Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Застосунок для відстеження витрат та доходів

(тема)

Виконав:
здобувач 3 року навчання,
групи КІУКІу-22-1
Роман БАЛАБАНОВ
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ст. викл. Ольга ЄРОШЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерія та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Балабанову Роману Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Застосунок для відстеження витрат та доходів

затверджена наказом по університету від “ 26 ” травня 2025 р. № 425 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 14 липня 2025р.

3. Вхідні дані до роботи

C++, PostgreSQL.

4. Перелік питань, що потрібно опрацювати у роботі

Огляд

Засоби розробки

Програмна реалізація

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Презентація – 12 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Вибір теми, узгодження з науковим керівником	27.05.2025 – 28.05.2025	
2	Збір та аналіз літературних джерел за темою	29.05.2025 – 30.05.2025	
3	Підготовка оглядово-аналітичного розділу	31.05.2025 – 01.06.2025	
4	Постановка задачі та формалізація вимог до ПЗ	02.06.2025 – 03.06.2025	
5	Проєктування архітектури застосунку та бази даних	04.06.2025 – 14.06.2025	
6	Реалізація програмного засобу	15.06.2025 – 30.06.2025	
7	Тестування, налагодження, внесення коригувань	01.07.2025 – 10.07.2025	
8	Оформлення пояснювальної записки	11.07.2025 – 14.07.2025	
9	Перевірка на плагіат, рецензування, попередній захист	10.07.2025	
10	Захист кваліфікаційної роботи	16.07.2025	

Дата видачі завдання " 26 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Ольга ЄРОШЕНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 49 с., 16 рис., 1 дод., 8 джерел.

КОНТРОЛЬ БЮДЖЕТУ, ДОДАТОК, C++, PostgreSQL.

Метою кваліфікаційної роботи є створення програми для контролю бюджету. Програма була розроблена для надання кінцевому користувачеві легко контролювати свої витрати. Робота розглядає технічну та бізнес частину розробки android-додатків.

Додаток ставить собі за мету допомогти користувачам стежити за своїми витратами, щоб надалі аналізувати які витрати є зайвими та на основі цього правильно планувати свій бюджет.

Кваліфікаційна робота включає цілі, визначення і загальний опис, описує технології, якими ми користуватимемося в розробка продукту, опис архітектури, бази даних та бізнес частині програми.

ABSTRACT

Bachelor’s thesis: 49 pages, 17 figures, 1 appendices, 8 sources.

BUDGET CONTROL, SUPPLEMENT, C++, PostgreSQL.

The purpose of the qualification work is to create a budget control program. The program was designed to allow the end user to easily control their expenses. The work considers the technical and business part of developing android applications.

The application aims to help users monitor their expenses in order to further analyze which expenses are unnecessary and based on this to correctly plan their budget.

The qualification work includes goals, definitions and a general description, describes the technologies that we will use in product development, describes the architecture, database and business part of the program.

ЗМІСТ

ВСТУП

Фінансове планування та контроль витрат є фундаментальними елементами ефективного управління фінансами, особливо в умовах динамічних і непередбачуваних економічних змін. Сучасні реалії, що характеризуються стрімким технологічним розвитком і необхідністю оптимального управління фінансовими ресурсами, формують нові виклики, які вимагають креативних і технологічно підкованих рішень.

У контексті зростаючої економічної нестабільності, трансформацій у суспільстві та підвищення обізнаності щодо фінансової грамотності, постає нагальна потреба у доступних і дієвих інструментах для ведення особистих фінансів. Зі збільшенням обсягу інформації, що стосується управління коштами, актуальним стає розроблення нових підходів до її аналізу та обробки з метою покращення досвіду кінцевого користувача.

Сьогодні на ринку існує велика кількість програм для фінансового обліку, проте багато з них не задовольняють сучасні запити споживачів - мають обмежений функціонал або незручний інтерфейс. Це формує потребу у створенні гнучких, зручних і функціонально насичених рішень. Саме тому дана розробка є своєчасною та доцільною, адже її мета - створення інструменту, який не лише дозволить контролювати витрати, а й забезпечить ефективне планування фінансів відповідно до особистих цілей користувача.

Застосунок у цьому контексті виступає як спеціалізоване програмне забезпечення, орієнтоване на вирішення конкретних задач користувача. Це поняття дозволяє відмежувати прикладні програми від системного ПЗ, такого як драйвери, операційні системи або середовища розробки [1].

Аналіз існуючих програм у сфері особистого фінансового менеджменту засвідчив потребу в комплексному рішенні, яке поєднує інтуїтивно зрозумілий інтерфейс, глибокий функціонал для аналізу та широкі можливості налаштування під індивідуальні потреби користувачів різних

категорій.

Ця робота має на меті створення високоефективного мобільного або десктопного застосунку для фінансового планування та обліку витрат. Її значення полягає у сприянні підвищенню фінансової свідомості населення та забезпеченні інструменту, що дозволить користувачам ефективно керувати власними коштами, покращуючи тим самим їхнє фінансове становище та стимулюючи розвиток суспільства загалом.

1 ОГЛЯД

1.1 Актуальність розробки вебзастосунку для створення та відстеження фінансових бюджетів та заощаджень

У наш час застосунки щільно інтегрувалися в повсякденне життя та стали невід'ємним його елементом. Ці програми значно полегшують вирішення повсякденних задач, роблячи їх швидшими та зручнішими. Завдяки застосункам можна легко спланувати день, зберігати цифровий гаманець під рукою, навчатися нового, переглядати або слухати освітній контент дорогою на роботу, а також підтримувати зв'язок з друзями, незалежно від їхнього місця перебування.

Багато сучасних застосунків перетворюють рутинні дії, як-от фізичні вправи чи вивчення мов, у захопливі ігри, де користувач стає головним героєм, поступово досягаючи поставлених цілей, долаючи рівень за рівнем. Таким чином, вебзастосунки виходять далеко за межі інструментів - вони стають особистими помічниками, які не лише допомагають, а й мотивують користувача.

Цифрові сервіси зазвичай «відчувають» потреби людини ще до того, як вона сама усвідомлює їх. Вони здатні ефективно оптимізувати складні процеси та рутинні обов'язки. У сучасній реальності, де технології стають частиною повсякденного життя, вебзастосунки відкривають шлях до вирішення навіть найскладніших проблем, виступаючи надійними цифровими партнерами.

Як зазначено в джерелі [2], однією з ключових переваг вебзастосунків є їхня доступність. Користувач може взаємодіяти з ними через будь-який пристрій із доступом до Інтернету, що забезпечує максимальну мобільність і зручність.

Фінансові операції також не обходяться без участі цифрових рішень.

Сучасний ритм життя вимагає швидких і простих способів управління грошима. Саме тому вебзастосунки стають дієвими інструментами для обліку доходів, витрат, формування бюджету та контролю заощаджень. Як підкреслюється у статті [3], розробка фінансових вебзастосунків є особливо актуальною в умовах сьогодення, коли важливо ефективно розпоряджатися ресурсами з урахуванням швидкості й зручності користування.

Розробка подібного вебзастосунку є особливо актуальною в умовах економічної нестабільності, коли зростає потреба в раціональному плануванні особистих фінансів. Цифрові інструменти дають змогу формувати адаптивні бюджети, які можна оперативно коригувати у відповідь на зміни у фінансовому становищі. Це забезпечує постійний контроль над витратами та доходами у зручному форматі — у будь-який момент і з будь-якого пристрою.

Крім того, на тлі зростаючого інтересу до фінансової грамотності, створення таких вебзастосунків позитивно впливає на підвищення обізнаності користувачів у сфері особистих фінансів. Як зазначено у джерелі [4], вміння складати бюджет є базовою складовою фінансової компетентності. Завдяки інтуїтивним інтерфейсам, аналітичним інструментам і вбудованим освітнім матеріалам користувачі отримують змогу глибше розуміти власне фінансове становище та приймати зважені рішення.

Отже, є ціла низка чинників, що обґрунтовують доцільність створення вебзастосунку для ведення бюджету та обліку заощаджень. Такий сервіс стане ефективним засобом для користувачів у процесі фінансового планування, сприятиме більшій стабільності та надаватиме зручні інструменти для контролю за особистими коштами. Крім того, він допоможе стежити за витратами й формувати відповідальне ставлення до управління фінансами.

1.2 Огляд існуючих рішень фінансових вебзастосунків

На сьогодні існує велика кількість застосунків, призначених для фінансового планування та моніторингу витрат. Як свідчать результати дослідження [5], близько 40% респондентів активно використовують подібні сервіси, що підтверджує їхню популярність серед користувачів. Кожен із цих застосунків намагається запропонувати щось унікальне, щоб привернути увагу цільової аудиторії.

Деякі застосунки орієнтовані на новачків і пропонують максимально простий та інтуїтивно зрозумілий інтерфейс, що дозволяє швидко розпочати роботу без необхідності вивчення складних інструкцій. Інші сервіси передбачають функціонал для ведення сімейного бюджету, що забезпечує спільне управління фінансами всіма членами родини.

Окрему категорію складають програми, які візуалізують фінансові дані - витрати та доходи - у вигляді графіків і діаграм. Це дозволяє користувачам наочно аналізувати свій фінансовий стан і робити обґрунтовані рішення щодо подальшого розподілу коштів.

Надалі буде детально розглянуто та проаналізовано окремі приклади таких застосунків.

1.2.1 Monefy

Monefy - це мобільний застосунок, який дозволяє ефективно управляти особистими фінансами. Його головна мета - зробити процес обліку витрат максимально простим і сприяти економії коштів.

Однією з ключових переваг програми є інтуїтивно зрозумілий інтерфейс. При відкритті застосунку одразу відображається діаграма, що ілюструє витрати за різними категоріями. Додавання нової транзакції - доходу або витрати - виконується в один дотик. Кожен запис можна супроводити примітками та прикріпити до відповідної категорії. Крім того,

програма дозволяє переглядати історію операцій у межах обраної категорії, а також здійснювати пошук за примітками та категоріями.

Користувачі, які оформили Premium-підписку, мають розширені можливості - наприклад, створення власних категорій витрат і доходів. На рисунку 1.1 представлено приклад інтерфейсу застосунку.

У розділі налаштувань користувач може встановити базову валюту. Преміум-функціонал дає змогу проводити операції в декількох валютах одночасно.

За замовчуванням, у застосунку вже передбачені два рахунки: "Готівка" та "Платіжна картка". Також можна створити додаткові рахунки з власними назвами та вказати початковий баланс. При наявності Premium-доступу для кожного рахунку можна обрати окрему валюту, відмінну від основної. Аналізувати бюджет можливо як окремо по кожному рахунку, так і загалом по всіх.

Рисунок 1.1 – Головна сторінка застосунку “Monefy”

Користувачам також доступна можливість перегляду фінансових даних

за різні періоди: день, тиждень, місяць, рік, увесь час або в межах конкретного проміжку чи вибраної дати.

Однією з додаткових функцій є режим «Бюджет», у якому користувач задає бажану межу витрат. У цьому режимі облік витрат не базується на вказаних доходах, а орієнтується на фіксовану суму, визначену користувачем заздалегідь. Такий підхід дозволяє зручно контролювати витрати, не перевищуючи встановленого бюджету. Інтерфейс застосунку в режимі «Бюджет» зображено на рисунку 1.2.

Рисунок 1.2 – Головна сторінка застосунку в режимі “Бюджет”

Застосунок Monefy підтримує функцію експорту даних у форматі .svg, однак слід враховувати, що зворотне завантаження інформації з цього файлу не передбачено.

Серед недоліків програми можна відзначити відсутність автоматичної синхронізації між різними пристроями через спільний обліковий запис. Проте користувачі, які оформили Premium-підписку, отримують доступ до функції синхронізації даних за допомогою хмарних сервісів Dropbox та Google Drive.

1.2.2 Goodbudget

Goodbudget - це сучасний фінансовий застосунок, який реалізує концепцію управління бюджетом за принципом «конвертів». Як зазначається у статті [6], суть методу полягає в тому, що користувач спочатку створює окремі "конверти" для різних категорій витрат, а потім розподіляє бюджетні кошти між ними. Такий підхід дозволяє краще контролювати фінанси та уникати надмірних витрат.

Завдяки Goodbudget відпадає потреба в традиційних паперових конвертах - усі розрахунки здійснюються у цифровому форматі. Згідно з рейтингом Forbes Advisor [7], цей застосунок входить до переліку десяти найкращих програм для ведення особистих бюджетів.

Під час створення конвертів користувач може обрати тип витрат - регулярні (щомісячні або щорічні) чи одноразові. Крім того, є можливість фіксувати інформацію про погашення боргів. Приклад інтерфейсу створення конвертів наведено на рисунку 1.3.

У застосунку Goodbudget користувачі мають змогу створювати різні типи рахунків - це можуть бути готівкові, банківські або кредитні рахунки. При додаванні нової транзакції можна вибрати її тип: витрата, дохід або переказ між наявними рахунками чи конвертами.

Під час фіксації витрат необхідно зазначити суму, обрати відповідний конверт і рахунок, з якого списуються кошти. Додатково можна внести номер чека та залишити примітку до фінансової операції.

Також передбачена можливість створення повторюваних транзакцій з

індивідуальним вибором інтервалів (щоденно, щотижнево тощо). Користувачі можуть активувати надсилання нагадувань про ці операції на електронну пошту, що підвищує зручність у плануванні регулярних витрат. Інтерфейс головної сторінки застосунку представлено на рисунку 1.4.

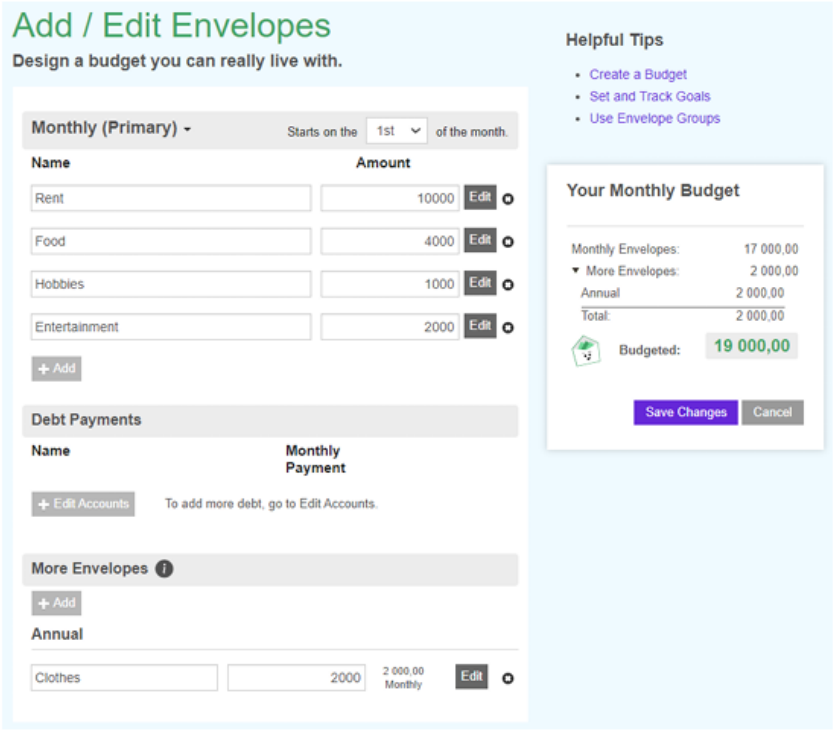

Рисунок 1.3 – Сторінка створення конвертів

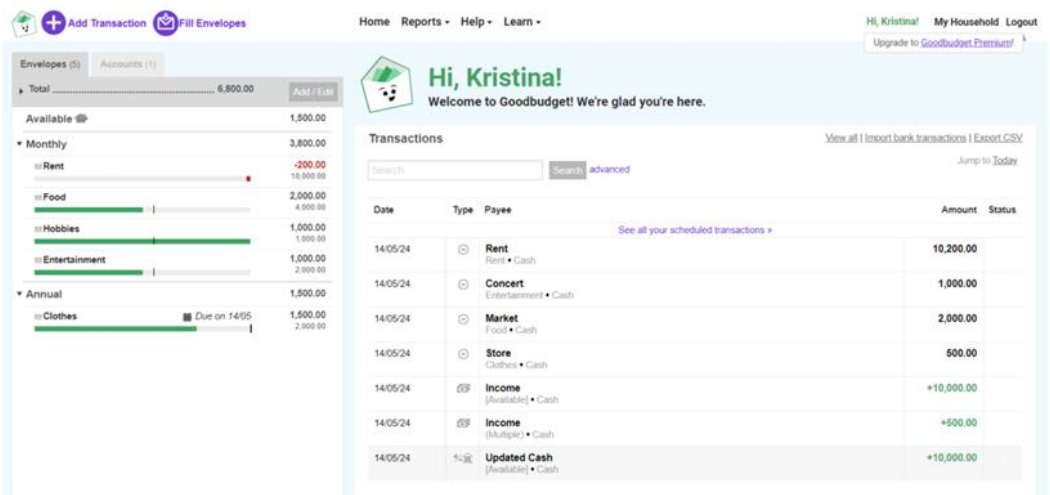

Рисунок 1.4 – Головна сторінка застосунку

Goodbudget пропонує розширений функціонал пошуку по транзакціях. Користувач може знаходити необхідні записи за ключовими словами, типом транзакції, назвою конверта чи рахунку, номером чеку, сумою, датою тощо, що значно полегшує навігацію у фінансових даних.

Також програма підтримує імпорт банківських виписок у форматах QFX та OFX, а експорт даних можливий у файл формату .svg.

Завдяки функції автоматичної синхронізації дані застосунку оновлюються в режимі реального часу на всіх пристроях користувача, що забезпечує зручність використання як на смартфоні, так і на комп’ютері.

1.2.3 Money Manager Money

Manager - це застосунок, призначений для контролю, аналізу та планування особистих витрат. Окрім фіксації фінансових операцій, він дозволяє користувачу формувати детальні звіти про витрати за потребою.

Після запуску програми на головній сторінці (рисунок 1.5) відразу відображається перелік усіх здійснених операцій, що забезпечує зручний огляд фінансових потоків. Для додавання доходів або витрат достатньо натиснути одну кнопку. При створенні транзакції користувач може вказати суму, вибрати категорію, обрати рахунок, через який проходить операція, а також додати примітки і прикріпити фотографію, наприклад, чека або квитанції.

Крім того, у застосунку доступна функція створення переказів між власними рахунками. Усі операції можна налаштовувати як повторювальні з вибором різних періодів регулярності.

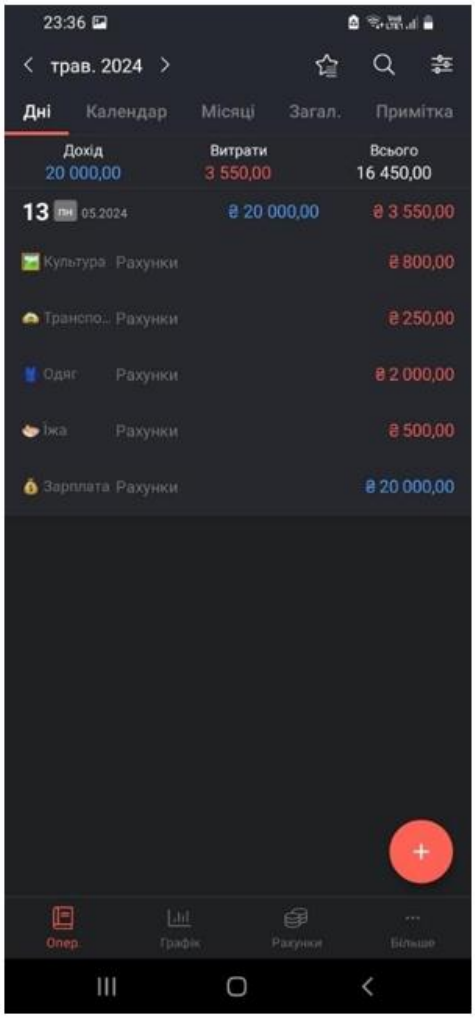

Рисунок 1.5 – Головна сторінка застосунку

Застосунок надає можливість переглядати фінансові операції за різні періоди: окремі дні, місяці або за весь час. Однією з унікальних функцій є відображення транзакцій у вигляді календаря, що дозволяє користувачам легко визначати дні з більшими або меншими витратами.

Крім того, всі операції можна фільтрувати за типом, категорією чи конкретним рахунком. Окрема сторінка (рисунок 1.6) містить діаграми, які відображають розподіл доходів та витрат, що допомагає краще аналізувати фінансовий стан.

Рисунок 1.6 – Сторінка графіку фінансових операцій

На окремій сторінці застосунок дозволяє створювати різні типи рахунків, відображати власні заощадження, інвестиції, позики та інші фінансові активи. Користувач може переглядати всі операції, пов’язані з рахунками, а також бачити поточний баланс і доступний залишок.

Також доступні графіки, які ілюструють динаміку фінансових операцій по рахунках за останні кілька місяців, що допомагає користувачу порівнювати і аналізувати зміни у своїх фінансах.

Для користувачів із Premium-підпискою передбачена функція підключення ПК Менеджера, що дозволяє управляти всіма даними безпосередньо з комп’ютера через Wi-Fi-з’єднання.

1.2.4 Spendee

Spendee - це застосунок, який допомагає легко контролювати особисті

фінанси через систему віртуальних гаманців, у яких користувач може фіксувати свої фінансові операції.

При створенні транзакції доступний вибір її типу: витрата, дохід або переказ між існуючими гаманцями. Користувач може вказати категорію операції, додати примітку, окрему позначку та обрати валюту. Категорії та позначки можна вибирати зі стандартного списку або створювати власні, персоналізовані варіанти.

Для зручного управління категоріями створена окрема сторінка, яка допомагає підбирати найвідповідніші варіанти під конкретні потреби користувача. Також можна додавати фотографії чеків або квитанцій до кожної транзакції.

Інтерфейс сторінки гаманця представлений на рисунку 1.7.

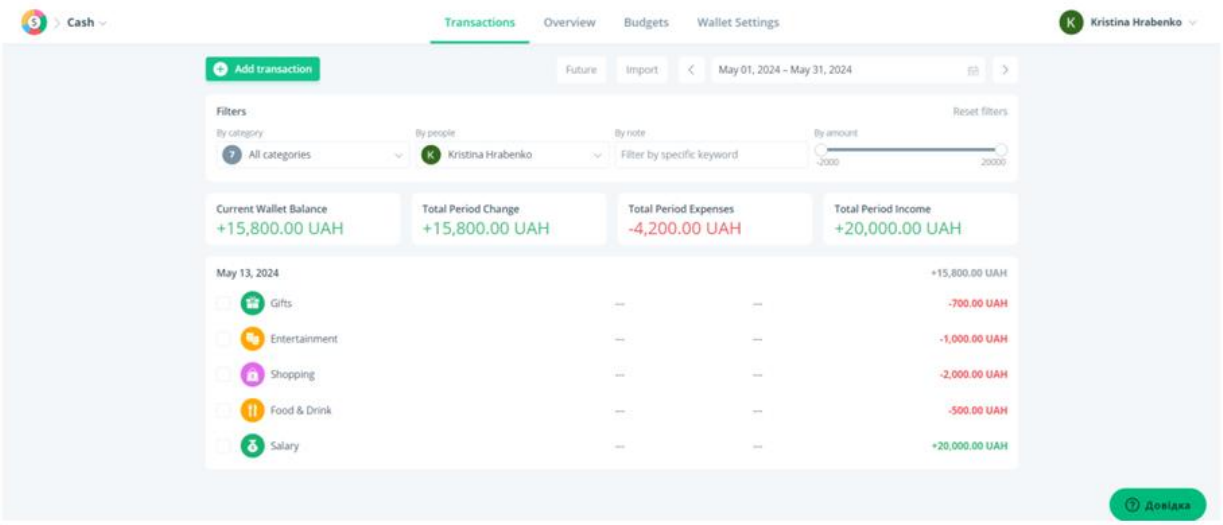

Рисунок 1.7 – сторінка транзакцій конкретного гаманця

Для користувачів із Premium-підпискою застосунок пропонує можливість додавати співучасників до гаманця, що дає змогу спільно відслідковувати всі транзакції. Біля кожної операції відображається ім’я користувача, який її створив.

Крім того, застосунок має розширені можливості фільтрації фінансових

операцій - за вибраними категоріями, авторами транзакцій, ключовими словами та сумою платежу.

1.2.5 Money Lover

Money Lover - це застосунок, який передбачає вибір головної мети користувача, серед яких відстеження платежів, управління боргами, скорочення витрат або накопичення коштів.

При створенні транзакції можна вказати тип операції, обрати категорію, додати нотатку, вибрати гаманець, з якого здійснюється операція, позначити подію, налаштувати нагадування та додати фотографію. Категорії доступні зі стандартного набору, але користувач також може створювати власні.

Крім того, операції можуть бути встановлені як повторювані.

Застосунок дозволяє формувати звіти у вигляді діаграм і графіків. Всі транзакції та звіти доступні на сторінці «Операції» (рисунок 1.8) і можуть переглядатися за різними часовими періодами: день, тиждень, місяць, квартал, рік або довільний інтервал.

Особливою рисою цього застосунку є система бюджетів, які створюються для кожної окремої категорії. Користувач встановлює ліміт витрат на певну категорію за місяць, який можна налаштувати як повторюваний.

На сторінці бюджетів (рисунок 1.9) доступна загальна діаграма, що відображає стан усіх бюджетів за вибраний місяць.

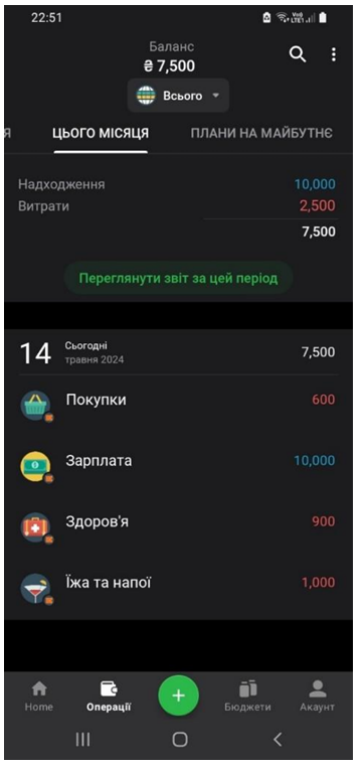

Рисунок 1.8 – Сторінка операцій застосунку

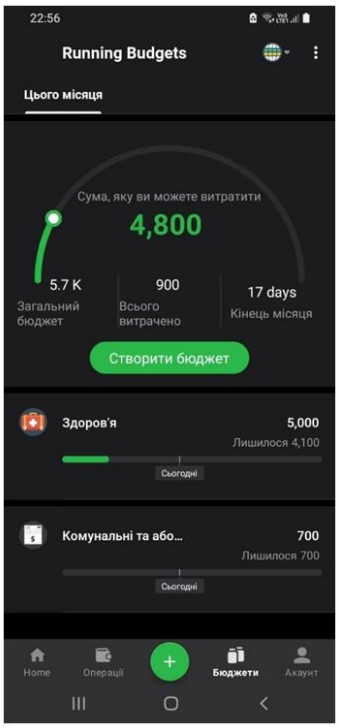

Рисунок 1.9 – Сторінка бюджетів застосунку

Користувач має змогу створювати різні рахунки та відслідковувати свої борги, які можуть бути як зобов’язаннями користувача, так і боргами, що інші мають повернути йому.

Під час створення гаманців застосунок дозволяє підключати банківські картки, хоча доступні варіанти банків обмежені. Крім того, передбачені спеціальні гаманці для кредитів та накопичень.

Для зручності обробки даних користувач може експортувати інформацію у форматі CSV або напряму до Google Sheets. При цьому можна обирати, які саме гаманці та типи операцій слід включити до експорту.

2 ЗАСОБИ РОЗРОБКИ

2.1 Фреймворк Flask

Flask - це легкий веб-фреймворк на мові програмування Python, який дає змогу оперативно створювати веб-додатки з мінімальними зусиллями. Його основною метою є забезпечення простого та гнучкого середовища для розробки, що робить його зручним вибором для невеликих і середніх проектів, а також для швидкого створення прототипів. Основна ідея Flask полягає в мінімалізмі: фреймворк пропонує лише базовий функціонал, надаючи розробнику свободу самостійно обирати додаткові модулі чи бібліотеки згідно з вимогами конкретного проєкту.

Ключовою функцією Flask є організація маршрутів - зв’язок між запитами користувача та відповідями, що повертаються у відповідь. Це дає змогу ефективно керувати вмістом веб-сторінок і API. Завдяки підтримці численних сторонніх розширень, можливості фреймворка можуть бути легко розширені - наприклад, для роботи з базами даних, автентифікації користувачів, обробки форм тощо.

Однією з головних переваг Flask вважається його лояльність до структури проєкту: він не вимагає суворого дотримання певних архітектурних шаблонів, що дозволяє розробникам організовувати код відповідно до власних уподобань. Це приваблює програмістів, які прагнуть мати повний контроль над своїми рішеннями. До того ж, детальна документація і жвава спільнота роблять Flask чудовим вибором навіть для початківців у сфері веб-розробки [3].

Фреймворк дозволяє оперативно створювати і тестувати веб-застосунки, що особливо цінно в умовах, де швидкість розробки є критичною. Його компактність і модульна структура сприяють зручному масштабуванню та інтеграції з іншими інструментами. Саме завдяки цим

якостям Flask здобув популярність серед сучасних веб-розробників, які шукають оптимальне поєднання простоти та функціональності.

2.2 Фреймворк SQLAlchemy

Flask-SQLAlchemy є додатковим модулем для Flask, який значно спрощує роботу з базами даних у веб-застосунках. Це розширення дозволяє застосовувати об'єктно-орієнтований підхід до взаємодії з базою даних, зменшуючи потребу в написанні прямого SQL-коду. Завдяки цьому, розробка стає простішою та зрозумілішою, що позитивно впливає на продуктивність.

Цей інструмент надає доступ до повного функціоналу бібліотеки SQLAlchemy, включаючи опис моделей, формування запитів, управління сесіями та реалізацію міграцій. Структура бази даних описується у вигляді класів Python, що полегшує її розробку й обслуговування. Крім того, Flask-SQLAlchemy автоматично встановлює з’єднання з базою даних і легко інтегрується з іншими елементами екосистеми Flask, що забезпечує злагоджену роботу додатка.

Однією з вагомих переваг використання Flask-SQLAlchemy є можливість створювати чистий, структурований і легко підтримуваний код. Об'єктно-орієнтований синтаксис робить код більш зрозумілим, особливо в масштабних проєктах із розгалуженою структурою даних. Ще однією важливою особливістю є зручність у проведенні міграцій баз даних - цей процес стає значно менш трудомістким, що особливо актуально для динамічних проєктів.

Завдяки підтримці кількох типів баз даних, таких як SQLite, MySQL, PostgreSQL та інших, Flask-SQLAlchemy може бути використаний у найрізноманітніших проектах. Його розширюваність та активна підтримка спільнотою розробників забезпечують надійну роботу з новими версіями Flask і SQLAlchemy. Саме тому це розширення користується популярністю серед фахівців, які прагнуть створювати ефективні та масштабовані веб-

рішення.

2.3 Фреймворк Alembic

Alembic - це Python-бібліотека, призначена для контролю версій баз даних та здійснення міграцій. Вона створена спеціально для використання разом із SQLAlchemy і значно полегшує процес відстеження змін у структурі бази даних. Завдяки Alembic розробники можуть впевнено управляти модифікаціями, що вносяться до схеми, з можливістю їх застосування або скасування у разі потреби.

Цей інструмент дає змогу вести історію змін бази даних, що є надзвичайно важливим у динамічному середовищі розробки, особливо в командних проєктах, де над системою можуть одночасно працювати кілька фахівців. Наявність контрольованої історії змін дозволяє швидко відкотитися до попереднього стану схеми, якщо виникає така потреба.

Однією з визначальних функцій Alembic є можливість автоматичного створення міграцій - система здатна генерувати потрібні інструкції для оновлення бази даних на основі змін у моделях SQLAlchemy. Це суттєво знижує навантаження на розробника, мінімізуючи ручну роботу та ймовірність помилок у міграційних скриптах.

Завдяки простому й зручному інтерфейсу, Alembic допомагає сконцентруватися на логіці додатка, не витрачаючи багато часу на технічні деталі оновлення схеми бази. Його можна ефективно застосовувати як у невеликих проектах, так і в складних корпоративних системах.

Ще однією сильною стороною Alembic є його тесна інтеграція з SQLAlchemy, що забезпечує безшовну взаємодію та узгодженість між Python-моделями та реальною структурою бази. Для розробників, які вже використовують SQLAlchemy, впровадження Alembic у робочий процес є природним і майже безболісним кроком.

2.4 Фреймворк Flask-WTF

Flask-WTF - це розширення для фреймворку Flask, яке забезпечує зручну інтеграцію з бібліотекою WTForms - потужним інструментом для створення та обробки веб-форм. Завдяки цьому модулю процес роботи з формами у Flask-додатках стає значно простішим і зрозумілішим, адже Flask-WTF поєднує функціонал WTForms із можливостями Flask, створюючи ефективний інструмент для організації форм.

Головна мета використання Flask-WTF полягає в полегшенні створення форм і перевірки введених даних. Розробники можуть описувати форми за допомогою Python-класів, замість написання HTML вручну. Це суттєво пришвидшує розробку. Крім того, Flask-WTF автоматично генерує HTML-код для форм і реалізує перевірку даних на серверному боці, що сприяє підвищенню безпеки застосунку - зокрема, гарантує правильність введення даних користувачами.

Серед ключових переваг цього розширення - тісна інтеграція з іншими компонентами Flask, такими як автентифікація та сесії. Це дозволяє без значних змін у коді додавати форми до вже наявних проєктів. Flask-WTF також реалізує захист від атак типу CSRF (міжсайтової підміни запитів), що додає ще один рівень безпеки при роботі з формами.

Додатково, розробники мають змогу налаштовувати форми під власні потреби: створювати індивідуальні поля, валідатори та логіку обробки. Завдяки цьому Flask-WTF є дуже гнучким і придатним як для простих, так і для складних проєктів. Широка документація та активна спільнота роблять цей інструмент доступним як для новачків, так і для досвідчених фахівців. Саме тому Flask-WTF часто обирають при створенні сучасних веб-застосунків на Flask.

2.5 Бібліотека Bcrypt

Bcrypt - це Python-бібліотека, яка застосовується для безпечного хешування паролів у веб-застосунках. Вона відіграє ключову роль у захисті облікових даних користувачів, забезпечуючи їх збереження у захищеному вигляді. Алгоритм, що використовується в Bcrypt, розроблений таким чином, щоб бути стійким до атак типу перебору (brute-force) та атак із використанням rainbow-таблиць.

Основне завдання Bcrypt - забезпечити надійне хешування паролів, яке суттєво ускладнює розкриття справжніх значень навіть у випадку компрометації бази даних. На відміну від звичайних хеш-функцій, цей алгоритм використовує унікальну соль (salt), яка додається до кожного пароля перед хешуванням. Завдяки цьому навіть ідентичні паролі матимуть різні хеші, що значно підвищує загальний рівень безпеки системи.

Серед переваг використання Bcrypt - висока стійкість до сучасних методів злому, а також можливість налаштувати рівень складності хешування. Це дозволяє адаптувати обчислювальні витрати до актуального рівня технічного розвитку. Bcrypt автоматизує додавання солі та повторне хешування, зменшуючи ризики, пов’язані з ручним налаштуванням алгоритмів захисту.

Завдяки поєднанню надійності, гнучкості та простоти впровадження, Bcrypt є популярним вибором серед розробників, які прагнуть забезпечити високий рівень захисту паролів у своїх веб-проєктах.

2.6 Бібліотека Bootstrap

Bootstrap - одна з найпоширеніших бібліотек стилів (CSS), яка значно полегшує створення адаптивного веб-інтерфейсу. Вона набула великої популярності серед веб-розробників завдяки простоті застосування, широкому набору готових компонентів і підтримці кросбраузерності. На

сьогодні Bootstrap вважається де-факто стандартом у розробці сучасних веб-застосунків.

У межах даної дипломної роботи використовувалася третя версія цієї бібліотеки - Bootstrap 3. Основний файл стилів bootstrap.css містить оформлення для базових HTML-елементів, зокрема кнопок, форм, таблиць, блоків навігації, заголовків та підзаголовків [4].

Однією з ключових особливостей Bootstrap 3 є використання сітки на 12 колонок, яка дозволяє формувати адаптивні макети сторінок. За допомогою спеціальних класів можна легко налаштовувати кількість колонок і їхню ширину залежно від розміру екрана. Це забезпечує зручне компонування контенту для різних пристроїв, включаючи смартфони, планшети та десктопи.

Бібліотека також містить розширений набір текстових стилів для форматування заголовків, абзаців, списків тощо. Є можливість вирівнювання тексту, зміни типографіки та інші засоби для підвищення зручності читання.

Форми в Bootstrap 3 мають заздалегідь визначені стилі, що покращують їхній вигляд та функціональність. Передбачені класи для текстових полів, кнопок, елементів введення, що дозволяє створювати різні типи форм — вертикальні, горизонтальні, з вбудованими елементами.

Завдяки принципам респонсивного дизайну, на яких побудовано Bootstrap 3, вебсторінки автоматично підлаштовуються під параметри екрана користувача, що підвищує зручність перегляду контенту на різних пристроях [5].

2.7 Система контролю версій Git

Git - це система керування версіями, яка дозволяє розробникам контролювати процес розробки програмного забезпечення, фіксувати внесені зміни та організовувати спільну роботу над проєктом. За допомогою Git можна зберігати різні стани файлів, перемикатися між гілками, об'єднувати

їх, а також відслідковувати, хто і коли вніс певні зміни.

У системі всі зміни фіксуються у вигляді комітів, кожен з яких зберігає інформацію про автора, дату та вміст змін. Git базується на моделі гілкування та злиття (branching & merging), що дозволяє гнучко управляти різними напрямами розробки. Для забезпечення цілісності даних використовуються криптографічні механізми, які зв’язують коміти з відповідними авторами та їхніми діями.

Особливістю Git є локальна обробка змін: усі операції відбуваються на комп’ютері розробника, а синхронізація з віддаленим репозиторієм виконується лише за потреби. Це забезпечує автономність та швидкість роботи.

У межах проєкту використовувався популярний хостинг-контроль версій - GitHub. Цей сервіс дозволяє користувачам створювати віддалені репозиторії, в яких зберігається код, історія змін, документація тощо. Репозиторії можуть бути публічними (доступними для перегляду та редагування всіма користувачами) або приватними, коли доступ обмежується тільки автором або запрошеними учасниками команди.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Архітектура застосунку

Розроблений застосунок Expense Tracker побудований за принципами багаторівневої архітектури з використанням підходу Domain-Driven Design (DDD). Це дозволяє чітко розділити відповідальності між частинами системи, що спрощує підтримку, розширення й тестування. Застосунок має чотири основні шари:

1. Domain (Доменна логіка)

Це "серце" застосунку, де зосереджена основна бізнес-логіка. Головна модель — Transaction, яка описує фінансову операцію. Вона містить такі дані:

- Id — унікальний ідентифікатор;
- Amount — сума;
- Date — дата створення;
- Description — опис;
- TransactionType — тип (витрата, дохід, переказ);
- FromAccountId, ToAccountId — рахунки (для переказів);
- Category — категорія транзакції.

Валідація (наприклад, перевірка, щоб сума не була від’ємною) реалізована безпосередньо в цьому шарі або через окремі перевірки (специфікації).

2. Core (Application Layer)

Цей шар відповідає за зв'язок між логікою й інфраструктурою. Основні компоненти:

- TransactionService — керує створенням, оновленням і видаленням транзакцій, а також оновлює баланси акаунтів;
- інтерфейси (ITransactionRepository, IAccountService) — дозволяють

легко змінювати реалізацію без змін у логіці;

- DTO та сервіси обробки запитів.

3. Infrastructure (Інфраструктура)

Тут реалізовано роботу з базою даних PostgreSQL за допомогою Entity Framework Core. Основні складові:

- TransactionRepository — зберігає та читає дані про транзакції, акаунти й категорії;
- AppDbContext — описує зв’язки між таблицями, налаштування полів і обмежень;
- міграції та ініціалізація бази даних;
- за потреби тут також можна підключити логування або сторонні сервіси.

4. Presentation (WPF UI)

Інтерфейс користувача побудований на WPF з використанням шаблону MVVM — це дозволяє розділити логіку, уніфікувати прив'язку даних та забезпечити простоту тестування.

Ключові елементи інтерфейсу:

- форма додавання нової транзакції (з вибором типу, рахунку, категорії тощо);
- таблиця з переглядом усіх транзакцій, із фільтрами та сортуванням;
- екран перегляду залишку по акаунтах;
- повідомлення про помилки або успішні дії.

Конфігурація запуску:

Під час запуску застосунку відбувається:

- ініціалізація залежностей через DI-контейнер;
- підключення до PostgreSQL і конфігурація AppDbContext;
- автоматичне застосування міграцій (оновлення схеми бази, якщо потрібно) (лістинг 3.1).

Лістинг 3.1 – Автоматичне застосування міграцій під час запуску застосунку

```
using (var scope = serviceProvider.CreateScope())
{
    var dbContext =
scope.ServiceProvider.GetRequiredService<AppDbContext>();
    dbContext.Database.Migrate();
}
```

Дані для з’єднання з базою зберігаються в appsettings.json або окремому конфігураційному файлі WPF-застосунку.

3.2 Реалізація функціональності

Основні доменні сутності системи.

У процесі розробки системи обліку особистих фінансів були реалізовані ключові доменні моделі, які відповідають основним поняттям предметної області: акаунти, бюджети, категорії та повторювані транзакції. Уся логіка побудована на основі принципів Domain-Driven Design (DDD) із чітким поділом на доменну модель, інфраструктуру та інтерфейс користувача (WPF).

1. Модель Account (рахунок)

Модель Account зберігає інформацію про рахунок користувача:

- назва рахунку;
- поточний баланс (Money — окрема структура для суми з валютою);
- коментар (Note);
- ознака виключення з підрахунків (IsExcluded);
- дата створення та останнього оновлення.

Зміни рахунку автоматично оновлюють поле ModifiedAt для фіксації останніх змін.

EF Core мапить поля типів decimal, bool, timestamp, string на відповідні типи PostgreSQL (numeric, boolean, timestamp with time zone, text).

2. Модель Budget (бюджет)

Модель Budget описує бюджетні ліміти в межах певного періоду:

- назва бюджету;.
- ліміт витрат (SpendingLimit);
- поточна сума витрат (CurrentSpending);
- прив’язані категорії;
- період дії (дата початку та завершення).

Додані перевірки: дата початку не може бути після дати завершення, ліміт має бути додатнім.

Складні типи (Money) мапляться на колонки Amount + CurrencyCode.

3. Модель Category (категорія)

Модель Category представляє тип доходу або витрат:

- назва категорії;
- ознака IsIncome, що вказує — це дохід чи витрата.

Методи оновлення дозволяють змінювати назву або тип категорії. Категорії використовуються для фільтрації та групування транзакцій у звітах.

4. Модель RecurringTransaction (повторювана транзакція)

Ця модель реалізує механізм періодичних транзакцій:

- сума (Money);
- тип транзакції (витрата, дохід, переказ);
- періодичність (наприклад, щомісяця);
- дата останнього виконання;
- пов’язані акаунти та категорії.

Підтримуються обмежені (з кінцевою датою) або нескінченні повторення. PostgreSQL дозволяє працювати з такими типами даних за допомогою interval, timestamp, boolean.

Реалізація репозиторіїв:

Усі репозиторії реалізовано через Entity Framework Core і підтримують повний набір CRUD-операцій.

AccountRepository

- GetAllAsync() — отримати всі рахунки;

- GetActiveAccountsAsync() — фільтрація без IsExcluded;
- GetByIdAsync(int id) — пошук за ID;
- AddAsync(), UpdateAsync(), DeleteAsync() — стандартні дії.

BudgetRepository

- повертає бюджети разом із пов’язаними категоріями (Include);
- підтримує повноцінний CRUD .

CategoryRepository:

- простий CRUD для категорій (додати, змінити, видалити, отримати).

RecurringTransactionRepository:

- повертає всі повторювані транзакції з прив’язаними об’єктами;
- стандартні методи пошуку, додавання, оновлення та видалення.

Контекст бази даних (DatabaseContext)

Клас AppDbContext відповідає за конфігурацію моделі даних:

- імена таблиць і колонок використовують нижній регістр (accounts, categoryid);
- зв’язки описані через Fluent API (HasForeignKey, WithMany, OnDelete);
- вбудовані типи (OwnsOne) застосовуються для складних типів (Money);
- автоматичне оновлення CreatedAt і ModifiedAt у всіх сутностях (лістинг 3.2).

Лістинг 3.2 - Автоматичне встановлення CreatedAt та ModifiedAt у методі SaveChangesAsync

```
public override Task<int> SaveChangesAsync()
{
    var now = DateTime.UtcNow;
    foreach (var entry in ChangeTracker.Entries<EntityBase>())
    {
        if (entry.State == EntityState.Added)
            entry.Property("CreatedAt").CurrentValue = now;
        if (entry.State == EntityState.Modified)
            entry.Property("ModifiedAt").CurrentValue = now;
```

```
        }
        return base.SaveChangesAsync(cancellationToken);
    }
```

Ця структура дозволяє гнучко управляти даними, забезпечує точність для фінансових розрахунків та спрощує масштабування проєкту.

3.3 Тестування застосунку

Після завершення реалізації основної функціональності застосунку Expense Tracker було проведено ручне тестування з метою перевірки коректності роботи основних сценаріїв. Тестування охоплювало перевірку WPF-інтерфейсу (рисунок 3.1), правильність взаємодії з базою даних PostgreSQL (рисунок 3.2) через репозиторії, а також логіку оновлення балансу акаунтів.

Основні тестовані сценарії:

Створення нової транзакції (витрата, дохід, переказ)

Data Output | Messages | Notifications

	id [PK] integer	amount numeric (18,2)	date timestamp without time zone	note text	categoryid integer	accountid integer	iscleared boolean	type integer	targetaccountid integer	createdat timestamp without time zone	modifiedat timestamp without time zone	currencycode character varying (3)
1	20	50.00	2025-06-05 18:07:46.69674	burger	2	16	false	0	[null]	2025-06-05 15:08:12.568419	[null]	USD
2	21	20.00	2025-06-05 18:09:13.43427	adwawd	2	16	false	2	17	2025-06-05 15:09:20.377268	[null]	USD
3	22	20.00	2025-06-05 18:57:51.3835	dawdawd	11	16	false	0	16	2025-06-05 15:58:19.378526	[null]	USD
4	23	50.00	2025-06-05 18:58:29.696009	dwadwa	5	16	false	1	16	2025-06-05 15:58:37.007331	[null]	USD
5	24	35.00	2025-06-05 18:58:43.717417	adwawd	2	16	false	2	17	2025-06-05 15:58:50.765858	[null]	USD

Рисунок 3.1 - Скрін із самого застосунку

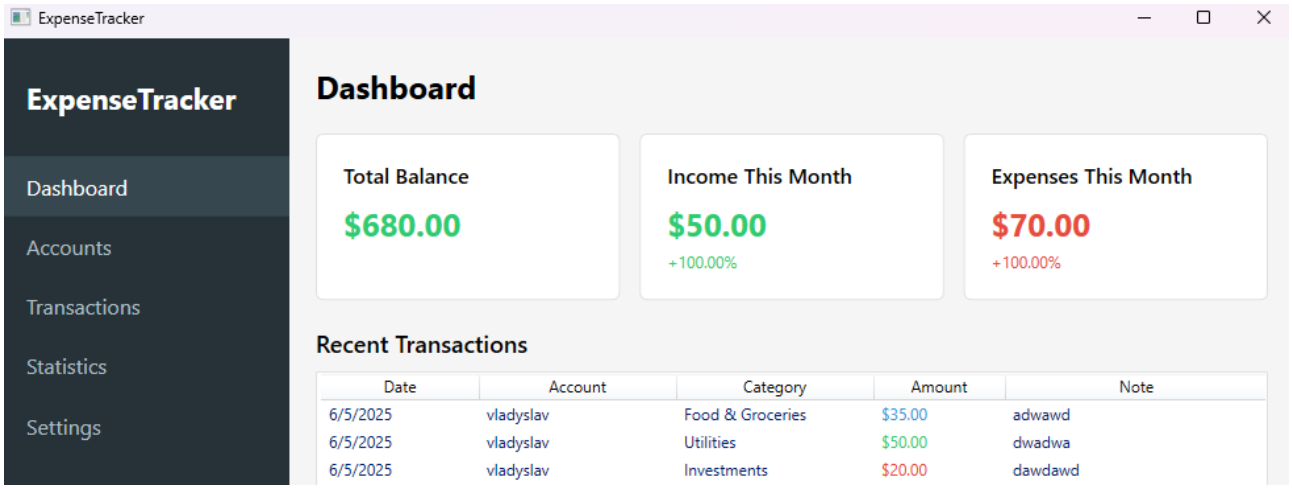

Рисунок 3.2 - Скрін із PostgreSQL

Очікуваний результат: транзакція успішно зберігається у базі даних, баланс акаунтів оновлюється згідно з типом операції (витрати зменшують баланс, доходи збільшують, переказ змінює баланс двох акаунтів).

Спроба збереження транзакції з порожніми або некоректними полями (рисунок 3.3).

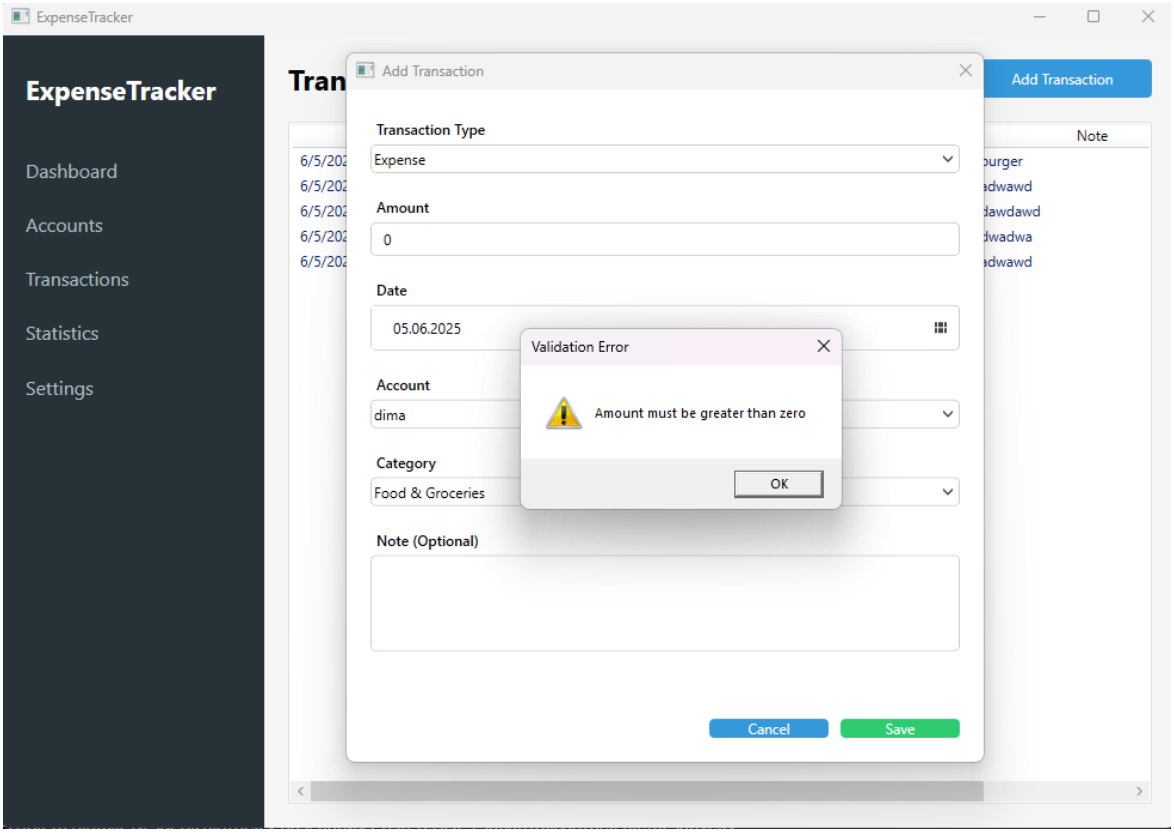

Рисунок 3.3 - Помилка транзакції

Очікуваний результат: транзакція не зберігається, відображається повідомлення про помилку валідації (наприклад, "Сума має бути додатнім числом", "Оберіть акаунт").

Введення некоректної суми (від’ємне значення, нечисловий текст) (рисунок 3.4).

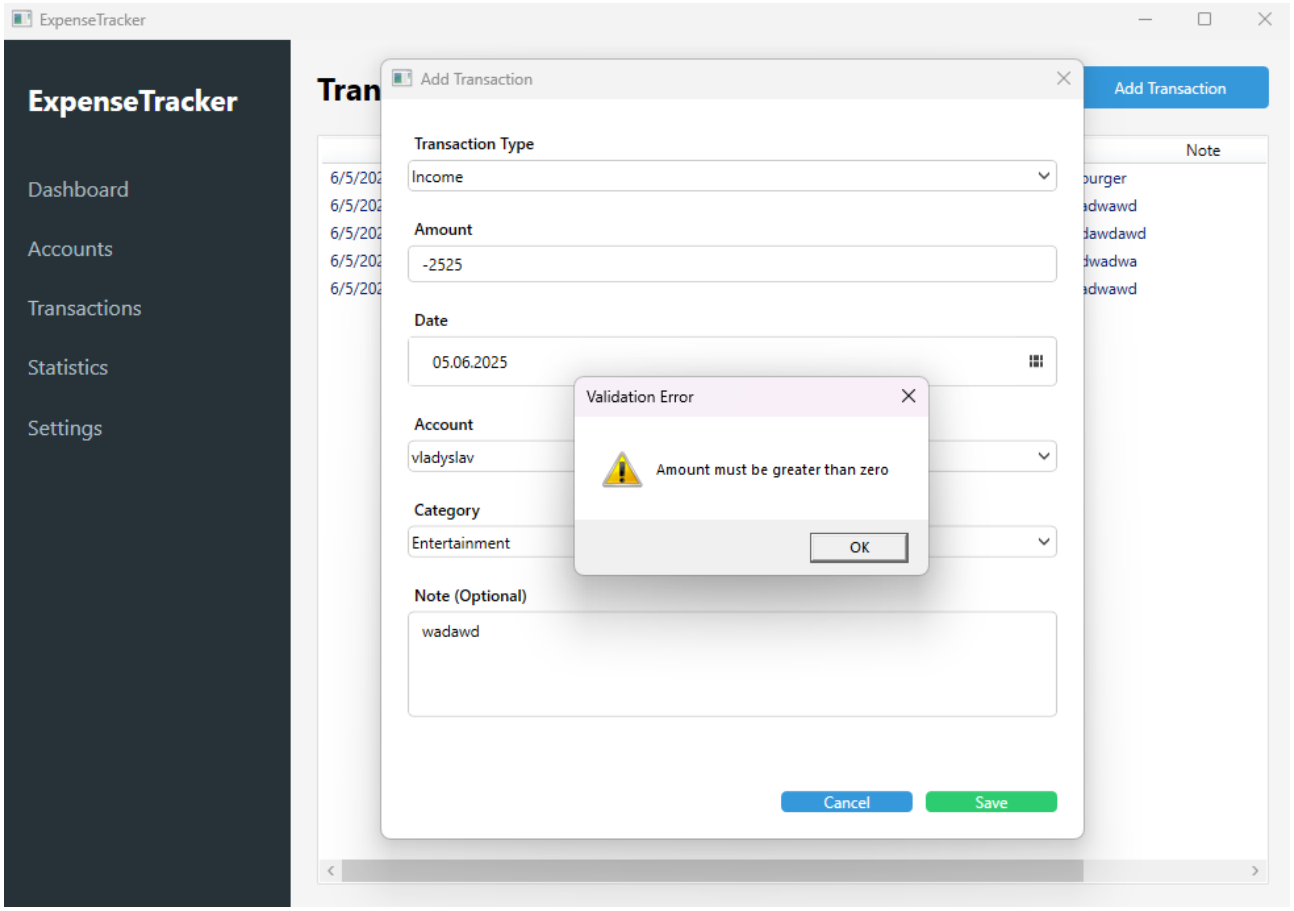

Рисунок 3.4 - Некоректна сума

Очікуваний результат: з’являється повідомлення про помилку, транзакція не додається.

Перевірка сортування транзакцій за датою (рисунок 3.5, 3,6).

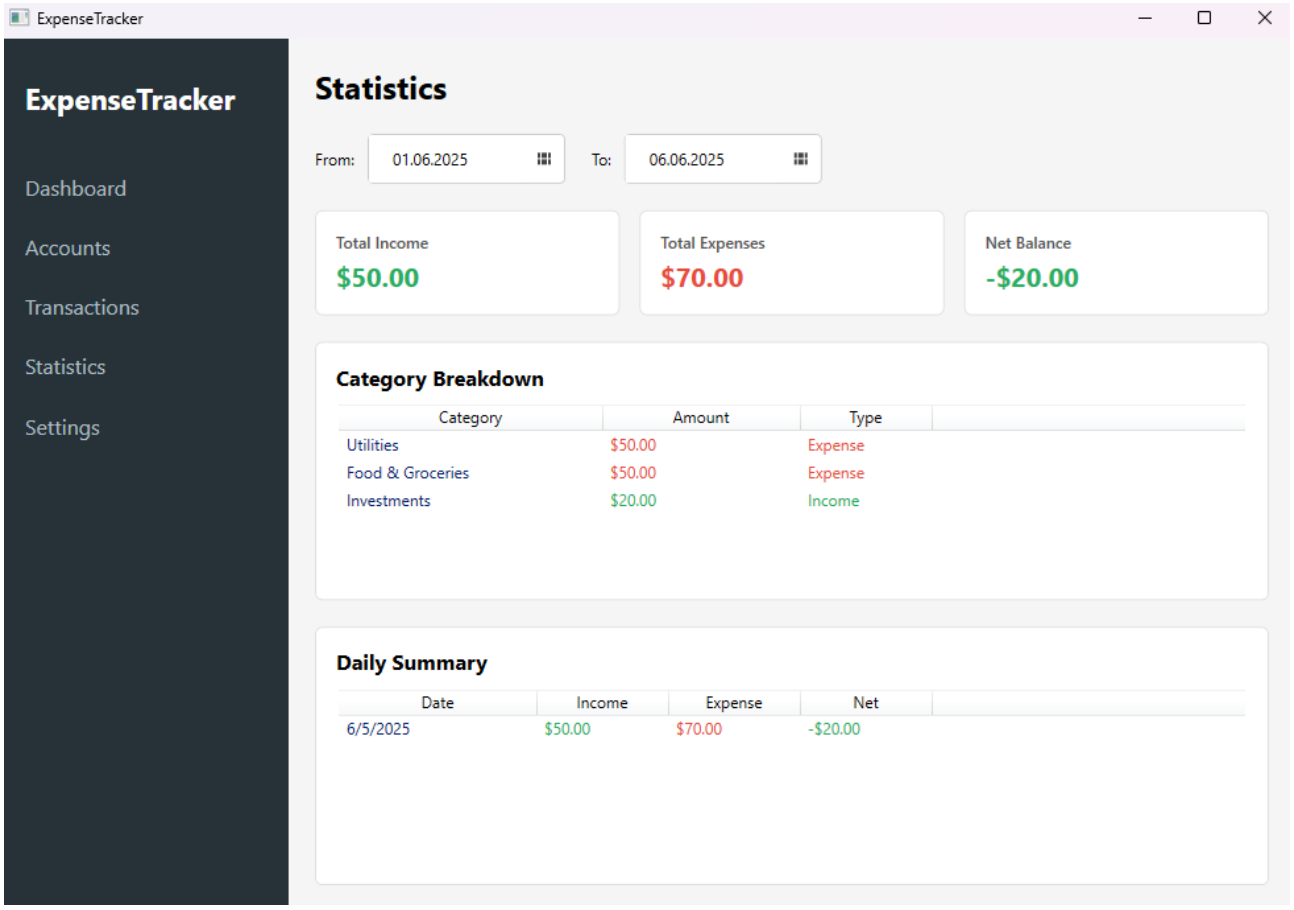

Рисунок 3.5 - Сортування за датами

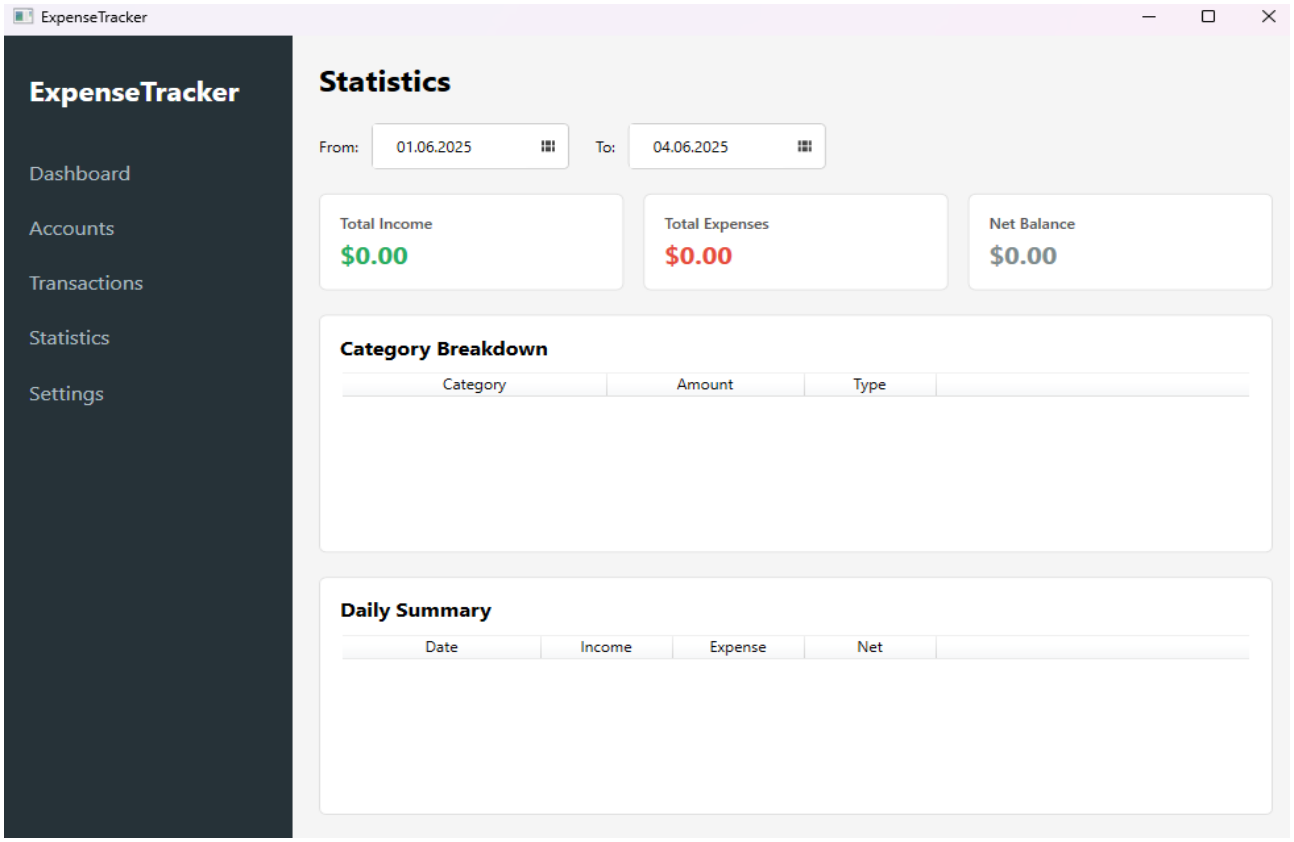

Рисунок 3.6 - Сортування

Очікуваний результат: список транзакцій у UI оновлюється та сортується у порядку спадання дати (остання транзакція — зверху).

Перевірка збереження даних при повторному запуску застосунку (рисунок 3.7).

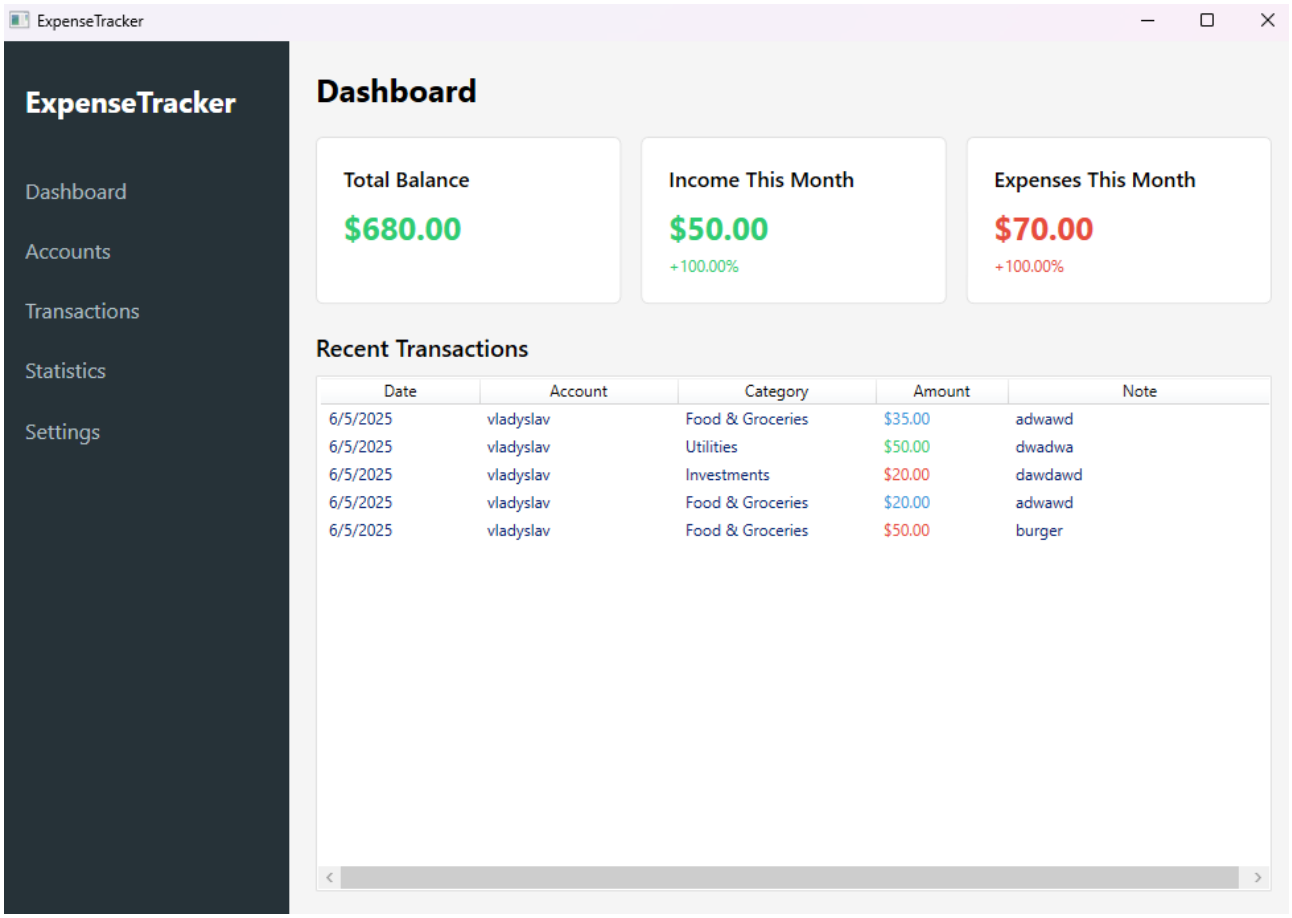

Рисунок 3.7 - Збереження

Очікуваний результат: усі акаунти, транзакції, категорії та бюджети зберігаються у базі даних PostgreSQL і завантажуються після запуску.

Результати тестування:

Усі ключові функції працюють стабільно:

- транзакції додаються;
- редагуються;
- видаляються;
- баланси оновлюються коректно;
- валідація на стороні UI та сервісів забезпечує захист від

некоректного вводу;

- не виявлено критичних помилок у взаємодії з PostgreSQL;
- дані зберігаються надійно між сесіями запуску застосунку;
- усі тестові сценарії пройдені успішно.

У процесі розробки настільного застосунку Expense Tracker було реалізовано базову функціональність для обліку особистих фінансів. Застосунок дозволяє користувачу:

- додавати нові транзакції (витрати, доходи, перекази) із зазначенням опису, суми, акаунта, категорії та дати;
- переглядати всі транзакції у вигляді списку;
- бачити актуальний баланс по кожному акаунту;
- зберігати всі фінансові дані в базі даних PostgreSQL.

Архітектура застосунку побудована відповідно до принципів Domain-Driven Design (DDD) та чітко поділена на логічні шари:

- Domain – містить основні бізнес-об’єкти (транзакції, акаунти, типи транзакцій тощо);
- Core (Application) – реалізує сервіси та бізнес-логіку, зокрема логіку оновлення балансів;
- Infrastructure – відповідає за взаємодію з базою даних через реалізацію репозиторіїв;
- Presentation (WPF) – забезпечує взаємодію з користувачем через графічний інтерфейс.

Використання WPF дозволило створити зручний, локальний графічний інтерфейс для настільного використання, а застосування PostgreSQL забезпечує надійне збереження даних і потенціал для масштабування.

ВИСНОВКИ

В ході виконання даної роботи розроблено веб-застосунок, який дозволяє користувачам ефективно керувати своїми фінансами, вести облік доходів та витрат, аналізувати фінансовий стан та планувати майбутні витрати, шляхом перелічених функціональних можливостей:

- створення власних облікових записів;
- можливості імпортувати дані з csv файлу або внесення вручну;
- створення місячних планів та внесення витрат;
- моніторингу витрат у вигляді таблиці або графіків.

Проведено детальне дослідження предметної області управління особистим бюджетом та аналіз існуючих програмних систем з цієї області. Цей аналіз дозволив зрозуміти основні проблеми та потреби користувачів у сфері фінансового управління.

За результатом аналізу існуючих програмних систем було виявлено як позитивні, так і негативні аспекти їхньої реалізації. Цей аналіз допоміг врахувати кращі практики та уникнути помилок при розробці власного веб-застосунку.

Розроблене програмне забезпечення було піддане тестуванню, яке підтвердило правильну роботу системи. Були проведені різні види тестів, включаючи модульні, інтеграційні тести, а також тести в реальних умовах.

Розроблена система може бути використана будь-якою особою, яка бажає покращити свої навички керування особистими фінансами, шляхом планування та відстеження своїх доходів і витрат.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Greenberg Miguel. Flask Web Development: Developing Web Applications with Python: A Guide, Sebastopol, USA, ed. O'Reilly Media, 2018. 308 с.

2. Справочник CSS [Електронний ресурс] — Режим доступу: http://htmlbook.ru/css

3. Офіційний сайт Bootstrap 3 [Електронний ресурс] — Режим доступу: http://bootstrap-3.ru/index.php Дата доступу: 01.06.2017

4. Swigart Al, Automate boring things with Python: Practical Programming for Beginners: A Guide. San Francisco, USA, ed. No Starch Press, 2015. 244 с.

5. Fiale Chris, SQL: A guide to learning the language: a manual. Berkeley, USA, ed. Peachpit Press, 2003. - 456 с.

6. Когаловский М. Р. Энциклопедия технологий баз данных / М. Р. Когаловский. — М.: Финансы и статистика, 2002. — 800 с.

7. David M. Beazley and Brian C. Jones. Python Cookbook: Recipes for mastering Python 3: A guide. Sebastopol, USA, publisher. O'Reilly Media, 2013. 303 с.

8. Єрошенко О.А., Балабанов Р.М. Застосунок для відстеження витрат та доходів. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: тези доповідей 15 міжнародної науково-технічної конференції 24-25 квітня 2025 року. Баку, Харків, Жиліна. 2025. С. 137.