ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «Телеграм-бот для управління особистими фінансами з алгоритмами фільтрації та агрегації»

Ступінь вищої освіти – бакалавр
Спеціальність – 122 «Комп'ютерні науки»
Освітня програма «Комп'ютерні науки»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: здобувач 4 курсу
групи КН-22
Михайло НАДОПТА
Керівник: к.військ.н., с.н.с., доцент,
Володимир ТРОЦЬКО

Засвідчую, що кваліфікаційна робота оформлена відповідно до ДСТУ 3008:2015 та не містить запозичень з праць інших авторів без відповідних посилань.
Здобувач: _____
(підпис)

м. Київ – 2026 рік

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ЗАТВЕРДЖУЮ:
завідувач кафедри комп'ютерних наук
_____ССергій МІЧКІВСЬКИЙ
«___»_____20___р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Надопта Михайло Андрійович

Тема роботи	Телеграм-бот для управління особистими фінансами з алгоритмами фільтрації та агрегації
Номер та дата наказу про затвердження теми	№ 133-7 від 22 грудня 2025 року
Коротка постановка завдання	Розробка Телеграм-бот для управління особистими фінансами з алгоритмами фільтрації та агрегації
Посилання на джерела інформації (не більше п'яти найменувань, які рекомендує науковий курівник)	**SR Analytics.** How AI Personal Finance Tools Are Revolutionizing Money Management in 2026. 2025. URL: https://sranalytics.io/blog/ai-personal-finance/ (дата звернення: 02.04.2026). **Siddharth S.** Mastering Python for Data Science. 2nd ed. Birmingham : Packt Publishing, 2023. 450 p. **Telegram Bot API.** Official documentation for developers. 2026. URL: https://core.telegram.org/bots/api (дата звернення: 02.04.2026). **Kelliher C.** Quantitative Finance with Case Studies in Python: A Practical Guide to Investment Management, Trading and Financial Engineering. 2nd ed. Boca Raton : Chapman and Hall/CRC, 2025. 468 p. **McKinney W.** Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter. 3rd ed. Boston : O'Reilly Media, 2022. 578 p.
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має містити теоретичне, системотехнічне або експериментальне дослідження за темою роботи, яку слід розглядати як складне спеціалізоване завдання або практичну проблему в галузі комп'ютерних наук, яка характеризується комплексністю та невизначеністю умов і потребує застосування теорій і методів інформаційних технологій.

Дата видачі завдання 28 грудня 2025 р.

Керівник — Володимир ТРОЦЬКО

Здобувач освітнього ступеня бакалавра — Михайло НАДОПТА

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
	Підготовчий етап		
1.	Вибір напряму дослідження та керівника	01.12.2025 р.	Виконано
2.	Формування теми та призначення керівника	15.12.2025 р.	Виконано
3.	Затвердження теми кваліфікаційної роботи	22.12.2025 р.	Виконано
4.	Затвердження завдання на кваліфікаційну роботу	26.12.2025 р.	Виконано
	Основний етап		
5.	Розробка концепції кваліфікаційної роботи	23.01.2026 р.	Виконано
6.	Підбір та вивчення джерел інформації з напряму дослідження. Огляд існуючих аналогів	13.03.2026 р.	Виконано
7.	Затвердження розширеної постановки завдання. Підготовка та подання керівнику розділу 1 кваліфікаційної роботи	20.03.2026 р.	Виконано
8.	Проектування. Підготовка та подання керівнику розділу 2 кваліфікаційної роботи	27.03.2026 р.	Виконано
9.	Підготовка доповіді для експертизи стану виконання кваліфікаційної роботи (проміжний контроль)	30.03–04.04.2026 р.	Виконано
10.	Реалізація. Підготовка та подання керівнику розділу 3 кваліфікаційної роботи	06.04.2026 р.	Виконано
11.	Підготовка та подання керівнику першого варіанту всієї кваліфікаційної роботи	13.04.2026 р.	Виконано
12.	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника та представлення керівнику доопрацьованого варіанту кваліфікаційної роботи	20.04.2026 р.	Виконано
	Завершальний етап		
13.	Представлення рукопису для перевірки на плагіат	27.04–03.05.2026 р.	Виконано
14.	Підготовка презентації та доповіді на передзахист	04.05–10.05.2026 р.	Виконано
15.	Передзахист кваліфікаційної роботи	11.05–15.05.2026 р.	Виконано
16	Доопрацювання роботи за результатами предзахисту	18.05–05.06.2026 р.	Виконано
17	Експертиза роботи керівником та зовнішнім експертом	08.06–14.06.2026 р.	Виконано
18	Доопрацювання доповіді та презентації для захисту	08.06–14.06.2026 р.	Виконано
19.	Захист кваліфікаційної роботи	15.06–21.06.2026 р.	Виконано

Керівник Володимир ТРОЦЬКО

Здобувач освітнього ступеня бакалавра Михайло НАДОПТА

Надопта М.А. Телеграм-бот для управління особистими фінансами з алгоритмами фільтрації та агрегації.

Пояснювальна записка до кваліфікаційної роботи за спеціальністю 122 – Комп'ютерні науки (освітня програма – Комп'ютерні науки), бакалавр. – Вищий навчальний заклад «Університет економіки та права «КРОК», Навчально-науковий інститут інформаційно-комунікаційних технологій, кафедра комп'ютерних наук, Київ, 2026.

У кваліфікаційній роботі описано розробку Telegram-бота та веб-додатку для управління особистими фінансами з використанням алгоритмів фільтрації та агрегації фінансових даних. Розроблена система забезпечує автоматизований облік доходів і витрат, збереження фінансових операцій у базі даних, їх подальшу обробку, категоризацію та формування аналітичної інформації для користувача. Програмний продукт поєднує можливості Telegram-бота для швидкого додавання фінансових записів і веб-додатку для детального перегляду статистики, історії операцій, бюджетів та звітів. У системі реалізовано OCR-обробку чеків, планування бюджету, контроль лімітів за категоріями, підтримку повторюваних транзакцій, фінансову аналітику та експорт звітів у форматах CSV і PDF. Для реалізації програмного продукту використано Python, aiogram, FastAPI, SQLAlchemy Async, PostgreSQL, Tesseract OCR та Docker Compose, що забезпечує асинхронну обробку запитів, надійне збереження даних і можливість розгортання системи в контейнеризованому середовищі. Проведене тестування підтвердило коректність роботи основних функцій системи та її придатність до практичного використання.

Ключові слова: Telegram-бот, особисті фінанси, алгоритми фільтрації, алгоритми агрегації, веб-додаток, OCR, аналітика.

Табл. 10. Малюнок. 18. Бібліограф. 20.

Nadopta M.A. Telegram bot for personal finance management with filtering and aggregation algorithms.

Explanatory note of the qualification work in specialty 122 – Computer Science (educational program – Computer Science), Bachelor's degree. – Higher Educational Institution "University of Economics and Law "KROK", Educational and Scientific Institute of Information and Communication Technologies, Department of Computer Science, Kyiv, 2026.

The qualification work describes the development of a Telegram bot and a web application for managing personal finances using algorithms for filtering and aggregating financial data. The developed system provides automated accounting of income and expenses, saving financial transactions in the database, their further processing, categorization and generation of analytical information for the user. The software product combines the capabilities of a Telegram bot for quickly adding financial records and a web application for detailed viewing of statistics, transaction history, budgets and reports. The system implements OCR-processing of checks, budget planning, control of limits by categories, support for recurring transactions, financial analytics and export of reports in CSV and PDF formats. Python, aiogram, FastAPI, SQLAlchemy Async, PostgreSQL, Tesseract OCR and Docker Compose were used to implement the software product, which provides asynchronous processing of requests, reliable data storage and the ability to deploy the system in a containerized environment. The testing confirmed the correct operation of the main functions of the system and its suitability for practical use.

Key words: Telegram bot, personal finance, filtering algorithms, aggregation algorithms, Web App, OCR, analytics.

Table 10. Fig. 18. Bibliography. 20.

ЗМІСТ

ВСТУП

Актуальність теми. Актуальність теми зумовлена зростаючою потребою користувачів у зручному та швидкому контролі особистих фінансів. В умовах сучасного темпу життя люди щоденно здійснюють значну кількість фінансових операцій, що потребують систематизації та обліку. Відсутність структурованого підходу до ведення бюджету ускладнює контроль доходів і витрат, а також знижує ефективність фінансового планування. Багато користувачів стикаються з труднощами під час аналізу власних витрат через хаотичне збереження інформації або її повну відсутність. У результаті це негативно впливає на здатність приймати обґрунтовані фінансові рішення.

Сучасні програмні засоби для ведення особистих фінансів часто мають складний інтерфейс і надмірну кількість функцій. Через це користувачі можуть втрачати мотивацію до регулярного використання подібних систем, особливо якщо процес введення даних займає багато часу. Крім того, необхідність встановлення окремих застосунків не завжди є зручною для повсякденного використання. Це створює потребу у більш простих і доступних рішеннях, які б поєднували швидкість взаємодії та необхідний функціонал. Особливо актуальними стають інструменти, що дозволяють мінімізувати складність фінансового обліку та підвищити комфорт користування.

Одним із перспективних рішень є використання платформи Telegram для реалізації функцій управління особистими фінансами. Telegram забезпечує швидку взаємодію з користувачем без необхідності встановлення окремого програмного забезпечення, оскільки більшість користувачів уже активно використовують месенджер у повсякденному житті. Телеграм-боти дозволяють оперативно фіксувати доходи та витрати, отримувати статистику та взаємодіяти із системою у звичному середовищі. Це сприяє підвищенню регулярності використання системи та мотивації до ведення фінансового обліку. Таким чином, розробка телеграм-бота для управління особистими фінансами є актуальним і практично значущим напрямом дослідження.

Мета дослідження. Метою даного дослідження є розробка та обґрунтування ефективного підходу до створення телеграм-бота для управління особистими фінансами, який забезпечує автоматизований облік доходів і витрат, використовуючи алгоритми фільтрації та агрегації даних для підвищення точності аналізу, зручності користування та швидкості обробки інформації.

Для досягнення мети проєкту будуть виконані такі завдання:

1) розглянути існуючі аналоги;
2) проаналізувати існуючі телеграм-боти та програмні засоби для обліку особистих фінансів, визначити їхні переваги та недоліки;
3) визначити вимоги до програмного забезпечення системи;
4) дослідити методи та підходи до фільтрації й агрегації фінансових даних користувачів;
5) спроєктувати архітектуру програмного забезпечення Telegram-бота з використанням Python;
6) спроєктувати структуру бази даних для зберігання фінансової інформації;
7) провести тестування розробленого телеграм-бота та оцінити його ефективність.

Об’єктом дослідження є процеси обліку, фільтрації, агрегації та аналізу фінансових даних користувачів у системах управління особистими фінансами.

До цих процесів належить збір інформації про доходи та витрати, її структуризація, збереження, обробка та подальше використання для формування аналітичних висновків. Особлива увага приділяється автоматизації зазначених процесів та підвищенню ефективності роботи з фінансовими даними в умовах зростання їх обсягів.

Предметом дослідження є алгоритми фільтрації та агрегації фінансових даних, а також програмні засоби їх реалізації у вигляді Telegram-бота для управління особистими фінансами користувачів. У межах дослідження

розглядаються методи обробки даних, що дозволяють здійснювати вибірку інформації за заданими критеріями, обчислення узагальнених показників, формування статистики та аналітичних звітів. Також досліджуються підходи до реалізації цих алгоритмів із використанням сучасних технологій програмування та інтеграції з платформою Telegram.

Методи дослідження. Дослідження проводилося з використанням комплексу загальнонаукових і спеціальних методів, зокрема аналізу предметної області та існуючих аналогів, що дозволило визначити їхні підходи, переваги та недоліки. Для побудови структури системи застосовано методи системного аналізу та моделювання, що забезпечили проєктування архітектури програмного засобу та логіки його функціонування. У процесі розробки використано сучасні методи програмування із застосуванням мови Python, бібліотек та API платформи Telegram, а також алгоритмічні методи фільтрації та агрегації фінансових даних. Додатково застосовано методи тестування програмного забезпечення для перевірки коректності роботи системи та оцінки її ефективності.

Практичне значення. Практичне значення кваліфікаційної роботи полягає у можливості використання розробленого телеграм-бота для управління особистими фінансами з метою обліку, фільтрації та аналізу доходів і витрат користувачів. Запропоноване програмне рішення дозволяє оптимізувати процес фінансового планування, підвищити прозорість фінансової активності та сприяти формуванню раціональних фінансових рішень.

Структура та обсяг пояснювальної записки. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та списку посилань (20 найменувань). Пояснювальна записка містить 18 рисунків, 10 таблиць. Загальний обсяг пояснювальної записки складає 55 сторінок, основний зміст викладено на 1 сторінці.

РОЗДІЛ 1
ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ ТЕЛЕГРАМ-БОТУ ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ З АЛГОРИТМАМИ ФІЛЬТРАЦІЇ ТА АГРЕГАЦІЇ

1.1 Предметна область

Предметна область дослідження охоплює системи управління особистими фінансами користувачів, що забезпечують облік, зберігання, обробку та аналіз фінансових даних. До таких даних належать відомості про доходи, витрати, фінансові категорії, часові періоди та інші показники фінансової активності.

В умовах зростання обсягу фінансової інформації та необхідності оперативного контролю власного бюджету особливої актуальності набувають програмні рішення, що дозволяють автоматизувати процеси збору та аналізу фінансових даних.

Телеграм-боти як інструменти взаємодії з користувачами забезпечують зручний доступ до функціоналу без встановлення додаткового програмного забезпечення, використовуючи можливості месенджера Telegram.

У межах предметної області розглядаються процеси фільтрації фінансових даних за різними критеріями (категорія, дата, тип операції) та їх агрегації з метою отримання узагальнених показників, статистики та аналічної інформації. Важливою складовою є застосування алгоритмів обробки даних для формування рекомендацій, що сприяють підвищенню ефективності фінансового планування користувачів.

Сьогодні системи управління особистими фінансами набувають особливої популярності серед широкого кола користувачів. Це зумовлено як зростанням фінансової грамотності населення, так і розвитком цифрових технологій, що спрощують доступ до інструментів обліку та аналізу витрат.

Все більше людей прагнуть контролювати свої доходи та витрати в режимі реального часу, планувати бюджет та оптимізувати фінансові звички.

Особливу роль у цьому процесі відіграють мобільні та месенджер-орієнтовані рішення, зокрема телеграм-боти, які дозволяють швидко та зручно взаємодіяти з системою без необхідності встановлення окремих додатків. Завдяки простоті використання, доступності та інтеграції з повсякденними засобами комунікації, такі рішення стають дедалі більш затребуваними.

Крім того, сучасні користувачі очікують від подібних систем не лише збереження даних, але й надання аналітики, візуалізації та рекомендацій на основі обробки фінансової інформації. Це сприяє активному розвитку інтелектуальних функцій, таких як автоматичне визначення категорій витрат, прогнозування бюджету та персоналізовані поради щодо фінансового планування.

Таким чином, предметна область поєднує питання обліку особистих фінансів, алгоритмічної обробки даних та розробки програмних засобів у вигляді телеграм-бота для забезпечення зручного та ефективного управління фінансовими ресурсами.

1.2 Огляд аналогів

На українському ринку існують кілька Телеграм-ботів для управління особистими фінансами з алгоритмами фільтрації та агрегації:

Telegram Budget Bot (рис. 1.1) є одним із найбільш затребуваних сервісів для управління особистими фінансами, що функціонує на базі месенджера Telegram. Основною перевагою даного рішення є простота використання та швидкий доступ до функціоналу без необхідності встановлення окремого програмного забезпечення.

Сервіс підтримує введення фінансових даних як у текстовому форматі, так і за допомогою голосових повідомлень, що значно спрощує процес фіксації витрат і доходів у повсякденному житті користувача. Завдяки вбудованим

алгоритмам обробки даних, бот автоматично визначає категорію витрат, що дозволяє зменшити кількість ручних дій та підвищити точність обліку.

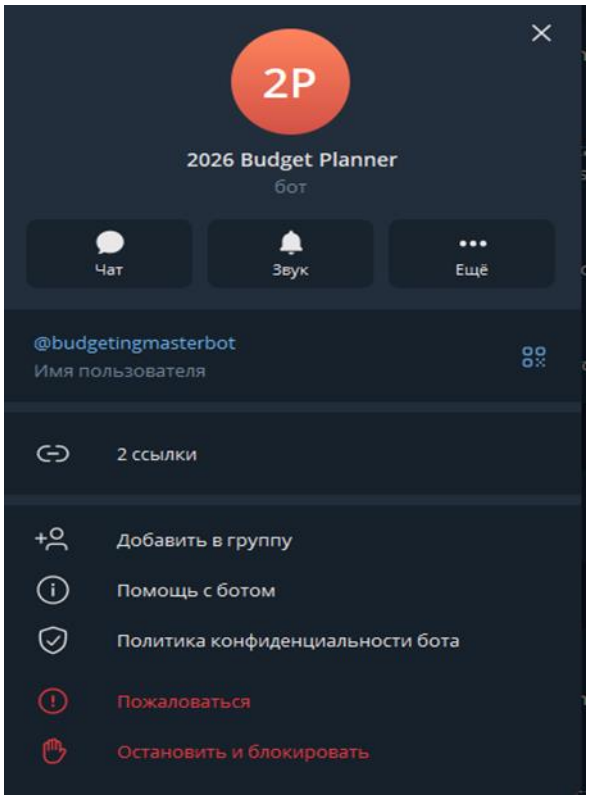

Рисунок 1.1 – Телеграм-Бот "Telegram Budget Bot" [2]

Джерело: розроблено автором

Крім того, Telegram Budget Bot забезпечує підтримку мультивалютності, що є особливо важливим для користувачів, які здійснюють операції в різних валютах або працюють із міжнародними фінансовими потоками. Система також надає можливість формування звітів за різними періодами часу, що дозволяє аналізувати фінансову активність, відстежувати тенденції витрат та приймати обґрунтовані рішення щодо планування бюджету.

Функціональні можливості сервісу поділяються на безкоштовну версію та преміум-план. Безкоштовна версія надає базовий набір інструментів для обліку фінансів, тоді як преміум-підписка відкриває доступ до розширеної аналітики, детальних звітів, додаткових категорій та персоналізованих рекомендацій [2].

Telexpense Bot (рис. 1.2) – це Telegram-бот (@telexpense_bot), призначений для автоматизованого обліку особистих фінансів із використанням інтеграції з Google Sheets. Основною особливістю даного сервісу є зберігання всіх фінансових даних безпосередньо у таблиці користувача, що забезпечує гнучкість у роботі з інформацією та можливість її подальшого аналізу за допомогою інструментів Google Sheets.

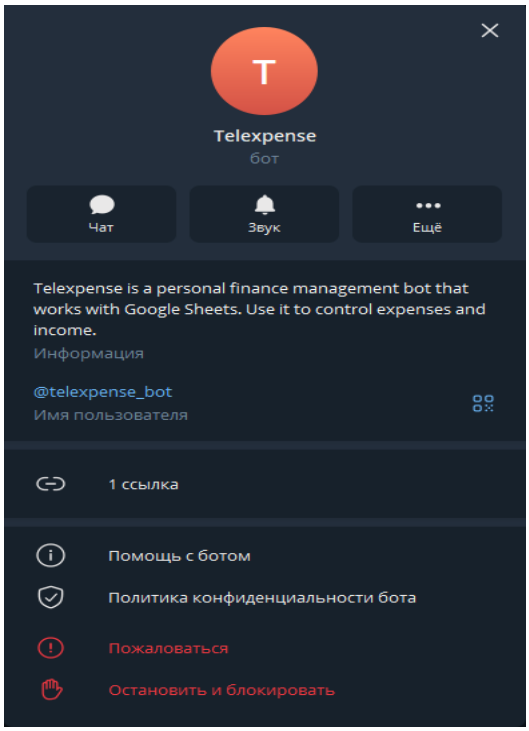

Рисунок 1.2 – Телеграм-Бот"Telexpense" [3]

Джерело: розроблено автором

Бот дозволяє додавати записи про витрати, доходи та перекази між рахунками безпосередньо в інтерфейсі месенджера Telegram, що значно спрощує процес обліку фінансів і робить його більш швидким та зручним.

Введені дані автоматично синхронізуються з таблицею, де вони структуруються за відповідними полями, такими як дата, сума, категорія та тип операції.

Однією з переваг Telexpense Bot є можливість налаштування структури таблиці відповідно до потреб користувача, включаючи створення власних

категорій, фільтрів та формул для аналізу даних. Це робить сервіс особливо корисним для користувачів, які потребують більш гнучкого та кастомізованого підходу до управління фінансами.

Крім того, використання Google Sheets як сховища даних забезпечує доступ до інформації з будь-якого пристрою, а також дозволяє використовувати вбудовані засоби візуалізації, такі як графіки та діаграми, для більш глибокого аналізу фінансової активності [3].

Finny (рис. 1.3) – це Telegram-бот, що функціонує як розумний фінансовий помічник на базі технологій штучного інтелекту [1]. На відміну від класичних систем обліку, даний сервіс орієнтований не лише на фіксацію фінансових операцій, але й на їх глибокий аналіз із подальшим формуванням рекомендацій для користувача.

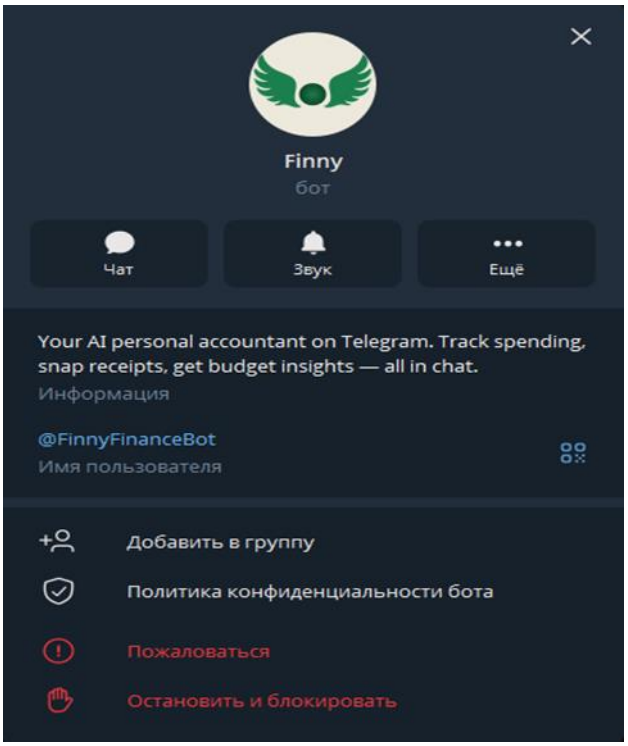

Рисунок 1.3 – Телеграм-Бот “ Finny ” [1]

Джерело: розроблено автором

Однією з ключових функцій є розумний облік фінансів. Бот автоматично обробляє інформацію про доходи та витрати, визначає закономірності у фінансовій поведінці користувача та на основі цього надає персоналізовані поради щодо оптимізації бюджету. Це дозволяє не лише відстежувати фінансовий стан, а й активно впливати на нього.

Важливим аспектом роботи Finny є орієнтація на економію коштів. Система аналізує структуру витрат і допомагає виявити приховані резерви бюджету, завдяки чому користувач може заощадити до 10% коштів без суттєвого зниження рівня комфорту. Такий підхід базується на виявленні неефективних або зайвих витрат.

Крім того, бот виконує функцію фінансового наставника, надаючи рекомендації щодо управління борговими зобов'язаннями, зокрема зниження платежів за кредитами та іпотекою. Це може включати поради щодо оптимізації графіків платежів, перегляду умов кредитування або підвищення ефективності розподілу доходів.

Недоліками Telegram Budget Bot є обмежена кастомізація безкоштовної версії, часткова доступність аналітичних функцій лише у premium-підписці та відсутність повноцінного WebApp для детальної роботи з даними. Недоліками Telexpense Bot є залежність від Google Sheets, що ускладнює роботу з великими обсягами даних, а також обмежені можливості бюджетування та автоматизованої аналітики. Finny має закриту реалізацію алгоритмів аналізу даних, що не дозволяє користувачу повністю контролювати логіку формування рекомендацій, а також орієнтований переважно на консультаційний підхід, а не на детальний персоналізований облік фінансів.

Таблиця 1.1 – Порівняльна характеристика аналогів та розроблюваного програмного продукту

Критерій	Telegram Budget Bot	Telexpense Bot	Finny	Розроблюваний продукт

Швидке введення операцій	Так	Так	Так	Так
OCR-обробка чеків	Обмежено або відсутня	Ні	Частково	Так
WebApp для детальної аналітики	Ні	Ні, основа — Google Sheets	Обмежено	Так
Бюджети та категорійні ліміти	Частково	Обмежено	Так	Так
Рекурентні операції	Обмежено	Ні	Частково	Так
Експорт CSV/PDF	Частково	Через Google Sheets	Обмежено	Так
Відкритість і контроль логіки	Обмежено	Частково	Ні	Так, власна реалізація

Порівняння показує, що існуючі рішення забезпечують базові функції фінансового обліку, однак мають обмеження щодо гнучкої аналітики, OCR-обробки чеків, контролю рекурентних операцій та відкритості логіки обробки даних. Тому розроблюваний продукт орієнтований на поєднання швидкого Telegram-інтерфейсу з повноцінним WebApp, власною базою даних, алгоритмами фільтрації, агрегації та розширеними сценаріями фінансового планування.

1.3 Визначення потенційних конкурентних переваг розробки

Проаналізувавши потенційних конкурентів, можна визначити такі потенційні переваги розроблюваного Telegram-бота та веб-застосунку:

1. Розроблюваний телеграм-бот для управління особистими фінансами має низку потенційних конкурентних переваг порівняно з існуючими аналогами, представленими на ринку програмних засобів фінансового обліку.

2. Однією з ключових переваг є використання платформи Telegram як основного інтерфейсу взаємодії з користувачем. Це забезпечує швидкий доступ до функціоналу без необхідності встановлення окремого програмного

забезпечення, знижує поріг входу для користувачів та підвищує рівень зручності використання сервісу.

3. Важливою конкурентною перевагою є реалізація алгоритмів фільтрації та агрегації фінансових даних, які дозволяють користувачам отримувати зведену аналітичну інформацію за різними критеріями: періодом часу, категоріями витрат і доходів, типами операцій. Це забезпечує наочний аналіз фінансової активності та сприяє кращому розумінню структури власного бюджету.

4. Ще однією перевагою є простота та інтуїтивність користувацького інтерфейсу. Командний формат взаємодії через телеграм-бот дозволяє швидко вносити дані про доходи та витрати у текстовому вигляді, що скорочує час на ведення фінансового обліку та підвищує мотивацію користувачів до регулярного використання системи.

5. Розробка також передбачає можливість масштабування та розширення функціоналу, зокрема додавання нових алгоритмів аналізу даних, формування персоналізованих рекомендацій, підтримки мультивалютності та інтеграції з іншими сервісами. Це забезпечує гнучкість програмного рішення та його адаптацію до індивідуальних потреб користувачів.

6. Крім того, конкурентною перевагою є локальна спрямованість та орієнтація на реальні потреби користувачів, що дозволяє створити ефективний інструмент контролю особистих фінансів без перевантаження зайвими функціями, характерними для складних фінансових застосунків.

7. Таким чином, сукупність зазначених переваг робить розроблюваний телеграм-бот конкурентоспроможним рішенням у сфері управління особистими фінансами та обґрунтовує доцільність його створення і впровадження.

1.4 Постановка задачі

Постановка задачі кваліфікаційної роботи полягає у розробці програмного засобу у вигляді Telegram-бота для управління особистими фінансами користувачів із використанням алгоритмів фільтрації та агрегації даних. Основною метою є забезпечення зручного, швидкого та ефективного обліку доходів і витрат, а також автоматизованого аналізу фінансової активності користувачів.

Вхідними даними системи є повідомлення користувача про фінансові операції, зокрема витрати або доходи, що містять інформацію про суму та опис транзакції. Вихідними даними є підтвердження успішного додавання транзакції, а також зведена статистика доходів і витрат за обраними періодами часу.

У процесі виконання роботи необхідно розробити Telegram-бот, який дозволяє користувачам вносити інформацію про фінансові операції (доходи та витрати), зберігати ці дані у базі даних, здійснювати їх обробку та надавати аналітичні звіти у зручному для сприйняття вигляді.

Функціональні вимоги до системи передбачають, що користувач повинен мати змогу вводити інформацію про транзакції (сума, опис, категорія), а також отримувати зведену інформацію про фінансову активність за день, тиждень, місяць або довільно обраний період.

Нефункціональні вимоги включають забезпечення надійності та безпеки збереження даних користувачів, високої продуктивності обробки запитів, можливості масштабування системи при збільшенні кількості користувачів, а також зручності використання та доступності сервісу.

Для досягнення поставленої мети необхідно вирішити такі основні задачі:

1) проаналізувати предметну область управління особистими фінансами та існуючі Telegram-боти й програмні засоби фінансового обліку, визначити їх переваги та недоліки;
2) сформувати функціональні та нефункціональні вимоги до Telegram-бота;
3) спроєктувати загальну архітектуру програмного забезпечення та логіку взаємодії користувача з системою;
4) розробити структуру бази даних для зберігання інформації про користувачів, доходи, витрати та категорії фінансових операцій;
5) реалізувати алгоритми фільтрації фінансових даних за різними критеріями (дата, категорія, тип операції);
6) реалізувати алгоритми агрегації даних для формування узагальнених показників, статистики та звітів за вибраними періодами;
7) реалізувати базовий функціонал Telegram-бота засобами мови програмування Python із використанням відповідних бібліотек та API Telegram;
8) провести тестування розробленого Telegram-бота та оцінити коректність його роботи.

Результатом виконання поставлених задач має стати працездатний Telegram-бот для управління особистими фінансами, який забезпечує зручний облік доходів і витрат, аналіз фінансової активності користувачів та підтримує можливість подальшого розширення функціоналу.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи було розглянуто предметну область управління особистими фінансами користувачів та визначено основні особливості використання Telegram-ботів як програмних засобів для обліку та аналізу фінансової інформації. Обґрунтовано актуальність розробки такого програмного рішення в умовах зростання потреби в зручному, швидкому та

доступному контролі доходів і витрат. Встановлено, що сучасні користувачі потребують інструментів, які дозволяють не лише зберігати фінансові записи, але й оперативно отримувати узагальнену інформацію про власну фінансову активність.

Проведено аналіз існуючих Telegram-ботів і програмних продуктів для ведення особистих фінансів. Досліджено їхні функціональні можливості, переваги та недоліки, а також особливості реалізації окремих функцій. На основі аналізу встановлено, що більшість існуючих рішень мають або надмірно складний функціонал, або недостатній рівень аналітичної обробки даних, або обмежені можливості гнучкого налаштування під потреби конкретного користувача. Це створює передумови для розробки більш зручного, адаптивного та функціонального програмного продукту.

У першому розділі також визначено потенційні конкурентні переваги розроблюваного Telegram-бота. До ключових переваг віднесено використання платформи Telegram як звичного середовища взаємодії для користувача, що знижує поріг входу та спрощує регулярне використання системи. Важливими аспектами є застосування алгоритмів фільтрації та агрегації фінансових даних, можливість швидкого доступу до статистики, підтримка зручного введення операцій та орієнтація на подальше розширення функціоналу. Крім того, передбачено можливість масштабування системи та інтеграції додаткових модулів у майбутньому.

Окрему увагу приділено формуванню постановки задачі кваліфікаційної роботи. Було визначено основну мету розробки та сформовано перелік завдань, необхідних для її досягнення. Окреслено функціональні та нефункціональні вимоги до програмного забезпечення, зокрема вимоги до зручності використання, надійності збереження даних, продуктивності обробки запитів та можливості подальшого розвитку системи. Також визначено вимоги до структури бази даних і механізмів обробки фінансової інформації, що є необхідною основою для подальшого проєктування програмного продукту.

У процесі дослідження встановлено, що використання Telegram-ботів для фінансового обліку є перспективним напрямом розвитку програмних рішень. Поєднання простоти використання, доступності месенджера та автоматизованої обробки даних дозволяє підвищити ефективність контролю особистих фінансів. Водночас використання алгоритмів фільтрації та агрегації забезпечує формування узагальненої статистики, аналітичних звітів і більш наочного представлення структури доходів та витрат. Це сприяє прийняттю більш обґрунтованих фінансових рішень користувачами.

Отримані результати першого розділу є теоретичною та методичною основою для подальшого виконання кваліфікаційної роботи. Вони створюють підґрунтя для проєктування архітектури системи, розробки структури бази даних, визначення основних користувацьких сценаріїв та реалізації алгоритмів фільтрації й агрегації фінансових даних. На основі проведеного аналізу буде реалізовано програмний продукт у вигляді Telegram-бота та веб-додатку для управління особистими фінансами. Практична реалізація, тестування та опис використання системи будуть розглянуті у наступних розділах роботи.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТЕЛЕГРАМ-БОТУ ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ

2.1 Моделювання поведінки продукту

Проєктування поведінки програмного продукту виконано з урахуванням того, що система має два взаємопов'язані канали взаємодії: Telegram-бот для швидких операцій та Telegram WebApp для детальної роботи з фінансовими даними. Такий підхід дозволяє поєднати простоту введення даних у чаті з можливостями повноцінного графічного інтерфейсу для аналітики, бюджетування та експорту звітів.

Основним актором системи є користувач Telegram, який запускає бота, додає доходи й витрати, переглядає статистику та керує бюджетом. У системі також працюють автоматизовані процеси: перевірка платежів, нагадування та контроль бюджетних лімітів.

Поведінкова модель системи будується навколо життєвого циклу фінансового запису. Запис може бути створений вручну через меню, швидким текстовим повідомленням або після OCR-розпізнавання чека. Після створення запис проходить валідацію, зберігається у базі даних, бере участь у фільтрації та агрегації, відображається в історії операцій, аналітиці, бюджетних розрахунках і експортованих звітах.

Діаграма варіантів використання відображає межі програмного продукту та розподіл функцій між чат-інтерфейсом і WebApp. Telegram-бот орієнтований на швидкі дії: додавання запису, отримання короткої статистики та обробку чеків.

WebApp використовується для операцій, які потребують ширшого екрана: редагування записів, перегляд дашборду, робота з бюджетами, лімітами, аудитом та експортом.

Рисунок 2.1 – Діаграма варіантів використання системи управління особистими фінансами

Джерело: розроблено автором

Процес додавання фінансової операції починається з отримання повідомлення або натискання кнопки. Далі система визначає тип сценарію, виділяє суму, опис, дату та категорію, перевіряє коректність даних і створює запис. Якщо введені дані неповні або помилкові, користувач отримує пояснення та можливість повторити введення. Такий механізм знижує ризик потрапляння некоректних даних до фінансової статистики.

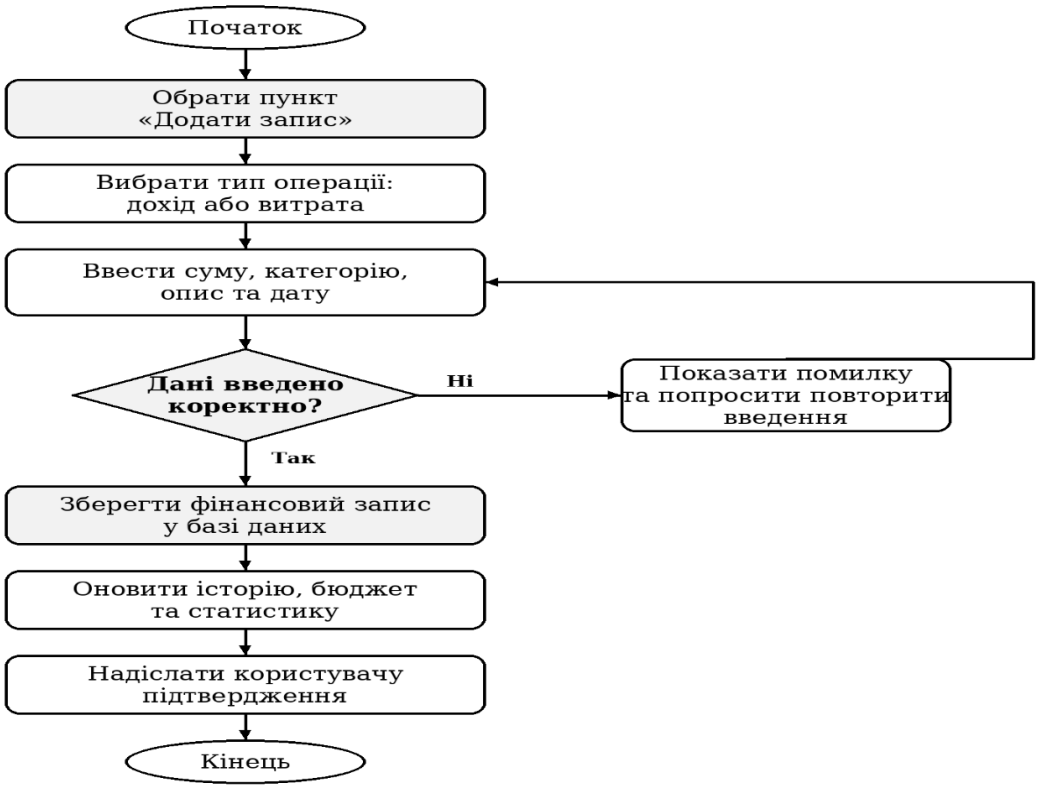

Рисунок 2.2 – Діаграма діяльності процесу додавання фінансової операції

Джерело: розроблено автором

Окремим сценарієм є OCR-обробка чека. Користувач надсилає фотографію, бот завантажує файл через Telegram API, виконує попередню обробку зображення, передає його до OCR-сервісу та аналізує розпізнаний текст. Система шукає суму, назву магазину, дату та можливі позиції чека. Після цього користувачу пропонується підтвердити результат або перейти до ручного редагування.

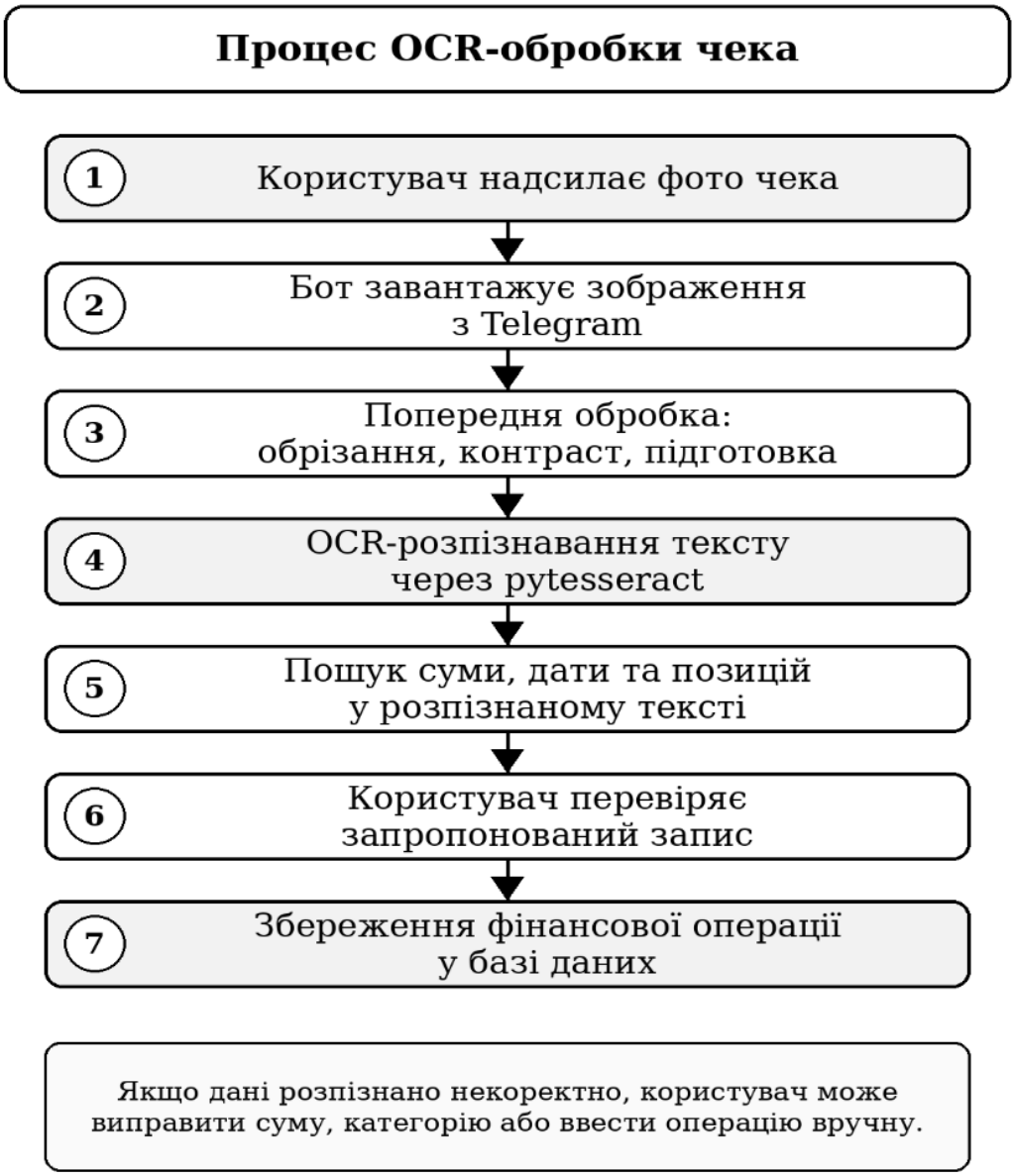

Рисунок 2.3 – Діаграма процесу OCR-обробки чека

Джерело: розроблено автором

Формування аналітичного звіту виконується на основі фільтрації та агрегації збережених записів. Користувач задає період або відкриває відповідний розділ WebApp. Система отримує записи з бази даних, застосовує критерії відбору, групує операції за категоріями, типами та датами, після чого обчислює доходи, витрати, баланс, середній і максимальний розмір витрат, частки категорій та ступінь використання бюджетних лімітів.

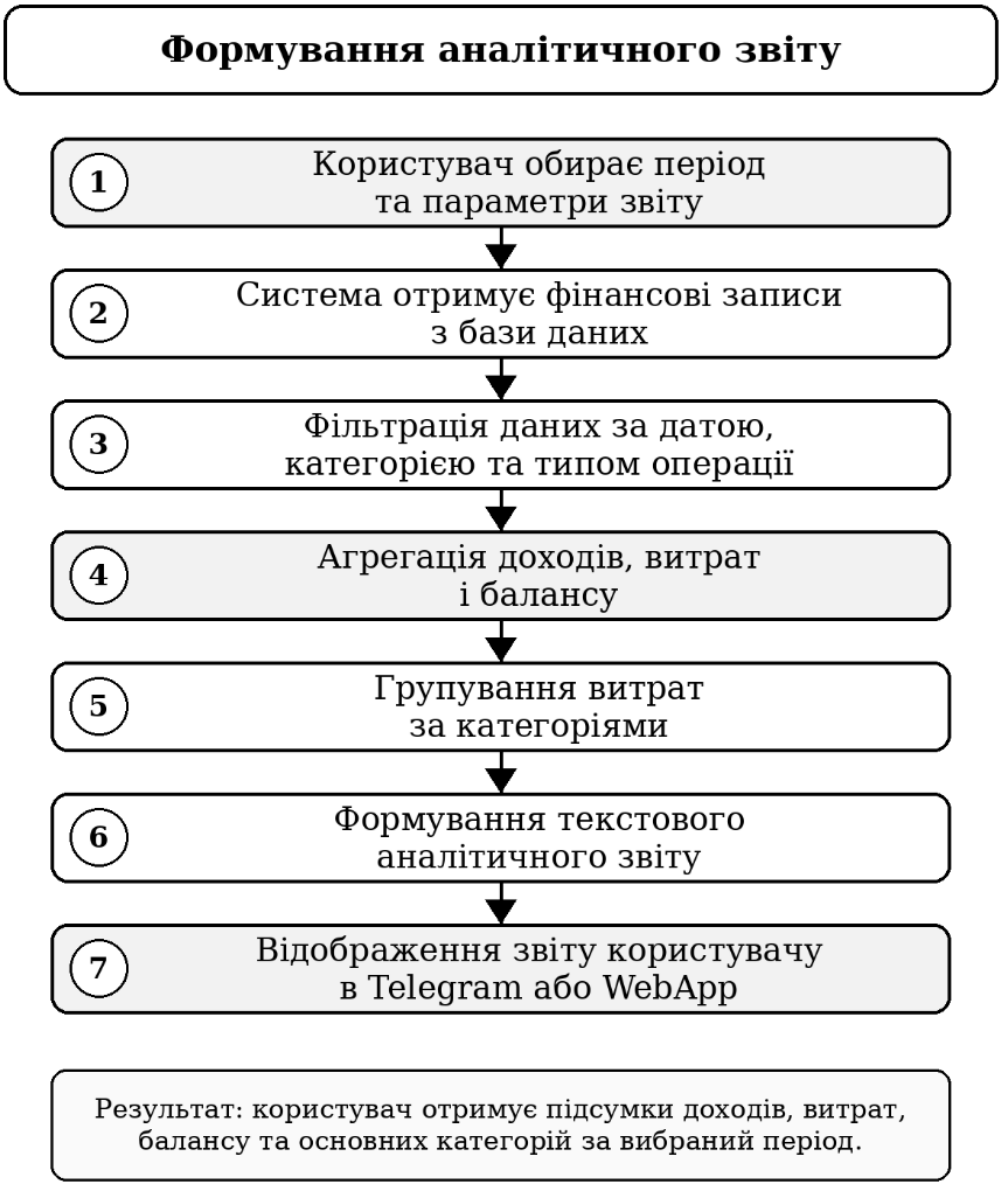

Рисунок 2.4 – Діаграма процесу формування аналітичного звіту

Джерело: розроблено автором

У таблиці 2.1 наведено основні поведінкові сценарії програмного продукту.

Таблиця 2.1 – Основні поведінкові сценарії програмного продукту

Сценарій	*Ініціатор*	*Вхідні дані*	*Основний результат*

Запуск бота	Користувач	Команда /start	Створення або отримання профілю, показ головного меню
Швидке додавання витрати	Користувач	Текстове повідомлення із сумою та описом	Створення запису після валідації та підтвердження
Ручне додавання операції	Користувач	Тип, категорія, сума, дата, опис	Збереження доходу або витрати у базі даних
OCR-обробка чека	Користувач	Фотографія чека	Розпізнавання суми та пропозиція збереження операції
Перегляд історії	Користувач	Період, категорія, тип операції	Список відфільтрованих записів
Формування аналітики	Користувач / WebApp	Період та параметри фільтрації	Звіт із доходами, витратами, балансом і категоріями
Контроль бюджету	Система	Фактичні витрати та задані ліміти	Визначення відсотка використання бюджету
Рекурентні операції	Фоновий процес	Активні шаблони платежів	Автоматичне створення запису або нагадування

2.2 Моделювання архітектури та структури продукту

Архітектура програмного продукту побудована за багаторівневим принципом і складається з інтерфейсного, прикладного, сервісного, рівня доступу до даних, рівня збереження та інфраструктурного рівня. Така структура відповідає вимогам до програмного продукту, оскільки забезпечує розділення відповідальності, підтримуваність коду, можливість тестування окремих компонентів та подальше масштабування.

Інтерфейсний рівень представлено Telegram-ботом і Telegram WebApp. Прикладний рівень містить обробники aiogram для Telegram-сценаріїв та маршрути FastAPI для веб-застосунку. Сервісний рівень реалізує бізнес-логіку: створення фінансових записів, фільтрацію, агрегацію, бюджетні обчислення, OCR-розпізнавання, категоризацію та роботу з рекурентними операціями. Рівень доступу до даних реалізовано через SQLAlchemy Async, а рівень збереження – через PostgreSQL 16.

У Docker Compose середовищі система поділена на три основні сервіси: db, finance-bot і finance-web. Сервіс db відповідає за збереження даних у PostgreSQL. Сервіс finance-bot запускає Telegram-бота та фонові цикли. Сервіс finance-web запускає FastAPI-застосунок, який обслуговує WebApp та REST API. Обидва прикладні сервіси працюють з однією базою даних, тому зміни, виконані через один інтерфейс, одразу доступні в іншому.

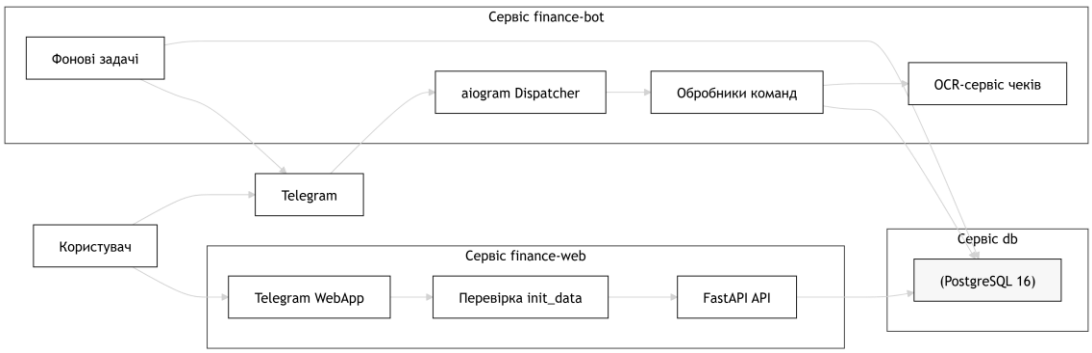

Рисунок 2.5 – Загальна архітектура телеграм-боту та веб-застосунку

Джерело: розроблено автором

Структура програмних модулів побудована відповідно до принципу розділення відповідальності. Файл app/bot.py є точкою входу Telegram-бота. Пакет app/handlers містить маршрути обробки команд, меню, швидкого введення, ручного додавання, OCR і звітів. Пакет app/services містить прикладну логіку, а app/repositories забезпечує роботу з базою даних. Веб-

частина зосереджена в app/web і включає маршрути API, перевірку Telegram WebApp init_data та статичні файли інтерфейсу.

Таблиця 2.2 – Рівні архітектури, їхні компоненти та призначення

Рівень архітектури	*Компоненти*	*Призначення*
Інтерфейсний	Telegram-бот, Telegram WebApp	Введення даних, навігація, перегляд статистики та дашборду
Прикладний	aiogram handlers, FastAPI routes	Обробка команд, повідомлень, API-запитів і користувацьких сценаріїв
Сервісний	record_service, aggregation_service, ocr_service, category_classifier	Бізнес-логіка, OCR, категоризація, агрегація, бюджетні розрахунки
Доступ до даних	SQLAlchemy Async, repositories	Асинхронна робота з моделями та запитами до БД
Збереження	PostgreSQL 16	Зберігання користувачів, записів, категорій, бюджетів та журналу аудиту
Інфраструктура	Docker Compose, .env, volume pgdata	Запуск сервісів, конфігурація, мережа та постійне збереження даних

Модель даних орієнтована на персоналізований облік фінансів. Центральною сутністю є Record – фінансова операція користувача. Вона пов’язана з User та Category. Додаткові сутності BudgetPlan, CategoryLimit і RecurringRecord забезпечують бюджетування, контроль лімітів і автоматизацію регулярних платежів. UserSettings зберігає персональні налаштування, а AuditLog фіксує важливі дії в системі.

Структура програмних модулів системи

Основний пакет застосунку: app/

Група модулів	Склад програмних модулів
Telegram-бот	bot.py; handlers/common.py; routes_menu.py; routes_manual_add.py; routes_quick_expense.py; routes_ocr.py; routes_reports.py.
WebApp та API	web_main.py; api/webapp/init; api/webapp/dashboard; api/webapp/records; api/webapp/budget; api/webapp/export; перевірка Telegram init_data.
Сервісні модулі	OCR-сервіс чеків; класифікатор категорій; аналітика та агрегація; експорт CSV/PDF; фонові задачі; нагадування.
Рівень даних	PostgreSQL 16; SQLAlchemy Async; моделі бази даних; асинхронні сесії; операції з фінансовими записами.
Розгортання	Docker Compose; service: db; service: finance-bot; service: finance-web; спільна мережа компонентів системи.

Рисунок 2.6 – Структура програмних модулів системи

Джерело: розроблено автором

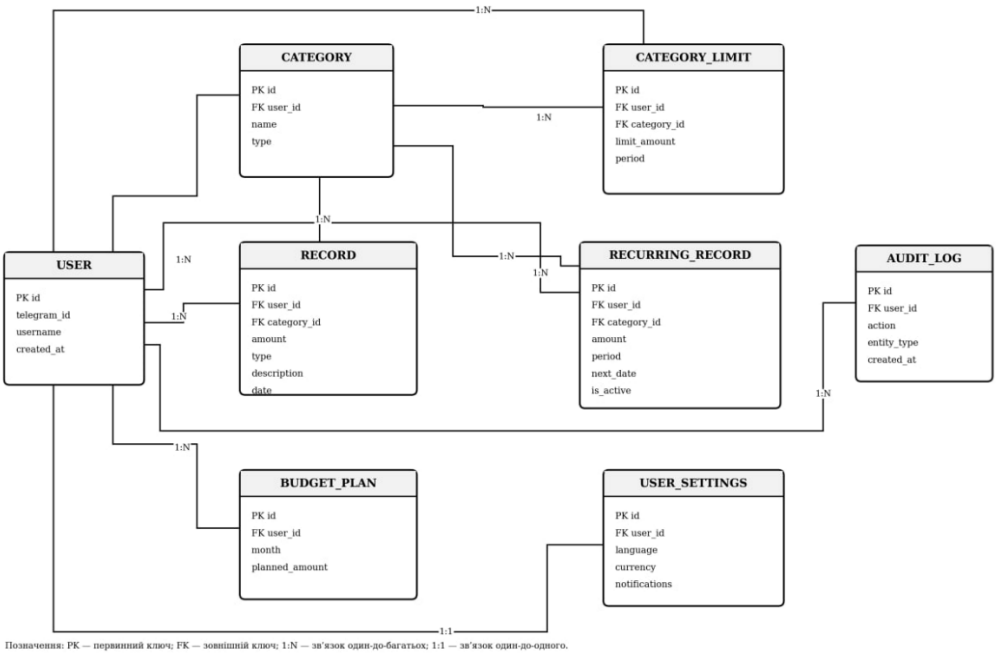

Рисунок 2.7 – ER-діаграма бази даних системи управління особистими фінансами

Джерело: розроблено автором

Таблиця 2.3 – Основні сутності бази даних програмного продукту

Сутність	*Основні поля*	*Призначення*
User	id, telegram_id, username, created_at	Ідентифікація користувача Telegram та зв’язок з його даними
Category	id, user_id, name, type	Групування доходів і витрат за користувацькими категоріями
Record	id, user_id, category_id, amount, type, description, date	Збереження фінансової операції
BudgetPlan	id, user_id, period_start, period_end, planned_income, planned_expense	Збереження плану доходів і витрат на період
CategoryLimit	id, user_id, category_id, limit_amount, period	Контроль витрат за окремими категоріями
RecurringRecord	id, user_id, category_id, amount, period, next_date, is_active	Опис регулярних платежів і доходів
UserSettings	id, user_id, language, currency, notifications	Персональні налаштування інтерфейсу та повідомлень
AuditLog	id, user_id, action, entity_type, created_at	Фіксація важливих змін і дій користувача

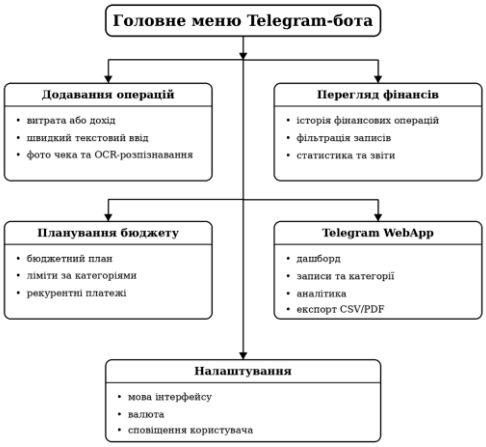

Рисунок 2.8 – Структура інтерфейсу Telegram-бота та веб-застосунку

Джерело: розроблено автором

2.3 Опис алгоритмів та математичних моделей

У програмному продукті використано алгоритми швидкого розпізнавання фінансової операції з тексту, OCR-обробки чеків, фільтрації записів, агрегації фінансових показників, контролю бюджетних лімітів та обробки рекурентних операцій. Вони забезпечують не лише збереження даних, а й автоматизовану обробку фінансової інформації.

Алгоритм швидкого введення працює з короткими повідомленнями користувача. На першому кроці виконується нормалізація тексту: видалення зайвих пробілів, приведення суми до числового формату та виділення ключових слів. На другому кроці система визначає суму, опис і потенційну категорію. Якщо категорію визначити неможливо, користувачу пропонується ручний вибір.

Категоризація фінансових операцій реалізована комбінованим підходом. Спочатку система аналізує історію операцій конкретного користувача та намагається знайти подібні записи. Далі виконується нечітке порівняння текстового опису з наявними категоріями та ключовими словами. Якщо попередні етапи не дають достатньо надійного результату, може застосовуватися TF-IDF-представлення тексту для визначення найбільш ймовірної категорії. Такий підхід поєднує персоналізацію, простоту правил і можливість статистичного аналізу текстового опису операції.

Під час дослідження методів фільтрації та агрегації фінансових даних було розглянуто два основні підходи: обробку даних на стороні прикладного застосунку та обробку на рівні бази даних. У першому випадку записи завантажуються у пам’ять застосунку та обробляються засобами Python, однак такий підхід призводить до збільшення використання оперативної пам’яті та навантаження на сервер при роботі з великими обсягами даних. Другий підхід передбачає виконання фільтрації та агрегації безпосередньо засобами PostgreSQL із використанням SQL-запитів, операторів WHERE, GROUP BY,

SUM та індексів бази даних. У межах роботи було обрано другий підхід, оскільки він забезпечує кращу продуктивність, зменшує обсяг переданих даних між сервісами та дозволяє ефективно масштабувати систему при збільшенні кількості фінансових записів користувачів. Додатково досліджено можливість використання асинхронного доступу до бази даних через SQLAlchemy Async, що дозволяє паралельно обробляти запити Telegram-бота та WebApp без блокування основного потоку виконання.

Фільтрація записів формалізується як вибір підмножини R′ із множини всіх фінансових записів R за набором критеріїв: періодом P, типом операції T, множиною категорій C та діапазоном сум A. Такий підхід використовується для історії операцій, звітів, дашборду та експорту.

$$R' = \{ r \in R \mid date(r) \in P \land type(r) = T \land category(r) \in C \land amount(r) \in A \},$$

де R – множина всіх фінансових записів; R′ – відфільтрована множина; P – період; T – тип операції; C – множина категорій; A – допустимий діапазон сум.

Агрегація даних використовується для обчислення доходів, витрат і балансу за період. Загальна сума доходів визначається як сума записів типу income, загальна сума витрат – як сума записів типу expense, а баланс – як різниця між доходами і витратами.

$$Income(P) = \Sigma\ amount(r), \text{ якщо } r \in R' \text{ та } type(r) = income$$

$$Expense(P) = \Sigma\ amount(r), \text{ якщо } r \in R' \text{ та } type(r) = expense$$

$$Balance(P) = Income(P) - Expense(P)$$

Для аналізу структури витрат обчислюється частка кожної категорії. Цей показник застосовується для побудови діаграм, визначення найбільших напрямів витрат і формування рекомендацій щодо оптимізації бюджету.

$$Share(c, P) = Expense(c, P) / Expense(P) \times 100\%$$

Контроль бюджетних лімітів виконується шляхом порівняння фактичних витрат за категорією з установленим користувачем лімітом. Якщо відсоток

використання наближається до критичного значення або перевищує 100%, система може сформувати попередження.

Usage(c, P) = Expense(c, P) / Limit(c, P) × 100%

Рекурентні операції обробляються фоновим циклом. Система періодично перевіряє активні шаблони, порівнює next_date з поточною датою та створює новий фінансовий запис або надсилає нагадування. Після виконання дата наступного запуску оновлюється відповідно до періодичності: день, тиждень або місяць.

Таблиця 2.4 – Алгоритми, використані у програмному продукті

Алгоритм	*Вхідні дані*	*Основний принцип*	*Результат*
Швидке текстове введення	Повідомлення користувача	Пошук суми, опису та ключових слів категорії	Попередньо сформований фінансовий запис

Таблиця 2.4 – Алгоритми, використані у програмному продукті

OCR чека	Фотографія чека	Попередня обробка зображення, OCR, пошук підсумкової суми	Пропозиція збереження операції
Фільтрація	Записи та критерії відбору	Вибір підмножини за датою, типом, категорією, сумою	Коректний список операцій
Агрегація	Відфільтровані записи	SUM, GROUP BY, розрахунок балансу	Доходи, витрати, баланс, частки категорій
Контроль бюджету	Фактичні витрати та ліміти	Порівняння витрат із планом	Відсоток використання та попередження
Рекурентні операції	Активні шаблони платежів	Перевірка next_date і оновлення дати	Автопост або нагадування

Висновки до розділу 2

У другому розділі кваліфікаційної роботи виконано проєктування програмного продукту для управління особистими фінансами. Визначено основні принципи функціонування системи та її структурну організацію. Особливу увагу приділено моделюванню поведінки користувача під час взаємодії з програмним продуктом. Також розглянуто автоматизовані процеси, що забезпечують підтримку роботи системи.

У процесі проєктування побудовано поведінкову модель програмного продукту. Визначено ключові сценарії взаємодії користувача з Telegram-ботом і WebApp. Описано процеси додавання доходів і витрат, перегляду статистики та використання аналітичних функцій. Окремо розглянуто OCR-обробку чеків і механізми формування фінансових звітів.

Розроблено архітектурну модель системи, яка охоплює Telegram-бот, Telegram WebApp та FastAPI API. Визначено взаємодію між сервісним рівнем, рівнем доступу до даних та системою збереження інформації. Для роботи з базою даних передбачено використання асинхронного доступу із застосуванням PostgreSQL. Такий підхід забезпечує масштабованість, продуктивність та зручність супроводу програмного продукту.

У межах другого розділу також спроєктовано модель бази даних і структуру програмних модулів. Визначено основні сутності, необхідні для збереження фінансової інформації користувачів та налаштувань системи. Описано інтерфейсні сценарії роботи та механізми взаємодії між компонентами програмного забезпечення. Значну увагу приділено алгоритмам фільтрації, агрегації, бюджетного контролю та підтримки рекурентних операцій.

Отримані результати другого розділу є основою для подальшої практичної реалізації програмного продукту. Виконане проєктування дозволяє перейти до етапу програмної реалізації окремих компонентів системи. Сформована архітектура забезпечує можливість розширення функціоналу в

майбутньому. Практична реалізація та тестування розробленого рішення будуть розглянуті у наступному розділі.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Реалізація та конструювання програмного продукту

Реалізація програмного продукту виконана з використанням асинхронного Python-стеку. Основними технологіями є Python, aiogram, FastAPI, SQLAlchemy Async, PostgreSQL 16, pytesseract, Pillow і Docker Compose. Обраний стек забезпечує одночасну обробку Telegram-повідомлень, HTTP-запитів WebApp, операцій з базою даних, OCR-розпізнавання та фонових задач.

Telegram-бот відповідає за швидку взаємодію з користувачем. Він обробляє команди, текстові повідомлення, натискання кнопок і фотографії чеків. Для маршрутизації подій використовується aiogram, що дозволяє поділити логіку на окремі модулі: меню, швидке введення, ручне додавання, OCR, звіти та налаштування.

Веб-застосунок реалізовано на FastAPI. Він обслуговує Telegram WebApp і надає REST API за шляхом /api/webapp/*. Через API користувач може працювати із записами, категоріями, дашбордом, аналітикою, бюджетом, лімітами, рекурентними операціями, налаштуваннями, журналом аудиту та експортом звітів.

Проєкт запускається через Docker Compose. Сервіс db працює на PostgreSQL 16 і зберігає дані у volume pgdata. Сервіс finance-bot запускає Telegram-бота з app/bot.py. Сервіс finance-web запускає FastAPI веб-застосунок командою python -m app.web_main. Налаштування токена, параметрів бази даних, URL WebApp та рівня логування виконуються через файл .env.

Точкою входу Telegram-бота є app/bot.py. У цьому модулі створюються екземпляри Bot і Dispatcher, підключаються маршрутизатори, виконується видалення webhook перед polling, а також запускаються фонові цикли

автопостингу рекурентних записів і нагадувань. Така організація дозволяє відокремити запуск системи від прикладної логіки окремих сценаріїв.

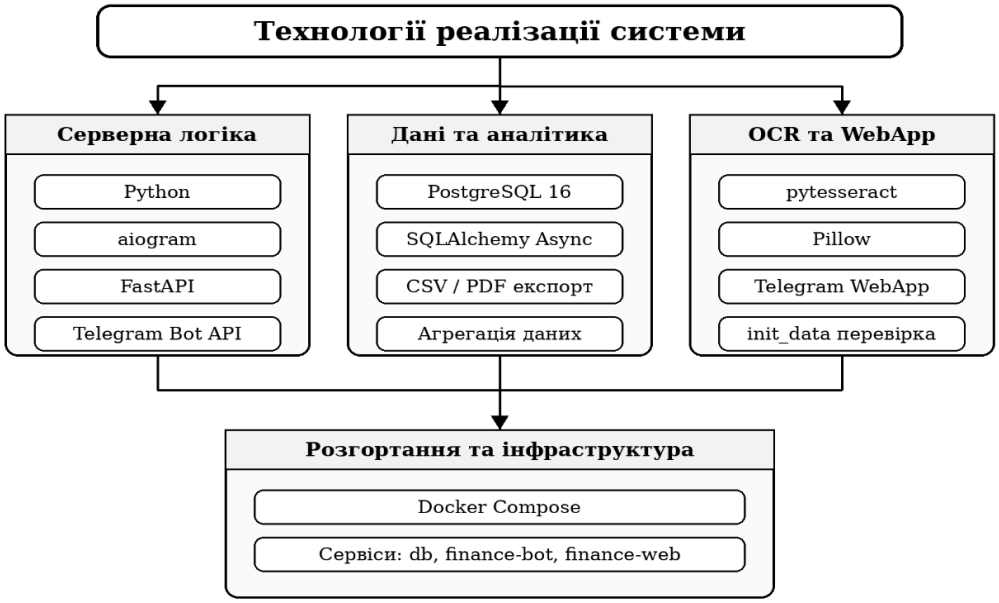

Рисунок 3.1 – Технології реалізації телеграм-боту та веб-застосунку

Джерело: розроблено автором

Структура програмних модулів системи

Основний пакет застосунку: app/

Група модулів	Склад програмних модулів
Telegram-бот	bot.py; handlers/common.py; routes_menu.py; routes_manual_add.py; routes_quick_expense.py; routes_ocr.py; routes_reports.py.
WebApp та API	web_main.py; api/webapp/init; api/webapp/dashboard; api/webapp/records; api/webapp/budget; api/webapp/export; перевірка Telegram init_data.
Сервісні модулі	OCR-сервіс чеків; класифікатор категорій; аналітика та агрегація; експорт CSV/PDF; фонові задачі; нагадування.
Рівень даних	PostgreSQL 16; SQLAlchemy Async; моделі бази даних; асинхронні сесії; операції з фінансовими записами.
Розгортання	Docker Compose; service: db; service: finance-bot; service: finance-web; спільна мережа компонентів системи.

Рисунок 3.2 – Структура програмних модулів системи

Джерело: розроблено автором

Таблиця 3.1 – Призначення основних модулів програмного продукту

Модуль	*Призначення*
app/bot.py	Запуск Telegram-бота, Dispatcher, polling, фонові задачі
app/handlers/common.py	Спільний контекст, підключення БД, OCR-сервісу та класифікатора
app/handlers/routes_menu.py	Головне меню, команди /start, /help, налаштування і WebApp
app/handlers/routes_quick_expense.py	Швидке створення витрати з текстового повідомлення
app/handlers/routes_manual_add.py	Покрокове ручне додавання доходів і витрат
app/handlers/routes_ocr.py	Отримання фото чека, OCR-розпізнавання і підтвердження запису
app/handlers/routes_reports.py	Списки, статистика, бюджетні команди і звіти
app/services/*	Бізнес-логіка: записи, агрегація, OCR, категоризація, бюджет
app/repositories/*	Запити до бази даних і робота з фільтрами
app/web_main.py	Запуск FastAPI веб-застосунку
app/web/auth.py	Перевірка Telegram WebApp init_data
app/models.py / app/schemas.py	ORM-моделі бази даних і схеми валідації даних

Реалізація сховища даних базується на ORM-моделях SQLAlchemy. Дані користувачів ізолюються за user_id, що дозволяє кожному користувачу працювати лише зі своїми фінансовими записами, категоріями, бюджетами та налаштуваннями. Асинхронний доступ до бази даних підвищує стабільність системи при одночасній роботі Telegram-бота і WebApp.

Веб-застосунок використовує Telegram WebApp init_data для ідентифікації користувача. Модуль auth.py перевіряє достовірність отриманих даних, після чого залежності FastAPI отримують відповідний профіль користувача з бази даних. Це забезпечує зв’язок між Telegram-профілем і веб-сесією без створення окремої системи авторизації.

Програмний код проєкту структуровано як репозиторій, що містить основні пакети застосунку, конфігураційні файли Docker, приклади змінних середовища, документацію та каталог автоматизованих тестів. Така організація спрощує розгортання програмного продукту, повторну перевірку працездатності системи та подальший супровід коду. Для локального запуску передбачено використання файлу .env, а для повного запуску системи — docker-compose.yml, який піднімає базу даних, Telegram-бота та WebApp API.

Таблиця 3.2 – Основні API-маршрути веб-застосунку

Маршрут	*Метод*	*Призначення*
/api/webapp/bootstrap	GET	Початкове завантаження даних WebApp
/api/webapp/categories	GET	Отримання списку категорій
/api/webapp/dashboard	GET	Дані для дашборду
/api/webapp/records	GET/POST/PATCH/DELETE	CRUD-операції з фінансовими записами
/api/webapp/analytics	GET	Аналітика за періодами та категоріями
/api/webapp/budget	GET	Отримання даних бюджету
/api/webapp/budget/month	POST/PUT	Збереження плану бюджету на місяць
/api/webapp/budget/category-limits	GET/POST	Робота з лімітами категорій
/api/webapp/recurring	GET/POST/PATCH/DELETE	Керування рекурентними операціями
/api/webapp/settings	GET/PUT	Перегляд і зміна налаштувань користувача

Таблиця 3.2 – Основні API-маршрути веб-застосунку

/api/webapp/export/csv	GET	Формування CSV-звіту
/api/webapp/export/pdf	GET	Формування PDF-звіту
/api/webapp/health	GET	Перевірка працездатності сервісу

3.2 Тестування програмного продукту

Тестування програмного продукту виконувалося для перевірки коректності роботи Telegram-бота, WebApp, бази даних, OCR-сервісу, алгоритмів фільтрації та агрегації, а також фонових задач. Оскільки система складається з кількох компонентів, тестування поділено на функціональне, модульне та інтеграційне.

Автоматизована частина тестування виконувалася із застосуванням pytest та pytest-asyncio для перевірки сервісної логіки, асинхронних функцій і роботи окремих модулів. Для перевірки сценаріїв взаємодії з веб-застосунком передбачено E2E-підхід із використанням Playwright, що дозволяє перевіряти відкриття сторінок WebApp, відображення елементів інтерфейсу та коректність базових користувацьких сценаріїв. Ручне сценарне тестування застосовувалося для перевірки Telegram-бота, OCR-обробки чеків і взаємодії між ботом, WebApp та базою даних.

Функціональне тестування Telegram-бота охоплює запуск через команду /start, відображення головного меню, додавання доходу, додавання витрати, швидке створення витрати з тексту, ручне додавання, перегляд історії, отримання статистики, відкриття WebApp і реакцію на некоректні дані.

OCR-сервіс тестувався на фотографіях чеків різної якості. Перевірялося отримання файлу від Telegram, попередня обробка зображення, розпізнавання тексту, пошук підсумкової суми та формування пропозиції для користувача.

Якщо розпізнавання недостатньо точне, система повинна запропонувати ручне введення замість автоматичного збереження помилкового запису.

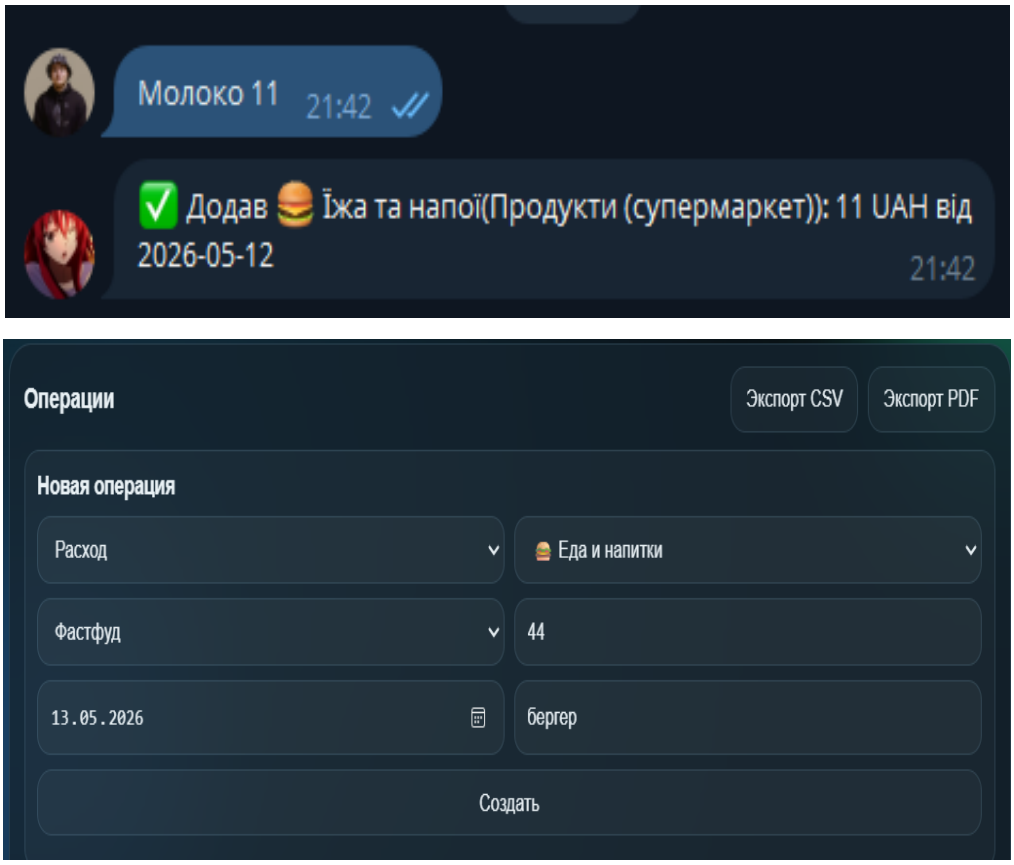

Рисунок 3.3 – Приклад тестування додавання фінансової операції

Джерело: розроблено автором

Тестування WebApp включало перевірку API-маршрутів, відображення дашборду, списку записів, аналітики, бюджету, рекурентних операцій, налаштувань та експорту CSV/PDF. Окремо перевірялася автентифікація через Telegram WebApp init_data, оскільки від неї залежить безпечне отримання даних конкретного користувача.

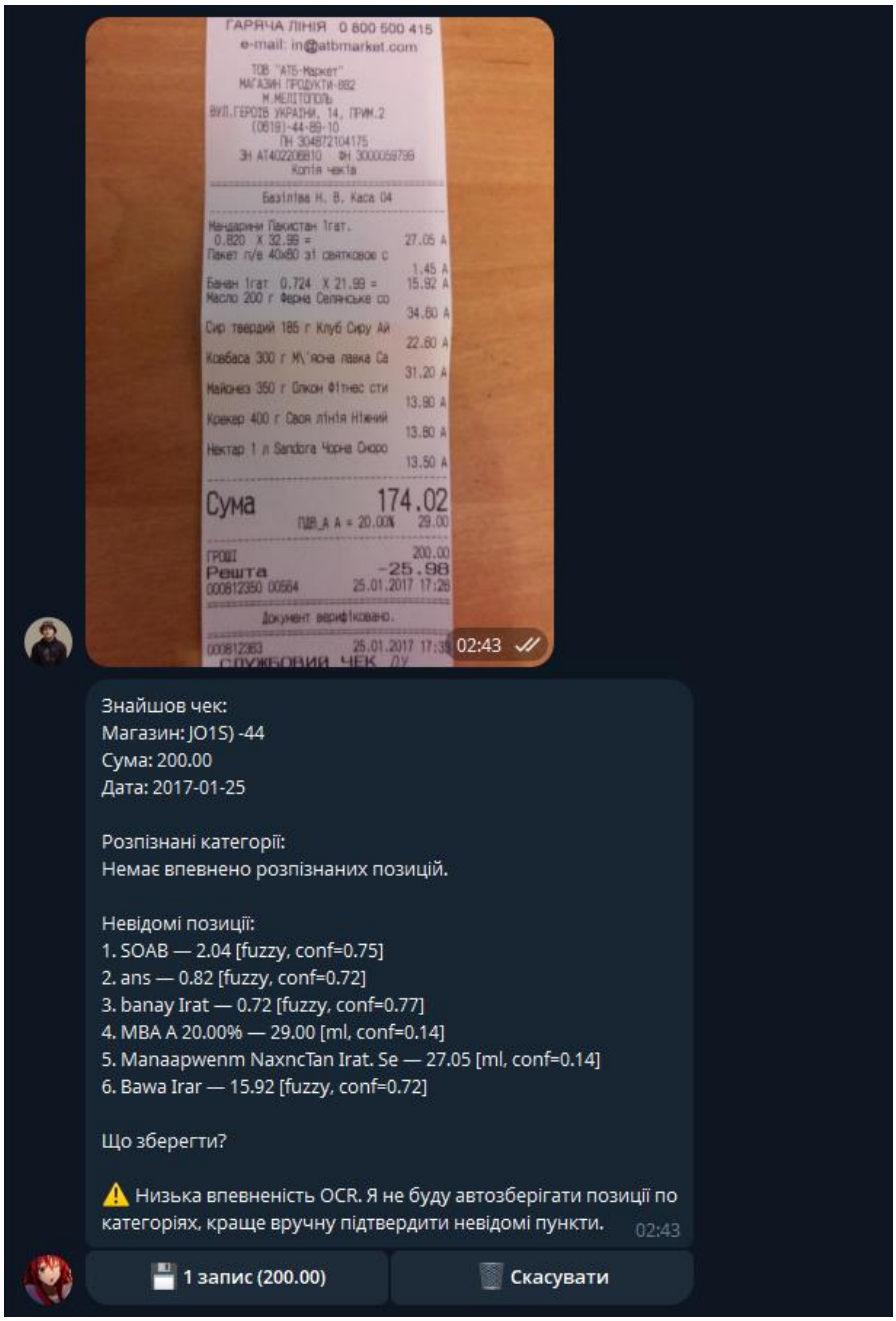

Рисунок 3.4 – Приклад тестування OCR-обробки чека

Джерело: розроблено автором

Для перевірки коректності роботи системи було виконано модульне тестування функцій виділення суми з тексту, підбору категорії, фільтрації записів, агрегації доходів і витрат, розрахунку бюджетних лімітів, визначення наступної дати рекурентної операції та перевірки Telegram WebApp init_data. Критерієм успіху був збіг отриманого результату з очікуваним для нормальних

і граничних вхідних даних. Тестування виконувалося автором роботи у локальному Docker Compose-середовищі шляхом ручної перевірки функціональних сценаріїв та аналізу фактичних результатів роботи системи. Такий підхід дозволив своєчасно виявити та усунути можливі помилки у роботі програмного продукту.

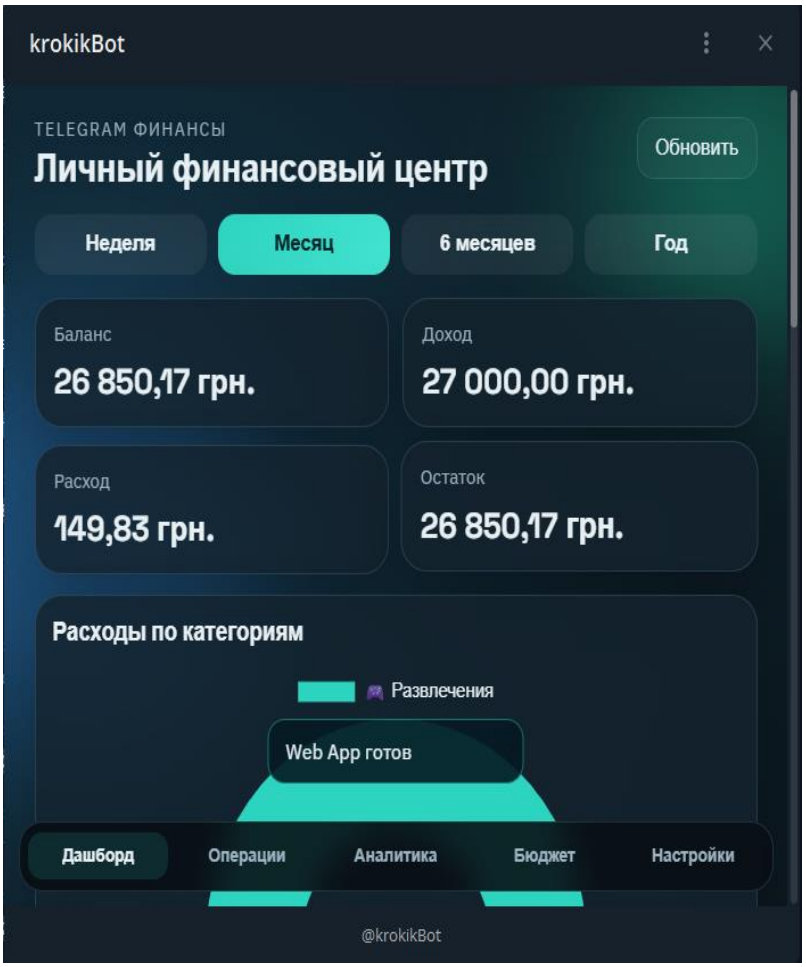

Рисунок 3.5 – Приклад тестування веб-застосунку

Джерело: розроблено автором

Таблиця 3.3 – Функціональне тестування основних сценаріїв

№	*Сценарій*	*Очікуваний результат*	*Статус*
ТС-01	Запуск /start	Показано привітання та головне меню	Успішно

TC-02	Швидке додавання витрати	Визначено суму, опис і запропоновано категорію	Успішно
TC-03	Ручне додавання операції	Запис збережено у базі даних	Успішно
TC-04	Введення некоректної суми	Користувач отримує повідомлення про помилку	Успішно
TC-05	Надсилання фото чека	Виконано OCR і запропоновано збереження	Успішно
TC-06	Фільтрація записів	Повернуто записи, що відповідають критеріям	Успішно
TC-07	Агрегація даних	Пораховано доходи, витрати та баланс	Успішно
TC-08	Контроль бюджету	Показано відсоток використання ліміту	Успішно
TC-09	Рекурентний запис	Створено запис або сформовано нагадування	Успішно
TC-10	Експорт звіту	Сформовано CSV або PDF-файл	Успішно
TC-11	Відкриття WebApp	Завантажено дашборд користувача	Успішно
TC-12	Невалідні init_data	Запит відхилено	Успішно

Інтеграційне тестування виконується у Docker Compose-середовищі, що дозволяє перевірити спільну роботу всіх компонентів системи в умовах, наближених до реального розгортання. Перевіряється запуск усіх сервісів, доступ finance-bot і finance-web до PostgreSQL, робота спільної мережі, читання змінних середовища, збереження даних у volume, а також

узгодженість даних між Telegram-ботом і WebApp. Окремо перевіряється ситуація, коли фінансова операція створюється через Telegram-бота та після цього коректно відображається у веб-додатку. Додатково перевіряється коректність обміну даними між компонентами системи, працездатність основних API-запитів, стабільність роботи сервісів під час одночасного виконання запитів та збереження даних після перезапуску контейнерів.

Таблиця 3.4 – Напрями модульного тестування

Функція / модуль	*Що перевіряється*	*Критерій успіху*
parse_quick_expense	Виділення суми та опису з тексту	Сума і опис визначені коректно
classify_category	Підбір категорії за ключовими словами	Категорія відповідає змісту операції або запитується вручну
filter_records	Фільтрація за датою, типом, категорією і сумою	У результаті немає зайвих записів
aggregate_records	Розрахунок доходів, витрат і балансу	Показники збігаються з ручним розрахунком
calculate_budget_usage	Порівняння витрат з лімітом	Відсоток використання обчислено правильно
next_recurring_date	Визначення наступної дати регулярної операції	Дата відповідає заданій періодичності
validate_init_data	Перевірка даних Telegram WebApp	Валідні дані приймаються, невалідні відхиляються
export_csv/pdf	Формування звітів	Файл створено та містить коректні дані

3.3 Використання програмного продукту

Використання програмного продукту починається із запуску Telegram-бота командою /start. Після цього система створює або знаходить профіль користувача за Telegram ID, відображає привітання та головне меню. Головне меню є центральною точкою навігації для швидких дій.

Користувач може швидко додати витрату текстовим повідомленням, наприклад: «250 грн обід». Система виділяє суму та опис, пропонує категорію і просить підтвердити збереження. Для точнішого введення доступний ручний сценарій через меню, де користувач послідовно обирає тип операції, категорію, суму, дату та опис.

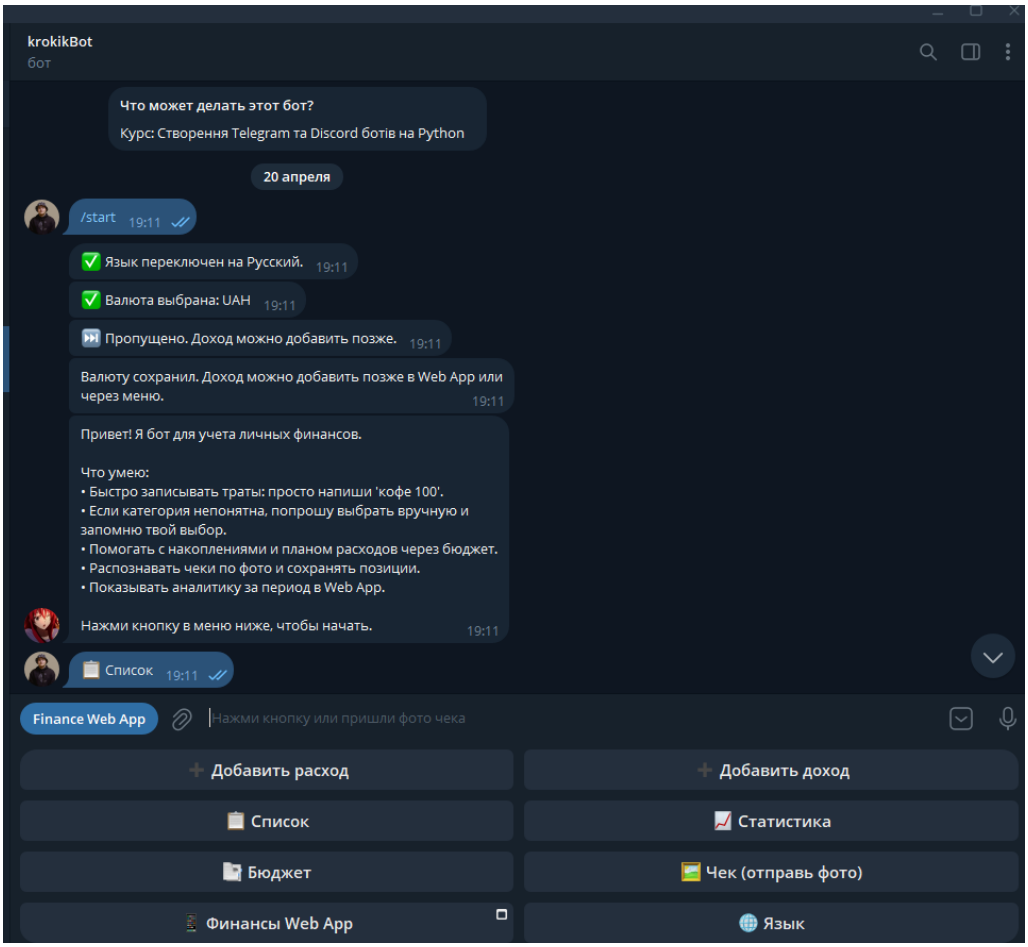

Рисунок 3.6 – Приклад головного меню Telegram-бота

Джерело: розроблено автором

Для обліку покупок за чеками користувач надсилає фотографію. Бот виконує OCR-розпізнавання, аналізує отриманий текст і пропонує зберегти операцію. Якщо розпізнана сума або категорія викликає сумнів, користувач може скоригувати результат або перейти до ручного введення.

WebApp використовується для детальної роботи з фінансовими даними. У ньому користувач може переглядати дашборд, редагувати записи, аналізувати витрати, налаштовувати бюджет, встановлювати ліміти за категоріями, керувати рекурентними платежами, переглядати журнал аудиту та експортувати звіти.

Дашборд показує загальні доходи, витрати, баланс, динаміку за періодами і структуру витрат за категоріями. Саме WebApp є зручним для перегляду великих обсягів інформації, тоді як Telegram-бот оптимізований для швидкого введення й коротких відповідей.

Рисунок 3.7 – Приклад дашборду веб-застосунку

Джерело: розроблено автором

Розділ бюджету дозволяє встановлювати планові суми доходів і витрат на період, а також ліміти за категоріями. Система порівнює фактичні витрати із запланованими значеннями та показує ступінь використання ліміту.

Рекурентні операції застосовуються для регулярних платежів: підписок, оренди, інтернету або комунальних послуг.

Функція експорту формує звіти у CSV або PDF. CSV зручний для подальшої обробки у табличних редакторах, а PDF – для збереження або надсилання. У результаті користувач отримує інструмент як для щоденного введення витрат, так і для детального аналізу фінансової поведінки. Це підвищує зручність використання системи.

Таблиця 3.5 – Інструкція користувача для основних сценаріїв

Дія користувача	*Спосіб виконання*	*Результат*
Запустити бота	Команда /start	Створено профіль і показано головне меню
Додати витрату	Текстове повідомлення або меню	Витрату збережено в базі даних
Додати дохід	Пункт меню ручного додавання	Дохід збережено в базі даних
Обробити чек	Надсилання фотографії	OCR-результат запропоновано до підтвердження
Переглянути історію	Команда або WebApp	Показано список відфільтрованих записів
Отримати статистику	Telegram або WebApp	Показано доходи, витрати, баланс і категорії
Налаштувати бюджет	WebApp → Бюджет	Збережено план і категорійні ліміти
Створити рекурентний платіж	Розділ рекурентних операцій	Увімкнено автопост або нагадування
Експортувати звіт	WebApp → Експорт CSV/PDF	Сформовано файл звіту

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи виконано реалізацію програмного продукту для управління особистими фінансами. Розроблено систему, що складається з Telegram-бота, FastAPI веб-застосунку та

PostgreSQL бази даних. Описано використані технології, структуру програмних модулів і принципи взаємодії між компонентами системи.

У процесі реалізації створено API-маршрути, сервіси обробки фінансових операцій, OCR-розпізнавання чеків, а також механізми фільтрації та агрегації даних. Окрему увагу приділено реалізації Telegram WebApp, механізмам автентифікації та фоновим задачам. Це забезпечило повноцінну роботу системи та підтримку основних сценаріїв використання.

У межах третього розділу проведено функціональне, модульне та інтеграційне тестування програмного продукту. Перевірено запуск бота, додавання доходів і витрат, OCR-обробку чеків, фільтрацію, агрегацію, бюджетування, рекурентні операції та експорт звітів. Результати тестування підтвердили стабільність і коректність роботи системи.

Також перевірено взаємодію між Telegram-ботом, WebApp і базою даних у Docker Compose-середовищі. Оцінено узгодженість даних, доступ до сервісів і коректність обробки запитів. Це дозволило підтвердити надійність реалізованої архітектури програмного продукту.

Також у розділі наведено інструкцію користувача, що демонструє практичне використання розробленої системи. Описано основні сценарії взаємодії з Telegram-ботом і веб-застосунком, а також процеси введення та аналізу фінансових даних. Отримані результати підтверджують готовність програмного продукту до практичного використання. Реалізована система забезпечує зручний облік фінансів, аналіз витрат і підтримку фінансового планування користувачів.

До обмежень реалізованого програмного продукту можна віднести залежність якості OCR-розпізнавання від чіткості фотографії чека, відсутність прямої інтеграції з банківськими API, необхідність стабільного доступу до Telegram та обмеженість автоматичної категоризації у випадках неоднозначного текстового опису операції. Водночас зазначені обмеження не знижують практичної цінності розробки, оскільки система передбачає ручне

підтвердження та редагування даних, а її архітектура дозволяє надалі додати банківські інтеграції, кешування важких аналітичних запитів і розширені моделі прогнозування витрат.

ВИСНОВКИ

У кваліфікаційній роботі розроблено програмний продукт у вигляді Telegram-бота та веб-застосунку для управління особистими фінансами користувачів. Створена система забезпечує автоматизований облік доходів і витрат, збереження фінансових операцій у базі даних, їх подальшу обробку, фільтрацію, агрегацію та представлення результатів у зручному для користувача вигляді. Реалізовано функціонал роботи з категоріями, бюджетами, лімітами за окремими напрямами витрат, рекурентними операціями, OCR-обробкою чеків та експортом звітів у форматах CSV і PDF. Особливу увагу приділено використанню алгоритмів фільтрації та агрегації фінансових даних, оскільки саме вони забезпечують формування аналітичної інформації, розрахунок доходів, витрат, балансу, часток категорій та інших показників фінансової активності.

У процесі виконання роботи було досліджено предметну область управління особистими фінансами та визначено основні потреби користувачів у програмних засобах такого типу. Проведено аналіз існуючих аналогів, зокрема Telegram-ботів і сервісів для ведення фінансового обліку, розглянуто їхні функціональні можливості, переваги та недоліки. На основі проведеного аналізу сформовано вимоги до програмного продукту та визначено основні напрями його реалізації. Встановлено, що актуальним є створення рішення, яке поєднує простоту швидкого введення даних через Telegram-бота з можливостями детального аналізу фінансової інформації у веб-застосунку.

У другому розділі виконано проєктування поведінки системи, архітектури програмного продукту, структури бази даних і механізмів взаємодії між компонентами. Було визначено основні користувацькі сценарії, зокрема запуск бота, додавання доходів і витрат, OCR-обробку чеків, перегляд історії операцій, формування статистики, контроль бюджету, роботу з рекурентними операціями та експорт звітів. Архітектуру системи побудовано за багаторівневим принципом, що дозволило розділити інтерфейсний,

прикладний, сервісний рівні, рівень доступу до даних та рівень збереження інформації. Такий підхід забезпечує підтримуваність програмного коду, можливість тестування окремих компонентів і подальше розширення функціоналу.

У третьому розділі реалізовано програмний продукт із використанням сучасних технологій розробки програмного забезпечення. Telegram-бот реалізовано з використанням Python та aiogram, веб-застосунок і API — з використанням FastAPI, доступ до бази даних організовано через SQLAlchemy Async, а для збереження інформації використано PostgreSQL. OCR-обробку чеків реалізовано із застосуванням Tesseract OCR, а запуск компонентів системи організовано за допомогою Docker Compose. Такий технологічний стек забезпечив асинхронну обробку запитів, стабільну роботу Telegram-бота та WebApp, а також можливість розгортання системи в контейнеризованому середовищі.

У межах роботи проведено функціональне, модульне та інтеграційне тестування основних сценаріїв роботи системи. Перевірено запуск Telegram-бота, додавання доходів і витрат, швидке введення операцій, OCR-розпізнавання чеків, фільтрацію записів, агрегацію фінансових даних, формування аналітики, контроль бюджетних лімітів, роботу з рекурентними операціями, відкриття WebApp та експорт звітів. Інтеграційне тестування підтвердило коректну взаємодію між Telegram-ботом, веб-застосунком і базою даних у Docker Compose-середовищі. Результати тестування підтвердили стабільність, коректність і працездатність реалізованого рішення в умовах інтегрованого середовища.

Мету кваліфікаційної роботи досягнуто, оскільки створено програмний продукт, який автоматизує процес управління особистими фінансами та забезпечує користувачу зручний доступ до фінансової аналітики. Розроблена система дозволяє спростити процес ведення особистого бюджету, підвищити прозорість фінансової активності та забезпечити більш обґрунтоване

прийняття фінансових рішень. Практичне значення роботи полягає в можливості використання створеного Telegram-бота та веб-застосунку як інструмента щоденного обліку доходів і витрат, аналізу фінансової поведінки та планування бюджету.

Розроблена система має потенціал подальшого вдосконалення та масштабування відповідно до потреб користувачів. Перспективними напрямами розвитку можуть бути прогнозування витрат, персоналізовані фінансові рекомендації, підтримка мультивалютності, інтеграція з банківськими сервісами, розширення механізмів автоматичної категоризації операцій та покращення якості OCR-обробки чеків. Реалізація цих можливостей дозволить підвищити ефективність фінансового планування, розширити функціональність програмного продукту та покращити користувацький досвід.

Вихідний код розробленого Telegram-бота та веб-застосунку розміщено у відкритому репозиторії GitHub за посиланням URL:https://github.com/mi-show/Telegram-Finance-Bot-Web-App

ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАННЯ

1. Телеграм-бот “Finny”. URL: https://finnybot.com/ (дата звернення: 02.04.2026).

2. Телеграм-бот “Telegram Budget Bot”. URL: https://telegram.myshell.ai/budgeting (дата звернення: 02.04.2026).

3. Телеграм-бот “Telexpense Bot”. URL: https://github.com/pavelmakis/telexpense/wiki (дата звернення: 02.04.2026).

4. Aiogram 3.28.1 documentation. URL: https://docs.aiogram.dev/en/latest/ (дата звернення: 12.05.2026).

5. Docker Compose documentation. URL: https://docs.docker.com/compose/ (дата звернення: 12.05.2026).

6. Docker Compose file reference. URL: https://docs.docker.com/reference/compose-file/ (дата звернення: 12.05.2026).

7. FastAPI documentation. URL: https://fastapi.tiangolo.com/ (дата звернення: 12.05.2026).

8. McKinney W. Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter. 3rd ed. Boston : O’Reilly Media, 2022. 578 p.

9. Pillow documentation. URL: https://pillow.readthedocs.io/ (дата звернення: 12.05.2026).

10. PostgreSQL 16 Documentation. URL: https://www.postgresql.org/docs/16/ (дата звернення: 12.05.2026).

11. Pytest documentation. URL: https://docs.pytest.org/ (дата звернення: 12.05.2026).

12. Pytesseract. Python-tesseract OCR wrapper. URL: https://pypi.org/project/pytesseract/ (дата звернення: 12.05.2026).

13. SQLAlchemy 2.0 Documentation. Asynchronous I/O asyncio. URL: https://docs.sqlalchemy.org/en/20/orm/extensions/asyncio.html (дата звернення: 12.05.2026).

14. Telegram Bot API. Official documentation for developers. URL: https://core.telegram.org/bots/api (дата звернення: 12.05.2026).

15. Telegram Mini Apps. Init Data documentation. URL: https://docs.telegram-mini-apps.com/platform/init-data (дата звернення: 12.05.2026).

16. Telegram Mini Apps. Web Apps documentation. URL: https://core.telegram.org/bots/webapps (дата звернення: 12.05.2026).

17. Playwright documentation. URL: https://playwright.dev/python/ (дата звернення: 03.06.2026).

18. Scikit-learn documentation. Feature extraction. URL:https://scikit-learn.org/stable/modules/feature_extraction.html (дата звернення: 03.06.2026).

19. RapidFuzz documentation. URL: https://rapidfuzz.github.io/RapidFuzz/ (дата звернення: 03.06.2026).

20. Chart.js documentation. URL: https://www.chartjs.org/docs/latest/ (дата звернення: 03.06.2026).