Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютерна система анкетування абітурієнтів з використанням сервера на базі Raspberry Pi*

Виконав: студент 4 курсу, групи СІ-42
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

		Юрков М. С.
	(підпис)	(прізвище та ініціали)
Керівник		Жаровський Р.О.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Тиш Є.В.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		Осухівська Г.М.
	(підпис)	(прізвище та ініціали)
Рецензент		Дмитроца Л.П.
	(підпис)	(прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
«25» квітня 2026 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студента *Юркова Миколи Сергійовича*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютерна система анкетування абітурієнтів з використанням сервера на базі Raspberry Pi*

Керівник роботи *кандидат технічних наук, доцент кафедри КС Жаровський Руслан Олегович*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від *« 24 » квітня 2026 року № 4/9-188*
2. Термін подання студентом завершеної роботи *20.06.2026*
3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
1. Аналіз технічного завдання
2. Проєктна частина
3. Практична частина
4. Безпека життєдіяльності, основи охорони праці.
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

1. *Загальна структура комп'ютерної системи анкетування*
2. *Структура комп'ютерної системи анкетування абітурієнтів*
3. *Структурна схема програмних компонентів системи*
4. *Діаграма прецедентів комп'ютерної системи анкетування*
5. *ER-діаграма бази даних*
6. *Блок-схема роботи*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання *25.04.2026 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Розробка технічного завдання*	*26.01 – 02.02*	
2.	*Робота над першим розділом «Аналіз технічного завдання»*	*03.02 – 15.02*	
3.	*Робота над другим розділом «Проєктна частинв»*	*20.04 – 25.04*	
4.	*Робота над третім розділом «Практична частина»*	*26.04 – 05.05*	
5.	*Робота над четвертим розділом «Безпека життєдіяльності, основи охорони праці»*	*07.05 – 25.05*	
6.	*Оформлення пояснювальної записки і графічного матеріалу*	*26.05 – 07.06*	
7.	*Перевірка на академічний плагіат, перевірка керівником та консультантами*	*08.06 – 14.06*	
8.	*Попередній захист кваліфікаційної роботи бакалавра*	*15.06 – 21.06*	
9.	*Захист кваліфікаційної роботи бакалавра*	*27.06*	

Студент _____ *Юрков Микола Сергійович*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Жаровський Руслан Олегович*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Юрков М. С. Комп’ютерна система анкетування абітурієнтів з використанням сервера на базі Raspberry Pi: робота на здобуття кваліфікаційного ступеня бакалавра: спец. 123 – комп’ютерна інженерія. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2026.

Ключові слова: Raspberry Pi, Telegram-бот, анкетування, абітурієнт, SQLite, Google Sheets, Python, aiogram, systemd, watchdog.

У кваліфікаційній роботі розроблено комп’ютерну систему анкетування абітурієнтів із використанням сервера на базі Raspberry Pi. Система призначена для автоматизованого збору контактних та освітніх даних вступників через Telegram-бота, збереження результатів у локальній базі даних SQLite та дублювання інформації в Google Sheets.

У роботі проаналізовано предметну область систем анкетування, обґрунтовано вибір апаратних і програмних засобів, розроблено архітектуру системи, алгоритми роботи, структуру бази даних та інформаційні потоки.

Програмну частину реалізовано мовою Python із використанням бібліотеки aiogram. Передбачено адміністративну панель, розсилки, динамічні анкети, розіграші, журналювання, автозапуск через systemd і watchdog-контроль.

Проведене тестування підтвердило працездатність системи та її готовність до практичного використання.

ANNOTATION

Yurkov M. S. Computer System for Applicant Surveying Using a Raspberry Pi-Based Server: Bachelor’s Graduation Thesis: speciality 123 – computer engineering. Ternopil: Ternopil Ivan Puluj National Technical University, 2026.

Keywords: Raspberry Pi, Telegram-bot, questionnaire, applicant, SQLite, Google Sheets, Python, aiogram, systemd, watchdog.

In the qualification work, a computer system for questionnaires for applicants was developed using a server based on Raspberry Pi. The system is designed for automated collection of contact and educational data of applicants via Telegram-bot, saving results in a local SQLite database and duplicating information in Google Sheets.

The work analyzed the subject area of questionnaire systems, justified the choice of hardware and software, developed the system architecture, operating algorithms, database structure and information flows.

The software part was implemented in Python using the aiogram library. An administrative panel, mailings, dynamic questionnaires, draws, logging, autorun via systemd and watchdog control are provided.

The testing confirmed the system's operability and its readiness for practical use.

ЗМІСТ

ВСТУП

У сучасних умовах цифровізації освітніх процесів важливого значення набувають засоби автоматизованого збору, збереження та опрацювання інформації від користувачів. Одним із таких напрямів є використання комп'ютерних систем анкетування, які дають змогу швидко отримувати структуровані дані, зменшувати обсяг ручної роботи, підвищувати оперативність комунікації та спрощувати подальший аналіз отриманих результатів.

Особливо актуальним є застосування таких систем у роботі з абітурієнтами. Під час вступної кампанії, днів відкритих дверей, профорієнтаційних заходів і консультацій виникає потреба швидко збирати контактні дані потенційних вступників, фіксувати їхні освітні інтереси, рік вступу, тип закладу освіти та інші відомості, необхідні для подальшої комунікації. Використання звичайних паперових анкет або розрізнених електронних форм ускладнює облік, пошук, фільтрацію, розсилання повідомлень і контроль стану опрацювання заявок.

Для вирішення цієї задачі доцільним є створення комп'ютерної системи анкетування, яка поєднує зручний для користувача інтерфейс Telegram-бота та серверну частину, розгорнуту на Raspberry Pi. Такий підхід дозволяє реалізувати доступне середовище взаємодії для абітурієнтів, забезпечити локальне збереження даних, інтеграцію з Google Sheets, адміністративне керування, розсилки, журналювання подій і контроль працездатності системи.

Метою кваліфікаційної роботи є розроблення комп'ютерної системи анкетування абітурієнтів із серверною частиною на базі Raspberry Pi, яка забезпечує проходження анкети через Telegram-бота, збереження результатів у локальній базі даних, експорт даних у Google Sheets та адміністрування процесу збору інформації.

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Аналіз предметної області

Системи анкетування та опитування призначені для автоматизованого збирання, структуризації, збереження та подальшої обробки відповідей користувачів. Вони використовуються в освіті, маркетингу, соціологічних дослідженнях, медицині, кадровому менеджменті, державному управлінні, технічній підтримці та інших сферах, де необхідно отримати формалізовану інформацію від групи респондентів.

Системи анкетування можуть реалізовуватися у різному вигляді. Вибір конкретного варіанта залежить від вимог до доступності, автономності, захисту даних, кількості користувачів, способу адміністрування та наявної технічної інфраструктури. Узагальнену роль системи анкетування в процесі збору даних наведено на рис. 1.1.

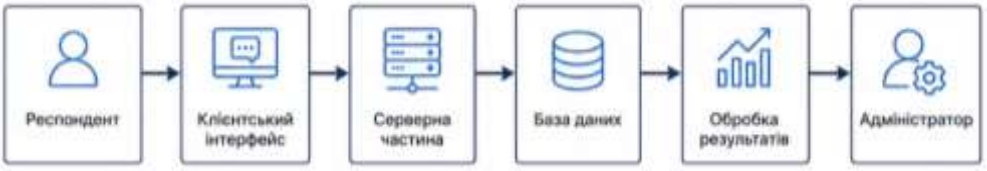

Рисунок 1.1 – Узагальнена роль системи анкетування в процесі збору даних

З рисунка видно, що система анкетування є проміжною ланкою між користувачем, який надає відповіді, та адміністратором, який використовує отримані дані для подальшої роботи. При цьому ключове значення має не лише інтерфейс введення, а й серверна логіка, сховище даних, канали передавання інформації та засоби адміністрування [2].

1.2 Загальна структура комп’ютерних систем анкетування

Більшість сучасних систем анкетування має багаторівневу структуру (рис. 1.2). До її складу входять клієнтський рівень, мережевий рівень, серверний рівень, рівень збереження даних, адміністративний рівень та рівень інтеграції із зовнішніми сервісами. Такий поділ дає змогу окремо розглядати функції введення даних, передавання повідомлень, обробки запитів, збереження інформації та формування результатів.

Рисунок 1.2 – Загальна структура комп’ютерної системи анкетування

Наведена структура є типовою для більшості систем опитування незалежно від конкретної реалізації. Відмінності полягають у виборі клієнтського інтерфейсу, серверної платформи, протоколу передавання даних, типу бази даних та способу адміністрування [3].

1.2.1 Клієнтська частина систем анкетування

Клієнтська частина системи анкетування забезпечує введення даних користувачем, відображення запитань, вибір варіантів відповіді, первинну перевірку введених значень і передавання результатів на сервер. Її реалізація залежить від обраного типу системи.

Це може бути браузер, який виконує HTML-розмітку, CSS-стилі та JavaScript-код. У мобільних системах клієнтська частина реалізується у вигляді Android або iOS застосунку та використання апаратних можливостей смартфона. У чат-ботах клієнтський рівень забезпечується платформою месенджера. Основна логіка при цьому виконується на сервері, а клієнтський пристрій використовується лише як засіб відображення діалогу. У кіоск-системах клієнтська частина реалізується за допомогою сенсорного екрана, повноекранного браузера або спеціалізованої локальної програми. До апаратного складу такої системи входять сенсорний дисплей, мінікомп'ютер, корпус, блок живлення, мережевий інтерфейс і, за потреби, додаткова периферія [4].

1.2.2 Серверна частина систем анкетування

Серверна частина є основним обчислювальним рівнем системи анкетування. Вона приймає запити або повідомлення від клієнтів, виконує логіку анкети, перевіряє введені дані, взаємодіє з базою даних, формує статистику, забезпечує експорт результатів і виконує інтеграцію із зовнішніми сервісами.

У простих системах сервер може бути реалізований як один програмний процес, що приймає дані, обробляє їх і записує у файл або локальну базу. У складніших реалізаціях серверна частина містить вебсервер, API-сервер, сервер застосунків, базу даних, модуль автентифікації, планувальник задач, систему логування та засоби резервного копіювання [5, 6].

Типова серверна частина системи анкетування містить модуль приймання запитів або повідомлень, модуль керування сценарієм анкети, модуль перевірки даних, модуль роботи з базою даних, адміністративний модуль, модуль експорту, модуль інтеграції із зовнішніми сервісами та модуль логування.

Апаратна частина системи залежить від її архітектури. Серверне обладнання може бути реалізоване на фізичному сервері, VPS, хмарній віртуальній машині, локальному ПК або Raspberry Pi.

Мережеве обладнання включає маршрутизатор, комутатор, точку доступу Wi-Fi та мережеві кабелі. Для серверної роботи бажаним є Ethernet-з'єднання, оскільки воно забезпечує вищу стабільність порівняно з Wi-Fi [7].

1.2.3 Бази даних і збереження результатів

Сховище даних використовується для збереження відповідей користувачів, службових ідентифікаторів, часу заповнення анкети, статусу запису та результатів обробки. У невеликих системах можуть використовуватися CSV або JSON-файли, однак для структурованого збереження доцільніше застосовувати бази даних.

SQLite підходить для локальних і малонавантажених систем, оскільки не потребує окремого серверного процесу й зберігає дані у файлі. MySQL, MariaDB або PostgreSQL доцільно використовувати у складніших системах із великою кількістю користувачів, багатокористувацьким доступом і розвиненою структурою даних.

Окремим варіантом є дублювання результатів у Google Таблиці або інші хмарні табличні сервіси. Такий підхід зручний для перегляду, фільтрації та спільної роботи з даними. При цьому локальна база може використовуватися як основне або резервне сховище, а таблиця — як інструмент адміністрування та аналізу.

1.2.4 Мережеві протоколи та способи обміну даними

Системи анкетування працюють як мережеві інформаційні системи, тому важливим є вибір способу обміну даними між клієнтом, сервером і зовнішніми сервісами.

У вебсистемах застосовуються HTTP та HTTPS. HTTP використовується для передавання вебсторінок і форм, а HTTPS забезпечує шифрування даних під час передавання. Для систем, які працюють із персональними даними, використання HTTPS є обов'язковим.

У вебзастосунках і мобільних застосунках часто використовується REST API. Дані між клієнтом і сервером передаються у форматі JSON. Такий підхід дозволяє використовувати одну серверну частину для різних типів клієнтів.

Для систем реального часу може застосовуватися WebSocket, який забезпечує постійне двостороннє з'єднання між клієнтом і сервером. Його доцільно використовувати для панелей моніторингу, онлайн-статистики або динамічного відображення відповідей [8].

У чат-ботах обмін даними відбувається через API месенджера (табл. 1.1).

Таблиця 1.1 – Протоколи та механізми обміну даними в системах анкетування

Протокол або механізм	Призначення	Типове застосування
HTTP	Передавання вебсторінок і форм	Вебанкети, локальні вебсистеми
HTTPS	Захищене передавання даних	Анкети з персональними даними
REST API	Обмін структурованими даними	Вебзастосунки, мобільні застосунки
JSON	Формат подання даних	API-запити, відповіді сервера
WebSocket	Постійне двостороннє з'єднання	Панелі реального часу
Polling	Періодичне отримання оновлень	Telegram-боти без публічного сервера
Webhook	Надсилання подій на сервер	Чат-боти з HTTPS endpoint
SSH	Віддалене адміністрування	Керування сервером Raspberry Pi
SFTP/SCP	Передавання файлів	Оновлення проєкту, резервні копії
SMTP	Передавання електронних повідомлень	Email-сповіщення та розсилки

Для Telegram-ботів використовуються два режими: polling і webhook. У режимі polling сервер періодично звертається до Telegram і отримує нові повідомлення. Такий режим простіший для розгортання на Raspberry Pi, оскільки не потребує відкритого порту, доменного імені та публічного HTTPS-

сервера. У режимі webhook Telegram сам надсилає події на адресу сервера, але для цього потрібен публічно доступний HTTPS endpoint.

Для адміністрування серверів на базі Linux використовуються SSH, SFTP або SCP. SSH забезпечує віддалений доступ до сервера, перегляд логів і керування службами. SFTP та SCP застосовуються для передавання файлів проєкту, конфігурацій і резервних копій.

1.2.5 Захист даних і адміністрування

Системи анкетування часто працюють із персональними даними, тому повинні мати базові механізми захисту. Технічний захист передбачає використання захищених каналів передавання даних, обмеження доступу до адміністративних функцій, зберігання токенів і ключів поза програмним кодом, резервне копіювання та журналювання помилок. Токени доступу, паролі, API-ключі та файли службових облікових записів доцільно зберігати у конфігураційних файлах або змінних середовища.

Адміністративна частина системи повинна забезпечувати перегляд статистики, експорт результатів, отримання повідомлень про нові анкети, керування розсилками та контроль роботи серверної частини. У простих системах адміністрування може виконуватися через службові команди, у складніших — через вебпанель.

Для серверів на базі Raspberry Pi важливими є віддалене адміністрування через SSH, контроль стану служби, перегляд логів, оновлення програмного забезпечення та резервне копіювання бази даних [9].

1.3 Приклади технічної реалізації систем анкетування

Прикладом самостійно розгорнутої вебсистеми є LimeSurvey (рис. 1.3). Це спеціалізована система для створення складних опитувань, яка встановлюється на власний сервер і працює через вебінтерфейс. Технічно вона

потребує вебсервера, PHP-середовища та бази даних. Її перевагою є підтримка різних типів запитань, умовної логіки переходів, керування респондентами та експорту результатів. Недоліком є більша складність розгортання й адміністрування порівняно з простими формами.

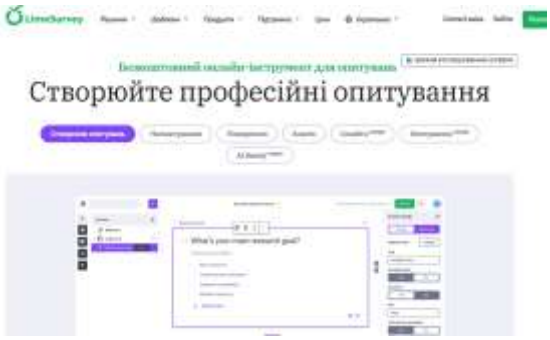

Рисунок 1.3 – Система опитування LimeSurvey

Окрему групу становлять хмарні сервіси форм, зокрема Google Forms і Microsoft Forms (рис. 1.4). У цих системах користувач створює анкету через вебінтерфейс, поширює посилання серед респондентів, а відповіді зберігаються на серверах постачальника сервісу.

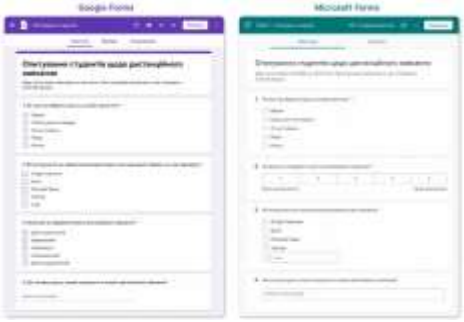

Рисунок 1.4 – Системи опитування Google Forms і Microsoft Forms

У Google Forms результати можуть автоматично передаватися в Google Таблиці, що спрощує подальше сортування, фільтрацію та спільну роботу з даними. Microsoft Forms має подібну архітектуру та інтегрується з Microsoft 365, Excel і Teams. Перевагою таких сервісів є швидке створення форми без

власного сервера, а недоліком — залежність від зовнішньої платформи та обмежений контроль над внутрішньою логікою системи.

Мобільні системи анкетування застосовуються тоді, коли необхідна робота зі смартфонами або планшетами, підтримка офлайн-режиму, локального кешування, push-повідомлень, камери чи геолокації (рис. 1.5).

Рисунок 1.5 – Системи опитування ODK та KoboToolbox

У такій архітектурі клієнтська частина реалізується як Android або iOS застосунок, а серверна частина зазвичай надає API для синхронізації результатів. Прикладами таких рішень є ODK та KoboToolbox. Їхньою перевагою є можливість тимчасової автономної роботи, а недоліком — потреба у встановленні та налаштуванні мобільного застосунку.

Чат-боти реалізують анкетування у вигляді послідовного діалогу в месенджері (рис.1.6).

Рисунок 1.6 – Приклад реалізації анкетування за допомогою Telegram-бота

Модулі анкетування також можуть бути частиною систем дистанційного навчання. Наприклад, Moodle містить засоби тестування, опитування та збору зворотного зв’язку. Клієнтська частина реалізується через вебкабінет користувача, а серверна частина працює на сервері LMS і використовує її базу даних та систему авторизації.

Локальні кіоск-системи є прикладом програмно-апаратної реалізації анкетування. Апаратно така система може складатися із сенсорного дисплея, мінікомп’ютера або одноплатного комп’ютера, корпусу, блока живлення, мережевого інтерфейсу та локального сховища даних. Програмна частина може бути реалізована як повноекранний браузер, локальний вебзастосунок або спеціалізована десктопна програма. Перевагою є контрольоване апаратне середовище, а недоліком — необхідність фізичного обслуговування обладнання.

CRM-рішення використовуються тоді, коли анкетування є частиною ширшого процесу роботи з контактами. У таких системах анкета поєднується з історією комунікації, сегментацією користувачів, плануванням дзвінків, електронними розсилками та аналітикою.

Порівняльну характеристику основних реалізацій систем анкетування наведено в табл. 1.2.

З наведеного аналізу видно, що системи анкетування можуть мати різну програмно-апаратну організацію. Хмарні форми найпростіші у впровадженні, але не дають повного контролю над серверною частиною [10]. Вебсистеми забезпечують більшу гнучкість, проте потребують налаштування серверної інфраструктури. Мобільні системи ефективні для польових умов, але вимагають встановлення застосунку. Кіоск-системи мають чітко визначене апаратне середовище, однак потребують фізичного обслуговування. Чат-боти забезпечують зручну діалогову взаємодію та можуть працювати на компактному сервері.

Таблиця 1.2 – Порівняння варіантів технічної реалізації систем анкетування

Варіант реалізації	Приклад системи	Клієнтська частина	Серверна частина	Особливості застосування
Вебсистема	LimeSurvey	Браузер користувача	Вебсервер, PHP-середовище, база даних	Підходить для складних опитувань, потребує власного серверного розгортання
Хмарна форма	Google Forms, Microsoft Forms	Браузер або мобільний застосунок	Сервери постачальника сервісу	Швидке створення форм, мінімальні вимоги до адміністрування, обмежений контроль над серверною частиною
Мобільна система	ODK, KoboToolbox	Android/iOS застосунок	Сервер синхронізації, база даних	Підтримка офлайн-режиму, доцільна для польового збору даних
Чат-бот	Telegram-бот	Месенджер користувача	Сервер бота, база даних, API месенджера	Діалогова взаємодія, можливість розсилок, придатність для запуску на Raspberry Pi
LMS-модуль	Moodle	Вебкабінет користувача	Сервер LMS, база даних освітньої платформи	Доцільний для студентів і викладачів з наявними обліковими записами
Кіоск-система	Сенсорний термінал реєстрації	Сенсорний екран, локальний інтерфейс	Мінікомп’ютер, локальна або віддалена база даних	Підходить для очних заходів і приймальних комісій
CRM-рішення	CRM-модуль збору контактів	Вебінтерфейс CRM	CRM-сервер, база контактів, модулі аналітики	Доцільне для комплексної роботи з великою кількістю контактів

Для кваліфікаційної роботи доцільною є гібридна архітектура, у якій клієнтський інтерфейс реалізується через поширений засіб комунікації, серверна частина розміщується на Raspberry Pi, дані зберігаються в локальній базі та за потреби передаються у зовнішній табличний сервіс. Такий підхід дозволяє розглядати систему не лише як програмну форму збору відповідей, а як завершений програмно-апаратний комплекс.

1.4 Постановка задач кваліфікаційної роботи

Проведений аналіз існуючих систем анкетування показав, що для задач збору інформації від абітурієнтів доцільним є використання клієнт-серверної

архітектури, яка поєднує доступний користувацький інтерфейс, локальну серверну частину та засоби централізованого збереження й обробки даних. Особливий інтерес становлять рішення на основі чат-ботів, оскільки вони не потребують встановлення додаткового програмного забезпечення, забезпечують зручну взаємодію з користувачем та можуть функціонувати на малопотужних серверних платформах.

Метою кваліфікаційної роботи є розробка комп'ютерної системи анкетування абітурієнтів з використанням сервера на базі Raspberry Pi, яка забезпечує автоматизований збір, збереження та обробку анкетних даних із використанням Telegram-бота як клієнтського інтерфейсу.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– провести аналіз сучасних систем анкетування та визначити особливості їх програмно-апаратної реалізації;

– дослідити архітектури клієнт-серверних систем збору даних та обґрунтувати вибір Telegram-бота як клієнтської частини системи;

– проаналізувати можливості одноплатного комп'ютера Raspberry Pi та обґрунтувати його використання як серверної платформи;

– розробити загальну структуру комп'ютерної системи анкетування абітурієнтів;

– реалізувати серверну частину системи на базі Raspberry Pi з підтримкою автоматичного запуску та безперервної роботи;

– розробити Telegram-бот для проходження анкетування абітурієнтами;

– забезпечити автоматичне збереження результатів анкетування у локальному сховищі та їх передачу до Google Sheets;

– реалізувати адміністративні функції перегляду та експорту отриманих даних;

– провести тестування основних функціональних можливостей розробленої системи та оцінити її працездатність в умовах реальної експлуатації.

РОЗДІЛ 2 ПРОЄКТНА ЧАСТИНА

2.1 Функціональні можливості та архітектура системи

На відміну від звичайної електронної форми, розроблювана система є програмно-апаратним комплексом. Клієнтська взаємодія реалізується через Telegram-бота, а серверна частина розміщується на Raspberry Pi. Такий підхід дає змогу поєднати доступність месенджера для абітурієнтів із можливістю автономного серверного розгортання. Raspberry Pi у системі виконує роль локального сервера, на якому працює програмне забезпечення бота, база даних, модулі адміністрування, розсилок, логування, моніторингу та контролю працездатності.

До основних функціональних можливостей системи належать: покрокове анкетування абітурієнтів у Telegram; збереження заявок у локальній базі даних SQLite; дублювання даних у Google Sheets; повідомлення адміністраторів про нові заявки; перегляд статистики; пошук і фільтрація заявок; зміна статусів; експорт у CSV; створення інформаційних і сегментованих розсилок; автоматичні нагадування; ведення журналу подій; файлове логування; перевірка стану системи; створення додаткових анкет без зміни програмного коду; проведення розіграшів; запуск як системної служби на Raspberry Pi; watchdog-контроль активності сервісу.

Архітектура системи побудована за клієнт-серверним принципом. Клієнтська частина представлена Telegram-клієнтом користувача. Серверна частина реалізована у вигляді Python-застосунку, що працює на Raspberry Pi та взаємодіє з Telegram Bot API. Для отримання повідомлень використовується режим polling, за якого сервер періодично звертається до Telegram API та отримує нові події.

Клієнт-серверну архітектуру проєктованої системи наведено на рис. 2.1.

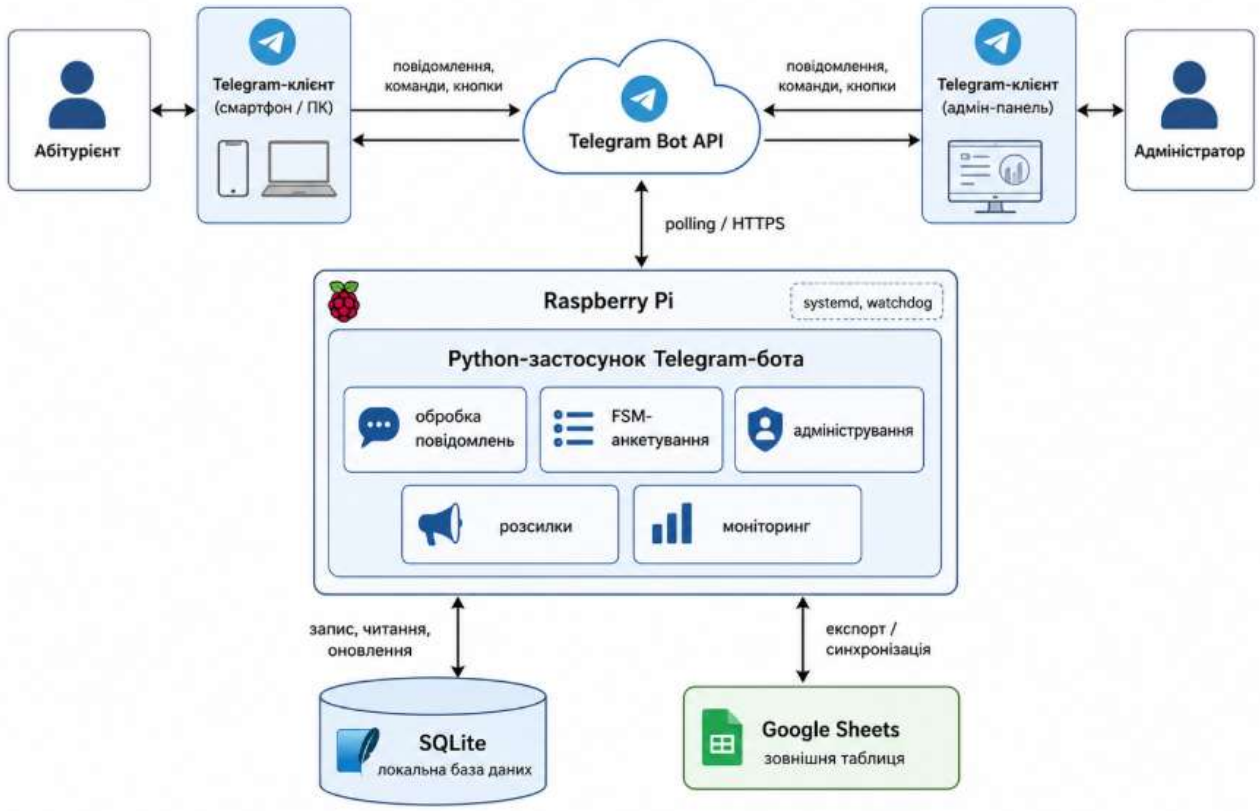

Рисунок 2.1 – Клієнт-серверна архітектура комп’ютерної системи анкетування абітурієнтів

У наведеній архітектурі Telegram виконує роль зовнішньої комунікаційної платформи, Raspberry Pi — роль локального сервера, SQLite — основного сховища даних, а Google Sheets — додаткового інструмента перегляду та спільної роботи з результатами анкетування. Адміністративна взаємодія також реалізується через Telegram, тому система не потребує окремої вебпанелі [11].

Узагальнене призначення основних компонентів системи наведено в табл. 2.1.

Таблиця 2.1 – Призначення основних компонентів системи

Компонент системи	Призначення
Абітурієнт	Користувач, який проходить анкетування та отримує інформаційні повідомлення
Telegram-клієнт	Інтерфейс взаємодії користувача із системою
Telegram Bot API	Платформа обміну повідомленнями між користувачем і серверною частиною
Raspberry Pi	Апаратна серверна платформа для запуску програмного забезпечення
Python-застосунок	Основна серверна логіка анкетування, адміністрування та обробки даних
SQLite	Локальна база даних для збереження заявок, розсилок, подій, анкет і службової інформації
Google Sheets	Зовнішній сервіс для перегляду та спільної роботи з результатами
Адміністратор	Користувач із розширеними правами керування системою
APScheduler	Планувальник задач для запуску розсилок і нагадувань
systemd	Засіб запуску Telegram-бота як системної служби
Watchdog	Механізм контролю активності системного сервісу
Журнал подій і лог-файл	Засоби діагностики та контролю роботи системи

Проєктована система має модульну структуру, що дає змогу розділити функції введення даних, обробки команд, збереження інформації, адміністрування, інтеграції із зовнішніми сервісами та контролю працездатності. Такий підхід спрощує подальше розширення системи, зокрема додавання нових типів анкет, розширення фільтрів або підключення інших зовнішніх сервісів.

2.2 Обґрунтування вибору апаратних і програмних засобів

Як апаратну платформу доцільно використати Raspberry Pi (рис. 2.2). Для задачі анкетування продуктивності Raspberry Pi достатньо, оскільки система не виконує ресурсоємних обчислень, а основне навантаження пов’язане з обробкою повідомлень, записом у базу даних і періодичним зверненням до зовнішніх API [9].

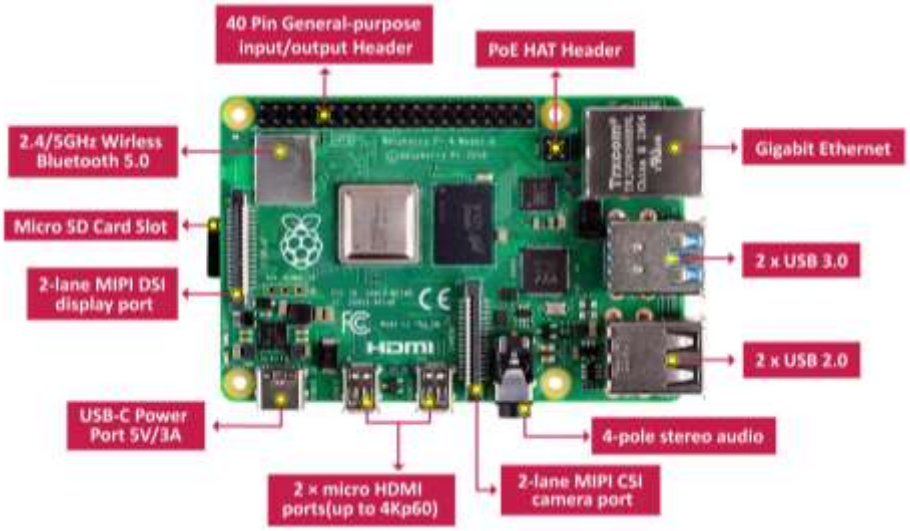

Рисунок 2.2 – Компоненти Raspberry Pi

Використання Raspberry Pi як сервера має такі переваги: компактність, низьке енергоспоживання, можливість цілодобової роботи, підтримка Python, наявність Ethernet/Wi-Fi, доступність системних служб Linux, можливість віддаленого адміністрування через SSH. Крім того, така платформа дає змогу розглядати систему не лише як програмний чат-бот, а як завершений програмно-апаратний комплекс.

Для роботи серверної частини використовується Raspberry Pi OS. Linux-середовище дозволяє встановлення Python, створення віртуального середовища, керування залежностями, запуск застосунку через systemd, перегляд журналів і налаштування watchdog-таймера.

Основною мовою програмування обрано Python 3. Вибір Python обґрунтований наявністю бібліотек для роботи з Telegram Bot API, SQLite, Google Sheets API, планувальниками задач, HTTP-запитами та конфігураційними файлами. Для розробки Telegram-бота використовується бібліотека aiogram 3, яка підтримує асинхронну модель роботи, обробку команд, повідомлень, callback-запитів і FSM-механізм для покрокових сценаріїв анкетування.

Для збереження даних використовується SQLite. Така база даних не потребує окремого серверного процесу, зберігає дані у файлі та добре

підходить для локального розгортання на Raspberry Pi. Для асинхронної взаємодії з базою даних використовується aiosqlite [12].

Для передавання даних у Google Таблиці використовується Google Sheets API через бібліотеку gspread. Google Sheets у системі не є основним сховищем, а виконує роль додаткового сервісу для перегляду, фільтрації та спільної роботи з результатами анкетування. Якщо інтеграція з Google Sheets недоступна, система продовжує працювати з локальною базою SQLite.

Для розсилок і нагадувань використовується APScheduler. Він дає змогу періодично перевіряти чергу повідомлень і запускати надсилання у визначений час. Для перевірки мережевого з'єднання використовується aiohttp, а для зберігання конфігураційних параметрів — python-dotenv.

Перелік основних апаратних і програмних засобів наведено в табл. 2.2.

Таблиця 2.2 – Апаратні та програмні засоби проєктованої системи

Засіб	Призначення в системі
Raspberry Pi	Локальний сервер для запуску Telegram-бота
Raspberry Pi OS	Операційна система серверної платформи
Python 3	Мова реалізації серверної частини
aiogram 3	Бібліотека для роботи з Telegram Bot API
FSM aiogram	Механізм покрокового проходження анкети
SQLite	Локальна база даних
aiosqlite	Асинхронна робота з SQLite
gspread	Взаємодія з Google Sheets
APScheduler	Планування розсилок і нагадувань
aiohttp	Перевірка доступності мережевих ресурсів
python-dotenv	Завантаження параметрів із .env-файлу
systemd	Автозапуск і контроль сервісу бота
watchdog-таймер	Періодична перевірка активності сервісу
SSH	Віддалене адміністрування Raspberry Pi

Обрана комбінація засобів забезпечує достатню функціональність, невисокі вимоги до апаратних ресурсів, простоту розгортання та можливість подальшого розширення системи. Використання режиму polling дозволяє запускати бота на Raspberry Pi без білого IP, доменного імені та налаштування вхідних з'єднань на роутері.

2.3 Розробка структурної схеми системи

Структурна схема системи відображає склад програмних модулів і зв’язки між ними. Центральним елементом є серверний Python-застосунок, який виконується на Raspberry Pi. Він приймає події від Telegram Bot API, обробляє команди користувачів, керує сценаріями анкетування, взаємодіє з базою даних, виконує експорт, розсилки, логування та моніторинг.

До складу системи входять такі основні модулі: модуль Telegram-бота, модуль FSM-анкетування, модуль адміністрування, модуль роботи із заявками, модуль бази даних, модуль Google Sheets [14], модуль розсилок і нагадувань, модуль динамічних анкет, модуль розіграшів, модуль моніторингу, модуль конфігурації та модуль watchdog-контролю. Структурну схему системи наведено на рис. 2.3.

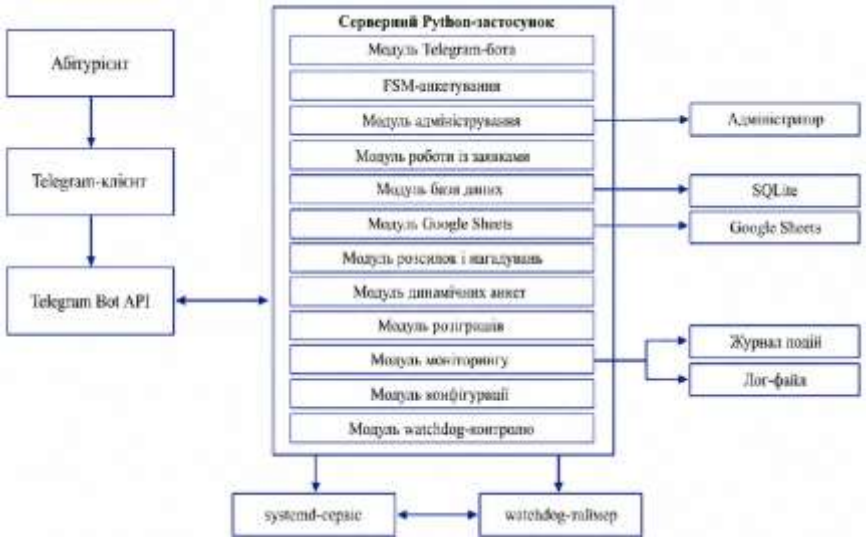

Рисунок 2.3 – Структурна схема програмних компонентів системи

Модуль Telegram-бота відповідає за отримання повідомлень, команд і callback-запитів від користувачів. Модуль FSM-анкетування забезпечує послідовне проходження основної анкети. Модуль адміністрування реалізує службові команди, розмежування прав, роботу зі статусами, заявками та статистикою. Модуль бази даних забезпечує створення таблиць, додавання,

оновлення, пошук і вибірку записів. Модуль Google Sheets виконує передавання результатів у зовнішню таблицю. Модуль розсилок і нагадувань обробляє черги повідомлень і виконує надсилання за розкладом. Модуль моніторингу відповідає за перевірку мережі, ведення логів і повідомлення адміністраторів про критичні події.

Для опису взаємодії користувачів із системою доцільно використати діаграму прецедентів (рис. 2.4). У системі передбачено три ролі: User (абітурієнт), Модератор та Адміністратор.

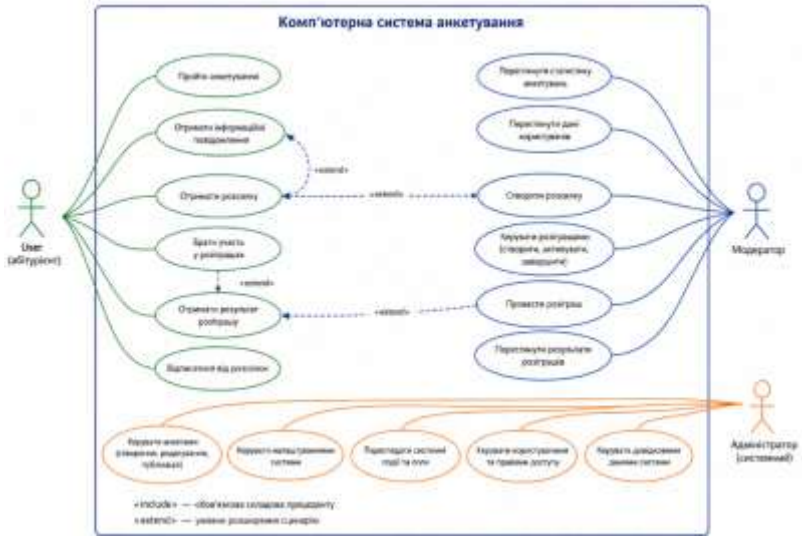

Рисунок 2.4 – Діаграма прецедентів комп’ютерної системи анкетування

Користувач (абітурієнт) є основним джерелом даних для системи. Він може проходити анкетування, отримувати інформаційні повідомлення, підписуватися на розсилки, брати участь у розіграшах, отримувати результати їх проведення та відмовлятися від подальших розсилок. Основний сценарій взаємодії користувача із системою полягає у заповненні анкети та наданні контактних даних для подальшого зв’язку.

Модератор виконує функції оперативного керування процесом роботи системи. До його повноважень належить перегляд статистики анкетувань, перегляд даних користувачів, створення розсилок, організація та проведення розіграшів, а також контроль результатів їх виконання. Саме модератор

формує інформаційні повідомлення та здійснює комунікацію з потенційними абітурієнтами.

Адміністратор відповідає за функціонування програмно-апаратного комплексу в цілому. Його функції включають керування анкетами, налаштування системи, перегляд системних подій і журналів роботи, керування користувачами та правами доступу, а також підтримку довідкових даних. Таким чином адміністратор забезпечує працездатність і безпечне функціонування системи.

На діаграмі також показані зв’язки типу include та extend. Зв’язок include використовується для відображення обов’язкових складових сценарію. Наприклад, процес проходження анкетування включає введення контактних даних користувача. Зв’язок extend застосовується для опису умовних або додаткових сценаріїв. Так, отримання розсилки можливе лише після створення відповідної розсилки модератором, а отримання результатів розіграшу є наслідком виконання прецеденту проведення розіграшу. Аналогічно інформаційні повідомлення надсилаються користувачам лише після їх формування та ініціювання модератором.

2.4 Розробка алгоритму роботи системи

Алгоритм роботи системи охоплює запуск бота, взаємодію користувача з Telegram-інтерфейсом, проходження анкети, перевірку введених даних, збереження результатів, експорт до Google Sheets і повідомлення адміністратора. Окремі алгоритмічні гілки передбачені для адміністративних команд, розсилок, динамічних анкет і моніторингу працездатності.

Основний алгоритм роботи системи наведено на рис. 2.5.

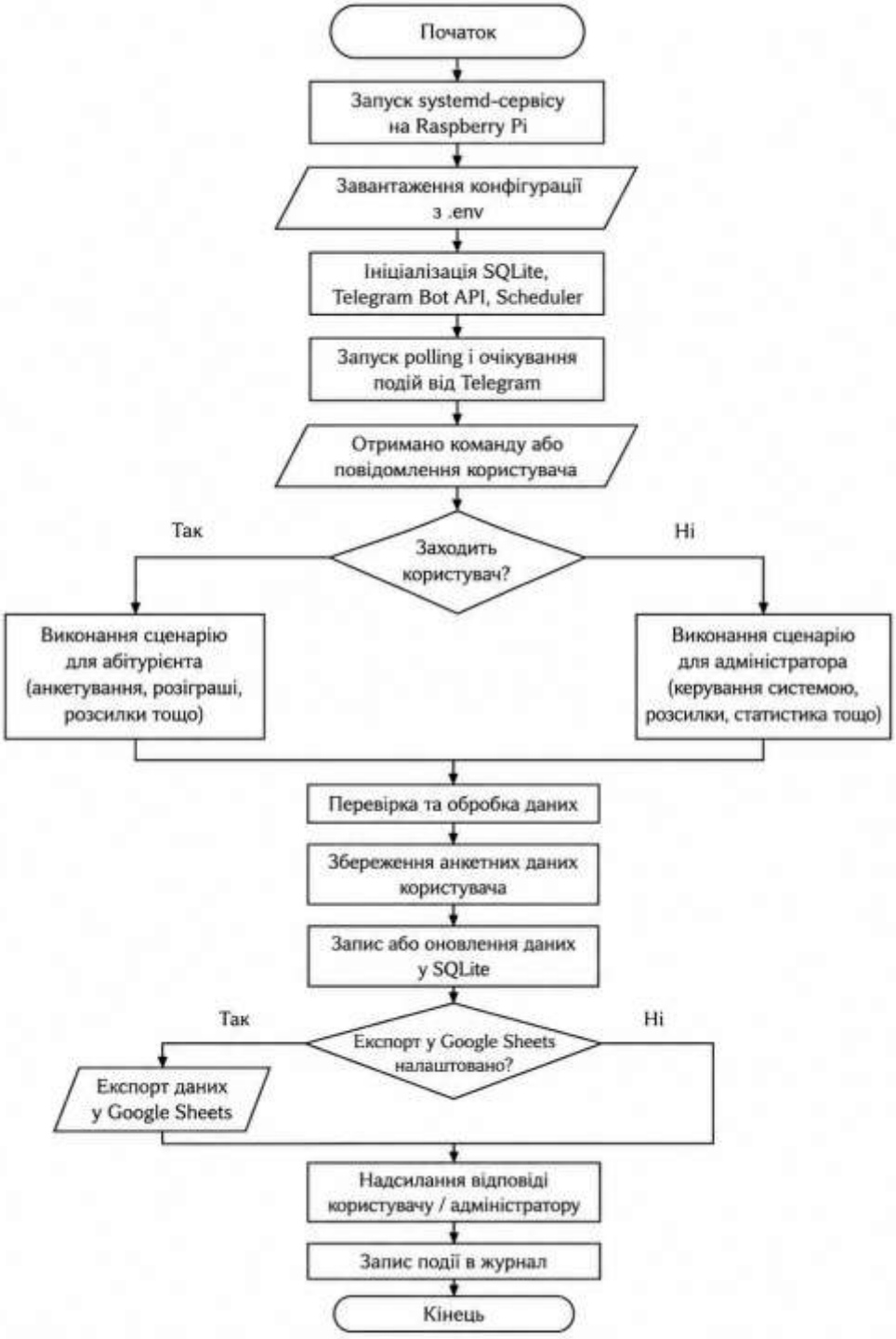

Рисунок 2.5 – Узагальнений алгоритм роботи системи

Під час проходження анкети використовується FSM-модель. Кожне поле анкети відповідає окремому стану. Перехід до наступного стану виконується лише після отримання та перевірки поточної відповіді. Такий підхід дозволяє уникнути пропуску обов’язкових полів і спростити логіку взаємодії з користувачем. Алгоритм проходження анкети наведено на рис. 2.6.

Рисунок 2.6 – Алгоритм проходження анкети абітурієнтом

Окрім алгоритму, доцільно описати послідовність взаємодії компонентів системи. Діаграма послідовності для процесу подання заявки наведена на рис. 2.7.

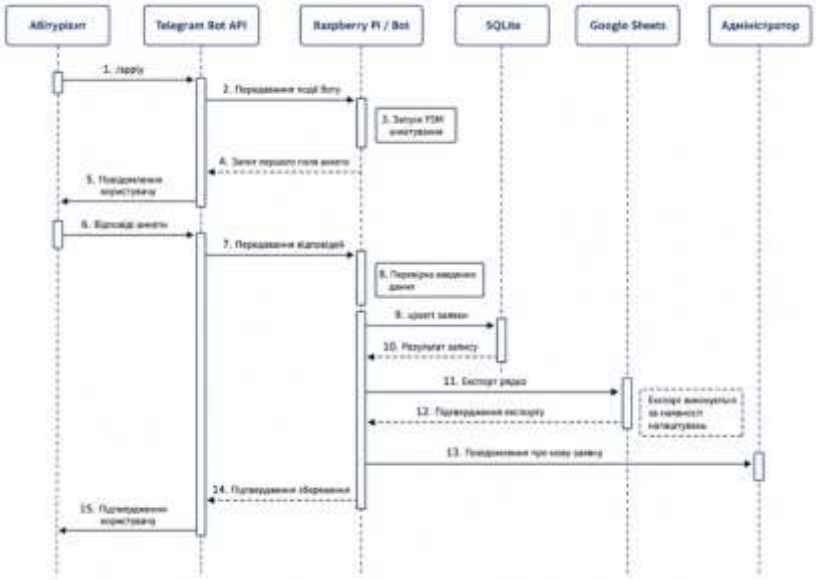

Рисунок 2.7 – Діаграма послідовності процесу подання заявки абітурієнтом

Діаграма послідовності для роботи модератора із заявками наведена на рис. 2.8.

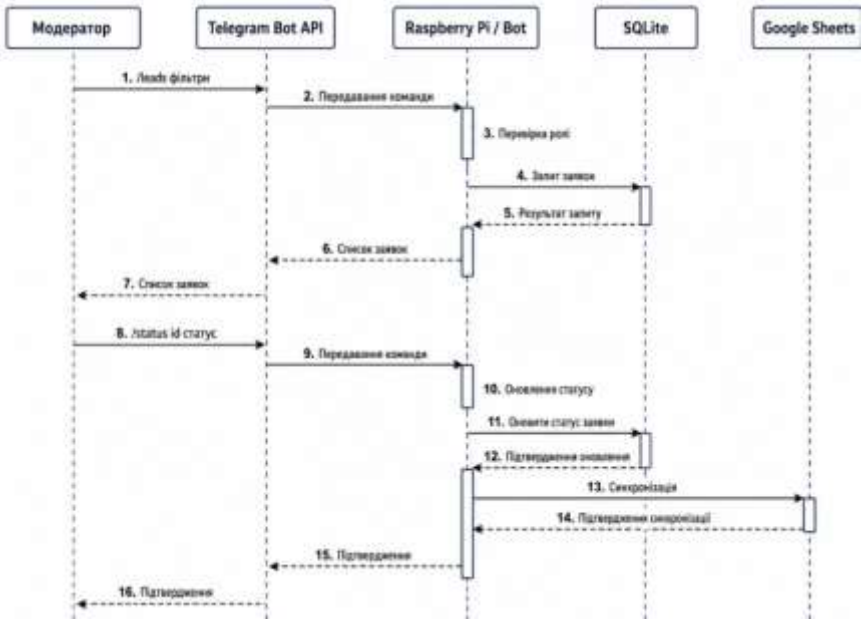

Рисунок 2.8 – Діаграма послідовності роботи модератора із заявками

Діаграма послідовності для сегментованої розсилки наведена на рис. 2.9.

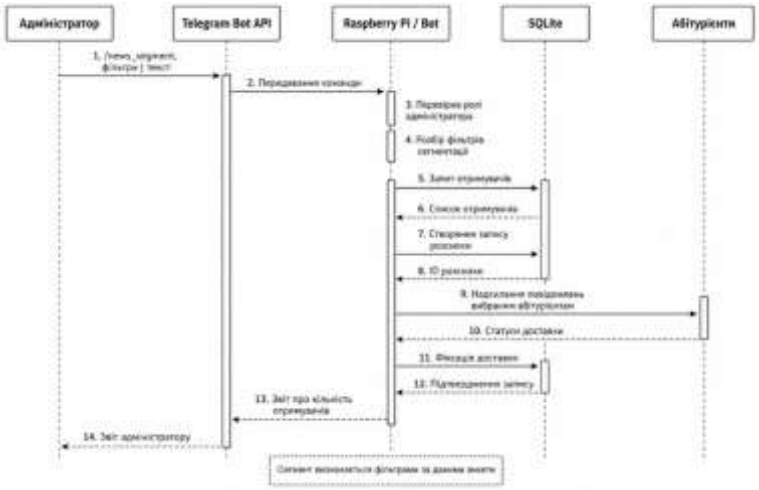

Рисунок 2.9 – Діаграма послідовності виконання сегментованої розсилки

Для контролю працездатності системи передбачено окремий алгоритм моніторингу. Він включає перевірку доступності мережі, запис подій у лог і

сповіщення адміністратора у випадку зміни стану мережевого підключення. Зовнішній watchdog-таймер періодично перевіряє активність systemd-сервісу [15].

Діаграму послідовності контролю працездатності наведено на рис. 2.10.

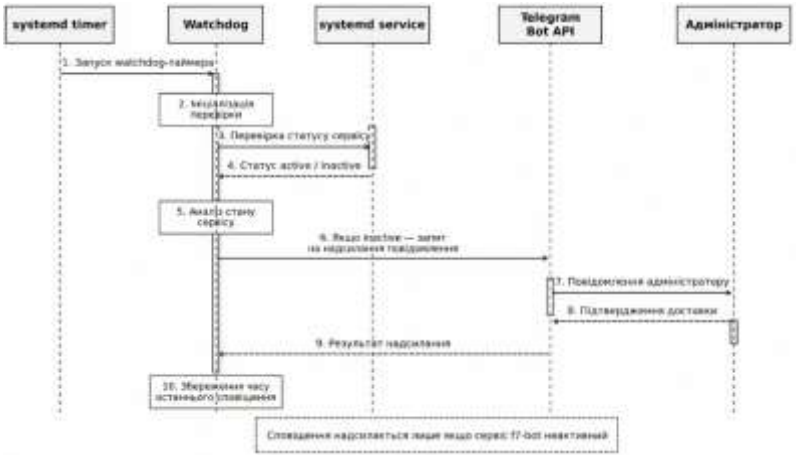

Рисунок 2.10 – Діаграма послідовності watchdog-контролю системного сервісу

Таким чином, алгоритмічна частина системи охоплює як користувацькі сценарії, так і службові процеси, необхідні для стабільної роботи на сервері Raspberry Pi.

2.5 Проєктування структури бази даних

Для збереження даних комп’ютерної системи анкетування використовується база даних SQLite. Такий вибір є доцільним для розгортання системи на Raspberry Pi, оскільки SQLite не потребує окремого серверного процесу, має невеликі вимоги до ресурсів і зберігає всі дані в одному локальному файлі. Вона зберігає основні заявки абітурієнтів, інформаційні розсилки, автоматичні нагадування, системні події, результати розіграшів.

Центральною таблицею є leads, у якій зберігаються основні заявки абітурієнтів. Первинним ключем цієї таблиці є telegram_id, що дозволяє ідентифікувати користувача Telegram і уникати дублювання записів. Якщо

абітурієнт повторно проходить анкетування, система може оновити його попередні дані.

Таблиця scheduled_messages використовуються для організації інформаційних повідомлень і відкладених нагадувань. Для сегментації отримувачів у них передбачено збереження фільтрів, що дає змогу надсилати повідомлення не всім користувачам, а певній групі абітурієнтів.

Таблиця event_logs призначена для фіксації службових подій, помилок і дій користувачів. Вона використовується для діагностики роботи системи, контролю адміністративних операцій і аналізу можливих збоїв.

Для підтримки динамічних анкет використовуються таблиці survey_forms, survey_questions, survey_submissions і survey_answers. Така структура дозволяє створювати додаткові анкети без зміни програмного коду, задавати питання різних типів і зберігати відповіді користувачів у структурованому вигляді.

Таблиця giveaways використовується для збереження інформації про розіграші серед абітурієнтів, зокрема назви призу, умов участі, переможця та адміністратора, який виконав розіграш.

Структуру бази даних наведено на рис. 2.11. Основні таблиці бази даних наведено в табл. 2.3.

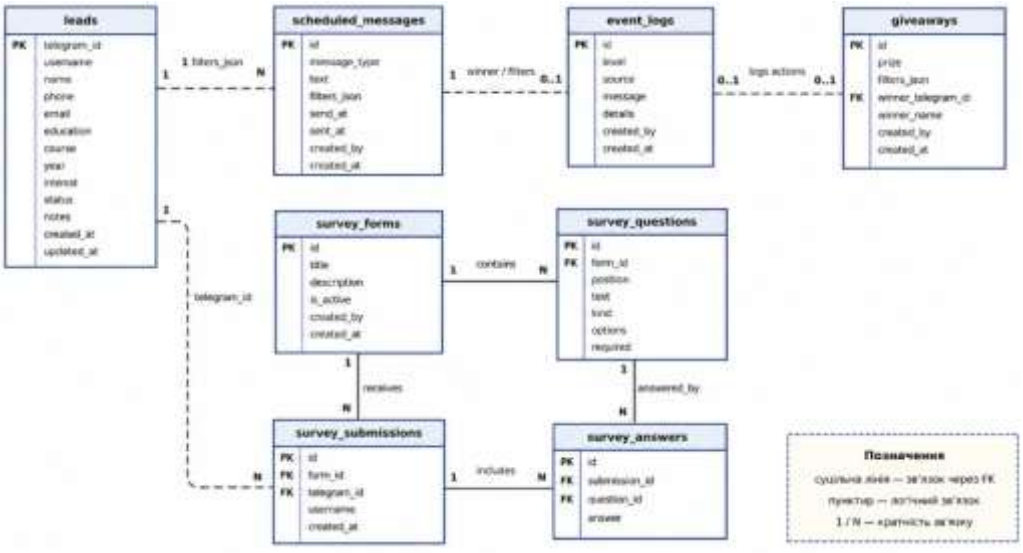

Рисунок 2.11 – ER-діаграма бази даних

Таблиця 2.3 – Основні таблиці бази даних системи

Таблиця	Призначення
leads	Збереження основних заявок абітурієнтів
scheduled_messages	Збереження інформаційних розсилок і запланованих повідомлень автоматичних нагадувань
event_logs	Журнал системних подій, помилок і дій користувачів
giveaways	Збереження результатів розіграшів
survey_forms	Збереження додаткових динамічних анкет
survey_questions	Збереження питань динамічних анкет
survey_submissions	Фіксація факту проходження анкети користувачем
survey_answers	Збереження відповідей на питання динамічних анкет

Основні зв’язки між таблицями формуються навколо стандартної анкети, динамічних анкет і службових функцій системи. Таблиця leads використовується як основне джерело даних про абітурієнтів. Таблиці newsletters, reminders і giveaways логічно пов’язані з користувачами через Telegram ID та фільтри сегментації. Для динамічних анкет одна форма може містити багато питань, одна форма може мати багато проходжень, а одне проходження може містити багато відповідей.

2.6 Проєктування інформаційних потоків

Інформаційні потоки системи визначають порядок передавання даних між абітурієнтом, Telegram, сервером Raspberry Pi, базою даних, Google Sheets та адміністратором. Система має кілька основних напрямів передавання інформації: користувацький потік анкетування, адміністративний потік керування заявками, потік експорту даних, потік розсилок і службовий потік моніторингу (рис. 2.12).

Основний інформаційний потік починається з повідомлення користувача в Telegram. Telegram передає подію через Bot API, після чого серверна частина на Raspberry Pi обробляє її відповідно до поточного стану FSM. Якщо користувач проходить анкету, введені дані тимчасово

зберігаються в контексті стану, а після завершення записуються до SQLite. Далі система може передати дані в Google Sheets і повідомити адміністратора.

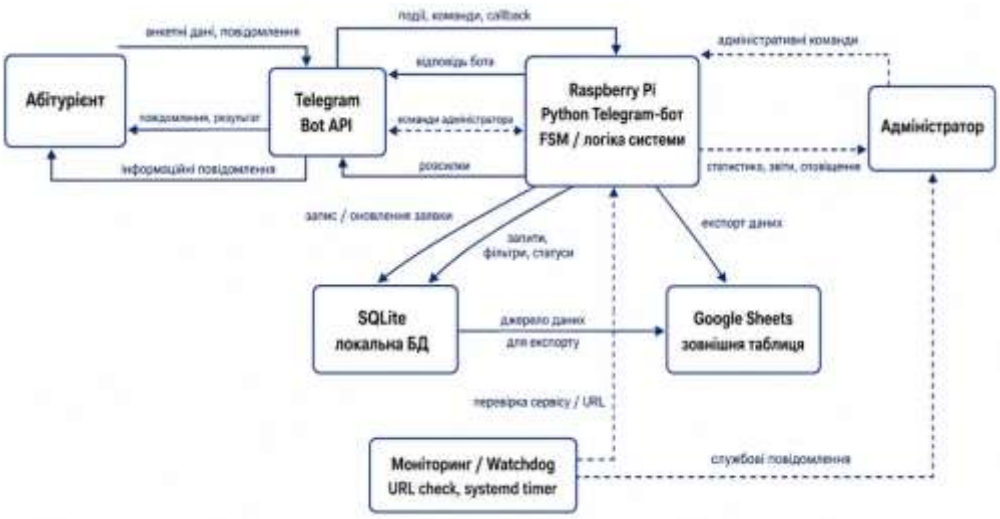

Рисунок 2.12 – Схема інформаційних потоків системи

Під час адміністративної роботи інформаційний потік має зворотний характер. Адміністратор надсилає команду боту, сервер перевіряє роль користувача, виконує запит до бази даних, формує відповідь і надсилає її через Telegram. Якщо команда пов’язана з експортом або розсилкою, додатково виконується взаємодія з Google Sheets або надсилання повідомлень вибраним користувачам.

Потік експорту до Google Sheets використовується для дублювання актуального стану заявок у зовнішню таблицю. При цьому SQLite залишається основним джерелом даних. Це важливо, оскільки зовнішній сервіс може бути тимчасово недоступним, а локальна база забезпечує автономність роботи системи.

Потік моніторингу не пов’язаний безпосередньо з анкетуванням, але є важливим для стабільної роботи. Фоновий монітор перевіряє доступність контрольного URL, а watchdog-таймер перевіряє активність systemd-сервісу. У разі виявлення проблеми адміністратор отримує службове повідомлення [16].

2.7 Захист даних, адміністрування та контроль роботи системи

Оскільки система анкетування працює з персональними даними абітурієнтів, під час її проєктування передбачено базові механізми захисту, розмежування доступу та контролю роботи серверної частини. Захист реалізується на рівні конфігурації, адміністративних прав, локального збереження даних, журналювання подій і контролю працездатності системи.

Конфіденційні параметри системи не зберігаються безпосередньо в програмному коді. До таких параметрів належать токен Telegram-бота, ідентифікатори адміністраторів, шлях до файлу сервісного акаунта Google та параметри доступу до Google Sheets. Для їх зберігання використовується файл .env, що дозволяє відокремити службові ключі від основного програмного коду.

Розподіл доступу до адміністративних функцій виконується за Telegram ID користувача та його роллю. У системі передбачено дві основні ролі: Administrator і moderator. Адміністратор має доступ до системних налаштувань, журналів, анкет, моніторингу та керування системою загалом. Модератор працює із заявками абітурієнтів, переглядає статистику, виконує фільтрацію даних, створює розсилки, проводить розіграші та переглядає результати анкетування.

Для контролю стану заявок використовується система статусів new, viewed, contacted, interested, inactive. Вона дає змогу фіксувати етап опрацювання звернення абітурієнта та впорядковувати подальшу роботу з користувачами.

Адміністрування виконується через Telegram-команди та кнопкове меню бота, тому окрема вебпанель не потрібна. Контроль роботи системи забезпечується журналом подій, файловим логуванням, службовою командою /health, моніторингом мережевого з’єднання та watchdog-контролем системної служби. Якщо основний сервіс бота перестає працювати, watchdog-таймер фіксує проблему і надсилає повідомлення адміністратору.

Основні засоби захисту та контролю роботи системи наведено в табл. 2.4.

Таблиця 2.4 – Засоби захисту та контролю роботи системи

Напрям	Засіб реалізації
Захист службових ключів	Зберігання токена бота та Google credentials у .env і службових файлах
Розмежування доступу	Перевірка Telegram ID та ролі користувача
Захист адміністративних функцій	Доступ лише для користувачів зі службовими ролями
Контроль стану заявок	Використання статусів new, viewed, contacted, interested, inactive
Журналювання	Запис системних подій і помилок у журнал
Діагностика роботи	Команда /health і файлові логи
Контроль мережі	Перевірка доступності зовнішніх сервісів
Автозапуск	Запуск бота через systemd
Виявлення зупинки сервісу	Watchdog-таймер
Резервування даних	Копіювання SQLite-файлу та експорт даних

Таким чином, захист і контроль роботи системи забезпечуються поєднанням конфігураційного захисту, рольового доступу, журналювання, службового моніторингу та watchdog-контролю. Такі рішення є достатніми для локального серверного розгортання системи на Raspberry Pi.

Висновки до розділу 2

У другому розділі виконано проєктування комп’ютерної системи анкетування абітурієнтів із використанням сервера на базі Raspberry Pi. Обґрунтовано вибір Raspberry Pi як апаратної платформи для розгортання серверної частини, а також визначено основні програмні засоби реалізації: Python, Telegram Bot API, SQLite, Google Sheets API, планувальник задач, засоби конфігурації, логування, systemd та watchdog-контроль.

Розроблено структурну схему системи, діаграму прецедентів, основний алгоритм роботи, алгоритм проходження анкети та діаграми послідовностей для ключових процесів: подання заявки, роботи модератора із заявками, виконання сегментованої розсилки та контролю роботи системного сервісу.

Спроєктовано структуру бази даних, яка забезпечує збереження заявок абітурієнтів, розсилок, нагадувань, системних подій, динамічних анкет і результатів розіграшів. Також визначено інформаційні потоки між Telegram-клієнтом, Telegram Bot API, сервером Raspberry Pi, SQLite, Google Sheets та адміністративною частиною системи.

Розглянуто засоби захисту й адміністрування: зберігання службових параметрів поза програмним кодом, перевірку Telegram ID і ролі користувача, журналювання подій, команду контролю стану, автозапуск через systemd та watchdog-моніторинг.

Отримані результати проєктування є основою для подальшої програмної реалізації, розгортання системи на Raspberry Pi та перевірки працездатності основних функцій.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Структура програмного забезпечення системи

Програмне забезпечення комп'ютерної системи анкетування абітурієнтів реалізовано у вигляді модульного Python-застосунку. Основна програмна частина розміщена в папці bot, а файли для розгортання на Raspberry Pi винесені в окрему папку deploy/raspberry-pi. Така структура дає змогу відокремити логіку Telegram-бота, роботу з базою даних, адміністративні функції, службові сервіси, конфігурацію та засоби запуску системи.

Програмна частина системи складається з головного модуля запуску, модулів конфігурації, роботи з базою даних, обробників Telegram-команд, службових сервісів, засобів контролю доступу та файлів розгортання.

Основні файли та модулі програмного забезпечення наведено в табл. 3.1.

Таблиця 3.1 – Структура програмного забезпечення системи

Файл або модуль	Призначення
bot/main.py	Точка входу в програму, запуск бота, підключення маршрутизаторів, запуск планувальника та polling
bot/config.py	Завантаження параметрів із .env-файлу
bot/constants.py	Опис статусів заявок, типів питань і полів сегментації
bot/database.py	SQLite-схема бази даних і методи роботи з даними
bot/formatters.py	Формування текстових відповідей для адміністративних команд
bot/keyboards.py	Формування Telegram-клавіатур і кнопок
bot/security.py	Перевірка ролей і прав доступу користувачів
bot/segments.py	Обробка фільтрів сегментації для заявок і розсилок
bot/handlers/public.py	Обробка команд користувача та стандартної анкети абітурієнта
bot/handlers/admin.py	Адміністративні команди, статистика, заявки, статуси, розсилки, анкети, розіграші, логи
bot/handlers/surveys.py	Проходження динамічних анкет користувачами

Продовж. табл. 3.1

Файл або модуль	Призначення
bot/services/broadcasts.py	Надсилання запланованих повідомлень, розсилок і нагадувань
bot/services/google_sheets.py	Опційна інтеграція з Google Sheets
bot/services/monitoring.py	Логування, перевірка мережі та службові повідомлення адміністраторам
deploy/raspberry-pi/f7-admissions-bot.service	Systemd-служба для запуску Telegram-бота на Raspberry Pi
deploy/raspberry-pi/f7-admissions-watchdog.service	Systemd-служба для запуску watchdog-перевірки
deploy/raspberry-pi/f7-admissions- watchdog.timer	Таймер для періодичного запуску watchdog-контролю
deploy/raspberry-pi/watchdog.py	Скрипт перевірки активності основного systemd-сервісу
.env.example	Приклад конфігураційного файлу
requirements.txt	Перелік бібліотек, необхідних для роботи системи
docs/ARCHITECTURE.md	Опис архітектури системи
docs/DATABASE_LOGICAL_ STRUCTURE.md	Опис логічної структури бази даних

Головним файлом запуску є bot/main.py. У ньому завантажуються налаштування, створюється об’єкт бази даних, ініціалізується Telegram-бот, підключаються маршрутизатори обробників команд, запускається планувальник задач і стартує режим polling. Також у цьому файлі запускається фоновий моніторинг мережі та надсилається службове повідомлення адміністраторам про готовність системи до роботи.

Модуль bot/config.py відповідає за завантаження параметрів із .env-файлу. До таких параметрів належать токен Telegram-бота, список адміністраторів, ролі користувачів, шлях до SQLite-бази, шлях до лог-файлу, URL каналу, назва Google-таблиці, шлях до файлу сервісного акаунта Google, URL для перевірки мережі та інтервали службових перевірок. Винесення цих параметрів у конфігураційний файл дозволяє змінювати середовище запуску без редагування програмного коду.

Модуль bot/database.py реалізує рівень доступу до даних. У ньому описано створення таблиць leads, scheduled_messages, event_logs, giveaways, survey_forms, survey_questions, survey_submissions і survey_answers. Через цей модуль виконуються додавання та оновлення заявок, пошук і фільтрація

абітурієнтів, підрахунок кількості заявок, зміна статусів, запис службових подій, створення запланованих повідомлень, збереження результатів розіграшів і відповідей на динамічні анкети.

Окрему роль у структурі програми відіграє папка bot/handlers, яка містить обробники Telegram-команд. Модуль public.py реалізує користувацьку частину системи: команду /start, запуск анкети командою /apply, скасування дії командою /cancel і покрокове заповнення основної анкети. Для цього використовується FSM-модель, у якій окремими станами є введення імені, телефону, email, типу закладу освіти, курсу, року вступу та рівня зацікавленості в IT-сфері.

Модуль admin.py реалізує адміністративну частину системи. У ньому описано команди /admin, /stats, /leads, /find, /status, /export_csv, /news, /news_segment, /schedule_news, /remind, /forms, /form_create, /question_add, /form_activate, /giveaway, /events, /logs і /health. Ці команди забезпечують перегляд статистики, роботу із заявками, зміну статусів, експорт даних, створення розсилок і нагадувань, керування динамічними анкетами, проведення розіграшів, перегляд журналу подій і перевірку стану системи.

Модуль surveys.py відповідає за проходження додаткових динамічних анкет. Якщо адміністратор створив і активував анкету, користувач може пройти її командою /survey. Питання такої анкети зберігаються в базі даних, а відповіді користувачів записуються у таблиці проходжень і відповідей.

Папка bot/services містить службові модулі. Модуль broadcasts.py перевіряє чергу запланованих повідомлень у таблиці scheduled_messages і надсилає їх відповідним користувачам. У цій таблиці зберігаються як інформаційні розсилки, так і нагадування, а їх тип визначається полем message_type. Модуль google_sheets.py виконує передавання заявок до Google Sheets, якщо така інтеграція налаштована. Модуль monitoring.py відповідає за файлове логування, перегляд останніх рядків логу, перевірку доступності мережі та надсилання службових повідомлень адміністраторам.

Допоміжні модулі забезпечують підтримку основної логіки системи. У constants.py описано статуси заявок, допустимі типи питань і поля сегментації. У security.py реалізовано перевірку ролей owner, moderator і viewer. У segments.py реалізовано розбір фільтрів виду education=Коледж;year=2026. У keyboards.py сформовано звичайні та inline-клавіатури Telegram для анкети, вибору відповідей, адміністративної панелі та переходу до каналу.

Для розгортання системи на Raspberry Pi використовується папка deploy/raspberry-pi. Основний файл f7-admissions-bot.service забезпечує запуск Telegram-бота як systemd-служби. Файли f7-admissions-watchdog.service і f7-admissions-watchdog.timer використовуються для періодичного запуску watchdog-перевірки. Скрипт watchdog.py перевіряє активність основного сервісу f7-admissions-bot і в разі його зупинки надсилає службове повідомлення адміністратору через Telegram.

Перелік зовнішніх Python-бібліотек наведено у файлі requirements.txt. До основних залежностей належать aiogram, aiosqlite, apscheduler, python-dotenv, gspread, google-auth і aiohttp. Вони забезпечують роботу Telegram-бота, асинхронну взаємодію з SQLite, планування задач, завантаження конфігурації, інтеграцію з Google Sheets і перевірку мережевих ресурсів.

Таким чином, програмне забезпечення системи має чітку модульну структуру. Кожен файл виконує окрему функцію, що спрощує налагодження, супровід, тестування, перенесення системи на Raspberry Pi та подальше розширення її функціональних можливостей.

3.2 Налаштування та розгортання компонентів системи

Після розроблення програмної структури системи було виконано налаштування її основних компонентів і розгортання на серверній платформі Raspberry Pi. До складу компонентів, які потребують налаштування, належать середовище Raspberry Pi OS, Telegram-бот, конфігураційний файл системи,

локальна база даних SQLite, інтеграція з Google Sheets, systemd-служба автозапуску та watchdog-контроль.

Розгортання системи виконується у такій послідовності: створення Telegram-бота й отримання токена доступу, підготовка Raspberry Pi як серверної платформи, копіювання файлів проєкту, створення віртуального середовища Python, встановлення залежностей, налаштування конфігураційного файла, ініціалізація бази даних, ручна перевірка запуску, налаштування експорту в Google Sheets, запуск застосунку як systemd-служби та підключення watchdog-контролю.

3.2.1 Створення Telegram-бота та отримання токена доступу

Для взаємодії абітурієнтів із системою використовується Telegram-бот. Перед розгортанням серверного застосунку необхідно створити бота та отримати токен доступу до Telegram Bot API. Для цього використовується службовий бот Telegram — BotFather.

Спочатку в Telegram відкривається пошук і вводиться назва @BotFather. Серед результатів потрібно вибрати офіційного BotFather. Після відкриття діалогу надсилається команда: `/start`

У результаті BotFather виводить перелік доступних команд для створення та налаштування ботів.

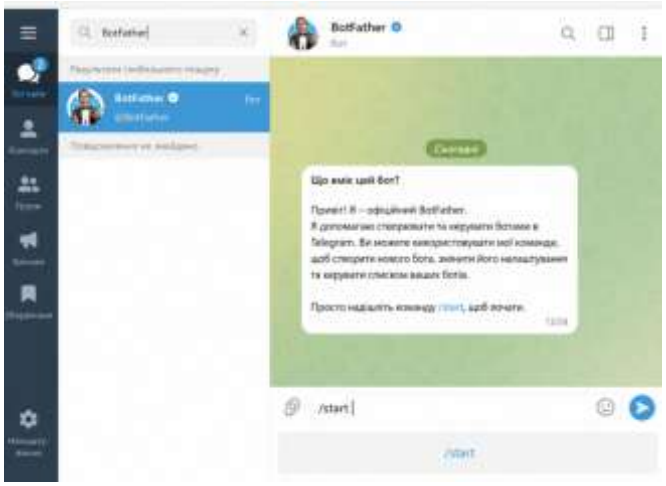

Рисунок 3.1 – Відкриття офіційного бота BotFather у Telegram

Для створення нового бота надсилається команда: `/newbot`. Після цього BotFather запитує назву бота, яка буде відображатися користувачам у Telegram. Для системи анкетування абітурієнтів можна використати назву F7 Admissions Bot або F7 Анкетування абітурієнтів. Після введення назви BotFather запитує username бота. Username повинен бути унікальним і завершуватися словом bot, наприклад: F7_TNTU_bot. Якщо введений username уже зайнятий, BotFather повідомляє про це, після чого потрібно ввести інший варіант. У результаті успішного створення бота BotFather надсилає токен доступу (рис. 3.2).

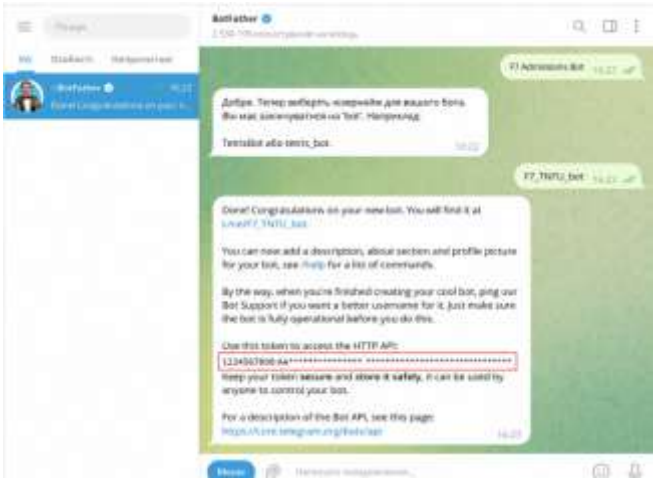

Рисунок 3.2 – Отримання токена доступу Telegram-бота

Отриманий токен використовується в конфігураційному файлі .env у параметрі BOT_TOKEN. Після цього серверна частина системи зможе підключатися до Telegram Bot API та отримувати повідомлення користувачів.

3.2.2 Підготовка серверного середовища Raspberry Pi

На наступному етапі готується Raspberry Pi як локальний сервер. Для цього виконується оновлення списку пакетів і встановлення базових компонентів, необхідних для роботи Python-застосунку:

```
sudo apt update
sudo apt install -y python3 python3-venv python3-pip git sqlite3
```

Результатом виконання цих команд є встановлення Python 3, модуля створення віртуального середовища, менеджера пакетів pip, системи керування версіями git і консольної утиліти sqlite3. Утиліта sqlite3 використовується для перевірки створеної локальної бази даних. Для перевірки встановлених компонентів використано команди:

```
python3 --version
sqlite3 –version
```

Після встановлення системних пакетів створюється робочий каталог проєкту:

```
sudo mkdir -p /opt/f7-admissions-bot
sudo chown -R pi:pi /opt/f7-admissions-bot
```

Каталог /opt/f7-admissions-bot використовується для розміщення програмного коду, конфігураційного файла, бази даних, логів і службових файлів запуску. Після копіювання файлів проєкту до цього каталогу його структура повинна містити папки bot, deploy, docs, файл requirements.txt, README.md і .env.example. Після цього створюється віртуальне середовище Python і встановлюються залежності проєкту:

```
cd /opt/f7-admissions-bot
python3 –m venv .venv
.venv/bin/pip install --upgrade pip
.venv/bin/pip install -r requirements.txt
```

У результаті створюється ізольоване середовище .venv, у яке встановлюються бібліотеки aiogram, aiosqlite, apscheduler, python-dotenv, gspread, google-auth і aiohttp. Вони забезпечують роботу Telegram-бота, SQLite-бази, планувальника задач, конфігурації, Google Sheets та мережевого моніторингу.

Наступним етапом налаштовується конфігураційний файл:

```
cp .env.example .env
nano .env
```

У файлі .env задаються основні параметри роботи системи:

```
BOT_TOKEN=token-from-botfather
ADMIN_IDS=123456789
ADMIN_ROLES=123456789:owner
DATABASE_PATH=./data/f7_admissions.sqlite3
LOG_FILE=./logs/f7_admissions.log
CHANNEL_URL=https://t.me/computer_engineering_tntu
SPREADSHEET_NAME=Leads
GOOGLE_SERVICE_ACCOUNT_FILE=
HEALTHCHECK_URL=https://api.telegram.org
NETWORK_CHECK_INTERVAL=60
SCHEDULER_INTERVAL_SECONDS=60
```

Параметр BOT_TOKEN містить токен Telegram-бота. ADMIN_IDS і ADMIN_ROLES задають адміністраторів та їхні ролі. DATABASE_PATH визначає шлях до SQLite-бази даних, LOG_FILE — шлях до лог-файлу, а GOOGLE_SERVICE_ACCOUNT_FILE використовується для підключення експорту в Google Sheets. Реальний .env не повинен публікуватися, оскільки містить службові параметри доступу.

3.2.3 Ініціалізація локальної бази даних SQLite

Окремим компонентом системи є локальна база даних SQLite. На відміну від MySQL або PostgreSQL, SQLite не потребує встановлення та запуску окремого серверного процесу. База даних зберігається у файлі, а таблиці створюються програмно під час ініціалізації застосунку. Для явної ініціалізації бази виконується команда:

```
.venv/bin/python - <<'PY'
import asyncio
import os
from pathlib import Path
from dotenv import load_dotenv
from bot.database import Database

load_dotenv()
database_path = Path(os.getenv("DATABASE_PATH",
"./data/f7_admissions.sqlite3"))

async def main():
    db = Database(database_path)
    await db.init()
    print(f"SQLite database initialized: {database_path}")
asyncio.run(main())
PY
```

Результатом виконання команди є створення каталогу data, файла бази даних f7_admissions.sqlite3 і необхідних таблиць. Після виконання скрипт виводить повідомлення:

```
SQLite database initialized: data/f7_admissions.sqlite3
```

Далі перевірено створення таблиць бази даних:

```
sqlite3 data/f7_admissions.sqlite3 ".tables"
```

Результатом виконання команди є виведення переліку таблиць:

```
event_logs          scheduled_messages   survey_questions
giveaways           survey_answers       survey_submissions
leads               survey_forms
```

Таблиця leads використовується для збереження основних заявок абітурієнтів. Таблиця scheduled_messages зберігає інформаційні розсилки та нагадування. Таблиця event_logs використовується для журналу системних подій. Таблиця giveaways зберігає результати розіграшів, а таблиці survey_forms, survey_questions, survey_submissions і survey_answers забезпечують роботу динамічних анкет.

3.2.4 Ручний запуск і перевірка Telegram-бота

Після налаштування конфігурації та бази даних виконується ручний запуск Telegram-бота:

```
.venv/bin/python -m bot.main
```

Результатом виконання команди є запуск серверного Python-застосунку, ініціалізація бази даних, підключення до Telegram Bot API, запуск планувальника задач і перехід у режим очікування повідомлень.

На цьому етапі в Telegram відкривається створений бот і надсилається команда: /start (рис.3.4). Якщо токен указано правильно, бот надсилає стартове повідомлення та пропонує почати анкетування.

Рисунок 3.4 – Перевірка запуску Telegram-бота командою /start

Після цього виконується перевірка основної анкети командою: /apply

Користувач послідовно вводить ім’я, номер телефону, email, тип закладу освіти, курс, рік вступу та рівень зацікавленості в IT-сфері. Результатом завершення анкети є повідомлення користувачу про прийняття заявки та повідомлення адміністратору про нову заявку.

Для перевірки запису в базу даних після проходження тестової анкети виконується запит:

```
sqlite3 data/f7_admissions.sqlite3 "SELECT telegram_id, name, phone, email, education, year, status FROM leads;"
```

Результатом виконання команди є виведення записів із таблиці leads. Приклад вмісту тестового запису:

```
123456789 | Тестовий користувач| +380XXXXXXXXX | test@example.com | Школа |2026|new
```

3.2.5 Налаштування експорту результатів у Google Sheets

Google Sheets у системі використовується як додатковий сервіс для перегляду та спільної роботи з результатами анкетування. Основним

сховищем залишається SQLite, тому навіть у разі відсутності доступу до Google Sheets система продовжує приймати заявки.

Спочатку в Google Drive створюється нова електронна таблиця. Їй задається назва, яка надалі буде вказана у конфігураційному файлі системи. У проєкті використовується назва Leads (рис.3.5), тому параметр у .env має відповідати назві створеної таблиці: SPREADSHEET_NAME=Leads

Рисунок 3.5 – Створення Google-таблиці для збереження заявок абітурієнтів

Після створення таблиці відкривається Google Cloud Console (рис.3.6). У ній створюється новий проєкт або вибирається наявний. Наприклад, можна створити проєкт із назвою F7 Admissions Bot.

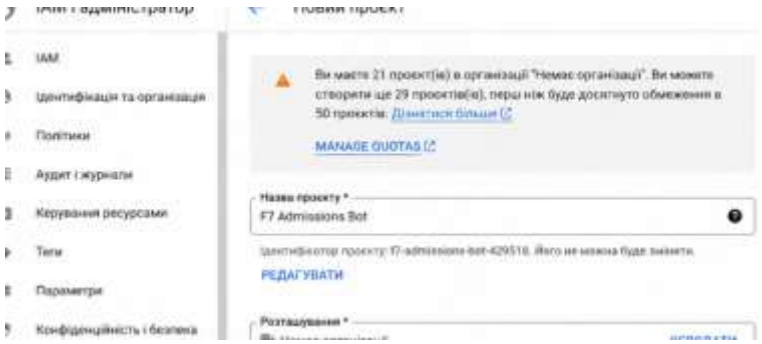

Рисунок 3.6 – Створення проєкту в Google Cloud Console

Необхідно ввімкнути Google Sheets API. Для цього в Google Cloud Console відкривається розділ API Library, у пошуку вводиться Google Sheets API, після чого відкривається сторінка API та натискається кнопка Enable (рис. 3.7). Активація API потрібна для того, щоб серверний застосунок міг додавати рядки до таблиці через програмний інтерфейс.

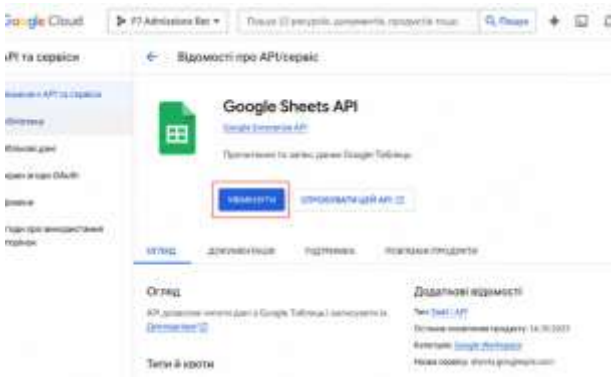

Рисунок 3.7 – Увімкнення Google Sheets API для проєкту

Далі створюється службовий акаунт. У Google Cloud Console відкривається розділ IAM & Admin → Service Accounts, після чого створюється новий service account (рис.3.8). Йому задається назва, наприклад f7-admissions-sheets. Після створення потрібно скопіювати email службового акаунта, оскільки саме цій адресі буде надано доступ до Google-таблиці.

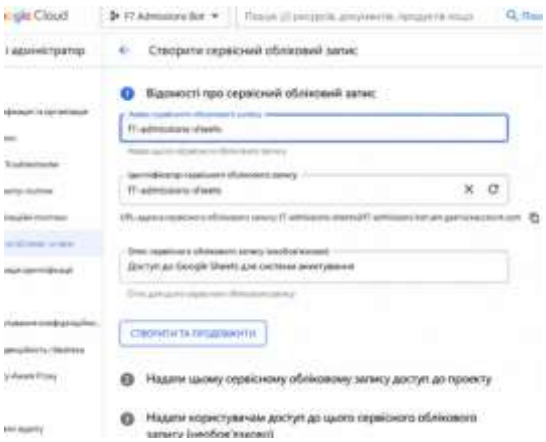

Рисунок 3.8 – Створення service account для доступу до Google Sheets

Для службового акаунта створюється ключ доступу у форматі JSON. Для цього відкривається створений service account, переходиться до вкладки Keys, вибирається Add key → Create new key, після чого задається тип ключа JSON. У результаті Google Cloud завантажує JSON-файл із параметрами доступу.

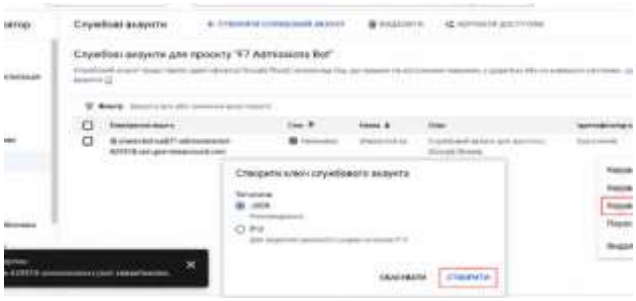

Рисунок 3.9 – Створення JSON-ключа службового акаунта

Отриманий JSON-файл переноситься на Raspberry Pi в захищений каталог. Для цього в каталозі проєкту створюється папка secrets:

```
mkdir -p /opt/f7-admissions-bot/secrets
chmod 700 /opt/f7-admissions-bot/secrets
```

Створюється каталог для службових ключів з обмеженим доступом. JSON-файл розміщується в цьому каталозі:

```
/opt/f7-admissions-bot/secrets/google-service-account.json
```

Після цього шлях до JSON-файла задається у .env:

```
GOOGLE_SERVICE_ACCOUNT_FILE=/opt/f7-admissions-
bot/secrets/google-service-account.json
```

Наступним кроком Google-таблиці надається доступ для службового акаунта. Для цього відкривається створена таблиця Leads, натискається кнопка Share, і в поле доступу вставляється email service account. Для коректної роботи експорту акаунту надається роль редактора. Після цього серверний застосунок отримує право додавати нові рядки до таблиці. Після завершення налаштування Google Sheets перезапускається systemd-служба Telegram-бота:

```
sudo systemctl restart f7-admissions-bot
```

Далі через Telegram проходиться тестова анкета командою:

```
/apply
```

Після завершення анкети дані зберігаються в SQLite, а також передаються до Google Sheets. Результатом коректного налаштування

експорту є поява нового рядка в таблиці Leads. У рядку відображаються дата, Telegram ID, username, ім’я, телефон, email, тип закладу освіти, курс, рік вступу, інтерес до IT-сфери та статус заявки.

Рисунок 3.10 – Результат експорту тестової заявки до Google Sheets

3.2.6 Налаштування автозапуску через systemd і watchdog-контролю

Після перевірки ручного запуску систему налаштовано як systemd-службу. Для цього файл служби копіюється до системного каталогу:

```
sudo cp deploy/raspberry-pi/f7-admissions-bot.service
/etc/systemd/system/f7-admissions-bot.service
sudo systemctl daemon-reload
sudo systemctl enable f7-admissions-bot
sudo systemctl start f7-admissions-bot
```

Результатом виконання цих команд є реєстрація служби f7-admissions-bot у systemd, додавання її до автозапуску та запуск Telegram-бота як системного сервісу. Стан служби перевіряється командою:

```
sudo systemctl status f7-admissions-bot
```

в результаті має бути результат:

```
Active: active (running)
```

Це підтверджує, що бот працює як системна служба.

Для контролю працездатності основного сервісу налаштовується watchdog-механізм. Він потрібний для виявлення ситуацій, коли основний бот зупинився і самостійно вже не може повідомити адміністратора про проблему. Watchdog реалізовано як окремий скрипт і systemd-таймер, який періодично запускає перевірку активності основного сервісу. Налаштування watchdog виконується такими командами (рис.3.11).

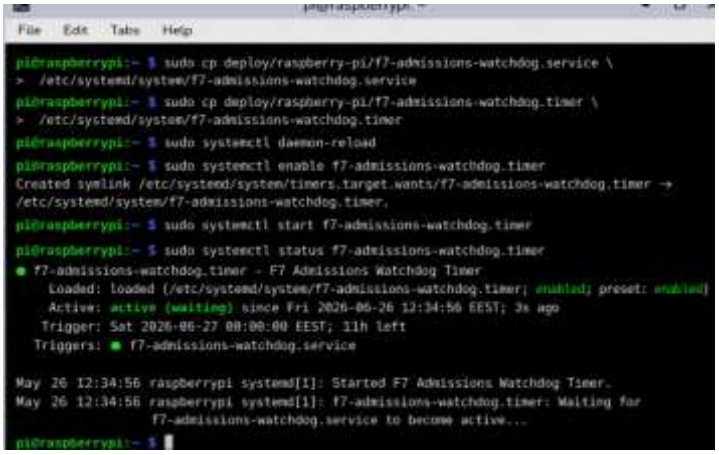

Рисунок 3.11 – Налаштування watchdog

Результатом виконання команд є реєстрація watchdog-служби та запуск таймера, який періодично перевіряє активність основного сервісу.

Активність таймера перевіряється командою:

```
systemctl list-timers | grep f7-admissions
```

Результатом виконання команди є відображення таймера f7-admissions-watchdog.timer у списку активних systemd-таймерів. Якщо основний сервіс f7-admissions-bot переходить у неактивний стан, watchdog зчитує параметри з .env і надсилає службове повідомлення адміністратору в Telegram.

Таким чином, у процесі розгортання було налаштовано всі основні компоненти системи: Telegram-бот, серверне середовище Raspberry Pi, Python-віртуальне середовище, конфігураційний файл, SQLite-базу даних, інтеграцію з Google Sheets, systemd-автозапуск і watchdog-контроль. Після виконання цих етапів система готова до роботи як автономний серверний застосунок для анкетування абітурієнтів.

3.3 Тестування працездатності системи

Після налаштування та розгортання компонентів системи було виконано тестування її працездатності. Метою тестування є перевірка того, що система коректно виконує основні функції: приймає повідомлення користувачів через

Telegram, забезпечує проходження анкети, зберігає заявки в SQLite, передає результати в Google Sheets, надає адміністративні функції, виконує розсилки, веде журнал подій і підтримує службовий контроль стану.

Тестування проводилося після запуску системи на Raspberry Pi. Перевірка охоплювала користувацьку частину Telegram-бота, адміністративну частину, локальну базу даних, інтеграцію з Google Sheets, розсилки, динамічні анкети, розіграші, логування та watchdog-контроль.

На першому етапі перевірено доступність Telegram-бота для користувача. Після відкриття створеного бота F7 Admissions Bot у Telegram було надіслано стартову команду. У відповідь бот відобразив привітальне повідомлення, коротко пояснив призначення системи та запропонував перейти до заповнення анкети. Це підтверджує, що серверна частина коректно підключена до Telegram Bot API та приймає повідомлення користувачів (рис.3.12).

Рисунок 3.12 – Перевірка відповіді Telegram-бота на стартову команду

Наступним етапом перевірено проходження основної анкети абітурієнтом. Користувач запускав сценарій анкетування, після чого бот послідовно запитував ім’я, номер телефону, email, тип закладу освіти, курс, рік вступу та рівень зацікавленості в IT-сфері. Частина відповідей вводилася вручну, а частина вибиралася за допомогою кнопок Telegram-клавіатури.

Під час тестування встановлено, що бот коректно переходить між етапами анкети, не пропускає обов’язкові поля та завершує сценарій лише

після отримання всіх необхідних даних. Після завершення анкетування користувач отримував повідомлення про успішне прийняття заявки, а адміністратор — службове повідомлення про нову заявку (рис. 3.13).

Рисунок 3.13 – Проходження основної анкети абітурієнтом

Далі перевірено експорт результатів у Google Sheets. Після завершення тестової анкети в Google-таблиці Leads з’явився новий рядок із даними абітурієнта. Це підтверджує коректну роботу інтеграції з Google Sheets API, службового акаунта, JSON-ключа доступу та механізму дублювання даних із локальної бази до зовнішнього табличного сервісу.

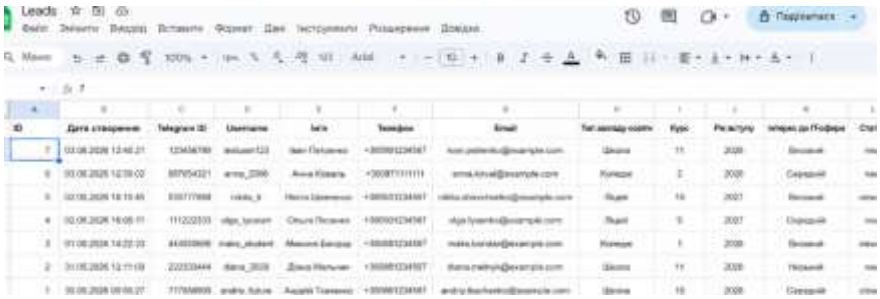

Рисунок 3.14 – Результат експорту заявки абітурієнта в Google Sheets

Для додаткового підтвердження роботи інтеграції було перевірено статистику використання API в Google Cloud Console. На панелі моніторингу відображалися звернення до Google Sheets API та Google Drive API. За

результатами перевірки Google Sheets API обробив 46 запитів без помилок. Частка помилок становила 0 %, медіанна затримка відповідей — 109 мс, а затримка 95-го перцентиля — 373 мс. Також було зафіксовано 4 запити до Google Drive API з нульовим відсотком помилок, медіанною затримкою 393 мс і затримкою 95-го перцентиля 511 мс (рис.3.15-3.16).

Рисунок 3.15 – Графіки активності Google Sheets API та Google Drive API

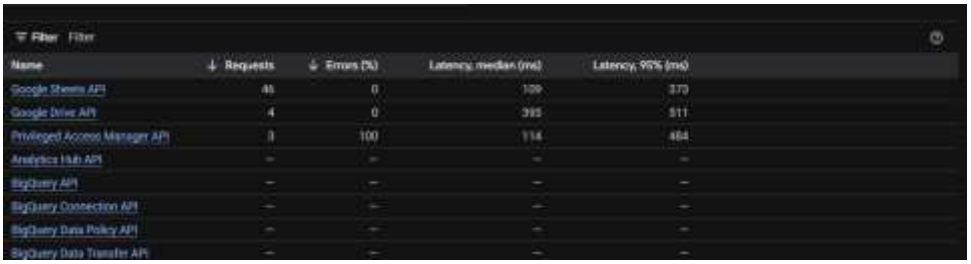

Рисунок 3.16 – Статистика запитів до API у Google Cloud Console

Наявність запитів до Google Sheets API свідчить про фактичне виконання обміну даними між серверним застосунком і Google-таблицею. Відсутність помилок у Google Sheets API підтверджує коректність налаштування Google Cloud-проєкту, службового акаунта, прав доступу до таблиці та механізму експорту результатів анкетування.

Окремо перевірено адміністративну частину системи. Для користувача з адміністративним Telegram ID було доступне меню керування. Адміністратор мав змогу переглядати статистику, список заявок, виконувати пошук, змінювати статуси, експортувати дані у CSV, створювати розсилки, переглядати події, логи та службовий стан системи (рис.3.17). Для звичайного користувача адміністративні функції були недоступні, що підтверджує роботу механізму розмежування прав.

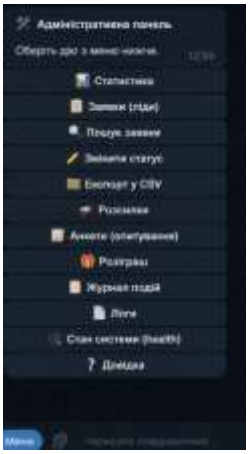

Рисунок 3.17 – Відображення адміністративної панелі Telegram-бота

Під час перевірки роботи із заявками було підтверджено, що адміністратор може переглядати останні заявки, виконувати пошук за контактними даними та змінювати статус заявки. Після зміни статусу повторний перегляд заявки показував оновлене значення. Це підтверджує коректну роботу операцій вибірки та оновлення записів у базі даних.

Після перевірки користувацьких і адміністративних функцій протестовано службові механізми. Команда перевірки стану системи виводила інформацію про роботу бота, кількість заявок, доступність зовнішніх сервісів, шлях до лог-файла та останні службові повідомлення. Перегляд журналу подій показав, що система фіксує створення заявок, зміну статусів, запуск розсилок, помилки інтеграції та інші важливі події.

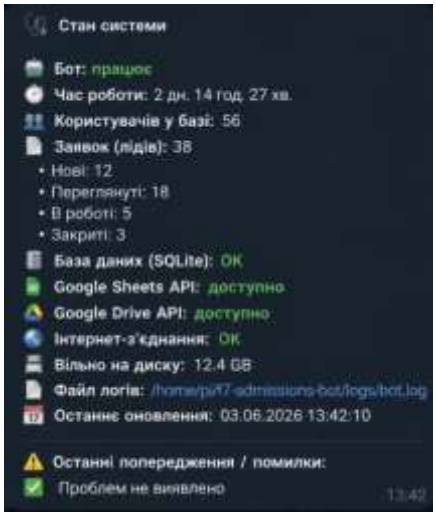

Рисунок 3.18 – Перевірка службового стану системи

Також перевірено watchdog-контроль. Після налаштування таймера система відображала активний watchdog-таймер, який періодично запускає перевірку основного сервісу Telegram-бота.

За результатами тестування встановлено, що система виконує основні функції, передбачені технічним завданням. Telegram-бот коректно приймає команди користувачів, забезпечує проходження анкети, зберігає заявки в SQLite, дублює дані в Google Sheets, підтримує адміністративні функції, розсилки, динамічні анкети, розіграші, журналювання та контроль працездатності. Наявність успішних запитів до Google Sheets API без помилок додатково підтверджує працездатність зовнішньої інтеграції. Отже, система готова до використання для анкетування абітурієнтів.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Психологічні причини нещасних випадків і травматизму

Безпека життєдіяльності вивчає закономірності виникнення небезпек у процесі життєдіяльності людини та розробляє заходи, спрямовані на збереження її життя, здоров'я і працездатності. Одним із важливих напрямів безпеки життєдіяльності є дослідження впливу психологічних факторів на поведінку людини в небезпечних ситуаціях. Практика свідчить, що значна частина нещасних випадків, аварій та надзвичайних подій пов'язана не лише з технічними несправностями чи несприятливими умовами навколишнього середовища, а й з особливостями психічної діяльності людини.

Встановлено, що людський фактор є однією з основних причин виникнення небезпечних ситуацій у різних сферах діяльності. Навіть за наявності сучасних технічних засобів захисту та чітко визначених правил безпеки людина може припускатися помилок, які призводять до виникнення небезпеки для себе та оточуючих [17].

Особливе значення для безпеки життєдіяльності має вивчення психічних станів людини. Втома, нервове напруження, тривога, емоційне виснаження та стрес знижують швидкість реакції, погіршують концентрацію уваги та здатність приймати правильні рішення. У таких умовах людина може неправильно оцінити обстановку, пропустити важливі сигнали небезпеки або виконати дії, що суперечать вимогам безпеки. Дослідження показують, що ризик виникнення нещасних випадків значно зростає під час тривалого фізичного або розумового навантаження, нестачі відпочинку та порушення режиму праці й сну.

Важливим завданням безпеки життєдіяльності є також аналіз поведінкових особливостей людини. Деякі люди схильні до невиправданого ризику, нехтування встановленими правилами та переоцінки власних можливостей. Такі риси характеру можуть стати причиною порушення вимог безпеки та виникнення небезпечних ситуацій. Водночас надмірна невпевненість, страх або паніка також негативно впливають на безпеку, оскільки ускладнюють прийняття рішень у критичних умовах [17].

У психологічній класифікації причин нещасних випадків виділяють порушення мотиваційної та виконавчої складових діяльності людини. Порушення мотиваційної складової проявляється у небажанні дотримуватися встановлених вимог безпеки, недооцінці небезпеки або свідомому ігноруванні правил. Порушення виконавчої складової виникає через зниження працездатності, втому, погіршення стану здоров’я або невідповідність психофізіологічних можливостей людини вимогам конкретної діяльності.

Безпека життєдіяльності враховує, що причини травматизму можуть формуватися на кількох рівнях. На індивідуальному рівні визначальне значення мають особливості нервової системи, темперамент, характер, рівень підготовки, життєвий досвід та стан здоров’я людини. На рівні найближчого соціального середовища важливу роль відіграють психологічний клімат у колективі, взаємовідносини між людьми, якість навчання та інструктажів. На суспільному рівні значення мають рівень культури безпеки населення, ефективність інформаційно-роз’яснювальної роботи та ставлення суспільства до питань безпеки [18].

Психологічні фактори особливо яскраво проявляються під час виникнення надзвичайних ситуацій. У стані небезпеки люди можуть втрачати здатність адекватно оцінювати обстановку, діяти хаотично або піддаватися панічним настроям. Паніка є одним із найнебезпечніших психологічних явищ, оскільки призводить до втрати самоконтролю, дезорганізації колективних дій та збільшення кількості постраждалих. Саме тому важливим елементом підготовки населення є формування психологічної готовності до дій у

надзвичайних ситуаціях, розвиток навичок самоконтролю та здатності діяти відповідно до встановлених алгоритмів поведінки.

Серед психологічних причин травматизму особливе місце займають неуважність, монотонність праці, інформаційне перевантаження та зниження рівня пильності. Тривале виконання однотипних операцій або перебування в умовах підвищеної відповідальності може спричиняти втрату концентрації уваги, що підвищує ймовірність помилкових дій. Крім того, негативний вплив на безпечну поведінку людини мають конфліктні ситуації, психоемоційні перевантаження, особисті проблеми та несприятливий психологічний клімат у колективі [18].

З метою зниження ризику нещасних випадків безпека життєдіяльності передбачає комплекс заходів, спрямованих на врахування людського фактора. До таких заходів належать професійне навчання, формування культури безпечної поведінки, проведення інструктажів і тренувань, психологічна підготовка до дій в екстремальних умовах, забезпечення раціонального режиму праці та відпочинку, а також створення сприятливого психологічного клімату. Важливе значення має виховання особистої відповідальності за власну безпеку та безпеку оточуючих.

Отже, безпека життєдіяльності розглядає психологічні фактори як одну з основних причин виникнення небезпечних ситуацій і нещасних випадків. Врахування особливостей психіки людини, її поведінки та психоемоційного стану дає можливість суттєво знизити рівень травматизму, підвищити ефективність профілактичних заходів та забезпечити належний рівень захисту людини в різних сферах життєдіяльності.

4.2 Вимоги до режимів праці і відпочинку при роботі з ВДТ

Сучасна професійна діяльність у сфері інформаційних технологій, комп’ютерної інженерії та автоматизованих систем управління пов’язана з тривалим використанням відеодисплейних терміналів (ВДТ). Робота за

комп’ютером супроводжується значним навантаженням на органи зору, нервову систему, опорно-руховий апарат та психоемоційну сферу людини. Тривале перебування перед екраном монітора без дотримання встановлених режимів праці та відпочинку може призвести до розвитку професійно зумовлених захворювань, зниження працездатності, погіршення самопочуття та підвищення ризику виникнення виробничих помилок [19].

Основною метою організації раціонального режиму праці та відпочинку є збереження здоров’я працівників, підтримання високого рівня працездатності та профілактика негативного впливу факторів виробничого середовища. Особливу увагу при роботі з ВДТ приділяють профілактиці зорового стомлення, зменшенню статичних навантажень на м’язи спини, шиї та верхніх кінцівок, а також запобіганню нервово-психічному перенапруженню.

Під час роботи з відеодисплейними терміналами працівник тривалий час перебуває у вимушеній робочій позі та виконує однотипні рухи руками під час роботи з клавіатурою і маніпулятором типу «миша». Одночасно відбувається постійне навантаження на зоровий аналізатор, оскільки необхідно сприймати значний обсяг інформації з екрана монітора. Це може викликати втому очей, сухість слизової оболонки, зниження гостроти зору, головний біль та загальну втому організму.

Відповідно до вимог охорони праці, тривалість безперервної роботи за комп’ютером повинна обмежуватися регламентованими перервами. Перерви необхідні для відновлення функціонального стану організму, зниження напруження зорової системи та попередження розвитку перевтоми. Під час перерв рекомендується змінювати вид діяльності, виконувати вправи для очей, розминку для м’язів шиї, плечового пояса та спини, а також здійснювати короткочасні прогулянки або іншу фізичну активність.

Одним із найбільш поширених негативних наслідків роботи з ВДТ є комп’ютерний зоровий синдром. Його основними проявами є подразнення очей, сльозотеча або сухість, погіршення фокусування зору, відчуття втоми та

дискомфорту. Для профілактики таких явищ рекомендується дотримуватися правила «20–20–20»: кожні 20 хвилин роботи переводити погляд на об’єкт, розташований на відстані приблизно 6 метрів, протягом не менше 20 секунд. Крім того, необхідно підтримувати оптимальну відстань від очей до екрана монітора, яка повинна становити 50–70 см залежно від розміру дисплея та індивідуальних особливостей працівника [19].

Раціональна організація режимів праці та відпочинку також передбачає чергування розумових і фізичних навантажень. У разі можливості доцільно поєднувати роботу за комп’ютером з іншими видами діяльності, що не потребують постійного спостереження за екраном. Такий підхід сприяє зниженню функціонального напруження організму та підвищенню ефективності праці.

Важливим фактором профілактики професійного стомлення є дотримання вимог ергономіки робочого місця. Робочий стіл, крісло, монітор та пристрої введення повинні бути розташовані таким чином, щоб забезпечувати природне положення тіла працівника. Верхня межа екрана повинна знаходитися на рівні очей або трохи нижче, а кут огляду має забезпечувати комфортне сприйняття інформації без надмірного нахилу голови. Неправильна організація робочого місця значно прискорює розвиток втоми та збільшує ризик виникнення захворювань опорно-рухового апарату.

Особливу увагу необхідно приділяти психоемоційному навантаженню працівників, які працюють із комп’ютерними системами. Висока відповідальність за результати роботи, необхідність швидкого прийняття рішень, обробка великих обсягів інформації та постійна концентрація уваги можуть призводити до розвитку стресових станів. Для зменшення негативного впливу таких факторів рекомендується забезпечувати сприятливий психологічний клімат у колективі, раціонально розподіляти навантаження та не допускати надмірної тривалості робочого дня [20].

Для працівників, діяльність яких пов’язана з розробкою, налагодженням та експлуатацією комп’ютерних систем, важливою складовою режиму праці є

дотримання встановленого графіка роботи та відпочинку. Регулярні перерви дозволяють підтримувати високу продуктивність праці, знижують ризик виникнення професійних захворювань і сприяють збереженню здоров’я працівників. Правильно організований режим праці та відпочинку є одним із найефективніших заходів охорони праці під час роботи з відеодисплейними терміналами.

Отже, дотримання вимог до режимів праці та відпочинку при роботі з ВДТ є необхідною умовою забезпечення безпечних і комфортних умов праці. Раціональне чергування роботи та відпочинку, виконання профілактичних вправ, дотримання ергономічних вимог і підтримання сприятливого психоемоційного стану дозволяють зменшити негативний вплив комп’ютерної техніки на організм людини та забезпечити високий рівень працездатності протягом тривалого часу.

ВИСНОВКИ

У кваліфікаційній роботі розроблено комп'ютерну систему анкетування абітурієнтів із серверною частиною на базі Raspberry Pi. У результаті виконання роботи отримано такі основні результати:

1. Проаналізовано предметну область систем анкетування та визначено доцільність використання Telegram-бота як клієнтського інтерфейсу для збору даних абітурієнтів.

2. Обґрунтовано вибір архітектури системи, у якій Telegram використовується як засіб взаємодії з користувачем, а Raspberry Pi — як локальна серверна платформа для запуску Python-застосунку, бази даних, служб логування, моніторингу та контролю працездатності.

4. Обґрунтовано вибір апаратних і програмних засобів реалізації. Для розробки використано Raspberry Pi, Python, бібліотеку aiogram, SQLite, Google Sheets API, APScheduler, python-dotenv, systemd і watchdog-механізм.

6. Розроблено структуру бази даних системи. Передбачено таблиці для збереження основних заявок абітурієнтів, запланованих повідомлень, службових подій, розіграшів, проходжень анкет і відповідей користувачів.

7. Реалізовано Telegram-бот для анкетування абітурієнтів. Бот забезпечує запуск анкети, покрокове введення даних користувача, збереження її в SQLite, експорт у Google Sheets надсилання повідомлення адміністратору.

8. Реалізовано адміністративну частину системи. Адміністратор може переглядати статистику та заявки, виконувати пошук, змінювати статуси, експортувати дані у CSV, створювати розсилки, керувати динамічними анкетами, проводити розіграші, переглядати журнал подій, лог-файли та службовий стан системи. Забезпечено базове розмежування доступу до службових функцій.

12. Проведено тестування працездатності системи. Перевірено відповідь Telegram-бота на стартову команду, проходження основної анкети, запис заявки в SQLite, експорт у Google Sheets, роботу адміністративної панелі,

пошук і зміну статусів заявок, CSV-експорт, розсилки, динамічні анкети, розіграші, журналювання, службовий стан і watchdog-контроль.

Отримані результати підтверджують, що розроблена система готова до практичного використання для автоматизованого анкетування абітурієнтів, формування бази контактів, організації подальшої комунікації та адміністрування процесу збору даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Жаровський Р.О., Луцик Н.С., Осухівська Г.М., Паламар А.М., Тиш Є.В. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти за спеціальністю 123 «Комп'ютерна інженерія» усіх форм навчання. Тернопіль: ТНТУ, 2024. 39 с.

2. Паламар М.І., Стрембіцький М.О., Паламар А.М. Проектування комп'ютеризованих вимірювальних систем і комплексів: навчальний посібник. Тернопіль: ТНТУ, 2019. 150 с.

3. Python Software Foundation. Python 3 Documentation. URL: https://docs.python.org/3/ (дата звернення: 05.06.2026).

4. Telegram. Telegram Bot API. URL: https://core.telegram.org/bots/api (дата звернення: 05.06.2026).

5. aiogram Team. aiogram 3 Documentation. URL: https://docs.aiogram.dev/en/stable/ (дата звернення: 05.06.2026).

6. Рожик А.М., Жаровський Р.О. Аналіз ефективності роботи адаптивної системи контролю доступу на основі нечіткої логіки. Матеріали наукової конференції ТНТУ. Тернопіль: ТНТУ, 2025. С. 12–13.

7. Рожик А.М., Жаровський Р.О. Методи та програмно-апаратні засоби ідентифікації працівників з метою визначення робочого часу та доступу до приміщення. Збірник тез доповідей науково-технічної конференції. Тернопіль: ТНТУ, 2025. С. 45.

8. Дячук О.А., Жаровський Р.О. Використання SDN для оптимізації передачі даних в комп'ютерних мережах. Матеріали XI науково-технічної конференції ТНТУ імені Івана Пулюя «Інформаційні моделі системи та технології». Тернопіль: ТНТУ, 2023. С. 149–150.

9. Raspberry Pi Ltd. Raspberry Pi OS Documentation. URL: https://www.raspberrypi.com/documentation/computers/os.html (дата звернення: 05.06.2026).

10. Дячук О.А., Жаровський Р.О. Управління потоком за критеріями доступності. Матеріали XI науково-технічної конференції ТНТУ імені Івана Пулюя «Інформаційні моделі системи та технології». Тернопіль: ТНТУ, 2023. С. 151.

11. Ковтун Н., Жаровський Р. Алгоритмічне забезпечення систем виявлення вторгнень. Матеріали XI науково-технічної конференції ТНТУ імені Івана Пулюя «Інформаційні моделі системи та технології». Тернопіль: ТНТУ, 2023. С. 156.

12. SQLite Consortium. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 05.06.2026).

13. Ковтун Н., Жаровський Р. Аналіз засобів протидії вторгненням і атакам на комп'ютерні системи. Матеріали XII Міжнародної науково-технічної конференції молодих учених та студентів «Актуальні задачі сучасних технологій». Тернопіль: ТНТУ, 2023. С. 453–454.

14. Google for Developers. Google Sheets API Documentation. URL: https://developers.google.com/sheets/api (дата звернення: 05.06.2026).

15. Микитишин А.Г., Митник М.М., Стухляк П.Д. Телекомунікаційні системи та мережі. Тернопіль: ТНТУ імені Івана Пулюя, 2017. 384 с.

16. Service unit configuration. URL: https://www.freedesktop.org/software /systemd/man/latest /systemd.service.html (дата звернення: 05.06.2026).

17. Жидецький В. Ц. Безпека життєдіяльності: підручник. Львів: Афіша, 2022. 320 с.

18. Запорожець О. І., Протоєрейський О. С., Франчук Г. М. Безпека життєдіяльності: підручник. К.: Центр учбової літератури, 2023. 448с.

19. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями: Наказ Міністерства соціальної політики України від 14.02.2018 р. № 207. URL: https://zakon.rada.gov.ua/laws/show/z0508-18 (дата звернення: 05.06.2026).

20. Жидецький В. Ц. Охорона праці користувачів комп'ютерів: підручник. Львів : Афіша, 2020. 176 с.

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра комп'ютерних систем та мереж

"Затверджую"

Завідувач кафедри КС

_____ Осухівська Г.М.

" 2 " лютого 2026 р.

КОМП'ЮТЕРНА СИСТЕМА АНКЕТУВАННЯ АБІТУРІЄНТІВ З ВИКОРИСТАННЯМ СЕРВЕРА НА БАЗІ RASPBERRY PI

ТЕХНІЧНЕ ЗАВДАННЯ

на 10 листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп'ютерна інженерія»

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник кваліфікаційної роботи	Студент групи СІ-42
_____ к.т.н., доц. Жаровський Р.О.	_____ Юрков М. С.
" 2 " лютого 2026 р.	" 2 " лютого 2026 р.

Тернопіль 2026

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: «Комп’ютерна система анкетування абітурієнтів з використанням сервера на базі Raspberry Pi».

Умовне позначення кваліфікаційної роботи: КС КРБ 123.200.00.00

1.2 Виконавець

Студент групи СІ-42, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Юрков М. С.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету № 4/9-188 від 24.04.2026 р.

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи – 26.01.2026 р.

Плановий термін завершення виконання кваліфікаційної роботи – 05.06.2026 р.

1.5 Порядок оформлення та пред’явлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ISO, ЕСКД, ЕСПД та ДСТУ.

Пред’явлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи. Попередній захист кваліфікаційної роботи відбувається при готовності роботи – наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

2.1 Призначення системи

Проєктована комп’ютерна система призначена для автоматизованого анкетування абітурієнтів із використанням Telegram-бота та серверної частини на базі Raspberry Pi.

Система повинна забезпечувати проходження анкети абітурієнтом у Telegram, збереження контактних та освітніх даних у локальній базі даних SQLite, дублювання результатів у Google Sheets, повідомлення адміністратора про нові заявки та подальше адміністрування зібраної інформації.

Система повинна виконувати ведення журналу подій і контроль працездатності серверного застосунку..

2.2 Мета створення системи

Метою створення комп’ютерної системи є підвищення ефективності збору та опрацювання анкетних даних абітурієнтів шляхом автоматизації процесу анкетування, збереження результатів, повідомлення відповідальних осіб і адміністрування заявок.

..

2.3 Характеристика об'єкту

Об’єктом автоматизації є процес збору, збереження, опрацювання та адміністрування анкетних даних абітурієнтів.

3 Вимоги до системи

3.1 Вимоги до системи в цілому

Система повинна забезпечувати автоматизований збір анкетних даних абітурієнтів, їх локальне збереження, дублювання у зовнішній табличний сервіс, адміністрування заявок і контроль працездатності серверної частини.

Система повинна працювати на Raspberry Pi як автономний серверний застосунок і забезпечувати доступ користувачів через Telegram без потреби у встановленні додаткового програмного забезпечення або використанні окремого вебінтерфейсу.

3.1.1 Вимоги до структури та функціонування системи

До складу системи повинні входити:

– одноплатний комп’ютер Raspberry Pi;

– операційна система Raspberry Pi OS;

– серверний Python-застосунок;

– Telegram-бот;

– Telegram Bot API;

– локальна база даних SQLite;

– модуль інтеграції з Google Sheets.

Система повинна забезпечувати такі основні функції:

– запуск Telegram-бота на Raspberry Pi;

– приймання команд і повідомлень користувачів;

– покрокове проходження анкети абітурієнтом;

– перевірку та обробку введених даних;

– збереження заявки в SQLite;

– експорт заявки в Google Sheets;

– перегляд журналу подій і лог-файлів;

– перевірку службового стану системи;

– автоматичний запуск після перезавантаження Raspberry Pi;

– watchdog-контроль активності основного сервісу.

3.1.2 Вимоги до способів та засобів зв’язку між компонентами системи

Обмін інформацією між користувачем і серверною частиною системи повинен здійснюватися через Telegram Bot API.

Для отримання повідомлень від Telegram повинен використовуватися режим polling, що дозволяє запускати систему на Raspberry Pi без налаштування публічного HTTPS-сервера, доменного імені та відкритих вхідних портів.

Для взаємодії компонентів системи повинні використовуватися:

– Telegram Bot API для обміну повідомленнями між користувачем і ботом;

– HTTPS-запити до зовнішніх сервісів;

– SQLite-файл для локального збереження даних;

– Google Sheets API для передавання результатів у зовнішню таблицю;

– systemd для запуску та контролю серверного застосунку;

– SSH для віддаленого адміністрування Raspberry Pi;

– файлове логування для діагностики роботи системи.

3.1.3 Вимоги до режимів функціонування системи

Система повинна підтримувати такі режими роботи:

– режим користувацького анкетування;

– режим адміністративного керування;

– режим експорту даних у Google Sheets;

– режим журналювання подій;

– режим перевірки службового стану системи;

– режим watchdog-контролю;

– режим автозапуску після перезавантаження Raspberry Pi.

У разі недоступності Google Sheets система повинна продовжувати роботу з локальною базою даних SQLite без припинення анкетування.

У разі зупинки основного сервісу watchdog-механізм повинен фіксувати проблему та надсилати службове повідомлення адміністратору.

3.1.4 Вимоги по діагностуванню системи

Для діагностування системи повинні використовуватися засоби операційної системи Raspberry Pi OS, systemd, файлове логування, журнал подій системи та службові команди Telegram-бота.

Для діагностики адміністратор повинен мати можливість переглядати службовий стан системи, журнал подій і фрагменти лог-файлів через Telegram-бота.

3.1.5 Перспективи розвитку, проектування системи

Передбачається можливість подальшого розвитку системи за такими напрямами:

– розширення переліку полів основної анкети;

– додавання нових типів динамічних анкет;

– реалізація вебпанелі адміністратора;

– інтеграція з CRM-системами;

– додавання email-розсилок;

– реалізація резервного копіювання SQLite-бази;

– підключення додаткових зовнішніх сервісів аналітики;

– розширення системи ролей доступу;

– підтримка декількох факультетів або освітніх програм;

– формування звітів за періодами вступної кампанії;

– розгортання системи на VPS або хмарній платформі за потреби масштабування.

3.2 Показники призначення

Система повинна передбачати можливість масштабування за рахунок модульної структури програмного забезпечення та використання стандартних засобів Python, SQLite, Telegram Bot API і Google Sheets API.

3.2.1 Вимоги до надійності

Система повинна забезпечувати працездатність:

– при тривалій безперервній роботі Raspberry Pi;

– після перезавантаження серверної платформи;

– у разі короткочасної недоступності Google Sheets;

– у разі тимчасових мережевих збоїв;

– при повторному проходженні анкети користувачем;

– при зупинці та повторному запуску systemd-служби.

Для підвищення надійності повинні використовуватися:

– локальна база даних SQLite як основне сховище;

– Google Sheets як додаткове, а не основне сховище;

– systemd-служба з автоматичним перезапуском;

– watchdog-таймер для контролю активності сервісу;

– файлове логування;

Для захисту апаратури від стрибків напруги і комутаційних завад повинні застосовуватися мережні фільтри.

3.3 Вимоги до безпеки

Система повинна забезпечувати базовий рівень захисту даних і службових параметрів.

Конфіденційні параметри системи не повинні зберігатися безпосередньо в програмному коді.

Для зберігання службових параметрів повинен використовуватися .env-файл, який не повинен публікуватися у відкритому доступі.

Адміністративні функції повинні бути доступні лише користувачам, Telegram ID яких вказано в конфігурації системи. Доступ до службових функцій повинен обмежуватися роллю користувача.

Система повинна вести журнал подій, у якому фіксуються основні дії користувачів, створення заявок, зміна статусів, запуск розсилок, помилки інтеграції та службові повідомлення.

Під час підготовки демонстраційних матеріалів і скріншотів повинні використовуватися тестові дані або приховуватися персональні дані реальних користувачів.

3.3.1 Вимоги до експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи

Система повинна експлуатуватися на справному одноплатному комп’ютері Raspberry Pi з установленою Raspberry Pi OS або іншою сумісною Linux-системою.

Умови експлуатації апаратної частини повинні відповідати умовам нормальної роботи побутового та офісного комп’ютерного обладнання:

– температура повітря в межах від +10 °C до +35 °C;

– відносна вологість повітря при 25 °C у межах від 30 % до 80 %;

– стабільне електроживлення;

– наявність мережевого підключення до Інтернету;

– захист Raspberry Pi від механічних пошкоджень, пилу та вологи.

Періодичне технічне обслуговування і тестування технічних засобів повинні включати обслуговування і тестування всіх використовуваних засобів. На підставі результатів тестування технічних засобів повинні

проводитися аналіз причин виникнення виявлених дефектів і прийматися заходи по їх ліквідації.

3.4 Вимоги до захисту інформації від несанкціонованого доступу

Система повинна забезпечувати обмеження доступу до адміністративних функцій на основі Telegram ID користувача та його ролі.

Звичайний користувач повинен мати доступ лише до проходження анкети, перегляду доступних користувацьких функцій, отримання інформаційних повідомлень і проходження додаткових анкет.

Адміністратор повинен мати доступ до службових функцій керування системою, перегляду заявок, статистики, розсилок, журналу подій, логів і стану системи.

Для захисту службових ключів повинні використовуватися:

– .env-файл для токена бота та службових параметрів;

– окремий захищений каталог для JSON-файла Google service account;

– обмеження прав доступу до конфігураційних і службових файлів;

– недопущення публікації токенів і приватних ключів у репозиторіях або звітах.

Усі важливі дії адміністраторів і службові події повинні реєструватися в журналі подій.

3.4.1 Вимоги по стандартизації і уніфікації

Система повинна розроблятися з використанням стандартних засобів і технологій, що спрощують її супровід, перенесення та подальший розвиток.

Для реалізації програмної частини повинні використовуватися стандартні бібліотеки

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ

Комплект документації повинен складатись з:

– пояснювальну записку;

– структурну, функціональну схему системи;

– блок-схеми алгоритмів роботи;

– схему електричну принципову;

– результати тестування.

*Примітка: У комплект документації можуть вноситися зміни та доповнення в процесі розробки.

5 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання КРБ

№ етапу	Назва етапу виконання кваліфікаційної роботи	Термін виконання
1	Розробка технічного завдання	26.01 – 02.02
2	Робота над першим розділом «Аналіз технічного завдання»	03.02 – 15.02
3	Робота над другим розділом «Проєктна частинв»	20.04 – 25.04
4	Робота над третім розділом «Практична частина»	26.04 – 05.05
5	Робота над четвертим розділом «Безпека життєдіяльності, основи охорони праці»	07.05 – 25.05
6	Оформлення пояснювальної записки і графічного матеріалу	26.05 – 7.06
7	Перевірка на академічний плагіат, перевірка керівником та консультантами	8.06 – 14.06
8	Попередній захист кваліфікаційної роботи бакалавра	15.06 – 21.06
9	Захист кваліфікаційної роботи бакалавра	27.06

6 Додаткові умови виконання кваліфікаційної роботи

Під час виконання кваліфікаційної роботи у дане технічне завдання можуть вноситися зміни та доповнення.

Додаток Б

Лістинг програмного забезпечення

Лістинг А.1 – Головний файл запуску системи `main.py`

```
from __future__ import annotations

import asyncio
import logging

from aiogram import Bot, Dispatcher, F
from aiogram.fsm.storage.memory import MemoryStorage
from aiogram.types import Message
from apscheduler.schedulers.asyncio import AsyncIOScheduler

from bot.config import load_settings
from bot.database import Database
from bot.handlers.admin import build_admin_router
from bot.handlers.public import build_public_router
from bot.handlers.surveys import build_survey_router
from bot.services.broadcasts import BroadcastService
from bot.services.google_sheets import GoogleSheetsExporter
from bot.services.monitoring import (
    RuntimeStatus,
    network_monitor,
    notify_admins,
    setup_logging,
)

logger = logging.getLogger(__name__)


async def main() -> None:
    settings = load_settings()
    setup_logging(settings.log_file)
    logger.info("Starting F7 admissions bot")

    db = Database(settings.database_path)
    await db.init()
    await db.log_event("info", "system", "Bot process started")

    bot = Bot(token=settings.bot_token)
    sheets = GoogleSheetsExporter(settings)
    runtime_status = RuntimeStatus()

    dispatcher = Dispatcher(storage=MemoryStorage())
    dispatcher.include_router(
        build_admin_router(bot, settings, db, sheets, runtime_status)
    )
    dispatcher.include_router(build_public_router(bot, settings, db, sheets))
    dispatcher.include_router(build_survey_router(db))

    @dispatcher.message(F.text)
    async def fallback(message: Message) -> None:
        await message.answer(
            "Щоб заповнити анкету, натисни /apply. "
            "Для додаткової анкети використай /survey."
        )
```

```
    broadcast_service = BroadcastService(bot, db)
    scheduler = AsyncIOScheduler()
    scheduler.add_job(
        broadcast_service.send_pending,
        "interval",
        seconds=settings.scheduler_interval_seconds,
        id="scheduled-messages",
    )
    scheduler.start()

    asyncio.create_task(network_monitor(bot, settings, runtime_status))
    await notify_admins(
        bot,
        settings.admin_ids,
        "F7 admissions bot запущено і готовий до роботи.",
    )

    try:
        await dispatcher.start_polling(bot)
    except Exception as exc:
        logger.exception("Bot crashed")
        await db.log_event("error", "system", "Bot crashed",
details=str(exc))
        await notify_admins(
            bot,
            settings.admin_ids,
            "Критична помилка: бот аварійно завершив роботу. Перевірте
логи.",
        )
        raise
    finally:
        await db.log_event("info", "system", "Bot process stopped")
        logger.info("F7 admissions bot stopped")


if __name__ == "__main__":
    asyncio.run(main())
```

Лістинг А.2 – FSM-сценарій основної анкети `public.py`

```
from __future__ import annotations

import logging

from aiogram import Bot, F, Router
from aiogram.filters import Command, CommandStart
from aiogram.fsm.context import FSMContext
from aiogram.fsm.state import State, StatesGroup
from aiogram.types import Message

from bot.config import Settings
from bot.database import Database, Lead
from bot.keyboards import (
    channel_keyboard,
    course_keyboard,
    education_keyboard,
    interest_keyboard,
    phone_keyboard,
    remove_keyboard,
```

```
    start_keyboard,
    year_keyboard,
)
from bot.services.google_sheets import GoogleSheetsExporter

logger = logging.getLogger(__name__)


class LeadForm(StatesGroup):
    """FSM states for the standard admission form."""

    name = State()
    phone = State()
    email = State()
    education = State()
    course = State()
    year = State()
    interest = State()


def build_public_router(
    bot: Bot,
    settings: Settings,
    db: Database,
    sheets: GoogleSheetsExporter,
) -> Router:
    router = Router()

    @router.message(CommandStart())
    async def start(message: Message) -> None:
        await message.answer(
            "Вітаємо!\n\n"
            "Спеціальність F7 «Комп'ютерна інженерія».\n"
            "Натисни «Почати», щоб залишити заявку.",
            reply_markup=start_keyboard(),
        )

    @router.message(F.text == "Почати")
    @router.message(Command("apply"))
    async def apply(message: Message, state: FSMContext) -> None:
        await state.clear()
        await state.set_state(LeadForm.name)
        await message.answer("Як тебе звати?",
reply_markup=remove_keyboard())

    @router.message(Command("cancel"))
    async def cancel(message: Message, state: FSMContext) -> None:
        await state.clear()
        await message.answer(
            "Дію скасовано. Щоб почати анкету знову, натисни /apply."
        )

    @router.message(LeadForm.name)
    async def get_phone(message: Message, state: FSMContext) -> None:
        await state.update_data(name=(message.text or "").strip())
        await state.set_state(LeadForm.phone)
        await message.answer(
            "Надішли номер телефону або натисни кнопку нижче.",
            reply_markup=phone_keyboard(),
        )

    @router.message(LeadForm.phone)
    async def get_email(message: Message, state: FSMContext) -> None:
```

```
        phone = message.contact.phone_number if message.contact else
message.text
        await state.update_data(phone=(phone or "").strip())
        await state.set_state(LeadForm.email)
        await message.answer("Вкажи свій email:",
reply_markup=remove_keyboard())

    @router.message(LeadForm.email)
    async def get_education(message: Message, state: FSMContext) -> None:
        await state.update_data(email=(message.text or "").strip())
        await state.set_state(LeadForm.education)
        await message.answer(
            "Де ти зараз навчаєшся?",
            reply_markup=education_keyboard(),
        )

    @router.message(LeadForm.education)
    async def get_course_or_year(message: Message, state: FSMContext) ->
None:
        education = (message.text or "").strip()
        await state.update_data(education=education)

        if education.lower() == "коледж":
            await state.set_state(LeadForm.course)
            await message.answer("Обери курс навчання:",
reply_markup=course_keyboard())
        else:
            await state.update_data(course="-")
            await state.set_state(LeadForm.year)
            await message.answer("Обери рік вступу:",
reply_markup=year_keyboard())

    @router.message(LeadForm.course)
    async def get_year(message: Message, state: FSMContext) -> None:
        await state.update_data(course=(message.text or "").strip())
        await state.set_state(LeadForm.year)
        await message.answer("Обери рік вступу:",
reply_markup=year_keyboard())

    @router.message(LeadForm.year)
    async def get_interest(message: Message, state: FSMContext) -> None:
        await state.update_data(year=(message.text or "").strip())
        await state.set_state(LeadForm.interest)
        await message.answer(
            "Наскільки тебе цікавить IT-сфера?",
            reply_markup=interest_keyboard(),
        )

    @router.message(LeadForm.interest)
    async def finish_form(message: Message, state: FSMContext) -> None:
        await state.update_data(interest=(message.text or "").strip())
        data = await state.get_data()

        lead = Lead(
            telegram_id=message.from_user.id,
            username=message.from_user.username or "no_username",
            name=data["name"],
            phone=data["phone"],
            email=data["email"],
            education=data["education"],
            course=data["course"],
            year=data["year"],
            interest=data["interest"],
```

```
        )

        await db.upsert_lead(lead)
        await db.log_event(
            "info",
            "lead",
            f"Lead {lead.telegram_id} saved",
            created_by=lead.telegram_id,
        )

        try:
            sheets.append_lead(lead)
        except Exception as exc:
            logger.exception("Google Sheets export failed")
            await db.log_event(
                "error",
                "google_sheets",
                "Google Sheets export failed",
                details=str(exc),
            )

        await state.clear()
        await message.answer(
            "Дякуємо! Заявку збережено. "
            "Найближчим часом з вами зв'яжеться представник факультету.",
            reply_markup=channel_keyboard(settings),
        )

    return router
```

Лістинг А.3 – Інтеграція з Google Sheets `google_sheets.py`

```
from __future__ import annotations

from datetime import datetime
from pathlib import Path

import gspread
from google.oauth2.service_account import Credentials

from bot.config import Settings
from bot.database import Lead


HEADERS = [
    "date",
    "telegram_id",
    "username",
    "name",
    "phone",
    "email",
    "education",
    "course",
    "year",
    "interest",
    "status",
]


class GoogleSheetsExporter:
```

```
    """Optional Google Sheets integration.

    The bot continues working with SQLite even if Google credentials are
absent.
    """

    def __init__(self, settings: Settings) -> None:
        self.settings = settings

    @property
    def enabled(self) -> bool:
        return bool(
            self.settings.google_service_account_file
            and Path(self.settings.google_service_account_file).exists()
        )

    def append_lead(self, lead: Lead) -> None:
        if not self.enabled:
            return

        credentials = Credentials.from_service_account_file(
            self.settings.google_service_account_file,
            scopes=[
                "https://www.googleapis.com/auth/spreadsheets",
                "https://www.googleapis.com/auth/drive",
            ],
        )

        client = gspread.authorize(credentials)
        sheet = client.open(self.settings.spreadsheet_name).sheet1

        if not sheet.get_all_values():
            sheet.append_row(HEADERS)

        sheet.append_row(
            [
                datetime.now().strftime("%Y-%m-%d"),
                lead.telegram_id,
                lead.username,
                lead.name,
                lead.phone,
                lead.email,
                lead.education,
                lead.course,
                lead.year,
                lead.interest,
                lead.status,
            ]
        )
```

Лістинг А.4 – Логування та моніторинг мережі `monitoring.py`

```
from __future__ import annotations

import asyncio
import logging
from dataclasses import dataclass
from logging.handlers import RotatingFileHandler
from pathlib import Path
```

```
import aiohttp
from aiogram import Bot

from bot.config import Settings


logger = logging.getLogger(__name__)


@dataclass
class RuntimeStatus:
    network_online: bool = True
    last_network_error: str | None = None


def setup_logging(log_file: Path) -> None:
    """Configure file logs and console logs for systemd journal."""

    log_file.parent.mkdir(parents=True, exist_ok=True)
    formatter = logging.Formatter(
        "%(asctime)s %(levelname)s %(name)s: %(message)s",
        datefmt="%Y-%m-%d %H:%M:%S",
    )

    file_handler = RotatingFileHandler(
        log_file,
        maxBytes=1_000_000,
        backupCount=5,
        encoding="utf-8",
    )
    file_handler.setFormatter(formatter)

    console_handler = logging.StreamHandler()
    console_handler.setFormatter(formatter)

    root = logging.getLogger()
    root.setLevel(logging.INFO)
    root.handlers.clear()
    root.addHandler(file_handler)
    root.addHandler(console_handler)


def tail_log(log_file: Path, lines: int = 60) -> str:
    if not log_file.exists():
        return "Лог-файл ще не створено."
    content = log_file.read_text(
        encoding="utf-8",
        errors="replace",
    ).splitlines()
    return "\n".join(content[-lines:]) or "Лог-файл порожній."


async def notify_admins(bot: Bot, admin_ids: set[int], text: str) -> None:
    for admin_id in admin_ids:
        try:
            await bot.send_message(admin_id, text)
        except Exception:
            logger.exception("Could not notify admin %s", admin_id)


async def check_network(
    url: str,
    timeout_seconds: int = 10,
```

```
) -> tuple[bool, str | None]:
    try:
        timeout = aiohttp.ClientTimeout(total=timeout_seconds)
        async with aiohttp.ClientSession(timeout=timeout) as session:
            async with session.get(url) as response:
                if response.status < 500:
                    return True, None
                return False, f"HTTP {response.status}"
    except Exception as exc:
        return False, str(exc)


async def network_monitor(
    bot: Bot,
    settings: Settings,
    status: RuntimeStatus,
) -> None:
    """Background network availability monitor."""

    while True:
        online, error = await check_network(settings.healthcheck_url)

        if online and not status.network_online:
            status.network_online = True
            status.last_network_error = None
            logger.info("Network restored")
            await notify_admins(
                bot,
                settings.admin_ids,
                "Мережеве з'єднання відновлено.",
            )

        if not online and status.network_online:
            status.network_online = False
            status.last_network_error = error
            logger.warning("Network unavailable: %s", error)
            await notify_admins(
                bot,
                settings.admin_ids,
                f"Виявлено проблему з мережею: {error}",
            )

        await asyncio.sleep(settings.network_check_interval)
```