Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

BACKEND-РОЗРОБКА СИСТЕМИ ТЕСТУВАННЯ ЧАТ-БОТІВ

BACKEND DEVELOPMENT OF THE CHATBOT TESTING SYSTEM

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗс-31
Кікець Микола Ігорович
_____
(підпис)

Керівник:
к.т.н., доцент
Суринович Олена Миколаївна
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» 06 2024 р.
к.т.н., доцент
Гарант освітньої програми:
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Н. Ліщина*
«*2*» *02* 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Кікецю Миколі Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *Backend-розробка системи тестування чат-ботів*

Керівник роботи: *к.т.н., доцент, Суринович Олена Миколаївна*
затверджені наказом закладу вищої освіти від «*30» грудня 2023 р.* № *477/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.

5. Перелік графічного матеріалу: *9 рисунків.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Суринович О. М.*		
Специфікації вимог до розробленої системи	*Суринович О. М.*		
Розробка об'єкта проєктування	*Суринович О. М.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту		3,4%	
Академічна доброчесність	*Ящук А. А.*		

7. Дата видачі завдання *«2» лютого 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*07.02.2024 р.*	вик.
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*27.02.2024 р.*	вик.
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*12.03.2024 р.*	вик
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*17.04.2024 р.*	вик
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2024 р.*	вик
6	*Тестування та налагодження об'єкта проектування*	*01.06.2024 р.*	вик
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*05.06.2024 р.*	вик

Здобувач вищої освіти _____ (підпис) *Кікець М. І.* (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) *Суринович О. М.* (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

ВІДГУК
КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач вищої освіти *Кікець Микола Ігорович*
(прізвище, ім'я, по батькові)
група *ІПЗс-31*

Тема кваліфікаційної роботи:
«Backend-розробка системи тестування чат-ботів»

Керівник *Суринович Олена Миколаївна*

Актуальність теми
У зв'язку з інтеграцією різних телеграм чат-ботів у сучасних фірмах, існує необхідність їх тестування на автоматичному рівні. Тому тема дослідження є на сьогодні дійсно актуальною
Об'єкт дослідження
Backend-розробка для системи тестування чат-ботів

Характеристика теоретичного рівня, наявності самостійних розробок і практичної значимості роботи:
Автор аналізує різні методи та засоби для розробки системи тестування чат-ботів. Визначає найкращі з них і обґрунтовує своє рішення. Зміст роботи відповідає обраній темі. Позитивними рисами кваліфікаційної роботи бакалавра є системність та послідовність викладення матеріалу, а також якісна його практична реалізація

Зауваження та недоліки: *Суттєвих недоліків в роботі не виявлено.*

Загальний висновок:
Кваліфікаційна робота бакалавра виконана на високому рівні. Представлена робота виконана відповідно до вимог, в якій логічно й послідовно описано хід виконання поставленого завдання. Пояснювальна записка відповідає стандартам до її оформлення.

Керівник _____ ***(Суринович О. М)***
(підпис) (прізвище, ім'я, по батькові)

«6» червня 2024 р.

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія
на кваліфікаційну роботу

Здобувач вищої освіти: Кікець Микола Ігорович

Тема: *Backend-розробка системи тестування чат-ботів*

Коротка характеристика кваліфікаційної роботи:

Кваліфікаційна робота бакалавра складається з трьох розділів, в яких проведено аналіз предметної області, здійснена чітка постановка завдань за темою роботи, проведено дослідження методів та засобів для розробки Backend системи, здійснено практичну реалізацію і проведено дослідження результату ефективності системи, а також здійснено аналіз отриманих результатів.

Самостійні розробки і пропозиції автора: *Автором було самостійно розроблено Backend систему на базі фреймворку Remix.Run.*

Практичне значення роботи

Можливість застосування автоматизованої системи тестування чат-ботів у сучасних підприємствах

Недоліки: : *Суттєвих недоліків в роботі не виявлено.*

Загальний висновок

Кваліфікаційна робота бакалавра виконана на високому рівні. Робота виконана з розумінням предмету. Матеріал викладено в чіткій та зручній формі. Вважаю, що кваліфікаційна робота відповідає вимогам до випускних робіт і заслуговує оцінки «відмінно», а її авторові, студенту Кікецю М.І., може бути присвоєний кваліфікаційний рівень "бакалавр" зі спеціальності «Інженерія програмного забезпечення»

Рецензент: *Заблоцький В. Ю., к. т. н., доц., завідувач кафедри електроніки та телекомунікацій*

Рецензент кваліфікаційної роботи _____
(підпис)

«6» червня 2024 р.

АНОТАЦІЯ

Кікець М. І. Backend-розробка системи тестування чат-ботів. Рукопис.

Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2024.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі здійснено аналіз сучасного стану проблеми автоматизації тестування чат-ботів і сформульовано завдання. В другому розділі визначено вимоги до розроблюваної системи, а також засоби, методи і алгоритми розробки. У третьому розділі розроблено Backend систему для проведення автоматизованого тестування чат-ботів. У висновках підведено підсумки дослідження. Було створено сервер, що дозволяє автоматизувати тестування в чат-ботах telegram, що є корисним інструментом для розробників і тестувальників і може бути використано в реальних сценаріях.

Ключові слова: Backend, чат-бот, автоматизована система, тестування, UML, Telegram.

ABSTRACT

Kikets M. I. Backend Development of the Chatbot Testing System. Manuscript.

Bachelor's qualifying thesis of the educational program “Software Engineering”. Lutsk National Technical University. Lutsk, 2024.

The bachelor's qualifying thesis consists of an introduction, three sections, conclusions, a list of used sources and annexes.

In the first chapter, an analysis of the current state of the problem of automation testing of chat bots and the task was formulated. The second section defines the requirements for the developed system, as well as means, methods and development algorithms. In the third chapter, a Backend system for automation testing of chat bots was developed and tested. The results of the research are summarized. A server was created that allows you to automate testing in telegram chatbots, which is a useful tool for developers and testers and can be used in real scenarios.

Keywords: Backend, chat-bot, automated system, testing, UML, Telegram.

ЗМІСТ

ВСТУП

Актуальність теми «Backend-розробка системи тестування чат-ботів» є надзвичайно важливою в контексті сучасного розвитку цифрових технологій. Telegram, як один із найпопулярніших месенджерів у світі, має мільйони активних користувачів та надає розробникам потужні інструменти для створення ботів, які виконують різноманітні функції: від автоматизації бізнес-процесів до інтерактивних ігор та сервісів. Зростання кількості таких ботів та їхнього використання у повсякденному житті підвищує вимоги до їх якості та надійності.

Успіх телеграм ботів у значній мірі залежить від їхньої стабільності та безперебійної роботи, адже користувачі очікують високого рівня продуктивності та швидкого реагування. Недоліки в роботі ботів можуть призвести до негативних відгуків, втрати користувачів та шкоди для репутації розробників. Це особливо критично для ботів, які використовуються у фінансових та інших важливих додатках, де помилки можуть мати серйозні наслідки, включаючи фінансові втрати та ризики для безпеки даних користувачів.

Тестування телеграм ботів є ключовим етапом розробки, що дозволяє виявляти та усувати баги до запуску, забезпечуючи оптимальну взаємодію з користувачами. Ефективне тестування сприяє покращенню користувацького досвіду, що є важливим фактором для залучення та утримання користувачів. Крім того, економія ресурсів та оптимізація процесів розробки є суттєвими перевагами, оскільки виявлення та виправлення помилок на ранніх етапах є менш затратним, ніж після впровадження продукту.

Автоматизація тестування значно підвищує ефективність цього процесу, дозволяючи швидко і точно перевіряти функціональність ботів за допомогою імітації дій реальних користувачів. Це дозволяє розробникам зосередитися на вдосконаленні функціоналу та якості продукту, мінімізуючи рутинні завдання. В умовах зростаючих загроз кіберзлочинності, безпека та конфіденційність

даних користувачів є першочерговими завданнями. Тестування допомагає виявляти вразливості та забезпечувати високий рівень захисту, що є особливо важливим для ботів, які працюють з чутливою інформацією.

Крім того, інтеграція телеграм ботів з іншими системами та сервісами вимагає комплексного підходу до тестування, щоб забезпечити коректну взаємодію між усіма компонентами. Усе це підкреслює важливість створення ефективних систем тестування, здатних гарантувати високу якість, стабільність та безпеку телеграм ботів. Таким чином, розробка та впровадження таких систем є необхідним кроком для забезпечення успішного функціонування ботів у сучасному цифровому середовищі, що робить цю тему надзвичайно актуальною для дослідження.

Мета дослідження – розробити Backend-платформу для тестування чат-ботів.

Завдання до виконання:

1) дослідити існуючі методи та інструменти для тестування телеграм ботів, проаналізувати їхні переваги, недоліки та обмеження;

2) розробити концепцію системи автоматизованого тестування, визначити основні функції та вимоги до неї;

3) вибрати мови програмування, фреймворки та інструменти, які найбільш підходять для вирішення поставлених завдань;

4) реалізувати систему автоматизації тестування;

5) провести тестування системи.

Об’єкт дослідження – Backend-платформа для тестування чат-ботів.

Предмет дослідження – технології розробки альтернативних платформ автоматизованого тестування.

Апробація результатів дослідження. Здолбіцька Н. В., Савич І. М., Кікець М. І. Розробка сервісу автоматизованого тестування чат-ботів. IX Міжнародна науково-практична конференція «Інформаційні технології в освіті, науці і виробництві» тези доп., 25-26 травня 2023 р. Луцьк. 2023. С. 168-170 [1]. Конкурс стартап проектів ЛНТУ.

РОЗДІЛ 1
АНАЛІЗ BACKEND-РОЗРОБКИ СИСТЕМИ ТЕСТУВАННЯ ЧАТ-БОТІВ

1.1 Аналіз сучасного стану проблеми

Сучасний стан проблеми тестування телеграм ботів характеризується значним зростанням їх популярності та складності, що підвищує вимоги до якості та надійності цих ботів. Попит на телеграм боти постійно зростає, оскільки вони широко використовуються для автоматизації бізнес-процесів, інтерактивних додатків та багатьох інших завдань. Це зумовлює необхідність ретельного тестування, адже помилки у роботі ботів можуть призвести до негативних наслідків, включаючи втрату користувачів та шкоду для репутації розробників.

Основні виклики включають складність функціоналу, забезпечення безпеки даних, сумісність з різними платформами, автоматизацію тестування та постійну підтримку й оновлення ботів. Сучасні телеграм боти можуть виконувати широкий спектр функцій, що вимагає ретельного тестування різноманітних сценаріїв використання. Це трудомісткий процес, що вимагає значних ресурсів та компетенції. Крім того, в умовах сучасного цифрового середовища питання безпеки та конфіденційності даних є критично важливими. Телеграм боти часто працюють з чутливою інформацією, і виявлення та усунення вразливостей є складним завданням.

Сумісність з різними платформами також додає додаткової складності до процесу тестування. Телеграм боти можуть використовуватися на різних пристроях і операційних системах, що вимагає забезпечення їхньої коректної роботи в різних середовищах. Автоматизація тестування є важливим аспектом, що дозволяє підвищити ефективність процесу, але створення автоматизованих тестів для телеграм ботів є складним завданням, що вимагає спеціальних інструментів та методик.

Постійне оновлення ботів для додавання нових функцій та виправлення багів вимагає регулярного тестування та забезпечення стабільної роботи після

кожного оновлення. Це може бути складним і ресурсоємним процесом. На даний момент існує кілька підходів до тестування телеграм ботів, таких як ручне тестування, автоматизоване тестування, стрес-тестування та тестування безпеки. Однак, всі вони мають свої обмеження та проблеми.

Ручне тестування, хоча і є одним з найпростіших методів, є надзвичайно трудомістким та схильним до людських помилок. Це робить його менш ефективним для великих проектів або складних ботів з багатофункціональним інтерфейсом. Автоматизоване тестування, хоча і значно підвищує точність і ефективність, вимагає створення спеціальних скриптів та інструментів, що може бути технічно складним і потребує значних початкових витрат ресурсів.

Стрес-тестування дозволяє перевірити, як бот працює під високим навантаженням, що є важливим для виявлення проблем з продуктивністю. Однак, налаштування інструментів для стрес-тестування, таких як JMeter або Gatling, також вимагає значних знань та ресурсів. Тестування безпеки, яке включає виявлення вразливостей та забезпечення захисту даних користувачів, є критично важливим, особливо для ботів, що обробляють конфіденційну інформацію. Інструменти для цього, такі як OWASP ZAP або Burp Suite [2], можуть бути корисними, але також вимагають спеціальних знань.

Обмежені можливості автоматизації та відсутність стандартизації ускладнюють створення уніфікованих рішень та порівняння результатів тестування. Існує велика різноманітність методів і підходів до тестування телеграм ботів, але відсутність єдиних стандартів ускладнює забезпечення послідовності та якості тестування. Це призводить до непослідовності в якості та ефективності тестування, що може негативно вплинути на кінцевий продукт.

Обмежена підтримка нових функцій телеграм та висока ресурсомісткість тестування також є значними перешкодами. Телеграм постійно оновлює свою платформу, додаючи нові функції, які потребують постійного оновлення інструментів для тестування. Відставання у підтримці нових функцій може призводити до проблем з тестуванням та виявленням помилок. Тестування телеграм ботів є ресурсоємним процесом, що вимагає значних обчислювальних

потужностей та часу, особливо при використанні автоматизованих тестів або стрес-тестування.

Забезпечення реалістичності тестів, включаючи імітацію реальних сценаріїв використання, є складним завданням, що вимагає врахування різноманітних факторів, таких як поведінка користувачів, мережеві умови та можливі зовнішні впливи. Імітація реальних умов використання дозволяє виявити потенційні проблеми на ранніх етапах, але це також є складним і ресурсомістким процесом.

Таким чином, забезпечення високої якості, стабільності та безпеки телеграм ботів вимагає подальшого розвитку інструментів та методик тестування. Впровадження нових технологій, таких як штучний інтелект та машинне навчання, може значно покращити процес тестування та забезпечити високу якість телеграм ботів. Штучний інтелект може бути використаний для автоматизації складних тестових сценаріїв та виявлення аномалій у роботі ботів, що дозволить значно підвищити ефективність тестування.

Подальший розвиток та вдосконалення інструментів для тестування телеграм ботів є необхідним для забезпечення їх стабільної та безпечної роботи. Це включає розробку нових методів автоматизації тестування, створення стандартів для забезпечення якості та послідовності тестування, а також використання передових технологій для покращення процесу тестування. Успішне вирішення цих викликів дозволить забезпечити стабільну та безпечну роботу телеграм ботів, що є критично важливим для їх успішного функціонування в сучасному цифровому середовищі.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Після проведеного аналізу було виявлено, що тестування телеграм ботів є складним і багатогранним процесом, який потребує спеціальних інструментів та методик для забезпечення їхньої якості, стабільності та безпеки. Завданням кваліфікаційної роботи бакалавра є розробка системи для автоматизованого

тестування телеграм ботів, яка дозволить підвищити якість їхньої роботи та зменшити кількість помилок, що можуть виникати під час використання. Це завдання передбачає вирішення кількох ключових аспектів, кожен з яких відіграє важливу роль у створенні надійної системи тестування.

Поставлені наступні завдання до виконання:

1) дослідити існуючі методи та інструменти для тестування телеграм ботів, проаналізувати їхні переваги, недоліки та обмеження;

2) розробити концепцію системи автоматизованого тестування, визначити основні функції та вимоги до неї, розробити ефективні алгоритми для автоматизації процесів тестування;

3) вибрати мови програмування, фреймворки та інструменти, які найбільш підходять для вирішення поставлених завдань;

4) реалізувати систему, забезпечивши її інтеграцію з Telegram чат-ботами та можливість автоматизації тестування;

5) провести тестування системи.

Виконання цих завдань дозволить створити ефективну систему автоматизованого тестування телеграм ботів, що сприятиме підвищенню їхньої якості та надійності. Така система дозволить розробникам ботів швидше виявляти та виправляти помилки, знижувати ризики, пов’язані з використанням ботів, та забезпечувати високу якість послуг для користувачів. Це є критично важливим для їх успішного функціонування в сучасному цифровому середовищі, де вимоги до якості програмного забезпечення постійно зростають.

Висновки до розділу 1

Було здійснено аналіз сучасного стану проблеми тестування телеграм ботів, що виявив складність і багатогранність цього процесу, а також необхідність використання спеціальних інструментів та методик для забезпечення їхньої якості, стабільності та безпеки. Завданням кваліфікаційної роботи бакалавра є розробка системи для автоматизованого тестування

телеграм ботів, яка дозволить підвищити якість їхньої роботи та зменшити кількість помилок, що можуть виникати під час використання.

Необхідно дослідити існуючі методи та інструменти для тестування телеграм ботів, провести аналіз їхніх переваг, недоліків та обмежень задля формування уявлення про поточний стан та виявити потребу в удосконаленні процесу тестування.

Проектування архітектури системи та вибір технологій для її реалізації є ключовими для забезпечення високої продуктивності, надійності та можливості масштабування системи. Потрібно вибрати мови програмування, фреймворки та інструменти, які найбільш підходять для вирішення поставлених завдань.

Наступним необхідно реалізувати систему і провести її тестування перевіривши її ефективність.

Виконавши завдання, буде створено ефективну систему автоматизованого тестування чат-ботів, для підвищення їхньої якості та надійності.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Провівши аналіз проблематики автоматизації тестування чат-ботів необхідно визначити технологічний стек для розробки Backend-у. В наш час є великий вибір мов та технологій для розробки програмного забезпечення. Проаналізувавши існуючі мови програмування, ми змогли визначити кілька мов, які могли б підійти для подальшої розробки:

- Java – це об’єктно-орієнтована мова програмування, створена Sun Microsystems у 1995 році [3]. Сьогодні він вважається популярною кросплатформною мовою програмування, використовуваним при написанні різних типів додатків: веб-сайтів, комп’ютерних і мобільних додатків та інших.;
- Python – це високорівнева інтерпретована об’єктно-орієнтована мова програмування із суворою динамічною типізацією, розроблена в 1990 році [4]. Вважається універсальним рішенням для написання великої кількості додатків. Сьогодні використовується для написання веб-сервісів, штучного інтелекту, обробці великої кількості даних, написання чат-ботів, інтеграції різних сервісів та інших типів робіт.;
- JavaScript – це динамічна, об’єктно-орієнтована прототипна мова програмування, яка утворена у 1995 році і є реалізацією стандарту ECMAScript [5]. Популярне рішення при створення кросплатформ, веб-додатків, універсальних систем і телеграм чат-ботів.

Варто звернути увагу що при формуванню веб-сайтів необхідно також проаналізувати методи формування сторінок. Існують три методи формування веб-сторінки [6]:

- Client-Side Render (з англ. – рендеринг на стороні клієнта) – це процес передачі інструкцій відображення продукту на сторону клієнта і формування сторінки за допомогою потужностей клієнтського персонального комп’ютера.

Найчастіше використовується при реалізації месенджерів, оскільки таке рішення дозволяє сформувати сторінку, без необхідності його повторного оновлення на сервері;

- Server-Side Render (з англ. – рендеринг на серверній стороні) – це формування сторінки відображення для клієнта на сервері, де знаходиться продукт і подальша передача її. Популярне рішення для правильного формування SEO, безпеки даних, а також успішної передачі даних;
- Pre-render (з англ. – попередній рендер) – це відокремлений тип формування сторінок, який відрізняється від попередніх методів способом надсилання даних. Pre-render маючи скелет-заготовку сторінки має можливість відразу відправити відповідь до клієнта, що пришвидшує відповідь від сервера в рази.

Також необхідно провести аналіз API Telegram, який надає потужні інструменти для розробників, які бажають створювати боти та інтеграції з платформою. Воно складається з кількох ключових компонентів, включаючи Bot API та Telegram API [7]. Bot API дозволяє створювати і керувати ботами, забезпечуючи доступ до функціональності платформи Telegram, такої як обробка повідомлень, управління чатами, надсилання медіа, отримання оновлень та інше. Цей API значно полегшує процес створення ботів, дозволяючи розробникам зосередитися на логіці та функціональності їхніх додатків.

Один з основних аспектів Bot API – його простота і зручність використання. API використовує стандартний протокол HTTPS для взаємодії між сервером бота і Telegram серверами, що забезпечує високу сумісність і легкість інтеграції. Для початку роботи з Bot API розробникам потрібно лише створити бота через BotFather, спеціальний бот у Telegram, який генерує унікальний токен для доступу до API. Використовуючи цей токен, розробники можуть виконувати різноманітні запити, включаючи надсилання і отримання повідомлень, керування чатами і користувачами, а також інтеграцію з іншими сервісами.

Ще одним важливим аспектом є гнучкість і розширюваність Bot API. Він підтримує різні типи медіа, такі як текст, зображення, відео, документи, а також інтерактивні елементи, такі як кнопки і інлайн-клавіатури. Це дозволяє створювати багатофункціональні боти з різноманітним інтерфейсом користувача. Крім того, API забезпечує підтримку вебхуків і довгого опитування (рис. 2.1), що дозволяє розробникам вибирати найбільш підходящий метод отримання оновлень в залежності від їхніх потреб і ресурсів.

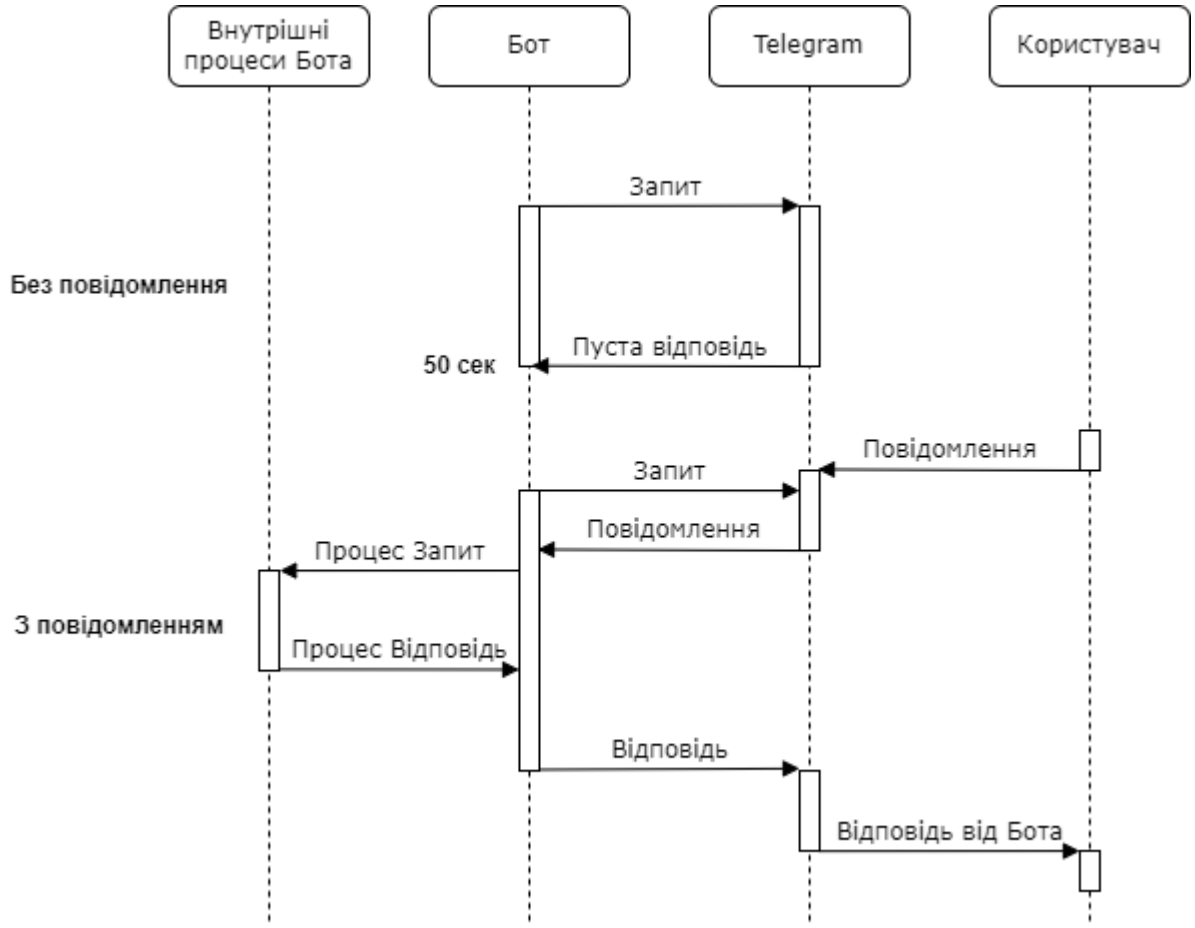


Рисунок 2.1 – Схема комунікації Телеграму і ботів в режимі довгого опитування

Нефункціональні вимоги включають здатність системи обробляти понад 1000 запитів від різних ботів одночасно, забезпечення безпеки зберігання сценаріїв і даних користувачів, а також надійність у витримуванні навантаження від великої кількості запитів. Хоча швидкість обробки запитів не

є пріоритетною, система повинна бути достатньо ефективною для роботи з великими обсягами даних.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для розробки системи автоматизованого тестування телеграм ботів необхідно ретельно підібрати відповідні інструменти, методи та алгоритми, які забезпечать ефективність, надійність та масштабованість системи. Нижче наведено обґрунтування вибору технологій і методів, що будуть використані в рамках цієї роботи.

Для розробки Backend-у автоматизованої системи тестування було вибрано наступні технології:

1) TypeScript: забезпечує типізацію та інші можливості, які підвищують надійність і безпеку коду [8]. Це дозволяє розробникам виявляти помилки на ранніх етапах розробки та підвищує продуктивність;

2) Remix.Run – є сучасним фреймворком для розробки веб-застосунків, який забезпечує високу продуктивність і зручність у використанні [9]. Він дозволяє легко створювати серверні і клієнтські частини програми, що є необхідним для клієнт-серверної архітектури;

3) MongoDB – документо-орієнтованою базою даних, яка забезпечує високу продуктивність і масштабованість [10]. Вона добре підходить для зберігання JSON-даних, що є основним форматом сценаріїв тестування у цій системі;

4) Telegraf.js (Telegram Lib) – бібліотека для роботи з API Telegram. Вона забезпечує зручний інтерфейс для інтеграції з Telegram і підтримує всі необхідні функції для створення та тестування телеграм ботів [11].

Також необхідно провести опитування та анкетування задля формування вимог від розробників та користувачів, щоби забезпечити чітке розуміння необхідної функціональності та нефункціональних аспектів системи.

Використання UML (Unified Modeling Language) дозволить візуалізувати структуру та динаміку системи. Діаграми класів, діаграми послідовностей та діаграми розгортання забезпечують зрозуміле представлення архітектури системи та її компонентів.

Після опитування було зрозуміло що використання нотації у форматі JSON є простішим для розробників і візуально зрозумілий. Для обробки сценаріїв тестування у форматі JSON необхідно використовувати алгоритми парсингу, які дозволяють розбирати JSON-дані та перетворювати їх у відповідні структури даних для подальшої обробки. Це забезпечує зручність у роботі зі сценаріями та дозволяє автоматизувати процес тестування.

Основні вимоги до системи автоматизованого тестування телеграм ботів включають:

1) функціональні вимоги:

- підтримка введення інформації про сценарії тестування у форматі JSON;
- зміна існуючих сценаріїв тестування;
- імітація взаємодії з телеграм ботами, включаючи надсилання та отримання повідомлень;
- автоматичне виявлення помилок у відповідях ботів та генерація звітів;
- архівація сценаріїв тестування та результатів;
- пошук за сценаріями та результатами тестування;
- забезпечення захисту і безпеки даних, розмежування прав доступу до сценаріїв та звітів;
- забезпечення цілісності бази даних;

2) нефункціональні вимоги:

- продуктивність, задля здатності системи обробляти понад 1000 запитів від різних ботів одночасно;
- безпека зберігання сценаріїв і даних користувачів;
- надійність витримувати навантаження від великої кількості запитів;

− інтуїтивно зрозумілий інтерфейс для розробників і звичайних користувачів.

Висновки до розділу 2

Було здійснено аналіз існуючих мов програмування, фреймворків та технологій для розробки системи автоматизованого тестування телеграм ботів. Згідно з цим аналізом, було обрано оптимальні інструменти та методи, які найкраще відповідають вимогам проекту та забезпечують високу ефективність, надійність і масштабованість системи.

Вибір TypeScript обґрунтовано його можливістю забезпечити надійність і безпеку коду через типізацію, що є важливим аспектом при розробці складних програмних систем. Використання Remix.Run дозволяє створити зручний і продуктивний інтерфейс користувача, що відповідає сучасним вимогам до веб-застосунків. MongoDB обрана для зберігання даних завдяки її продуктивності та зручності у роботі з JSON-форматом, що є ключовим у даному проекті.

Застосування бібліотеки Telegraf.js (Telegram Lib) забезпечує інтеграцію з API Telegram, що дозволяє ефективно імітувати взаємодію з телеграм ботами та автоматизувати процес тестування. Це дозволяє розробникам зосередитися на створенні якісних сценаріїв тестування, не витрачаючи зайвий час на реалізацію базової функціональності.

Обрані методи виявлення та аналізу вимог, зокрема інтерв’ю та анкетування, дозволяють точно визначити потреби розробників і користувачів, що забезпечує створення системи, яка максимально відповідає їхнім очікуванням. Використання UML для моделювання системи забезпечує чітке розуміння структури та динаміки системи, що полегшує процес її розробки та подальшої підтримки.

Алгоритми обробки сценаріїв тестування, повідомлень та генерації звітів дозволяють автоматизувати основні процеси тестування, підвищуючи ефективність і точність виявлення помилок у роботі ботів. Це, у свою чергу,

сприяє покращенню якості телеграм ботів і зниженню ризиків, пов’язаних з їхнім використанням.

Вимоги до системи, включаючи функціональні та нефункціональні аспекти, були визначені на основі аналізу потреб користувачів та специфіки проекту. Це дозволяє створити систему, яка здатна обробляти великий обсяг запитів, забезпечувати безпеку даних та зручність використання, що є критично важливими для її успішної експлуатації.

Таким чином, було вибрано оптимальні засоби, методи та алгоритми для розробки системи автоматизованого тестування телеграм ботів, що дозволяє забезпечити її високу ефективність, надійність та відповідність сучасним вимогам до програмного забезпечення.

РОЗДІЛ 3
РОЗРОБКА BACKEND СИСТЕМИ ТЕСТУВАННЯ ЧАТ-БОТІВ

3.1 Практична реалізація об’єкта проєктування

Практична реалізація об’єкта проєктування включає кілька етапів, кожен з яких спрямований на створення функціональної та надійної системи автоматизованого тестування телеграм ботів. Цей процес включає проектування архітектури системи, розробку коду, тестування та впровадження, а також постійне вдосконалення та підтримку системи.

Під час розробки системи створювалася структура, аналогічна до Telegram Bot API [11], для ефективної взаємодії з ботами та серверами Telegram. Після аналізу посилань було виявлено кілька ключових частин, які складають цю структуру (рис. 3.1):

- базове посилання – це основне посилання, куди боти надсилають свої запити. Воно визначає адресу сервера, з яким бот спілкується;
- ключове слово – використовується для розпізнавання типу запиту та розділення інформації між користувачами та чат-ботами. Воно допомагає серверам Telegram ідентифікувати, які саме дії потрібно виконати з отриманою інформацією;
- ідентифікатор бота – це унікальний цифровий ключ, який визначає порядковий номер створеного бота в системі. Кожен бот має свій власний ідентифікатор, який використовується для розпізнавання його запитів на сервері;
- авторизаційний ключ – це хеш-ключ, який використовується для авторизованого виконання методів на стороні серверів Telegram. Він забезпечує безпечну ідентифікацію та авторизацію ботів під час взаємодії з серверами Telegram;
- метод – це посилання на виконання необхідних функцій для функціонування бота. Метод визначає, яку саме дію потрібно виконати з отриманою інформацією і які дані передати для обробки.

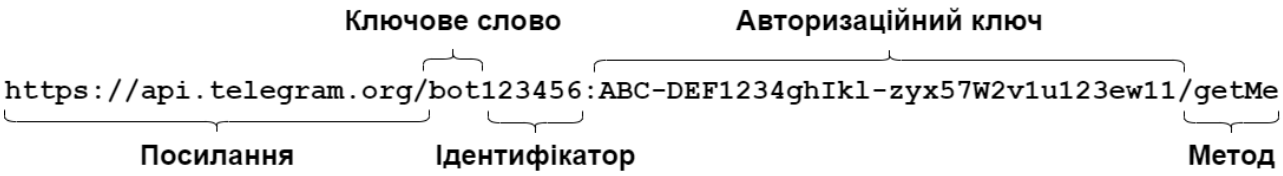


Рисунок 3.1 – Схема-аналіз посилання Телеграм чат-ботів

Ці компоненти утворюють основу для взаємодії ботів та серверів Telegram, забезпечуючи правильне і ефективне виконання запитів та обмін даними між ними. Кожна частина URL має своє специфічне призначення і сприяє чіткій ідентифікації запитів та відповідей, що відправляються та отримуються ботами. Така структуризація дозволяє забезпечити безперебійну роботу ботів та надійну передачу інформації, що є критично важливим для їх функціонування.

Задля реалізації подібної структури Remix.Run дозволяє використовувати структуру папок як блоки посилань [13] (рис. 3.2). Це означає, що кожна папка може відповідати певному сегменту URL, дозволяючи легко організовувати і керувати маршрутами в додатку. Така структура дає можливість використовувати посилання максимально подібно до Telegram, де кожен сегмент URL має своє визначене значення і функцію.

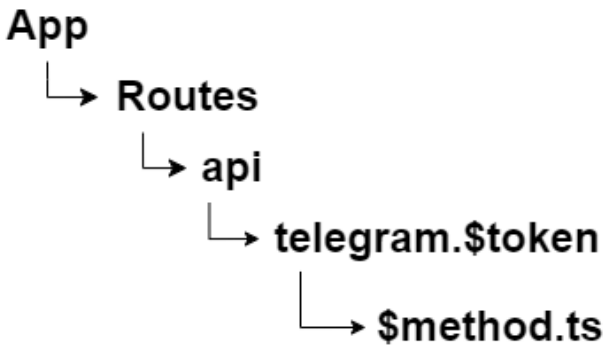


Рисунок 3.2 – Структура папок посилання в Remix.Run

Подана структура також надає можливість отримати токен і метод, необхідний для запуску і роботи бота. Завдяки чіткій і логічній організації

даних, розробники можуть швидко знайти необхідну інформацію про створеного бота, включаючи його ідентифікатор, токен доступу і доступні методи. Це значно спрощує процес інтеграції та управління ботами, забезпечуючи зручність та ефективність у роботі.

Для пошуку необхідної інформації по створеному боту, а саме який бот і який метод боту необхідно виконати, була сформована спеціальна функція для пошуку бота в базі даних і запуску методу після перевірки існування бота (див. додаток Б). Ця функція дозволяє забезпечити, що тільки валідні і автентифіковані боти можуть виконувати свої методи, що додає рівень безпеки і надійності до системи. Візуальний процес комунікації між системою та чат-ботами можна побачити на рисунку 3.3.

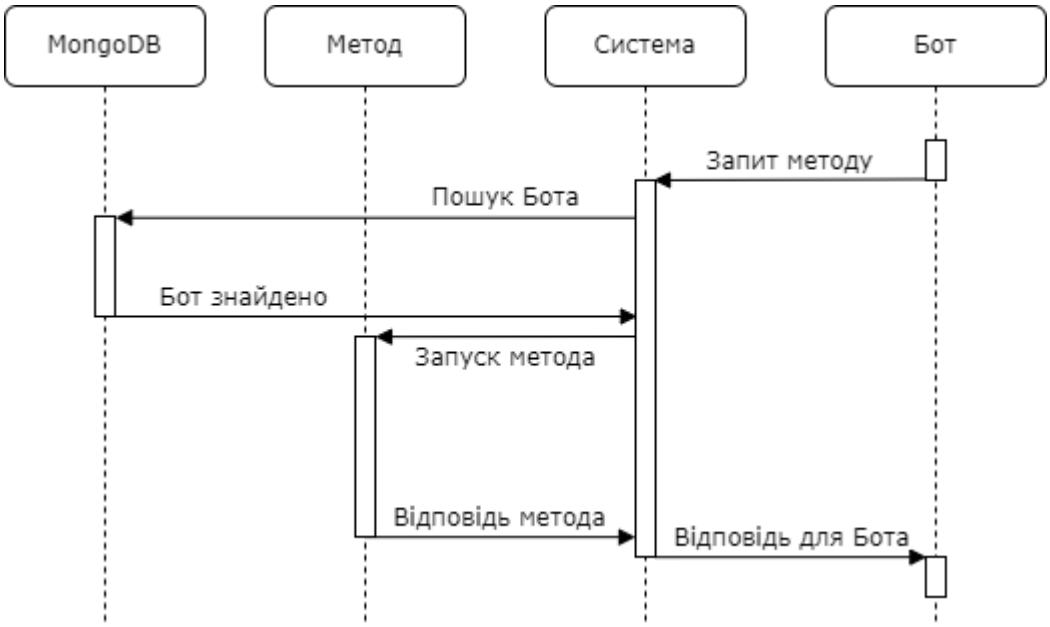


Рисунок 3.3 – Процес опрацювання запиту чат-бота системою

Для забезпечення безпеки користувачів, усі паролі та дані, які потребують шифрування, без можливості відновлення даних шифруються за допомогою Bcrypt [14]. Дана технологія забезпечує високий рівень безпеки за рахунок використання адаптивного алгоритму хешування, який включає в себе кілька важливих аспектів, що роблять його надійним інструментом для захисту паролів.

Основні переваги Bcrypt:

− автоматична генерація солі, яка дає можливість Bcrypt автоматично генерувати унікальну сіль для кожного пароля перед його хешуванням. Сіль (salt) – це випадковий набір символів, який додається до пароля перед процесом хешування, щоб зробити кожен хеш унікальним. Це забезпечує захист від атак за словником та атак з попередньо обчисленими хешами (rainbow tables);

− адаптивний алгоритм у Bcrypt дає можливість використовувати алгоритм шифрування, який можна налаштувати на різні рівні складності. Це дозволяє збільшувати час, необхідний для хешування пароля, що ускладнює проведення брутфорс-атак. В разі підвищення обчислювальних потужностей атакуючих можна підвищити складність хешування, що забезпечить довготривалу ефективність захисту;

− висока відмінність хешів, на відміну від методів шифрування, таких як MD5 і SHA256, які також генерують хеші на основі вхідних даних, Bcrypt забезпечує максимальну відмінність хешів навіть при введенні однакових значень. Це означає, що навіть якщо два користувачі мають однаковий пароль, їхні хеші будуть абсолютно різними завдяки унікальній солі, яка додається до кожного пароля перед його хешуванням.

Технічні аспекти використання Bcrypt:

− процес хешування. Коли користувач створює новий пароль, система автоматично генерує випадкову сіль і додає її до пароля. Потім комбінований пароль і сіль хешуються за допомогою алгоритму Bcrypt, що дає кінцевий хеш, який зберігається в базі даних. Це гарантує, що кожен хеш унікальний і не може бути використаний для відновлення оригінального пароля;

− перевірка паролів. Коли користувач вводить свій пароль для входу в систему, система бере введений пароль і ту саму сіль, яка використовувалася при створенні хеша, і повторно хешує їх за допомогою Bcrypt. Якщо отриманий хеш збігається з тим, що зберігається в базі даних, користувач успішно авторизується. Цей процес гарантує, що навіть якщо база даних буде

скомпрометована, зловмисники не зможуть отримати доступ до оригінальних паролів користувачів.

Порівняння з іншими методами:

1) MD5: MD5 є застарілим алгоритмом хешування, який був широко використовуваним раніше, але зараз вважається ненадійним через свою вразливість до колізій та можливість швидкого обчислення. Це робить MD5 поганим вибором для захисту паролів, оскільки зловмисники можуть легко відновити оригінальні дані за допомогою підготовлених таблиць (rainbow tables);

2) SHA256: SHA256 є більш надійним, ніж MD5, але також має свої недоліки. Хоча він забезпечує більшу стійкість до колізій, все ж таки не має адаптивності, що ускладнює захист від брутфорс-атак. Крім того, відсутність солі у базовій реалізації залишає хеші вразливими до атак за словником та використання попередньо обчислених хешів.

Приклад використання bcrypt для шифрування паролів наведено у лістингу 3.1.

Лістинг 3.1 – Шифрування за допомогою Bcrypt

```
let res = await User.findOne({ email: email });
if (res == null)
  return {
    result: false,
    reason: "user.noexist",
  };
//* 1. check if pass is right
if (!(await bcrypt.compare(password as string, res.password))) {
  if (
    res.hashHistory.some((e: { hash: string; expired: Date }) => {
      return bcrypt.compareSync(password as string, e.hash);
    })
```

```
    )
      return {
        result: false,
        reason: "password.isold",
      };
    return {
      result: false,
      reason: "password.wrong",
    };
  }
```

Кінець лістингу 3.1

Щоб реалізувати пов’язану структуру даних, необхідно створити UML-діаграму бази даних. Впровадження структури бази даних дозволяє краще зрозуміти структуру даних програми та її взаємозв’язки, що є важливим кроком у процесі розробки складних програмних систем.

Створення UML-діаграми бази даних дозволяє візуалізувати різні компоненти бази даних і їхні зв’язки, що сприяє більш ефективному проектуванню і розробці. UML-діаграма відображає таблиці бази даних, їхні атрибути, а також відношення між ними, що допомагає краще зрозуміти, як дані будуть зберігатися і взаємодіяти між собою.

Для зберігання даних необхідно реалізувати таблиці наступних об’єктів (рис. 3.4):

- користувачі – для збереження даних та надання доступу до системи;
- боти – для зберігання інформації відносно чат-бота;
- інструкції – необхідні для проведення тестування;
- тести – для зберігання фактичних даних про пройдений тест.

Маючи сформовані думки щодо форми бази даних, було сформовано зв’язок між кодом і БД за допомогою Mongoose ORM. Фрагмент коду формування документу таблиці користувачів подано в лістингу 3.2.

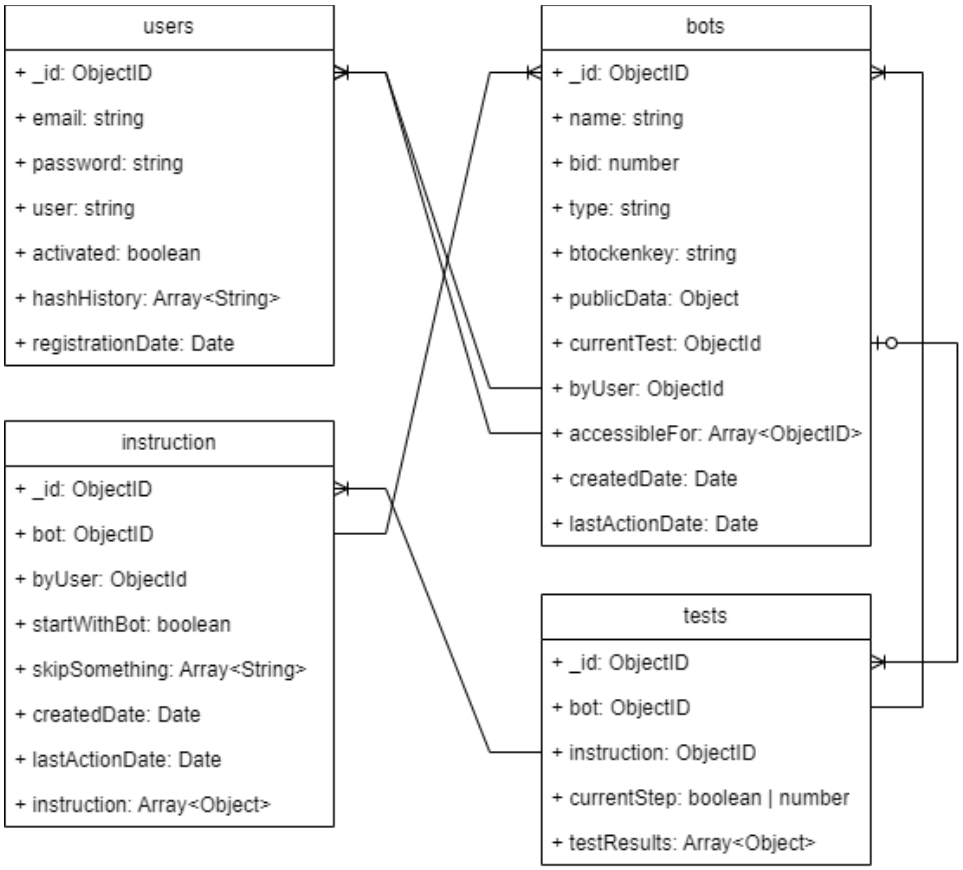


Рисунок 3.4 – Діаграма схема зв’язку таблиць

Лістинг 3.2 – Формування об’єкту користувача за допомогою Mongoose ORM

```
//User Schema
export const userSchema = new Schema<IUser>(
 {
   email: { type: String, required: true },
   password: String,
   user: String,
   activated: { type: Boolean, default: false },
   hashHistory: [{ hash: String, expired: Date }],
   registrationDate: { type: Date, default: Date.now },
 },
 { versionKey: false }
);
```

```
export const User = mongoose.models.users || model<IUser>("users", userSchema);
```

Кінець лістингу 3.2

3.2 Тестування та налагодження інформаційно-комп'ютерної системи

Для перевірки функціональності системи на помилки потрібно сформувати приклад боту і його функції відповідно до сформованих вимог системи. Врахувавши всі необхідності, було прийнято рішення про створення прикладу бота, який можна використовувати для проведення тестувань. Цей бот служитиме еталоном для перевірки коректності роботи системи тестування, забезпечуючи реалістичні умови для виявлення можливих помилок.

Написання боту виконувалося за допомогою бібліотеки Telegraf.js, яка є офіційною бібліотекою для інтеграції з Telegram API. Ця бібліотека забезпечує зручний інтерфейс для розробки ботів, спрощуючи процес взаємодії з Telegram серверами. Однією з ключових змін було використання локального сервера замість стандартного URL посилання для підключення бота. Це дозволяє більш точно і ефективно тестувати систему в умовах, наближених до реальних. Звичайний скелет простого боту телеграм з використанням бібліотеки Telegraf.js можна ознайомитися на лістингу 3.3.

Лістинг 3.3 – Приклад простого боту Telegram

```
let bot = new Telegraf(`${process.env.BOT_TEST_TOKEN}`, {
  telegram: {
    apiRoot: `${process.env.TAPI_ROOT}`, // Міняти в .env
  },
});
bot.on("message", (ctx) => {
  console.log(ctx.message);
});
```

```
bot.launch();
```

Кінець лістингу 3.3

Використовуючи даний скелет можливо формувати автоматизовані тести які зможуть перевіряти функціональність системи під час тестування. Такі тести надають можливість отримувати візуальну картину про стан додатку на етапі розробки і запобігти виникнення помилок ще на етапі розробки. Усунення таких помилок надає можливість гарантувати надійність і довершеність програмного забезпечення.

Після проведеного тестування стало очевидним, що для повного розуміння можливостей системи необхідно перевірити її здатність витримувати великі навантаження. Для цього були проведені декілька тестів із запуском кількох ботів одночасно. Це дозволило оцінити, як система справляється з одночасним виконанням великої кількості завдань і наскільки ефективно використовуються ресурси комп’ютера.

Тестування проводилося на комп’ютері з наступними характеристиками:

- процесор: Intel Core i7 10750H з тактовою частотою 2,6 ГГц, який забезпечує високу обчислювальну потужність для виконання багатозадачних процесів;
- оперативна пам’ять: 32 ГБ ОЗУ, що дозволяє обробляти великий обсяг даних і виконувати численні процеси одночасно без значних затримок;
- накопичувач: 2 ТБ SSD, що забезпечує швидке зчитування і запис даних, що є важливим для швидкого доступу до сценаріїв тестування та збереження результатів;
- графічна карта: NVIDIA GeForce RTX 2070 Max-Q, яка, хоча і не є критично важливою для тестування ботів, сприяє загальній продуктивності системи;
- операційна система: Linux (дистрибутив EndeavourOS), що забезпечує стабільне і ефективне середовище для виконання тестів.

Під час тестування було запущено автоматичний сценарій, що передбачав одночасний запуск кількох ботів. Для кращої точності тестування використовувалася одна інструкція для всіх ботів, з прикладом якої можливо ознайомитися у додатку В. Результати виконання тесту були записані і проаналізовані, а також було зібрано дані про використані ресурси, зокрема оперативну пам’ять, процесорний час і обсяг використаного дискового простору.

Проведений тест прийнято вважати підведеним до реальності без врахування витраченого часу на приєднання бота до системи, оскільки тести проводилися локально та за одним комп’ютером. Врахувавши ці вимоги можливо робити подальший аналіз необхідностей, а саме необхідні ресурси при навантаженні.

На рисунку 3.5 показано результати проведених тестів. З графіку видно, що зі збільшенням кількості тестових ботів закономірно збільшується і витрата ресурсів оперативної пам’яті. Це свідчить про те, що система ефективно масштабується при збільшенні навантаження, однак також показує, що для підтримки великої кількості одночасно працюючих ботів потрібні значні ресурси.

Детальний аналіз результатів тестування показав, що система здатна стабільно працювати при великому навантаженні, проте важливо правильно планувати ресурси для уникнення потенційних проблем з продуктивністю. Зокрема, було відзначено, що при одночасному запуску понад 1000 ботів система починає використовувати значно більше оперативної пам’яті, що може призвести до її нестачі при недостатньому обсязі ОЗУ.

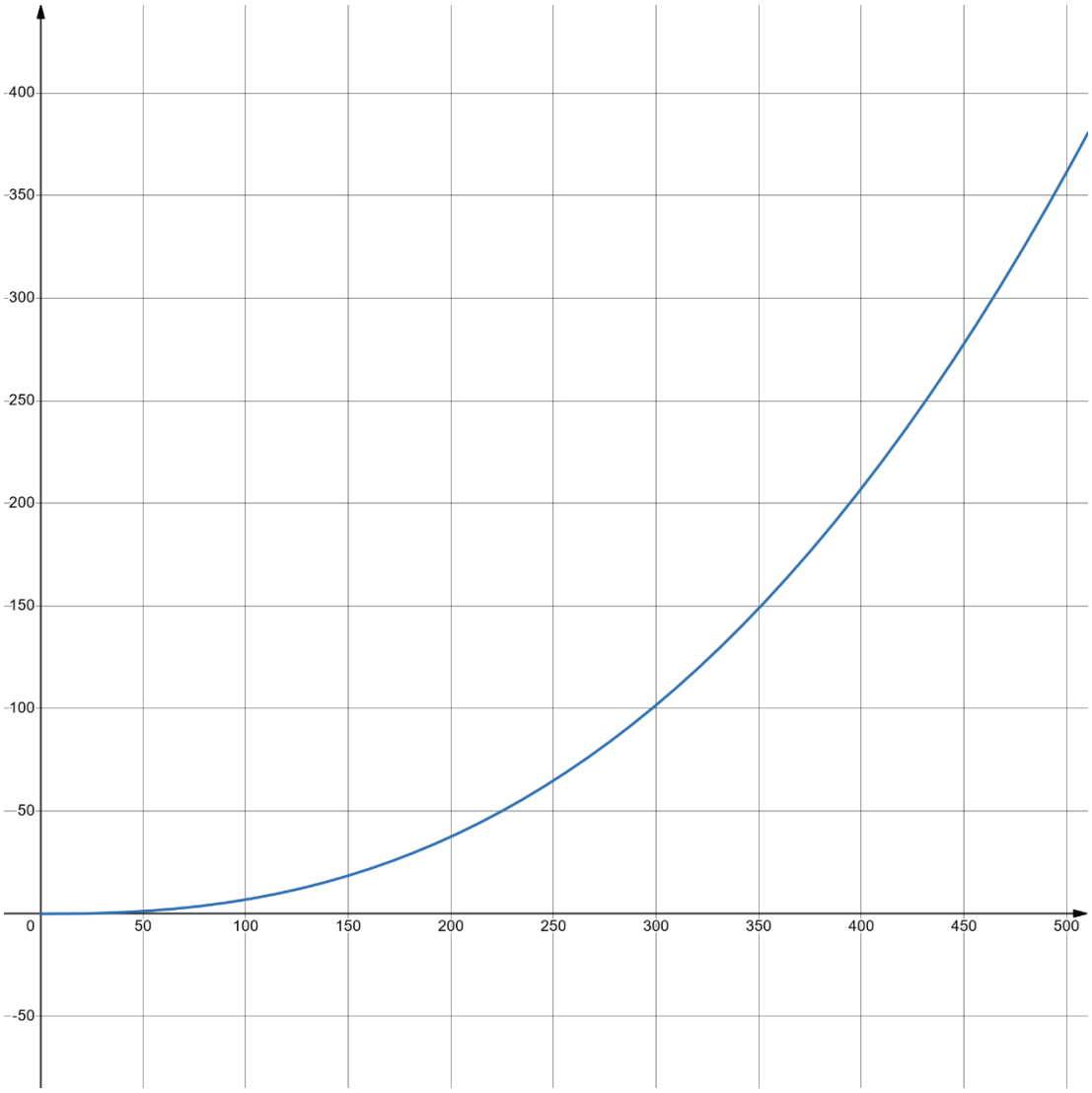


Рисунок 3.5 – Графік тенденції навантаження

Також було виявлено, що процесор працює на високій потужності, але завдяки високопродуктивному Intel Core i7 10750H, система змогла підтримувати стабільну роботу без значних затримок чи збоїв. Це свідчить про те, що для ефективної роботи системи при великому навантаженні важливо використовувати потужний процесор і достатній обсяг оперативної пам’яті.

Підсумовуючи, проведене тестування підтвердило, що розроблена система здатна ефективно працювати при великих навантаженнях, однак для забезпечення її стабільності необхідно враховувати потребу в достатніх апаратних ресурсах. Це включає потужний процесор, значний обсяг оперативної пам’яті і швидкий накопичувач для забезпечення швидкого

доступу до даних. Важливість цих факторів слід враховувати при плануванні розгортання системи на різних платформах і середовищах.

Враховуючи пройдені тести було визначено мінімальні умови необхідні для Backend системи для функціонування з мінімальним та середнім навантаженням.

Мінімальні вимоги при навантажені в 1000 одночасних підключень можливо вважати наступними:

- процесор на 4 ядра з мінімальною тактовою частотою на 2,6-3,2 Ггц;
- 8 Гб оперативної пам’яті з частотою мінімум 2600 Мгц;
- 40 Гб внутрішньої пам’яті для збереження даних в БД;
- ОС Linux.

Мінімальними вимогами при середньому навантажені в 100 000 одиниць ботів можна вважати наступними:

- процесор на 8 ядер з мінімальною тактовою частотою на 2,8-3,6 Ггц;
- 16 Гб Оперативної пам’яті з частотою мінімум 3200 Мгц;
- 256 Гб внутрішньої пам’яті для збереження даних БД;
- ОС Linux.

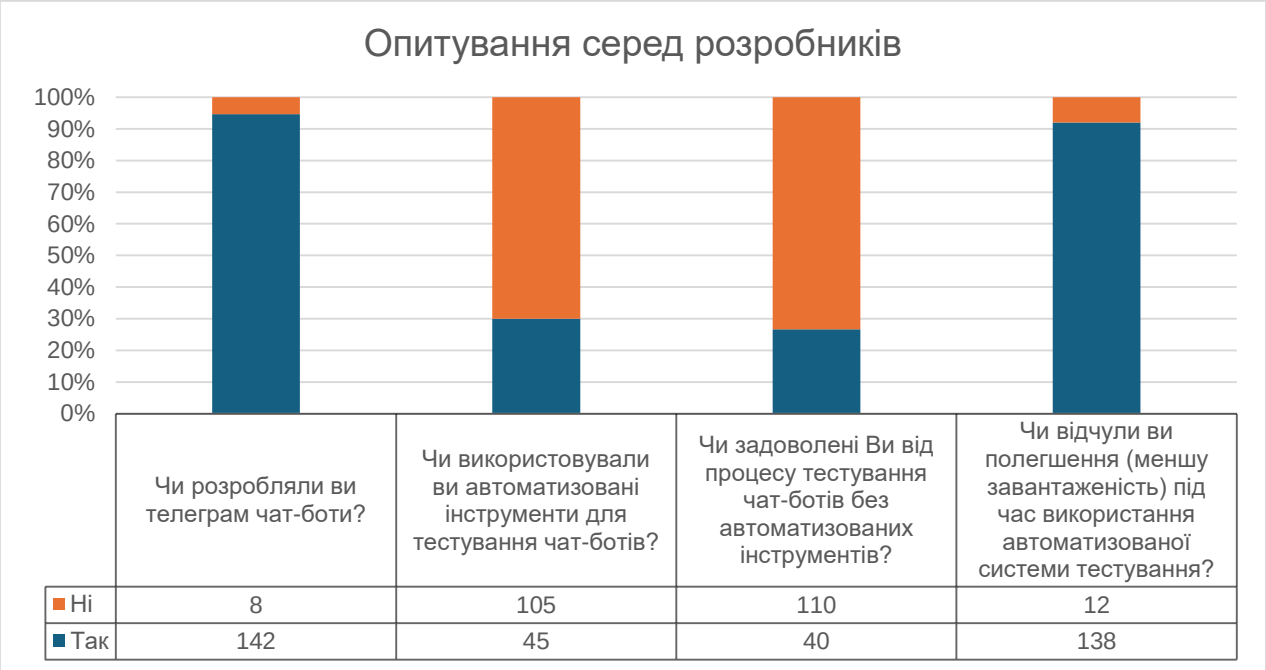
3.3 Опитування необхідності розробки системи автоматизованого тестування чат-ботів

Для отримання даних ефективності та необхідності автоматизованої системи тестування, було прийнято рішення провести опитування серед розробників, студентів, а також фірм, які мали досвід з розробкою чат-ботів. Це дозволило зібрати корисну інформацію від безпосередніх користувачів та оцінити поточний стан процесу тестування чат-ботів.

В загальному було опитано 150 розробників та 20 фірм. Кожному рецензенту було задано 4 запитання стосовно розробки та тестування чат-ботів: чи розробляли ви телеграм чат-боти, чи використовували ви автоматизовані інструменти для тестування чат-ботів, чи задоволені ви від процесу тестування

чат-ботів без автоматизованих інструментів, чи відчули ви полегшення (меншу завантаженість) під час використання автоматизованої системи тестування.

Серед опитаних 150 розробників (рис. 3.6) було виявлено, що 94,7 % з них розробляли телеграм чат-боти, але лише 30 % використовували автоматизовані інструменти для тестування. Більше половини респондентів висловили незадоволення процесом ручного тестування. Проте, 92 % розробників відчули полегшення при використанні автоматизованої системи тестування.



	Чи розробляли ви телеграм чат-боти?	Чи використовували ви автоматизовані інструменти для тестування чат-ботів?	Чи задоволені Ви від процесу тестування чат-ботів без автоматизованих інструментів?	Чи відчули ви полегшення (меншу завантаженість) під час використання автоматизованої системи тестування?
Ні	8	105	110	12
Так	142	45	40	138

Рисунок 3.6 – Діаграма опитування серед розробників

Серед 20 опитаних фірм (рис. 3.7), що займаються розробкою чат-ботів, 95 % розробляють телеграм чат-боти, але лише 20 % з них використовують автоматизовані інструменти для тестування. Незадоволення процесом ручного тестування було висловлено 80 % фірм. Водночас, 90 % фірм відчули значне полегшення під час використання автоматизованої системи тестування.

Рисунок 3.7 – Діаграма опитування серед фірм

Дані опитування показали, що серед розробників та фірм існує значна потреба у впровадженні автоматизованих систем тестування чат-ботів. Більшість респондентів виявили незадоволення від процесу ручного тестування та відчули значне полегшення при використанні автоматизованих інструментів. Ці дані підкреслюють важливість розробки та впровадження систем автоматизованого тестування для забезпечення високої якості та надійності телеграм ботів.

3.4 Економічне обґрунтування розробки

Враховуючи можливості автоматизованого тестування в порівнянні з ручним, можливо визначити наступні ключові аспекти використання системи автоматизованого тестування:

- автоматизовані тести дозволяють значно скоротити час, необхідний для проведення тестування. Ручне тестування часто є тривалим і монотонним процесом, особливо коли йдеться про повторювані тести. Автоматизація цих процесів звільняє розробників від рутинних завдань, даючи їм можливість

зосередитися на більш критичних аспектах розробки, таких як написання нового функціоналу або вдосконалення існуючого коду. Це дозволяє значно підвищити продуктивність команди та скоротити час виходу продукту на ринок;

− автоматизовані тести дозволяють значно покращити покриття тестування. Завдяки точності та послідовності автоматизованих тестів можна перевіряти більше деталей і сценаріїв, ніж при ручному тестуванні. Це забезпечує більш глибоке і всебічне тестування функціоналу, що допомагає виявити більше потенційних помилок і недоліків у коді. Крім того, автоматизоване тестування дозволяє легко запускати комплексні тести на різних платформах і середовищах, що сприяє більшій сумісності та стабільності продукту;

− автоматизоване тестування дозволяє виявляти помилки на ранніх стадіях розробки. Часті запуски автоматизованих тестів під час кожного циклу розробки допомагають знайти і виправити помилки ще до того, як вони стануть критичними. Це значно знижує витрати на виправлення помилок у пізніших етапах розробки або після випуску продукту. Завчасне виявлення проблем також сприяє підвищенню якості кінцевого продукту і задоволенню користувачів;

− процес розробки автоматизованих тестів часто веде до створення більш чистого і керованого коду. Розробники змушені детально продумувати і структурувати свій код, щоб його було легко тестувати. Це сприяє застосуванню кращих практик програмування, таких як модульність, повторне використання коду і написання зрозумілого та читабельного коду. В результаті код стає більш стійким до змін і легше підтримується в довгостроковій перспективі.

Автоматизація тестування Telegram чат-ботів дозволяє зекономити час на проведення тестування і значно спростити цей процес. Інтеграція автоматизованих тестів до чат-ботів забезпечує більш точне і регулярне тестування, що сприяє підвищенню якості кінцевого продукту. Спеціальна

інструкція визначає результат виконання роботи чат-бота, що дозволяє легко оцінити його ефективність і виявити можливі недоліки.

Для отримання порівняльних даних було проведено опитування серед студентів другого та третього курсу. У студентів запитували, чи вони розробляли телеграм чат-ботів, скільки часу в середньому вони витрачали на це і скільки операцій було виконано з ботом для тестування. Учасникам опитування було надано пробний доступ до системи, близько 95 % з яких вказали на значне полегшення, скорочення часу та звільнення від процесу тестування.

На графіку нижче (рис. 3.8) представлена приблизна економія часу в порівнянні між ручним і автоматизованим тестуванням.

Графік ілюструє значне скорочення часу, необхідного для виконання тестів, при використанні автоматизованого підходу. Це демонструє ефективність автоматизованого тестування і його переваги перед ручним тестуванням, особливо при тестуванні складних і багатофункціональних систем, таких як Telegram чат-боти.

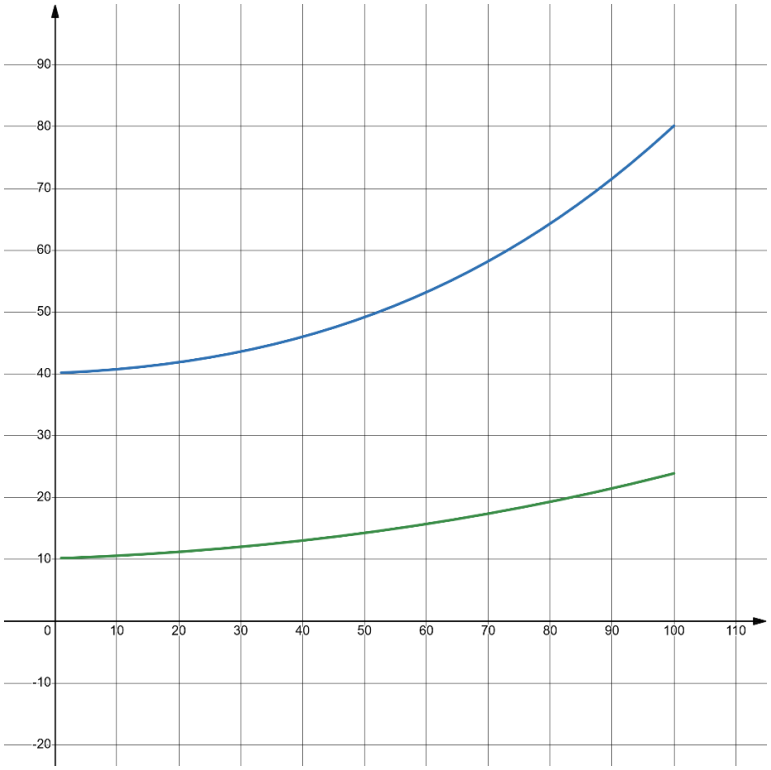


Рисунок 3.8 – Графік порівняння затраченого часу

3.5 Перспективи подальших досліджень та вдосконалень

Розробка системи тестування чат-ботів відкриває широкий спектр можливостей для подальших досліджень та вдосконалень, спрямованих на покращення якості та ефективності роботи цих систем. Однією з головних напрямків є інтеграція методів машинного навчання та штучного інтелекту для автоматизованого аналізу результатів тестування. Використання таких методів дозволить не лише автоматично виявляти помилки, але й передбачати потенційні проблеми у функціонуванні чат-ботів. Це, в свою чергу, надасть розробникам можливість отримувати рекомендації щодо оптимізації роботи чат-ботів та покращення їх взаємодії з користувачами.

Іншим перспективним напрямком є вдосконалення користувацького інтерфейсу та функціональних можливостей системи тестування. Зокрема, розробка інструментів для візуалізації процесу тестування та результатів дозволить розробникам швидше ідентифікувати та усувати проблеми. Такі інструменти можуть включати графічні відображення послідовностей діалогів, аналіз часу відповіді та інші метрики, що надають розгорнуту картину про ефективність роботи чат-бота. Крім того, інтеграція з популярними системами управління проектами та платформами для розробки програмного забезпечення, такими як JIRA, GitHub, та інші, забезпечить більш зручний та безперервний процес тестування.

Подальші дослідження також можуть зосередитися на розширенні підтримки різних платформ та мов програмування, які використовуються для розробки чат-ботів. Це дозволить зробити систему більш універсальною та корисною для широкого кола розробників. Наприклад, підтримка популярних фреймворків для створення чат-ботів, таких як Microsoft Bot Framework, Google Dialogflow, та інші, значно розширить можливості тестування. Крім того, варто дослідити можливості інтеграції з хмарними сервісами для забезпечення масштабованості та підвищення продуктивності системи тестування, що стане особливо актуальним для великих проектів з високим навантаженням.

Ще одним важливим напрямком є інтеграція інших платформ, які застосовують чат-боти, для створення комплексної системи тестування. Це включає в себе різні канали комунікації, такі як соціальні мережі, месенджери, веб-сайти та мобільні додатки. Розробка таких інтеграцій дозволить тестувати чат-ботів у реальних умовах, забезпечуючи повний цикл перевірки їх роботи на всіх можливих платформах. Це потребує створення складніших алгоритмів тестування, які зможуть оцінювати якість та узгодженість відповіді чат-бота у різних форматах і середовищах.

Загалом, подальший розвиток системи тестування чат-ботів може значно покращити якість та ефективність процесу розробки, сприяючи створенню більш надійних, функціональних та безпечних чат-ботів, здатних задовольнити зростаючі потреби користувачів та бізнесу.

Висновки до розділу 3

Було розроблено програмне забезпечення, яке дозволяє проводити автоматизоване тестування телеграм ботів. Ця система надає розробникам ефективний інструмент для перевірки функціональності їхніх ботів, автоматизуючи процес тестування за допомогою сценаріїв у форматі JSON. Завдяки цьому підходу, розробники можуть зосередитися на вдосконаленні функціональності та покращенні користувацького досвіду, замість витрачати час на ручне тестування.

Під час розробки системи було проведено численні тести на навантаження, щоб визначити її здатність обробляти велику кількість запитів одночасно. Тестування показало, що система здатна ефективно обробляти понад 1000 запитів від різних ботів, що демонструє її високу продуктивність і масштабованість. Це дозволяє забезпечити стабільну роботу сервісу навіть при значному збільшенні кількості користувачів і обсягу даних.

Також було визначено необхідні ресурси для запуску сервісу, включаючи вимоги до серверів, баз даних і інших компонентів інфраструктури. Система

розгорнута на Linux серверах, що забезпечує її надійність та можливість у розгортанні та масштабуванні. Використання TypeScript та Remix.Run дозволило створити зручний API інтерфейс для комунікації з ботами.

Було проведено опитування серед розробників, студентів і фірм, які мають досвід у розробці чат-ботів, щоб оцінити ефективність та необхідність автоматизованої системи тестування. Загалом опитали 150 розробників і 20 фірм, запитуючи про їхній досвід з телеграм чат-ботами, використання автоматизованих інструментів для тестування, задоволеність процесом ручного тестування та відчуття полегшення від автоматизації. Виявилося, що 94.7 % розробників і 95 % фірм розробляють телеграм чат-боти, але автоматизовані інструменти використовують лише 30 % розробників і 20 % фірм. Більшість респондентів незадоволені ручним тестуванням, причому 92 % розробників і 90 % фірм відчули полегшення завдяки автоматизації. Дані опитування підкреслюють значну потребу у впровадженні автоматизованих систем тестування для забезпечення високої якості та надійності телеграм ботів.

Було проведено порівняння між ручним і автоматизованим тестуванням, яке показало значну перевагу автоматизації в контексті зменшення часу на тестування, покращення покриття тестів і завчасного виявлення помилок. Автоматизоване тестування дозволяє знайти більше помилок на ранніх етапах розробки, що знижує витрати на їх виправлення в подальшому. Це призводить до покращення якості кінцевого продукту і задоволеності користувачів.

Економічне обґрунтування розробки автоматизованої системи тестування показало, що інвестиції в автоматизацію швидко окупаються за рахунок зменшення витрат на ручне тестування та підвищення ефективності розробки. Автоматизація дозволяє розробникам виконувати більше тестів за коротший час, що сприяє швидшому випуску продуктів на ринок і підвищенню їхньої якості. Це також забезпечує конкурентну перевагу компанії на ринку, оскільки дозволяє швидше реагувати на зміни та випускати оновлення продукту з мінімальними ризиками.

Проведені дослідження і аналіз продемонстрували, що впровадження автоматизованої системи тестування не лише економить час і зусилля розробників, але й забезпечує високу надійність і стабільність роботи телеграм ботів. Система дозволяє виявляти помилки і проблеми в роботі ботів ще на етапі розробки, що значно зменшує ризики виникнення критичних помилок у продуктивному середовищі.

Крім того, автоматизація тестування сприяє створенню більш чистого і підтримуваного коду. Процес розробки автоматизованих тестів вимагає від розробників структурувати свій код і дотримуватися кращих практик програмування. Це не лише покращує якість коду, але й робить його більш зрозумілим і легким для подальшої підтримки і розширення.

Важливо також відзначити, що система забезпечує безпеку зберігання сценаріїв тестування і даних користувачів. Використання сучасних методів шифрування, таких як Bcrypt, гарантує захист чутливої інформації і знижує ризики несанкціонованого доступу до даних. Це особливо важливо в контексті зростаючих вимог до безпеки даних і конфіденційності користувачів.

Загалом, розробка та впровадження автоматизованої системи тестування телеграм ботів довела свою ефективність і доцільність. Вона забезпечує значні переваги у порівнянні з ручним тестуванням, сприяє покращенню якості продукту і задоволеності користувачів, а також знижує витрати і підвищує ефективність розробки. Це робить автоматизоване тестування невід’ємною частиною сучасного процесу розробки програмного забезпечення.

ВИСНОВКИ

У ході виконання дослідження було здійснено всебічний аналіз предметної області, що дало змогу глибше зрозуміти сучасний стан проблеми та визначити основні потреби користувачів. Аналіз аналогічних програм та систем, таких як OWASP ZAP або Burp Suite, допоміг обґрунтувати вибір напрямку розробки та підходів до вирішення поставлених задач.

Постановка завдання передбачала розробку Backend-у для сервісу тестування телеграм ботів. Для вирішення цього завдання були визначені вимоги до розроблюваної системи. Специфікація вимог включала в себе необхідні функціональні та технічні характеристики сервісу, що дозволяють автоматизувати процес тестування ботів за допомогою спеціальних сценаріїв у форматі JSON.

Досліджуючи існуючі методи та інструменти для тестування телеграм чат-ботів було визначено, що ручне тестування являється основним дієздатним методом тестування. Існуючий метод автоматизованого тестування чат-ботів, потребує часу для правильності налаштування, але даний метод не дає можливості контролювати реальний сценарій виконання роботи, а також працює лише з текстовими чат-ботами.

Розробляючи концепцію системи було прийнято рішення, щодо створення системи яка максимально подібна до концепції роботи Telegram з ботами, спростивши процес налаштування ботів та їхнього підключення до нової системи.

Для реалізації системи було обрано наступні фреймворки та мови програмування:

- TypeScript – для простої типізації даних;
- Remix.Run – для керування архітектурою системи;
- MongoDB – для збереження даних базованих на документах і з динамічно визначеними даними;

- Telegraf.js (Telegram Lib) – для тестування і інтеграції системи з телеграм ботами.

Система була реалізована з використанням наведених технології і забезпечує стабільну роботу в комунікації між чат-ботами та самою системою.

Було проведено тестування системи з використанням реальних телеграм ботів та оцінено ефективність за такими критеріями, як точність тестування, продуктивність, стабільність та безпека. Виявлені недоліки були виправлені, а результати роботи відображені у відповідній документації та звіті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Здолбіцька Н. В., Савич І. М. Кікець М. І. Розробка сервісу автоматизованого тестування чат-ботів. IX Міжнародна науково-практична конференція «Інформаційні технології в освіті, науці і виробництві» тези доп., 25-26 травня 2023 р. Луцьк. 2023. С. 168-170.

2. Burp Suite vs. OWASP ZAP – Which is Better for API Security Testing? | APIsec. URL: https://www.apisec.ai/blog/burp-suite-vs-zap (дата звернення: 12.05.2024).

3. Java – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Java (дата звернення: 19.05.2024).

4. Python – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Python (дата звернення: 19.05.2024).

5. JavaScript – Вікіпедія. URL: https://uk.wikipedia.org/wiki/JavaScript (дата звернення: 19.05.2024).

6. Client-Side Rendering vs Server-Side Rendering vs Pre-rendering: Key Differences – Clockwise Software. URL: https://clockwise.software/blog/client-side-vs-server-side-vs-pre-rendering/ (дата звернення: 20.05.2024).

7. Telegram APIs. URL: https://core.telegram.org/api (дата звернення: 21.05.2024).

8. TypeScript: JavaScript With Syntax For Types. URL: https://www.typescriptlang.org/ (дата звернення: 22.05.2024).

9. Remix – Build Better Websites. URL: https://remix.run/ (дата звернення: 22.05.2024).

10. What is MongoDB? – MongoDB Manual v7.0. URL: https://www.mongodb.com/docs/manual/ (дата звернення: 22.05.2024).

11. telegraf.js – v4.16.3. URL: https://telegraf.js.org/ (дата звернення: 21.05.2024).

12. Bots: An introduction for developers. URL: https://core.telegram.org/bots (дата звернення: 23.05.2024).

13. Route File Naming | Remix. URL: https://remix.run/docs/en/main/file-conventions/routes (дата звернення: 24.05.2024).

14. Bcrypt – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Bcrypt (дата звернення: 19.05.2024).

ДОДАТКИ

Додаток А

Апробація результатів кваліфікаційної роботи бакалавра

УДК 004.65

РОЗРОБКА СЕРВІСУ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ЧАТ-БОТІВ

Здолбіцька Ніна Василівна
Луцький національний технічний університет, к.т.н.,
доцент кафедри комп'ютерних наук, ninazdolb@lntu.edu.ua
Савич Іван Миколайович
Луцький національний технічний університет, здобувач,
savych.vana@gmail.com
Кікець Микола Ігорович
Луцький національний технічний університет, здобувач,
mykola.kikets@gmail.com

Чат-боти стали невід'ємною частиною онлайн-бізнесів, оскільки вони дозволяють розв'язувати завдання, які раніше були неможливими: допомога клієнтам 24/7, вирішення запитань без участі людини та підтримка кількох клієнтів одночасно. Інші переваги, які надають бізнесам чат-боти, включають швидке розв'язання запитань, зменшення витрат на операційну діяльність, зменшення помилок людського фактору, багатокористувацьку підтримку та автоматизацію певних задач.

Метою роботи по розробці сервісу для автоматизованого тестування чат-ботів є створення інструменту, який допоможе розробникам і тестувальникам ефективно перевіряти функціональність та якість чат-ботів.

Основні цілі роботи включають:

1. Автоматизація тестування: Сервіс повинен надавати зручний інтерфейс для створення тестових сценаріїв, в яких можна визначити послідовність запитів та очікуваних відповідей для чат-бота. Це дозволить автоматично виконувати тестові сценарії і перевіряти правильність роботи бота.

2. Параметризація тестів: Сервіс повинен підтримувати налаштування параметрів для тестування, наприклад, вказувати різні вхідні дані для кожного тестового сценарію або задавати різні умови тестування. Це дозволить проводити різноманітні тести для виявлення проблем та забезпечення належної роботи чат-бота у різних умовах.

3. Запис та аналіз результатів: Сервіс повинен забезпечувати збереження результатів тестування для подальшого аналізу. Це включає збереження журналів запитів та відповідей, виявлення помилок або несправностей у роботі бота, а також генерацію звітів про тестування.

Існують різні підходи до тестування чат-ботів, включаючи перевірку їх конверсаційного потоку, проведення тестування з орієнтацією на конкретну галузь [1-2], тестування розробників, тестування на основі взаємодії з користувачами [3], перевірку обробки помилок чат-ботом та автоматизацію процесу тестування [4].

Тестування чат-ботів є безперервним процесом, оскільки розробники постійно вносять зміни до функціоналу та оновлюють його.

За результатами аналізу українських та англомовних публікацій, можна сказати, що проблематика тестування чат-ботів є досить актуальною та складною. Основні проблеми, які виникають в процесі тестування чат-ботів, це:

1. Різноманітність запитань та відповідей: користувачі можуть поставити запитання на різні теми, а чат-бот повинен правильно відповісти на всі запитання, що може бути складно.
2. Нестабільність роботи: чат-бот може неправильно розпізнавати запитання, давати неправильні відповіді, або не працювати взагалі через технічні проблеми.
3. Відсутність стандартів та методів тестування: існує багато методів тестування чат-ботів, але немає однієї універсальної методики, яка підійшла б для всіх видів чат-ботів.
4. Недостатня якість навчальних даних: навчальні дані є ключовим фактором для підвищення якості чат-ботів. Однак, недостатня якість навчальних даних може призвести до низької якості відповідей чат-бота.

Отже, в сучасному стані проблематика тестування чат-ботів є досить складною та потребує комплексного підходу до вирішення. Необхідно розробляти стандарти та методики тестування, вдосконалювати навчальні дані та використовувати

різноманітні методи тестування, щоб забезпечити високу якість та ефективність роботи чат-ботів.

Тестування ботів. Це включає в себе розуміння того, які техніки тестування можуть використовуватися для перевірки роботи ботів, включаючи автоматизоване тестування, ручне тестування та тестування з використанням реальних користувачів.

Аналіз результатів тестування. Це включає в себе збір та аналіз даних про роботу ботів, оцінку ефективності тестування та виявлення проблем, що можуть виникнути при використанні ботів.

Розробка рекомендацій для покращення роботи ботів. Це включає в себе формулювання пропозицій та рекомендацій щодо покращення функціональності ботів, підвищення їх ефективності та якості взаємодії з користувачами.

У загальному, предметна область сервісу по тестуванню ботів є складною та має досить великий обсяг знань, які потрібні для ефективної роботи в цій сфері. Важливо мати глибокі знання про функціональні можливості ботів.

Список використаних джерел

1. The Chatbot Testing Checklist: Tools, Techniques, and Metrics to Include in Your Testing Strategy. URL: https://chatbotslife.com/the-chatbot-testing-checklist-tools-techniques-and-metrics-to-include-in-your-testing-strategy-3478a74eb215 (дата звернення: 28.04.2023).

2. Тестування чат-ботів. training qatestlab.com URL: https://training.qatestlab.com/blog/technical-articles/chat-bots-testing/ (дата звернення: 28.04.2023).

3. Josip Bozic, Franz Wotawa. Testing Chatbots Using Metamorphic Relations. 31th IFIP International Conference on Testing Software and Systems (ICTSS), Paris, France, 2019. 41-55.

4. Кравчук С. Проблеми автоматизації тестування програмного забезпечення. Актуальні задачі сучасних технологій: матеріали VII міжнар. наук.-техн. конф. мол. учен. та студ., 28–29 листопада 2018 р. Тернопіль, 2018. С. 95.

Додаток Б

Лістинг коду визначення бота та визначення методу

```
import { ActionArgs, json, LoaderArgs } from "@remix-run/node";
import db from "~/classes/Database.server";
import { Bot } from "~/classes/models";
import { Util as util, BotSplit } from "../../../classes/Util.server";

export async function loader({ request, params }: LoaderArgs) {
  let query = new URL(request.url).searchParams;
  let data: any = {};
  for (let a in query.keys()) {
    data[a] = query.getAll(a);
    if (data[a].lenght == 1) data[a] = data[a][0];
  }
  return json(
    await checkBotAndExec(params.method as string, params.token as string, data)
  );
}

export async function action({ request, params }: ActionArgs) {
  let data = await request.json();
  // console.log(await bot?.getObject());
  // let bdata = BotSplit(params.token || ":");
  return json(
    await checkBotAndExec(params.method as string, params.token as string, data)
  );
}
```

```
async function checkBotAndExec(method: string, token: string, data: any) {
  let conn = await db();
  let bdata = BotSplit(token || ":");
  if (process.env.NODE_ENV !== "development" && bdata[1] === "")
    return { ok: false, error_code: 404, description: "No token provided" };
  let bot = await Bot.findOne({ bid: bdata[0], token: bdata[1] });
  if (!bot)
    return { ok: false, error_code: 404, description: "Bot not founded" };
  let results:
    | {
        ok: false;
        error_code: number;
        description: any;
      }
    | { ok: true; result: any };
  data.__bot = bot;
  // console.log(data);
  if (Object.keys(util).some((e) => e == method)) {
    results = await util.runMethod(method || "", data);
  } else {
    results = {
      ok: false,
      error_code: 404,
      description: "Not found",
    };
  }
  return results;
}
```

Додаток В

Формат інструкції для бота

```
{
 "instructionFor": "",
 "instructionType": "user|{chat}|{intent}",
 "instructions": [
  {
   "action": "sleep", // Нічого не робити (спати)
   "timeout": 15000 // Скільки часу спати
  },
  {
   "action": "waitForAction", // очікування на дію від бота
   "timeout": 15000, // Скільки часу чекати до підтвердження неправильності запиту
   "type": "message|edit|callback|delete", // який тип дії слід очікувати
   "actionData": {
    // Дані про дію
    "__contains": "/RegExp/", // Текст повинен містити ...
    "__exact": "string" // Текст на 100 % складається з цього
    //... Тут міститься додаткова інформація щоб розуміти що конкретно очікувати.
   }
  },
  {
   "action": "sendMessage", // Надіслати дані до бота.

   "actionData": {
    //... Тут міститься додаткова інформація щоб розуміти що конкретно відправити.
```

```
      }
    },
    {
      "action": "sendIntent", // Надіслати інтент до бота.
      "actionData": {
        //... Тут містяться дані про Інтент який відправляється до бота
      }
    }
  ]
}
```