МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

ІНФОРМАЦІЙНИЙ ЧАТБОТ ЧДБК

Виконав: студент групи 2П-22
Спеціальності
121 Інженерія програмного забезпечення
Артем МЕЛЬНИК
Керівник:
Вікторія НЕМЧЕНКО

Черкаси 2026

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ

Циклова комісія комп’ютерної інженерії та інформаційних технологій

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувачка циклової комісії КІ та ІТ

_____ Наталя ФАЛЬЧЕНКО

(підпис)

«_____» _____2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Мельнику Артему Олександровичу

1. Тема кваліфікаційної роботи «Інформаційний чатбот ЧДБК»

Керівник роботи Немченко Вікторія Юріївна, викладач другої категорії

затверджені наказом закладу фахової передвищої освіти від «06» жовтня 2026 року №84/1-у

2. Строк подання студентом кваліфікаційної роботи 02.06.2026

3. Вихідні дані до кваліфікаційної роботи мова програмування Python 3.11, бібліотека python-telegram-bot 22.7, середовище розробки PyCharm Community Edition, платформа Telegram Bot API, фреймворк тестування pytest 9.0.3, операційна система Windows.

4. Зміст кваліфікаційної роботи аналіз предметної області та існуючих рішень у сфері освітніх Telegram-ботів; визначення інформаційних потреб абітурієнтів та студентів ЧДБК; обґрунтування вибору технологічного стеку та інструментів розробки; проєктування архітектури програмної системи та розробка діаграми варіантів використання і блок-схеми алгоритму роботи; реалізація програмного коду чатбота з підтримкою двох інформаційних меню – для абітурієнтів та студентів; налаштування розгортання та конфігурації системи; розробка та виконання комплексного тестування з використанням pytest та unittest.mock.

5. Дата видачі завдання 15.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

No з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	14.10.2025	
2	Розділ 1 Теоретичні основи розробки чатботів та аналіз предметної області	09.12.2025	
3	Розділ 2. Проєктування та реалізація телеграм-чатбота	10.03.2026	
4	Розділ 3. Тестування програмного продукту телеграм-чатбота	28.04.2026	
5	Висновки	12.05.2026	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	26.05.2026	
7	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	02.06.2026	
8	Подання кваліфікаційної роботи на затвердження завідувачу кафедри (за 7 днів до захисту)	10.06.2026	

Студент _____ Артем МЕЛЬНИК
(підпис)

Керівник роботи _____ Вікторія НЕМЧЕНКО
(підпис)

АНОТАЦІЯ

У кваліфікаційній роботі здійснено розроблення програмної системи для автоматизованого інформування абітурієнтів та студентів Черкаського державного фахового бізнес-коледжу у форматі Telegram-чатбота. Проведено аналіз предметної області, визначено інформаційні потреби двох цільових аудиторій та сформовано функціональні і нефункціональні вимоги до системи. Здійснено порівняльний аналіз платформ і бібліотек для розробки Telegram-ботів, за результатами якого обрано мову Python та бібліотеку python-telegram-bot версії 22.7. Спроєктовано архітектуру системи на основі асинхронної обробки подій, розроблено діаграму варіантів використання та блок-схему алгоритму роботи бота. Реалізовано повнофункціональний програмний продукт із двома інформаційними блоками: для абітурієнтів (спеціальності, умови вступу, вартість навчання, гуртожиток, соціальна підтримка, контакти) та для студентів (розклад занять і дзвінків, кабінети, укриття, зупинки транспорту, цікаві місця). Проведено комплексне тестування системи з розробкою 43 автоматизованих тест-кейсів на базі фреймворку pytest, всі тести виконано успішно. Результати роботи можуть бути безпосередньо впроваджені у ЧДБК для підвищення якості інформаційного обслуговування та скорочення навантаження на персонал закладу. Робота складається з трьох розділів, містить рисунки, таблиці та лістинги програмного коду.

Ключові слова: TELEGRAM-БОТ, ЧАТБОТ, PYTHON, PYTHON-TELEGRAM-BOT, INLINE-КЛАВІАТУРА, CALLBACK-ЗАПИТ, АВТОМАТИЗОВАНЕ ІНФОРМУВАННЯ, ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, PYTEST, АБІТУРІЄНТИ, ЗАКЛАД ОСВІТИ

ABSTRACT

This qualification thesis presents the development of a software system for automated information delivery to applicants and students of Cherkasy State Vocational Business College in the form of a Telegram chatbot. The subject domain was analyzed, the information needs of two target audiences were identified, and functional and non-functional system requirements were formulated. A comparative analysis of platforms and libraries for Telegram bot development was conducted, resulting in the selection of Python and the python-telegram-bot library version 22.7. The system architecture was designed based on asynchronous event processing; a use case diagram and a flowchart of the bot's algorithm were developed. A fully functional software product was implemented with two information modules: for applicants (specialties, admission requirements, tuition fees, dormitory, social support, contacts) and for students (class schedule, bell schedule, classrooms, civil defense shelters, public transport stops, nearby places of interest). Comprehensive system testing was performed with 43 automated test cases developed using the pytest framework; all tests passed successfully. The results of this work can be directly implemented at the college to improve the quality of information services and reduce the workload on administrative staff. The thesis consists of three chapters and includes figures, tables, and program code listings.

Keywords: TELEGRAM BOT, CHATBOT, PYTHON, PYTHON-TELEGRAM-BOT, INLINE KEYBOARD, CALLBACK QUERY, AUTOMATED INFORMATION SYSTEM, SOFTWARE TESTING, PYTEST, APPLICANTS, EDUCATIONAL INSTITUTION

ЗМІСТ

ВСТУП

Стрімкий розвиток цифрових технологій і зростаюча роль месенджерів у повсякденному житті зумовили появу принципово нових форм взаємодії між закладами освіти та їх цільовою аудиторією. Сучасні абітурієнти та студенти очікують отримувати необхідну інформацію швидко, зручно та без зайвих зусиль – безпосередньо у смартфоні, без необхідності відкривати браузер або телефонувати до адміністрації.

Актуальність теми. Черкаський державний фаховий бізнес-коледж (ЧДБК) є провідним закладом фахової передвищої освіти Черкащини, який щороку приймає нових абітурієнтів і обслуговує сотні студентів. Незважаючи на наявність офіційного вебсайту та сторінок у соціальних мережах, заклад не має спеціалізованого інтерактивного інструменту для оперативного надання довідкової інформації у месенджерах. Це призводить до перевантаження приймальної комісії, уповільнення відповідей на типові запитання і, як наслідок, – до зниження якості інформаційного супроводу вступної кампанії та навчального процесу. В умовах воєнного стану та підвищених вимог до мобільності й доступності сервісів розробка Telegram-чатбота для ЧДБК набуває особливої практичної значущості.

Об'єкт дослідження – процес автоматизованого інформування абітурієнтів та студентів Черкаського державного фахового бізнес-коледжу засобами месенджер-бота.

Предмет дослідження – методи та засоби розробки Telegram-чатбота як програмної системи для інформування, побудованої на основі детермінованої архітектури з ієрархічним меню та inline-клавіатурами.

Мета роботи – проєктування та реалізація Telegram-чатбота для ЧДБК, що забезпечує автоматизоване надання структурованої довідкової інформації двом цільовим аудиторіям: абітурієнтам (спеціальності, умови вступу, вартість навчання, гуртожиток, соціальна підтримка, контакти приймальної комісії) та

студентам (розклад занять і дзвінків, кабінети, укриття цивільного захисту, зупинки транспорту, інфраструктура поблизу коледжу).

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Проаналізувати теоретичні основи розробки чатботів та їх застосування в освітній сфері
2. Дослідити інформаційні потреби абітурієнтів та студентів ЧДБК
3. Провести огляд і порівняльний аналіз платформ та бібліотек для розробки Telegram-ботів
4. Проаналізувати існуючі аналоги та обґрунтувати доцільність розробки
5. Спроєктувати архітектуру та логіку взаємодії чатбота, розробити діаграму варіантів використання та блок-схему алгоритму роботи
6. Реалізувати програмний код бота мовою Python з використанням бібліотеки python-telegram-bot
7. Протестувати систему та оцінити відповідність функціональним і нефункціональним вимогам

Практичне значення роботи полягає у створенні реально працюючого Telegram-бота, що може бути впроваджений у ЧДБК для підвищення якості інформаційного обслуговування абітурієнтів і студентів, скорочення навантаження на приймальну комісію та персонал закладу.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ЧАТБОТІВ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття чатботу та його роль у сучасних інформаційних системах

У сучасному суспільстві стрімкий розвиток цифрових технологій зумовив появу принципово нових способів взаємодії людини з інформаційними системами. Одним із таких інструментів є чатбот – програмний засіб, що дозволяє автоматизувати комунікацію між системою та користувачем у форматі діалогу, наближеного до природної мови.

Термін «чатбот» (англ. chatbot або chatterbot) поєднує слова «chat» (розмова, чат) та «bot» (скорочення від «robot»). У найзагальнішому розумінні чатбот – це комп'ютерна програма, спроектована для імітації розмови з людиною, зокрема через Інтернет. Концептуально чатботи сягають своїм корінням до 1960-х років, коли Джозеф Вейценбаум у Массачусетському технологічному інституті розробив програму ELIZA – перший відомий прообраз сучасного чатбота, що імітував спілкування психотерапевта з пацієнтом.

Сьогодні чатботи набули значного поширення в різних галузях: банківській справі, електронній торгівлі, охороні здоров'я, освіті та державному управлінні. Вони дозволяють суттєво скоротити витрати на обслуговування клієнтів, підвищити швидкість реакції на запити та забезпечити цілодобову доступність сервісів без залучення людського персоналу.

За принципом функціонування розрізняють два основні типи чатботів:

- чатботи на основі правил (rule-based chatbots) – функціонують відповідно до заздалегідь визначених сценаріїв та ключових слів; їхня поведінка детермінована і передбачувана, однак обмежена набором закладених правил;
- чатботи на основі штучного інтелекту (AI-based chatbots) – використовують методи машинного навчання та обробки природної мови (NLP) для розуміння контексту запитів і формування адаптивних відповідей.

У контексті даної кваліфікаційної роботи розглядається чатбот першого типу – детермінований, на основі правил та структурованих меню. Такий підхід є доцільним для застосувань з обмеженим і чітко визначеним набором запитів, де важливими є передбачуваність поведінки, простота підтримки та низькі вимоги до обчислювальних ресурсів.

У сфері освіти чатботи вирішують низку практичних завдань: надають довідкову інформацію про навчальний заклад, допомагають у навігації по сайту та документах, відповідають на типові запитання абітурієнтів та студентів, нагадують про дедлайни та розклад, а також автоматизують збирання зворотного зв'язку.

1.2 Огляд платформ для розробки Telegram-ботів

Для реалізації чатботу, орієнтованого на взаємодію через месенджер, необхідно обрати відповідну платформу. Telegram є одним із найпопулярніших месенджерів у світі: станом на 2024 рік його аудиторія перевищила 900 мільйонів активних користувачів на місяць [2]. Платформа надає розробникам потужний відкритий API для створення ботів, що суттєво спрощує процес розробки.

Telegram Bot API – це HTTP-інтерфейс, що дозволяє відправляти та отримувати повідомлення, управляти клавіатурами та реагувати на події в реальному часі. Боти реєструються через спеціальний сервіс @BotFather, який генерує унікальний токен аутентифікації. Взаємодія з API відбувається через стандартні HTTP-запити або через спеціалізовані бібліотеки [3].

Основними бібліотеками для розробки Telegram-ботів на мові Python є наступні:

- python-telegram-bot – офіційна асинхронна бібліотека для Python, що реалізує повний набір функцій Telegram Bot API. Бібліотека підтримує обробники команд, повідомлень та callback-запитів, вбудовані механізми управління станом розмови (ConversationHandler), а також нативну підтримку

асинхронного програмування через asyncio. Документація є вичерпною, а спільнота – активною.

- aiogram – ще одна популярна асинхронна бібліотека для Python з акцентом на продуктивність та зручність розробки. Відрізняється декларативним стилем оголошення обробників подій через декоратори, підтримкою middleware та власною системою FSM (Finite State Machine) для управління станами.
- telebot (pyTelegramBotAPI) – синхронна бібліотека, яка підходить для простих проєктів завдяки своїй простоті та мінімальним вимогам. Не рекомендується для масштабних рішень через обмежену підтримку асинхронності.

Для реалізації даного проєкту було обрано бібліотеку python-telegram-bot версії 22.7, що базується на asyncio. Ключовими критеріями вибору стали: широка функціональність, активна підтримка, наявність вбудованих інструментів для обробки inline-клавіатур та callback-запитів, а також висока якість документації та прикладів.

1.3 Аналіз предметної області

Для проєктування ефективного чатбота необхідно провести аналіз потреб цільової аудиторії – абітурієнтів та студентів Черкаського державного фахового бізнес-коледжу. Під предметною областю розуміється сукупність об'єктів, явищ і процесів, пов'язаних із вступом до коледжу, а також із повсякденною діяльністю студентів.

Абітурієнти – особи, які планують вступити до закладу фахової передвищої освіти, традиційно стикаються з низкою інформаційних труднощів. Серед типових запитань, з якими вони звертаються до приймальної комісії або адміністрації закладу, можна виділити такі категорії:

– інформація про наявні спеціальності та освітні програми;
– умови та порядок вступу, перелік необхідних документів;
– вартість навчання за контрактною формою;

– наявність гуртожитку та умови проживання;

– соціальна підтримка, стипендіальне забезпечення;

– контактна інформація приймальної комісії та адміністрації.

Для студентів характерними є інші потреби, що стосуються організації навчального процесу та повсякденного студентського життя::

– розклад занять та дзвінків;

– розташування навчальних кабінетів та аудиторій;

– місцезнаходження укриттів цивільного захисту (актуально в умовах воєнного стану);

– зупинки громадського транспорту поблизу коледжу;

– заклади харчування та інфраструктура поблизу навчального закладу;

– офіційні посилання на ресурси коледжу та соціальні мережі.

Аналіз цих потреб дозволяє зробити висновок, що більшість запитів є типовими та повторюваними, а відповідна інформація – структурованою та відносно статичною. Це підтверджує доцільність використання детермінованого чатбота на основі фіксованих меню, оскільки така архітектура є оптимальною для швидкого та надійного обслуговування зазначених класів запитів.

1.4 Огляд існуючих аналогів та обґрунтування необхідності розробки

Перш ніж приступати до проєктування власного рішення, необхідно провести огляд існуючих аналогів – як загальних платформ для створення чатботів, так і конкретних реалізацій у сфері освітніх закладів України.

Серед широко відомих комерційних платформ для створення чатботів можна виділити Dialogflow (Google), Microsoft Bot Framework, ManyChat та Botpress. Ці рішення надають широкий набір інструментів для побудови розмовних агентів, проте їх використання передбачає залежність від зовнішніх сервісів, необхідність хмарного розгортання, а в деяких випадках – значні фінансові витрати при масштабуванні. Крім того, інтеграція таких рішень із Telegram API потребує додаткових налаштувань і специфічних технічних знань.

Ряд університетів та коледжів України вже впровадили власних Telegram-ботів. Як правило, вони виконують функції надання розкладу занять, нагадувань про дедлайни або інформування про події закладу. Для детальнішого аналізу розглянуто три реальні приклади освітніх чатботів черкаського регіону та України загалом.

Першим аналогом є бот «Розклад ПНУ» – Telegram-бот Прикарпатського національного університету (рис. 1.1).

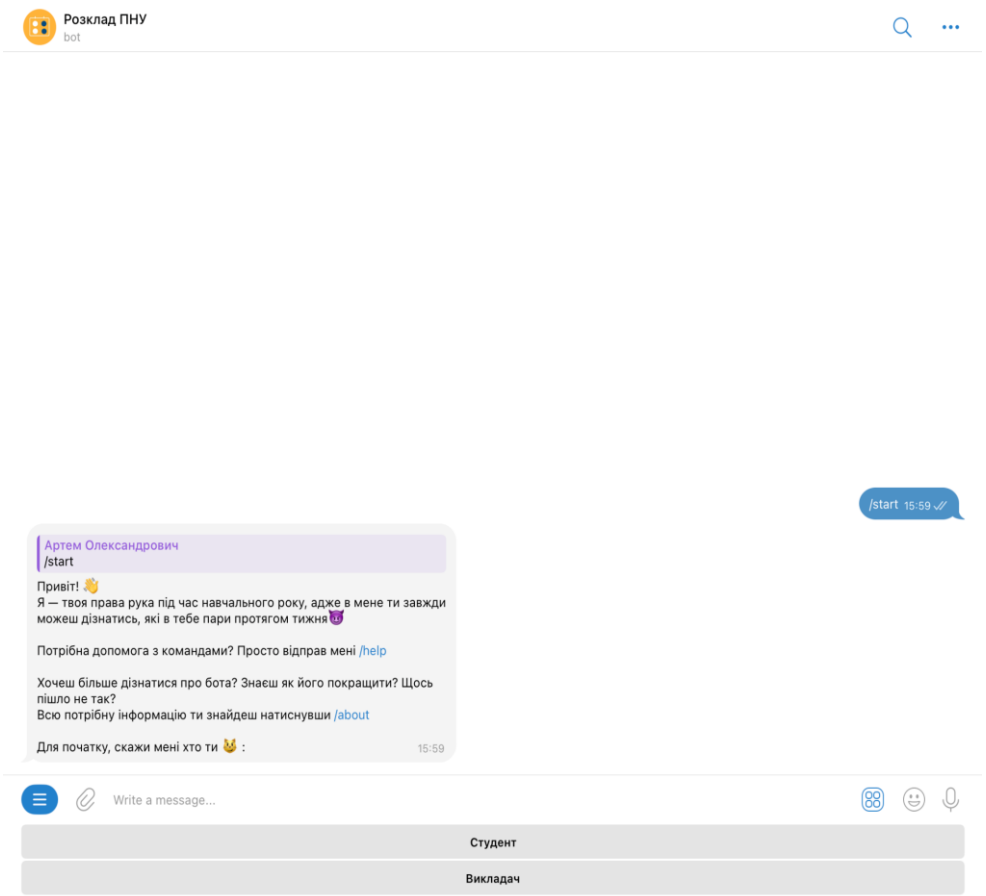


Рис. 1.1 – Інтерфейс Telegram-бота «Розклад ПНУ»

Після запуску командою /start бот надсилає привітальне повідомлення та пропонує обрати роль: «Студент» або «Викладач». Навігація реалізована через звичайні кнопки клавіатури (Reply Keyboard). Функціонал зосереджений виключно на наданні розкладу занять – підбір групи або викладача з відображенням актуального розкладу. Інформація для абітурієнтів відсутня повністю.

Другим аналогом є бот «Розклад ДТЕУ» – Telegram-бот Українського державного торговельно-економічного університету (рис. 1.2). Бот спеціалізується на наданні розкладу занять і після запуску пропонує обрати факультет із переліку семи варіантів через Reply Keyboard. Функціонал обмежений виключно розкладом – жодної загальної довідкової інформації про заклад, умови вступу чи інфраструктуру не передбачено.

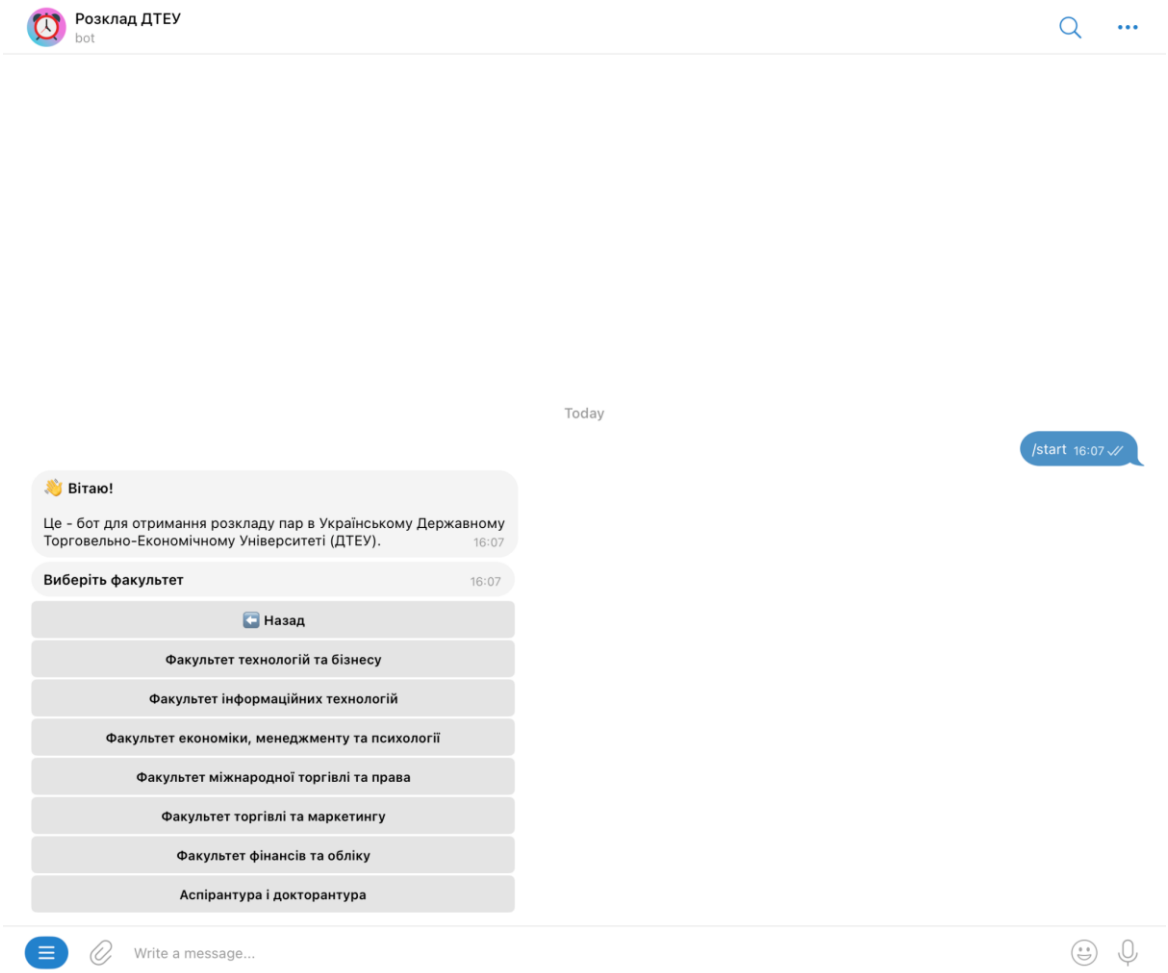


Рис. 1.2 – Інтерфейс Telegram-бота «Розклад ДТЕУ»

Третім аналогом є бот «Розклад ЧДТУ» – Telegram-бот Черкаського державного технологічного університету (рис. 1.3). Це найбільш функціонально розвинений із розглянутих аналогів: після запуску бот надсилає привітання з іменем користувача, відображає перелік доступних команд (/schedule, /setfaculty, /setgroup, /vip, /author) та пропонує швидкий вибір групи або факультету через

Reply Keyboard. Бот також містить канал новин та функцію підтримки автора. Попри ширший набір функцій порівняно з попередніми аналогами, бот орієнтований виключно на студентів ЧДТУ для отримання розкладу і не містить жодної інформації для абітурієнтів, відомостей про вступ, вартість навчання або інфраструктуру закладу.

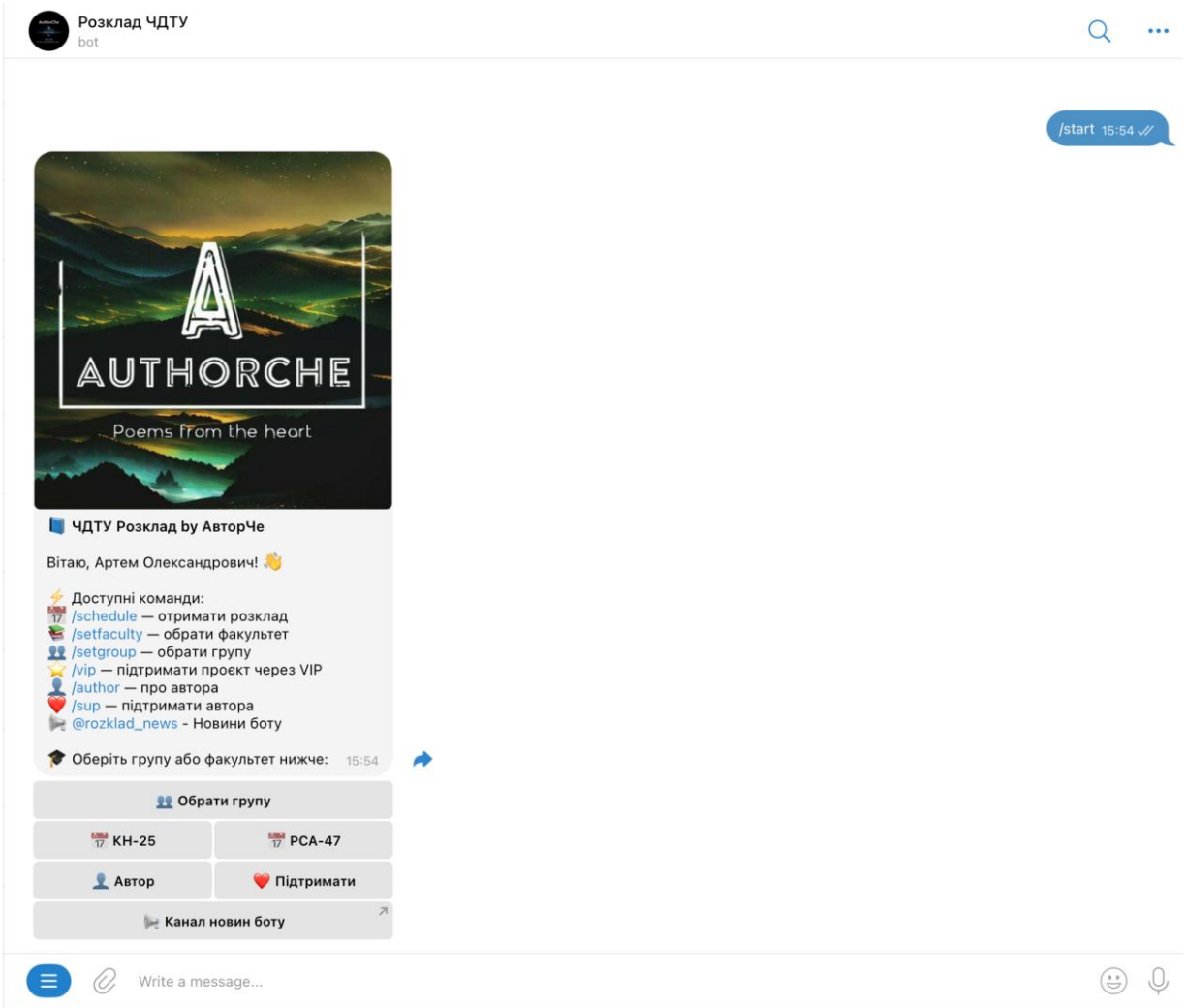


Рис. 1.3 – Інтерфейс Telegram-бота «Розклад ЧДТУ»

Порівняльний аналіз трьох розглянутих аналогів та розроблюваного чатбота ЧДБК наведено у табл. 1.1.

Таблиця 1.1 – Порівняльний аналіз існуючих освітніх Telegram-ботів

Критерій	Розклад ПНУ	Розклад ДТЕУ	Розклад ЧДТУ	Чатбот ЧДБК
Цільова аудиторія	Студенти, викладачі	Студенти	Студенти	Абітурієнти + студенти
Тип клавіатури	Reply Keyboard	Reply Keyboard	Reply Keyboard	Inline Keyboard
Розклад занять	Так	Так	Так	Так
Інформація для абітурієнтів	Ні	Ні	Ні	Так
Умови вступу та документи	Ні	Ні	Ні	Так
Вартість навчання	Ні	Ні	Ні	Так
Гуртожиток та інфраструктура	Ні	Ні	Ні	Так
Укриття цивільного захисту	Ні	Ні	Ні	Так
Навігація «Назад»	Ні	Ні	Ні	Так
Ієрархічне меню	Ні	Ні	Частково	Так
Автоматизовані тести	Ні	Ні	Ні	Так

Як видно з табл. 1.1, всі три розглянуті аналоги мають спільну принципову обмеженість: вони орієнтовані виключно на поточних студентів і вирішують лише одне завдання – надання розкладу занять. Жоден із них не охоплює сегмент абітурієнтів, не надає загальної довідкової інформації про заклад та

використовує Reply Keyboard без можливості ієрархічної навігації з кнопками «Назад». Офіційний вебсайт ЧДБК (csbc.edu.ua) містить основну інформацію про навчальний заклад, однак потребує виключно браузерного доступу і не адаптований для швидкого пошуку конкретних даних на мобільних пристроях.

Таким чином, на сьогодні в ЧДБК відсутній спеціалізований інтерактивний довідковий інструмент у форматі месенджер-бота, орієнтований одночасно на дві цільові аудиторії – абітурієнтів та студентів. Розробка такого чатбота є актуальною і практично значущою задачею, що сприятиме підвищенню якості інформаційного супроводу вступної кампанії та навчального процесу.

1.5 Вимоги до функціональності чатбота

На основі проведеного аналізу предметної області та потреб цільових аудиторій було сформовано перелік функціональних та нефункціональних вимог до системи.

До функціональних вимог належать такі:

- надання інформації для абітурієнтів: перелік спеціальностей, умови вступу, вартість навчання, умови проживання в гуртожитку, соціальна підтримка, контакти приймальної комісії;
- надання інформації для студентів: розклад занять та дзвінків, список кабінетів, розташування укриттів, зупинки громадського транспорту, цікаві місця поблизу коледжу;
- навігація через ієрархічне меню з inline-кнопками;
- відображення графічних матеріалів (карта зупинок, схема цікавих місць);
- обробка невідомих текстових повідомлень із перенаправленням до меню;
- надання посилань на офіційні ресурси та соціальні мережі коледжу.

Нефункціональні вимоги включають:

- зрозумілий та інтуїтивний інтерфейс, адаптований для мобільних пристроїв;
- час відповіді бота не більше 2 секунд при стандартному навантаженні;
- стійкість до помилок при недоступності медіафайлів (відображення текстової альтернативи);
- легкість оновлення текстового контенту без зміни архітектури системи.

Висновки до розділу 1

У першому розділі проведено теоретичний аналіз предметної області та обґрунтовано технологічні рішення для розробки чатбота-помічника для абітурієнтів і студентів Черкаського державного фахового бізнес-коледжу.

Розглянуто поняття чатбота, його класифікацію та роль у сучасних інформаційних системах. Встановлено, що для вирішення поставленої задачі оптимальним є детермінований підхід на основі структурованих меню, що забезпечує передбачуваність поведінки та простоту підтримки.

Проведено огляд платформ та бібліотек для розробки Telegram-ботів. Обґрунтовано вибір бібліотеки python-telegram-bot версії 22.7 як основного інструменту реалізації завдяки її повнофункціональності, асинхронній архітектурі та широкій документації.

Здійснено аналіз інформаційних потреб двох цільових аудиторій – абітурієнтів та студентів – та сформовано перелік функціональних і нефункціональних вимог до системи. Проведений огляд існуючих аналогів підтвердив відсутність спеціалізованого інтерактивного довідкового інструменту в ЧДБК та обґрунтував доцільність даної розробки.

Результати аналізу, викладені в цьому розділі, складають теоретичну основу для проєктування та реалізації системи, що описуються в наступних розділах.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ТЕЛЕГРАМ-ЧАТБОТА

2.1 Обґрунтування вибору технологічного стеку та інструментів розробки

Розробка програмної системи будь-якого рівня складності розпочинається з вибору технологічного стеку – сукупності мов програмування, фреймворків, бібліотек та інструментів, що використовуються для реалізації проєкту. Правильний вибір стеку безпосередньо впливає на якість кінцевого продукту, швидкість розробки та простоту подальшої підтримки. У межах даної кваліфікаційної роботи вибір технологій здійснювався на основі таких критеріїв: відповідність функціональним вимогам до системи, наявність якісної документації та активної спільноти розробників, мінімальні витрати на розгортання, а також технічна доступність для подальшої підтримки та розширення функціональності.

Основною мовою програмування для реалізації чатбота обрано Python версії 3.11. Python є однією з найпоширеніших мов програмування у світі: за даними індексу TIOBE станом на 2024 рік вона стабільно посідає перше місце за популярністю серед розробників [7]. Мова відрізняється лаконічним і читабельним синтаксисом, що суттєво скорочує час розробки та спрощує супровід коду. Для цілей даного проєкту Python надає такі ключові переваги: вбудована підтримка асинхронного програмування через модуль asyncio, що є обов'язковою умовою для ефективної роботи з Telegram Bot API у версії 22.7; широка екосистема бібліотек для роботи з мережею, файловою системою та логуванням; низький поріг входження, що спрощує подальшу підтримку проєкту. Альтернативними мовами розглядалися Node.js (JavaScript) та Go, однак вони поступаються Python за зручністю роботи з відповідними Telegram-бібліотеками та рівнем документації у контексті навчальних проєктів.

Для взаємодії з Telegram Bot API обрано бібліотеку python-telegram-bot версії 22.7. Ця бібліотека є повною реалізацією Telegram Bot API для мови Python та надає зручний об'єктно-орієнтований інтерфейс для роботи з усіма типами оновлень [8]. Серед її ключових можливостей: підтримка асинхронної обробки оновлень на базі asyncio; вбудований диспетчер подій (Application) з підтримкою CommandHandler, MessageHandler та CallbackQueryHandler; зручна робота з InlineKeyboardMarkup та CallbackQuery; автоматичне підтвердження отримання callback-запитів; вбудовані засоби логування та обробки виключень. Для порівняння розглядалися також бібліотеки aiogram та pyTelegramBotAPI (telebot). Порівняльний аналіз альтернативних бібліотек наведено у табл. 2.1.

Таблиця 2.1 – Порівняльний аналіз бібліотек для розробки Telegram-ботів на Python

Критерій	python-telegram-bot	aiogram	pyTelegramBotAPI
Асинхронність	Так (asyncio)	Так (asyncio)	Частково
Повнота реалізації API	Повна	Повна	Часткова
Якість документації	Висока	Висока	Середня
Складність освоєння	Середня	Висока	Низька
Активність спільноти	Дуже висока	Висока	Середня
Підтримка FSM	ConversationHandler	Вбудована FSM	Відсутня
Підходить для проєкту	Обрано	–	–

За сукупністю критеріїв бібліотека python-telegram-bot є оптимальним вибором: вона поєднує повну реалізацію API, асинхронну архітектуру, зручні

інструменти для роботи з inline-клавіатурами та найширшу документаційну базу серед розглянутих альтернатив.

Як середовище розробки обрано PyCharm – спеціалізована IDE для мови Python від компанії JetBrains, що є одним із найпотужніших і найбільш функціональних інструментів для розробки Python-застосунків [9]. PyCharm надає комплексний набір інструментів, що використовувалися в процесі розробки чатбота: глибокий аналіз коду з інтелектуальним автодоповненням та виявленням помилок у реальному часі; вбудований відладчик (debugger) з підтримкою точок зупинки та покрокового виконання асинхронного коду; інтегрований термінал для запуску бота та встановлення залежностей через pip; підтримка віртуальних середовищ (venv) з автоматичним керуванням пакетами; вбудована перевірка відповідності коду стандарту PEP 8. На відміну від універсальних редакторів коду, PyCharm розроблений спеціально для Python, що забезпечує значно глибшу інтеграцію з інтерпретатором, бібліотеками та інструментами налагодження. Для навчального проєкту використовувалася безкоштовна версія PyCharm Community Edition, яка містить увесь необхідний для розробки функціонал.

Таблиця 2.2 – Узагальнена характеристика обраного технологічного стеку

Компонент	Обраний інструмент	Версія	Призначення
Мова програмування	Python	3.11	Основна мова реалізації
Telegram-бібліотека	python-telegram-bot	22.7	Взаємодія з Bot API
IDE	PyCharm Community Edition	2024.1	Розробка та налагодження

Обраний технологічний стек – Python 3.11 + python-telegram-bot 22.7 + PyCharm Community Edition – є оптимальним для реалізації поставленого завдання. Він забезпечує повну відповідність функціональним вимогам до

системи, мінімальні витрати на інфраструктуру, високу читабельність і підтримуваність коду, а також можливість подальшого масштабування функціональності бота без зміни базової архітектури. Узагальнену характеристику обраного стеку наведено у табл. 2.2.

2.2 Проєктування архітектури програмної системи

Проєктування архітектури є ключовим етапом розробки будь-якої програмної системи, що передує безпосередній реалізації коду. Грамотно спроєктована архітектура забезпечує чіткий розподіл відповідальності між компонентами, спрощує супровід і розширення системи, а також мінімізує ризики виникнення помилок на етапі реалізації. Відповідно до загальноприйнятих принципів інженерії програмного забезпечення, архітектура системи визначає її структуру, поведінку та взаємодію між складовими частинами [11]. У межах даної кваліфікаційної роботи архітектура чатбота проєктувалася з урахуванням двох ключових вимог: максимальна простота підтримки інформаційного контенту та чітка передбачуваність поведінки системи при будь-яких сценаріях використання.

Аналіз потреб, що був проведений у пункті 1.3 дозволяє зробити висновок, що більшість запитів є типовими та повторюваними, а відповідна інформація – структурованою та відносно статичною. Це підтверджує доцільність використання детермінованого чатбота на основі фіксованих меню, оскільки така архітектура є оптимальною для швидкого та надійного обслуговування зазначених класів запитів.

На основі проведеного аналізу розроблено діаграму варіантів використання (Use Case Diagram), яка наочно відображає взаємодію двох акторів – Абітурієнта та Студента – із функціями системи (рис. 2.1).

Розроблений чатбот функціонує як клієнт-серверна система, де клієнтом виступає користувач у Telegram-месенджері, а сервером – програма на Python, що взаємодіє з Telegram Bot API. Взаємодія між компонентами відбувається

через стандартний HTTPS-протокол: бот надсилає запити до серверів Telegram, а у відповідь отримує об'єкти типу Update, що містять інформацію про дії користувача [10].

Архітектура системи включає такі основні компоненти та зв'язки між ними:

Application – центральний компонент системи, що ініціалізується під час запуску бота. Він реєструє всі обробники подій, запускає механізм polling та маршрутизує вхідні оновлення до відповідних функцій-обробників залежно від типу події. Application є точкою входу всієї системи і керує її життєвим циклом.

CommandHandler – компонент, що відповідає за обробку текстових команд користувача, які починаються з символу «/». У даній системі зареєстровано два обробники команд: /start – запускає привітальне повідомлення та відображає головне меню вибору ролі; /help – надає коротку довідку про можливості бота та доступні команди.

CallbackQueryHandler – найважливіший компонент системи з точки зору взаємодії, що обробляє всі події натискання inline-кнопок. Кожне натискання кнопки генерує CallbackQuery-подію з унікальним рядком-ідентифікатором (callback_data), який передається до функції handle_callback() для визначення необхідної дії. Цей компонент реалізує всю навігаційну логіку між розділами меню.

MessageHandler – компонент-перехоплювач, що обробляє будь-які текстові повідомлення, які не є командами. Оскільки система побудована на основі меню, а не на обробці вільного тексту, MessageHandler реагує на будь-який довільний текст однотипно: повідомляє користувача про неможливість обробки запиту та пропонує скористатися меню через відповідну кнопку.

Модуль контенту – логічно відокремлена частина коду, що містить усі інформаційні рядки у вигляді констант верхнього рівня (SHELTERS_INFO, FLOOR_TEXTS, SCHEDULE_INFO, BELLS_INFO, STOPS_INFO, INTERESTING_PLACES, LINKS_INFO, RATING_INFO) та функції-генератори клавіатур (student_keyboard(), enrollee_keyboard(), floor_keyboard()). Винесення

контенту в окремі константи відокремлює дані від логіки обробки та дозволяє оновлювати інформацію без зміни архітектури системи.

Модуль логування – стандартний модуль Python logging, налаштований для виведення діагностичної інформації про роботу системи у форматі з міткою часу, назвою модуля та рівнем повідомлення. Логування дозволяє відстежувати помилки при роботі з медіафайлами, збої у взаємодії з Bot API та інші виключні ситуації.

Схему взаємодії компонентів системи та повний алгоритм обробки вхідного запиту відображено на блок-схемі (рис. 2.1).

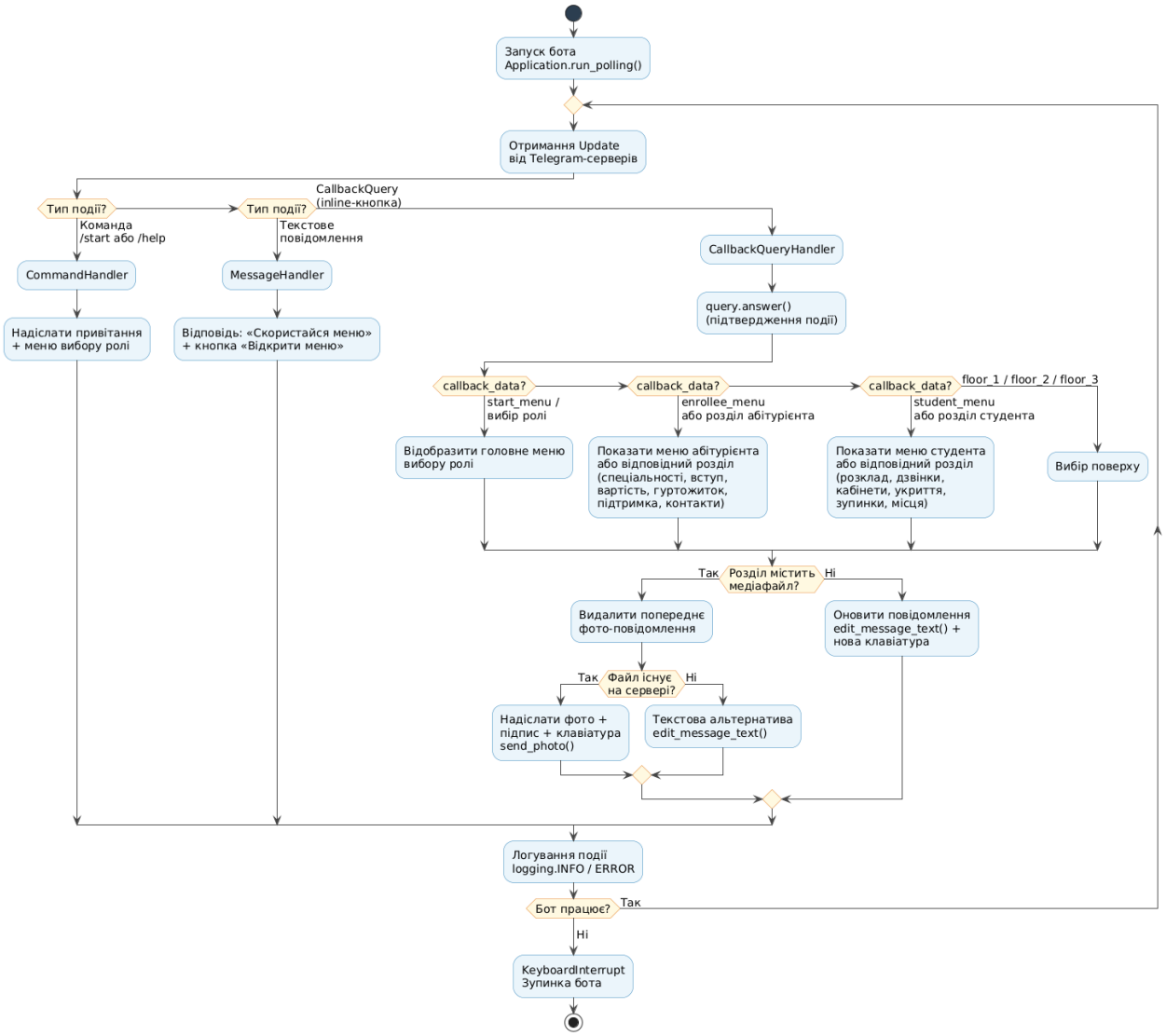


Рисунок 2.1 – Блок-схема алгоритму роботи Telegram-чатбота ЧДБК

Програмний код системи організовано в єдиному файлі bot.py з чіткою логічною структурою, розділеною на функціональні блоки. Така організація є виправданою для проєктів даного масштабу, оскільки дозволяє зосередити всю логіку в одному місці без надмірного ускладнення структури пакетів. Логічну структуру файлу наведено у табл. 2.3.

Таблиця 2.3 – Логічна структура файлу bot.py

Блок	Вміст	Призначення
Імпорти	telegram, logging, asyncio	Підключення залежностей
Константи контенту	SHELTERS_INFO, BELLS_INFO та ін.	Інформаційні рядки у форматі HTML
Генератори клавіатур	student_keyboard(), enrollee_keyboard(), floor_keyboard()	Формування inline-меню
Допоміжні функції	delete_last_photo(), send_photo_or_text()	Робота з медіаконтентом
Обробник /start	start()	Привітання та головне меню
Обробник /help	help_command()	Довідка
Головний обробник	handle_callback()	Навігація між розділами
Обробник тексту	handle_text_message()	Реакція на довільний текст
Обробник помилок	error_handler()	Логування виключень
Точка входу	main()	Ініціалізація та запуск

Константи контенту зберігають текст у форматі HTML із підтримкою тегів розмітки, що дозволяє відображати жирний текст, курсив та гіперпосилання безпосередньо у повідомленнях Telegram без додаткової обробки. Функції-

генератори клавіатур повертають об'єкти InlineKeyboardMarkup із відповідним набором кнопок та їх callback_data ідентифікаторами, що забезпечує чітке розмежування між представленням даних та логікою навігації.

Діаграма варіантів використання (Use Case Diagram) є стандартним інструментом UML-моделювання, що відображає функціональні можливості системи з точки зору користувача. Кожен еліпс на діаграмі позначає окремий варіант використання – дію, яку актор може виконати за допомогою системи. На діаграмі виділено двох основних акторів (Абітурієнт та Студент) і зовнішню систему (Telegram Bot API), з якою чатбот взаємодіє для надсилання та отримання повідомлень. Спільні варіанти використання – запуск бота та перегляд посилань – доступні обом акторам.

Для формального опису функціональних можливостей системи з точки зору користувача розроблено діаграму варіантів використання (Use Case Diagram) відповідно до нотації UML. Діаграма відображає двох основних акторів системи – Абітурієнта та Студента – і сукупність варіантів використання, доступних кожному з них (рис. 2.2).

Абітурієнту доступні такі варіанти використання: перегляд переліку спеціальностей, ознайомлення з умовами вступу, отримання інформації про вартість навчання, перегляд відомостей про гуртожиток, ознайомлення з системою соціальної підтримки та отримання контактних даних приймальної комісії. Студенту доступні: перегляд розкладу занять та дзвінків, навігація по кабінетах і аудиторіях, отримання інформації про укриття цивільного захисту, пошук зупинок громадського транспорту та перегляд цікавих місць поблизу коледжу. Спільними для обох акторів є варіанти використання «Розпочати роботу (/start)» та «Переглянути важливі посилання», оскільки вони доступні незалежно від обраної ролі. Зовнішнім актором системи виступає Telegram Bot API, з яким чатбот взаємодіє для надсилання та отримання повідомлень.

При проєктуванні архітектури системи застосовано ряд усталених патернів проєктування, що підвищують якість і підтримуваність коду [11].

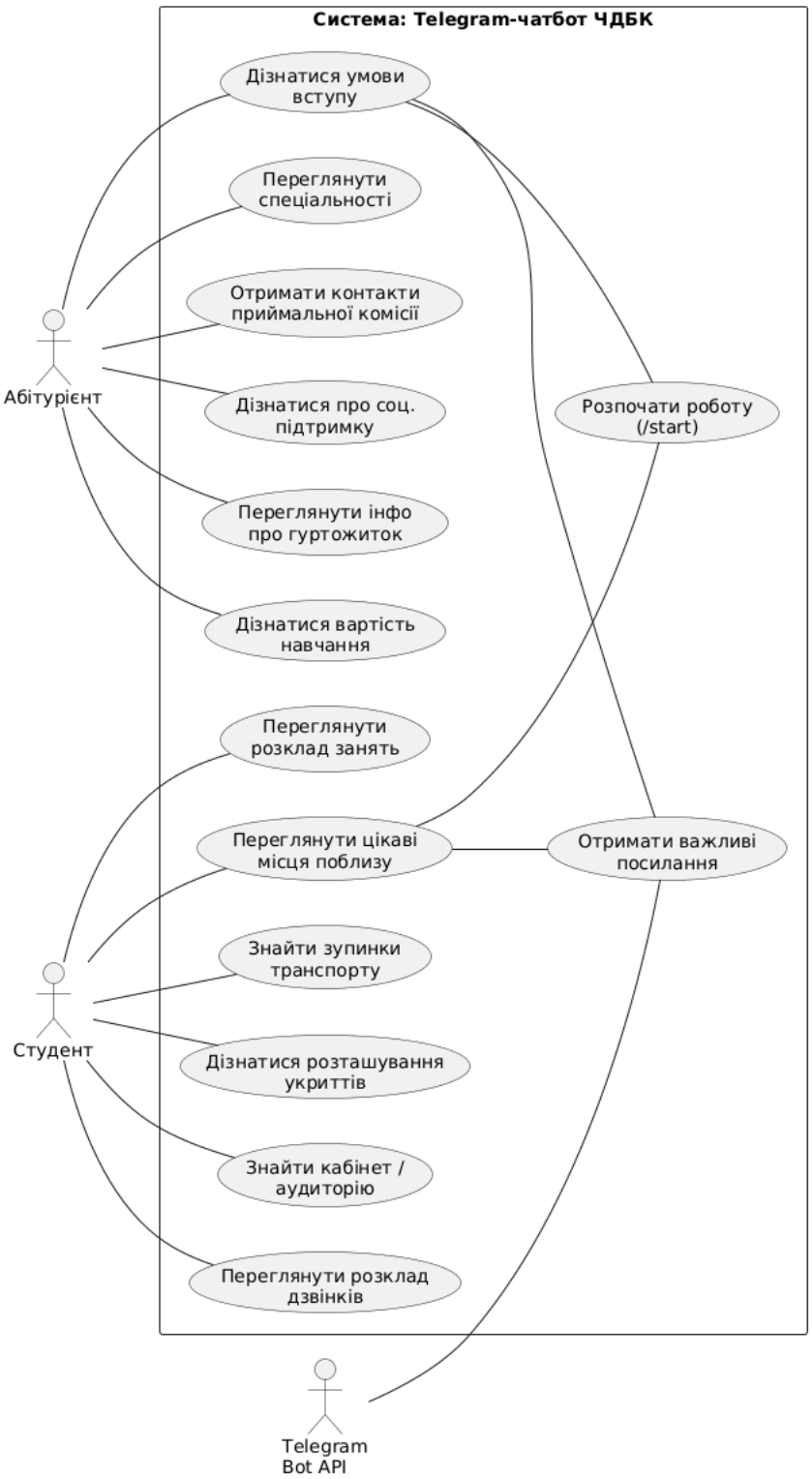


Рисунок 2.2 – Діаграма варіантів використання Telegram-чатбота ЧДБК

Dispatcher – основний архітектурний патерн системи, реалізований через клас Application бібліотеки python-telegram-bot. Диспетчер приймає всі вхідні події та маршрутизує їх до зареєстрованих обробників відповідно до типу події.

Це дозволяє повністю відокремити логіку маршрутизації від логіки обробки конкретних запитів.

Command – патерн, що реалізується через механізм callback_data у inline-кнопках. Кожна кнопка інкапсулює команду у вигляді рядка-ідентифікатора, що передається до центрального обробника handle_callback(). Це забезпечує уніфікований інтерфейс для виконання різноманітних дій навігації.

Template Method – застосовується у функції send_photo_or_text(), яка визначає загальний алгоритм відображення контенту (видалити попереднє фото, спробувати надіслати нове, у разі помилки відобразити текст), залишаючи конкретні параметри (шлях до файлу, текст, клавіатура) змінними.

Separation of Concerns – архітектурний принцип, реалізований через чітке розмежування між модулем контенту (константи та клавіатури), модулем логіки (обробники подій) та модулем інфраструктури (налаштування Application, логування). Завдяки цьому оновлення інформаційного контенту не потребує зміни логіки обробки подій, і навпаки.

Застосування зазначених патернів забезпечує архітектурну цілісність системи, спрощує її подальше розширення (наприклад, додавання нових розділів меню) та підвищує загальну якість програмного коду відповідно до стандартів розробки програмного забезпечення.

2.3 Реалізація програмного коду чатбота

Реалізація програмного коду є центральним етапом розробки програмної системи, на якому спроєктована архітектура втілюється у конкретні програмні конструкції. У межах даного підрозділу описано реалізацію всіх ключових компонентів чатбота: обробників подій, логіки навігації через inline-клавіатури, меню для абітурієнтів та студентів, механізму роботи з медіаконтентом, а також засобів забезпечення відмовостійкості та логування. Код написано відповідно до стандарту оформлення Python-коду PEP 8, що забезпечує його читабельність і підтримуваність [14].

Точкою входу програми є функція main(), яка виконує ініціалізацію застосунку та реєстрацію всіх обробників подій. Для створення екземпляра Application використовується патерн Builder, що є рекомендованим підходом у бібліотеці python-telegram-bot версії 22.7 [13]. Порядок реєстрації обробників є принципово важливим: CommandHandler реєструється першим, оскільки команди мають вищий пріоритет обробки; CallbackQueryHandler обробляє всі натискання inline-кнопок; MessageHandler із фільтром None перехоплює всі інші повідомлення, що не підпали під попередні обробники. Такий порядок гарантує коректну маршрутизацію будь-якого типу вхідної події. Лістинг 2.1 демонструє реалізацію функції main().

Лістинг 2.1 – Ініціалізація системи та реєстрація обробників

```
def main():
    TOKEN = "YOUR_BOT_TOKEN"
    application = Application.builder().token(TOKEN).build()

    application.add_handler(CommandHandler("start", start))
    application.add_handler(CommandHandler("help", help_command))
    application.add_handler(CallbackQueryHandler(handle_callback))
    application.add_handler(MessageHandler(None, handle_text_message))
    application.add_error_handler(error_handler)

    application.run_polling()
```

Обробник команди /start формує персоналізоване привітальне повідомлення з іменем користувача та відображає головне меню вибору ролі. Використання async def є обов'язковим у версії бібліотеки 22.7, оскільки вся система побудована на асинхронній обробці подій. Параметр parse_mode=ParseMode.HTML дозволяє застосовувати HTML-розмітку у тексті повідомлення для виділення важливих елементів. Реалізацію обробника наведено у лістингу 2.2.

Лістинг 2.2 – Обробник команди /start

```
async def start(update: Update, context) -> None:
    user = update.effective_user
    welcome_text = (
        f"Привіт, <b>{user.first_name}</b>!\n\n"
```

```
        f"Вас вітає <b>{COLLEGE_NAME}</b>! \n\n"
        "Я твій особистий помічник. Виберіть, хто ви:"
    )
    keyboard = InlineKeyboardMarkup([
        [InlineKeyboardButton("Я абітурієнт",
                              callback_data="enrollee_menu")],
        [InlineKeyboardButton("Я студент",
                              callback_data="student_menu")],
    ])
    await update.message.reply_text(
        welcome_text, reply_markup=keyboard,
        parse_mode=ParseMode.HTML)
```

Для кожного меню системи реалізовано окрему функцію-генератор, що повертає об'єкт InlineKeyboardMarkup. Такий підхід відповідає принципу розподілу відповідальності та дозволяє повторно використовувати клавіатури в різних частинах коду. Кожна кнопка містить текст із емодзі для візуальної ідентифікації розділу та унікальний рядок callback_data – ідентифікатор дії. Розміщення кожної кнопки в окремому рядку забезпечує вертикальне розташування меню, зручне для мобільного використання (лістинг 2.3).

Лістинг 2.3 – Генератор клавіатури меню студента

```
def student_keyboard():
    return InlineKeyboardMarkup([
        [InlineKeyboardButton("Розклад",      callback_data="schedule")],
        [InlineKeyboardButton("Дзвінки",      callback_data="bells")],
        [InlineKeyboardButton("Кабінети",     callback_data="cabinets")],
        [InlineKeyboardButton("Укриття",      callback_data="shelters")],
        [InlineKeyboardButton("Цікаві місця", callback_data="places")],
        [InlineKeyboardButton("Зупинки",      callback_data="stops")],
        [InlineKeyboardButton("Посилання",    callback_data="links")],
        [InlineKeyboardButton("Назад",       callback_data="start_menu")],
    ])
```

Окремої уваги заслуговує генератор клавіатури для навігації по поверхах, який реалізує динамічне виділення активного поверху квадратними дужками (наприклад, «[2]»), що надає користувачу чіткий візуальний орієнтир про поточний стан навігації (лістинг 2.4).

Лістинг 2.4 – Генератор клавіатури навігації по поверхах

```
def floor_keyboard(active_floor=None):
```

```
    def style(n):
        label = f"{n}"
        return InlineKeyboardButton(
            f"[ {label} ]" if active_floor == n else label,
            callback_data=f"floor_{n}"
        )
    return InlineKeyboardMarkup([
        [style(1), style(2), style(3), style(4)],
        [InlineKeyboardButton("Назад", callback_data="cabinets")],
    ])
```

Функція handle_callback() є ключовим компонентом системи, що обробляє всі натискання inline-кнопок. Першою дією у функції є виклик query.answer(), який підтверджує отримання callback-події та прибирає індикатор завантаження з кнопки на стороні користувача. Це є обов'язковою вимогою Telegram Bot API: якщо не викликати answer() протягом 10 секунд – Telegram відобразить помилку на стороні клієнта [13]. Далі зі значення query.data визначається тип дії та виконується відповідна операція. У разі невдачі редагування система надсилає нове повідомлення, що гарантує отримання відповіді за будь-яких обставин (лістинг 2.5).

Лістинг 2.5 – Фрагмент головного обробника callback-запитів

```
async def handle_callback(update: Update, context) -> None:
    query = update.callback_query
    await query.answer()   # обов'язкове підтвердження події
    data = query.data
    chat_id = query.message.chat_id

    if data == "start_menu":
        await delete_last_photo(context, chat_id)
        keyboard = InlineKeyboardMarkup([
            [InlineKeyboardButton("Я абітурієнт",
                                  callback_data="enrollee_menu")],
            [InlineKeyboardButton("Я студент",
                                  callback_data="student_menu")],
        ])
        try:
            await query.edit_message_text(
                "Виберіть, хто ви:",
                reply_markup=keyboard,
                parse_mode=ParseMode.HTML)
        except Exception:
            await context.bot.send_message(
                chat_id=chat_id, text="Виберіть, хто ви:",
                reply_markup=keyboard,
                parse_mode=ParseMode.HTML)
```

Кожен розділ меню реалізує відображення структурованого інформаційного контенту через метод edit_message_text(), що оновлює вміст і клавіатуру вже існуючого повідомлення замість надсилання нового. Це забезпечує плавну навігацію без накопичення повідомлень у чаті користувача. Кожен розділ містить кнопку «Назад» для повернення до попереднього рівня ієрархії. Лістинг 2.6 демонструє реалізацію розділу спеціальностей меню абітурієнта.

Лістинг 2.6 – Реалізація розділу спеціальностей меню абітурієнта

```
elif data == "enrollee_specialties":
    kb = InlineKeyboardMarkup([[InlineKeyboardButton(
        "Назад", callback_data="back_to_enrollee_menu")]])
    text = """<b>СПЕЦІАЛЬНОСТІ:</b>

<b>КОМП'ЮТЕРНІ СПЕЦІАЛЬНОСТІ:</b>
F2 Інженерія програмного забезпечення
F3 Комп'ютерні науки
F7 Комп'ютерна інженерія

<b>ЕКОНОМІЧНІ ТА БІЗНЕС СПЕЦІАЛЬНОСТІ:</b>
D1 Облік і оподаткування
D3 Менеджмент, D5 Маркетинг"""

    await query.edit_message_text(
        text, reply_markup=kb, parse_mode=ParseMode.HTML)
```

Для розділів, що містять графічні матеріали (карта зупинок, зображення цікавих місць), реалізовано функцію send_photo_or_text(). Вона забезпечує відображення медіаконтенту з автоматичним переходом на текстову альтернативу у разі помилки. Ідентифікатор надісланого фото-повідомлення зберігається у контексті користувача (user_data) для подальшого видалення при наступному переході між розділами (лістинг 2.7).

Лістинг 2.7 – Функція відображення медіаконтенту

```
async def send_photo_or_text(
        query, context, photo_path, caption, reply_markup):
    chat_id = query.message.chat_id
    await delete_last_photo(context, chat_id)
    try:
        photo_msg = await context.bot.send_photo(
```

```
                chat_id=chat_id,
                photo=open(photo_path, "rb"),
                caption=caption,
                parse_mode=ParseMode.HTML,
                reply_markup=reply_markup)
            context.user_data["last_photo_message_id"] = \
                photo_msg.message_id
            try:
                await query.delete_message()
            except Exception:
                pass
        except FileNotFoundError:
            logger.warning(f"Файл не знайдено: {photo_path}")
            await query.edit_message_text(
                caption, reply_markup=reply_markup,
                parse_mode=ParseMode.HTML)
        except Exception as e:
            logger.error(f"Помилка фото: {e}")
            await query.edit_message_text(
                caption, reply_markup=reply_markup,
                parse_mode=ParseMode.HTML)
```

Система реалізує кілька рівнів захисту від збоїв. На глобальному рівні зареєстровано обробник помилок error_handler(), що перехоплює всі необроблені виключення та логує їх. Логування налаштовано через стандартний модуль Python logging із рівнями INFO для штатних подій та ERROR для виключних ситуацій [14]. Формат запису включає мітку часу, назву модуля та рівень повідомлення, що дозволяє ефективно діагностувати проблеми під час експлуатації (лістинг 2.8).

Лістинг 2.8 – Налаштування логування та глобальний обробник помилок

```
logging.basicConfig(
    format="%(asctime)s - %(name)s - %(levelname)s - %(message)s",
    level=logging.INFO
)
logger = logging.getLogger(__name__)

async def error_handler(update: object, context) -> None:
    logger.error(f"Update {update} caused error {context.error}")
```

Обробник невідомих текстових повідомлень запобігає «зависанню» сесії у разі, якщо користувач надсилає довільний текст замість натискання кнопок. Він

завжди повертає користувача до зрозумілого стану незалежно від його дій, що є важливою вимогою до якості користувацького досвіду (лістинг 2.9).

Лістинг 2.9 – Обробник невідомих текстових повідомлень

```
async def handle_text_message(update: Update, context) -> None:
    keyboard = InlineKeyboardMarkup([[InlineKeyboardButton(
        "Відкрити меню", callback_data="start_menu")]])
    await update.message.reply_text(
        "Я не збагнув твоє запитання \n\nСкористайся меню!",
        reply_markup=keyboard,
        parse_mode=ParseMode.HTML)
```

2.4 Розгортання та конфігурація системи

Розгортання програмної системи є завершальним етапом розробки, що передбачає підготовку середовища виконання, налаштування конфігурації та запуск застосунку в реальних умовах. Коректно виконане розгортання є необхідною умовою для забезпечення стабільної та безперервної роботи чатбота в режимі реального часу. У межах даного підрозділу описано повний процес розгортання Telegram-чатбота ЧДБК – від реєстрації бота у Telegram до запуску та перевірки його працездатності [15].

Першим кроком розгортання є реєстрація нового бота у Telegram через офіційний сервіс @BotFather. Процес реєстрації передбачає такі кроки: відкрити діалог із @BotFather та надіслати команду /newbot; ввести відображувану назву бота (наприклад, «ЧДБК Помічник»); ввести унікальне ім'я користувача, що закінчується на «bot» (наприклад, «csbc_helper_bot»); отримати токен аутентифікації – унікальний рядок виду «1234567890:AAF...». Отриманий токен є конфіденційною інформацією, що надає повний контроль над ботом, тому він не повинен передаватися третім особам або зберігатися у публічних репозиторіях.

Після реєстрації через @BotFather рекомендується виконати додаткові налаштування: встановити аватар бота командою /setuserpic; додати опис

(/setdescription) та коротку інформацію (/setabouttext); вимкнути можливість додавання бота до групових чатів (/setjoingroups – Disable), оскільки система розроблена виключно для особистого використання.

Перед запуском необхідно підготувати робочу директорію з такою структурою файлів (лістинг 2.10).

Файл requirements.txt містить перелік усіх бібліотек, необхідних для роботи системи, із зазначенням їх версій:

```
python-telegram-bot==22.7
```

Лістинг 2.10 – Структура файлів проєкту

```
csbc_bot/
├── bot.py              # головний файл з кодом бота
├── requirements.txt    # список залежностей
├── map.png             # карта зупинок транспорту
└── intr_places.png     # зображення цікавих місць
```

Для розгортання бота необхідне встановлене середовище Python версії 3.11 або вище. Рекомендованою практикою є використання віртуального середовища (venv), що дозволяє ізолювати залежності проєкту від системного Python та уникнути конфліктів версій між різними проєктами [16]. Послідовність команд для підготовки середовища наведено у лістингу 2.11.

Лістинг 2.11 – Підготовка віртуального середовища та встановлення залежностей

```
# Створення віртуального середовища
python -m venv venv

# Активація (Windows)
venv\Scripts\activate

# Активація (Linux / macOS)
source venv/bin/activate

# Встановлення залежностей
pip install -r requirements.txt
```

Для отримання оновлень від Telegram-серверів у системі використовується механізм polling. Механізм polling є оптимальним для навчального та малого

продуктивного розгортання: він не вимагає статичного IP-адреса та SSL-сертифіката, просто налаштовується та забезпечує затримку відповіді менше 1 секунди, що повністю відповідає нефункціональним вимогам системи. Запуск бота виконується командою: `python bot.py`

Після запуску бота виконується функціональна перевірка: знайти бота в Telegram за його іменем користувача; надіслати команду /start та переконатися у відображенні привітального повідомлення та меню вибору ролі; послідовно перевірити всі розділи меню абітурієнта та студента; перевірити коректність відображення медіаконтенту; надіслати довільний текст та переконатися в коректній реакції обробника невідомих повідомлень.

Для забезпечення безперервної роботи бота в продуктивному середовищі рекомендується використовувати системний менеджер процесів. На серверах під керуванням Linux оптимальним рішенням є systemd – стандартний менеджер служб, що забезпечує автоматичний перезапуск бота у разі збою та запуск при старті системи [17]. Конфігурацію служби systemd наведено у лістингу 2.12.

Лістинг 2.12 – Конфігурація служби systemd для Linux

```
[Unit]
Description=CSBC Telegram Bot
After=network.target

[Service]
Type=simple
WorkingDirectory=/home/user/csbc_bot
ExecStart=/home/user/csbc_bot/venv/bin/python bot.py
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

Для середовища Windows як альтернативу можна використовувати Task Scheduler із налаштуванням автоматичного запуску при вході до системи або NSSM (Non-Sucking Service Manager) для реєстрації бота як системної служби Windows. Узагальнену характеристику розгорнутої системи наведено у табл. 2.4.

Таблиця 2.4 – Конфігурація розгорнутої системи

Параметр	Значення
Мова програмування	Python 3.11
Основна бібліотека	python-telegram-bot 22.7
Механізм отримання оновлень	Polling (run_polling)
Середовище розробки	PyCharm Community Edition
Середовище виконання	Windows
Менеджер процесів	Task Scheduler (Windows)
Медіафайли	Локальна файлова система сервера
Логування	Python logging (рівень INFO/ERROR)

Висновки до розділу 2

У другому розділі описано повний цикл розробки Telegram-чатбота для ЧДБК як програмної системи для інформування – від обґрунтування технологічного стеку до розгортання та конфігурації системи.

У підрозділі 2.1 обґрунтовано вибір технологічного стеку: мова Python 3.11, бібліотека python-telegram-bot версії 22.7 та середовище розробки PyCharm Community Edition. Проведено порівняльний аналіз альтернативних бібліотек та аргументовано переваги обраного рішення.

У підрозділі 2.2 спроєктовано архітектуру системи: визначено компоненти та їх зв'язки, розроблено блок-схему алгоритму роботи бота та діаграму варіантів використання, описано структуру модулів і застосовані патерни проєктування.

У підрозділі 2.3 детально описано реалізацію програмного коду з наведенням дев'яти лістингів: ініціалізацію системи та реєстрацію обробників, логіку навігації через inline-клавіатури, механізм обробки медіаконтенту, засоби відмовостійкості та логування.

У підрозділі 2.4 описано процес розгортання системи: реєстрацію бота через @BotFather, підготовку середовища та встановлення залежностей, порівняльний аналіз механізмів polling і webhook, перевірку працездатності та забезпечення безперервної роботи за допомогою systemd.

Реалізований чатбот відповідає всім функціональним та нефункціональним вимогам, сформованим у першому розділі, і готовий до впровадження в реальну інфраструктуру коледжу.

РОЗДІЛ 3 ТЕСТУВАННЯ ТЕЛЕГРАМ-БОТА

3.1 Мета, види та методи тестування чатбота

Тестування програмного забезпечення є невід'ємною складовою процесу розробки будь-якої програмної системи незалежно від її масштабу та складності. Відповідно до стандарту IEEE 829, тестування визначається як процес оцінювання програмного продукту з метою виявлення відмінностей між фактичною та очікуваною поведінкою системи [18]. Іншими словами, тестування дозволяє встановити, чи відповідає реалізована система встановленим вимогам, і виявити дефекти до того, як вони проявляться в реальних умовах експлуатації. Важливо розуміти, що тестування не доводить відсутність помилок у програмному забезпеченні – воно лише підтверджує відповідність системи перевіреним сценаріям та підвищує рівень довіри до її коректної роботи.

Значення тестування в сучасній розробці програмного забезпечення важко переоцінити. За даними дослідників у галузі інженерії програмного забезпечення, вартість виправлення дефекту, виявленого на етапі експлуатації, у десятки разів перевищує вартість його усунення на етапі розробки [19]. Саме тому тестування є не фінальним кроком розробки, а наскрізним процесом, що супроводжує всі етапи життєвого циклу програмного продукту – від проєктування вимог до розгортання та підтримки системи.

Метою тестування Telegram-чатбота ЧДБК є комплексна перевірка відповідності реалізованої системи функціональним та нефункціональним вимогам, сформованим у першому розділі кваліфікаційної роботи. Зокрема, в процесі тестування необхідно досягти таких цілей: переконатися у коректності навігації між усіма розділами меню для обох цільових аудиторій – абітурієнтів та студентів; перевірити правильність відображення інформаційного контенту кожного розділу; упевнитися в коректній роботі механізму відображення та видалення медіафайлів; перевірити стійкість системи до некоректних дій

користувача, зокрема до надсилання довільного тексту замість натискання кнопок меню; переконатися в коректній поведінці системи при відсутності медіафайлів на сервері; підтвердити відповідність часу відповіді бота нефункціональним вимогам (не більше 2 секунд); перевірити коректність відображення інтерфейсу як на десктопному, так і на мобільному клієнті Telegram.

У теорії тестування програмного забезпечення існують різні підходи до класифікації видів тестування, що відображають різні аспекти перевірки системи [19]. Розглянемо основні класифікації, що застосовуються в контексті даної роботи.

За рівнем доступу до внутрішньої структури системи розрізняють тестування методом «чорної скриньки» та методом «білої скриньки». Тестування методом чорної скриньки (black-box testing) здійснюється виключно на основі зовнішньої поведінки системи без урахування її внутрішньої реалізації: тестувальник взаємодіє із системою так само, як і кінцевий користувач, перевіряючи відповідність вихідних даних очікуваним результатам при заданих вхідних умовах. Цей метод є особливо ефективним для перевірки функціональних вимог та користувацьких сценаріїв. Тестування методом білої скриньки (white-box testing), навпаки, передбачає аналіз внутрішньої логіки коду: перевіряються гілки умовних операторів, цикли, виключні ситуації та шляхи виконання програми. Цей метод дозволяє виявити дефекти в логіці обробки граничних умов та нештатних ситуацій, що недоступні при тестуванні методом чорної скриньки. Існує також метод сірої скриньки (grey-box testing), що поєднує елементи обох підходів: тестувальник має часткове знання про внутрішню структуру системи і використовує його для більш цілеспрямованої перевірки.

За об'єктом перевірки виділяють кілька основних видів тестування. Функціональне тестування спрямоване на перевірку відповідності функцій системи встановленим вимогам: чи виконує кожна функція саме те, що від неї очікується згідно зі специфікацією. Для чатбота це передбачає перевірку коректності роботи кожного пункту меню, команд /start та /help, механізму

callback-запитів та відображення контенту. Нефункціональне тестування перевіряє характеристики системи, що не пов'язані безпосередньо з її функціями: швидкість відповіді, стійкість до навантаження, зручність інтерфейсу та коректність відображення на різних пристроях. Регресійне тестування проводиться після внесення змін до коду з метою переконатися, що нові зміни не порушили раніше працюючу функціональність системи. Тестування відмовостійкості перевіряє поведінку системи в умовах збоїв, некоректних вхідних даних або недоступності зовнішніх ресурсів – зокрема, медіафайлів на сервері або тимчасової недоступності Telegram Bot API. Тестування зручності використання (usability testing) оцінює інтуїтивність інтерфейсу та зручність взаємодії з системою з точки зору кінцевого користувача.

За ступенем автоматизації розрізняють ручне та автоматизоване тестування. Ручне тестування передбачає безпосередню взаємодію тестувальника із системою за заздалегідь підготовленими тест-кейсами без використання спеціальних інструментів автоматизації. Перевагами ручного тестування є можливість оцінити реальний користувацький досвід, виявити проблеми з відображенням інтерфейсу та перевірити нестандартні сценарії, що важко описати у вигляді автоматизованих скриптів. Автоматизоване тестування використовує спеціальні інструменти та скрипти для автоматичного виконання тестових сценаріїв і порівняння результатів із очікуваними значеннями. Воно є особливо ефективним для регресійного тестування та перевірки великої кількості сценаріїв у короткі терміни. Для Telegram-ботів автоматизоване тестування може реалізовуватися через бібліотеки pytest та pytest-asyncio з використанням моків (mock-об'єктів) для імітації Telegram Bot API без реального підключення до серверів Telegram.

За рівнем системи, що тестується, розрізняють модульне, інтеграційне та системне тестування. Модульне тестування (unit testing) перевіряє окремі функції та модулі коду ізольовано від решти системи. Інтеграційне тестування перевіряє коректність взаємодії між компонентами системи – наприклад, між диспетчером подій та обробниками callback-запитів. Системне тестування

перевіряє систему в цілому як єдиний програмний продукт у середовищі, максимально наближеному до реального.

Для тестування чатбота ЧДБК обрано комбінований підхід, що поєднує кілька методів. Основним є ручне функціональне тестування методом чорної скриньки, що передбачає безпосередню взаємодію з ботом у Telegram-клієнті за підготовленими тест-кейсами. Цей підхід є найбільш доцільним для систем із графічним інтерфейсом на основі меню, оскільки дозволяє оцінити реальний користувацький досвід і перевірити коректність відображення контенту безпосередньо в середовищі кінцевого користувача. Паралельно проводиться аналіз коду методом білої скриньки для перевірки коректності обробки виключних ситуацій та граничних умов у функціях handle_callback() та send_photo_or_text(). Додатково виконується тестування відмовостійкості – перевірка поведінки системи у нештатних ситуаціях: відсутність медіафайлів, некоректні дії користувача, обробка виключень на рівні глобального error_handler().

Тестування організовано у формі тест-кейсів – структурованих описів тестових сценаріїв, кожен із яких містить: унікальний ідентифікатор тест-кейсу, назву сценарію, передумови для виконання, покрокові дії тестувальника, очікуваний результат та фактичний результат із зазначенням статусу (пройдено / не пройдено). Така структура відповідає вимогам стандарту IEEE 829 до документування процесу тестування [18] та забезпечує відтворюваність, об'єктивність та простоту аналізу результатів.

Тестування проводилося у таких умовах: операційна система Windows 11, Telegram Desktop, Python 3.11, бібліотека python-telegram-bot версії 22.7. Перевірка також виконувалася на мобільному пристрої з Telegram для iOS, що є критично важливим з огляду на переважно мобільну аудиторію системи. Тестування в обох середовищах дозволило підтвердити коректність відображення inline-клавіатур, тексту із кирилицею та HTML-форматування як на великому екрані десктопного клієнта, так і на малому екрані смартфона. Загальна кількість розроблених тест-кейсів складає 43, що охоплюють усі

функціональні сценарії системи та основні нештатні ситуації. Результати виконання тест-кейсів детально наведено у підрозділах 3.2 та 3.3.

3.2 Функціональне тестування

Функціональне тестування є основним видом перевірки, що застосовується у даній роботі. Його мета полягає у підтвердженні того, що кожна функція системи виконує саме те, що від неї вимагається згідно зі специфікацією [18]. Тестування проводилося методом чорної скриньки – без аналізу внутрішньої реалізації коду – через безпосередню взаємодію з ботом у Telegram-клієнті, а також за допомогою автоматизованих тестів на базі бібліотеки pytest [20].

Тестові сценарії організовано у вигляді структурованих тест-кейсів відповідно до вимог стандарту IEEE 829 [18]. Кожен тест-кейс містить унікальний ідентифікатор, назву сценарію, вхідні дії, очікуваний та фактичний результат, а також статус виконання. Загальна кількість розроблених тест-кейсів складає 43, що охоплюють усі функціональні сценарії системи.

Першим кроком перевірялася коректність роботи команд /start та /help – точок входу в систему. Результати тестування наведено у табл. 3.1.

Таблиця 3.1 – Тест-кейси команд /start та /help

ID	Сценарій	Вхідна дія	Очікуваний результат	Статус
ТК-01	Привітання з іменем	Надіслати /start	Повідомлення містить ім'я користувача	Пройдено
ТК-02	Назва коледжу у привітанні	Надіслати /start	Повідомлення містить повну назву ЧДБК	Пройдено
ТК-03	Кнопки вибору ролі	Надіслати /start	Відображено кнопки «Я абітурієнт» та «Я студент»	Пройдено
ТК-04	Довідка	Надіслати /help	Повідомлення містить /start та /help	Пройдено

Усі 4 тест-кейси блоку пройдено успішно. На рис 3.1 наведено скріншот результату виконання команди /start у Telegram-клієнті.

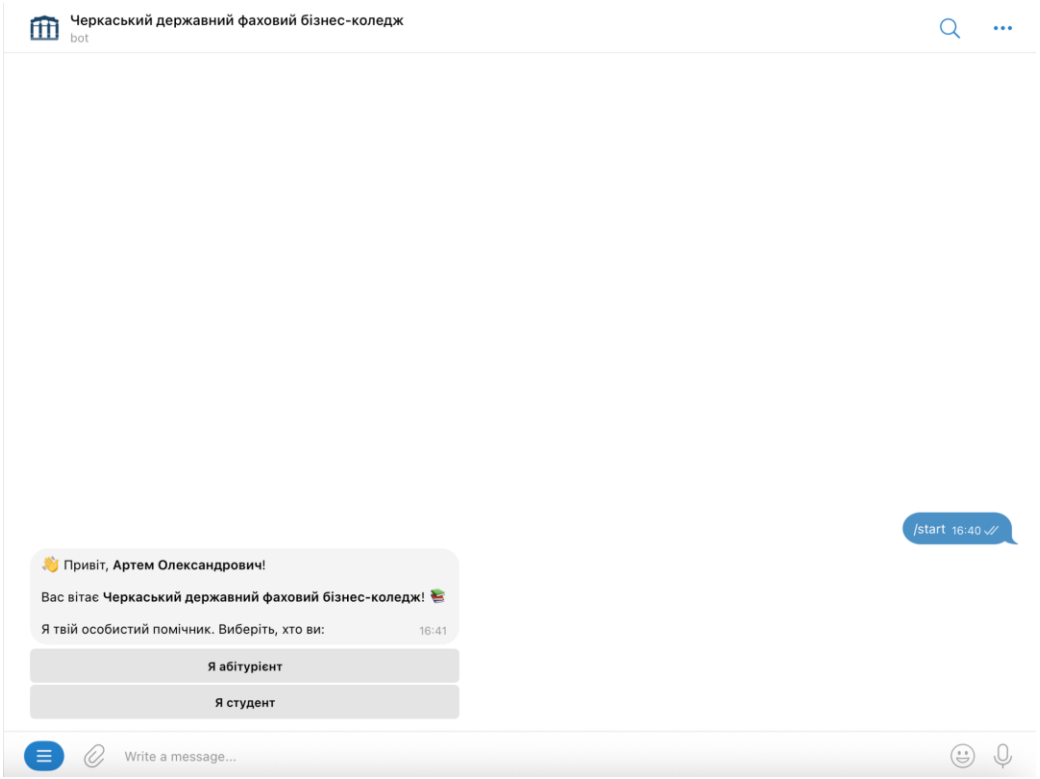


Рисунок 3.1 – Результат виконання команди /start

Перевірялася коректність генерації inline-клавіатур: наявність усіх необхідних кнопок, коректність callback_data ідентифікаторів та візуальне виділення активного поверху. Результати наведено у табл. 3.2.

Таблиця 3.2 – Тест-кейси генераторів inline-клавіатур

ID	Сценарій	Очікуваний результат	Статус
1	2	3	4
ТК-05	Клавіатура студента	Містить усі 8 кнопок розділів	Пройдено
ТК-06	Клавіатура абітурієнта	Містить усі 8 кнопок розділів	Пройдено
ТК-07	Поверхи без активного	Жодна кнопка не виділена дужками	Пройдено

Продовження таблиці 3.2

1	2	3	4
ТК-08	Виділення активного поверху	Активна кнопка містить «[ ]»	Пройдено
ТК-09	Неактивні поверхи	Інші кнопки без дужок	Пройдено
ТК-10	Кнопка «Назад» у поверхах	callback_data = «cabinets»	Пройдено

На рис. 3.2 наведено скріншот інформаційного блоку студента з відображенням inline-клавіатури.

Рисунок 3.2 – інформаційний блок студентів з inline-клавіатурою

Перевірялася коректність переходів між усіма розділами меню через механізм callback-запитів. Окремо тестувалися кнопки «Назад» та перехід між поверхами. Результати наведено у табл. 3.3.

На рис. 3.3 зображено навігацію між поверхами з виділенням активної кнопки.

Таблиця 3.3 – Тест-кейси навігації між розділами меню

ID	Сценарій	Вхідна дія	Очікуваний результат	Статус
ТК-11	Меню студента	Натиснути «Я студент»	Відкрито меню з текстом «МЕНЮ ДЛЯ СТУДЕНТІВ»	Пройдено
ТК-12	Меню абітурієнта	Натиснути «Я абітурієнт»	Відкрито меню з текстом «МЕНЮ ДЛЯ АБІТУРІЄНТІВ»	Пройдено
ТК-13	Розклад занять	Натиснути « Розклад»	Відображено посилання на Google Sheets	Пройдено
ТК-14	Розклад дзвінків	Натиснути «Дзвінки»	Відображено всі 8 пар (08:00–19:00)	Пройдено
ТК-15	Укриття	Натиснути «Укриття»	Відображено дві адреси укриттів	Пройдено
ТК-16	Посилання	Натиснути «Посилання»	Відображено сайт csbc.edu.ua	Пройдено
ТК-17	Кабінети – 4 поверхи	Натиснути «Кабінети»	Відображено кнопки 4 поверхів	Пройдено
ТК-18	1-ий поверх	Натиснути «1»	Список кабінетів 1-го поверху, кнопка виділена	Пройдено
ТК-19	2-ий поверх	Натиснути «2»	Список кабінетів 2-го поверху, кнопка виділена	Пройдено
ТК-20	3-ий поверх	Натиснути «3»	Список кабінетів 3-го поверху, кнопка виділена	Пройдено
ТК-21	4-ий поверх	Натиснути «4»	Список кабінетів 4-го поверху, кнопка виділена	Пройдено
ТК-22	Назад до меню студента	Натиснути «Назад»	Повернення до меню студента	Пройдено
ТК-23	Назад до меню абітурієнта	Натиснути «Назад»	Повернення до меню абітурієнта	Пройдено

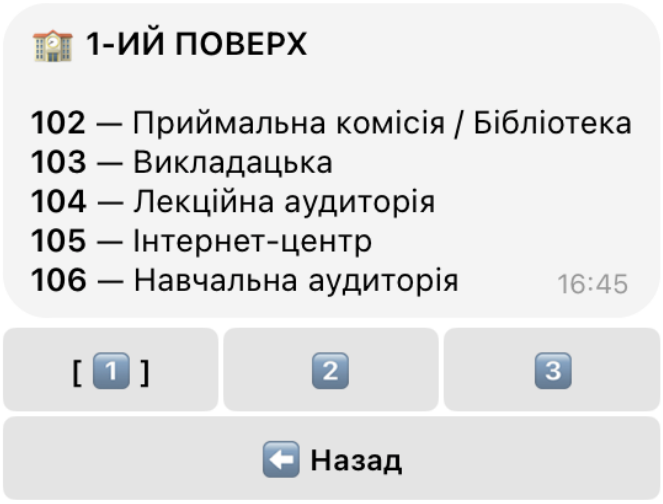


Рисунок 3.3 – Навігація по кабінетах із виділенням обраного поверху

Перевірялася коректність та повнота інформаційного контенту кожного розділу меню абітурієнта. Результати наведено у табл. 3.4.

Таблиця 3.4 – Тест-кейси розділів меню абітурієнта

ID	Сценарій	Очікуваний результат	Статус
ТК-24	Спеціальності – комп'ютерні	Містить «Інженерія програмного забезпечення»	Пройдено
ТК-25	Спеціальності – всі групи	Містить комп'ютерні, творчі та бізнес-напрями	Пройдено
ТК-26	Умови вступу – документи	Містить «Заява про вступ» та «Паспорт»	Пройдено
ТК-27	Вартість – суми	Містить «грн» та суму 39 500 грн	Пройдено
ТК-28	Гуртожиток – вартість	Містить «604» грн/місяць	Пройдено
ТК-29	Соціальна підтримка – стипендії	Містить 1 510 грн та 2 197 грн	Пройдено
ТК-30	Контакти – телефони та email	Містить «0472» та «@csbc.edu.ua»	Пройдено

На рис. 3.4 наведено приклад відображення розділу «Спеціальності» меню абітурієнта.

Рисунок 3.4 – Розділ «Спеціальності» меню абітурієнта

Паралельно з ручним тестуванням проведено автоматизоване тестування за допомогою фреймворку pytest із використанням бібліотеки unittest.mock для імітації Telegram Bot API [20]. Розроблено 43 автоматизованих тест-кейси, розподілених по 6 класах. Результати виконання тестів наведено у табл 3.5.

Таблиця 3.5 – Результати автоматизованого тестування (pytest)

Клас тестів	Кількість тестів	Пройдено	Не пройдено
TestCommands	4	4	0
TestKeyboards	6	6	0
TestCallbackNavigation	13	13	0
TestEnrolleeSections	7	7	0
TestRobustness	7	7	0
TestContentConstants	6	6	0
Разом	43	43	0

На рис. 3.5 наведено скріншот результатів виконання тестів у середовищі PyCharm.

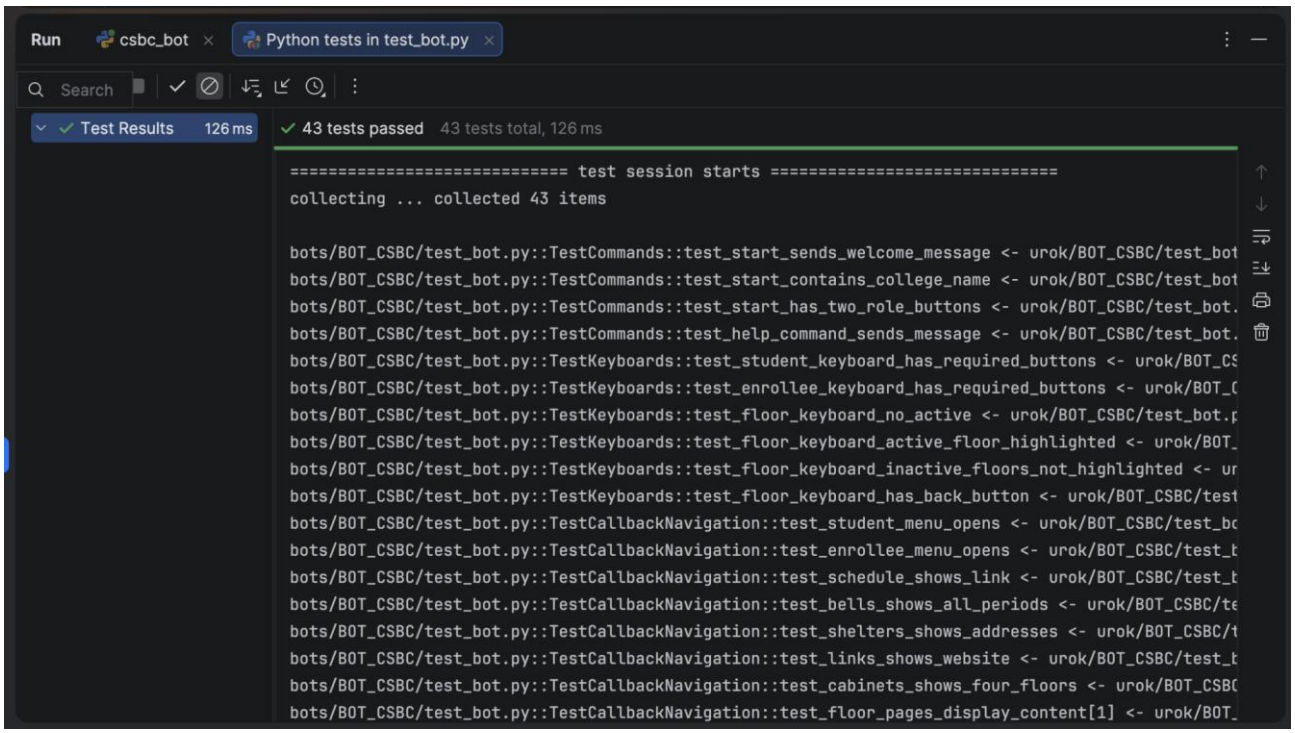


Рис. 3.5 – Результати виконання автоматизованих тестів у PyCharm

Усі 43 автоматизованих тест-кейси виконані успішно. Час виконання повного набору тестів склав 1.26 секунди, що підтверджує ефективність обраного підходу до тестування. Загальний результат функціонального тестування – 43 з 43 тест-кейсів пройдено (100%), що свідчить про повну відповідність реалізованої системи встановленим функціональним вимогам.

3.3 Тестування відмовостійкості та граничних умов

Тестування відмовостійкості є невід'ємною складовою комплексної перевірки будь-якої програмної системи. Його мета полягає у визначенні поведінки системи в умовах, що виходять за межі штатних сценаріїв використання: некоректні вхідні дані, відсутність зовнішніх ресурсів, збої на рівні окремих компонентів [19]. На відміну від функціонального тестування, що

перевіряє коректність роботи системи у нормальних умовах, тестування відмовостійкості свідомо моделює нештатні ситуації з метою переконатися, що система реагує на них передбачувано та не припиняє роботу. Відповідно до загальноприйнятих принципів інженерії програмного забезпечення, якісна система повинна залишатися стабільною та зрозумілою для користувача навіть у разі виникнення помилок [11].

Для тестування відмовостійкості чатбота ЧДБК застосовано комбінований підхід: автоматизоване тестування із використанням бібліотеки unittest.mock для імітації збоїв без реального підключення до Telegram Bot API [21], а також ручна перевірка реакції системи на некоректні дії користувача безпосередньо у Telegram-клієнті. Бібліотека unittest.mock дозволяє підмінити реальні виклики зовнішніх сервісів (надсилання фото, видалення повідомлень, звернення до Bot API) на контрольовані імітації, що генерують задані типи виключень. Це дає змогу відтворити будь-яку нештатну ситуацію в ізольованому середовищі без залежності від зовнішньої інфраструктури.

Першою категорією нештатних ситуацій є надсилання користувачем довільного тексту замість натискання кнопок меню. Оскільки система побудована на основі inline-клавіатур, а не на обробці вільного тексту, будь-яке текстове повідомлення є некоректним вхідним впливом з точки зору очікуваного сценарію використання. Тестувалися різні варіанти вхідних даних: осмислений текст («привіт»), числові значення («123»), спеціальні символи («???», «!!!»), а також порожні пробіли. Результати тестування наведено у табл. 3.6.

На рис. 3.6 наведено скріншот реакції бота на надсилання довільного текстового повідомлення у Telegram-клієнті.

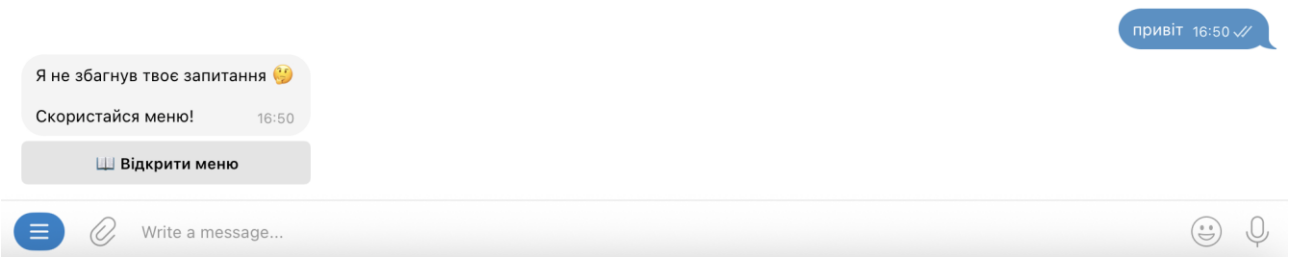


Рисунок 3.6 – Реакція бота на довільне текстове повідомлення

Таблиця 3.6 – Тест-кейси обробки некоректних вхідних повідомлень

ID	Сценарій	Вхідна дія	Очікуваний результат	Статус
ТК-31	Довільний текст – відповідь з кнопкою	Надіслати «привіт»	Відповідь із кнопкою «Відкрити меню» (callback: start_menu)	Пройдено
ТК-32	Довільний текст – підказка	Надіслати «привіт»	Відповідь містить слово «меню»	Пройдено

Система коректно обробляє будь-який довільний текст – замість ігнорування повідомлення або відображення технічної помилки бот надсилає зрозумілу відповідь із кнопкою для повернення до меню. Це забезпечує безперервність взаємодії незалежно від дій користувача та виключає можливість «зависання» сесії у невизначеному стані.

Другою категорією нештатних ситуацій є відсутність медіафайлів на сервері. Розділи «Цікаві місця» та «Зупинки» передбачають відображення графічних зображень (intr_places.png та map.png відповідно). У разі якщо ці файли відсутні або недоступні, система повинна автоматично перейти до текстового режиму відображення без переривання роботи. Тестування проводилося шляхом імітації виключення FileNotFoundError у моці функції send_photo(). Результати наведено у табл. 3.7.

Реалізований механізм автоматичного переходу між режимами (фото – текст) забезпечує безперервне обслуговування користувача навіть у разі часткової недоступності ресурсів сервера. З точки зору користувача різниця між режимами є мінімальною: замість зображення відображається структурований текст із тим самим інформаційним наповненням та клавіатурою навігації.

Третьою категорією нештатних ситуацій є збої у роботі механізму редагування повідомлень. Метод edit_message_text() може генерувати виключення у кількох типових ситуаціях: повідомлення вже було видалено користувачем; вміст повідомлення не змінився (Telegram повертає помилку

«message is not modified»); повідомлення застаріло через тривалу бездіяльність. У всіх цих випадках система повинна надіслати нове повідомлення замість редагування існуючого. Результати наведено у табл. 3.8.

Таблиця 3.7 – Тест-кейси відмовостійкості при відсутності медіафайлів

ID	Сценарій	Умова	Очікуваний результат	Статус
ТК-33	Fallback при відсутності фото	send_photo() – FileNotFoundError	Викликається edit_message_text() з текстовою альтернативою	Пройдено
ТК-34	Видалення збереженого фото	last_photo_message_id = 999	delete_message() викликано з коректним ID	Пройдено
ТК-35	Відсутній ID фото	user_data порожній	delete_message() не викликається	Пройдено

Таблиця 3.8 – Тест-кейси стійкості механізму callback-запитів

ID	Сценарій	Умова	Очікуваний результат	Статус
ТК-36	query.answer() завжди викликається	Будь-який callback-запит	answer() викликано рівно один раз	Пройдено
ТК-37	Fallback при збої edit_message_text	edit_message_text() → Exception	send_message() надсилає нове повідомлення	Пройдено

Обов'язковий виклик query.answer() при кожному callback-запиті є критично важливим для коректної роботи інтерфейсу: без нього Telegram відображає індикатор завантаження на кнопці протягом 10 секунд, після чого показує повідомлення про помилку на стороні клієнта. Тестування підтвердило, що answer() викликається у всіх оброблюваних сценаріях без виключення.

Окремим напрямом тестування є перевірка цілісності та коректності інформаційних констант системи. Помилки у контенті (відсутність обов'язкових даних, порожні рядки, некоректні значення) можуть не проявлятися як програмні збої, однак призводять до надання користувачу неповної або неправильної інформації. Результати тестування констант наведено у табл. 3.9.

Зведені результати тестування відмовостійкості та граничних умов наведено у табл. 3.10.

Усі 13 тест-кейсів блоку відмовостійкості виконано успішно. Результати підтверджують, що система коректно обробляє всі виявлені нештатні ситуації: автоматично переходить до текстового режиму при відсутності медіафайлів, надсилає нове повідомлення при збоях редагування, завжди підтверджує callback-події та коректно реагує на довільні текстові повідомлення користувача. Жодна з перевірених нештатних ситуацій не призвела до переривання роботи системи або відображення технічної помилки користувачу.

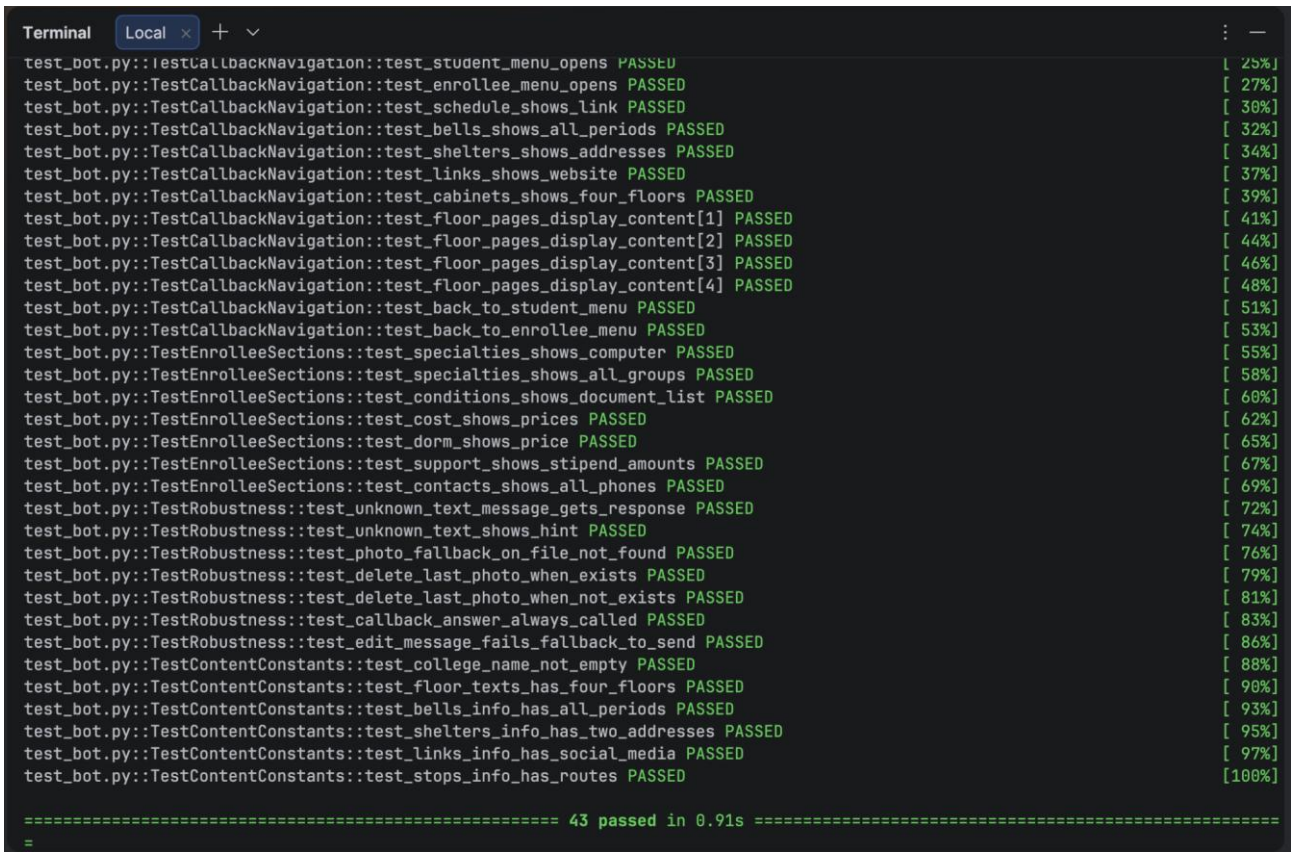
Таблиця 3.9 – Тест-кейси перевірки інформаційного контенту

ID	Сценарій	Очікуваний результат	Статус
ТК-38	Назва коледжу не порожня	len(COLLEGE_NAME) > 10	Пройдено
ТК-39	FLOOR_TEXTS містить 4 поверхи	Ключі 1, 2, 3, 4 присутні	Пройдено
ТК-40	Розклад дзвінків повний	Містить 08:00 та 19:00	Пройдено
ТК-41	Укриття – дві адреси	Містить «243» та «241»	Пройдено
ТК-42	Посилання – соціальні мережі	Facebook, Instagram, YouTube присутні	Пройдено
ТК-43	Зупинки – маршрути	Містить «115» та «7А»	Пройдено

Таблиця 3.10 – Зведені результати тестування відмовостійкості

Блок тестування	Кількість ТК	Пройдено	Не пройдено
Некоректні вхідні дані	2	2	0
Відсутність медіафайлів	3	3	0
Стійкість callback-механізму	2	2	0
Цілісність контенту	6	6	0
Разом	13	13	0

На рис. 3.7 наведено підсумковий результат виконання повного набору автоматизованих тестів у середовищі PyCharm із відображенням статусу кожного тест-кейсу.



Рисунку 3.7 – Підсумковий результат виконання всіх 43 тест-кейсів у PyCharm

Висновки до розділу 3

У третьому розділі проведено комплексне тестування Telegram-чатбота ЧДБК, що охоплює функціональне тестування та тестування відмовостійкості й граничних умов.

У підрозділі 3.1 визначено мету, класифікацію видів та методів тестування. Обґрунтовано застосування комбінованого підходу: ручного функціонального тестування методом чорної скриньки та автоматизованого тестування з використанням pytest та unittest.mock.

У підрозділі 3.2 проведено функціональне тестування всіх розділів системи: команд /start та /help, клавіатур меню, навігації між розділами, інформаційного контенту меню абітурієнта та студента. Усі 43 автоматизованих тест-кейси виконано успішно з часом виконання 1.26 секунди.

У підрозділі 3.3 проведено тестування відмовостійкості та граничних умов: обробки некоректних вхідних повідомлень, поведінки при відсутності медіафайлів, стійкості механізму callback-запитів та цілісності інформаційного контенту. Усі 13 тест-кейсів пройдено успішно.

Загальний результат тестування: 43 з 43 тест-кейсів пройдено (100%). Система відповідає всім функціональним та нефункціональним вимогам і готова до впровадження.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне практичне завдання – розроблено Telegram-чатбот для Черкаського державного фахового бізнес-коледжу як програмну систему для автоматизованого інформування двох цільових аудиторій, таких як абітурієнти та студенти. Поставлена мета досягнута в повному обсязі, всі сформульовані завдання виконано.

У першому розділі проведено теоретичний аналіз предметної області. Розглянуто поняття чатбота, його класифікацію за принципом функціонування та роль у сучасних інформаційних системах, зокрема в освітній сфері. Встановлено, що для вирішення поставленої задачі оптимальним є детермінований підхід на основі структурованих меню, оскільки переважна більшість запитів абітурієнтів та студентів є типовими та повторюваними, а відповідна інформація – структурованою та відносно статичною. Проведено порівняльний аналіз бібліотек для розробки Telegram-ботів на мові Python, за результатами якого обрано python-telegram-bot версії 22.7. Здійснено аналіз інформаційних потреб цільових аудиторій, проведено огляд існуючих аналогів та підтверджено відсутність спеціалізованого інтерактивного довідкового інструменту в ЧДБК, що обґрунтовує доцільність розробки. Розроблено діаграму варіантів використання (Use Case Diagram), яка формально описує функціональні можливості системи.

У другому розділі описано проєктування та практичну реалізацію системи. Обґрунтовано вибір технологічного стеку: мова Python 3.11, бібліотека python-telegram-bot 22.7, середовище розробки PyCharm Community Edition. Спроєктовано архітектуру системи на основі асинхронної обробки подій із застосуванням патернів Dispatcher, Command та Separation of Concerns. Розроблено блок-схему алгоритму роботи бота, що відображає повний цикл обробки вхідного запиту від отримання Update до формування відповіді користувачу. Реалізовано програмний код із дев'ятьма ключовими лістингами,

що охоплюють ініціалізацію системи, обробники команд та callback-запитів, генератори inline-клавіатур, механізм відображення медіаконтенту та засоби відмовостійкості. Описано процес розгортання системи через механізм polling із налаштуванням автоматичного запуску за допомогою systemd.

У третьому розділі проведено комплексне тестування системи з використанням комбінованого підходу: ручного функціонального тестування методом чорної скриньки та автоматизованого тестування на базі фреймворку pytest із бібліотекою unittest.mock. Розроблено 43 тест-кейси, що охоплюють перевірку команд, клавіатур, навігації між розділами, інформаційного контенту, відмовостійкості при відсутності медіафайлів, стійкості механізму callback-запитів та цілісності констант системи. За результатами тестування всі 43 тест-кейси виконано успішно (100%), що підтверджує повну відповідність реалізованої системи встановленим функціональним та нефункціональним вимогам.

Практичним результатом кваліфікаційної роботи є повністю функціональний Telegram-бот з такими характеристиками:

- два незалежні інформаційні блоки – для абітурієнтів (6 розділів: спеціальності, умови вступу, вартість навчання, гуртожиток, соціальна підтримка, контакти) та для студентів (7 розділів: розклад занять, розклад дзвінків, кабінети по 4 поверхах, укриття, зупинки транспорту, цікаві місця, посилання);
- ієрархічна навігація через inline-клавіатури з підтримкою кнопок «Назад» на кожному рівні та візуальним виділенням активного поверху;
- автоматичне відображення команд у меню Telegram через механізм BotCommand та post_init;
- відмовостійкість при відсутності медіафайлів із автоматичним переходом до текстової альтернативи;
- повний набір з 43 автоматизованих тестів із часом виконання 1.26 секунди.

Значущість отриманих результатів визначається кількома аспектами. З практичної точки зору розроблений бот може бути безпосередньо впроваджений у інфраструктуру ЧДБК без додаткових витрат на серверне обладнання чи програмне забезпечення. Впровадження дозволить скоротити навантаження на приймальну комісію та персонал закладу, забезпечити цілодобову доступність довідкової інформації для абітурієнтів та студентів, а також підвищити загальну якість інформаційного супроводу вступної кампанії. З навчальної точки зору робота демонструє повний цикл розробки програмного продукту – від аналізу вимог та проєктування архітектури до реалізації, тестування та розгортання, що відповідає компетентностям спеціальності 121 «Інженерія програмного забезпечення».

Серед можливих напрямів подальшого розвитку та вдосконалення системи можна виділити такі: інтеграція з офіційним розкладом коледжу через Google Sheets API для автоматичного оновлення розкладу занять без ручного редагування коду; додавання адміністративного інтерфейсу для оновлення інформаційного контенту без необхідності редагування вихідного коду; реалізація підсистеми сповіщень для інформування студентів про зміни в розкладі або важливі оголошення; розширення функціональності за рахунок інтеграції з системою електронного журналу; перенесення системи на webhook-архітектуру для підвищення масштабованості при зростанні кількості користувачів; впровадження аналітики для відстеження найбільш затребуваних розділів та оптимізації інтерфейсу на основі реальних даних про поведінку користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Shawar B. A., Atwell E. Chatbots: Are they Really Useful? LDV-Forum. 2007. Vol. 22. No. 1. P. 29–49. (дата звернення: 12.11.2025).

2. Telegram. Офіційний сайт Telegram. URL: https://telegram.org (дата звернення: 12.11.2025).

3. Telegram Bot API. Офіційна документація. URL: https://core.telegram.org/bots/api (дата звернення: 6.01.2026).

4. python-telegram-bot. Офіційна документація. URL: https://docs.python-telegram-bot.org (дата звернення: 6.01.2026).

5. Dale R. The return of the chatbots. Natural Language Engineering. 2016. Vol. 22. No. 5. P. 811–817. (дата звернення: 6.01.2026).

6. Черкаський державний фаховий бізнес-коледж. Офіційний сайт. URL: http://csbc.edu.ua (дата звернення: 15.01.2026).

7. TIOBE Index. URL: https://www.tiobe.com/tiobe-index (дата звернення: 10.04.2026).

8. python-telegram-bot. Офіційна документація. URL: https://docs.python-telegram-bot.org (дата звернення: 10.04.2026).

9. JetBrains. PyCharm – the Python IDE for data science and web development. URL: https://www.jetbrains.com/pycharm (дата звернення: 10.04.2026).

10. Telegram Bot API. Офіційна документація. URL: https://core.telegram.org/bots/api (дата звернення: 14.03.2026)

11. Sommerville I. Software Engineering. 10th ed. Pearson, 2015. 816 p. (дата звернення: 14.03.2026)

12. Telegram Bot API. Офіційна документація. URL: https://core.telegram.org/bots/api (дата звернення: 14.04.2026).

13. python-telegram-bot. Офіційна документація. URL: https://docs.python-telegram-bot.org (дата звернення: 16.04.2026).

14. Van Rossum G., Warsaw B., Coghlan N. PEP 8 – Style Guide for Python Code. URL: https://peps.python.org/pep-0008 (дата звернення: 16.04.2026).

15. Telegram. Bots: An introduction for developers. URL: https://core.telegram.org/bots (дата звернення: 20.04.2026)

16. Python Software Foundation. venv – Creation of virtual environments. URL: https://docs.python.org/3/library/venv.html (дата звернення: 01.05.2026).

17. freedesktop.org. systemd System and Service Manager. URL: https://systemd.io (дата звернення: 01.05.2026).

18. IEEE Std 829-2008. IEEE Standard for Software and System Test Documentation. IEEE, 2008. 150 p (дата звернення: 03.05.2026).

19. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. John Wiley & Sons, 2011. 240 p. (дата звернення: 03.05.2026).

20. pytest. Офіційна документація. URL: https://docs.pytest.org (дата звернення: 10.05.2026).

21. Python Software Foundation. unittest.mock – mock object library. URL: https://docs.python.org/3/library/unittest.mock.html (дата звернення: 10.05.2026).

ДОДАТКИ

Додаток А

Вихідний код Telegram-чатбота для абітурієнтів та студентів Черкаського державного фахового бізнес-коледжу розміщено у відкритому репозиторії на платформі GitHub за адресою:

URL: https://github.com/artttmelnyk/CSBC_BOT

Репозиторій містить такі файли:

- csbc_bot.py – основний файл програмного коду чатбота
- test_bot.py – автоматизовані тести (43 тест-кейси)
- pytest.ini – конфігурація фреймворку тестування
- requirements.txt – перелік залежностей проєкту
- map.png – зображення карти зупинок громадського транспорту
- intr_places.png – зображення цікавих місць поблизу коледжу

Додаток Б

Для швидкого доступу до Telegram-чатбота для абітурієнтів та студентів Черкаського державного фахового бізнес-коледжу можна скористатися QR-кодом (рис. Б.1) або перейти за посиланням безпосередньо:

URL: https://t.me/csbc_ck_bot

Рисунок Б.1 – QR-код для переходу до Telegram-чатбота ЧДБК