареєстровані та готові до використання.

Після успішного виведення інформації, програмне забезпечення для створення телеграм-ботів автоматично ініціює запуск телеграм-бота за допомогою наданого токену. Цей процес включає активацію всіх необхідних сервісів та налаштувань для забезпечення коректної роботи бота. Після цього користувач може перейти до налаштування бота згідно зі своїми потребами, використовуючи зручний та інтуїтивно зрозумілий інтерфейс платформи.

Наступний крок включає вибір шаблону для свого бота. Користувач може обрати з кількох запропонованих шаблонів, кожен з яких призначений для різних сценаріїв використання, таких як автоматизація відповіді на запити клієнтів, управління запитами на підтримку або організація маркетингових кампаній. Після вибору шаблону платформа автоматично адаптує бота відповідно до обраного шаблону, налаштовуючи всі необхідні параметри та інтеграції.

Після завершення процесу створення, налаштування та обрання шаблону для бота, користувач може переглядати статистику свого бота на спеціальній сторінці статистики. Це включає аналіз взаємодії користувачів з ботом, зібрані дані про кількість запитів, відповіді, рівень залученості користувачів та інші ключові показники ефективності. Така аналітика дозволяє користувачам краще розуміти, як їхні боти функціонують та де можуть бути покращення для підвищення їх ефективності.

Таким чином, детальна UML діаграма діяльності (рис. 2.9) не лише описує послідовність дій користувача, але й забезпечує зрозумілий і послідовний підхід до розробки, налаштування та аналізу роботи телеграм-ботів.

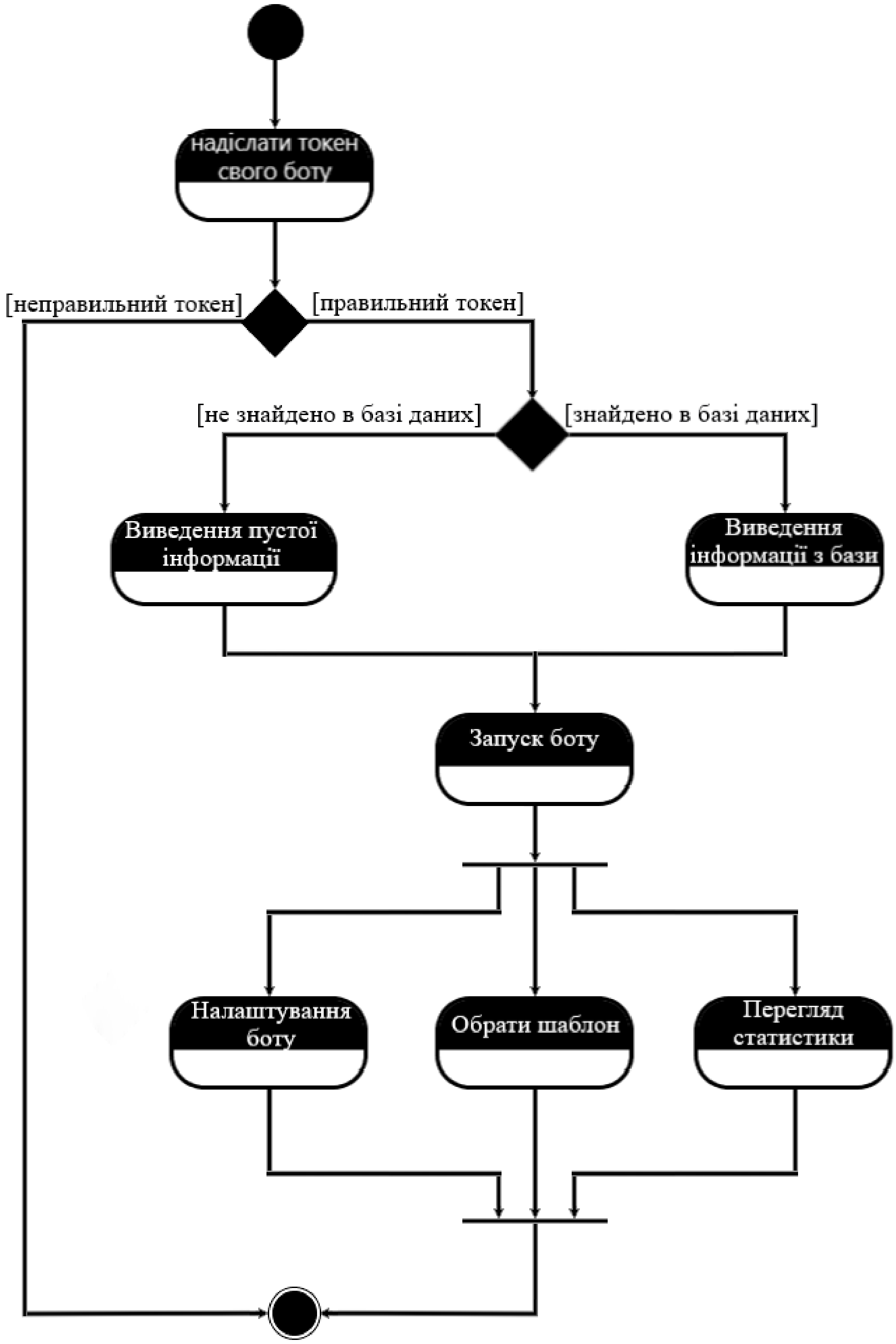

Рисунок 2.9 – Діаграма діяльності програмних засобів

Розробка діаграми діяльності була виконана у програмному забезпеченні Visio [13].

2.4 Розробка алгоритмів роботи системи

Для ефективної розробки програмного забезпечення, що дозволяє автоматизувати створення телеграм ботів, важливо розробити детальний алгоритм дій системи. Цей алгоритм буде основою для програмного модуля, який відповідає за логіку роботи ботів (рис. 2.10).

Першим кроком у цьому алгоритмі є завантаження головної сторінки програми. Ця сторінка є інтерактивним інтерфейсом, який надає користувачам доступ до конструктора сценаріїв та готових шаблонів. Завдяки цьому, користувачі можуть швидко та легко створювати нові телеграм боти або модифікувати існуючі.

Після доступу до головної сторінки, користувачу потрібно надіслати токен свого бота через безпечне з'єднання. Токен є унікальним ідентифікатором, який забезпечує зв'язок між ботом та сервером. Програма перевіряє введений токен на відповідність та наявність у базі даних. Якщо токен відсутній у базі, він автоматично додається, а головний модуль ботів перезапускається з оновленою інформацією.

У разі успішної перевірки токена, користувач перенаправляється на сторінку з інформацією про бота. На цій сторінці відображається детальна інформація з бази даних, включаючи статистику використання, налаштування та історію взаємодії бота з користувачами.

Крім того, програмне забезпечення надає можливість використання готових шаблонів для створення ботів, які можуть бути легко адаптовані під індивідуальні потреби користувачів. Це включає в себе широкий спектр сценаріїв взаємодії, від простих авто-відповідей до складних інтеграцій з зовнішніми сервісами та базами даних.

Завершальним етапом є тестування створеного бота. Користувач може запустити бота в тестовому режимі, щоб переконатися у його правильній роботі та відповідності всім вимогам. Після успішного тестування, бот готовий до запуску в реальних умовах.

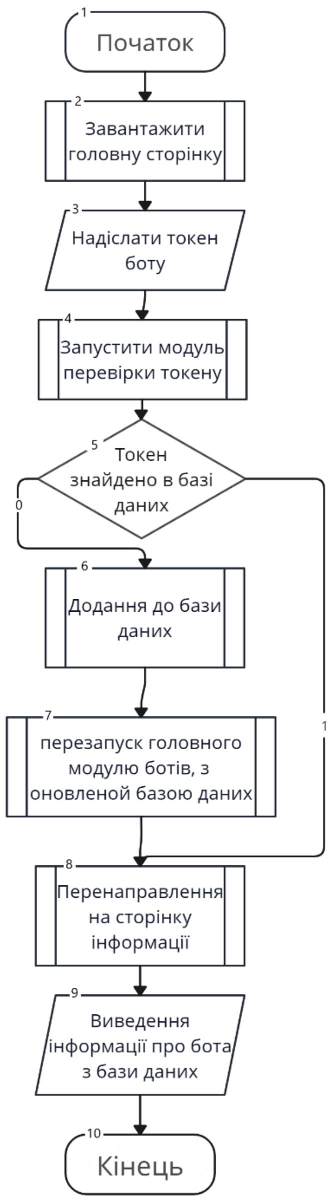

Рисунок 2.9 – Алгоритм роботи вебплатформи

Після завантаження головної сторінки та введення токена, програма переходить до модулю визначення спорідненості ботів. Якщо боти мають спорідненість, наприклад, за функціоналом або за сценарієм взаємодії, проводиться кореляційна лінія. У випадку відсутності спорідненості, програма створює площину для подальшої роботи з ботом.

Далі програма завантажує стандарти сценаріїв та шаблонів, які визначають основні точки взаємодії бота з користувачем. Ці точки позначаються на кореляційних лініях та площинах, що дозволяє точно визначити логіку роботи бота та його взаємодію з користувачем.

Після ідентифікації цих точок відбувається налаштування параметрів взаємодії, яке включає в себе вимірювання відстаней між різними елементами сценарію. Наприклад, може бути виміряно час відповіді бота на запит користувача або довжину тексту, який бот генерує у відповідь.

Завершальним етапом є виведення результатів налаштувань та тестування бота. Користувач отримує звіт про взаємодію бота з користувачами, ефективність використання сценаріїв та шаблонів, а також рекомендації щодо оптимізації роботи бота.

2.5 Висновки

У другому розділі дипломної роботи було проведено детальний аналіз вхідних та вихідних даних, необхідних для розробки програмного забезпечення, спрямованого на автоматизацію процесу створення телеграм ботів. Було розглянуто ключові аспекти створення графічного інтерфейсу, який забезпечує інтуїтивно зрозуміле та ефективне управління ботами.

Основний алгоритм роботи програмних засобів було розроблено з урахуванням потреб користувачів та особливостей взаємодії з телеграм ботами. Також було створено алгоритм визначення спорідненості ботів, що дозволяє оптимізувати процес їх створення та налаштування.

Для візуалізації роботи програмного продукту було розроблено модель за допомогою UML діаграм, яка відображає всі етапи від створення бота до його запуску та моніторингу. Це дозволяє користувачам глибше зрозуміти логіку роботи системи та ефективно управляти своїми ботами.

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ВЕБПЛАТФОРМИ ДЛЯ СТВОРЕННЯ ТЕЛЕГРАМ БОТІВ З ВИКОРИСТАННЯМ КОНСТРУКТОРА СЦЕНАРІЇВ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

У сучасному світі програмування, вибір правильних технологій для розробки програмного забезпечення є критично важливим. Це вимагає глибокого розуміння як поточних технічних трендів, так і довгострокових перспектив розвитку технологій. При створенні програмного забезпечення для телеграм ботів, особливу увагу слід приділити вибору мови програмування, фреймворків, баз даних, а також інструментів для фронтенду та бекенду.

Python є мовою вибору для багатьох розробників завдяки своїй універсальності та великій екосистемі. Вона ідеально підходить для швидкої розробки та прототипування, а також має сильну підтримку для роботи з асинхронним кодом через модуль asyncio. Це особливо важливо для телеграм ботів, які часто вимагають високої продуктивності та здатності обробляти велику кількість запитів одночасно.

Flask є вибором для бекенду завдяки своїй легковазі та гнучкості [14]. Він дозволяє розробникам легко інтегрувати різноманітні веб сервіси та створювати RESTful API, які є основою для взаємодії з телеграм ботами [15]. Sqlite3 пропонує простоту та надійність для зберігання даних, що робить його ідеальним для проектів, де не потрібна складна інфраструктура баз даних [16].

На фронтенді, HTML, CSS та JavaScript є основою для створення користувацьких інтерфейсів. Вони дозволяють створювати різноманітні веб сторінки, які можуть бути як простими, так і складними, залежно від потреб проекту. Використання сучасних фреймворків, таких як React або Angular, може додати додаткову міць та гнучкість у створенні інтерактивних веб додатків.

Aiogram3 є вибором для створення телеграм ботів на Python, оскільки вона надає багато інструментів для асинхронної роботи з Telegram API [16]. Це дозволяє розробникам створювати ботів, які можуть швидко реагувати на повідомлення користувачів та ефективно виконувати задачі.

Вибір asyncio як бібліотеки для асинхронного програмування є ключовим для підвищення продуктивності та ефективності телеграм ботів. Вона дозволяє розробникам використовувати асинхронні функції та корутини для оптимізації виконання коду, особливо в умовах високого навантаження.

Враховуючи ці технології, можна створити масштабовану та ефективну систему для створення телеграм ботів, яка буде відповідати сучасним вимогам до швидкості, безпеки та гнучкості. Вибір правильних інструментів та технологій є ключовим для успішної реалізації проекту, а також для забезпечення легкості підтримки та розвитку програмного забезпечення в майбутньому.

У таблиці 3.1 наведено порівняльну характеристику мов програмування для виявлення найбільш точної для виконання визначених задач.

Таблиця 3.1 – Порівняльна таблиц ямов програмування

Критерій	C	R	C++	Java	Python	JavaScript
Функціональність	+	–	+	+	+	+
Безпека даних	+/–	–	+/–	–	+	–
Вартість впровадження та підтримки	+	+	+	–	+	+
Крос-платформ. сумісність	–	–	–	+	+/–	–
Швидкодія	+	-	+	+	+	+

Простота використання	–	+	–	–	+	+
Розширюваність	+	+	+	+	+	+
Спільнота розробників	+	+	+	+	+	+

Отже, враховуючи, що тема передбачає автоматизацію створення телеграм-ботів, Python, з її простотою синтаксису та широким використанням у розробці ботів для Telegram, виявляється найбільш оптимальним вибором. Вона поєднує в собі надійність, швидкість та безпеку, що важливо для таких проектів, і має широкий набір інструментів для розробки. Таким чином, Python може бути найкращим варіантом для реалізації вашої вебплатформи для автоматизованого створення телеграм-ботів.

3.2 Розробка модуля перевірки токену, та додання його до бази даних

При розробці модуля перевірки токену для телеграм бота важливо враховувати ряд технічних аспектів, які забезпечують безпеку та надійність взаємодії з Telegram API. Для початку, необхідно здійснити перевірку токену, який є унікальним ідентифікатором бота в Telegram. Перевірка токену відбувається через запит до `api.telegram.org/bot{token}/getMe`, який дозволяє підтвердити автентичність бота та його доступ до Telegram API.

Даний запит відправляється до сервера Telegram і, у випадку успішної відповіді, повертає інформацію про бота, таку як його ім'я та username. Це дозволяє не тільки переконатися, що токен є дійсним, але й отримати основні дані про бота для подальшої роботи. У випадку помилки або невідповідності токену, система повинна надати користувачу відповідне повідомлення про помилку та інструкції для вирішення проблеми (рис. 3.1).

Після успішної перевірки токену, система переходить до наступного етапу - запуску бота. Запуск бота включає в себе ініціалізацію бота з використанням отриманого токену та налаштування webhook [17] або long polling [18] для обробки вхідних повідомлень від користувачів. Це вимагає встановлення зв'язку з серверами Telegram та налаштування обробників повідомлень, які будуть реагувати на команди користувачів.

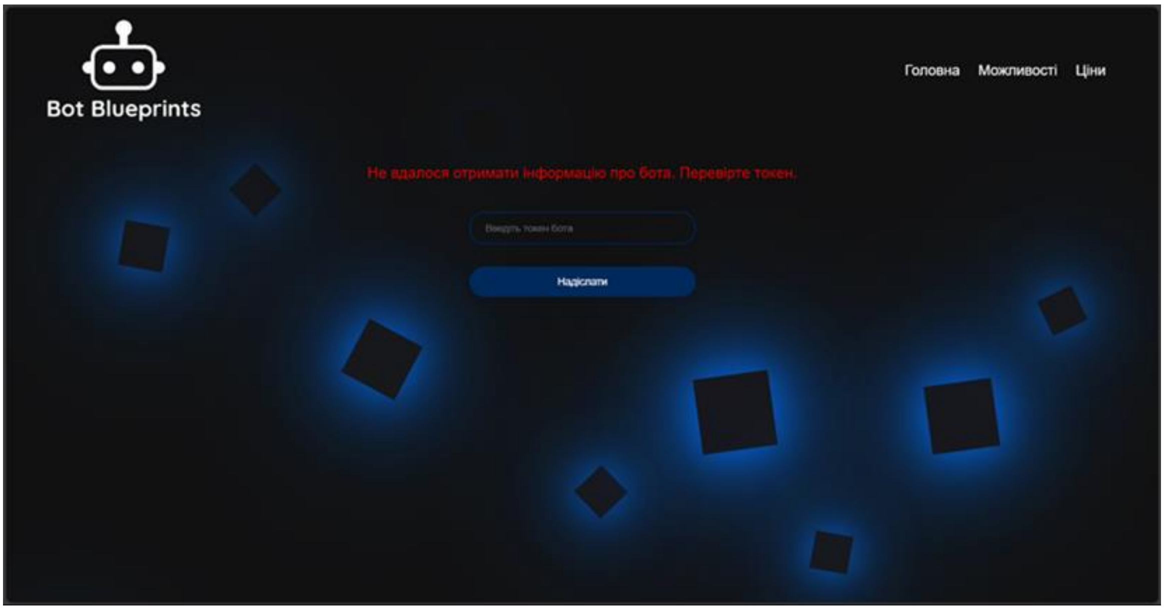

Рисунок 3.1 – Відправка невірно формату токену

У випадку, коли токен є дійсним та робочим, система перенаправляє користувача на сторінку з інформацією про бота, де проходить наступна перевірка наявності токену у базі даних. Натомість, якщо токен не пройшов перевірку, користувача перенаправляють на сторінку помилки, де надаються інструкції щодо виправлення помилок або повторного введення токену.

Після підтвердження дійсності токену, система ініціює процес перевірки наявності токену у базі даних (рис. 3.2). Цей крок є важливим для забезпечення синхронізації даних бота з існуючою інформацією та для уникнення дублювання записів. У випадку, коли токен вже присутній у базі, система перенаправляє користувача на сторінку з інформацією про бота.

Якщо ж токен відсутній у базі даних, це означає, що бот є новим або не був раніше зареєстрований. В такому випадку система автоматично додає токен до бази даних, що дозволяє інтегрувати бота з іншими компонентами системи та забезпечити його готовність до взаємодії з користувачами.

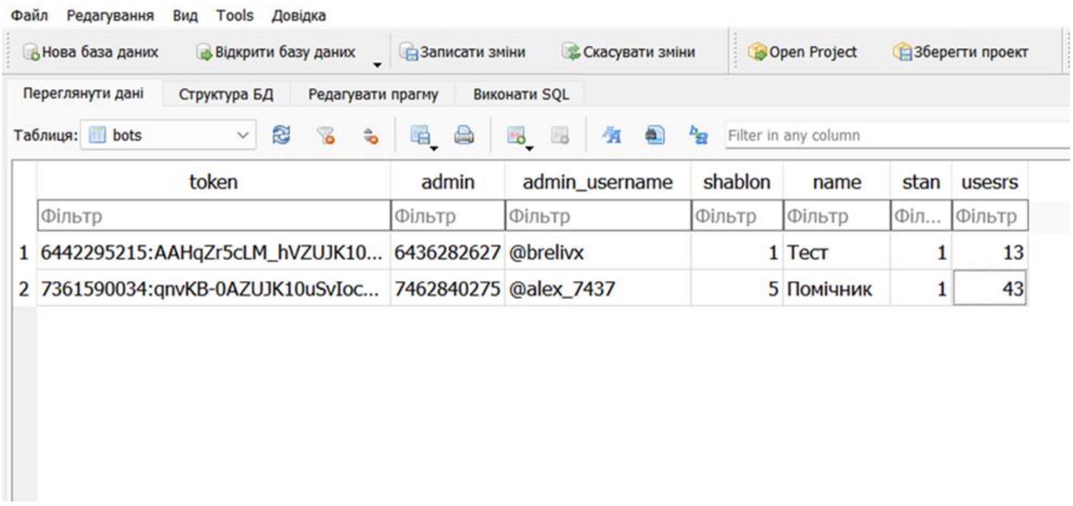

Рисунок 3.2 – База даних платформи

Після підтвердження дійсності токену, система ініціює процес перевірки наявності токену у базі даних. Цей крок є важливим для забезпечення синхронізації даних бота з існуючою інформацією та для уникнення дублювання записів. У випадку, коли токен вже присутній у базі, система перенаправляє користувача на сторінку з інформацією про бота.

Якщо ж токен відсутній у базі даних, це означає, що бот є новим або не був раніше зареєстрований. В такому випадку система автоматично додає токен до бази даних, що дозволяє інтегрувати бота з іншими компонентами системи та забезпечити його готовність до взаємодії з користувачами.

3.3 Розробка модуля перезапуску всіх ботів з оновленою базою даних

Основною задачею модуля є перезапуск всіх ботів з оновленою базою даних, що дозволяє інтегрувати нового бота в систему. Для цього було розроблено функцію BotReloader, який ініціалізує процес перезапуску. Коли новий токен додається до бази даних, функція BotReloader активується для оновлення системи, що забезпечує інтеграцію нових ботів та їх готовність до роботи з користувачами. Перезапуск виконується за допомогою модуля subprocess [19], який запускає новий процес в операційній системі серверу.

Після цього, файл перезавантажується з оновленою базою даних, готуючись приймати нові повідомлення від користувачів для всіх наявних токенів у базі даних, маючи у наявності першу основну команду /start для перевірки користувачем що бот успішно приєднаний.

3.4 Розробка фронтенду вебплатформи

На головній сторінці вебплатформи реалізовано захоплюючий анімований фон, який створює враження глибини та руху. Ця анімація створюється за допомогою HTML та CSS. Відповідний CSS-код встановлює анімаційний ефект, розташування та розміри анімованих елементів.

Зображення на фоні створюється за допомогою декількох круглих елементів з фоновим кольором, що змінюються в часі. Параметри анімації, такі як затримка, розміри та розташування, налаштовуються за допомогою CSS. Кожен елемент має свій власний час затримки, що створює враження руху та глибини (рис. 3.3).

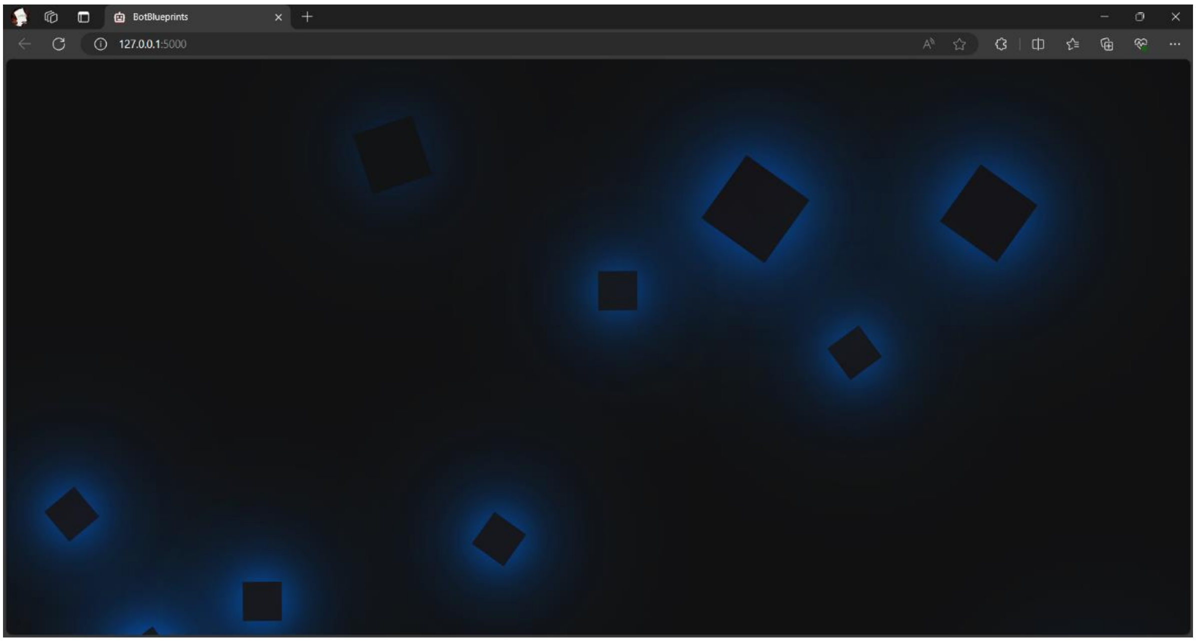

Рисунок 3.3 – Анімований фон платформи

Під час розробки, було використано адаптивний дизайн, щоб забезпечити коректне відображення вебплатформи на різних пристроях, включаючи мобільні. Для цього використовується CSS, який задає різні стилі для елементів в залежності від ширини екрану. Наприклад, при ширині екрану менше 600 пікселів, розмір контейнера, шапки і логотипу зменшується, щоб забезпечити оптимальне відображення на маленьких екранах.

Таким чином, користувачі можуть зручно користуватися веб платформою як на комп'ютері, так і на мобільних пристроях.

3.5 Висновки

У цьому розділі було здійснено комплексний підхід до розробки програмних засобів вебплатформи для створення телеграм ботів, використовуючи конструктор сценаріїв. В ході роботи було проведено варіантний аналіз та обґрунтовано вибір технологій, які найкраще відповідають потребам проекту.

Було вирішено використовувати Python з фреймворком Flask для бекенду, що забезпечує гнучкість та легкість у розробці веб додатків. SQLite3 було обрано як систему управління базами даних через її легкість та ефективність. Для фронтенду було вибрано HTML, CSS та JavaScript, що дозволяє створювати інтуїтивно зрозумілі та адаптивні користувацькі інтерфейси. Aiogram3, SQLite3 та asyncio [20] були обрані для створення ботів, забезпечуючи асинхронну роботу та високу продуктивність.

Продемонстровано розробку модулю перевірки токену був розроблений для забезпечення автентичності телеграм ботів. Процес включає перевірку токену через Telegram API та його додавання до бази даних, що є критично важливим для безпеки та стабільності роботи ботів.

Також було здійснено розробку модуля перезапуску ботів з оновленою базою даних, що дозволяє забезпечити актуальність даних та неперервну роботу ботів. Цей модуль використовує сучасні методи для моніторингу змін у базі даних та автоматичного оновлення конфігурацій ботів. Коли в базі даних відбуваються зміни, такі як оновлення інформації про користувачів або нові налаштування бота, модуль перезапуску миттєво вносить ці зміни, перезапускаючи відповідні процеси ботів.

Використання модуля subprocess для перезапуску процесів забезпечує ефективне управління ресурсами та зменшує час простою. Subprocess дозволяє запускати нові процеси та управляти ними безпосередньо з основного додатку, що спрощує виконання складних задач, таких як оновлення ботів у реальному часі. Завдяки цьому модулю, платформа може безперебійно працювати з великою кількістю ботів, швидко реагуючи на зміни та підтримуючи їхню актуальність.

Окрім того, було приділено значну увагу розробці фронтенду вебплатформи. Він був створений з акцентом на анімацію та адаптивність, що забезпечує сучасний вигляд і зручність користування платформою на різних пристроях. Використання CSS для створення анімованого фону додає динамічності та естетичної привабливості інтерфейсу. Анімовані елементи не лише покращують візуальне сприйняття, але й роблять взаємодію з платформою більш приємною та інтуїтивно зрозумілою для користувачів.

Таким чином, розробка модуля перезапуску ботів та фронтенду вебплатформи значно покращила функціональність та користувацький досвід. Актуальність даних, неперервна робота ботів, анімація, адаптивність та зручність використання – всі ці фактори роблять платформу потужним інструментом для автоматизації взаємодії з клієнтами та оптимізації бізнес-процесів.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Тестування програмних засобів вебплатформи для створення телеграм ботів є ключовим етапом, що забезпечує виявлення помилок та неполадок перед запуском системи в експлуатацію. Основна мета тестування полягає у перевірці відповідності програмного продукту встановленим вимогам та забезпеченні його якості та надійності.

Перед початком тестування встановлюється тестова версія вебплатформи на сервері з відповідною операційною системою, що дозволяє імітувати реальні умови використання. Важливою частиною підготовки до тестування є розробка документації, яка включає в себе детальні специфікації вимог та план тестування.

Функціональне тестування проводиться для перевірки коректності роботи всіх модулів вебплатформи, включаючи модулі перевірки токену, додавання токену до бази даних, перезапуску ботів з оновленою базою даних, а також фронтенду. Тести охоплюють перевірку введення та виведення даних, реакцію на некоректні вхідні дані та відповідність роботи модулів бізнес-логіці вебплатформи.

Тестування відмов спрямоване на виявлення можливих дефектів у програмному забезпеченні, аналізуючи його здатність відновлювати роботу після помилок або збоїв. Це включає тестування реакції системи на неправильні дії користувачів, нестандартні ситуації та перевірку стійкості системи при відсутності зв’язку з зовнішніми сервісами.

Після функціонального тестування та тестування відмов, наступним кроком є оцінка продуктивності програмного забезпечення. Цей етап включає в себе тести швидкодії, які вимірюють час відповіді програми на запити користувачів, а також здатність системи ефективно обробляти велику кількість операцій.

Тестування сумісності проводиться для переконання, що програмне забезпечення працює коректно на різних операційних системах та у різних середовищах. Це дозволяє виявити потенційні конфлікти з іншими програмами та забезпечити стабільну роботу вебплатформи незалежно від апаратних та програмних конфігурацій користувачів. Таке тестування є ключовим для забезпечення високої адаптивності та доступності сервісу для широкого кола користувачів.

Першим етапом тестування програмного забезпечення вебплатформи для створення телеграм ботів є перевірка реакції системи на введення невірного токену бота. Цей процес важливий для забезпечення безпеки та стабільності роботи платформи. Під час тестування, якщо користувач введе токен, який не відповідає формату, визначеному для токенів телеграм ботів, програма має ідентифікувати цю помилку та надати користувачу зрозуміле повідомлення про помилку з інструкціями щодо виправлення.

На рисунку 4.1 демонструється інтерфейс платформи, який відображає повідомлення про невірний токен та пропонує користувачеві ввести правильний токен для продовження роботи з платформою. Це забезпечує, що користувачі не зіткнуться з несподіваними помилками під час роботи з системою та допомагає уникнути потенційних збоїв у роботі ботів через неправильну конфігурацію.

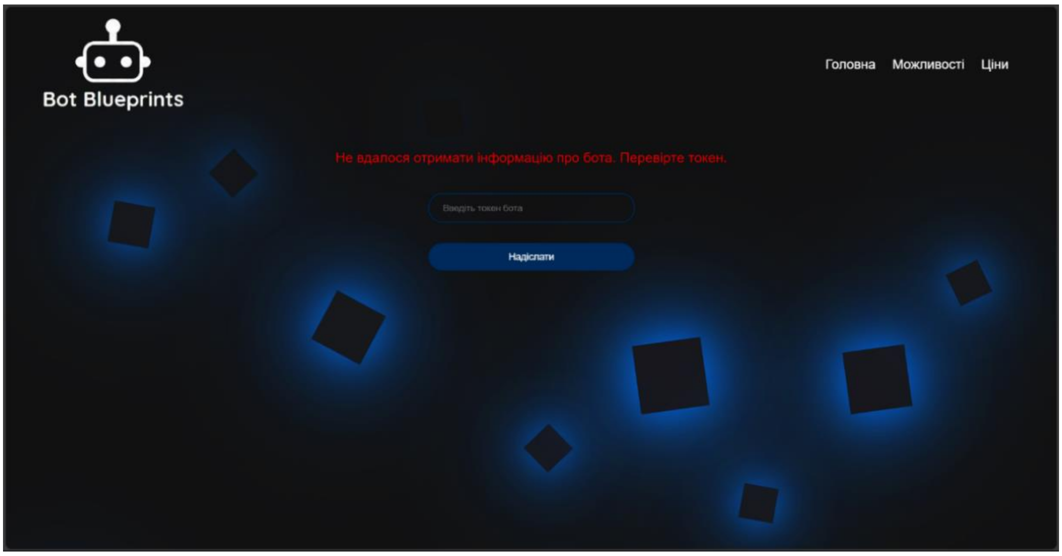

Рисунок 4.1 – Відправка невірно формату токену

У процесі тестування програмного забезпечення вебплатформи для створення телеграм ботів було проведено ряд перевірок, спрямованих на забезпечення стабільності та надійності системи. Після тестування на введення невірного формату токену, було проведено додаткове тестування, щоб перевірити реакцію системи на відсутність вхідних даних. Цей крок був важливим для забезпечення того, щоб у випадку, коли користувач не вводить токен, система коректно інформувала про необхідність його введення та надавала відповідні інструкції для продовження роботи. Результати цього тестування (рис. 4.2) показали, що програмне забезпечення ефективно обробляє сценарії без вхідних даних, надаючи користувачам чіткі повідомлення та керівництво для вирішення проблеми.

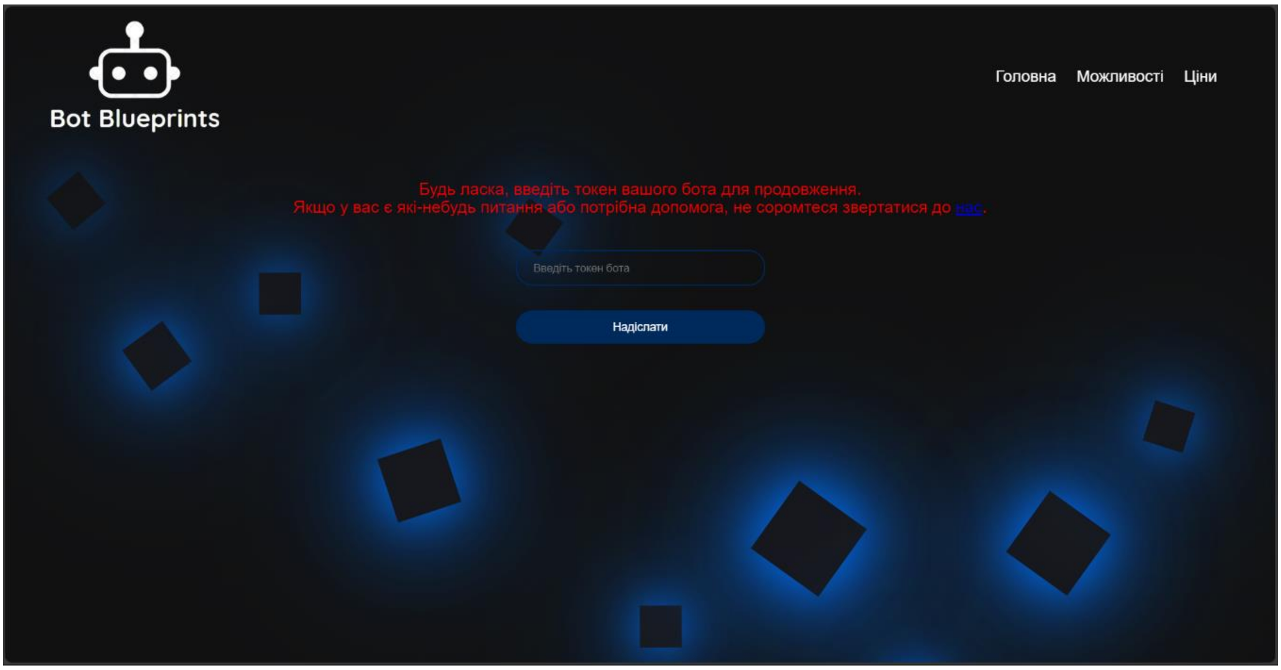

Рисунок 4.2 – Спроба надіслати без вхідних даних

Після завершення функціонального тестування, яке перевіряло правильність обробки вхідних даних, було ініційовано тестування на реакцію програми до надсилання пошкодженого токену. На рисунку 4.3 наведена реакція системи під час тестування на реакцію програми до введення пошкодженого токену, система має виявити невірний токен та надати користувачеві відповідне повідомлення. Це допомагає забезпечити безпеку та стабільність роботи платформи, а також надає користувачам зрозумілу інструкцію щодо виправлення помилки. Результати тестування свідчать про те, що платформа успішно обробляє такі випадки. У контексті тестування на введення токену, який вже використовується в іншій системі, платформа виявила такий випадок та проінформувала користувача про конфлікт. Система повинна надавати чітке повідомлення про те, що введений токен уже активовано в іншому середовищі, та пропонувати користувачу ввести інший токен для продовження роботи з платформою. Такий підхід дозволяє уникнути помилок взаємодії між різними системами та забезпечує коректну реєстрацію нових ботів.

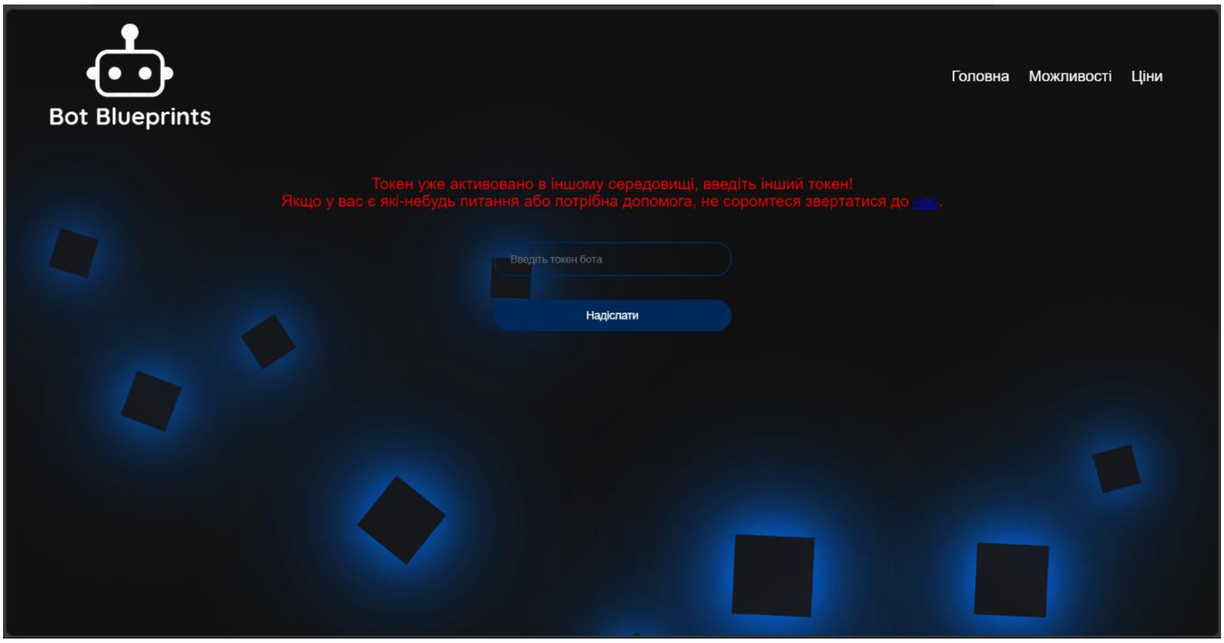

Рисунок 4.3 – Результат вводу токену який вже використовується

Після перевірки введення недійсних або пошкоджених даних у програмне забезпечення, наступним етапом тестування стало введення коректного токену. Цей процес був важливим для підтвердження, що система правильно обробляє валідні дані та здатна ефективно їх обробляти. Після успішного введення токену було проведено аналіз реакції програмного забезпечення, результати якого представлені на рисунку 4.4.

Крім того, було виконано тестування сумісності програмного забезпечення з різними браузерами та на пристроях з різними розмірами екранів, включаючи мобільні телефони, результати представлені на рисунку 4.5.

Це дозволило переконатися, що програмне забезпечення працює коректно не тільки на комп'ютерах з різними операційними системами, але й на різноманітних мобільних пристроях, забезпечуючи гнучкість та доступність вебплатформи для всіх користувачів. Результати тестування підтвердили високу адаптивність та оптимізацію програмного забезпечення, що свідчить про його ефективність у різних умовах використання.

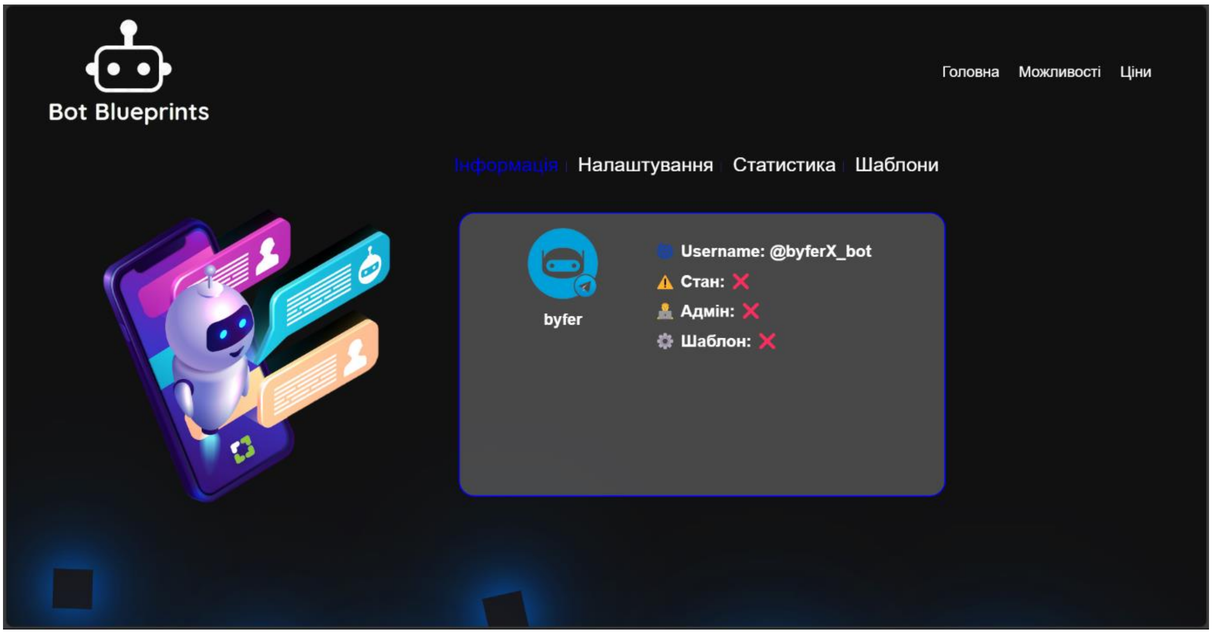

Рисунок 4.4 – Результат успішної відправки токену

Рисунок 4.5 – Результат успішного тестування роботи платформи на мобільному пристрої

У ході розробки платформи особливу увагу було приділено забезпеченню сумісності та коректного відображення головної сторінки у різних браузерах. Тестування відображення головної сторінки було проведено у таких популярних браузерах, як Google Chrome, Microsoft Edge, та Safari. В результаті тестування було перевірено:

- Google Chrome: У цьому браузері головна сторінка платформи відображається коректно, з чітким та зрозумілим інтерфейсом, швидким завантаженням та правильною роботою всіх інтерактивних елементів. Перевірено роботу токену та верифікації, а також відображення вітального екрану та робочого середовища (рис. 4.6).

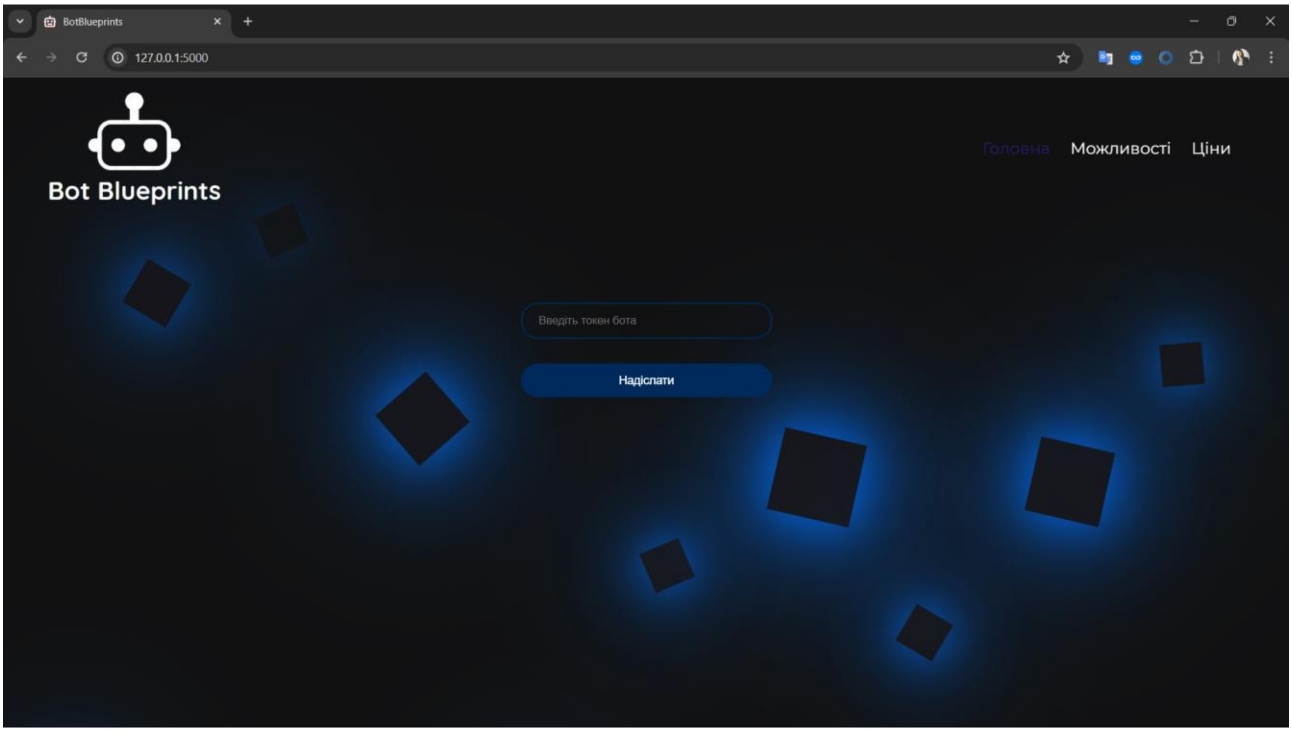

Рисунок 4.6 – Результат успішного тестування роботи платформи у браузері Google Chrome

- Microsoft Edge: Головна сторінка також працює стабільно, відображаючи всі елементи інтерфейсу без збоїв. У процесі тестування було підтверджено, що перевірка токену та перехід на вітальний екран відбуваються без проблем. Інтерактивні функції працюють так само ефективно, як і в Google Chrome (рис. 4.7).

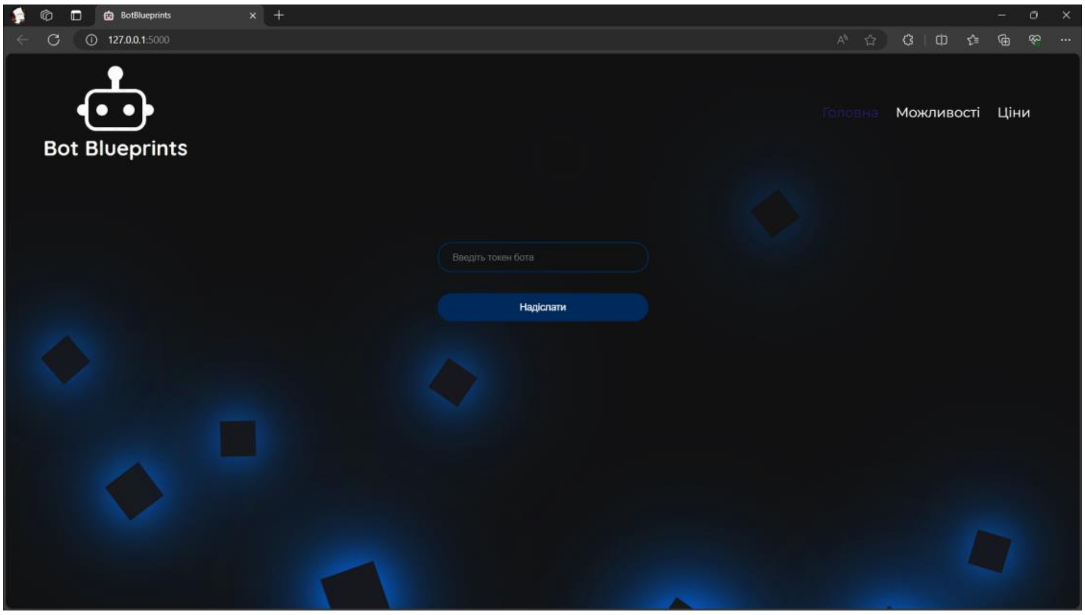

Рисунок 4.7 – Результат успішного тестування роботи платформи у браузері Microsoft Edge

- Safari: Тестування у Safari показало, що платформа сумісна з цим браузером. Головна сторінка завантажується швидко, всі елементи коректно відображаються, включаючи форми для введення токену та навігаційні елементи. Процес верифікації токену та доступ до робочого середовища також працюють без зауважень (рис. 4.7).

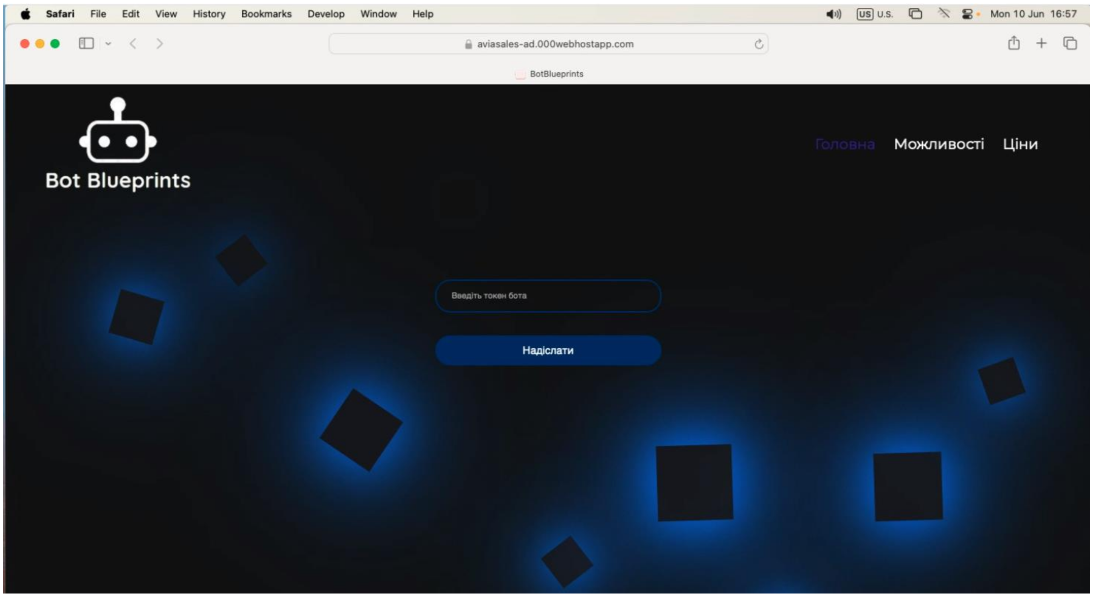

Рисунок 4.7 – Результат успішного тестування роботи платформи у браузері Microsoft Edge

Таким чином, завдяки ретельному тестуванню у різних браузерах, вдалося забезпечити високу якість користувацького досвіду незалежно від обраного браузера. Користувачі з різними вподобаннями та налаштуваннями браузерів можуть комфортно та ефективно працювати з платформою для створення та управління телеграм-ботами. Це гарантує, що платформа є доступною та зручною для широкого кола користувачів, що підвищує її загальну привабливість та функціональність.

Після завершення первинного тестування, яке забезпечило впевненість у коректній обробці вхідних даних, було ініційовано тестування роботи телеграм-бота. Цей етап тестування був критично важливим для перевірки того, як бот реагує на команди користувачів у реальних умовах. Тестування включало кілька ключових етапів, починаючи з відправлення команди /start боту та аналізу його відповіді. Мета цього тестування полягала в тому, щоб переконатися, що бот правильно інтерпретує команди, надає відповідні інструкції або інформацію, та взаємодіє з користувачами у відповідності до заданих сценаріїв.

На рисунку 4.8 представлено інтерфейс телеграм-бота з відповіддю на команду /start. Це початкова команда, яка запускає взаємодію між ботом та користувачем. В результаті тестування було підтверджено, що бот коректно ініціює взаємодію, надаючи вітальне повідомлення, яке містить основну інформацію про можливості та функції бота. Користувачі отримують чіткі інструкції щодо подальших дій, що значно полегшує їх взаємодію з ботом.

Також було проведено тестування обробки некоректних команд та неправильних даних, щоб переконатися, що бот здатен адекватно реагувати на помилки користувачів. Бот успішно надавав корисні підказки та інструкції щодо коректного введення команд, що підтверджує його здатність до адаптації та навчання користувачів.

Після завершення всіх тестувань, результати показали високу стабільність та надійність роботи телеграм-бота. Всі основні функції бота працювали без збоїв, а користувачі отримували необхідну інформацію швидко та точно. Таким чином, платформа для створення та управління телеграм-ботами довела свою ефективність та готовність до використання у реальних умовах.

Результати тестування підтвердили, що бот коректно ініціює взаємодію з користувачем, надаючи необхідну інформацію та вітаючи користувача у системі. Це свідчить про високу якість розробки та налаштування бота, що забезпечує користувачам зручний та інтуїтивно зрозумілий інтерфейс для взаємодії з автоматизованою системою.

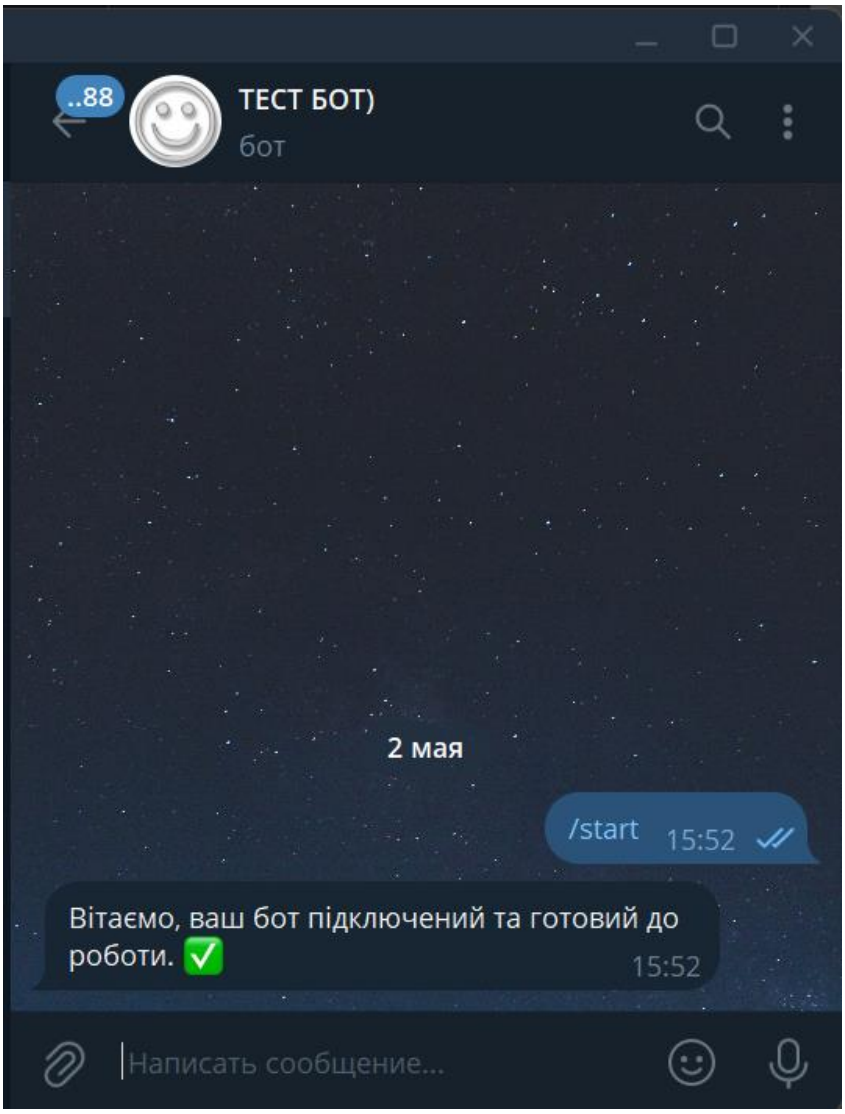

Рисунок 4.8 – Результати тестування боту

Після завершення первинного тестування, яке забезпечило впевненість у коректній обробці вхідних даних, було ініційовано тестування реакції телеграм бота на специфічні команди. Однією з ключових перевірок була реакція бота на команду /admin, яка призначена для власника боту. Цей тест мав на меті переконатися, що бот відрізняє власника від звичайних користувачів та надає доступ до адміністративних функцій лише авторизованим особам.

На рисунку 4.7 представлено відповідь бота на команду /admin, введену власником. Результати тестування показали, що бот коректно ідентифікує власника та надає йому відповідні адміністративні права. Коли та ж команда була введена звичайним користувачем, бот не надавав доступ до адміністративних функцій, що свідчить про його надійність та безпеку використання.

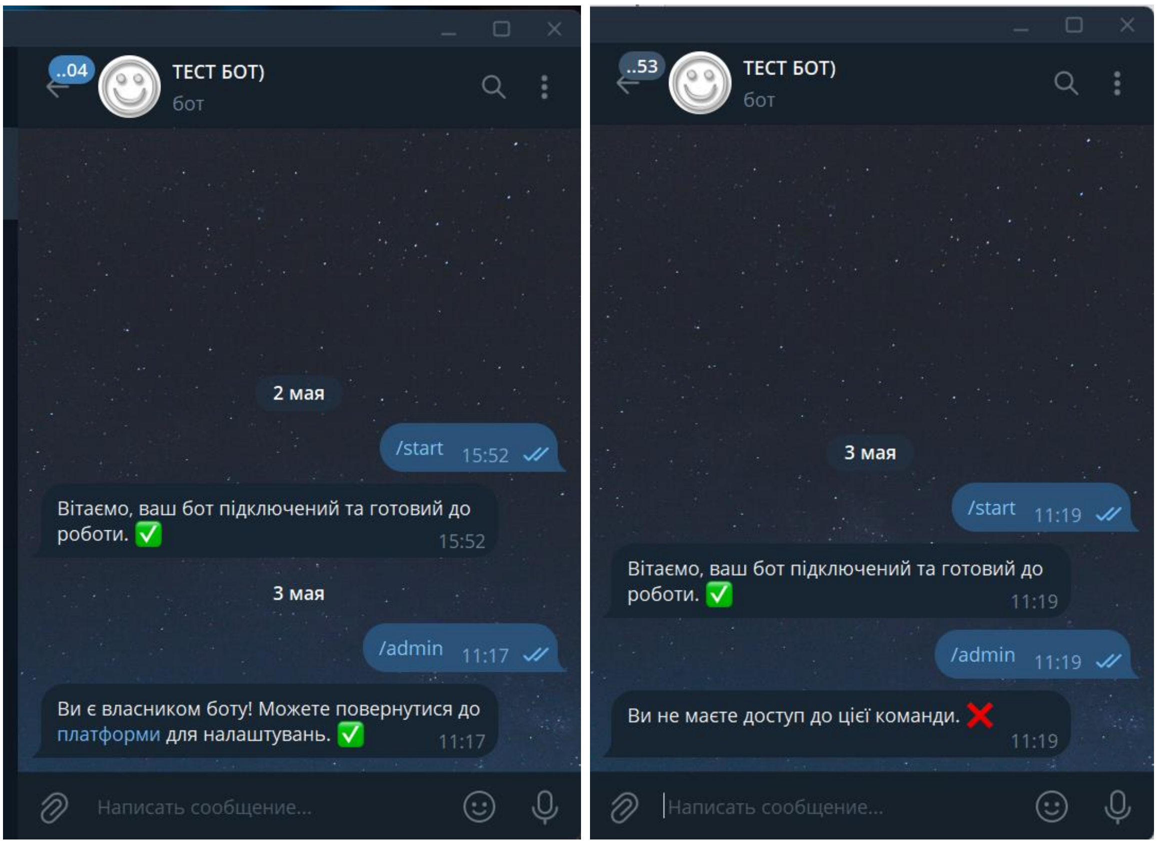

Рисунок 4.7 – Результати тестування команди /admin

В ході тестування програмного забезпечення було виконано процес експорту даних з бази даних боту за допомогою інструменту DB Browser for

SQLite. Цей процес включав перевірку коректності експортованих даних у формат .db, що є стандартним форматом для баз даних SQLite. Результати експорту були ретельно проаналізовані для забезпечення точності та цілісності даних. На рисунку 4.8 представлено інтерфейс DB Browser for SQLite з результатами експорту, демонструючи, що всі дані були успішно записані та готові до подальшого використання.

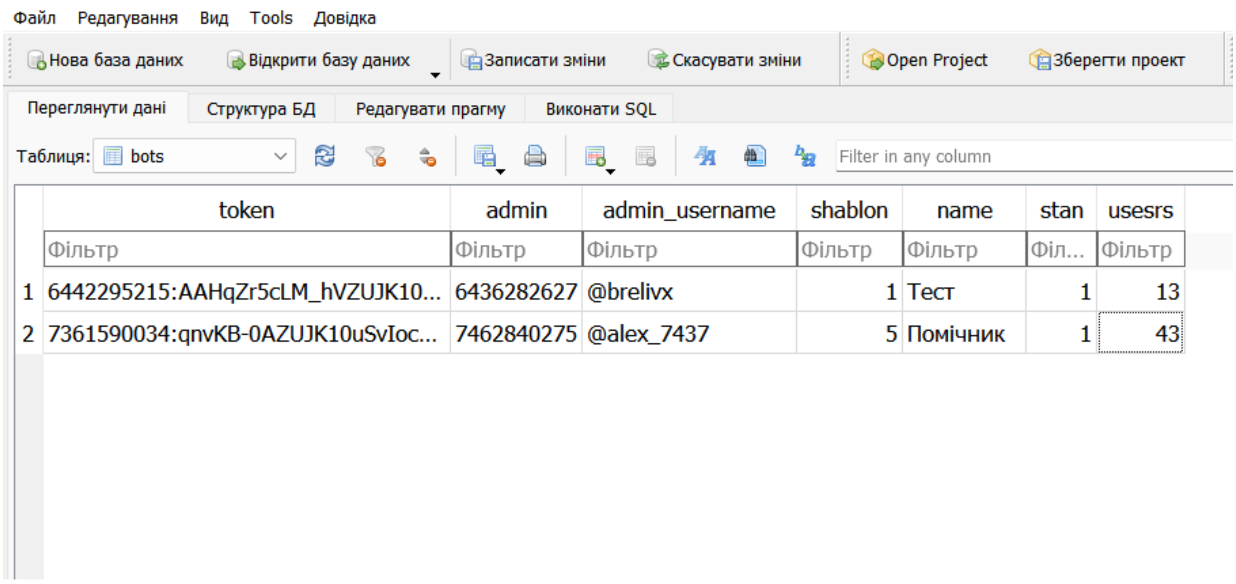

Рисунок 4.8 – Результат тестування експорту даних

Додатково до тестування швидкості відгуку системи на дії користувача було проведено аналіз часу, необхідного для автоматичного створення телеграм ботів через веб платформу. Під час тестування виявлено, що система здатна відповідати на взаємодію з користувачем з швидкістю, що не перевищує 0.3 секунди. Це означає, що користувачі отримують миттєвий зворотний зв'язок під час використання інтерфейсу програмного забезпечення.

У той же час, час, необхідний для відповіді користувачам ботів, складає не більше ніж 0.5 секунд результати тестування можна переглянути у терміналі серверу на рисунку 4.9. Це вказує на ефективність та швидкість обробки даних програмним забезпеченням під час виконання.

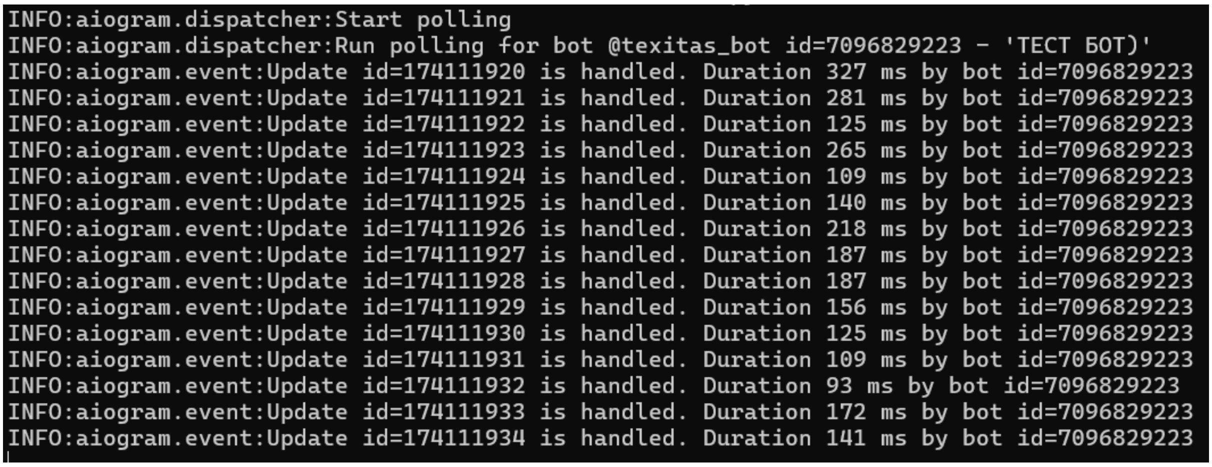

Рисунок 4.9 – Результат тестування швидкості

4.2 Розробка інструкції користувача

Керівництво користувача вебплатформи для створення телеграм-ботів включає визначення технічних вимог для ефективної роботи з конструктором сценаріїв та використання готових шаблонів. Наведено деталі щодо мінімальних та оптимальних конфігурацій, які забезпечать безперебійну роботу, в таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Вимоги до конфігурування	**Рекомендовано**	**Мінімально**
Браузер	Google Chrome, Microsoft Edge, Safari	-
Інтернет-з’єднання	5 Мбіт/с і вище	1,5 Мбіт/с
Розмір екрану	1600 * 900	300 * 500

Після входу на веб платформу користувач вводить токен свого бота, що ініціює процес перевірки токену. У разі успішної верифікації, користувача переадресовує на вітальний екран інформації бота, де він може ознайомитися з деталями свого телеграм-бота (рис. 4.9). Після цього, користувач очікує на завантаження робочого середовища, що дозволяє використовувати реалізований функціонал для створення та управління сценаріями бота.

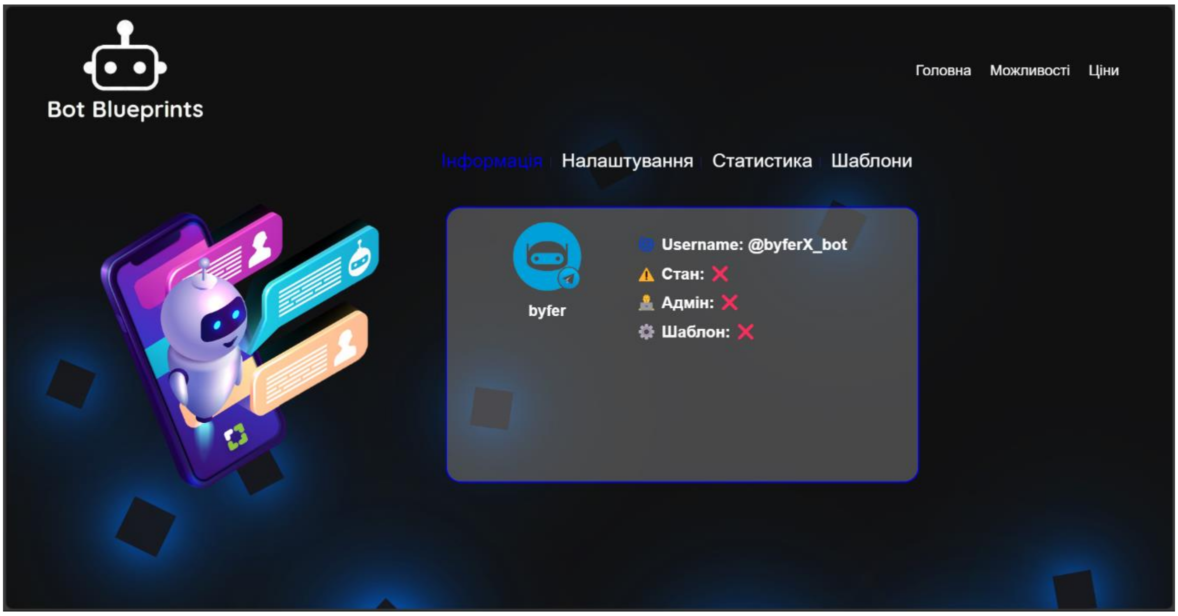

Рисунок 4.9 – Сторінка інформації про бота

Першим кроком у взаємодії з веб платформою є створення або імпорт сценарію для бота. Користувач може вибрати готовий шаблон або створити новий сценарій за допомогою конструктора. Після створення сценарію запускається модуль перевірки на відповідність заданим параметрам, і у разі виявлення помилок, користувач отримує повідомлення з рекомендаціями щодо їх виправлення.

Далі користувач може перейти до меню ‘Можливості’ (рис. 2.6) для отримання інструкцій, ‘Про програму’ для інформації про веб-платформу, або розпочати тестування бота, натиснувши відповідну кнопку. Після тестування сценарію, на екрані з’явиться інформація про його роботу та взаємодію з користувачем.

Після завершення тестування сценарію користувач також має можливість перейти до розділу ‘Статистика’ для отримання детальної інформації про активність бота (рис. 4.10). У цьому розділі відображаються ключові показники, що дозволяють оцінити ефективність роботи бота та його взаємодію з користувачами.

На сторінці ‘Статистика’ представлена наступна інформація:

- Користувачів всього: Загальна кількість користувачів, які коли-небудь взаємодіяли з ботом. Цей показник допомагає зрозуміти масштаб використання бота.
- Заблокували бота: Кількість користувачів, які заблокували бота. Цей показник важливий для оцінки рівня задоволеності користувачів та виявлення потенційних проблем у сценаріях взаємодії.
- Активні користувачі: Кількість користувачів, які активно взаємодіяли з ботом протягом останнього часу. Це дозволяє визначити рівень залученості користувачів.
- Нові за сьогодні: Кількість нових користувачів, які почали взаємодіяти з ботом протягом поточного дня. Цей показник допомагає відстежувати зростання популярності бота.

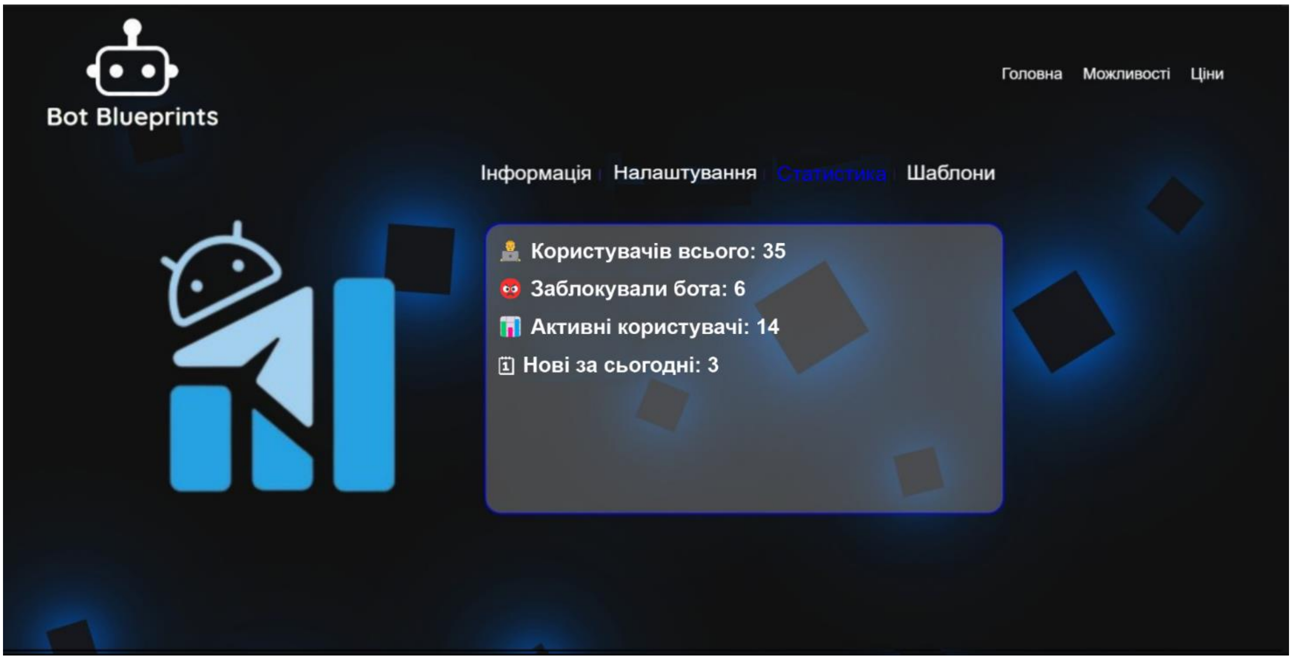

Рисунок 4.10 – Тестування коректного відображення сторінки “Статистика”

Всі інтерактивні елементи сторінки працюють належним чином, забезпечуючи користувачам зручний доступ до інформації.

Завдяки цьому користувачі можуть отримати повне уявлення про ефективність та популярність свого бота, а також вчасно виявити та виправити можливі проблеми. Статистика допомагає краще розуміти поведінку користувачів та адаптувати сценарії взаємодії для підвищення їхньої задоволеності.

Наступним важливим розділом для тестування є сторінка 'Шаблони' (рис. 4.11). Зараз на цій сторінці представлений один шаблон бота – 'Зворотній зв'язок'. Цей шаблон дозволяє користувачам легко створити бота для збору відгуків від клієнтів, що є важливим інструментом для покращення якості послуг та продукції.

Під час тестування сторінки 'Шаблони' було перевірено коректність відображення та функціональність існуючого шаблону. Було здійснено тестові запуски бота, створеного на основі шаблону 'Зворотній зв'язок', щоб переконатися, що він правильно збирає та обробляє відгуки користувачів. Тестування також включало перевірку можливості налаштування шаблону, щоб користувачі могли адаптувати бота під свої специфічні потреби.

Було також проведено тестування відображення сторінки 'Шаблони' у різних браузерах (Google Chrome, Microsoft Edge, Safari), щоб гарантувати її сумісність та коректне відображення у всіх популярних веб-браузерах. Результати тестування підтвердили, що сторінка функціонує належним чином, забезпечуючи зручний доступ до шаблону та його налаштувань.

У майбутньому планується додати нові шаблони для розширення можливостей платформи. Це можуть бути шаблони для різних типів бізнесу та сценаріїв взаємодії, такі як автоматизація відповіді на часто задавані питання, підтримка клієнтів, організація опитувань та багато іншого. Додавання нових шаблонів забезпечить користувачам ще більше варіантів для швидкого та ефективного створення телеграм-ботів, адаптованих до їхніх конкретних потреб.

Таким чином, тестування сторінки 'Шаблони' підтвердило її функціональність та готовність до використання. З додаванням нових шаблонів платформа стане ще більш потужним інструментом для автоматизації та оптимізації взаємодії з клієнтами.

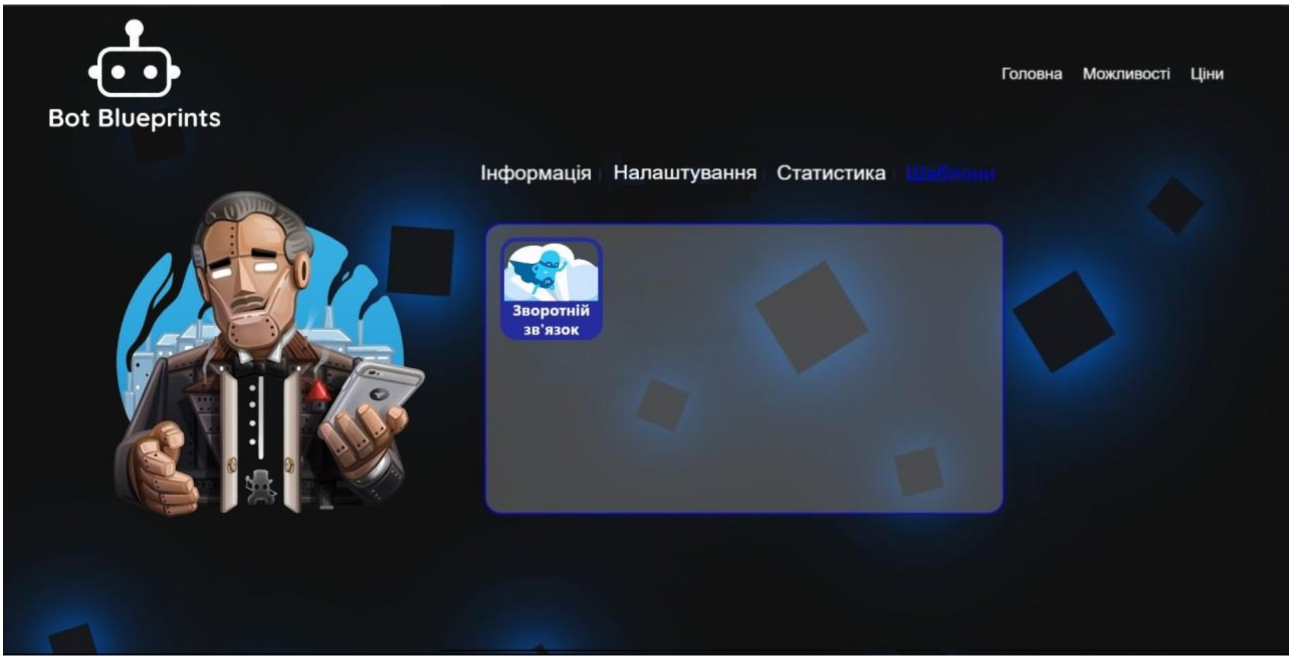

Рисунок 4.11 – Тестування коректного відображення сторінки “Шаблони”

4.3 Висновки

У четвертому розділі було проведено всебічне тестування програмних модулів вебплатформи для створення телеграм-ботів. Особлива увага була приділена перевірці реакції платформи на різноманітні сценарії взаємодії з користувачем. Тестування включало введення токену бота, перевірку його валідності, а також обробку вхідних даних з метою забезпечення коректної роботи всієї системи. Кожен етап взаємодії був ретельно протестований для виявлення можливих помилок та недоліків у роботі платформи.

Було розроблено декілька сценаріїв тестування, включаючи введення правильного та неправильного токену, щоб перевірити, як платформа обробляє ці дані та які повідомлення про помилки вона генерує. Також було протестовано поведінку системи при повторному введенні токену та взаємодії з базою даних для підтвердження, що система коректно оновлює інформацію про бота. Особливу увагу було приділено перевірці безпеки даних та захисту від несанкціонованого доступу, щоб забезпечити надійність роботи платформи у різних умовах.

Крім технічного тестування, було також розроблено детальну інструкцію користувача, яка допомагає новачкам швидко освоїтись та ефективно працювати з конструктором сценаріїв та шаблонами. Інструкція містить покрокові пояснення щодо створення та налаштування телеграм-ботів, включаючи вибір шаблонів, налаштування поведінки бота, та тестування створених сценаріїв. Вона забезпечує користувачам зручний та зрозумілий спосіб освоєння функціоналу платформи, що значно спрощує процес створення та управління телеграм-ботами.

Додатково в інструкції наведено приклади найпоширеніших сценаріїв використання платформи, що допомагають користувачам краще зрозуміти її можливості та швидше досягти бажаних результатів. Включення таких прикладів дозволяє користувачам не лише ознайомитися з основними функціями платформи, але й отримати уявлення про те, як ці функції можуть бути застосовані у реальних бізнес-сценаріях.

Таким чином, тестування програмних модулів та розробка інструкції користувача забезпечили високу якість та зручність використання вебплатформи для створення телеграм-ботів. Це дозволяє користувачам легко та ефективно створювати, налаштовувати та управляти телеграм-ботами, використовуючи всі можливості, які надає платформа. Завдяки цьому, платформа є надійним та потужним інструментом для автоматизації взаємодії з клієнтами та оптимізації бізнес-процесів.

ВИСНОВКИ

У бакалаврській дипломній роботі були розроблені модулі програмного забезпечення для вебплатформи, що дозволяє створювати телеграм-ботів з використанням конструктора сценаріїв. Для розробки використовувалося середовище програмування Sublime Text. Реалізація курсового проекту відповідає методичним вказівкам [21-24].

Під час виконання проекту було проведено аналіз сучасного стану технологій створення телеграм-ботів. Були розглянуті основні аналоги програмного продукту, виявлені їхні недоліки та порівняно з власним розробленим продуктом. На основі цього порівняння були сформульовані основні завдання курсового проекту.

Під час аналізу технологій розробки було обґрунтовано вибір мов програмування Python для бекенду з використанням Flask, request, sqlite3, для фронтенду - HTML, CSS, JavaScript, а для ботів - Python з використанням aiogram3, asyncio, sqlite3, subprocess.

У курсовому проєкті було зроблено наступне:

- розроблено модуль перевірки токену та додання його до бази даних;
- розроблено модуль перезапуску всіх ботів з оновленою базою даних;
- розроблено фронтенд вебплатформи.

Було розроблено схеми загального алгоритму роботи вебплатформи та модуля перевірки токену. Також було розроблено модель системи з використанням UML діаграм.

Тестування вебплатформи довело повну працездатність даного програмного продукту та відповідність поставленому технічному завданню. Також було розроблено інструкцію користувача.

ЛІТЕРАТУРА

1. Spacelab [Електронний ресурс] – Режим доступу до ресурсу: https://spacelab.ua/articles/yak-stvoriti-telegram-bota-na-python/ (дата звернення: 21.04.2024)
2. ITForce [Електронний ресурс] – Режим доступу до ресурсу: https://itforce.ua/blog/polezniye-telegram-boty/ (дата звернення: 21.04.2024)
3. Брильянт І. А., Чехместрук Р. Ю. Проектування та розробка вебплатформи для автоматизованого створення телеграм-ботів за допомогою інструменту конструювання сценаріїв // Стан, досягнення та перспективи інформаційних систем і технологій: матеріали XXIV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів, Одеса, 18-19 квітня 2024 р. Одеса: Видавництво ОНТУ, 2024. 498 с.
4. BotFather [Електронний ресурс] – Режим доступу до ресурсу: https://t.me/BotFather (дата звернення: 21.04.2024)
5. PuzzleBot [Електронний ресурс] – Режим доступу до ресурсу: https://puzzlebot.top/ (дата звернення: 21.04.2024)
6. SendPulse [Електронний ресурс] – Режим доступу до ресурсу: https://sendpulse.ua/ (дата звернення: 21.04.2024)
7. Gerabot [Електронний ресурс] – Режим доступу до ресурсу: https://gerabot.com/ (дата звернення: 21.04.2024)
8. BotMother [Електронний ресурс] – Режим доступу до ресурсу: https://app.botmother.com/ (дата звернення: 21.04.2024)
9. BotHelp [Електронний ресурс] – Режим доступу до ресурсу: https://bothelp.io/ (дата звернення: 21.04.2024)
10. Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/ (дата звернення: 22.04.2024)
11. Aiogram [Електронний ресурс] – Режим доступу до ресурсу: https://docs.aiogram.dev/ (дата звернення: 22.04.2024)

12. UML [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Unified_Modeling_Language (дата звернення: 22.04.2024)

13. Visio [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/uk-ua/microsoft-365/visio/flowchart-software (дата звернення: 22.04.2024)

14. Flask [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Flask (дата звернення: 23.04.2024)

15. Sqlite3 [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/uk/3.9/library/sqlite3.html (дата звернення: 23.04.2024)

16. Telegram API [Електронний ресурс] – Режим доступу до ресурсу: https://core.telegram.org/ (дата звернення: 23.04.2024)

17. Webhook [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Webhook (дата звернення: 24.04.2024)

18. Long polling [Електронний ресурс] – Режим доступу до ресурсу: https://uk.javascript.info/long-polling (дата звернення: 24.04.2024)

19. Subprocess [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/uk/3/library/subprocess.html (дата звернення: 24.04.2024)

20. Asyncio [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/uk/3/library/asyncio.html (дата звернення: 24.04.2024)

21. Ткаченко П. І. Тестування програмного забезпечення: методи та інструменти. Київ: ІТ Академія, 2020. 412 с. 23.

22. Бондаренко Н. А. Функціональне тестування: теорія і практика. Одеса: Південь, 2021. 385 с. 24.

23. Петренко О. Ю. Тестування відмов: новітні підходи та методи. Вінниця: Видавничий Дім, 2023. 290 с. 25.

24. Іванова Л. М. Тестування сумісності: посібник для тестувальників. Дніпро: Сучасні Технології, 2022. 305 с.

ДОДАТКИ

59

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
________О.Н. Романюк
«12» березня 2024 р.

Технічне завдання

на бакалаврську дипломну роботу «Проектування та розробка вебплатформи для автоматизованого створення телеграм-ботів за допомогою інструментів конструювання сценаріїв»

за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
________ к.т.н., доц. каф. ПЗ Чехместрук Р.Ю.
«12» березня 2024 р.

Виконав:
________ студент гр. 2ПІ-20б І. А. Брильянт
«12» березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Проектування та розробка вебплатформи для автоматизованого створення телеграм-ботів за допомогою інструментів конструювання сценаріїв».

Галузь застосування – соціальні послуги.

2. Підстава для розробки

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки

Метою дослідження є автоматизація створення телеграм-ботів, що спрощує процес їх розробки та налаштування, забезпечуючи високу доступність та адаптивність до індивідуальних потреб користувачів.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. Брильянт І. А., Чехместрук Р. Ю. Проектування та розробка вебплатформи для автоматизованого створення телеграм-ботів за допомогою інструменту конструювання сценаріїв // Стан, досягнення та перспективи інформаційних систем і технологій: матеріали XXIV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів, Одеса, 18-19 квітня 2024 р. Одеса: Видавництво ОНТУ, 2024. 498 с.

5. Технічні вимоги

Вихідні дані до роботи: середовище розробки sublime text, система управління базами даних – SQLite; мова запитів – SQL; інструменти для розробки веб-інтерфейсу: HTML, CSS, JavaScript; мова програмування: Python, бібліотеки: requests, flask, aiogram.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БДР.
2. Технічне завдання.
3. Лістинги програми

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ

9. Стадії та етапи розробки:

№ з\п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз завдання і вибір методу вирішення поставленої задачі дослідження	14.03.24 – 29.03.24
2	Вибір архітектури програмного компоненту	01.04.24 – 09.04.24
3	Розробка структури інтерфейсу	10.04.24 – 19.04.24
4	Вибір середовища та мови розробки	20.04.24 – 24.04.24
5	Розробка модулів програми	27.04.24 – 11.05.24
6	Тестування програми	11.05.24 – 15.05.24
7	Оформлення матеріалів до захисту БДР	16.05.24 – 30.05.24

10. Порядок контролю та прийняття

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

62

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Проектування та розробка вебплатформи для автоматизованого створення телеграм-ботів за допомогою інструментів конструювання сценаріїв

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Чехместрук Р.Ю.

Оригінальність	93,4
Схожість	6,6

Аналіз звіту подібності

▪ **Запозичення, виявлені у роботі, оформленні коректно і не містять ознак плагіату.**

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цілісності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховання недобросовісних запозичень.

Особа, відповідальна за перевірку ______ Черноволик Г.О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи ______ Брильянт І.А.

Керівник роботи ______ Чехместрук Р.Ю.

Додаток В – Лістинг програми

```
import requests
from flask import Flask, request, render_template_string, session, g
from flask_session import Session
from flask import redirect, url_for
import sqlite3

app = Flask(__name__)

# база даних
DATABASE = 'data.db'

def get_db():
    db = getattr(g, '_database', None)
    if db is None:
        db = g._database = sqlite3.connect(DATABASE)
    return db

@app.teardown_appcontext
def close_connection(exception):
    db = getattr(g, '_database', None)
    if db is not None:
        db.close()
# фуекція перевірки токену
def get_bot_info(token):
    url = f"https://api.telegram.org/bot{token}/getMe"
    response = requests.get(url)
    if response.status_code == 200:
        bot_info = response.json()
        text = f"
    else:
        text = "Не вдалось отримати інформацію про бота!"
    return text

# головна сторінка
@app.route('/')
def index():
    return '''
<!DOCTYPE html>
<html lang="ru">

<head>
    <meta charset="UTF-8">
    <title>BotBlueprints</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="icon" href="/static/favicon.ico" type="image/x-icon">
    <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;500&display=swap"
rel="stylesheet">
    <style>
```

```
@media only screen and (min-width: 601px) {
   .menu-toggle {
      display: none;
   }
}

/* курсор  */
* {
   cursor: url("/static/my-cursor.png"), auto;
}

.clickable:hover {
   cursor: url("/static/my-cursor2.png"), pointer;
}

body {
   background: #111;
   color: white;
   font-family: 'Montserrat', sans-serif;
   margin: 0;
   padding: 0;
   display: flex;
   flex-direction: column;
   align-items: center;
   justify-content: center;
}

.header {
   width: 100%;
   display: flex;
   justify-content: space-between;
   align-items: center;
   padding: 10px 50px;
   box-sizing: border-box;
}

.logo {
   max-width: 15%;
   height: auto;
   cursor: url("/static/my-cursor2.png"), pointer;
}

.menu {
   margin: 0;
   padding: 0;
   list-style: none;
}

.menu li {
   display: inline;
```

```
    margin-right: 20px;
}

.menu a {
    color: white;
    text-decoration: none;
    font-size: 1.2em;
}

.menu a:hover {
    color: #301e8b;
}

.search-box {
    margin-top: 200px;
}

.search-input {
    width: 300px;
    padding: 10px;
    font-size: 16px;
    margin-bottom: 10px;
    cursor: url("/static/my-cursor2.png"), pointer;
}

.search-button {
    background-color: #0074D9;
    color: white;
    border: none;
    padding: 10px 20px;
    font-size: 16px;
    cursor: url("/static/my-cursor2.png"), pointer;
}

.wrapper span {
    position: fixed;
    bottom: -180px;
    height: 50px;
    width: 50px;
    z-index: -1;
    background-color: #18191f;
    box-shadow: 0 0 50px #0072ff, 0 0 100px #0072ff, 0 0 150px #0072ff, 0 0 200px #0072ff;
    animation: animate 10s linear infinite;
}

.wrapper span:nth-child(1) {
    left: 60px;
    animation-delay: 0.6s;
}
```

```
.wrapper span:nth-child(2) {
   left: 10%;
   animation-delay: 3s;
   width: 60px;
   height: 60px;
}

.wrapper span:nth-child(3) {
   left: 20%;
   animation-delay: 2s;
}

.wrapper span:nth-child(4) {
   left: 30%;
   animation-delay: 5s;
   width: 80px;
   height: 80px;
}

.wrapper span:nth-child(5) {
   left: 40%;
   animation-delay: 1s;
}

.wrapper span:nth-child(6) {
   left: 50%;
   animation-delay: 7s;
}

.wrapper span:nth-child(7) {
   left: 60%;
   animation-delay: 6s;
   width: 100px;
   height: 100px;
}

.wrapper span:nth-child(8) {
   left: 70%;
   animation-delay: 8s;
}

.wrapper span:nth-child(9) {
   left: 80%;
   animation-delay: 6s;
   width: 90px;
   height: 90px;
}

.wrapper span:nth-child(10) {
   left: 90%;
```

```
  animation-delay: 4s;
}

.banner {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

.content h2 b {
  -webkit-text-fill-color: transparent;
  -webkit-text-stroke-width: 3px;
  -webkit-text-stroke-color: #fff;
  font-family: montserrat;
  font-size: 80px;
  text-transform: uppercase;
  letter-spacing: 12px;
}

@keyframes animate {
  0% {
    transform: translateY(0);
    opacity: 1;
  }

  80% {
    opacity: .7;
  }

  100% {
    transform: translateY(-800px) rotate(360deg);
    opacity: 0;
  }
}

.decor {
  position: relative;
  max-width: 400px;
  margin: 50px auto 0;
  border-radius: 30px;
}

.form-inner {
  padding: 50px;
}

.form-inner input {
  display: block;
  width: 100%;
```

```
    padding: 0 20px;
    margin-bottom: 10px;
    line-height: 40px;
    border-width: 0;
    border-radius: 20px;
    font-family: 'Roboto', sans-serif;
}

.form-inner input[type="submit"] {
    margin-top: 30px;
    background: #002a5c;
    color: white;
    font-size: 14px;
}

.form-inner input[type="submit"]:hover {
    border-bottom: 4px solid #301e8b;
    cursor: url("/static/my-cursor2.png"), pointer;
}

.form-inner h3 {
    margin-top: 0;
    font-family: 'Roboto', sans-serif;
    font-weight: 500;
    font-size: 24px;
    color: #707981;
}

.form-inner input[type="text"] {
    background-color: transparent;
    border: 2px solid #002a5c;
    width: 300px;
    transition: border-color 0.3s ease;
    box-sizing: border-box;
    color: white;
}

.form-inner input[type="text"]:hover {
    border-color: #301e8b;
    cursor: url("/static/my-cursor2.png"), pointer;
}

.form-inner input[type="submit"] {
    width: 300px;
}

@media only screen and (max-width: 600px) {
    .container {
        padding: 10px;
    }
```

```
.header {
   display: flex;
   justify-content: space-between;
   align-items: center;
   padding: 10px;
}

.logo {
   max-width: 30%;
   height: auto;
}

.menu {
   margin: 0;
   padding: 0;
   list-style: none;
   text-align: center;
}

.menu li {
   display: block;
   margin-bottom: 10px;
}

.menu a {
   color: white;
   text-decoration: none;
   font-size: 1.2em;
}

.menu {
   position: fixed;
   top: 10%;
   right: -100%;
   height: 20%;
   width: 50%;
   background: black;
   padding-top: 20px;
   transition: right 0.5s ease;
   z-index: 1000;
   border-radius: 20px;
}

.menu.show {
   right: 0;
}

.menu a:hover {
   color: #301e8b;
```

```
            }
        }
    </style>
</head>

<body>
    <div class="wrapper">
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
    </div>
    <div class="header">
        <img src="/static/logo.png" alt="Логотип" class="logo">
        <div class="menu-toggle" onclick="toggleMenu()">
            <img src="/static/menu-icon.png" alt="Меню" class="menu-icon">
        </div>
        <ul class="menu">
            <li><a href="/" class="clickable" style="color: #301e8b;">Головна</a></li>
            <li><a href="/possibilities" class="clickable">Можливості</a></li>
            <li><a href="/price" class="clickable">Ціни</a></li>
        </ul>
    </div>

    <script>
        function toggleMenu() {
            var menu = document.querySelector('.menu');
            menu.classList.toggle('show');
        }

        // Сенсор для телефонів
        var startX, startY, distX, distY;
        var threshold = 50; // Мінімальна дистанція свайпу в пікселях

        document.addEventListener('touchstart', function(e) {
            var touchObj = e.changedTouches[0];
            startX = touchObj.pageX;
            startY = touchObj.pageY;
        });

        document.addEventListener('touchmove', function(e) {
            if (!startX || !startY) return;

            var touchObj = e.changedTouches[0];
```

```
            distX = touchObj.pageX - startX;
            distY = touchObj.pageY - startY;

            // Якщо користувач свайпнув вправо на достатню відстань, звертаємо меню
            if (distX > threshold) {
                toggleMenu();
                startX = null;
                startY = null;
            }
        });
    </script>

    <form class="decor" action="/token" method="post">
        <div class="form-inner">
            <input type="text" name="token" placeholder="Введіть токен бота">
            <input type="submit" value="Надіслати">
        </div>
    </form>

</body>

</html>'''
# отримання токену
@app.route('/token', methods=['POST'])
def submit():
    token = request.form['token']
    if token == '':
        return '''<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>BotBlueprints</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="icon" href="/static/favicon.ico" type="image/x-icon">
    <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;500&display=swap"
rel="stylesheet">
    <style>
        @media only screen and (min-width: 601px) {
    .menu-toggle {
        display: none;
    }
}
        /* курсор  */
        * {
            cursor: url("/static/my-cursor.png"), auto;
        }
        .clickable:hover {
            cursor: url("/static/my-cursor2.png"), pointer;
        }
```

```css
body {
   background: #111;
   color: white;
   font-family: Arial, sans-serif;
   margin: 0;
   padding: 0;
   display: flex;
   flex-direction: column;
   align-items: center;
   justify-content: center;
}

.header {
   width: 100%;
   display: flex;
   justify-content: space-between;
   align-items: center;
   padding: 10px 50px;
   box-sizing: border-box;
}

.logo {
   max-width: 15%;
   height: auto;
   cursor: url("/static/my-cursor2.png"), pointer;
}

.menu {
   margin: 0;
   padding: 0;
   list-style: none;
}

.menu li {
   display: inline;
   margin-right: 20px;
}

.menu a {
   color: white;
   text-decoration: none;
   font-size: 1.2em;
}

.menu a:hover {
   color: #301e8b;
}
```

```
        .search-box {
            margin-top: 200px;
        }

        .search-input {
            width: 300px;
            padding: 10px;
            font-size: 16px;
            margin-bottom: 10px;
            cursor: url("/static/my-cursor2.png"), pointer;
        }

        .search-button {
            background-color: #0074D9;
            color: white;
            border: none;
            padding: 10px 20px;
            font-size: 16px;
            cursor: url("/static/my-cursor2.png"), pointer;
        }
.wrapper span {
    position: fixed;
    bottom: -180px;
    height: 50px;
    width: 50px;
    z-index: -1;
    background-color: #18191f;
    box-shadow: 0 0 50px #0072ff, 0 0 100px #0072ff, 0 0 150px #0072ff, 0 0 200px #0072ff;
    animation: animate 10s linear infinite;
}
.wrapper span:nth-child(1) {
    left: 60px;
    animation-delay: 0.6s;
}
.wrapper span:nth-child(2) {
    left: 10%;
    animation-delay: 3s;
    width: 60px;
    height: 60px;
}
.wrapper span:nth-child(3) {
    left: 20%;
    animation-delay: 2s;
}
.wrapper span:nth-child(4) {
    left: 30%;
    animation-delay: 5s;
    width: 80px;
    height: 80px;
```

```
}
.wrapper span:nth-child(5) {
   left: 40%;
   animation-delay: 1s;
}
.wrapper span:nth-child(6) {
   left: 50%;
   animation-delay: 7s;
}
.wrapper span:nth-child(7) {
   left: 60%;
   animation-delay: 6s;
   width: 100px;
   height: 100px;
}
.wrapper span:nth-child(8) {
   left: 70%;
   animation-delay: 8s;
}
.wrapper span:nth-child(9) {
   left: 80%;
   animation-delay: 6s;
   width: 90px;
   height: 90px;
}
.wrapper span:nth-child(10) {
   left: 90%;
   animation-delay: 4s;
}
.banner {
   display: flex;
   justify-content: center;
   align-items: center;
   height: 100vh;
}
.content h2 b {
   -webkit-text-fill-color: transparent;
   -webkit-text-stroke-width: 3px;
   -webkit-text-stroke-color: #fff;
   font-family: montserrat;
   font-size: 80px;
   text-transform: uppercase;
   letter-spacing: 12px;
}
@keyframes animate {
   0% {
      transform: translateY(0);
      opacity: 1;
   }
   80% {
```

```
        opacity: .7;
    }
    100% {
        transform: translateY(-800px) rotate(360deg);
        opacity: 0;
    }
}

.decor {
  position: relative;
  max-width: 400px;
  margin: 50px auto 0;
  border-radius: 30px;
}

.form-inner {
  padding: 50px;
}
.form-inner input{
  display: block;
  width: 100%;
  padding: 0 20px;
  margin-bottom: 10px;
  line-height: 40px;
  border-width: 0;
  border-radius: 20px;
  font-family: 'Roboto', sans-serif;
}
.form-inner input[type="submit"] {
  margin-top: 30px;
  background: #002a5c;
  color: white;
  font-size: 14px;
}
.form-inner input[type="submit"]:hover {
    border-bottom: 4px solid #301e8b;
    cursor: url("/static/my-cursor2.png"), pointer;
}

.form-inner h3 {
  margin-top: 0;
  font-family: 'Roboto', sans-serif;
  font-weight: 500;
  font-size: 24px;
  color: #707981;
}

.form-inner input[type="text"] {
    background-color: transparent;
    border: 2px solid #002a5c;
```

```
    width: 300px;
    transition: border-color 0.3s ease;
    box-sizing: border-box;
    color: white;
}

.form-inner input[type="text"]:hover {
    border-color: #301e8b;
    cursor: url("/static/my-cursor2.png"), pointer;
}


.form-inner input[type="submit"] {
    width: 300px;
}

.error-message {
    color: red;
    font-size: 20px;
    margin-bottom: -60px;
margin-top: 40px;
text-align: center;
}

@media only screen and (max-width: 600px) {
    .error-message {
        font-size: 20px;
        margin-bottom: -60px;
    }
            .container {
                padding: 10px;
            }
            .header {
                display: flex;
                justify-content: space-between;
                align-items: center;
                padding: 10px;
            }
            .logo {
                max-width: 30%;
                height: auto;
            }

.menu {
    margin: 0;
    padding: 0;
    list-style: none;
    text-align: center;
}
```

```
.menu a {
   color: white;
   text-decoration: none;
   font-size: 1.2em;
   display: block;
   margin-bottom: 10px;
}

.menu {
   position: fixed;
   top: 10%;
   right: -100%;
   height: 20%;
   width: 50%;
   background: black;
   padding-top: 20px;
   transition: right 0.5s ease;
   z-index: 1000;
   border-radius: 20px;
}

.menu.show {
   right: 0;
}

.menu ul {
   list-style: none;
   padding: 0;
   margin: 0;
   text-align: center;
}

.menu a:hover {
   color: #301e8b;
}
}

   </style>
</head>
<body>
   <div class="wrapper">
      <span></span>
      <span></span>
      <span></span>
      <span></span>
      <span></span>
      <span></span>
      <span></span>
      <span></span>
      <span></span>
```

```
        <span></span>
</div>
   <div class="header">
        <img src="/static/logo.png" alt="Логотип" class="logo">
        <div class="menu-toggle" onclick="toggleMenu()">
            <img src="/static/menu-icon.png" alt="Меню" class="menu-icon">
        </div>
        <ul class="menu">
            <li><a href="/" class="clickable">Головна</a></li>
            <li><a href="/possibilities" class="clickable">Можливості</a></li>
            <li><a href="/price" class="clickable">Ціни</a></li>
        </ul>
    </div>

<script>
    function toggleMenu() {
        var menu = document.querySelector('.menu');
        menu.classList.toggle('show');
    }

    var startX, startY, distX, distY;
    var threshold = 50;

    document.addEventListener('touchstart', function(e) {
        var touchObj = e.changedTouches[0];
        startX = touchObj.pageX;
        startY = touchObj.pageY;
    });

    document.addEventListener('touchmove', function(e) {
        if (!startX || !startY) return;

        var touchObj = e.changedTouches[0];
        distX = touchObj.pageX - startX;
        distY = touchObj.pageY - startY;
        if (distX > threshold) {
            toggleMenu();
            startX = null;
            startY = null;
        }
    });
</script>
<div class="error-message">
    Токен не введено, введіть токен!
</div>
<br>
<div class="error-message">
    Якщо у вас є які-небудь питання або потрібна допомога, не соромтеся звертатися до <a
href ='t.me/brelivx'>нас</a>.
</div>
```

```
<form class="decor" action="/token" method="post">
    <div class="form-inner">
        <input type="text" name="token" placeholder="Введіть токен бота">
        <input type="submit" value="Надіслати">
    </div>
</form>

</body>
</html>'''
    text = get_bot_info(token)
    if text == 'Не вдалось отримати інформацію про бота!':
        return '''<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>BotBlueprints</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="icon" href="/static/favicon.ico" type="image/x-icon">
    <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;500&display=swap"
rel="stylesheet">
    <style>
        @media only screen and (min-width: 601px) {
    .menu-toggle {
        display: none;
    }
}
        /* курсор  */
        * {
            cursor: url("/static/my-cursor.png"), auto;
        }
        .clickable:hover {
            cursor: url("/static/my-cursor2.png"), pointer;
        }


        body {
            background: #111;
            color: white;
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
        }

        .header {
            width: 100%;
```

```
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px 50px;
    box-sizing: border-box;
}

.logo {
    max-width: 15%;
    height: auto;
    cursor: url("/static/my-cursor2.png"), pointer;
}

.menu {
    margin: 0;
    padding: 0;
    list-style: none;
}

.menu li {
    display: inline;
    margin-right: 20px;
}

.menu a {
    color: white;
    text-decoration: none;
    font-size: 1.2em;
}

.menu a:hover {
    color: #301e8b;
}


.search-box {
    margin-top: 200px;
}

.search-input {
    width: 300px;
    padding: 10px;
    font-size: 16px;
    margin-bottom: 10px;
    cursor: url("/static/my-cursor2.png"), pointer;
}

.search-button {
    background-color: #0074D9;
    color: white;
```

```
            border: none;
            padding: 10px 20px;
            font-size: 16px;
            cursor: url("/static/my-cursor2.png"), pointer;
          }
.wrapper span {
   position: fixed;
   bottom: -180px;
   height: 50px;
   width: 50px;
   z-index: -1;
   background-color: #18191f;
   box-shadow: 0 0 50px #0072ff, 0 0 100px #0072ff, 0 0 150px #0072ff, 0 0 200px #0072ff;
   animation: animate 10s linear infinite;
}
.wrapper span:nth-child(1) {
   left: 60px;
   animation-delay: 0.6s;
}
.wrapper span:nth-child(2) {
   left: 10%;
   animation-delay: 3s;
   width: 60px;
   height: 60px;
}
.wrapper span:nth-child(3) {
   left: 20%;
   animation-delay: 2s;
}
.wrapper span:nth-child(4) {
   left: 30%;
   animation-delay: 5s;
   width: 80px;
   height: 80px;
}
.wrapper span:nth-child(5) {
   left: 40%;
   animation-delay: 1s;
}
.wrapper span:nth-child(6) {
   left: 50%;
   animation-delay: 7s;
}
.wrapper span:nth-child(7) {
   left: 60%;
   animation-delay: 6s;
   width: 100px;
   height: 100px;
}
.wrapper span:nth-child(8) {
```

```
    left: 70%;
    animation-delay: 8s;
}
.wrapper span:nth-child(9) {
    left: 80%;
    animation-delay: 6s;
    width: 90px;
    height: 90px;
}
.wrapper span:nth-child(10) {
    left: 90%;
    animation-delay: 4s;
}
.banner {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}
.content h2 b {
    -webkit-text-fill-color: transparent;
    -webkit-text-stroke-width: 3px;
    -webkit-text-stroke-color: #fff;
    font-family: montserrat;
    font-size: 80px;
    text-transform: uppercase;
    letter-spacing: 12px;
}
@keyframes animate {
    0% {
        transform: translateY(0);
        opacity: 1;
    }
    80% {
        opacity: .7;
    }
    100% {
        transform: translateY(-800px) rotate(360deg);
        opacity: 0;
    }
}

.decor {
  position: relative;
  max-width: 400px;
  margin: 50px auto 0;
  border-radius: 30px;
}

.form-inner {
```

```
  padding: 50px;
}
.form-inner input{
  display: block;
  width: 100%;
  padding: 0 20px;
  margin-bottom: 10px;
  line-height: 40px;
  border-width: 0;
  border-radius: 20px;
  font-family: 'Roboto', sans-serif;
}
.form-inner input[type="submit"] {
  margin-top: 30px;
  background: #002a5c;
  color: white;
  font-size: 14px;
}
.form-inner input[type="submit"]:hover {
   border-bottom: 4px solid #301e8b;
   cursor: url("/static/my-cursor2.png"), pointer;
}

.form-inner h3 {
  margin-top: 0;
  font-family: 'Roboto', sans-serif;
  font-weight: 500;
  font-size: 24px;
  color: #707981;
}

.form-inner input[type="text"] {
   background-color: transparent;
   border: 2px solid #002a5c;
   width: 300px;
   transition: border-color 0.3s ease;
   box-sizing: border-box;
   color: white;
}

.form-inner input[type="text"]:hover {
   border-color: #301e8b;
   cursor: url("/static/my-cursor2.png"), pointer;
}


.form-inner input[type="submit"] {
   width: 300px;
}
```

```
.error-message {
   color: red;
   font-size: 20px;
   margin-bottom: -60px;
margin-top: 40px;
text-align: center;
}

@media only screen and (max-width: 600px) {
   .error-message {
      font-size: 20px;
      margin-bottom: -60px;
   }
         .container {
            padding: 10px;
         }
         .header {
            display: flex;
            justify-content: space-between;
            align-items: center;
            padding: 10px;
         }
         .logo {
            max-width: 30%;
            height: auto;
         }

.menu {
   margin: 0;
   padding: 0;
   list-style: none;
   text-align: center;
}

.menu a {
   color: white;
   text-decoration: none;
   font-size: 1.2em;
   display: block;
   margin-bottom: 10px;
}

.menu {
   position: fixed;
   top: 10%;
   right: -100%;
   height: 20%;
   width: 50%;
   background: black;
   padding-top: 20px;
```

```
    transition: right 0.5s ease;
    z-index: 1000;
    border-radius: 20px;
}

.menu.show {
    right: 0;
}

.menu ul {
    list-style: none;
    padding: 0;
    margin: 0;
    text-align: center;
}

.menu a:hover {
    color: #301e8b;
}
}

    </style>
</head>
<body>
    <div class="wrapper">
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
</div>
   <div class="header">
        <img src="/static/logo.png" alt="Логотип" class="logo">
        <div class="menu-toggle" onclick="toggleMenu()">
            <img src="/static/menu-icon.png" alt="Меню" class="menu-icon">
        </div>
        <ul class="menu">
            <li><a href="/" class="clickable">Головна</a></li>
            <li><a href="/possibilities" class="clickable">Можливості</a></li>
            <li><a href="/price" class="clickable">Ціни</a></li>
        </ul>
    </div>

<script>
    function toggleMenu() {
```

```
        var menu = document.querySelector('.menu');
        menu.classList.toggle('show');
    }

    var startX, startY, distX, distY;
    var threshold = 50;

    document.addEventListener('touchstart', function(e) {
        var touchObj = e.changedTouches[0];
        startX = touchObj.pageX;
        startY = touchObj.pageY;
    });

    document.addEventListener('touchmove', function(e) {
        if (!startX || !startY) return;

        var touchObj = e.changedTouches[0];
        distX = touchObj.pageX - startX;
        distY = touchObj.pageY - startY;
        if (distX > threshold) {
            toggleMenu();
            startX = null;
            startY = null;
        }
    });
</script>
<div class="error-message">
    Не вдалося отримати інформацію про бота. Перевірте токен.
</div>
<form class="decor" action="/token" method="post">
    <div class="form-inner">
        <input type="text" name="token" placeholder="Введіть токен бота">
        <input type="submit" value="Надіслати">
    </div>
</form>

</body>
</html>'''
    else:
        return redirect(url_for('bot_info', token=token))
# сторінка інформації
@app.route('/bot_info/<token>')
def bot_info(token):
    db = get_db()
    cursor = db.cursor()
    cursor.execute("SELECT * FROM bots WHERE token = ?", (token,))
    bot_data = cursor.fetchall()
    cursor.close()
    if bot_data:
        for row in bot_data:
```

```
            admin_username = row[2]
            shablon = row[3]
            stan = row[5]

        if stan == 0:
            stan = '❌'
        else:
            stan = '✅'

        if admin_username == '0':
            admin = '❌'
        else:
            admin = admin_username

        if shablon == 0:
            shablon = '❌'
    else:
        stan = '❌'
        admin = '❌'
        shablon = '❌'

    url = f"https://api.telegram.org/bot{token}/getMe"
    response = requests.get(url)
    if response.status_code == 200:
        bot_info = response.json()
        text = ''
        if "username" in bot_info["result"]:
            username = bot_info["result"]["username"]
            name = bot_info["result"].get("first_name", "")
    else:
        return('ПОМИЛКА')
    return render_template_string('''
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>BotBlueprints</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="icon" href="/static/favicon.ico" type="image/x-icon">
    <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;500&display=swap"
rel="stylesheet">
    <style>
        @media only screen and (min-width: 601px) {
    .menu-toggle {
        display: none;
    }
}
        /* курсор  */
```

```
* {
   cursor: url("/static/my-cursor.png"), auto;
}
.clickable:hover {
   cursor: url("/static/my-cursor2.png"), pointer;
}


body {
       background: #111;
   color: white;
   font-family: Arial, sans-serif;
   margin: 0;
   padding: 0;
   display: flex;
   flex-direction: column;
   align-items: center;
   justify-content: center;
}

.header {
   width: 100%;
   display: flex;
   justify-content: space-between;
   align-items: center;
   padding: 10px 50px;
   box-sizing: border-box;
}

.logo {
   max-width: 15%;
   height: auto;
   cursor: url("/static/my-cursor2.png"), pointer;
}

.menu {
   margin: 0;
   padding: 0;
   list-style: none;
}

.menu li {
   display: inline;
   margin-right: 20px;
}

.menu a {
   color: white;
   text-decoration: none;
   font-size: 1.2em;
```

```
    }

    .menu a:hover {
      color: #301e8b;
    }


    .search-box {
      margin-top: 200px;
    }

    .search-input {
      width: 300px;
      padding: 10px;
      font-size: 16px;
      margin-bottom: 10px;
      cursor: url("/static/my-cursor2.png"), pointer;
    }

    .search-button {
      background-color: #0074D9;
      color: white;
      border: none;
      padding: 10px 20px;
      font-size: 16px;

      cursor: url("/static/my-cursor2.png"), pointer;
    }
    .wrapper span {
  position: fixed;
  bottom: -180px;
  height: 50px;
  width: 50px;
  z-index: -1;
  background-color: #18191f;
  box-shadow: 0 0 50px #0072ff, 0 0 100px #0072ff, 0 0 150px #0072ff, 0 0 200px #0072ff;
  animation: animate 10s linear infinite;
}
.wrapper span:nth-child(1) {
  left: 60px;
  animation-delay: 0.6s;
}
.wrapper span:nth-child(2) {
  left: 10%;
  animation-delay: 3s;
  width: 60px;
  height: 60px;
}
.wrapper span:nth-child(3) {
  left: 20%;
```

```
    animation-delay: 2s;
}
.wrapper span:nth-child(4) {
    left: 30%;
    animation-delay: 5s;
    width: 80px;
    height: 80px;
}
.wrapper span:nth-child(5) {
    left: 40%;
    animation-delay: 1s;
}
.wrapper span:nth-child(6) {
    left: 50%;
    animation-delay: 7s;
}
.wrapper span:nth-child(7) {
    left: 60%;
    animation-delay: 6s;
    width: 100px;
    height: 100px;
}
.wrapper span:nth-child(8) {
    left: 70%;
    animation-delay: 8s;
}
.wrapper span:nth-child(9) {
    left: 80%;
    animation-delay: 6s;
    width: 90px;
    height: 90px;
}
.wrapper span:nth-child(10) {
    left: 90%;
    animation-delay: 4s;
}
.banner {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}
.content h2 b {
    -webkit-text-fill-color: transparent;
    -webkit-text-stroke-width: 3px;
    -webkit-text-stroke-color: #fff;
    font-family: montserrat;
    font-size: 80px;
    text-transform: uppercase;
    letter-spacing: 12px;
```

```
}
@keyframes animate {
   0% {
      transform: translateY(0);
      opacity: 1;
   }
   80% {
      opacity: .7;
   }
   100% {
      transform: translateY(-800px) rotate(360deg);
      opacity: 0;
   }
}

.menus {
   margin: 20px;
   margin-right: -15%;
   padding: 0;
   list-style: none;
}

.menus li {
   display: inline;
   margin-right: 20px;
   position: relative;
}

.menus li:not(:last-child):after {
   content: "";
   position: absolute;
   top: 50%;
   right: -10px;
   transform: translateY(-50%);
   width: 1px;
   height: 80%;
   background-color: #301e8b;
}

.menus a {
   color: white;
   text-decoration: none;
   font-size: 25px;
}

.menus a:hover {
   color: #301e8b;
}

.container {
```

```
    width:85%;
    color: white;
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    align-items: left;
    justify-content: left;
}

.menu-image {
    display: none;
    width: 30%;
    margin-right: 20px;
}

.ava {
    width: 90px;
    margin-bottom: 10px;
}

@media only screen and (min-width: 768px) {
    .menu-image {
        display: block;
    }
}



.info {
    background-color: rgba(128, 128, 128, 0.5);
    border: 2px solid blue;
    border-radius: 20px;
    list-style-type: none;
    padding: 0;
    margin-left: 4%;
    margin-top: 2%;
    width: 47.5%;
}


.info_li {
    width: 70%;
    margin-left: 50%;
    margin-top: 2%;
    margin-bottom: 10px;
    font-size: 20px;
    color: white;
}
```

```
.username-link {
   font-size: 24px;
   color: blue;
   text-decoration: none;
}

.username-link:hover {
   color: navy;
}

.username-link2 {
   font-size: 1em;
   color: White;
   text-decoration: none;
}

.username-link2:hover {
   color: blue;
}

.avatar{
   width: 25%;
   margin-right: 80%;
   margin-top: 3%;
   align-items: center;
   text-align: center;
}

.text_info{
   font-size: 130%;
   width: 60%;
   margin-left: 40%;
   margin-top: -20%;
}

.em {
   width: 1.5em;
   height: 1.5em;
   vertical-align: middle;
   margin-right: 3%;
}

.text {
   margin-bottom: 10px;
}


@media only screen and (max-width: 600px) {

.text_info{
```

```
    font-size: 100%;
    width: 90%;
    margin-left: 5%;
    margin-top: 10%;
}

.em {
    width: 1.5em;
    height: 1.5em;
    vertical-align: middle;
    margin-right: 3%;
}

.text {
    margin-bottom: 10px;
}

.info {
    background-color: rgba(128, 128, 128, 0.5);
    border: 2px solid blue;
    border-radius: 20px;
    list-style-type: none;
    padding: 0;
    margin-left: 4%;
    margin-top: 2%;
    width: 90%;
}
.info_li {
    width: 150%;
    margin-left: -30%;
    margin-top: 2%;
    margin-bottom: 10px;
    font-size: 0.8em;
    color: white;
}

.ava{
    width: 80%;
}
.avatar{
    margin-left: 39%;
    margin-top: 3%;
    align-items: center;
    text-align: center;

}

            .container {
                padding: 10px;
            }
```

```
        .header {
          display: flex;
          justify-content: space-between;
          align-items: center;
          padding: 10px;
        }

        .logo {
          max-width: 30%;
          height: auto;
        }



.menu {
  margin: 0;
  padding: 0;
  list-style: none;
  text-align: center;
}


.menu a {
  color: white;
  text-decoration: none;
  font-size: 1.2em;
  display: block;
  margin-bottom: 10px;
}

.menu {
  position: fixed;
  top: 10%;
  right: -100%;
  height: 20%;
  width: 50%;
  background: black;
  padding-top: 20px;
  transition: right 0.5s ease;
  z-index: 1000;
  border-radius: 20px;
}
        .menu.show {
          right: 0;
        }

        .menu ul {
          list-style: none;
          padding: 0;
```

```
            margin: 0;
            text-align: center;
          }
          .menu a:hover {
            color: #301e8b;
          }
.menus {
  margin: 20px;
  padding: 0;
  list-style: none;
}

.menus li {
  display: inline;
  margin-right: 20px;
  position: relative;
}

.menus li:not(:last-child):after {
  content: "";
  position: absolute;
  top: 50%;
  right: -10px;
  transform: translateY(-50%);
  width: 1px;
  height: 80%;
  background-color: #301e8b;
}

.menus a {
  color: white;
  text-decoration: none;
  font-size: 12px;
}

.menus a:hover {
  color: #301e8b;
}
    }

  </style>
</head>
<body>
  <div class="wrapper">
    <span></span>
    <span></span>
    <span></span>
    <span></span>
    <span></span>
    <span></span>
```

```
        <span></span>
        <span></span>
        <span></span>
        <span></span>
</div>
  <div class="header">
        <img src="/static/logo.png" alt="Логотип" class="logo">
        <div class="menu-toggle" onclick="toggleMenu()">
            <img src="/static/menu-icon.png" alt="Меню" class="menu-icon">
        </div>
        <ul class="menu">
            <li><a href="/" class="clickable">Головна</a></li>
            <li><a href="/possibilities" class="clickable">Можливості</a></li>
            <li><a href="/price" class="clickable">Ціни</a></li>
        </ul>
    </div>

<script>
    function toggleMenu() {
        var menu = document.querySelector('.menu');
        menu.classList.toggle('show');
    }

    var startX, startY, distX, distY;
    var threshold = 50;

    document.addEventListener('touchstart', function(e) {
        var touchObj = e.changedTouches[0];
        startX = touchObj.pageX;
        startY = touchObj.pageY;
    });

    document.addEventListener('touchmove', function(e) {
        if (!startX || !startY) return;

        var touchObj = e.changedTouches[0];
        distX = touchObj.pageX - startX;
        distY = touchObj.pageY - startY;

        if (distX > threshold) {
            toggleMenu();
            startX = null;
            startY = null;
        }
    });
</script>


<ul class="menus">
    <li><a style="color: blue;" href="/bot_info/{{token}}" class="clickable">Інформація</a></li>
```

```
    <li><a href="/bot_settings/{{token}}" class="clickable">Налаштування</a></li>
    <li><a href="/" class="clickable">Статистика</a></li>
    <li><a href="/" class="clickable">Шаблони</a></li>
</ul>


<div class="container">
    <img src="/static/chatbot.png" class="menu-image" alt="Menu Image">


    <ul class="info">
    <div class='avatar'>
        <li class='info_li'><img src="/static/ava.png" class="ava" alt="ava">
        <br>
        <b>{{ name }}</b></li>
    </div>
    <div class='text_info'>
        <li class="text"><b>🌀 Username: <a href='https://t.me/{{username}}' target="_blank"
class="username-link2"> @{{username}}</a></b></li>
        <li class="text"><b>⚠️ Стан: {{stan}}</b></li>
        <li class="text"><b>🧑‍💻 Адмін: {{admin}}</b></li>
        <li class="text"><b>⚙️ Шаблон: {{shablon}}</b></li>
    </div>
     </ul>

</div>
</body>
</html>''', username = username, name = name, stan = stan, admin = admin, shablon=shablon,
token=token)

# бескінечний цикл
if __name__ == '__main__':
    app.run(debug=True)
```

Додаток Г – Ілюстративна частина

ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБПЛАТФОРМИ ДЛЯ АВТОМАТИЗОВАНОГО СТВОРЕННЯ ТЕЛЕГРАМ-БОТІВ ЗА ДОПОМОГОЮ ІНСТРУМЕНТІВ КОНСТРУЮВАННЯ СЦЕНАРІЇВ

Рисунок Г.1 – Титульний слайд

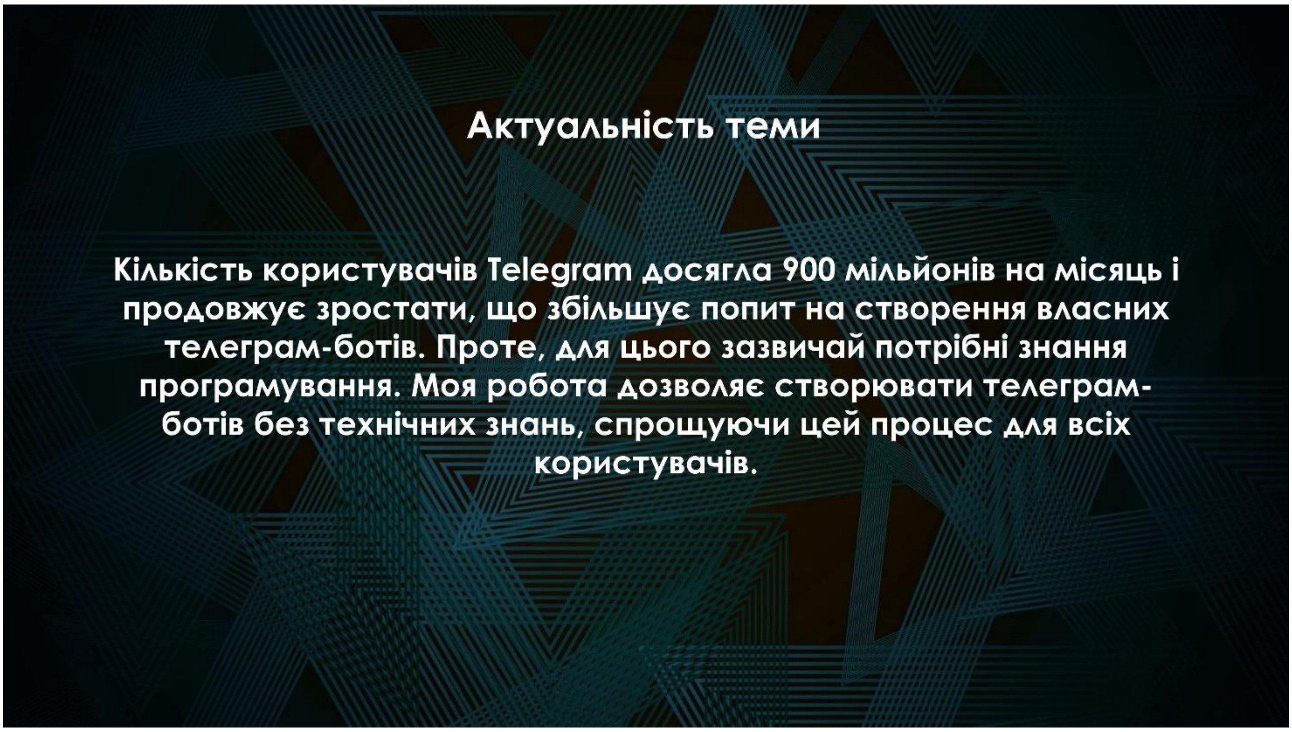

Рисунок Г.2 – Актуальність теми

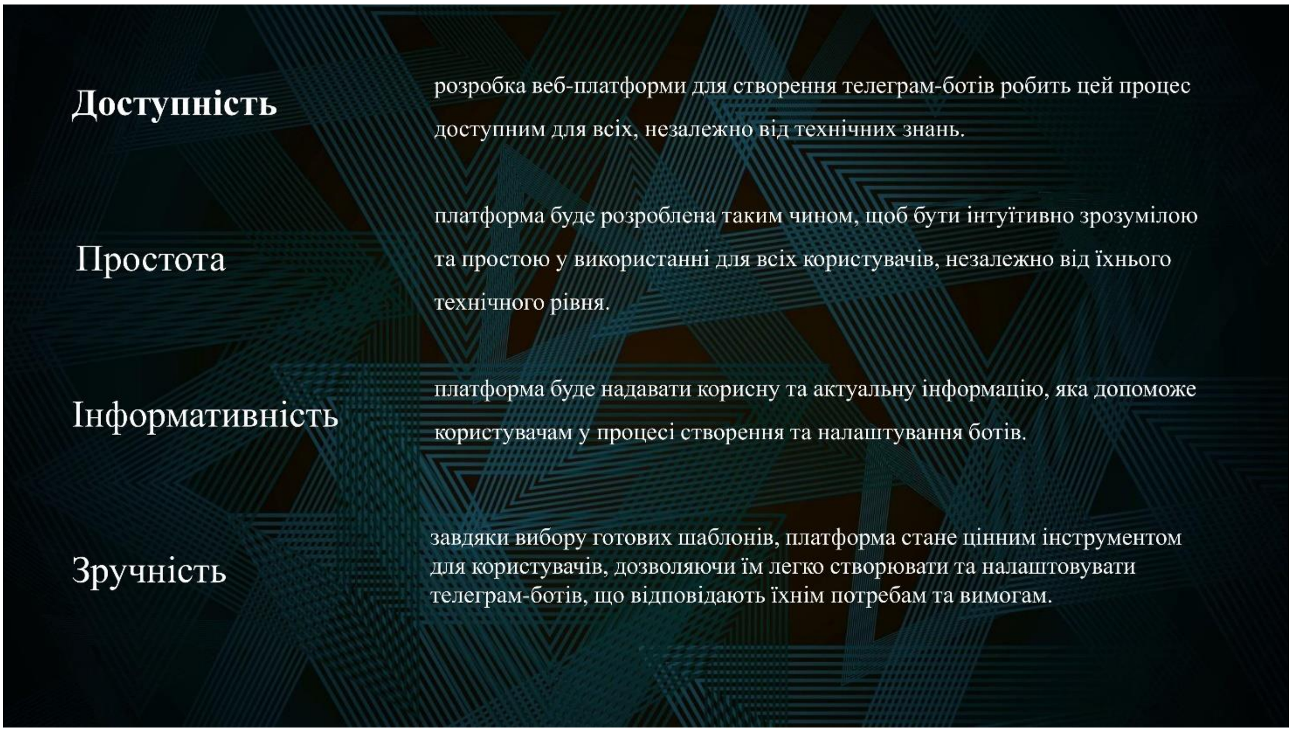

Рисунок Г.3 – Доступність, простота, інформативність та зручність вебплатформи

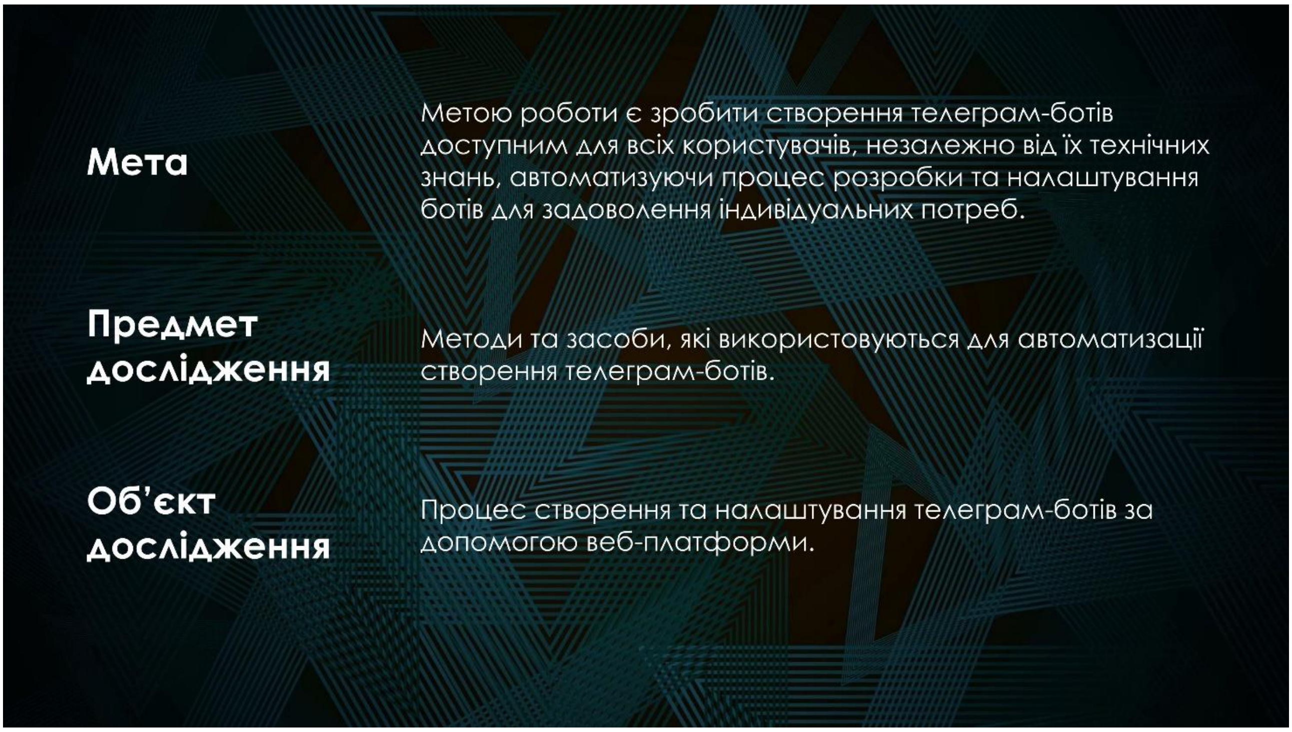

Рисунок Г.4 – Мета, об’єкт та предмет дослідження

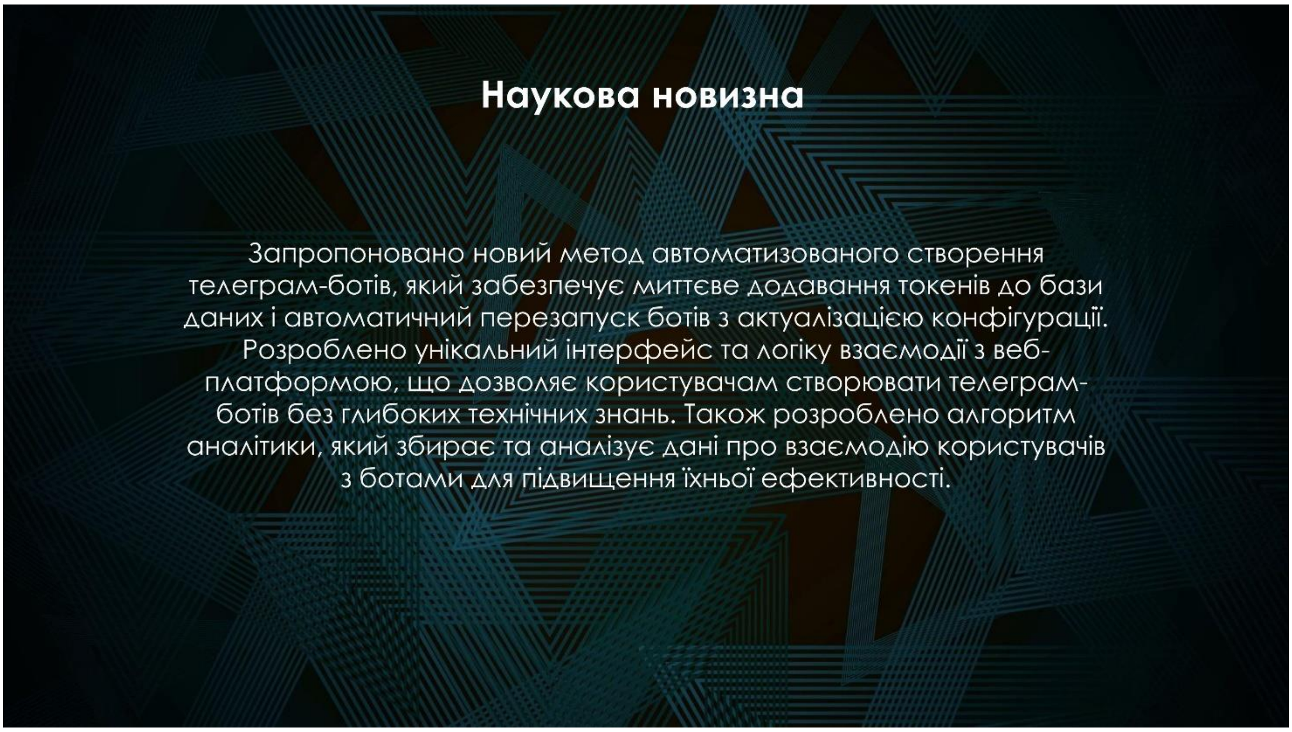

Рисунок Г.5 – Наукова новизна

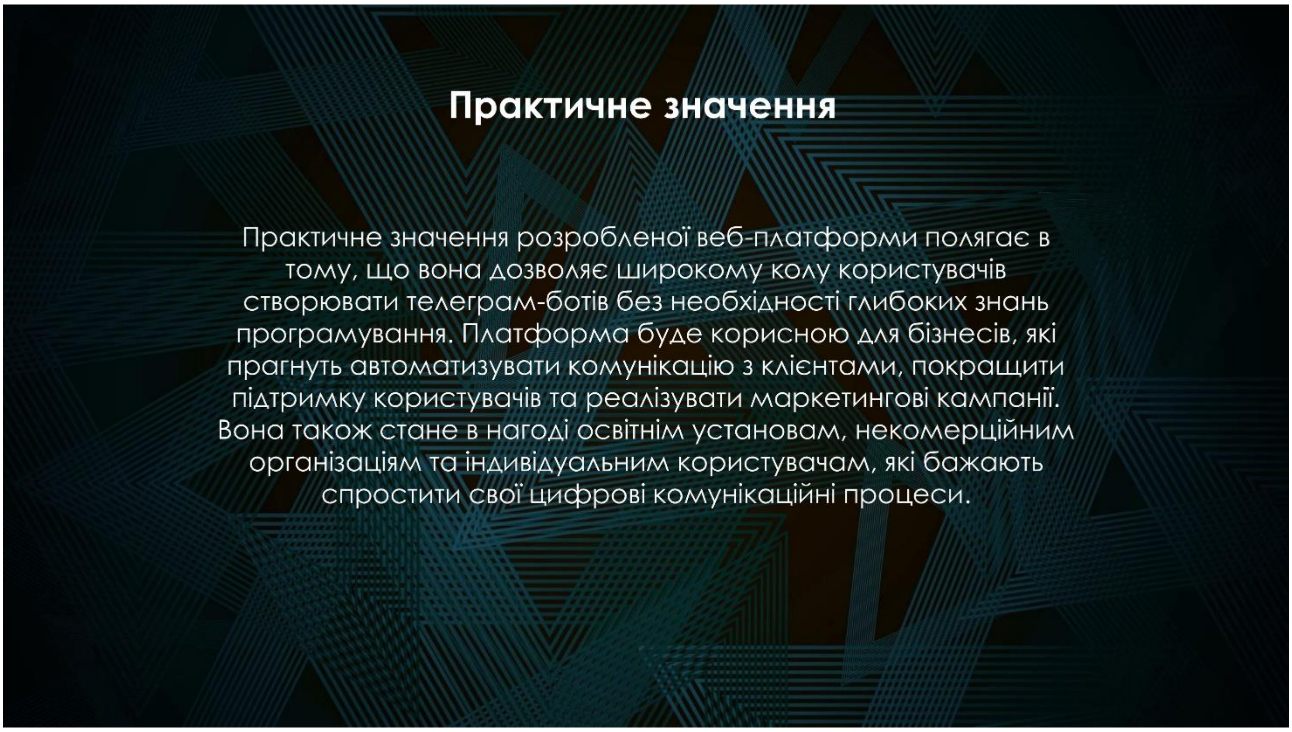

Рисунок Г.6 – Практичне призначення

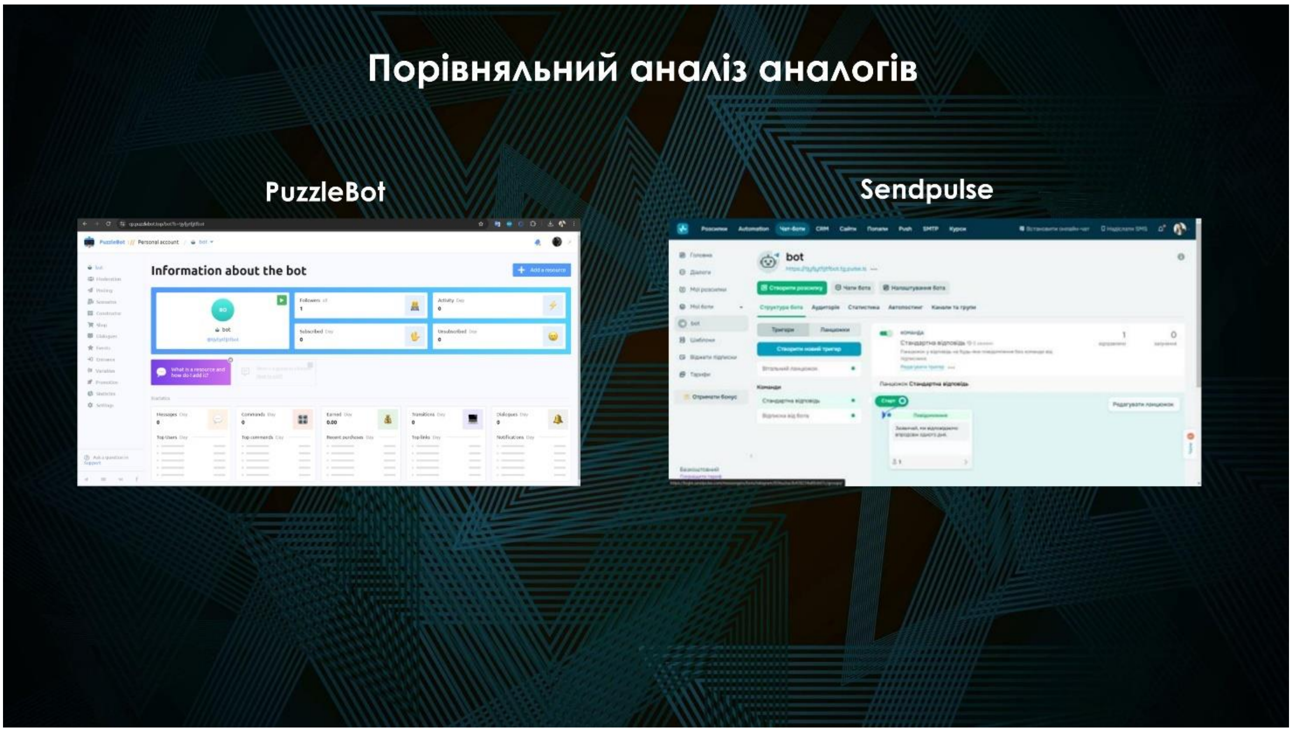

Рисунок Г.7 – Порівняльний аналіз аналогів

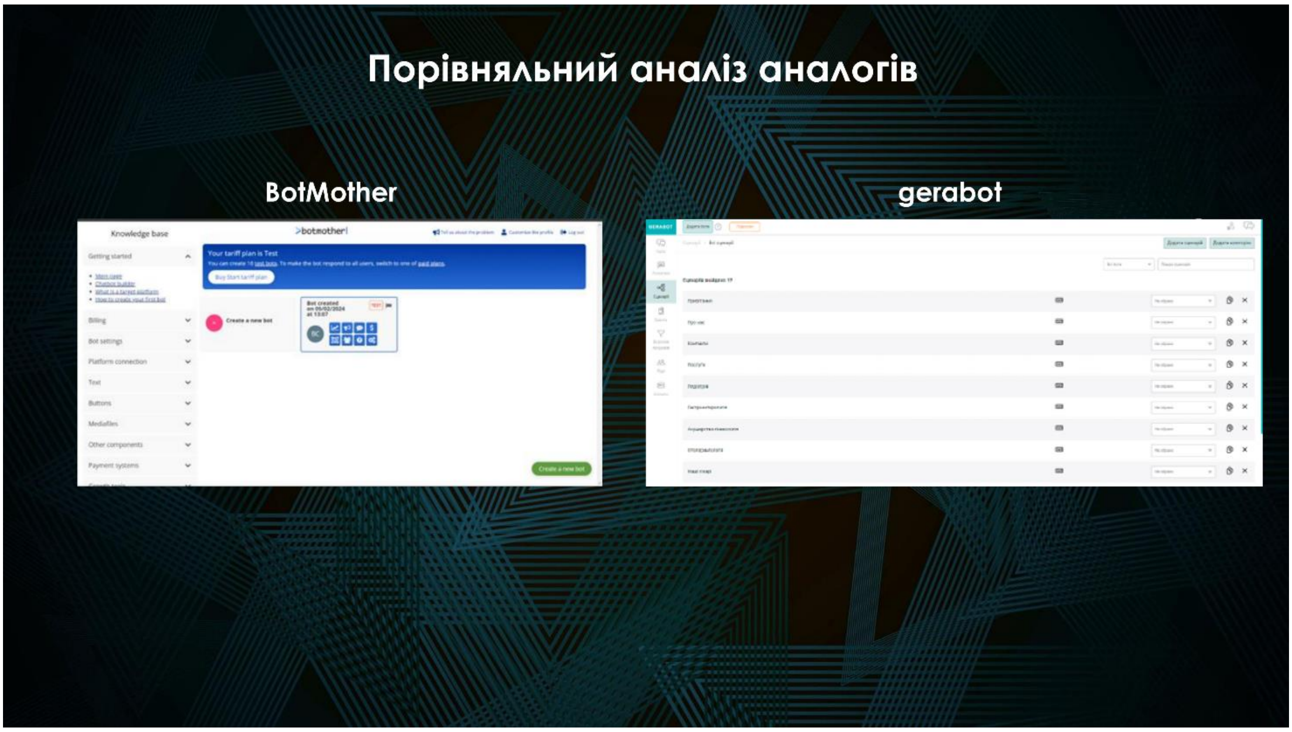

Рисунок Г.8 – Порівняльний аналіз аналогів

Порівняльний аналіз аналогів

Критерії	PuzzleBot	SendPulse	Gerabot	BotMother	BotHelp	Власний модуль
Інтуїтивність інтерфейсу	+	+	-	-	+	+
Гнучкість налаштувань	-	-	+	+	+	+
Підтримка конструктора сценаріїв	+	+	-	+	+	+
Інтеграція з іншими сервісами	+	+	+	+	-	+
Підтримка шаблонів	+	+	-	+	-	+
Загальна оцінка	80%	80%	40%	80%	60%	100%

Рисунок Г.9 – Порівняльний аналіз аналогів

Рисунок Г.10 – Технології розробки

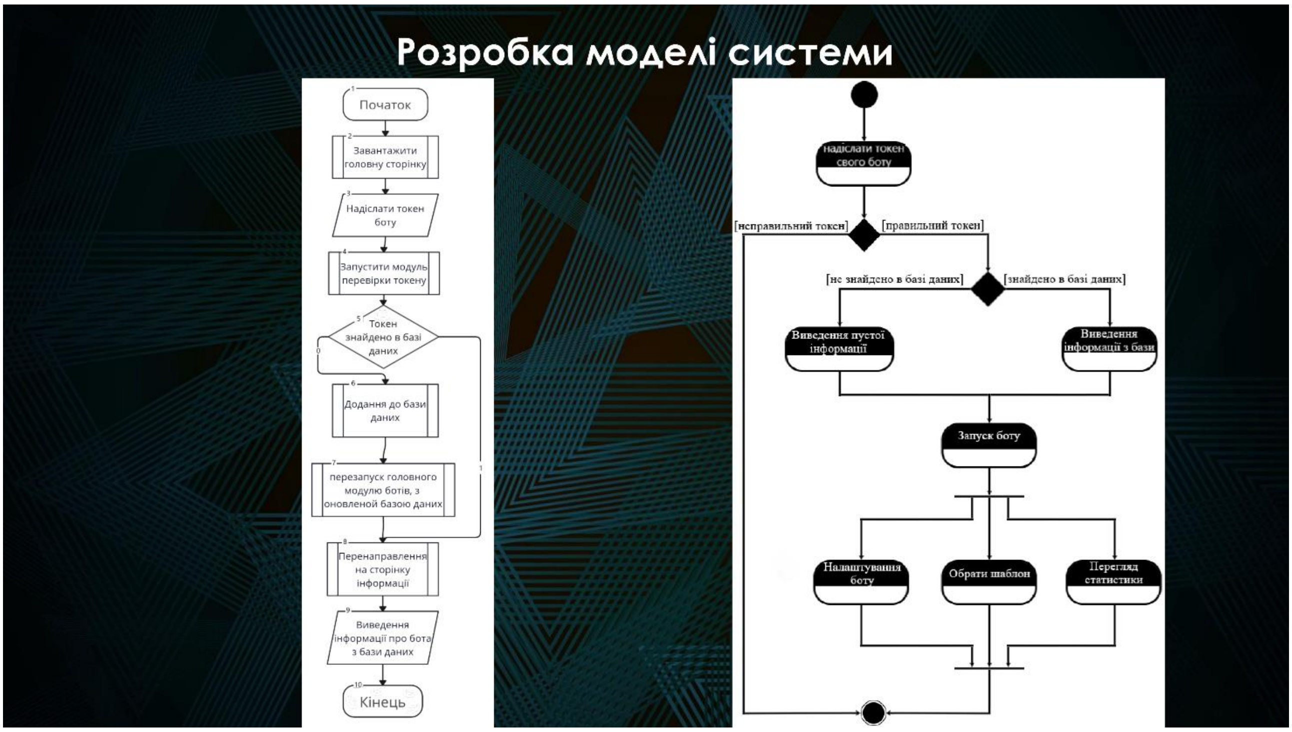

Рисунок Г.11 – Моделі системи

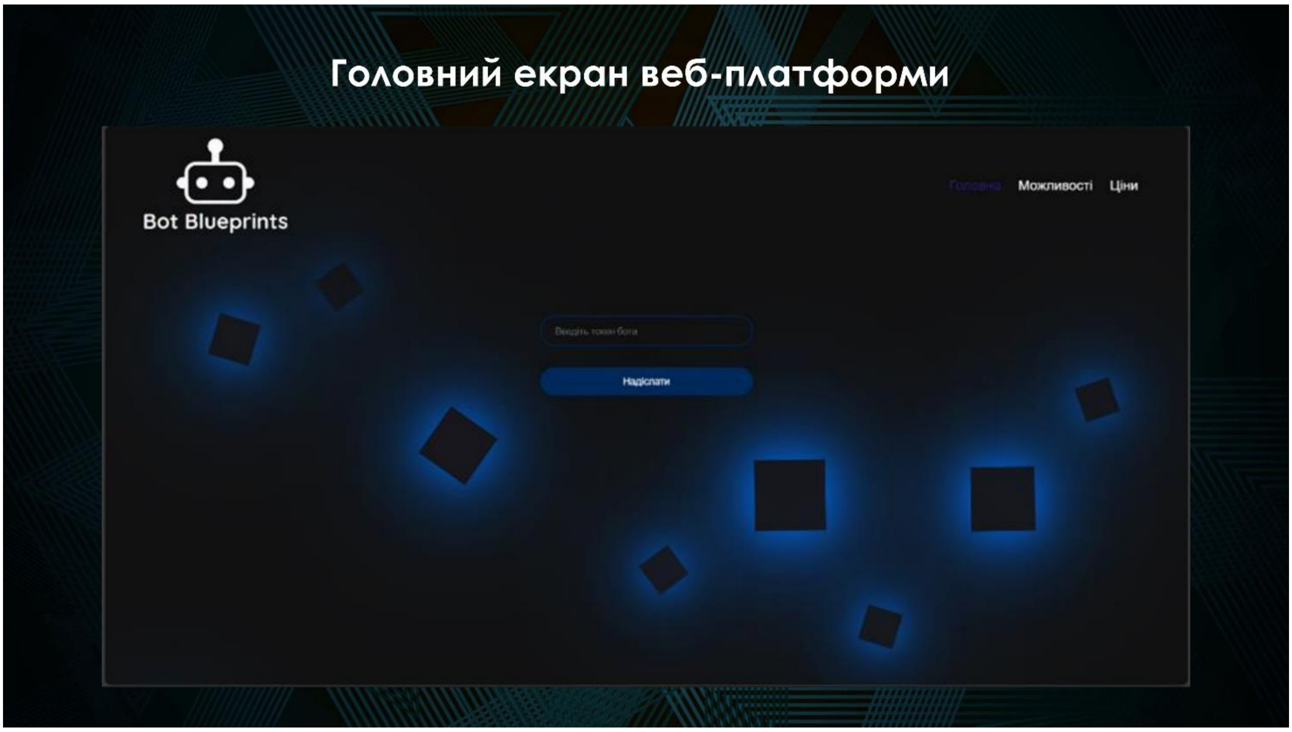

Рисунок Г.12 – Головний екран вебплатформи

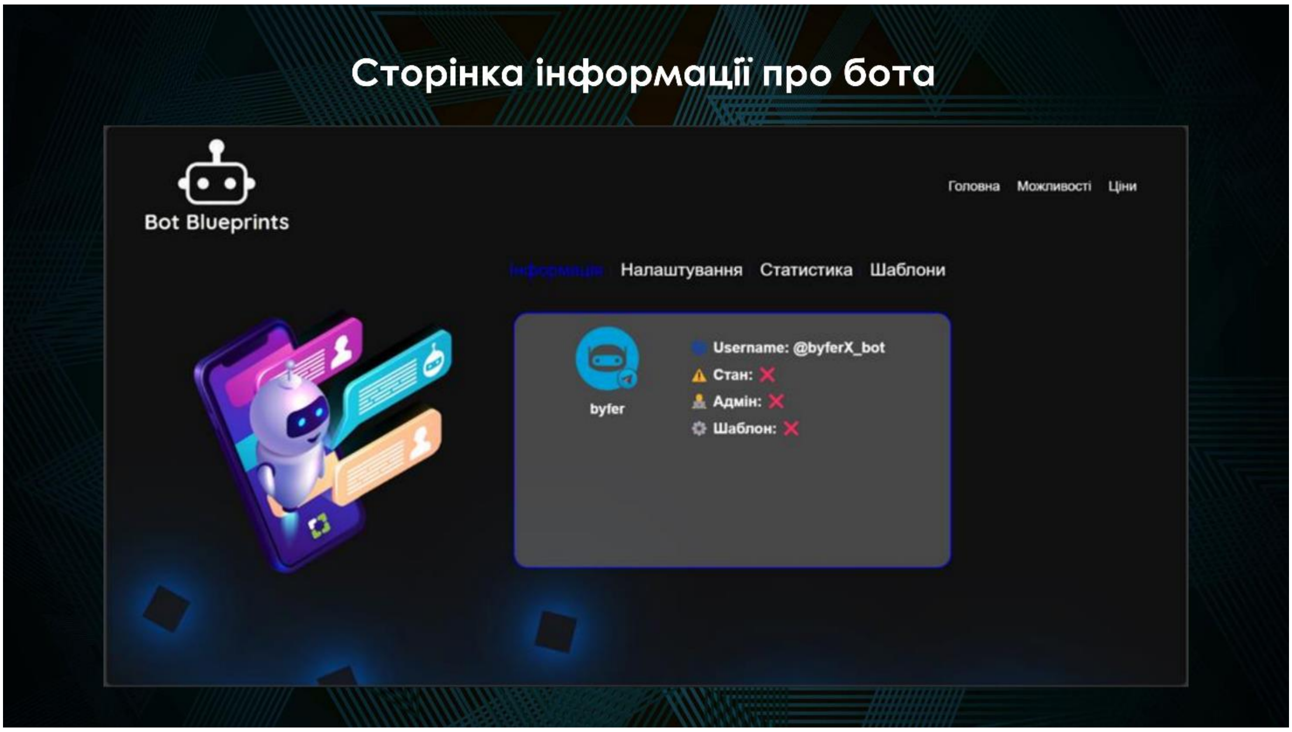

Рисунок Г.13 – Сторінка інформації про бота

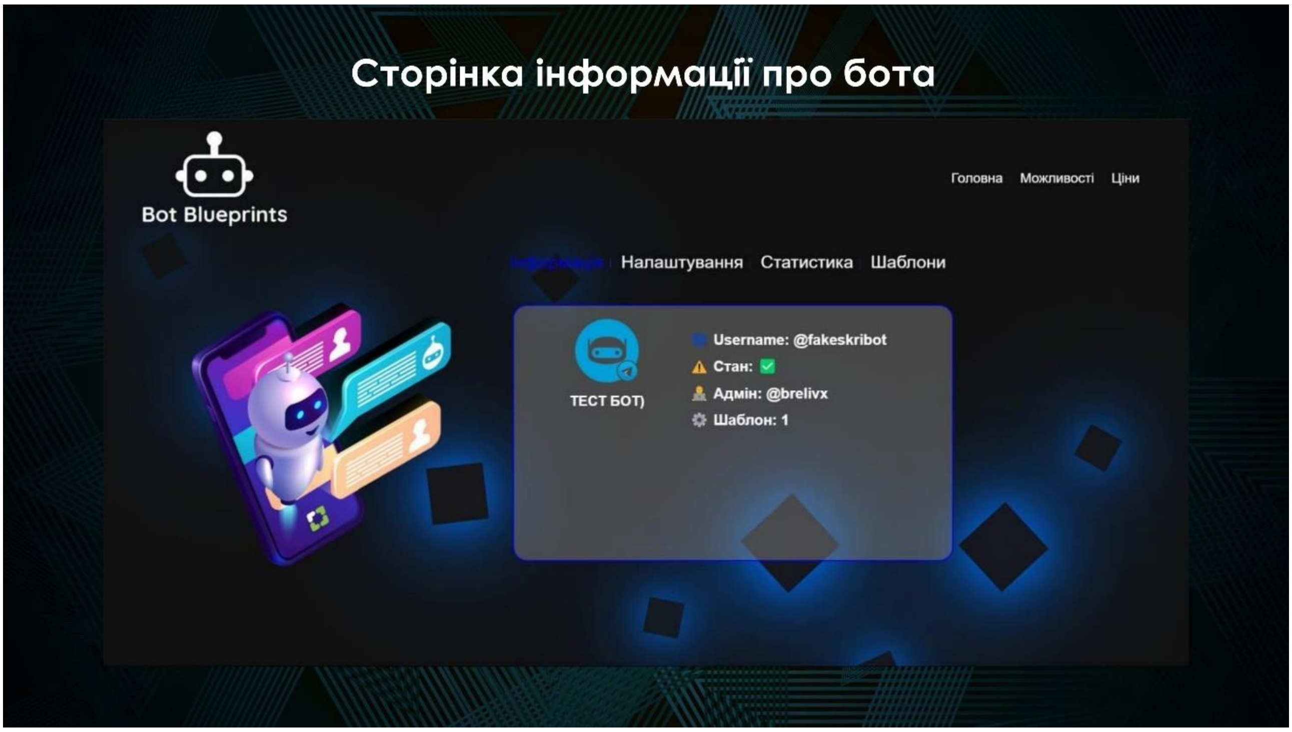

Рисунок Г.14 – Сторінка інформації про бота

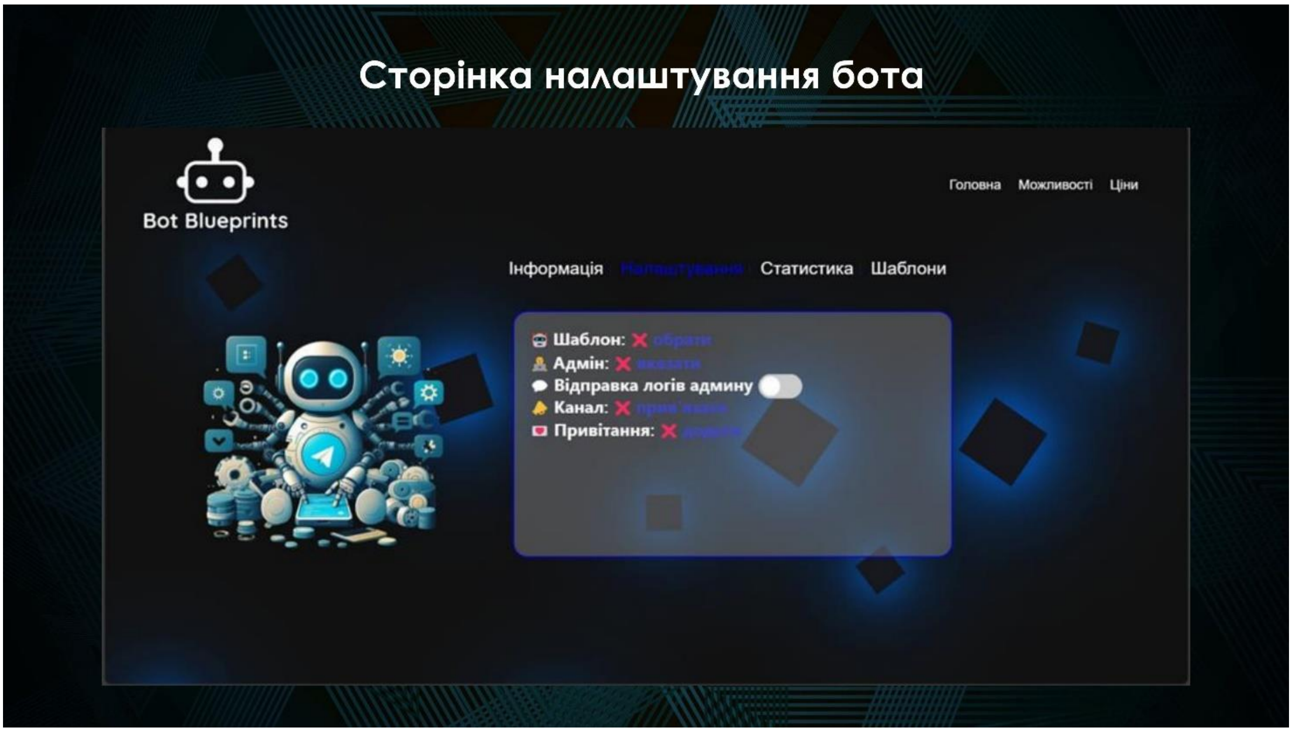

Рисунок Г.15 – Сторінка налаштування бота

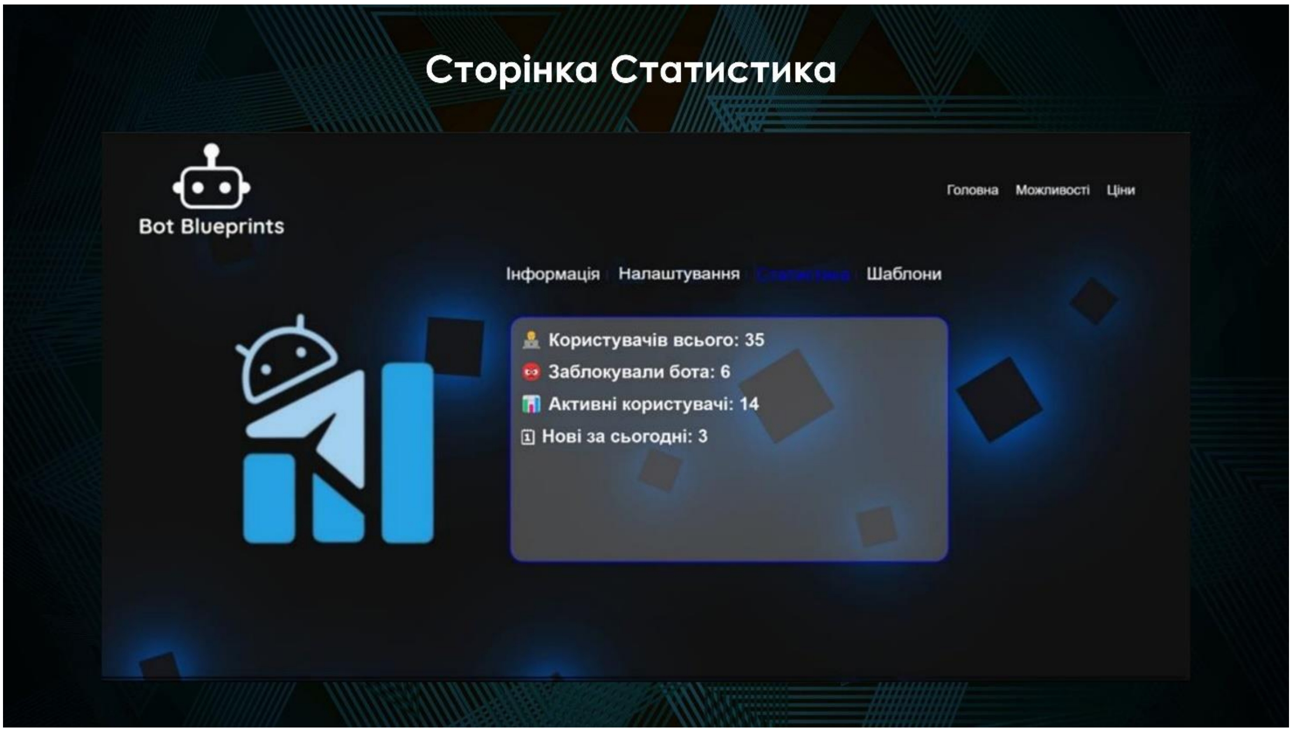

Рисунок Г.16 – Сторінка статистика

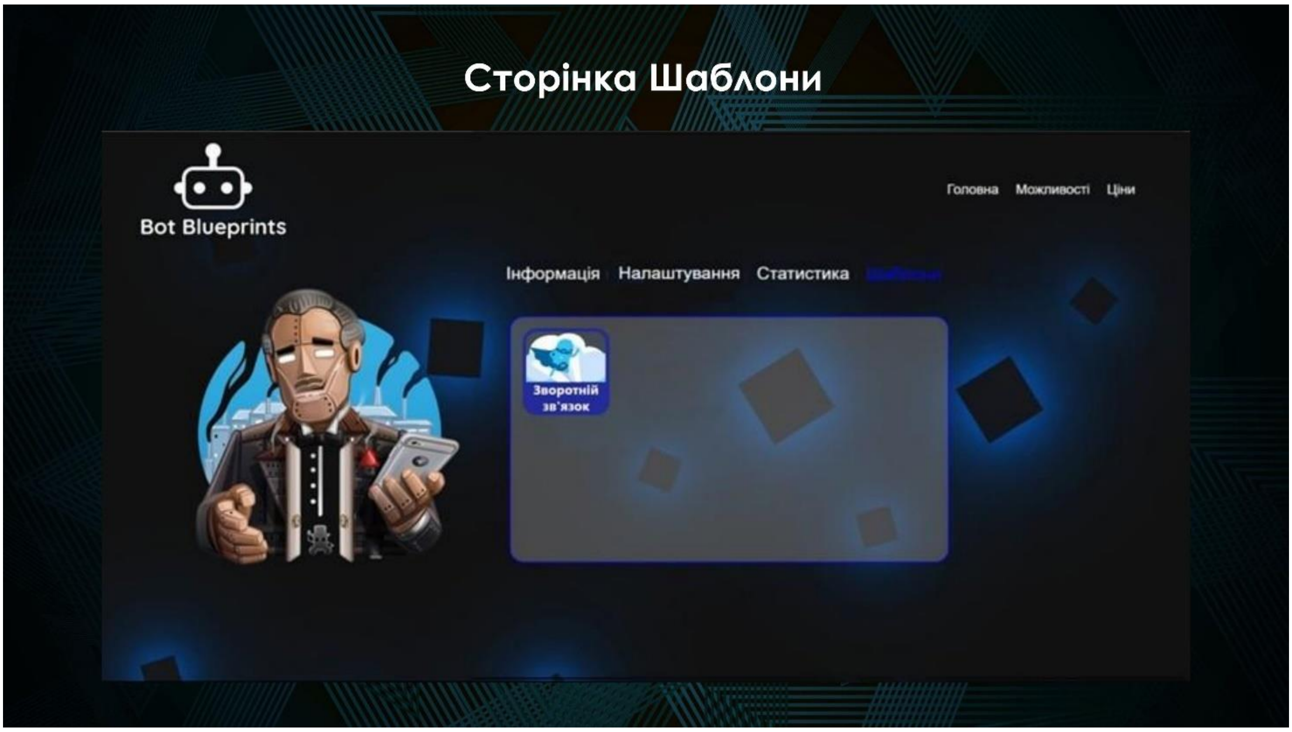

Рисунок Г.17 – Сторінка шаблони

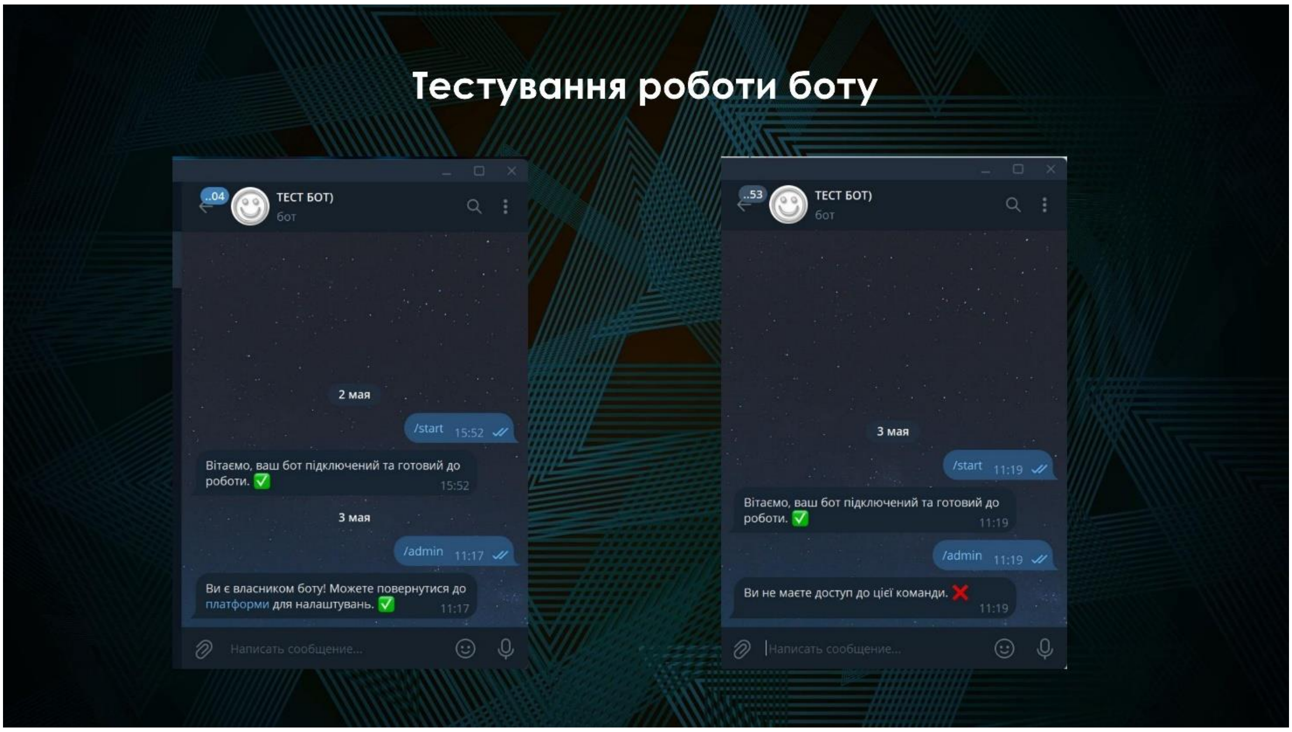

Рисунок Г.18 – Тестування роботи боту

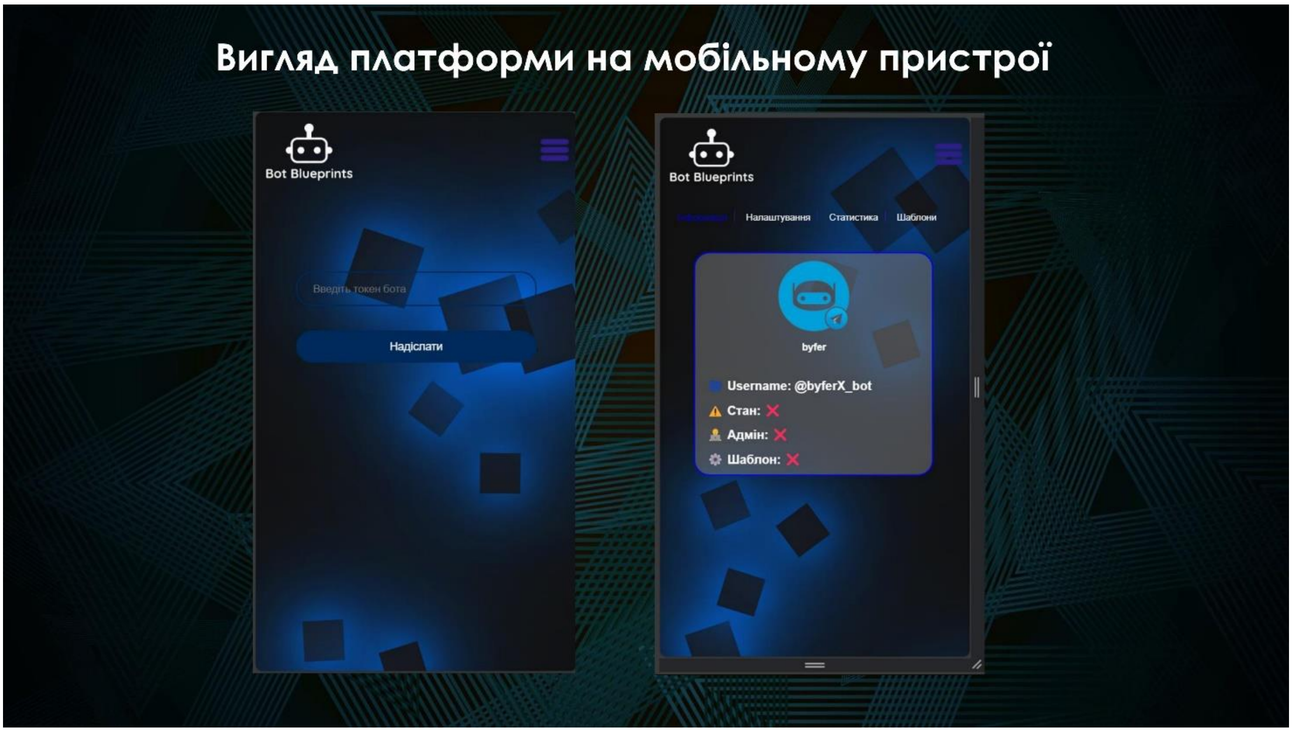

Рисунок Г.19 – Вигляд платформи на мобільному пристрої

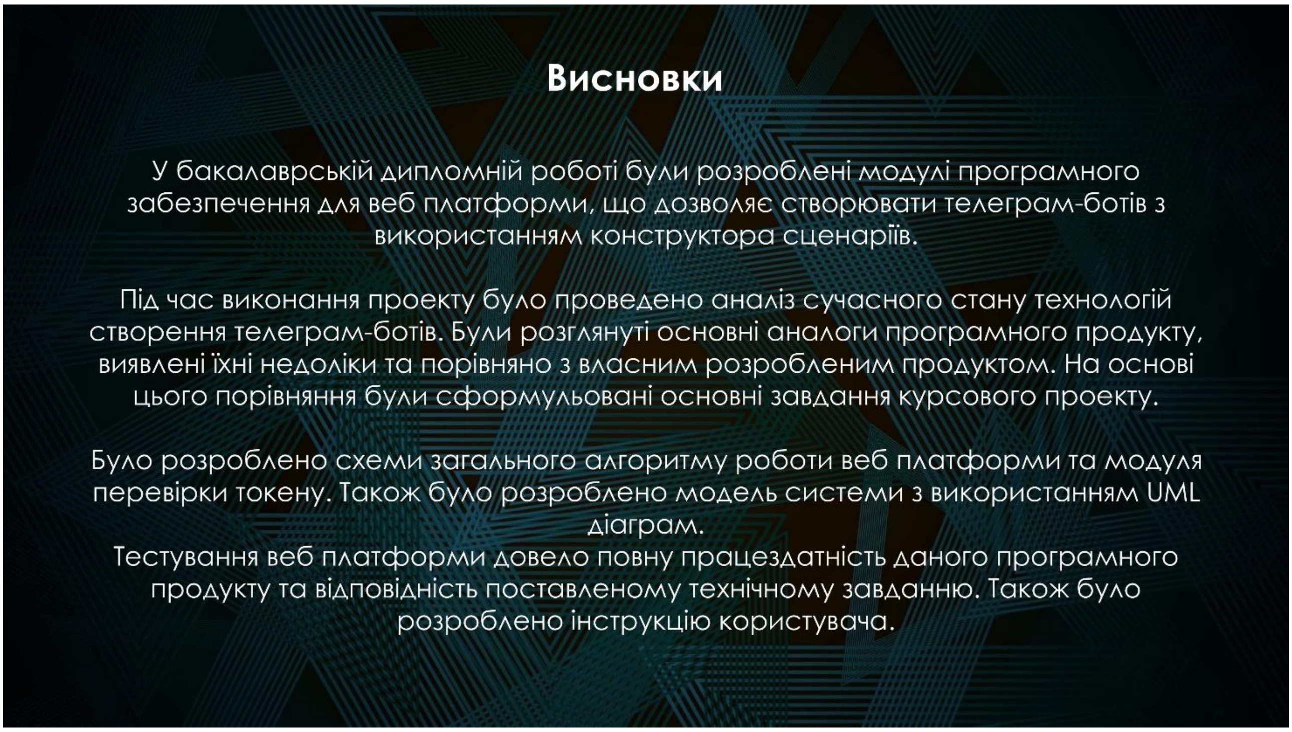

Рисунок Г.20 – Висновки

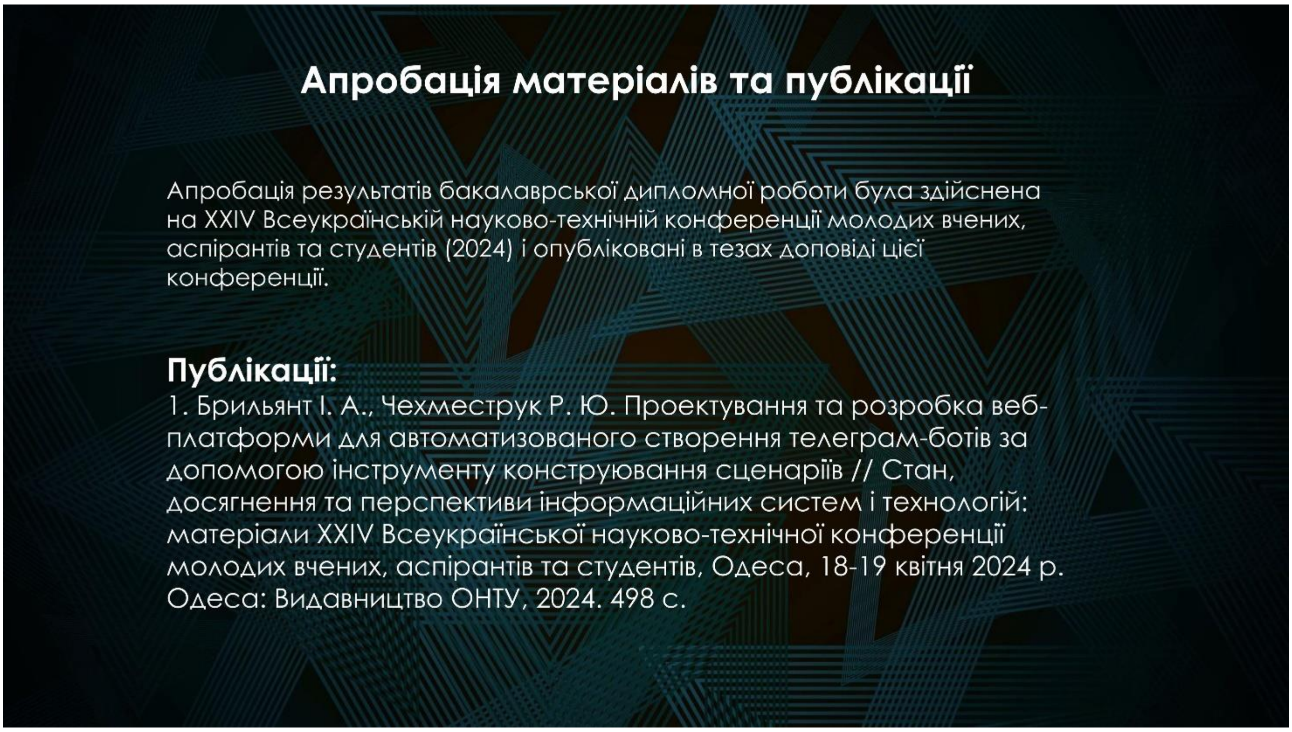

Рисунок Г.21 – Апробація матеріалів та публікація