

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Застосунок для планування бюджету»

Виконав студент групи ПЗ-21-1
ПРОКОПЧУК Олександра
Сергіївна

Керівник роботи:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент:
ЛОКТИКОВА Тамара Миколаївна

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна Вакалюк

«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ПРОКОПЧУК Олександра Сергіївна**

Керівник роботи: **Вакалюк Тетяна Анатоліївна**

Тема роботи: **«Застосунок для планування бюджету»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи:

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А	14 лютого 2025 р.	14 лютого 2025 р.
2	Вакалюк Т.А	14 лютого 2025 р.	14 лютого 2025 р.
3	Вакалюк Т.А	14 лютого 2025 р.	14 лютого 2025 р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб-застосунку для планування бюджету та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить 59 сторінок, 24 ілюстрації та 24 таблиці. Загальний обсяг 108 сторінок.

Метою роботи є створення додатку для моніторингу особистих фінансів з впровадженням у систему новітніх технологій, що відповідали за реалізацію, як візуальної, так і серверної складової.

Робота має у наявності чітко описати ключові завдання на розробку системи. Проведення порівняння конкурентно спроможних аналогів серед такого типу платформ. Було обрано клієнт-серверну архітектуру, фронтенд частина додатку була побудована на використанні React та Bootstrap, що дозволяє створювати адаптативний інтерфейс та інтерактивні елементи в ньому, а серверна частина була написана з використання Node.js, що забезпечує високофункціональність та швидкодійність.

Веб-застосунок надає клієнтам різними способами створювати транзакції, взаємодіяти з ними, встановлювати фінансові цілі, отримувати розширену статистику доходів і витрат у графічному вигляді.

КЛЮЧОВІ СЛОВА: ВЕБДОДАТОК, УПРАВЛІННЯ, ФІНАНСИ, ТРАНЗАКЦІЇ, ПЛАТЕЖІ, АВТОМАТИЗАЦІЯ, REACT, BOOTSTRAP.

					ІПЗ.КР.Б-121-25-ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Прокопчук О.С.			Застосунок для планування бюджету			Літ.	Арк.	Аркушів	
Перевір.		Вакалюк Т. А.								3	109
Керівник								Житомирська політехніка, група ІПЗ-21-1			
Н. контр.		Тичина О.П.									
Зав. каф.		Вакалюк Т. А.									

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

VS Code — Visual Studio Code

СУБД – система управління базами даних

ПЗ – програмне забезпечення

БД – база даних

CRUD – основні операції (Create, Read, Update, Delete), які виконуються з даними у системі

HTML — HyperText Markup Language

HTTP — Hypertext Transfer Protocol

JWT — JSON Web Token

JSON — JavaScript Object Notation

OCR – Optical Character Recognition

SPA – Single Page Application

					ІПЗ.КР.Б-121-25-ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Прокопчук О.С.			Застосунок для планування бюджету		Літ.	Арк.	Аркушів
Перевір.		Вакалюк Т. А.						4	109
Керівник				Житомирська політехніка, група ІПЗ-21-1					
Н. контр.		Тичина О.П.							
Зав. каф.		Вакалюк Т. А.							

ЗМІСТ

РЕФЕРАТ	3
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	4
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ЗАСТОСУНКУ ПЛАНУВАННЯ БЮДЖЕТУ	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту	12
1.3. Вибір архітектури додатку	18
1.4. Обґрунтування вибору інструментальних засобів.....	20
Висновок до першого розділу	23
РОЗДІЛ 2. ПРОЕКТУВАННЯ ЗАСТОСУНКУ ДЛЯ ПЛАНУВАННЯ БЮДЖЕТОМ.....	24
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури застосунок для планування бюджетом	24
2.2. Розробка бази даних системи	30
2.3 Проектування та реалізація алгоритмів роботи системи.....	36
2.4 Реалізація функціоналу веб-додатку для планування бюджету	39
Висновки до другого розділу.....	43
РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ	45
3.1 Структура інтерфейсу веб-додатку для планування бюджету	45
3.2 Статистика зборів та загальна статистика платформи	52
3.3 Тестування платформи	55
Висновки до третього розділу	57
ВИСНОВКИ.....	59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	60

		Прокопчук О.С.,			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ..... 63

Додаток А 64

Додаток Б..... 65

Додаток В..... 98

ВСТУП

Актуальність теми полягає у тому, що велика кількість населення країни потребує підвищення рівня своєї фінансової обізнаності у зв'язку із низкою економічних крик, що відбулась в Україні. У довгостроковій перспективні підвищується відсоток соціально-економічних ризиків, що призводить до слабкої конкуренції нашої країни з розвиненими державами. Коли використання цифрових технологій є повсякденним у житті громадян – застосунок для планування бюджету, моніторинг особистих коштів дозволяє автоматизувати процес аналізу бюджету. Веб-додаток потенційно може допомогти досягати фінансові цілі, які встановлює клієнт.

Мета випускної кваліфікаційної роботи бакалавра є розробка інноваційно-цифрового вирішення, яке спрямоване на підвищення відсотку грамотності громадян у фінансовій культурі. Проектування додатку фінансового трекеру надає користувач можливість структурувати платежі, здійснювати контроль доходів та витрат, а разом із тим збільшити ймовірність досягнення матеріальних цілей та фінансової стійкості.

Завданням випускної кваліфікаційної роботи бакалавра є реалізація застосунку з використання найновітніші технологій у комбінуванні з унікальним функціоналом, що забезпечить покриття всіх вимог вибагливих користувачів. Щоб досягти цього результат, треба провести детальний аналіз програмних забезпечень, що представлені на рингу суміжних додатків. Після отримання бажаних результатів, а саме розкрити сильні і слабкі сторони аналогів, що можна буде вдосконали та створення унікального функціоналу, що ще не був представлений на ринку еквівалентів.

Спроектувати серверну і клієнтську частину додатку, базу даних з використанням інноваційний інструментів та продумаю бізнес-логікою, що допомагає створити більш конкурентоспроможний продукт серед подібних застосунків.

Об'єктом дослідження є платформа для керування та відстеження особистих фінансів користувача, що включає створення безпечних взаємодій

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

із чутливими даними та забезпечення користувача зручним та зрозумілим зовнішнім виглядом програмного забезпечення.

Предметом дослідження є веб-застосунок для планування особистого бюджету, а саме його структура, функціональні модулі, способи автоматизації введення даних та мінімізація активного втручання клієнта, алгоритми оброблення статистичних фінансових даних. Акцентування уваги на візуального відображенню системи, є важливим кроком у реалізації продукту, надання йому інтуїтивно-зрозумілого інтерфейсу, що є легким у сприйнятті незалежно від віку та досвіду споживача.

За тематикою дипломної роботи були опубліковані тези [1-2].

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ЗАСТОСУНКУ ПЛАНУВАННЯ БЮДЖЕТУ

1.1. Постановка задачі

За нинішніх обставин стрімкого розвитку цифрових технологій та підвищення рівня активності фінансового сектору, формується попит користувачів на відстеження та управління особистими фінансами. Впродовж останніх декількох років Україна зазнала низку економічних криз, зокрема пандемію COVID-19, повномасштабну війну, зростання кредитної залежності населення та інфляційна нестабільність, що викликало зменшення рівня фінансової грамотності серед населення країни [3-4]. В межах зазначеної проблематики, громадяни потребують розробку практичного рішення, що сприятиме ефективнішому контролю за доходами та витратами, а також стане в пригоді для фінансового планування. Розробка подібного додатку потребує впровадження інноваційних програмних рішень, які об'єднують зручність користування, функціональність та інтеграційні можливості з іншими платформами або сервісами.

Кінцевий продукт розробки – це застосунок для планування бюджету, який позитивно впливатиме на ступінь володіння фінансовими знаннями серед населення країни та підвищить ефективність контролю бюджету, через передбачуване впровадження зручного та зрозумілого інтерфейсу. Зосередженість на таких тенденціях у проектуванні додатків схожого призначення, демонструє інноваційний підхід до сучасно-цифрових технологій та автоматизація рутинних справ по відношенню до грошових потоків.

Ключовими етапами проектування програмного забезпечення є:

1. Розробка візуального інтерфейсу;
2. Побудова архітектури системи;
3. Реалізація основного функціоналу;
4. Тестування продукту та його вдосконалення.

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

Початкова стадія проекту потребує розробки зовнішньої оболонки, що на пряму взаємодіє з користувачем системи. Пріоритетний напрямок – створення інтуїтивно зрозумілого інтерфейсу незалежно від віку та досвіду споживача, що відповідає сучасним стандартам UI/UX-дизайну. Для надання застосунку конкурентоспроможності серед інших представників на ринку зі схожим функціоналом, передбачається використання сучасних фреймворків, таких як React[5] та Bootstrap[10], що покращують адаптивність на різних пристроях та забезпечують раціональне використання часу для створення багатофункціональних інтерфейсів користувача, інтерактивних елементів на сторінці. З метою мінімізувати відсоток потенційних недоліків, які можуть виникнути при взаємодії користувача з додатком, було використано результати тестування для майбутнього покращення продукту.

Значну увагу на етапі створення архітектури системи було приділено питанню безпеки даних користувачів аби уникнути несанкціонованого витоку приватної інформації, що є особливо важливим у питаннях управліннь фінансами. Задіяно різноманітні технології, що підвищують рівень безпеки, такі як: JWT-токен [11] при аутентифікації, хешування паролів за допомогою bcrypt, валідація даних та фільтрація запитів за ідентифікатором користувача, що гарантують доступ лише до персональних даних. Важливо відмітити використання протоколу HTTPS [12] між клієнтом та сервером, який запобігає перехопленню даних в лиці осіб, що не мають відповідного права доступу.

Клієнтська частина охоплює питання з реєстрації нового користувача, редагування його даних, розширених можливостей взаємодії з транзакціями, категоризації товарів, створення цілей, додатковий функціонал інтеграції з банківською системою, сканування чеків та перегляд статистики.

Реалізація основного функціоналу є важливим етапом у створенні застосунку, по причині того, що він відповідає за розробку, як базових, так і розширених запитів користувача у системі. Разом із цим, впроваджується та підтримується база даних для зберігання та використання інформації.

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Окрему увагу слід приділити останньому етапу – комплексному тестуванню, що допомагає виявити похибки при проведенні перевірки на основі типових сценаріїв, як окремих частин додатку, так і платформи в цілому. Отримані результати використані для вдосконалення різних модулів застосунку, особливий акцент зроблений на інтерфейсі, безпеці та навантажуванні системи для оцінки продуктивності.

Після виконання всіх вищезазначених етапів формується програмний продукт для планування бюджету, що допоможе користувачу досягати своїх фінансових цілей та загалом покращити свою взаємодію з фінансами за рахунок перегляду статистики. Окрім цього, зроблений акцент на інноваційні технології, що взаємодіють зі штучним інтелектом, інтеграція з банківськими сервісами та клієнтоорієнтованість на закордонних користувачів. [7]

Основні модулі додатку:

- модуль користувача системи містить в собі різні методи автентифікації в систему, такі як реєстрація, вхід, обов'язкова наявність налаштувань безпеки, а також контролери, що надають змогу редагувати дані профілю та зберігання унікального ідентифікатора для майбутнього зв'язку;
- модуль транзакцій допомагає побудувати структуру для зберігання даних транзакцій певного клієнта, надає можливість на всю можливо взаємодію з ними для візуального відображення результатів, підтримує мультивалютність та конвертацію курсу;
- модуль обміну курсу валют надає можливість отримати актуальний курс валют, що використовуються у системі та відобразити в інтерфейсі;
- модуль категорій транзакцій зберігає, як основні, так і унікальні категорії створені конкретним користувачем для конкретизації платежів, які створюються вручну і підтримує переклад на мови, які є в додатку;
- модуль мультимовності дозволяє змінювати мову в програмному забезпеченні, що зберігається в окремому полі в базі даних та надає

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

можливість її автоматично підтягувати для майбутньої взаємодії з платформою;

- модуль матеріальних цілей допомагає користувачу створювати матеріальні цілі з дедлайном та визначеною сумою, що він планує досягнути, забезпечує перегляд процесу.

1.2. Аналіз аналогів програмного продукту

У сучасному світі значно зріс попит користувачів на мобільні додатки з фінансових послуг, що дає розуміння суттєвого підвищення конкурентності на ринку між даними застосунками [20]. Для досягнення цілі розробки конкурентоспроможного та високофункціонального програмного продукту, необхідно провести детальний аналіз існуючих аналогів на ринку. Ця стратегія допомагає не лише розкрити сильні або слабкі сторони цих додатків, а й зосередити увагу на аспектах, які можна вдосконалити на думку незадоволених користувачів та рецензентів з досвідом. Окрім цього, можна сфокусуватися на створенні оригінального функціоналу майбутнього продукту, що не фігурував ще на ринку еквівалентів.

Money Manager – є одним з даних представників, що містить в собі інтуїтивно зрозумілий інтерфейс, можливість планування фінансових цілей, перегляд особистих тенденцій витрат у різних виглядах і протягом різних часових відрізків. Проте, безкоштовний функціонал залишає бажати кращого, наприклад для більш детального аналізу вашого капіталу треба буде придбати платну версію або знайти ще один додаток для досягнення цієї мети [13].

Wallet by budgetbakers – широко використовуваний сервіс для автоматизації фінансів. Великий відсоток користувачів приваблює автоматична синхронізація з банківською системою, що значно полегшує взаємодію з платформою. Однак, багато користувачів скаржаться на обмежений функціонал безкоштовної версії та довготривалу відсутність оновлень протягом декількох років, що давно потребує додаток [14].

Monefy – відіграє роль особистого фінансового менеджера та простого інструменту для щоденного використання. У порівнянні з іншими

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

застосунками, він має обмежений спектр можливостей, хоча весь функціонал представлений у безкоштовній версії та присутня підтримка мультивалютності [15].

Saldo Finance – це єдиний український додаток для обліку фінансів на даний час, що надає можливість планування та корегування, як персонального, так і сімейного бюджету. Додаток підтримує різноманітний функціонал, що визнається необхідним серед користувачів [16].

Fast Budget – додаток, який допомагає контролювати особисті та сімейні фінанси, легкий у використанні, як зазначає розробник. Підтримує велику кількість валют та має в наявності функціонал синхронізації між пристроями. На жаль, присутні й негативні відгуки від користувачів на рахунок цього аспекту, тому що виникають проблеми з використанням [17].

Money Lover підтримує збереження даних на хмарі, що надає користувачам впевненості у відновленні їх у разі виникнення різних причин їх зникнення. Платформа має в собі більш широкий та детально продуманий функціонал у порівнянні з іншими представниками, але на жаль, досі наявна проблема щодо звітності, як і у інших представників [18].

Усі представлені аналоги пропонують користувачу облік грошових потоків, простий та зручний інтерфейс, різноманітний функціонал відповідно до обраного додатку, але не можуть одночасно повністю покривати всі вимоги вибагливих користувачів, аби задовільнити особисті потреби, клієнти змушені до паралельного використання декількох програм. Звертаючи уваги на вказанні чинники, зумовлюється необхідність у створенні більш вдосконаленого та унікального застосунку, який матиме можливість комбінувати найкращі аспекти та найновітніші технології, що забезпечить виконання запитів сучасних споживачів.

Користуючись випадком більш детального розгляду всіх вищеперелічених платформ, хочеться відмітити, що кожна з них має за мету покращити розуміння динаміки грошових потоків клієнта та підвищити ймовірність досягнення довгострокових фінансових цілей. Однак, деякі з

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

представників налаштовані на залучення клієнтів лише до преміум-версій, що в свою чергу потребує окрему плату. Для порівняння ключових функцій, які присутні у програмах зверніться до таблиці 1.1.

Таблиця 1.1

Аспекти	Money Manager	Wallet	Monefy	Saldo Finance	Fast Budget	Money Lover
Облік доходів і витрат	+	+	+	+	+	+
Планування витрат	+	+	-	+	+	+
Синхронізація з банківськими рахунками	-	+	-	-	+	+
Кастомізація категорій витрат	+	-	-	+	+	+
Сканування чеків	-	-	-	+	-	+
Синхронізація між пристроями	-	-	-	-	+	-
Перегляд статистики та її аналіз	+	+	+	+	+	+
Підтримка багатокористувацького режиму	-	-	-	+	-	-

Провівши аналіз різноманітних програмних забезпечень для керування бюджетом, можна прийти до висновку, що кожен із них має свої переваги та недоліки. Основні труднощі здебільшого стосуються наявності повнофункціональної безкоштовної версії, що є доволі важливим критерієм у виборі системи користувачем. Не менш суттєвим аспектом є питання синхронізації з банківськими системами, що може значно підвищити автоматизацію управління фінансами. Тим не менше, всі аналоги мають у наявності функціонал обліку доходів і витрат, а також перегляд статистики та її аналіз, який може бути продемонстрований у різних виглядах, починаючи від звіту, а закінчуючи різноманітними типами графіків [8].

Серед усіх представлених варіантів, найбільш конкурентоспроможними та багатфункціональними під час аналізу виявились Wallet, Fast Budget та Money Lover. В них зосереджена увага на автоматизації процесу створення транзакцій за допомогою використання банківських карток, інтеграції новітніх технологій для створення інтуїтивно зрозумілого інтерфейсу та покращення вже існуючого функціоналу.

Wallet – це застосунок для планування особистого бюджету, який допомагає його клієнтам слідкувати за доходами та витрати на щоденній основі. Як було зазначено раніше інтегрування з банківськими сервісами значно мінімізує необхідність ручного введення та робить його більш зручним у використанні. Простота дизайну дозволяє користувачу економити час для того, щоб розібратися з різноманітним функціоналом. На рис 1.1 можна переглянути інтерфейс додатку, який надає зручні інструменти для управління бюджетом.

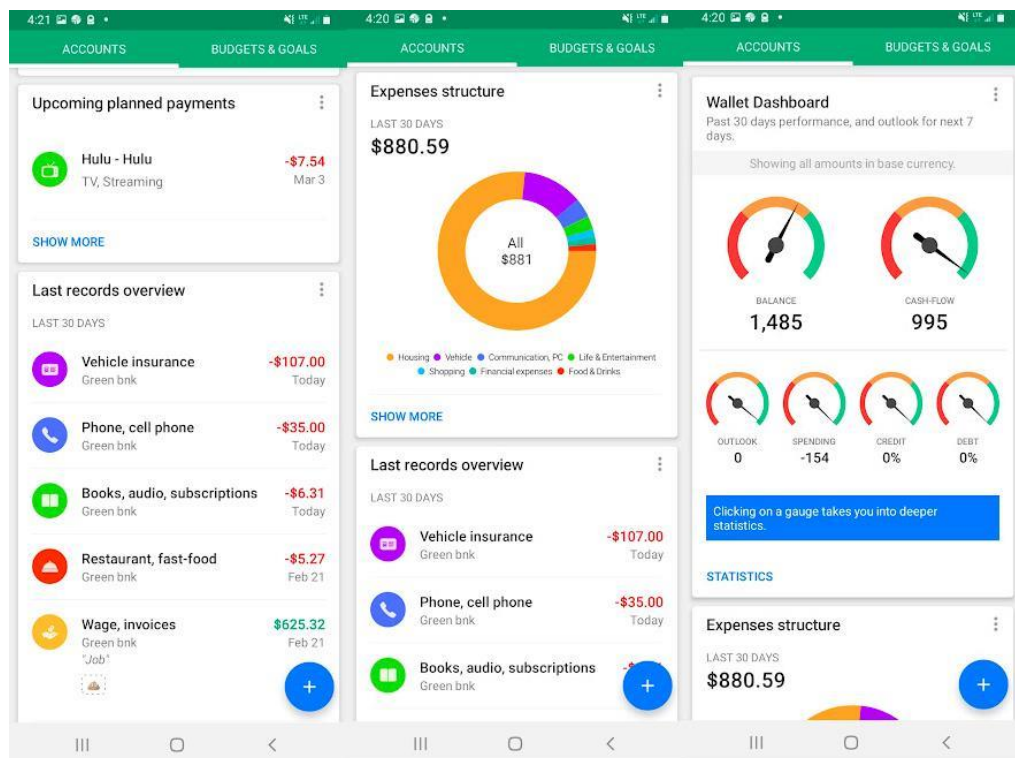


Рис 1.1 – Інтерфейс додатку Wallet

Fast Budget – це актуальний та зручний додаток, що надає можливість клієнтам візуалізувати їх фінансове становище за допомогою інтерфейсу, що орієнтований на легкість у сприйнятті користувачем. Платформа дозволяє

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

створювати необмежену кількість унікальних категорій транзакцій, що є доволі клієнтоорієнтованим кроком на ринку аналогічних програмних продуктів. У застосунку представлений широкий функціонал, зокрема, заплановані транзакції, встановлення фінансових цілей та поглиблений аналіз статистичних даних про прибутки та витрати у вигляді діаграм та графіків. Синхронізація між пристроями є ключовою характеристикою, що забезпечує доступ до фінансових даних з будь-якої точки світу та гарантує комфорт споживачів даним аспектом. На рис 1.2 представлені можливості, що надаються користувачу системи, але на жаль, не всі вони пропонуються у безкоштовній версії додатку, що може вплинути на рішення споживача щодо вибору цього аналогу поміж інших.

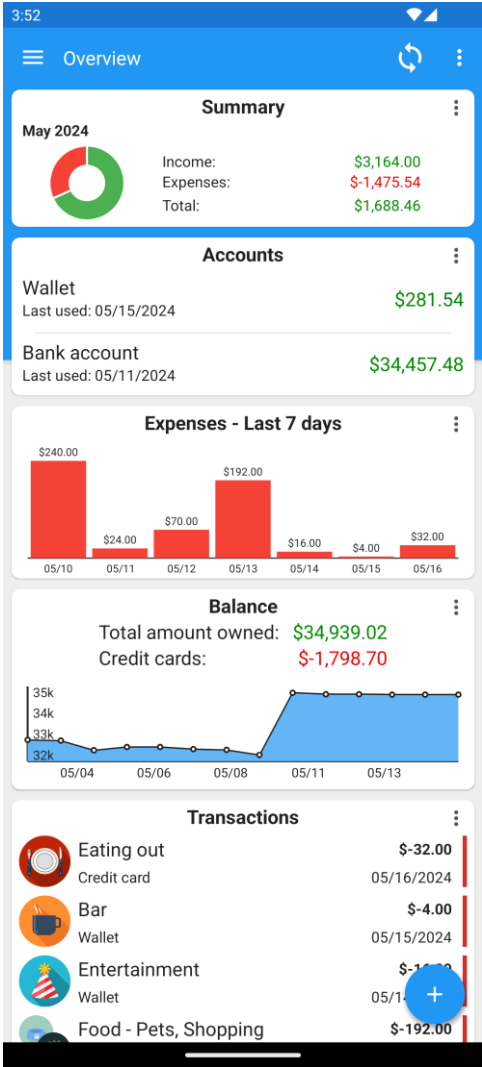


Рис 1.2 – Інтерфейс Fast Budget

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Money Lover – це програмне забезпечення для спостереження за грошовими надходженнями та витратами, що реалізується за допомогою застосування комплексних технологічних рішень. Платформа демонструє свою зацікавленість у користувачах з різних куточків світу – забезпеченням роботи з різними валютами, що спрощує його використання і для групи клієнтів, які працюють із різними джерелами доходу. На рис 1.3 можна розглянути достатньо простий зовнішній вигляд системи, що дозволяє не витратити зайвий час для того, щоб розібратися з функціоналом. Звітність з використанням наочних графіків дає розуміння звідки надходять ваші гроші і куди вони витрачаються.

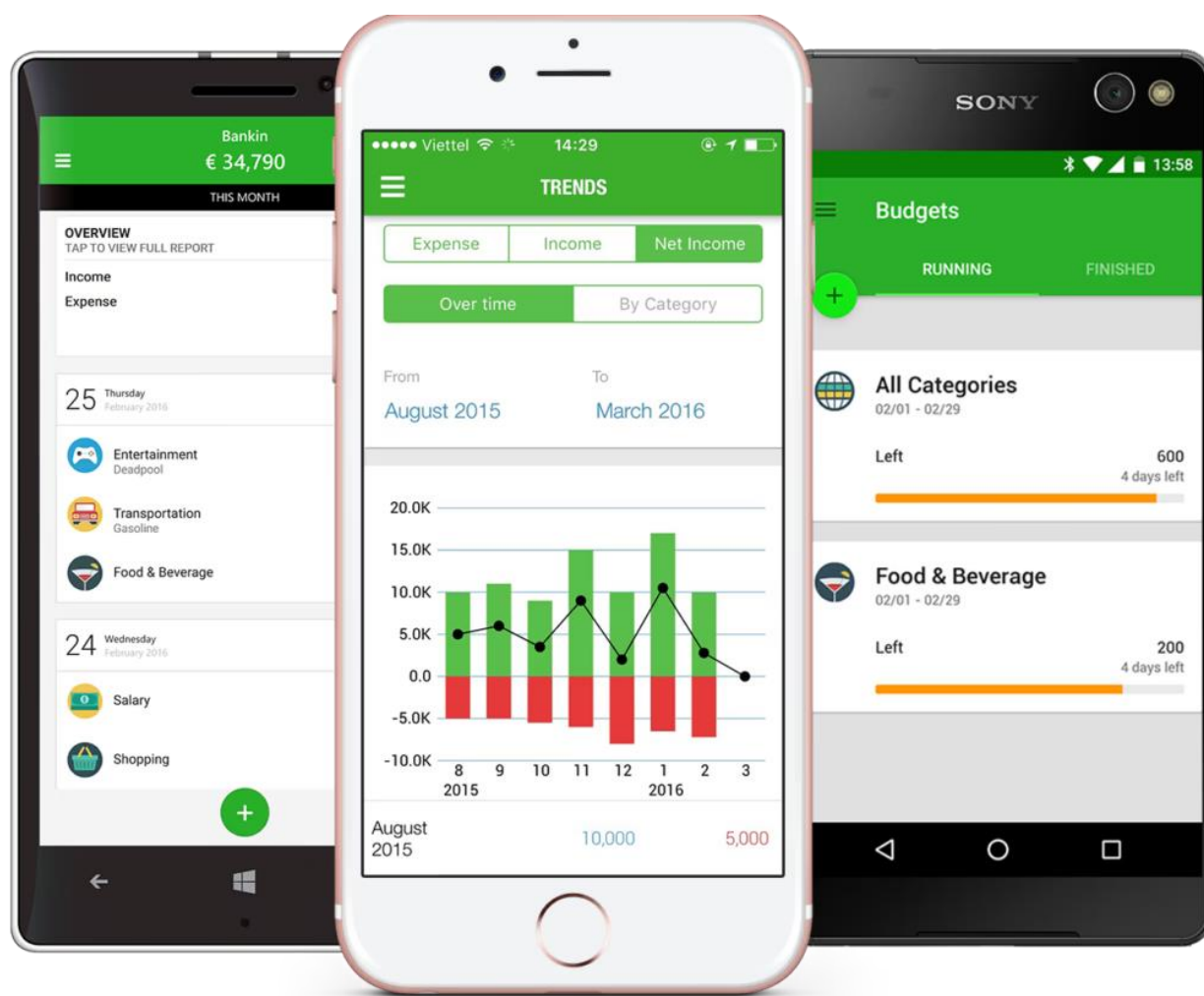


Рис 1.3 – Приклад інтерфейсу Money Lover

Враховуючи інформацію при детальному знайомстві з більш конкурентоспроможними спорідненими програмними продуктами для планування бюджету, можна дійти висновку, що зараз на ринку представлено

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

багато сильних претендентів. З метою досягнення унікального та новітнього продукту, необхідно покращити вже існуючий функціонал та додати оригінальний. Як перший пункт, було обрано інтеграцію з банківськими системами, що є надзвичайно важливим кроком у досягненні автоматизації процесу додання транзакцій та зменшене навантаження на клієнта. Наступним аспектом є кастомна категоризація доходів та витрат споживача, що відкриває перспективи для підкреслення унікальності своїм транзакціям та автоматичний переклад на інші мови, що зазначені у додатку [9].

Окрім цього, щоб забезпечити більший відсоток оригінальності програмному продукту, було прийнято рішення додати таку особливість, як сканування чеків з різних магазинів. Це також виступає ключовим фактором для автоматизації процесів додавання транзакцій, позбавляє необхідності активної участі користувача у внесенні даних, від нього лише вимагається перегляд зчитаної інформації та редагування у крайніх випадках. Не менш важливим впровадженням в систему є візуальна аналітика з використанням діаграм, графіків та просто звітів, що покращує розуміння динаміки фінансових потоків клієнта та підвищення рівня знань економічних процесів серед громадян країни в умовах активної експлуатації цифрових можливостей.

1.3. Вибір архітектури додатку

Процес розробки веб-додатку включає в себе декілька етапів, які відрізняються залежно від теми, обсягу, структури проекту, але всі вони мають мету створити функціональне, зручне для користувача та доступне програмне забезпечення, доступ до якого можна отримати з будь-якого куточку світу, маючи лише підключення до Інтернету.

Основною метою цієї роботи є проектування застосунку персонального планування бюджету, завдяки якому клієнт отримує особистого фінансового помічника, який реалізований за допомогою інтуїтивно-зрозумілого інтерфейсу для кожного покоління. Програмний додаток з відповідним напрямком, неможливо втілити без ключових аспектів: забезпечення постійної підтримки між всіма елементами системи, потреба у високому рівні заходів

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

безпеки, безперервна обробка запитів у режимі реального часу, динамічне розширення функціональних можливостей системи та інтерфейс, орієнтований на зручність користувача.

Для ідеї даного програмного забезпечення було обрано клієнт-серверну архітектуру. Вже з назви ми можемо зробити висновок, що у нас буде використовуватись модель, яка має дві основні частини – клієнт та сервер. У даного випадку, клієнт робить запити у формі JSON на сервер, а в той час сервер приймає HTTP-запити від клієнта, обробляє їх та відправляє відповідь, база даних PostgreSQL [21] в свою чергу інкапсулює доступ до даних, що дозволяє контролювати, як дані будуть використовуватись, змінюватись та забезпечує більший відсоток безпеки та гнучкості через обмеження прямого доступу до даних (рис 1.4).

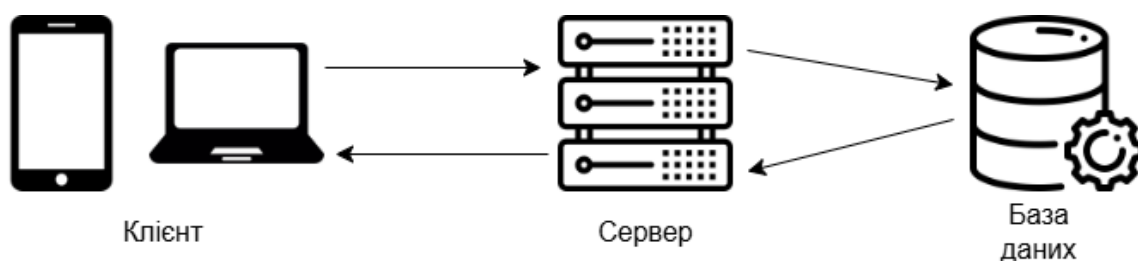


Рис 1.4 – Схема роботи клієнт-серверної архітектури

Із існуючих концепцій побудови модуля клієнт-сервер, було здійснено вибір на користь «слабкий клієнт – потужний сервер», що слугує розповсюдженим варіантом для створення веб-додатків, особливо тих, що працюють з великими обсягами даних і зумовлюють необхідність у сервері із високою обчислювальною потужністю [22].

З цього твердження, доцільно зробити висновок, що в нашому випадку, вся обробка бізнес-логіки, підключення до зовнішніх API та опрацювання даних, таких як: взаємодія з транзакціями, оновлення балансу, кастомізація категорій, зчитування чеків та інтеграція з банківськими системами відбувається на сервері.

Клієнтська частина навпаки відрізняється зниженою інтенсивністю навантаження, тобто ці компоненти відповідають лише за відображення візуальної складової для користувача аби мати змогу взаємодіяти з ними.

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

Складова може виконувати якісь мінімальні обчислення, які не справляють значного впливу на функціонал застосунку, обмежується виводом інформації, отриманої з серверу.

Таким чином, обрана нами архітектура забезпечує гнучкість, безпеку та ефективність роботи з даними, що є критично важливими факторами для програмного забезпечення, яке обробляє чутливу інформацію клієнтів. Завдяки чіткому розділенню функціонала між елементами системи, застосунок отримує високий рівень масштабованості та швидкого адаптування до зрізних змін у вимогах або структурі даних. Така стратегія допомагає легкому інтегруванню нового, більш сучасного функціоналу і забезпечує підтримку вже існуючого без загрози для цілісності системи.

Окрему увагу приділено аспектам безпеки: реалізовано багаторівневу автентифікацію, захист API та шифрування даних. Це мінімізує ризики витоку або несанкціонованого доступу до персональної інформації. Крім того, архітектура сприяє збереженню стабільної роботи системи навіть під навантаженням, що робить її надійною основою для реалізації функцій фінансового трекера або іншого сервісу, де ключову роль відіграє точність і безпека обробки даних.

1.4. Обґрунтування вибору інструментальних засобів

Початкова стадія процесу розробки передбачає визначення збалансованого варіанту технологічного стеку, який здатний підтримувати високу продуктивність, гарантує ефективну організацію роботи з даними та масштабованість системи. Підвищена вимога щодо конфіденційності та забезпечення високого рівня безпеки, є ключовим критерієм при взаємодії з чутливими даними користувачів у фінансових додатках. Разом із цим, була зосереджена увага на створенні інноваційного та інтуїтивно зрозумілого інтерфейсу, який зможе забезпечити належний рівень конкурентоспроможності з іншими середовищами.

У розробці фронтенд частини проекту, використовувались сучасні технології для надання їй покращеного результату поміж проаналізованих

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

аналогів. Використовувалась бібліотека React для створення зовнішнього вигляду системи, що пропонує широкий спектр інструментів та бібліотек для різних потреб та дозволяє повторне використання в різних частинах програми [6]. Експлуатація віртуально DOM та поєднання з компонентною структурою, дозволяє підвищити продуктивність та масштабованість веб-додатків. Окрім цього, потреба швидкого оновлення даних у реальному часі, обмежує необхідність повного повторного відображення сторінки, що стало ключовим фактором вибору саме цієї бібліотеки [5]. Імплементация Bootstrap фреймворку забезпечила швидку розробку платформи на базі готових компонентів та сприяла реалізації адаптивності інтерфейсу, у випадках, коли виникали труднощі або відсутність бажаного результату використовувалась додаткова стилізація CSS[23].

Серверна частина застосунку залучає Node.js у поєднанні з веб-фреймворком Express.js, що використовує простий спосіб для створення серверних додатків та API. [24-25] Сучасна платформа є однопоточною та швидко обробляє декілька запитів одночасно, що є ідеальним варіантом для динамічних середовищ, головною потребою яких є оновлення даних у реальному часі. Значущим аспектом у використанні Node.js, що істотно спрощує процес створення програмного забезпечення є можливість одночасного використання продукту, як фронтенд-розробки, так і бекенд-розробки. Окрім цього, важливою частиною програмної реалізації внутрішньої логіки системи приділено використанню Python та Tesseract [26]. Мова програмування використовувалась для підвищення потужності продукту та інтеграції з іншими системами, а програмне забезпечення для розпізнавання тексту з зображень, що у комбінації дозволило створити технологічно вдосконалений варіант сканування зображень та інтегрування їх в мою систему без необхідності залучення користувача для введення даних вручну.

В останній час виросла загроза несанкціонованого розголошення даних у зв'язку з підвищенням відсотка хакерських атак, що націлена на розкриття та розповсюдження особистих даних клієнтів. Для побудови достатньо

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

потужних захисних механізмів потрібно застосовувати збалансований підхід, який поєднує регулярне оновлення програмного забезпечення та вибору бази даних, що відзначається високим рівнем безпеки. Щоб отримати найкращий аналог серед усіх, треба провести порівняння існуючих та на базі цього зробити вибір, що буде відповідати усіх потребам застосунку. Проведено порівняння ключових характеристик, що допоможуть зробити правильний вибір (табл 1.2).

Таблиця 1.2

Критерії	MySQL	MongoDB	PostgreSQL	Cassandra	Oracle
Тип бази даних	Реляційна	Нереляційна	Реляційна	Нереляційна	Нереляційна
Підтримка транзакцій	ACID	Eventual Consistency	ACID	Eventual Consistency	ACID
Безпека та контроль доступу	Рольова система доступу, шифрування	Менша гнучкість у налаштуванні безпеки	Рольова система доступу, шифрування	Механізми безпеки, шифрування	Механізми безпеки, шифрування
Підтримка складних запитів	SQL, JOIN	Немає JOIN, обмежена агрегація	SQL, JOIN, агрегація	Обмежена підтримка запитів	SQL, JOIN, агрегації, підзапити
Масштабованість	Реплікація	Горизонтальне масштабування	Реплікація, шардінг	Горизонтальне масштабування	Реплікація, шардінг
Швидкість запису/читання	Висока швидкість для запитів	Швидке записування	Оптимізовано для складних запитів	Швидке записування для великих даних	Висока продуктивність

На підставі наведених аспектів порівняння можна зробити висновок, що PostgreSQL є найкращим вибором для створення фінансового трекера. Реляційна база даних у порівнянні з нереляційною, є найбільш раціональним

вибором для специфіки додатку. Наприклад, наявність підтримки транзакційної цілісності (ACID) [27] є критично важливою для подібних застосунків, де безпека даних, точність та консистентність необхідно для забезпечення їх подальшого функціонування. Конфіденційність даних також вирішується за рахунок вибору PostgreSQL для нашої розробки, у зв'язку з наявністю розширених можливостей безпеки, шифрування та контролю доступу.

Особлива увага приділяється питанням інформаційної безпеки. Для безпечної авторизації та автентифікації користувачів застосовується механізм JSON Web Token (JWT), що дозволяє уникнути зберігання сесій на сервері та підвищити захист від несанкціонованого доступу. З метою додаткового захисту персональних даних, зокрема паролів, впроваджено алгоритм хешування bcrypt. Уся передача даних між клієнтом і сервером відбувається з використанням захищеного протоколу HTTPS, що запобігає перехопленню конфіденційної інформації третіми особами. API-ключі та інші чутливі дані зберігаються у змінних середовища сервера, що знижує ризик їх витоку та підвищує рівень захисту інфраструктури.

Висновок до першого розділу

У першому розділі було виявлено низку проблем у додатках для планування бюджету на базі проведеного аналізу аналогів, що допомогли виявити, як сильні, так і слабкі сторони. Детальний огляд забезпечив формулювання основних потреб до програмного забезпечення, серед яких: унікальна категоризація витрат, сканування чеків та інтеграція з банківськими системами задля автоматизації процесу додання транзакцій.

Не залишено поза увагою поглиблений опис використаних технологій для застосунків контролю бюджету, як в бекенд-розробці, так і в фронтенд-розробці, аргументація вибору саме клієнт-серверної архітектури та гарантування високого рівня безпеки, за рахунок використання найкращої бази даних та різноманітних компонентів.

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

РОЗДІЛ 2. ПРОЕКТУВАННЯ ЗАСТОСУНКУ ДЛЯ ПЛАНУВАННЯ БЮДЖЕТОМ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури застосунку для планування бюджетом

Розроблення додатку, який орієнтований на щоденний моніторинг прибутку та витрат, що забезпечує додавання нових транзакцій, повна взаємодія з ними, наявність фінансового календаря та нагадування про планові оплати, а також здійснення аналізу на базі статистичних даних користувача. Основною метою є проектування застосунку, що націлений на довгострокову перспективу підвищення відсотку фінансової обізнаності серед громадян країни.

Основними ролями в системі виступають – звичайний користувач та адміністратор, кожен з них має чітко надані права і шляхи взаємодії з програмним застосунком, що відповідають їхнім повноваженням та цілям використання додатку.

Для наочного прикладу взаємодії учасників системи з елементами застосунку було створено діаграму варіантів використання (рис 2.1).

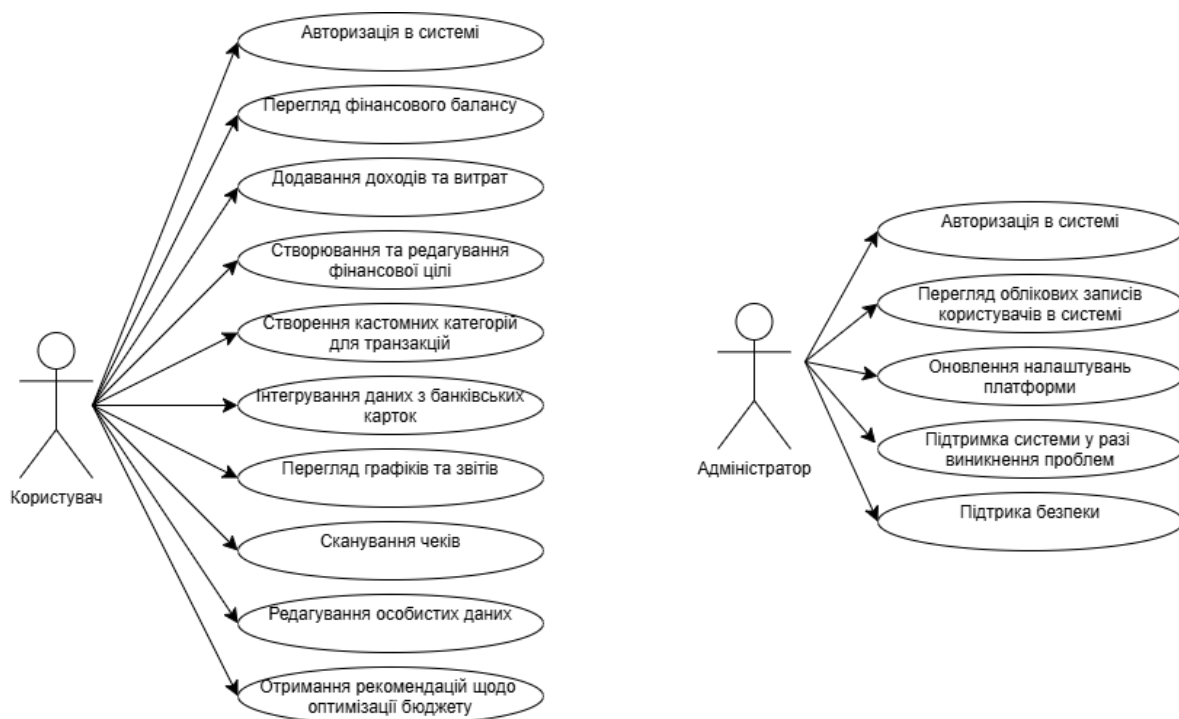


Рис 2.1 – Діаграма варіантів використання

Завдяки наявності цієї діаграми забезпечується можливість простеження реалізації бізнес-логіки зі сторони різних ролей в системі, проведення аналізу щодо майбутнього покращення продукту та визначення рівня складності окремих компонентів. Така стратегія, дозволяє сформулювати базу для подальшої структури об'єктно-орієнтованого програмування.

Опис діаграми варіантів використання наведено в таблицях 2.1 -2.6

Таблиця 2.1

Опис прецеденту «Авторизація в системі»

Назва варіанту використання	Авторизація в системі
Основні актори	Користувач
Короткий опис	Користувач має змогу створити новий обліковий запис або здійснити вхід у систему за допомогою спеціальних форм, ввівши дані в обов'язкові поля. Відповідно до валідаційної перевірки, системи надає доступ користувачу та або відхиляє його запит.

Таблиця 2.2

Опис прецеденту «Додавання доходів та витрат»

Назва варіанту використання	Додавання доходів та витрат
Основні актори	Користувач
Короткий опис	Користувач має можливість додати транзакцію, відповідно до її характеру, шляхом внесення даних до відповідних полів форми. Після додавання грошового переказу, який може відіграти роль, як прибутку, так і витрати, рахунок конкретного користувача оновиться, у відповідності до типу транзакції.

Таблиця 2.3

Опис прецеденту «Створення та редагування фінансових цілей»

Назва варіанту використання	Створення та редагування фінансових цілей
Основні актори	Користувач
Короткий опис	Користувач системи отримує можливість формування матеріальної цілі, яка потребує її назви, запланованої сума та приблизної дати її виконання, моніторинг прогресу мети. Можливо редагування у разі виникнення непередбачуваної фінансової витрати, що могла перенести дедлайни.

Таблиця 2.4

Опис прецеденту «Перегляд облікових записів користувачів в системі»

Назва варіанту використання	Перегляд облікових записів користувачів в системі
Основні актори	Адміністратор
Короткий опис	Адміністратор має можливість перегляду доданих користувачів в систему та відображення загальних даних, такі як ім'я та дата реєстрації.

Таблиця 2.5

Опис прецеденту «Перегляд графіків та звітів»

Назва варіанту використання	Перегляд графіків та звітів
Основні актори	Користувач
Короткий опис	Користувач з метою покращення розуміння динаміки особистих грошових потоків, користується можливістю перегляду проаналізованих статистичних даних у вигляді графіків та звітів за різні періоди часу.

Таблиця 2.6

Опис прецеденту «Створення кастомних категорій для транзакцій»

Назва варіанту використання	Створення кастомних категорій для транзакцій
Основні актори	Користувач
Короткий опис	Користувач може створювати приватні кастомні категорій для транзакцій, щоб створити більш конкретну або зрозумілішу назву для подальшого використання системою.

Обслуговування користувача зручною та ефективною системою для щоденного моніторингу особистих грошових потоків – головна здатність застосунку персонального фінансування. Додаток охоплює функціонал обліку доходів і витрат, який забезпечує можливість упорядковувати різноманітні фінансові дані, проводити моніторинг особистих коштів та отримати підвищення рівня знань у економічній галузі.

Варто відмітити, що використання програмного забезпечення має намір підвищити рівень фінансової обізнаності серед населення країни, тим самим

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

поліпшити якість економічної складової та привести державу до більшої конкурентоспроможності з розвиненими країнами. Це і є ключ до вирішення проблеми відсутності знань у сфері фінансів серед населення України.

Завдяки такому підходу створюються умови для більш усвідомленого управління особистими фінансами, підвищення фінансової стабільності та досягнення довгострокових цілей. Система також сприяє формуванню навичок планування, дисципліни у витратах та підвищенню впевненості користувача у власних фінансових рішеннях.

Функціональні вимоги

Застосунок повинен забезпечувати можливість реєстрації та авторизації для користувачів системи через email, також важливо відмітити, що в даному програмному забезпеченні присутнє розмежування на ролі: звичайний користувач системи та адміністратор додатку. Структурування за ролями передбачає різне візуальне відображення системи та надання доступу згідно з правами користувача.

Наявність парсингу офіційної сторінки Національного банку України надає можливість перегляду курсу валют та використання їх у додаванні транзакцій, що може значно полегшити взаємодію користувачів, які зараз знаходяться за кордоном або мають сторонні джерела доходу та витрат.

Користувач має змогу повної взаємодії з транзакціями, від їх додавання, редагування у разі наявності помилок та видалення, також у цій операції клієнт спроможний надавати більш конкретну інформацію про грошовий переказ, таку як: особиста кастомізація категорії, додавання нотаток та наявність валюти в якій відбувається платіж. Для більшої автоматизації системи було впроваджено розробку двох компонентів, що мінімізують залучення ручної роботи користувача, а саме сканування чеків та синхронізація з банківськими системами для комфортного імпорту розрахункової операції.

Важливою складовою є відображення проаналізованих даних на базі проведених грошових операцій, які можна буде переглянути у вигляді

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

графіків, звіту та дашбордів з можливістю фільтрації за різними даними, такими як: категорія, період часу та типом транзакції.

Також додаток має клієнтоорієнтованість на залучення користувачів з інших країн за рахунок наявності англійської та німецької мови, що в подальшому може розширюватись у разі наявності потреб інших мов.

Крім того, система повинна підтримувати інтеграції з зовнішніми сервісами через REST API, забезпечуючи автоматичний обмін даними та розширення функціональності платформи.

Нефункціональні вимоги

Програмний додаток має за ціль розробити інтуїтивно зрозумілий інтерфейс, щоб користувачі не відчували незручності у взаємодії з системою та мали можливість користуватись застосунком без спеціальної технічної підготовки. Час необхідний для освоєння інтерфейсу вебсайту – 3-5 секунд для нових користувачів та не перебільшувати 3 секунди для зареєстрованих.

Платформа повинна забезпечувати критично важливі характеристики, які спеціалізуються на подібних системах, такі як надійність і безпека, оскільки здійснюється взаємодія з чутливими даними користувачів програмного забезпечення. Не меш важливим фактором є доступність додатку протягом усієї доби у сучасному світі, де багато процесів відбувається в режимі реального часу та суттєво підвищують рівень зручності для користувачів.

Особливо важливим аспектом є продуктивність системи у додатках для планування бюджету, враховуючи те, що одночасно повинно проводитись обслуговування великої кількості споживачів без зниження швидкодії. Забезпечування швидкої обробки запитів сторінки та масового імпорту банківських сервісів є також критично важливим у аналогічних веб-застосунках.

Для підвищення майбутньої експлуатації та масштабування, програмне забезпечення повинно бути підготовленим до використання більш інноваційних технологій та розширення вже існуючого функціоналу

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

відповідно до зростання запитів споживачів системи. Оновлення платформи має відбуватись у фоновому або автоматичному режимі без переривання роботи користувачів.

Системні вимоги

Застосунки зі схожим функціонал повинні використовувати кроссбраузерне або кросплатформне рішення, яке націлене на оптимізацію роботи у різних браузерах (Google Chrome, Mozilla Firefox, Safari, Edge версії 90 або вище) або на різних платформах (операційних системах, пристроях) з єдиною кодовою базою, що збільшує потенційну можливість популяризацію додатку серед населення.

Для комфортної роботи з додатком, знадобиться пристрій, що буде мати мінімально оперативну пам'яті – 4 GB, але рекомендується наявність 8 GB, процесор повинен бути двоядерним із тактовою частотою 1.8 GHz або краще. Якщо говорити про операційну систему, то бажано мати Windows 10/11 чи macOS 10.15 або новішу версію. Також обов'язково з'єднання з інтернетом, що дозволить користуватись додатком.

Серверна ж частина передбачає роботу з використанням Node.js (версія 18 або новіша), PostgreSQL у ролі бази даних для збереження конфіденційної інформації клієнтів та JWT-авторизація для забезпечення безпеки.

2.2. Розробка бази даних системи

Гарантія безпечної обробки даних та їх збереження характеризується одним із самих важливих аспектів у розробці застосунків, що взаємодіють та використовують чутливі дані у реалізації програмного забезпечення. Виходячи з цих потреб, було прийнято рішення використовувати реляційну базу даних – PostgreSQL, що відрізняє її поміж інших аналогів наявністю підтримки транзакційної цілісності (ACID), властивості якої унеможливлюють порушення структури платежів та сприяють їх збереженню, навіть у разі виникнення проблем або збою системи.

Потенційна популяризація програмного забезпечення фінансового трекеру серед населення країни, потребує значного масштабування серверу

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

внаслідок залучення великої кількості користувачів, що одночасно будуть створювати транзакції, генерувати статистичні дані з використанням графіків чи сканувати чеки. Забезпечення потреби клієнта застосунку в швидкості та комфорті здатна виконати обрана нами база даних, яка одночасно підтримує вертикальне та горизонтальне масштабування. PostgreSQL може сприяти покращенню роботи, як одного серверу за рахунок додавання оперативної пам'яті або встановлення потужного процесору, так і додавання кількості серверів, що надають можливість розподіляти запити між собою при роботі з великими обсягами різного типу даних та створення кластеру бази даних за рахунок використання розширення Citus [29].

Вагома кількість високонавантажених компаній у сферах фінансового аналізу чи обробки чутливих даних користувачів для банківських або державних послуг, що претендують на світовий рівень успішно використовують PostgreSQL. Наприклад, можна розглянути відоме фінансово-технологічне підприємство Revolut [28], яке надає можливість клієнтам створювати мультивалютну дебетову карту, котра кодується практично у всьому світі, а всі платіжні операції можна проводити в одній валюті. Фірма використовує представлену СУБД для обробки та зберігання конфіденційних даних та транзакцій споживачів. З окладу на викладене, можна зробити висновок, що обраний приклад є достатньо показовим та відповідає запиту збереження чутливої інформації, яка належить клієнтам.

На рисунку 2.2 представлена схема бази даних для програмного забезпечення планування бюджету, яка представляє собою 6 основних таблиць кожна з яких зберігає інформацію відповідно до її назви.

Для реалізації структурованого зберігання різних даних споживачів та їх обробку в додатку фінансування, використовувались ключові об'єкти системи, які логічно пов'язані між собою зовнішніми ключами, завдяки яким гарантується гнучке та зручне використання системи у галузі статистичних даних та проведенні їх аналізу[30].

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

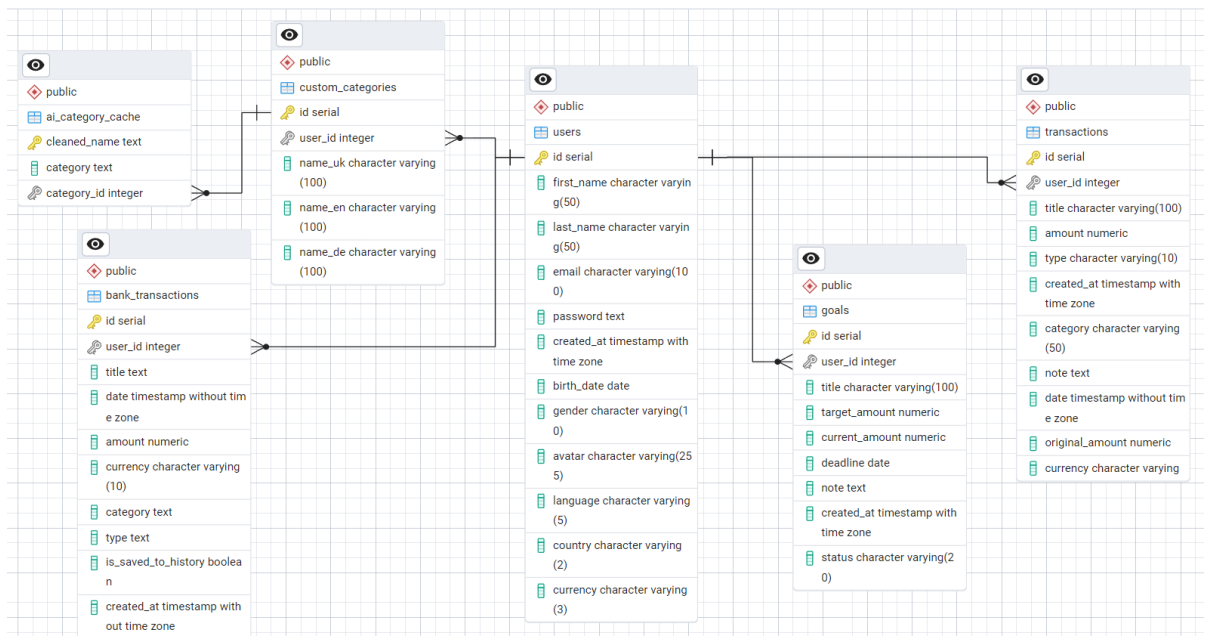


Рис 2.2 – Діаграма бази даних

База даних платформи складається з таблиць:

1. Users – зберігає всю інформацію про зареєстрованих користувачів, відіграє роль центральної таблиці в нашій схемі;
2. Transactions – містить всі фінансові операції користувачів, що створені власноруч та імпортовані з інших автоматизованих процесів незалежно від їх типу, дозволяючи переглядати поточний баланс та аналізувати витрати;
3. Goals – підтримує зберігання даних про матеріальні цілі, які ставлять користувачі;
4. Custom_categories – таблиця, яка архівує особисті категорії, які створені користувачем та підтримується мультимовністю;
5. Bank_transactions – складається лише з імпортованих транзакцій з банківського API;
6. Ai_category_cache – таблиця, що призначення для зберігання кешованих результатів проведення категоризації, здійснених за рахунок штучного інтелекту.

Опис кожної з таблиць бази даних наведений у таблицях 2.7 – 2.10:

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.7

Опис таблиці Users

Поле	Тип	Призначення
id	serial	Унікальний ідентифікатор користувача
first_name	varchar(50)	Ім'я користувача
last_name	varchar(50)	Прізвище користувача
email	varchar(100)	Пошта користувача
password	text	Хеш пароля користувача
created_at	Timestamp with time zone	Час реєстрації
birth_date	date	Дата народження користувача
gender	varchar(100)	Стать користувача
avatar	varchar(255)	Іконка користувача
language	varchar(5)	Мова інтерфейсу
country	varchar(2)	Країна проживання/ народження користувача
currency	varchar(3)	Валюта країни, що автоматично підв'язується після вибору

Таблиця 2.8

Опис таблиці Transaction

Поле	Тип	Призначення
id	serial	Унікальний ідентифікатор користувача
user_id	integer	Зовнішній ключ, що посилається на users(id)
title	varchar(100)	Назва транзакції
amount	numeric	Сума транзакції

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Проводження таблиці 2.8

type	varchar(10)	Тип транзакції: "income" або "expense"
category	varchar(50)	Категорія платежу
note	text	Додаткові нотатки до транзакції
date	timestamp with time zone	Дата транзакції (може задаватися вручну)
created_at	timestamp with time zone	Дата створення транзакції
original_amount	numeric	Першопочаткова сума, якщо вона вказана не в основній валюті системи
currency	varchar	Валюта в якій відбувається платіж

Таблиця 2.9

Опис таблиці Goals

Поле	Тип	Призначення
id	serial	Унікальний ідентифікатор користувача
user_id	integer	Зовнішній ключ, що посилається на users(id)
title	varchar(100)	Назва цілі
target_amount	numeric	Матеріальні ціль, яку треба буде досягти
current_amount	numeric	Поточний грошовий баланс
dedline	varchar(50)	Дата до якої плановано досягнути ціль
note	text	Додаткові нотатки до цілі
created_at	timestamp with time zone	Дата створення цілі

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Продовження таблиці 2.9

status	varchar(20)	Статус відповідно виконання цілі, на початку завжди задається, як «активний»
--------	-------------	--

Таблиця 2.10

Опис таблиці Custom_categories

Поле	Тип	Призначення
id	serial	Унікальний ідентифікатор користувача
user_id	integer	Зовнішній ключ, що посилається на users(id)
name_uk	varchar(100)	Назва кастомної категорії користувача на українській мові
name_en	varchar(100)	Переклад категорії на англійську мову
name_de	varchar(100)	Переклад категорії на німецьку мову

Таблиця 2.11

Опис таблиці bank_transactions

Поле	Тип	Призначення
id	serial	Унікальний ідентифікатор користувача
user_id	integer	Зовнішній ключ, що посилається на users(id)
title	varchar(100)	Назва транзакції
date	varchar(100)	Дата проведення транзакції
amount	numeric	Сума на яку проведена транзакція

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

Продовження таблиці 2.11

currency	varchar(10)	Валюта транзакції
category	text	Категорія транзакції
type	text	Тип транзакції
is_saved_to_history	boolean	Перевірка чи було додано транзакції в таблиця «transactions»

Таблиця 2.12

Опис таблиці ai_category_cache

Поле	Тип	Призначення
cleaned_name	text	Очищена назва товару, ключ для кешу – відбувається перегляд існування категорії чи ні.
category	text	Текстова категорія, що з'являється після взаємодії зі штучним інтелектом
category_id	integer	Зовнішній ключ, що вказує на таблицю кастомних категорій, що відповідає цьому товару.

2.3 Проектування та реалізація алгоритмів роботи системи

Веб-додаток реалізований у комбінації використання інноваційних технологій, що допомагають забезпечити повноцінне функціонування, як практичних вимог вибагливих користувачів системи та рецензентів, так і зручності та зрозумілості клієнту.

Програмне забезпечення реалізовано з розрахунком на доступність та простоту у використанні незалежно від віку користувача та досвіду роботи з суміжними програмними системами, що допомагає клієнтам без навантажень та посиленої концентрації зусиль користуватись усіма функціями платформи.

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

Інтерфейс цього веб-застосунку потребує бути інтуїтивно-зрозумілим, що зможе допомогти його популяризації серед потенційних клієнтів серед громадян країни.

На рис 2.3 представлено візуальне відображення потенційної схеми взаємодії користувача з системою, але у скороченій версії, залишаючи лише головні пункти у роботі з платформою, які передбачаються у функціоналі.

У Додатку А представлена розширена схема, яка детально демонструє весь ймовірний шлях клієнта, який націлений на використання програмного забезпечення на регулярній основі. Перш за все, користувач, який може стати потенційним споживачем продукту отримує перед очима сторінку входу, яка представляє собою форму: у разі існування користувача в системі, він вводить лише пароль та логін і переходить до наступних кроків у ланцюгу діаграми, якщо ж користувач відсутній в базі даних, то він зобов'язаний ввести потрібні дані для можливості зареєструватись.

Далі схема надає змогу обрати ключові кроки у взаємодії з платформою, візуально можна побачити, що вибірка стоїть між автоматизованими функціями додавання транзакцій в систему та ручними, якщо оптимізація процесів неможлива. Для підвищеного комфорту користувача у веб-застосунку було представлено дві основні інноваційні функції аби мінімізувати його активну унесенні даних: інтеграція з банківською системою та сканування чеків. У результаті експлуатації даних технологічних рішень від нього потрібно буде лише перегляд зчитаної інформації та редагування даних у крайніх випадках, що становиться незначний відсоток за статистикою.

Звичайно є і свої переваги в інтеракції з додаванням платежів вручну, до них можна віднести: чіткий контроль та точність введених даних, можливість кастомізації особистих категорій грошових операцій. Таку активність можна віднести до інтерактивної діяльності споживача з платформою, яка сприяє його інтенсивному залученню в роботу із додатком. Варто зазначити, що такий функціонал дозволяє потенційно довгостроковому клієнту відчувати себе

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

унікальним та підвищувати відсоток довіри до застосунку. Програмне забезпечення цим кроком передбачає підвищення рівня відповідальності людини за свої кошти, прийняття обґрунтованих рішень та мінімізувати ймовірність виникнення ризиків імпульсивних витрат, що створюють передумови для забезпечення довгострокової фінансової стабільності

Наступна функціональність дозволяє користувачу взаємодіяти з елементом логічної постановки матеріальної цілі у довготривалій чи короткотривалій перспективі. Учасник системи наділений здатністю додавання грошової цілі, яка перед офіційним занесенням в базу даних повинна пройти валідацію, щодо реалістичності її досягнення за зазначеним користувачем періодом. Логіка цієї перевірки базується на математичному та статистичному обчислюванні відсотку можливих відкладених грошей з місячного бюджету робітника країни. У випадку вибору дуже амбіційних цілей у питанні термінів виконання, які будуть перекривати базову вираховану суму відсотку доходу за місяць, яку користувач може заощаджувати на свою ціль, передбачено надсилання сповіщення із системи, щодо збільшення дедлайну виконання мети і до моменту поки споживач сам не вибере цю дату, операція не збережеться в систему.

Не менш цікавий є й наступний крок шляху, який демонструє доволі інноваційні аспекти, які заслуговують уваги, не тільки від користувачів застосунку, а й простих громадян країни, які хочуть підвищити свій рівень фінансової обізнаності, використовуючи прості функціональні компоненти, реалізовані із застосуванням лаконічної візуалізації. Перегляд статистики є інтерактивним кроком у роботі споживача з системою, представлені різні варіації графіків, такі як: Dashboard, Pie chart та Line graphs. Фільтрацію перелічених частин можна провести за категоріями та періодами часу для візуального представлення, а також завантажити звіт з вказаних складових.

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

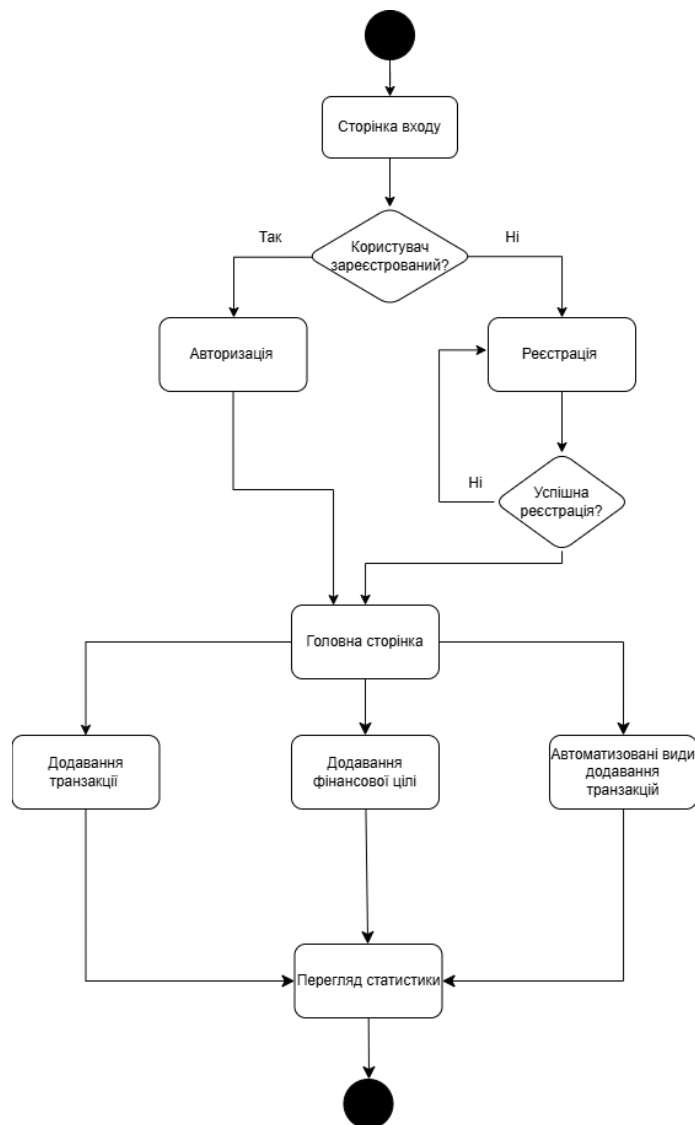


Рис 2.3 – Скорочений варіант діаграми активності

2.4 Реалізація функціоналу веб-додатку для планування бюджету

Реалізована платформа для моніторингу фінансів, відповідно до принципів запитів користувачів з використанням інноваційних технологій та привабливого зовнішнього вигляду, що відповідає потребам конкурентоспроможності вже існуючих розроблених застосунків на ринку. Використовуючи програму, споживач здатен здійснювати компетентне управління своїм капіталом, користуватись різними видами додавання транзакцій, починаючи від ручного вводу даних, закінчуючи залучанням сучасними технологічними інструментів у сьогоденне життя.

У застосунку реалізована взаємодія із реляційною базою даних PostgreSQL через модуль pg у Node.js, що дозволяє залишати чистий шлях

SQL-запитів та забезпечує зосередженість уваги на максимальному рівні контролю над запитами та оптимізації продуктивності. Здійснено відмову від використання ORM-інструментів для Node.js, таких як: Sequelize, TypeORM, Prisma та Bookshelf.js, що забезпечують абстрактну роботу над базами даних. Суттєво важливим у контексті роботи з чутливими фінансовими показниками є підтримка контролю над запитами та коефіцієнтом працездатності, що не належить до функціоналу цих ORM. До прикладу, системи фінансового управління в таких операціях, як підрахунки, фільтрація та групування, вимагають забезпечення максимально детальних і передбачуваних запитів, що у свою чергу не надає Sequelize, особливо при використанні складних запитів, що не оптимізуються для великої кількості запитів на оброблення.

Як було підкреслено раніше, увага розроблення програмної системи трекеру бюджету зосереджена на забезпеченні відповідного рівня безпеки інформації з обмеженим доступом, що фігурує у взаємодії з фінансовими даними. Задля того, уникнути несанкціонованого витоку даних та гарантувати захист конфіденційної інформації користувачів, використовується комбінування різних прогресивних інструментів.

В останній час стало неймовірно популярним використання JWT-токену при авторизації користувача, що не залишилось поза увагою нашого застосунку. Усі запити в програмній системі передаються через API, які в свою чергу захищені JWT-токеном, по причині того, що він одразу прив'язується до кожного запиту. Унаслідок цього, забезпечується доступ лише до персональних даних користувача та створюється обмежений шлях фільтрації за унікальним ідентифікатором користувача (`user_id`), що у свою чергу ізолює досягнення цієї інформації іншими шляхами та обумовлює її надання лише конкретним клієнтам. (додати додатки з jwt)

Окрім цього, було використано ще низку технологій, які потенційно збільшуються відсоток уникнення неконтрольованого поширення інформації з системи: хешування паролів за допомогою bcrypt, валідація введених даних

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

у формах, що проводяться за допомогою кастомних функцій та сторонніх модулів, а також використання захищеного з'єднання через HTTP-запити.

Вищезазначені ключові аспекти, можна детальніше розглянути в реалізованих фрагментах коду (див. Додаток Б). Файл Auth.js, в наявності якого, є модулі: створення JWT-токену, захищеність маршрутів за допомогою ідентифікатору конкретного користувача, SQL-запити через використання pool.query, хешування паролів та використання прив'язаного токену до інших функціональних можливостей додатку.

Після успішного входу в систему, користувачу надається можливість взаємодії з власним рахунком за допомогою додавання грошових операцій із застосуванням як ручних, так і автоматизованих методів. Представлена реалізація запиту від користувачів у вигляді зменшення їх активного втручання в систему, була виконана за допомогою інтегрування з банківськими сервісами, більш докладніше їх можна переглянути у наданих фрагментах коду. Наприклад, використання бібліотеки axios у файлі monobank.js надає зручний інтерфейс для надсилання HTTP-запит за допомогою посилання на секретний токен клієнта, а для його зберігання у свою чергу використовується dotenv. Наступний крок, у розробці цього інноваційного функціоналу представлений в файлі bank.js, що отримує виписку банківських операції за обраний користувачем період, переміщує їх в окрему таблицю в БД, а після проходження перевірки додає нові платіжні операції в історію спільних фінансових транзакцій для їх майбутнього інтегрування в проведення статистики на зібраних даних.

Не менш важливу роль у сприянні зростанню комфортних умов клієнта є сканування чеків, що допомагають мінімізувати його роль у введенні даних та забезпечує можливість економії його дорогоцінного часу. Функціонал розпізнання тексту з картинок/чеків, виконано у поєднанні Tesseract OCR та Python, а бібліотека pytesseract є обгорткою для цих компонентів. У файлі app.py (див. Додаток В) представлено зчитування товарів, фільтрація сміття з відсканованих рядків чеку, додавання евристики для врахування їх

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

особливостей та повернення результатів у форматі JSON та переходу на наступний крок автоматичної категоризації продуктів. Файли `categories.py` та `categorize_smart_router.py` реалізують ідею надавання категорій розпізнаних товарів з чеку за допомогою інтегрування функціоналу штучного інтелекту (OpenAI GPT), що значно пришвидшує роботу системи та оптимізує запити. В одному файлі представлено використання базової категоризації, а в іншому розширена з залученням унікальних категорій певного клієнта, що зберігається у кеші для уникнення повторних запитів.

Також, приділена увага й ручному додаванню грошових операцій, що націлені на опрацювання залишкових транзакцій у повсякденному житті користувача, після обробки з використанням автоматизованого функціоналу. Всі можливі взаємодії споживача з фінансовими операціями представлено у модулі `transactions.js` (див. Додаток Б), що реалізовує CRUD API для транзакцій у системі для планування бюджету. Імплементовано також додавання транзакцій у різних валютах, такий як USD, EUR та GBP, їх курс автоматично парситься з офіційної сторінки національного банку України з метою забезпечення мультивалютності середовища, функціонал представлений у `fetchExchangeRates.js` та `exchange.js`. Аби уникнути непорозумінь, система автоматично обраховує суму платежу у гривні, що дозволяє забезпечити узгодження фінансового балансу.

Останній функціонал, що може ініціювати користувач для створення статистичних даних – це управління матеріальними цілями, що реалізують CRUD-операції після проходження захищених маршрутів з влідацією даних на рівні сервера. У файлі `goals.js` реалізована підтримка встановленого дедлайну на виконання цієї цілі, передбачено поле накопичення для фіксації процесу та в подальшому використання цих даних для графічного відображення інформації по завершенню аналітичного процесу.

Після внесення записів про грошові операції в спільну історію транзакцій при різних умовах їх створення, вони починають відігравати роль статистичних даних для формування різних типів звіту. Аналітична довідка

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

може містити генерацію з використанням таких елементів, як графіки та повноцінний статистичний звіт, який надає можливість громадянину країни підвищувати свої знання у галузі фінансової обізнаності на прикладі своїх доходів і витрат, що можуть фільтруватись та одночасно з цим відображатись на екрані за категоріями та різними періодами часу, що значно полегшує сприйняття споживача даної інформації, на прикладі дашбордів, графіків та кругових діаграм. Весь цей процес представлений у таких файлах, кожна назва якого забезпечує файлу відповідний функціонал: CategoryPieChart.jsx, AnalyticsCharts.jsx, OverviewDashboard.jsx та ReportGenerator.jsx (див. Додаток Б).

Клієнтська частина веб-додатку була реалізована за допомогою сучасних інструментів, а саме поєднання React, Bootstrap та у крайньому випадку застосування CSS, якщо усунення визначеної проблеми було унеможливлене без його включення, Така комбінація репрезентує собою створення візуального відображення односторінкового застосунку (SPA), що формується локально та представляє собою повністю інтуїтивно зрозумілий й інтерактивний фронтенд, що проводить взаємодію з сервером за рахунок REST API.

Висновки до другого розділу

У другому розділі дипломної роботи був здійснений детальний аналіз, що мав на меті проведення всіх можливих сценаріїв взаємодії клієнта з програмним забезпеченням для планування бюджету. На основі отриманих даних цього дослідження була розроблена структура веб-застосунку, що включала в себе моделювання структури проекту та взаємодії між структурними елементами.

Також було розроблену базу даних для зберігання чутливої інформації, такої як особисті клієнтські дані, їх взаємодія з фінансами та статистичні дані, які знаходяться на серверній частині додатку.

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

Проведена детальна робота щодо опису послідовно-потенційних пунктів взаємодії користувача з програмних забезпеченням та в цілому описання алгоритму роботи веб-застосунку.

Окрім цього, була акцентована увага на поглибленому описі основного функціоналу програмної системи – моніторинг бюджету, ключові моменти взаємодії споживача з системою.

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ

3.1 Структура інтерфейсу веб-додатку для планування бюджету

Щоб клієнт зміг розпочати роботу з додатком, йому необхідно буде мати стабільне інтернет з'єднання на пристрої, що є ключовою потребою на даному етапі взаємодії. Розмір екрану девайсу або комп'ютера не має значення, тому що весь інтерфейс адаптований під будь-яку розмірність монітору аби забезпечити комфорт у використанні.

Перед тим, як клієнт отримає доступ до функціоналу програмного забезпечення, йому необхідно ввести дані, які очікує форма аби зареєструвати його у додатку. На рис 3.1 представлений вигляд форми входу у систему, якщо ж у клієнта ще не має акаунт, то він переходить на сторінку з формою реєстрації нового споживача (рис 3.2). Також на кожній сторінці представлена можливість перекладу сторінки, якщо користувач бажає продовжити взаємодію іншою мовою, то вона буде прив'язуватися до нього.

Рис 3.1 – Форма входу в систему

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

Реєстрація

UA

Ім'я

Прізвище

Ел. пошта

Пароль

Підтвердження паролю

Зареєструватися

Рис 3.2 – Форма реєстрації нового користувача

Після успішної автентифікації учасника системи, він буде бачити головну сторінку додатку, де представлений основний функціонал додатку (рис 3.3)

Finance Tracker

Мій цілі

UA

Мій акаунт

Вийти

Привіт, Оксана!

Поточний баланс

28264.43 ₴

Офіційний обмінний курс

USD: 41.488 ₴

EUR: 48.0825 ₴

GBP: 56.3822 ₴

Додати транзакцію

Отримати транзакції з банку

Розпізнавання чеків

Останні транзакції

Тип:

Категорія:

Усі

Усі

Переглянути статистику

Крем

13.06.2025

-700.00 ₴

воляга

12.06.2025

-14.70 ₴

молоко, вафлі, батончик, батончик

12.06.2025

-103.60 ₴

молоко, вафлі, батончик, батончик, пріжиджі

12.06.2025

-118.30 ₴

чіпси, куряче, картопля

12.06.2025

-182.35 ₴

чіпси, напій, куряче, картопля, філе

12.06.2025

-304.35 ₴

1 / 4

Рис 3.3 – Головна сторінка застосунку

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

На даний момент реалізовано лише один можливий поточний рахунок при використанні одним користувачем. Щоб оновити баланс, представлена можливість додавати транзакції, як вручну, так і більш автоматизованими шляхами. Спершу розглядає варіант додавання транзакцій власноруч, якщо у цьому є потреба. Поля у формі є обов'язковими, також вони проходять валідацію перед додаванням у базу даних, у цьому прикладі створення транзакції, я продемонструвала весь функціонал, що був заявлений у виконанні: надання назви, типізація платежі, додання нотаток у разі необхідності, використання іншої валюти та внесення кастомної категорії прив'язаної до певного користувача (рис 3.4).

The screenshot shows a web interface for a 'Finance Tracker'. At the top right, there are links for 'Моя ціна', 'UA', 'Мій акаунт', and 'Вийти'. The main heading is 'Додати транзакцію'. Below it, there are several input fields: 'Назва' (Name) with the value 'Подарунок подрузі', 'Сума' (Amount) with the value '55', 'Оберіть валюту' (Select currency) with a dropdown showing 'USD', 'Тип' (Type) with a dropdown showing 'Витрата', and 'Категорія' (Category) with a dropdown showing 'Подарунок'. Below the category dropdown, there is a text input field with the placeholder 'Create "Подарунок"' and a small note 'додатковий опис (необов'язково)'. At the bottom of the form is a green button labeled 'Зберегти'.

Рис 3.4 – Форма додання транзакції вручну

Після натискання на кнопку, грошова операція додається у спільну історію всіх платежів на акаунті користувача. У разі виникнення друкарської помилки чи похибки у введенні інших даних, система надає можливість редагування полів та їх автоматичного оновлення. Також, у разі зміни валюти, буде відбуватися миттєвий парсинг курсу вже правильно обраної валюти та сума фінансової операції оновлюється динамічно, без участі користувача у цьому процесі (рис 3.5).

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

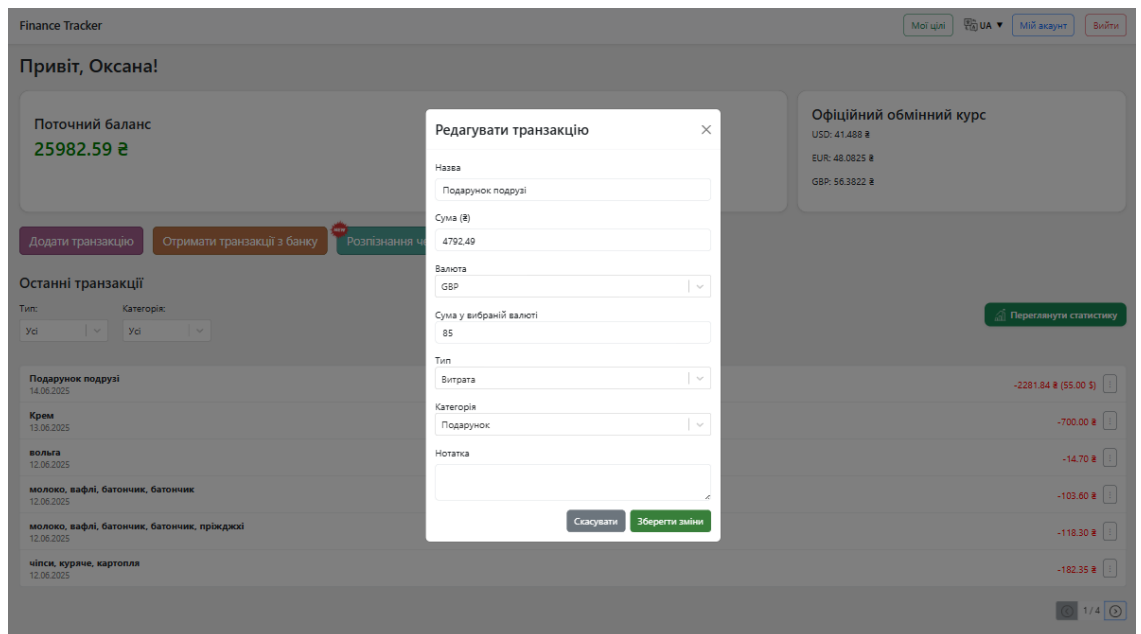


Рис 3.5 – Форма редагування транзакцій

Переходячи до автоматизованих видів додавання транзакцій, перша з них – виписка, яка отримується з вашої картки за місяць, можна реалізувати і за інший період, але буде обрано період саме 30 днів аби продемонструвати більш детальнішу роботи аналітики даних та можливість занести всі грошові операції одразу в спільну історію транзакцій та оновлення сторінки для відображення результатів (рис 3.6).

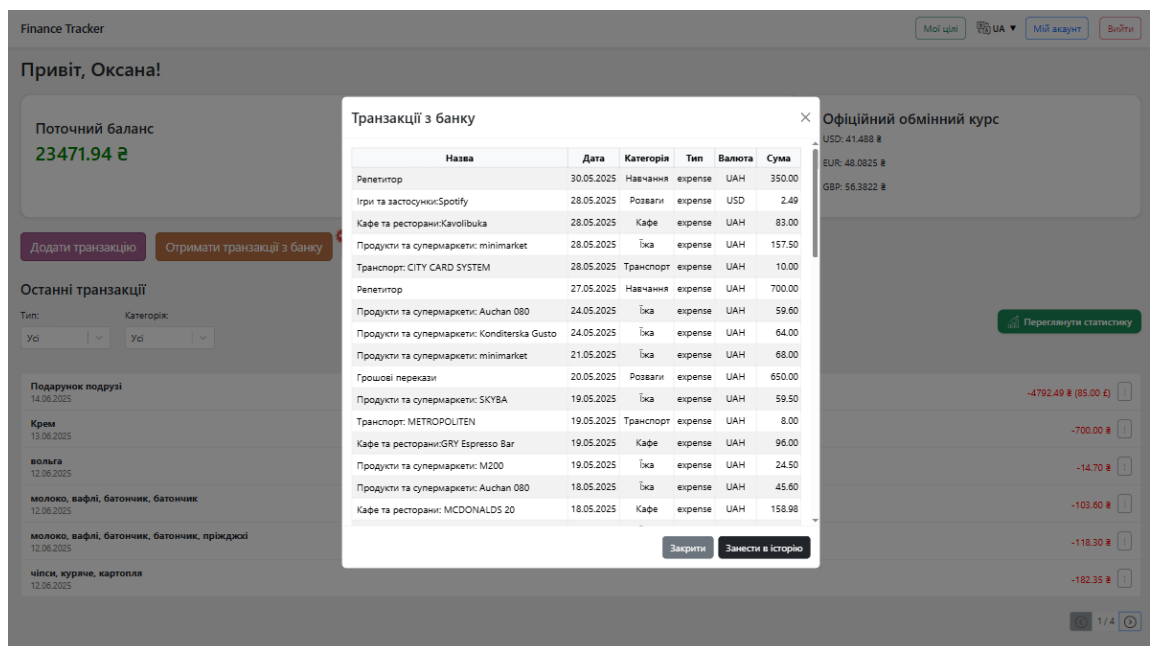


Рис 3.6 – Виписка з банківської картки

Після додання великої кількості транзакцій, можливість фільтрації є необхідною за допомогою якої користувач може полегшити сприйняття

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

інформації та здійснити швидкий пошук за необхідності. На даний момент фільтрація представлена за типом грошових операцій та всіх категорій конкретного користувача, включаючи ті, які він створює самостійно.

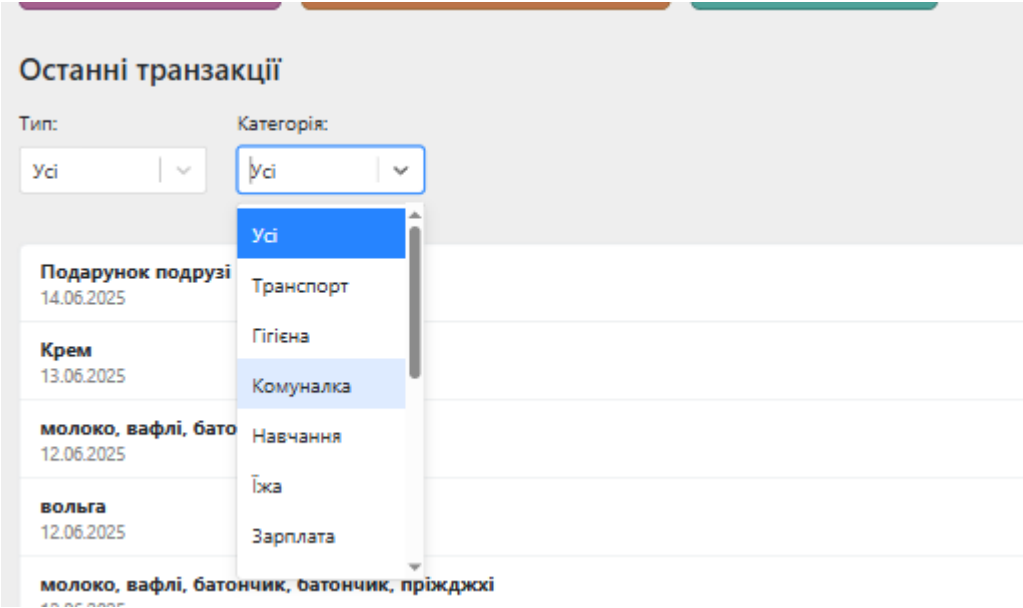


Рис 3.7 – Фільтрація фінансових платежів

Ще один з видів автоматизації процесу вводу даних – зчитування чеків. Клієнт має змогу завантажити на сторінку зображення чеку, наприклад з супермаркету та отримати рядки з відсканованою назвою товару та її ціною. Наразі успішне зчитування продуктів з чеку перевищує 70 відсотків, звичайно отримати вищий показник просканованих даних, залежить від багатьох факторів: якість зображення, тип чорнила, унікальність назви товару з можливим використанням спецсимволів може значно підвищити або знизити кількість правильно розпізнаних товарів. Але враховуючи можливу похибку, клієнту надана можливість редагування полів або взагалі не враховувати проблематично відскановані рядки у подальшу взаємодію з ними (рис 3.8).

Після збереження підтверджених товарів, система переходить до виконання категоризації з використанням штучного інтелекту (рис 3.9).

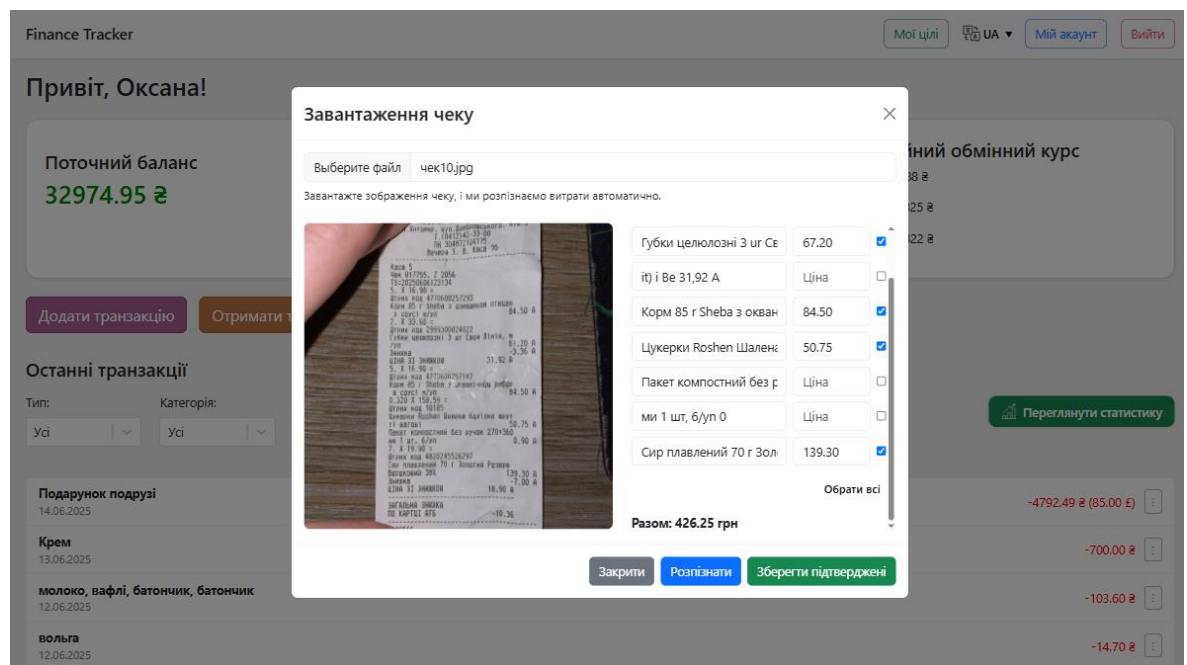


Рис 3.8 – Функціонал розпізнання чеків.

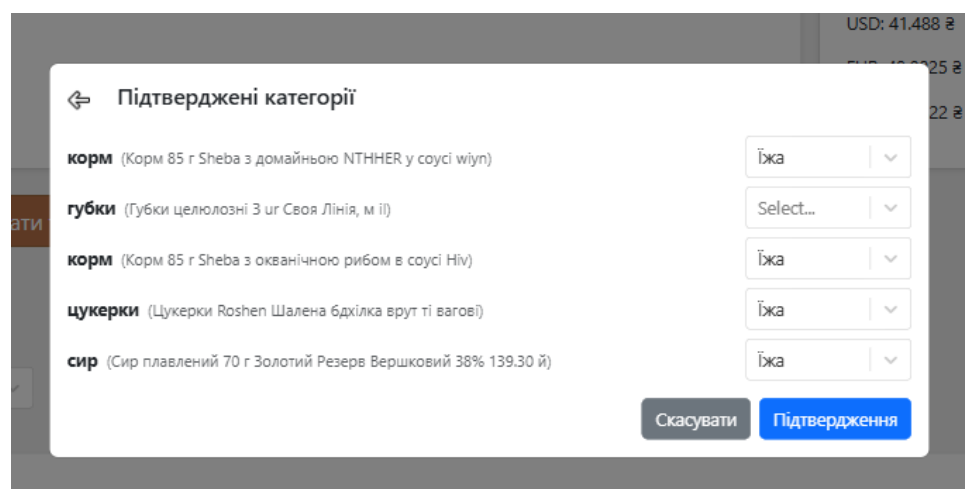


Рис 3.9 – Категоризація відсканованих товарів

Після підтвердження обраних товарів, додаток власноруч поділяє їх передачу в базу даних та запитання в спільну історії транзакцій, використовуючи групування продуктів за спільними категоріям, якщо ж вони всі різні, то вони будуть записані всі окремо.

Також, на платформі доступна функція редагування клієнта, в ній продемонстрована можливість зміни аватара користувача двома шляхами. Один з них, це вибір з вже існуючих іконок у веб-додатку, а інший – локальне додавання картинки з вашого пристрою, що зберігає картинку в системі для подальшої взаємодії з нею (рис 3.10).

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк. 50
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		

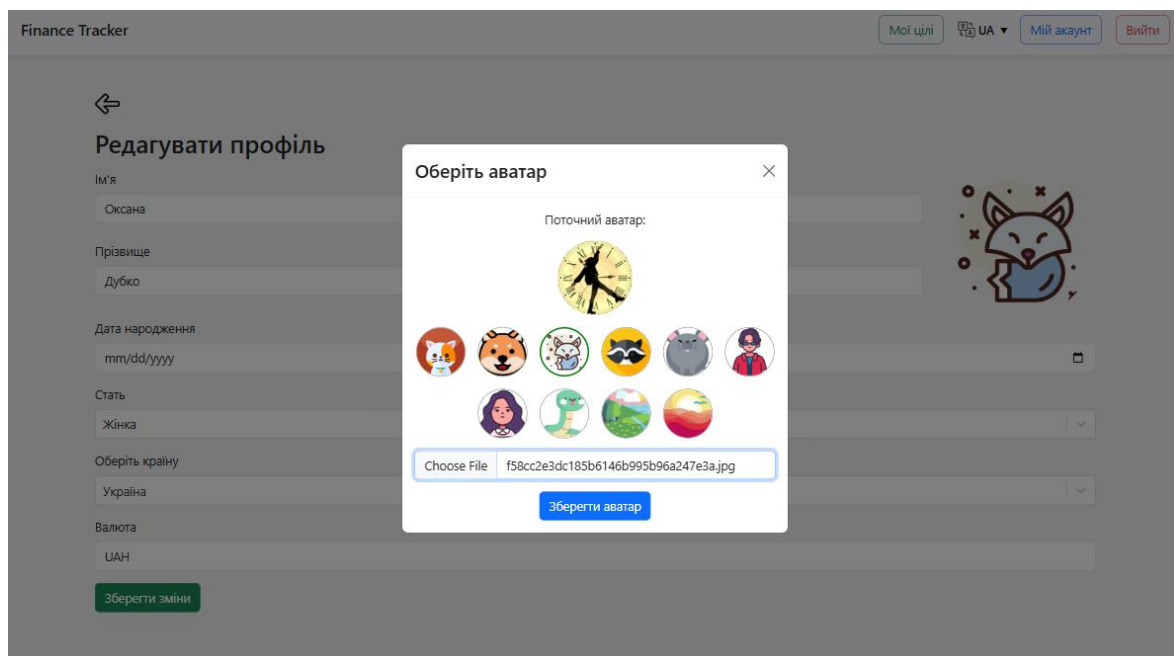


Рис 3.10 – Приклад зміни іконки користувача

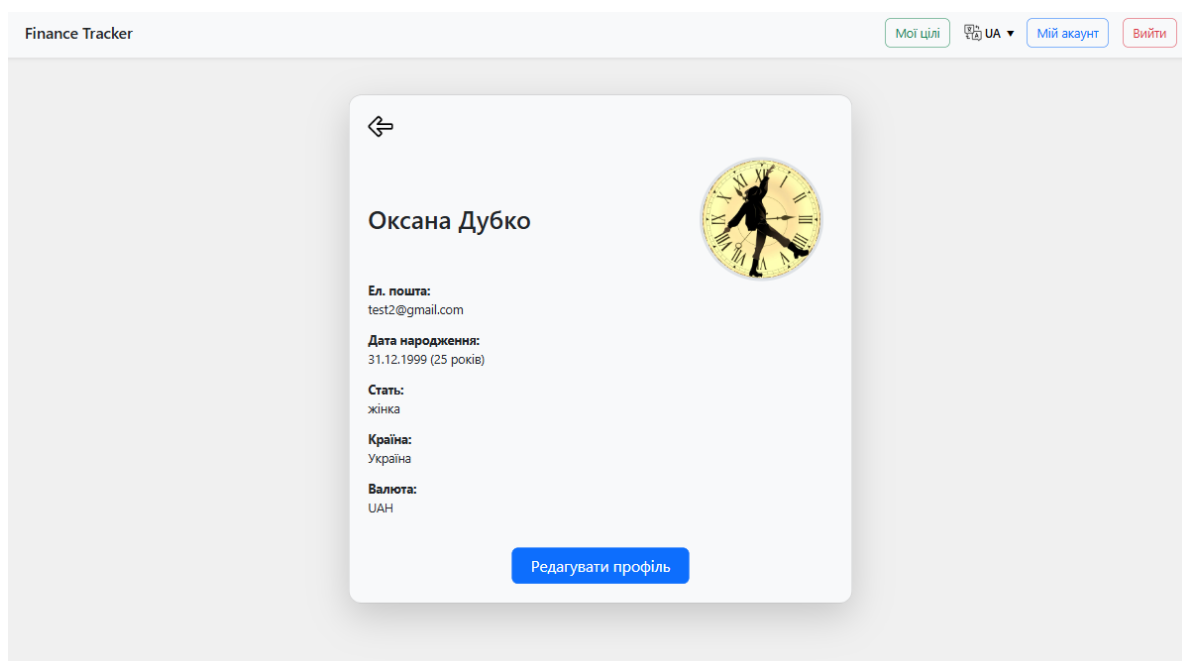


Рис 3.11 – Сторінка редагуванню профілю

У додатку також є можливість додавання матеріальної цілі, на сторінці представлена форма, поля якої повинні бути обов'язково правильно заповнені, пройдені валідацію і лише після цього додані в базу даних. На сторінці одразу можна переглянути список усіх раніше доданих цілей, їх дедлайн і статус (рис 3.12).

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

Рис 3.12 – Додавання цілі

Додаток прораховує логіку додавання дуже амбітних цілей, що перевищують поріг суми відсоткового значення ваших місячних доходів і надсилає вам повідомлення про можливість більш реалістичного досягнення цілі. Система надає точну дату після якої, клієнт здоровим образом зможе накопити кошти, якщо звичайно його дохід не буде змінюватись весь цей період (рис 3.13).

Рис 3.13 – Сповіщення системи про амбітність цілі

3.2 Статистика зборів та загальна статистика платформи

Останній функціонал, що представлений у веб-додатку – збір інформації про ваші витрати і доходи та перетворення цих даних у пряму статистику у вигляді графічних моделей, що є доволі простими у розумінні. Одна з них це

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

кругова діаграма, що фільтрується за особистими категоріями користувача та може опрацювати дані відповідно обраного періоду, вивівши на екран ще й і відсоткове значення представлених категорій відносно всіх витрат (рис 3.14).

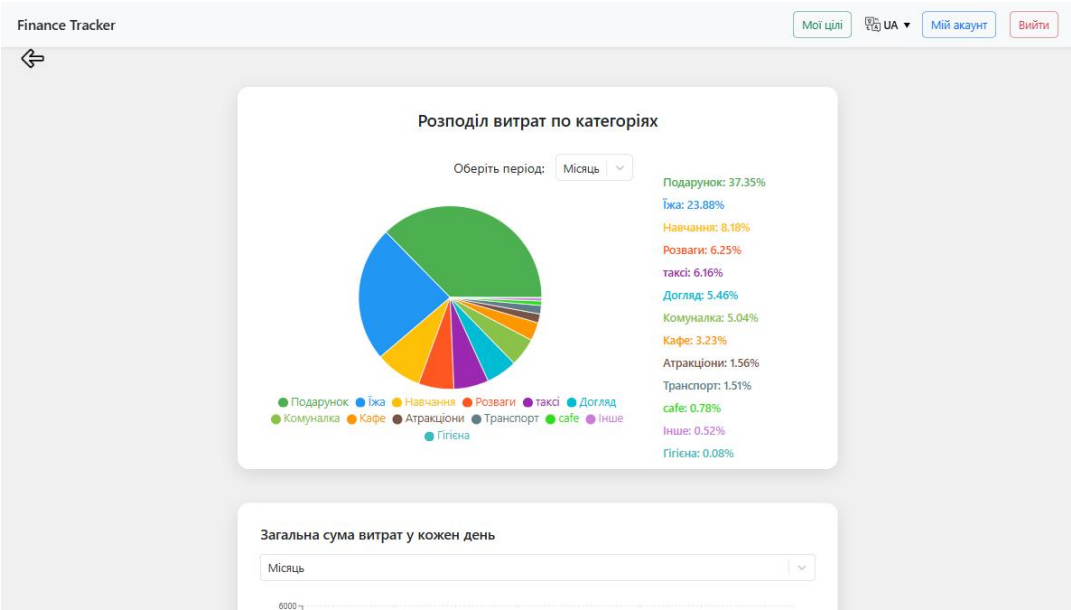


Рис 3.14 – Кругова діаграма розподілу витрат по категоріям

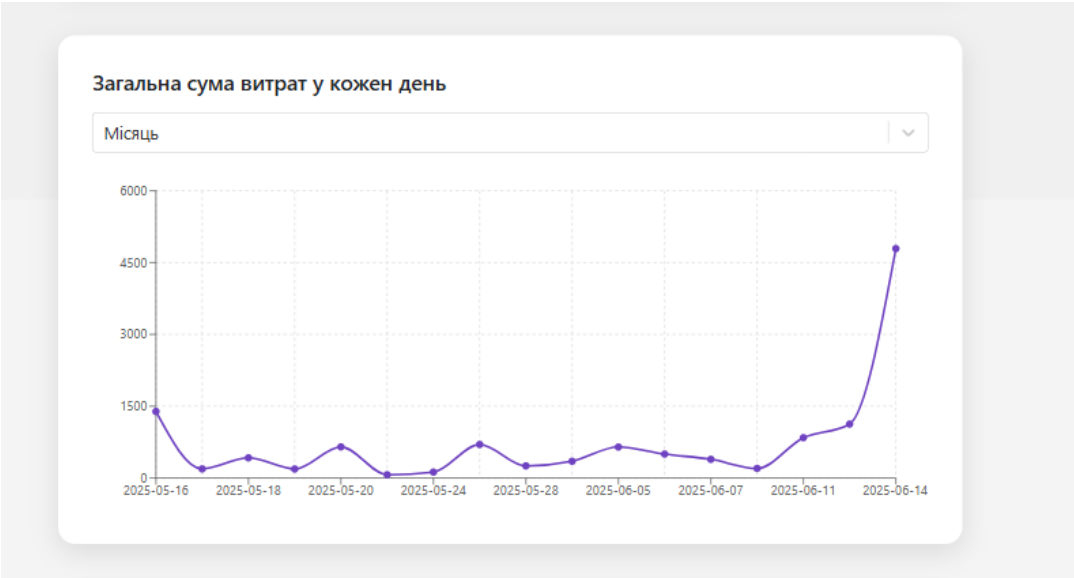


Рис 3.15 – Графік загальної суми витрат кожен день за вказаний період

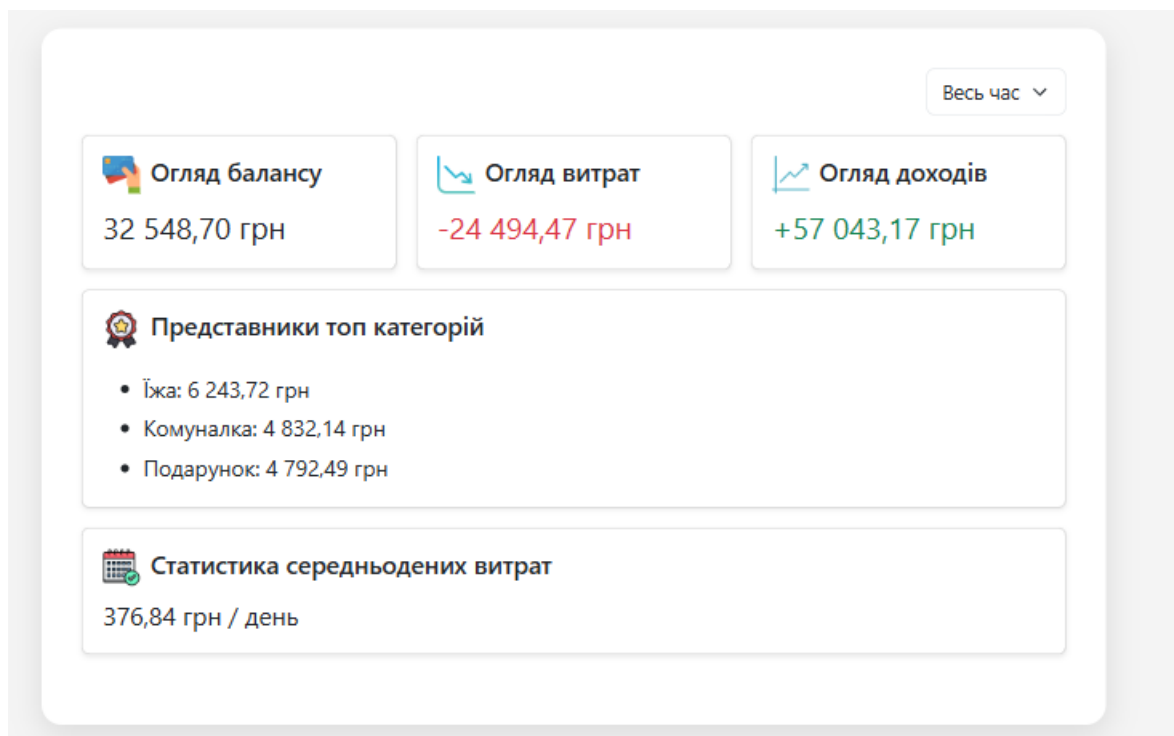


Рис 3.16 – Дашбоард різних даних користувача

На рис 3.15 представлений графік, що надає можливість користувачу переглянути загальну суму витрат кожен день за визначений період. Графік виглядає більш «живим» та реалістичним, коли статистичні дані будуть виводитись за довгий період, наприклад від 2 тижнів.

На рис 3.16 демонструється дашбоард в якому присутня інформація огляду надходжень та витрат, балансу, топ категорій витрат за обраний період часу та статичні дані щоденних витрат за вказаний час.

Для збільшення відсотку клієнтоорієнтованості на ринку суміжних програмних забезпечень, була реалізована можливість перегляду всіх аналітичних даних та візуально відображених графіків, що покращують розуміння динаміки грошових потоків користувача, а також надають можливість приймати більш раціональні рішення щодо подальшого планування бюджету (рис 3.17).

Окрім цього, зібрані дані, можна локально завантажити на ваш пристрій у .pdf форматі для зручного самостійного опрацювання.

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

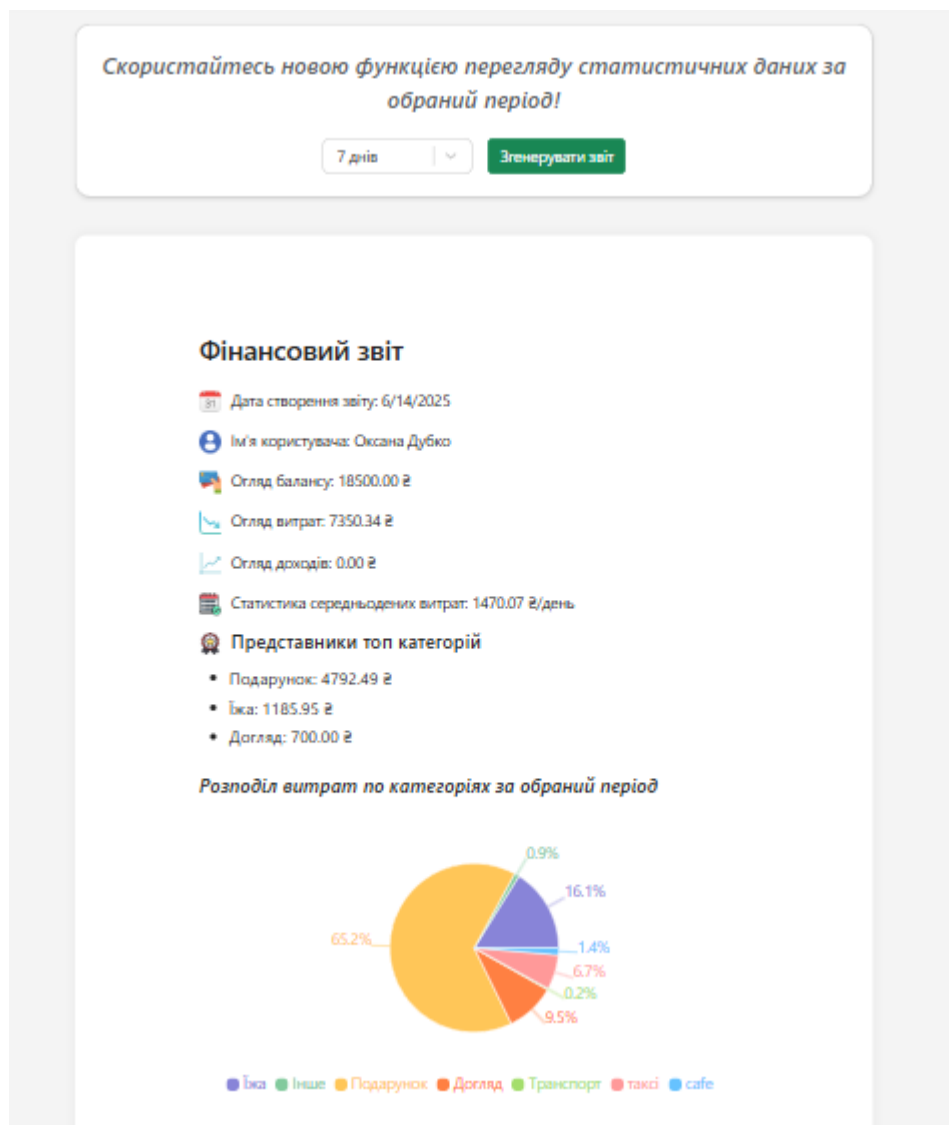


Рис 3.17 – Предперегляд оформленого статистичного звіту про транзакції

3.3 Тестування платформи

Тестування програмного забезпечення для планування бюджету за мету має перевірити правильність проходження різних тест кейсів, що створені на всіх етапах від автентифікації користувача до його виходу з системи. Для повноцінного тестування застосунку буде використано поєднання ручного проведення у різних браузерях та Postman для REST- запитів.

У таблиці 3.1 будуть знаходитись всі необхідні поля для тест-кейсів

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

Таблиця 3.1

№	Назва тесту	Вхідні дані	Очікуваний результат	Статус
1	Реєстрація нового користувача	Email: test@gmail.com Пароль: Password12345	Користувач створено успішно, збереження токена та його прив'язна до запитів	Pass
2	Вхід існуючого користувача	Існуючий email та пароль	Отримання JWT-токена, авторизація та перехід на головну сторінку	Pass
3	Додавання транзакції	Назва: «Кекс», тип: «витрата», валюта: UAH, сума: 100, категорія: «Їжа»	Поля пройшли валідацію, транзакцію додано в БД таблицю «transaction», відображення в списку	Pass
4	Додавання кастомної категорії	Назва: «Домашні улюбленці»	Поле пройшло валідацію, категорія зберігається в БД таблицю «custom_categories», відображення в списку категорій	Pass
5	Зміна мови інтерфейсу	Обрано мову «Англійська»	Оновлення сторінки, всі складові інтерфейсу перекладені	Pass
6	Зміна мови в профілі	Обрано мову «Німецька»	У БД поле language оновлюється, інтерфейс автоматично переключається	Pass

Продовження таблиці 3.1

7	Сканування чеків	Фото png, що завантажується з форми	Tesseract витягує поля товарів і ціну, відображення зчитаного матеріалу	Pass
8	Імпорт транзакцій з банківської системи	Масив транзакцій JSON, отриманий з Monobank API	Всі валідні транзакції збережено у БД «bank_trantasctions» через /batch з валютою	Pass
9	Категоризація товарів з чеку	Назва товару «Молоко Ферма 3,2% 980г»	GPT визначає ключове слово з рядку «молоко» і визначає категорію «Їжа»	Pass
10	Автоматичне додавання транзакцій	Підтвердження розпізнаних продуктів з чеку	Всі товари групуються за категоріями та конвертуються в транзакцію, їх додавання в БД	Pass
11	Генерація звіту статистики	Обрання періоду, натиснення на кнопку «Згенерувати звіт»	Створення PDF-файл з графіками, аналітичними даними та підсумками	Pass
12	Помилка під час категоризації товарів з GPT	Назва товару не зчитана, проблема на сервері GPT	Автоматично встановлюється категорія «Інше», вивід інформації про помилку	Pass

Висновки до третього розділу

Під час опрацювання цього розділу був реалізований детальний опис структури інтерфейсу веб-додатку для планування бюджету, що є ключовим аспектом у заохоченні людей користуватися саме цим програмним забезпеченням. Раціонально та практично побудований зовнішній вигляд

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

платформи, потенційно може підняти його актуальність серед населення країни та можна популяризація закордоном по причині використанні мультимовності застосунку.

Докладний опис функціоналу з яким зустрінеться клієнт, може потенційно допомогти йому більш глибоше розуміти вибір того чи іншого інструменту, який був використаний в інтуїтивно-зрозумілому інтерфейсі. Особливу увагу було приділено різним формат, таким як: логін, реєстрація, додавання транзакції та розпізнавання чеків, що є суттєвою важливою частиною у реалізації будь-якого типу веб-застосунків.

Також був проведений опис набору тест-кейсів, що допомогти протестувати середовище на критичні точки у взаємодії споживача з ними. Отримані результати, однозначно дали зрозуміти, що платформа відповідає вищезазначеним вимогам та здатна виконувати будь-який користувацький запит, що у комбінації цих двох аспектів, дає розуміння того, що програмне забезпечення готове до експлуатації потенційними користувачами у реальних умовах.

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Під час написання бакалаврської кваліфікаційної роботи було здійснено дослідження концептуальних і функціональних вимог для реалізації застосунків планування бюджету. На меті вищезазначеного аналізу було визначення ключових вимог до майбутнього програмного забезпечення, що включало детальну діагностику еквівалентних додатків на сьогоднішньому ринку. Виходячи з отриманих результатів – сформовано перелік функціональних і нефункціональних вимог, які стали підґрунтям для розробки архітектурного та візуального дизайну системи з метою опрацювання усіх запитів вимогливих користувачів.

У роботі був проведений детальний опис потенційно-можливої взаємодії нового користувача з програмним забезпеченням, описано імплементацію структурних елементів додатку та надано повний опис функціоналу інтерфейсу та зовнішнього вигляду, що є інтуїтивно-зрозумілим і легким у сприйнятті незалежно від віку та досвіду споживача.

За рахунок проведення достойного тестування з використанням браузерів, яке здійснювалось на базі тест-кейсів, що визначають ключові етапи у взаємодії споживачів з платформою, що у результаті надало застосунків без наявності проблем у використанні та помилок.

Результатом проведеної роботи є повністю готовий продукт до експлуатації, що надає можливість користувачу проводити керування особистими фінансами, можливість додавання платежів в ручному і автоматизованому варіанті з використанням інноваційних технологій та отримання статистичних даних у різних виглядах для їх самостійного опрацювання клієнтом.

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Прокопчук О.С., Вакалюк Т.А. Доцільність розробки додатку для персональних фінансів. Інформаційні технології – 2025: зб. тез XII Всеукраїнської науково-практичної конференції молодих учених, 15 трав. 2025 р., м. Київ / Київський столичний університет імені Бориса Грінченка; Відповід. за вип.: М.М. Астаф'єва, Д.М. Бодненко, О.М. Глушак, І.В. Машкіна, О.В. Локазюк, В.В. Прошкін, С.М. Шевченко. Київ : Київський столичний університет імені Бориса Грінченка, 2025. С. 185-187.
2. Прокопчук О.С., Вакалюк Т.А. Аналіз додатків для персональних фінансів. Інформаційні технології – 2025: зб. тез XII Всеукраїнської науково-практичної конференції молодих учених, 15 трав. 2025 р., м. Київ / Київський столичний університет імені Бориса Грінченка; Відповід. за вип.: М.М. Астаф'єва, Д.М. Бодненко, О.М. Глушак, І.В. Машкіна, О.В. Локазюк, В.В. Прошкін, С.М. Шевченко. Київ : Київський столичний університет імені Бориса Грінченка, 2025. С. 187-188.
3. Як пандемія COVID-19 вплинула на життя на економічні звички українців: [Електронний ресурс]. – Режим доступу: <https://salo.li/64604Bb>
4. Ukraine's Economy Faces Deep Strains Amid Prolonged War and Inflation: [Електронний ресурс]. – Режим доступу: <https://salo.li/0a16745>
5. React [Електронний ресурс]. – Режим доступу: <https://react.dev/>
6. Why user React? Top benefits for web development: [Електронний ресурс]. – Режим доступу: <https://www.netguru.com/blog/why-use-react>
7. 10 Key reasons why you should user React for web development: [Електронний ресурс]. – Режим доступу: <https://www.techmagic.co/blog/why-we-use-react-js-in-the-development>
8. Personal Finance Management Application: [Електронний ресурс]. – Режим доступу: <https://salo.li/B72719b>
9. AI powered expense & budget tracker: Web based financial planning system: [Електронний ресурс]. – Режим доступу:

		Прокопчук О.С.,			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

<https://www.ijstjournal.com/article/AI-Powered-Expense-&-Budget-Tracker:-A-Web-Based-Financial-Planning-System>

10. How to build a budgeting app: opportunities, challenges and practical tips: [Електронний ресурс]. – Режим доступу: <https://leobit.com/blog/how-to-build-a-budgeting-app-opportunities-challenges-and-practical-tips/>

11. Bootstrap: [Електронний ресурс]. – Режим доступу: <https://getbootstrap.com/>

12. What is JSON Web Token? : [Електронний ресурс]. – Режим доступу: <https://jwt.io/introduction>

13. HTTP протокол: [Електронний ресурс]. – Режим доступу: <https://hostiq.ua/wiki/ukr/http-https/>

14. Money Manager: [Електронний ресурс]. – Режим доступу: <https://www.realbyteapps.com/>

15. Wallet by budgetbakers: [Електронний ресурс]. – Режим доступу: <https://budgetbakers.com/>

16. Monefy: [Електронний ресурс]. – Режим доступу: <https://www.monefy.com/>

17. Saldo Finance: [Електронний ресурс]. – Режим доступу: <https://saldofinanceapp.com/>

18. Fast Budget: [Електронний ресурс]. – Режим доступу: <https://fastbudget.app/>

19. Money Lover: [Електронний ресурс]. – Режим доступу: <https://moneylover.me/>

20. State of Mobile Finance Apps 2024: [Електронний ресурс]. – Режим доступу: https://sensortower.com/blog/state-of-mobile-finance-apps-2024?utm_source=chatgpt.com

21. PostgreSQL: [Електронний ресурс]. – Режим доступу: <https://www.postgresql.org/>

		Прокопчук О.С.,			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

22. Клієнт-серверна архітектура: [Електронний ресурс]. – Режим доступу: <https://training.qatestlab.com/blog/technical-articles/client-server-architecture/>
23. Tailwinds CSS: [Електронний ресурс]. – Режим доступу: <https://tailwindcss.com/>
24. Node.js: [Електронний ресурс]. – Режим доступу: <https://nodejs.org/uk>
25. Express.js: [Електронний ресурс]. – Режим доступу: <https://expressjs.com/>
26. Tesseract OCR: [Електронний ресурс]. – Режим доступу: <https://github.com/tesseract-ocr/tesseract>
27. ACID: [Електронний ресурс]. – Режим доступу: <https://www.databricks.com/glossary/acid-transactions>
28. Що таке Revolut та як він працює: [Електронний ресурс]. – Режим доступу: <https://rates.fm/ua-uk/payment-systems/who-take-revolut-ta-yak-vin-pracyuye/>
29. Citus: [Електронний ресурс]. – Режим доступу: <https://www.citusdata.com/>
30. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns : Elements of Reusable Object-Oriented Software. – Boston : Addison-Wesley, 1995. –395с.–doi:10.5555/186897

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

ДОДАТКИ

		Прокопчук О.С..			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

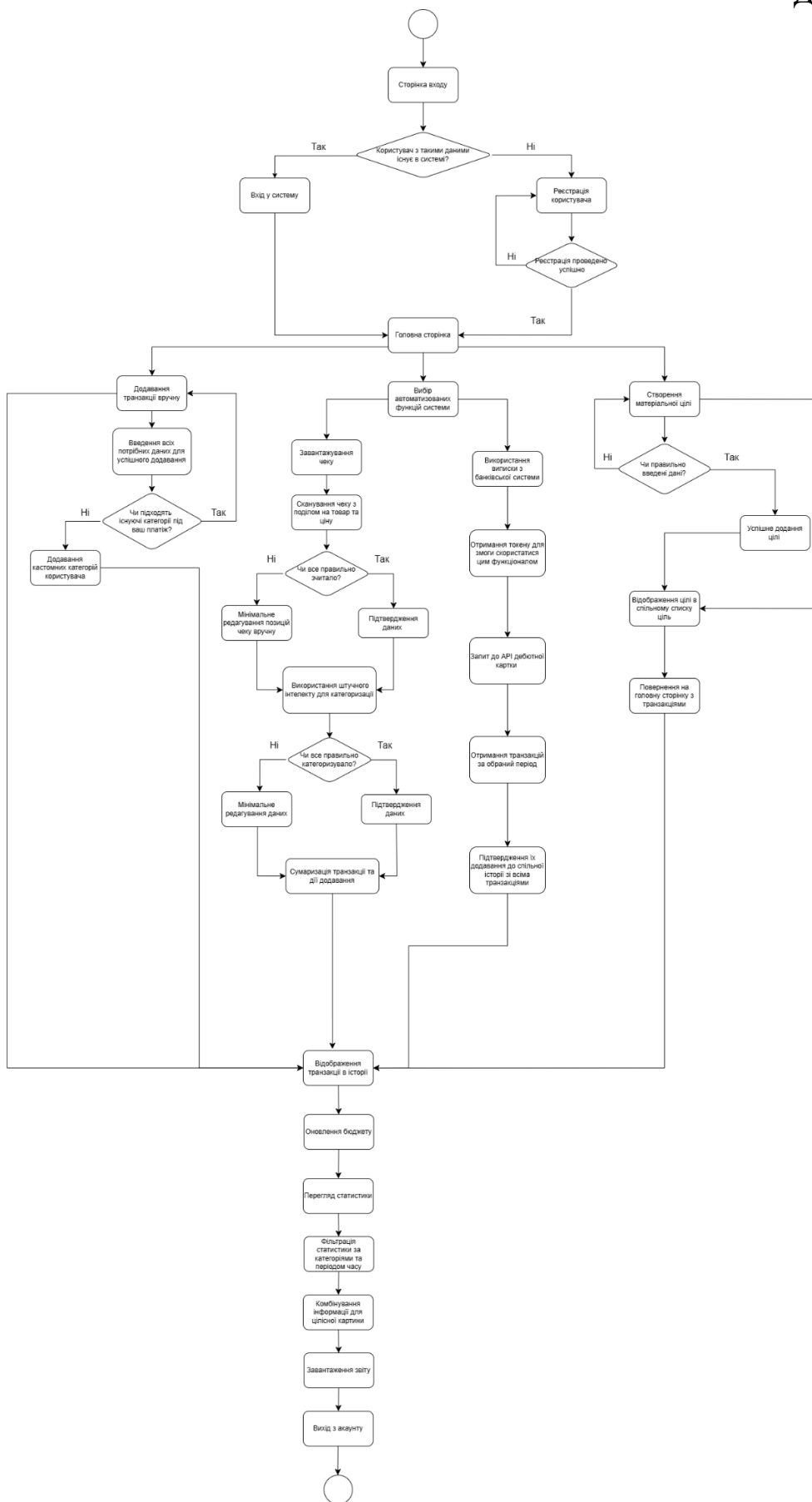


Рисунок А.1 – Повнорозмірна діаграма активності

```
Auth.js
const express = require("express");
const router = express.Router();
const multer = require("multer");
const fs = require("fs");
const path = require("path");
const pool = require("../db");
const dotenv = require("dotenv");

const jwt = require("jsonwebtoken");
const bcrypt = require("bcrypt");
require("dotenv").config();

const authenticateToken = require("../security/authenticateToken");

const upload = multer({ storage: multer.memoryStorage() });

dotenv.config();

const nameRegex = /^[A-Za-zА-Яа-яіїїЄІГґ's'-]+$/;
const emailRegex = /^[^\s@]+\@[^\s@]+\.[^\s@]+$/;
// Реєстрація користувача
router.post('/register', async (req, res) => {
    const { firstName, lastName, email, password } = req.body;

    if (!firstName || !nameRegex.test(firstName.trim())) {
        return res.status(400).json({ message: 'Некоректне ім'я' });
    }

    if (!lastName || !nameRegex.test(lastName.trim())) {
        return res.status(400).json({ message: 'Некоректне прізвище' });
    }

    if (!email || !emailRegex.test(email.trim())) {
        return res.status(400).json({ message: 'Некоректна електронна пошта' });
    }

    if (!password || password.length < 6) {
        return res.status(400).json({ message: 'Пароль має містити щонайменше 6 символів' });
    }

    try {
```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    const existingUser = await pool.query('SELECT * FROM users WHERE email
= $1', [email]);
    if (existingUser.rows.length > 0) {
        return res.status(400).json({ message: 'Користувач з такою поштою вже
існує' });
    }

    const hashedPassword = await bcrypt.hash(password, 10);

    const newUser = await pool.query(
        'INSERT INTO users (first_name, last_name, email, password) VALUES ($1,
$2, $3, $4) RETURNING *',
        [firstName, lastName, email, hashedPassword]
    );

    const user = newUser.rows[0];

    const token = jwt.sign(
        { id: user.id },
        process.env.JWT_SECRET || "secretkey",
        { expiresIn: "7d" }
    );

    delete user.password;
    res.status(201).json({
        user,
        token,
    });
} catch (error) {
    console.error(error.message);
    res.status(500).json({ message: 'Помилка сервера при створенні користувача'
});
}
});

// Вхід користувача
router.post('/login', async (req, res) => {
    const { email, password } = req.body;

    if (!email || !password) {
        return res.status(400).json({ message: 'Email і пароль обов'язкові' });
    }

    try {

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const existingUser = await pool.query('SELECT * FROM users WHERE email
= $1', [email]);
if (existingUser.rows.length === 0) {
    return res.status(400).json({ message: 'Користувача не знайдено' });
}

const user = existingUser.rows[0];

const isPasswordValid = await bcrypt.compare(password, user.password);
if (!isPasswordValid) {
    return res.status(401).json({ message: 'Невірний пароль' });
}

// Створення токена
const token = jwt.sign(
    { id: user.id, email: user.email },
    process.env.JWT_SECRET,
    { expiresIn: '7d' }
);

res.status(200).json({
    token,
    user: {
        id: user.id,
        first_name: user.first_name,
        last_name: user.last_name,
        email: user.email,
        avatar: user.avatar,
        language: user.language,
        currency: user.currency,
    }
});
} catch (error) {
    console.error('Помилка входу:', error.message);
    res.status(500).json({ message: 'Помилка сервера під час входу' });
}
});

// Захищений маршрут без :id у URL
router.put("/profile", authenticateToken, async (req, res) => {
    const id = req.user.id;
    const allowedFields = ["first_name", "last_name", "birth_date", "gender",
"country", "currency"];
    const fieldsToUpdate = [];
    const values = [];

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

```

const allowedGenders = ["чоловік", "жінка", "інше"];

if (field === "gender" && value && !allowedGenders.includes(value)) {
  return res.status(400).json({ message: "Некоректна стаття. Доступні: чоловік, жінка, інше" });
}

const nameRegex = /^[A-Za-zA-Яа-яІіЇіЄєҐґ's'-]+$/;

for (const field of allowedFields) {
  const value = req.body[field];

  if (value !== undefined) {
    if ((field === "first_name" || field === "last_name") && value.trim() !== "") {
      if (!nameRegex.test(value)) {
        return res.status(400).json({
          message: `Поле ${field} повинно містити лише літери (без цифр або символів)`
        });
      }
    }
  }

  if (field === "birth_date" && value.trim() === "") {
    fieldsToUpdate.push(`${field} = ${values.length + 1}`);
    values.push(null);
  } else {
    fieldsToUpdate.push(`${field} = ${values.length + 1}`);
    values.push(value);
  }
}

if (fieldsToUpdate.length === 0) {
  return res.status(400).json({ message: "Немає даних для оновлення" });
}

values.push(id);

const query = `
UPDATE users
SET ${fieldsToUpdate.join(", ")}
WHERE id = ${values.length}
RETURNING *`;

try {

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

```

const result = await pool.query(query, values);
res.json(result.rows[0]);
} catch (err) {
  console.error("Помилка оновлення профілю:", err);
  res.status(500).json({ message: "Не вдалося оновити профіль" });
}
});

// Зміна виключно аватара користувача
router.put("/profile/upload-avatar", authenticateToken,
upload.single("avatar_file"), async (req, res) => {
  const userId = req.user.id;
  let avatarPath = req.body.avatar;

  try {
    if (req.file) {
      const dir = path.join(__dirname, "..", "public", "avatars");

      if (!fs.existsSync(dir)) {
        fs.mkdirSync(dir, { recursive: true });
      }

      const fileName = `${Date.now()}_${req.file.originalname}`;
      const savePath = path.join(dir, fileName);
      fs.writeFileSync(savePath, req.file.buffer);

      avatarPath = `avatars/${fileName}`;
    }

    const result = await pool.query(
      `UPDATE users SET avatar = $1 WHERE id = $2 RETURNING *`,
      [avatarPath, userId]
    );

    res.json(result.rows[0]);
  } catch (err) {
    console.error("Помилка оновлення аватара:", err);
    res.status(500).json({ message: "Не вдалося оновити аватар", error:
err.message });
  }
});

// Збереження та зміна мови при умовах
router.put('/profile/language', authenticateToken, async (req, res) => {
  console.log("req.user:", req.user);

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

```

const userId = req.user.id;
const { language } = req.body;

try {
  const updatedUser = await pool.query(
    'UPDATE users SET language = $1 WHERE id = $2 RETURNING *',
    [language, userId]
  );
  res.json(updatedUser.rows[0]);
} catch (err) {
  console.error('Помилка оновлення мови:', err);
  res.status(500).json({ message: 'Не вдалося оновити мову' });
}
});

router.get("/users/me", authenticateToken, async (req, res) => {
  const userId = req.user.id;

  try {
    const result = await pool.query(
      "SELECT id, first_name, last_name FROM users WHERE id = $1",
      [userId]
    );

    if (result.rows.length === 0) {
      return res.status(404).json({ message: "User not found" });
    }

    res.json(result.rows[0]);
  } catch (err) {
    console.error("DB error:", err);
    res.status(500).json({ message: "Server error" });
  }
});
module.exports = router;

Utils/Monobank.js:
const axios = require("axios");
require("dotenv").config();

const MONOBANK_API = "https://api.monobank.ua/personal";

async function getClientInfo() {
  const response = await axios.get(`${MONOBANK_API}/client-info`, {
    headers: {

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

```

    "X-Token": process.env.MONOBANK_TOKEN,
  },
});
return response.data;
}

async function getTransactions(from, to) {
  const accountId = await getAccountId();
  const response = await
  axios.get(`${MONOBANK_API}/statement/${accountId}/${from}/${to}`, {
    headers: {
      "X-Token": process.env.MONOBANK_TOKEN,
    },
  });
  return response.data;
}

async function getAccountId() {
  const client = await getClientInfo();
  return client.accounts[0].id;
}

module.exports = {
  getClientInfo,
  getTransactions,
};

```

Bank.js

```

const express = require("express");
const router = express.Router();
const { getClientInfo, getTransactions } = require("../utils/monobank");
const pool = require("../db");

```

```

router.get("/client", async (req, res) => {
  try {
    const data = await getClientInfo();
    res.json(data);
  } catch (err) {
    res.status(500).json({ error: "Monobank API error" });
  }
});

```

```

router.get("/transactions", async (req, res) => {
  try {

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

```

const now = Math.floor(Date.now() / 1000);
const sevenDaysAgo = now - 7 * 24 * 60 * 60;

const data = await getTransactions(sevenDaysAgo, now);
res.json(data);
} catch (err) {
  console.error(err.message);
  res.status(500).json({ error: "Failed to fetch transactions" });
}
});

router.get("/bank-transactions", async (req, res) => {
  try {
    const userId = 2; // або Number(req.query.userId)
    const result = await pool.query(
      "SELECT * FROM bank_transactions WHERE user_id = $1 AND
is_saved_to_history = false ORDER BY date DESC",
      [userId]
    );
    res.json(result.rows);
  } catch (err) {
    console.error(" ПОМИЛКА ЗАПИТУ ДО БД:", err);
    res.status(500).json({ error: "Помилка при отриманні транзакцій" });
  }
});

router.post("/transfer-to-history", async (req, res) => {
  try {
    const userId = 2;

    const result = await pool.query(
      "SELECT * FROM bank_transactions WHERE user_id = $1 AND
is_saved_to_history = false",
      [userId]
    );

    const transactions = result.rows;

    for (const tx of transactions) {
      await pool.query(
        `INSERT INTO transactions (user_id, title, amount, currency, category,
date, type)
VALUES ($1, $2, $3, $4, $5, $6, $7)`,
        [userId, tx.title, tx.amount, tx.currency, tx.category, tx.date, tx.type]
      );
    }
  }
});

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

    await pool.query(
      `UPDATE bank_transactions SET is_saved_to_history = true WHERE id =
$1`,
      [tx.id]
    );
  }
  res.json({ success: true, count: transactions.length });

} catch (err) {
  console.error(" !!ПОМИЛКА при збереженні в історію:", err);
  res.status(500).json({ error: "Помилка при збереженні в історію" });
}
});
module.exports = router;

```

Transactions.js:

```

const express = require("express");
const router = express.Router();
const pool = require("../db");
const authenticateToken = require("../security/authenticateToken");
const { validateTransactionInput } = require("../utils/validators");

// додати транзакцію
router.post("/", authenticateToken, async (req, res) => {
  const userId = req.user.id;
  const { title, amount, original_amount, type, category, note, date, currency } =
    req.body;

  const errors = validateTransactionInput(req.body);
  if (errors.length > 0) {
    return res.status(400).json({ message: errors.join(", ") });
  }

  try {
    const result = await pool.query(
      `INSERT INTO transactions (
        user_id, title, amount, original_amount, type, category, note, date, currency,
        created_at
      )
      VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, DEFAULT)
      RETURNING *`,
      [userId, title.trim(), amount, original_amount, type, category || null, note || null,
        date || null, currency]
    );
  }
}

```

```

    res.json(result.rows[0]);
  } catch (err) {
    console.error("Помилка додавання:", err);
    res.status(500).json({ message: "Не вдалося додати транзакцію" });
  }
});

// отримати транзакцію
router.get("/", authenticateToken, async (req, res) => {
  console.log("Отримано user з токена:", req.user);
  const userId = req.user.id;

  try {
    const result = await pool.query(
      `SELECT * FROM transactions WHERE user_id = $1 ORDER BY created_at
DESC`,
      [userId]
    );
    res.json(result.rows);
  } catch (err) {
    console.error("Помилка отримання транзакцій:", err);
    res.status(500).json({ message: "Не вдалося отримати транзакції" });
  }
});

// Видалити транзакцію
router.delete("/:id", authenticateToken, async (req, res) => {
  const { id } = req.params;
  const userId = req.user.id;

  try {
    const result = await pool.query(
      `DELETE FROM transactions WHERE id = $1 AND user_id = $2
RETURNING *`,
      [id, userId]
    );

    if (result.rowCount === 0) {
      return res.status(404).json({ message: "Транзакцію не знайдено або вона не
ваша" });
    }

    res.json({ message: "Транзакцію видалено успішно" });
  } catch (err) {
    console.error("Помилка видалення:", err);
  }
});

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

    res.status(500).json({ message: "Не вдалося видалити транзакцію" });
  }
});

// Оновити транзакцію
router.put("/:id", authenticateToken, async (req, res) => {
  const { id } = req.params;
  const userId = req.user.id;
  const { title, amount, original_amount, type, category, note, currency } =
    req.body;

  const errors = validateTransactionInput(req.body);
  if (errors.length > 0) {
    return res.status(400).json({ message: errors.join(", ") });
  }

  try {
    const result = await pool.query(
      `UPDATE transactions
      SET title = $1,
          amount = $2,
          original_amount = $3,
          type = $4,
          category = $5,
          note = $6,
          currency = $7
      WHERE id = $8 AND user_id = $9
      RETURNING *`,
      [title.trim(), amount, original_amount, type, category || null, note || null,
        currency, id, userId]
    );

    if (result.rows.length === 0) {
      return res.status(404).json({ message: "Транзакція не знайдена або не ваша"
    });
    }

    res.json(result.rows[0]);
  } catch (err) {
    console.error("Помилка оновлення транзакції:", err);
    res.status(500).json({ message: "Не вдалося оновити транзакцію" });
  }
});

```

```
// додати список транзакцій одразу (batch insert)
router.post("/batch", authenticateToken, async (req, res) => {
  const userId = req.user.id;
  const transactions = req.body;

  if (!Array.isArray(transactions)) {
    return res.status(400).json({ message: "Очікується масив транзакцій" });
  }

  const validTransactions = transactions.filter(tx => {
    return tx.title && tx.amount && tx.type && tx.category && tx.currency;
  });

  if (validTransactions.length === 0) {
    return res.status(400).json({ message: "Немає жодної валідної транзакції" });
  }

  const values = validTransactions.map(tx => [
    userId,
    tx.title.trim(),
    tx.amount,
    tx.original_amount || tx.amount,
    tx.type,
    tx.category,
    tx.note || null,
    tx.date || new Date().toISOString().slice(0, 10),
    tx.currency,
  ]);

  const placeholders = values
    .map(
      (_, i) =>
        `($${i * 9 + 1}, $${i * 9 + 2}, $${i * 9 + 3}, $${i * 9 + 4}, $${i * 9 + 5},
        $${i * 9 + 6}, $${i * 9 + 7}, $${i * 9 + 8}, $${i * 9 + 9}, DEFAULT)`
    )
    .join(", ");

  const flatValues = values.flat();

  try {
    const result = await pool.query(
      `
      INSERT INTO transactions (
        user_id, title, amount, original_amount, type, category, note, date, currency,
        created_at`
    );
  }
}
```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

    ) VALUES ${placeholders}
    RETURNING *
    \,
    flatValues
  );

  res.status(201).json({ transactions: result.rows });
} catch (err) {
  console.error("Помилка batch вставки:", err);
  res.status(500).json({ message: "Не вдалося додати транзакції" });
}
});

module.exports = router;

Utils/fetchExchangeRates.js
const axios = require("axios");
const cheerio = require("cheerio");

async function fetchExchangeRates() {
  try {
    const { data } = await
    axios.get("https://bank.gov.ua/ua/markets/exchangerates");
    const $ = cheerio.load(data);

    const rates = {};
    $("tbody tr").each((_ , row) => {
      const currency = $(row).find("td").eq(1).text().trim();
      const rate = $(row).find("td").eq(4).text().replace(",", ".").trim();

      if (["USD", "EUR", "GBP"].includes(currency)) {
        rates[currency] = parseFloat(rate);
      }
    });

    return rates;
  } catch (error) {
    console.error("Помилка при парсингу курсу валют:", error);
    return null;
  }
}

module.exports = fetchExchangeRates;
Exchange.js:
const express = require("express");

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const router = express.Router();
const fetchExchangeRates = require("../utils/fetchExchangeRates");

router.get("/exchange/rates", async (req, res) => {
  try {
    const rates = await fetchExchangeRates(); // Викликаємо функцію парсингу
    if (rates) {
      res.json(rates); // Повертаємо курси валют у форматі JSON
    } else {
      res.status(500).json({ message: "Не вдалося отримати курси валют" });
    }
  } catch (error) {
    console.error("Помилка при отриманні курсів валют:", error);
    res.status(500).json({ message: "Помилка при обробці запиту" });
  }
});

module.exports = router;

```

Goals.js

```

const express = require("express");
const router = express.Router();
const pool = require("../db");
const authenticateToken = require("../security/authenticateToken");

// GET: отримати всі цілі користувача
router.get("/", authenticateToken, async (req, res) => {
  const userId = req.user.id;
  try {
    const result = await pool.query(
      `SELECT * FROM goals WHERE user_id = $1 ORDER BY created_at DESC`,
      [userId]
    );
    res.json(result.rows);
  } catch (err) {
    console.error("Помилка при отриманні цілей:", err);
    res.status(500).json({ message: "Не вдалося отримати цілі" });
  }
});

// POST: додати нову ціль
router.post("/", authenticateToken, async (req, res) => {
  const userId = req.user.id;
  const { title, target_amount, deadline, note } = req.body;

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const titleRegex = /^[A-Za-zA-Яа-яІііЄєҐґ's-]{2,50}$/;
if (!title || !titleRegex.test(title.trim())) {
    return res.status(400).json({ message: "Назва цілі має містити лише літери і
бути від 2 до 50 символів" });
}

if (target_amount === undefined || isNaN(target_amount) ||
Number(target_amount) < 0) {
    return res.status(400).json({ message: "Сума цілі повинна бути додатнім
числом" });
}

if (deadline && !/^\\d{4}-\\d{2}-\\d{2}$/.test(deadline)) {
    return res.status(400).json({ message: "Невірний формат дати.
Використовуйте YYYY-MM-DD" });
}

try {
    const result = await pool.query(
        `INSERT INTO goals (user_id, title, target_amount, deadline, note)
        VALUES ($1, $2, $3, $4, $5) RETURNING *`,
        [userId, title.trim(), target_amount, deadline || null, note || null]
    );
    res.json(result.rows[0]);
} catch (err) {
    console.error("Помилка при додаванні цілі:", err);
    res.status(500).json({ message: "Не вдалося додати ціль" });
}
});

// PUT: оновити ціль
router.put("/:id", authenticateToken, async (req, res) => {
    const userId = req.user.id;
    const { id } = req.params;
    const { title, target_amount, current_amount, deadline, note } = req.body;

    const titleRegex = /^[A-Za-zA-Яа-яІііЄєҐґ's-]{2,50}$/;
    if (!title || !titleRegex.test(title.trim())) {
        return res.status(400).json({ message: "Назва цілі має містити лише літери і
бути від 2 до 50 символів" });
    }

    if (target_amount === undefined || isNaN(target_amount) ||
Number(target_amount) < 0) {

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    return res.status(400).json({ message: "Сума цілі повинна бути додатнім
числом" });
}

if (current_amount !== undefined && (isNaN(current_amount) ||
Number(current_amount) < 0)) {
    return res.status(400).json({ message: "Поточна сума повинна бути числом не
менше 0" });
}

if (deadline && !/^\\d{4}-\\d{2}-\\d{2}$/.test(deadline)) {
    return res.status(400).json({ message: "Невірний формат дати.
Використовуйте YYYY-MM-DD" });
}

try {
    const result = await pool.query(
        `UPDATE goals
        SET title=$1, target_amount=$2, current_amount=$3, deadline=$4, note=$5
        WHERE id=$6 AND user_id=$7
        RETURNING *`,
        [title.trim(), target_amount, current_amount || 0, deadline || null, note || null, id,
userId]
    );
    res.json(result.rows[0]);
} catch (err) {
    console.error("Помилка при оновленні цілі:", err);
    res.status(500).json({ message: "Не вдалося оновити ціль" });
}
});

// DELETE: видалити ціль
router.delete("/:id", authenticateToken, async (req, res) => {
    const userId = req.user.id;
    const { id } = req.params;
    try {
        const result = await pool.query(
            `DELETE FROM goals WHERE id = $1 AND user_id = $2`,
            [id, userId]
        );
    }
    if (result.rowCount === 0) {
        return res.status(404).json({ message: "Ціль не знайдена або не належить
вам" });
    }
    res.json({ message: "Ціль видалено" });
});

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

    } catch (err) {
      console.error("Помилка при видаленні цілі:", err);
      res.status(500).json({ message: "Не вдалося видалити ціль" });
    }
  });

```

```
module.exports = router;
```

AnalyticsChart.jsx

```

import React, { useState, useMemo } from "react";
import { useTranslation } from "react-i18next";
import Select from "react-select";
import {
  ResponsiveContainer,
  LineChart,
  Line,
  XAxis,
  YAxis,
  Tooltip,
  CartesianGrid,
} from "recharts";

```

```

export default function AnalyticsCharts({ transactions }) {
  const { t } = useTranslation();
  const [period, setPeriod] = useState(30);

```

```

  const periodOptions = [
    { label: t("period.7d"), value: 7 },
    { label: t("period.14d"), value: 14 },
    { label: t("period.30d"), value: 30 },
    { label: t("period.365d"), value: 365 },
    { label: t("period.all"), value: "all" },
  ];

```

```

  const chartData = useMemo(() => {
    let filtered = transactions.filter((tx) => tx.type === "expense");

```

```

    if (period !== "all") {
      const now = new Date();
      const fromDate = new Date(now);
      fromDate.setDate(now.getDate() - period);

```

```

    filtered = filtered.filter((tx) => {
      const date = new Date(tx.date);

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    return date >= fromDate;
  });
}

const dailyTotals = {};

filtered.forEach((tx) => {
  const key = new Date(tx.date).toISOString().split("T")[0];
  dailyTotals[key] = (dailyTotals[key] || 0) + parseFloat(tx.amount) || 0;
});

const sortedDates = Object.keys(dailyTotals).sort();

return sortedDates.map((date) => ({ date, amount: dailyTotals[date] }));
}, [transactions, period]);

return (
  <div>
    <h5 className="mb-3">{t("statistics.daily_dynamics")}</h5>

    <div style={{ marginBottom: "1rem" }}>
      <Select
        options={periodOptions}
        value={periodOptions.find((opt) => opt.value === period)}
        onChange={(option) => setPeriod(option.value)}
        searchable={false}
        styles={{ container: (base) => ({ ...base, minWidth: 200 }) }}
      />
    </div>

    <ResponsiveContainer width="100%" height={320}>
      <LineChart data={chartData} margin={{ top: 20, right: 30, left: 0, bottom: 0
    }}>
        <CartesianGrid strokeDasharray="3 3" stroke="#e0e0e0" />
        <XAxis dataKey="date" tick={{ fontSize: 12 }} />
        <YAxis tick={{ fontSize: 12 }} />
        <Tooltip
          formatter={(value) => `${Number(value).toFixed(2)} ₪`}
          labelFormatter={(label) => ` $ {label}`}
        />
        <Line
          type="monotone"
          dataKey="amount"
          stroke="#6f42c1"
          strokeWidth={2}

```

```

        dot={{ r: 2.5, fill: "#6f42c1" }}
        activeDot={{ r: 5 }}
        animationDuration={600}
      />
    </LineChart>
  </ResponsiveContainer>
</div>
);
}

```

CategoryPieChart.jsx:

```

import React, { useState } from "react";
import { useTranslation } from 'react-i18next';
import Select from 'react-select';
import {
  PieChart,
  Pie,
  Cell,
  Tooltip,
  ResponsiveContainer,
  Legend,
} from "recharts";

```

```

const BASE_COLORS = [
  "#4CAF50", "#2196F3", "#FFC107", "#FF5722", "#9C27B0",
  "#00BCD4", "#8BC34A", "#FF9800", "#795548", "#607D8B",
];

```

```

const getRandomColor = () => {
  const letters = '0123456789ABCDEF';
  let color = '#';
  for (let i = 0; i < 6; i++) {
    color += letters[Math.floor(Math.random() * 16)];
  }
  return color;
};

```

```

export default function CategoryPieChart({ transactions, balance }) {
  const { t } = useTranslation();
  const [period, setPeriod] = useState("30d");

```

```

  const now = new Date();
  const periodStart = new Date(now);
  switch (period) {
    case "1d":

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

    periodStart.setDate(now.getDate() - 1);
    break;
case "7d":
    periodStart.setDate(now.getDate() - 7);
    break;
case "14d":
    periodStart.setDate(now.getDate() - 14);
    break;
case "30d":
    periodStart.setDate(now.getDate() - 30);
    break;
case "365d":
    periodStart.setFullYear(now.getFullYear() - 1);
    break;
case "all":
default:
    periodStart.setFullYear(2000);
    break;
}

const periodOptions = [
  { label: t("period.1d"), value: "1d" },
  { label: t("period.7d"), value: "7d" },
  { label: t("period.14d"), value: "14d" },
  { label: t("period.30d"), value: "30d" },
  { label: t("period.365d"), value: "365d" },
  { label: t("period.all"), value: "all" },
];

const expenses = transactions.filter((tx) => {
  return tx.type === "expense" && new Date(tx.date) >= periodStart;
});

const data = expenses.reduce((acc, tx) => {
  const category = tx.category || "Невідомо";
  const value = parseFloat(tx.amount);
  const found = acc.find((item) => item.name === category);

  if (found) {
    found.value += value;
  } else {
    acc.push({ name: category, value });
  }

  return acc;
});

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

```

    }, []);

    const translateCategory = (name) => {
      const key = `category_map.${name}`;
      const translated = t(key);
      return translated !== key ? translated : name;
    };

    const totalExpenses = data.reduce((sum, item) => sum + item.value, 0);

    const mappedData = data.map((item) => ({
      ...item,
      percent: parseFloat(((item.value / totalExpenses) * 100).toFixed(2)),
    })).sort((a, b) => b.percent - a.percent);

    const categoryColorMap = {};
    mappedData.forEach((item, index) => {
      categoryColorMap[item.name] = BASE_COLORS[index] || getRandomColor();
    });

    if (!balance || balance === 0 || data.length === 0) {
      return <p className="text-muted">{t("no_expenses_for_chart")}</p>;
    }

    return (
      <div>
        <h4 style={{ textAlign: "center", marginBottom: "1.5rem", fontWeight: 600 }}>
          {t("report.expense_distribution_title")}
        </h4>

        <div style={{ marginBottom: "1rem", textAlign: "center" }}>
          <label style={{ margin: "1rem", fontSize: "18px" }} htmlFor="period-select">{t("choose_period")}</label>
          <Select
            options={periodOptions}
            value={periodOptions.find((opt) => opt.value === period)}
            onChange={(selectedOption) => setPeriod(selectedOption.value)}
            isSearchable={false}
            styles={{
              container: (base) => ({ ...base, minWidth: 100, display: "inline-block" }),
              control: (base) => ({ ...base, borderRadius: 8 }),
            }}
          />
        </div>
      </div>
    );

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<div style={{ display: "flex", gap: "2rem", alignItems: "center", justifyContent:
"center", width: "100%", height: "350px" }}>
  <div style={{ width: "100%", height: "350px" }}>
    <ResponsiveContainer>
      <PieChart>
        <Pie
          data={mappedData}
          dataKey="percent"
          nameKey="name"
          cx="50%"
          cy="50%"
          outerRadius={130}
          labelLine={false}
          label={() => ""}
        >
          {mappedData.map((entry, index) => (
            <Cell key={index} fill={categoryColorMap[entry.name]} />
          ))}
        </Pie>
        <Tooltip
          formatter={(value, name, props) => [`$ {props.payload.value} %`,
            translateCategory(name)]}
        />
        <Legend formatter={translateCategory} />
      </PieChart>
    </ResponsiveContainer>
  </div>

  <div style={{ width: "40%" }}>
    <ul style={{ listStyle: "none", padding: 0 }}>
      {mappedData.map((entry, index) => (
        <li
          key={index}
          style={{
            marginBottom: "0.5rem",
            color: categoryColorMap[entry.name],
            fontWeight: 500,
          }}
        >
          {translateCategory(entry.name)}: {entry.percent}%
        </li>
      ))}
    </ul>
  </div>

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    </div>
  </div>
);
}

```

OverviewDashboard.jsx

```

import React, { useState, useMemo } from "react";
import { useTranslation } from "react-i18next";

```

```

import debitCard from "../assets/icon/debit-card.png";
import chartDown from "../assets/icon/chart-down.png";
import chartUp from "../assets/icon/arrow-chart.png";
import badge from "../assets/icon/badge.png";
import calendar from "../assets/icon/calendar.png";

```

```

export default function OverviewDashboard({ transactions }) {
  const { t } = useTranslation();

```

```

  const periodOptions = [
    { label: t("period.7d"), value: 7 },
    { label: t("period.14d"), value: 14 },
    { label: t("period.30d"), value: 30 },
    { label: t("period.all"), value: "all" },
  ];

```

```

  const [selectedPeriod, setSelectedPeriod] = useState("all");

```

```

  const filteredTransactions = useMemo(() => {
    if (selectedPeriod === "all") return transactions;
    const daysAgo = new Date();
    daysAgo.setDate(daysAgo.getDate() - selectedPeriod);
    return transactions.filter((tx) => new Date(tx.date) >= daysAgo);
  }, [transactions, selectedPeriod]);

```

```

  if (!filteredTransactions || filteredTransactions.length === 0) {
    return <p className="text-center text-muted">{t("no_transactions")}</p>;
  }

```

```

  const expenses = filteredTransactions.filter(tx => tx.type === "expense");
  const income = filteredTransactions.filter(tx => tx.type === "income");

```

```

  const totalIncome = income.reduce((sum, tx) => sum + parseFloat(tx.amount), 0);
  const totalExpenses = expenses.reduce((sum, tx) => sum +
    parseFloat(tx.amount), 0);

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

```

const parsedBalance = totalIncome - totalExpenses;

const expenseDays = new Set(expenses.map(tx => tx.date));
const avgPerDay = expenseDays.size > 0 ? totalExpenses / expenseDays.size : 0;

const topCategories = Object.entries(
  expenses.reduce((acc, tx) => {
    acc[tx.category] = (acc[tx.category] || 0) + parseFloat(tx.amount);
    return acc;
  }, {})
)
.sort((a, b) => b[1] - a[1])
.slice(0, 3);

const currencyFormatter = new Intl.NumberFormat("uk-UA", {
  style: "currency",
  currency: "UAH",
  minimumFractionDigits: 2,
});

return (
  <div className="mb-4">
    <div className="mb-3 text-end">
      <select
        className="form-select w-auto d-inline-block"
        value={selectedPeriod}
        onChange={(e) => setSelectedPeriod(e.target.value === "all" ? "all" :
Number(e.target.value))}
      >
        {periodOptions.map((option) => (
          <option key={option.value} value={option.value}>
            {option.label}
          </option>
        ))}
      </select>
    </div>

    <div className="row g-3">
      <div className="col-md-4">
        <div className="card shadow-sm">
          <div className="card-body">
            <h5 className="card-title">
              <img
                src={debitCard}
                alt="icon"

```

```

        style={{ width: "30px", height: "30px", marginRight: "8px",
verticalAlign: "middle" }}
      />
      {t("overview.balance")}
    </h5>
    <p className="card-text fs-
4">{currencyFormatter.format(parsedBalance)}</p>
  </div>
</div>
</div>

<div className="col-md-4">
  <div className="card shadow-sm">
    <div className="card-body">
      <h5 className="card-title">
        <img
          src={chartDown}
          alt="icon"
          style={{ width: "30px", height: "30px", marginRight: "8px",
verticalAlign: "middle" }}
        />
        {t("overview.expenses")}
      </h5>
      <p className="card-text fs-4 text-danger">-
{currencyFormatter.format(totalExpenses)}</p>
    </div>
  </div>
</div>

<div className="col-md-4">
  <div className="card shadow-sm">
    <div className="card-body">
      <h5 className="card-title"><img
        src={chartUp}
        alt="icon"
        style={{ width: "30px", height: "30px", marginRight: "8px",
verticalAlign: "middle" }}
      />
      {t("overview.income")}
    </h5>
    <p className="card-text fs-4 text-
success">+{currencyFormatter.format(totalIncome)}</p>
  </div>
</div>
</div>

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

```

<div className="col-12">
  <div className="card shadow-sm">
    <div className="card-body">
      <h5 className="card-title mb-3">
        <img
          src={badge}
          alt="icon"
          style={{ width: "30px", height: "30px", marginRight: "8px",
verticalAlign: "middle" }}
        />
        {t("overview.top_categories")}
      </h5>
      <ul className="mb-0" style={{ fontSize: "1.1rem", lineHeight: "1.8" }}>
        {topCategories.map(([cat, amount]) => (
          <li key={cat}>
            {t(`category_map.${cat}`)} !== `category_map.${cat}` ?
t(`category_map.${cat}`) : cat}: {currencyFormatter.format(amount)}
          </li>
        ))}
      </ul>
    </div>
  </div>
</div>

<div className="col-12">
  <div className="card shadow-sm">
    <div className="card-body">
      <h5 className="card-title">
        <img
          src={calendar}
          alt="icon"
          style={{ width: "30px", height: "30px", marginRight: "8px",
verticalAlign: "middle" }}
        />
        {t("overview.average_daily_expenses")}
      </h5>
      <p className="card-text fs-5">
        {currencyFormatter.format(avgPerDay)} / {t("day")}
      </p>
    </div>
  </div>
</div>
</div>
</div>

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		90

```
);
}
```

ReportGenerator.jsx

```
import React, { useState, useMemo, useRef } from "react";
import { useTranslation } from "react-i18next";
import jsPDF from "jspdf";
import html2canvas from "html2canvas";
import {
  PieChart, Pie, Cell, Tooltip,
  LineChart, Line, XAxis, YAxis, CartesianGrid, ResponsiveContainer, Legend
} from "recharts";
import Select from "react-select";
```

```
import debitCard from "../assets/icon/debit-card.png";
import chartDown from "../assets/icon/chart-down.png";
import chartUp from "../assets/icon/arrow-chart.png";
import badge from "../assets/icon/badge.png";
import calendar from "../assets/icon/calendar.png";
import calendarDate from "../assets/icon/calendar-date.png";
import profileUser from "../assets/icon/profile-user.png";
```

```
const COLORS = ["#8884d8", "#82ca9d", "#ffc658", "#ff8042", "#a4de6c",
"#ff9999", "#66c2ff"];
```

```
export default function ReportGenerator({ transactions, user, balance }) {
  const { t } = useTranslation();
  const [period, setPeriod] = useState("7");
  const [showReport, setShowReport] = useState(false);
  const page1Ref = useRef(null);
  const page2Ref = useRef(null);
```

```
  const periodOptions = [
    { value: "7", label: t("period.7d") },
    { value: "14", label: t("period.14d") },
    { value: "30", label: t("period.30d") },
    { value: "all", label: t("period.all") },
  ];
```

```
  const iconStyle = {
    width: "24px",
    height: "24px",
    marginRight: "10px",
    verticalAlign: "middle",
```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

const filteredTx = useMemo(() => {
  if (!transactions) return [];
  if (period === "all") return transactions;
  const days = parseInt(period, 10);
  const fromDate = new Date(Date.now() - days * 24 * 60 * 60 * 1000);
  return transactions.filter(tx => new Date(tx.date) >= fromDate);
}, [transactions, period]);

const income = filteredTx.filter(tx => tx.type === "income");
const expenses = filteredTx.filter(tx => tx.type === "expense");

const totalIncome = income.reduce((sum, tx) => sum + parseFloat(tx.amount), 0);
const totalExpenses = expenses.reduce((sum, tx) => sum +
parseFloat(tx.amount), 0);
const parsedBalance = typeof balance === "number" ? balance :
parseFloat(balance) || 0;

const avgPerDay = (() => {
  const days = [...new Set(expenses.map(tx => tx.date.split("T")[0]))];
  return days.length > 0 ? totalExpenses / days.length : 0;
})();

const topCategories = useMemo(() => {
  const catMap = {};
  expenses.forEach((tx) => {
    const cat = tx.category;
    catMap[cat] = (catMap[cat] || 0) + parseFloat(tx.amount);
  });
  return Object.entries(catMap)
    .sort((a, b) => b[1] - a[1])
    .slice(0, 3)
    .map(([name, value]) => ({ name, value }));
}, [expenses]);

const allCategoryData = useMemo(() => {
  const map = {};
  expenses.forEach(tx => {
    const cat = tx.category;
    map[cat] = (map[cat] || 0) + parseFloat(tx.amount);
  });
  return Object.entries(map).map(([name, value]) => ({ name, value }));
}, [expenses]);

```

```
const lineChartData = useMemo(() => {
  const dayMap = {};
  filteredTx.forEach(tx => {
    const date = tx.date.split("T")[0];
    if (!dayMap[date]) dayMap[date] = { date, income: 0, expense: 0 };
    dayMap[date][tx.type] += parseFloat(tx.amount);
  });
  return Object.values(dayMap).sort((a, b) => a.date.localeCompare(b.date));
}, [filteredTx]);
```

```
const customStyles = {
  container: (base) => ({
    ...base,
    display: "inline-block",
    width: "auto",
    minWidth: 160,
    maxWidth: "100%",
  }),
  control: (base) => ({
    ...base,
    borderRadius: 8,
    paddingLeft: 4,
    paddingRight: 4,
  }),
  menu: (base) => ({
    ...base,
    width: "auto",
    minWidth: "100%",
  }),
};
```

```
const formatCurrency = (value) => ` ${value.toFixed(2)} ₴`;
```

```
const handleDownloadAfterPreview = async () => {
  const pdf = new jsPDF("p", "mm", "a4");
  const width = pdf.internal.pageSize.getWidth();
```

```
const processPage = async (ref, addNewPage = false) => {
  if (!ref.current) return;
  const canvas = await html2canvas(ref.current);
  const height = (canvas.height * width) / canvas.width;
  const imgData = canvas.toDataURL("image/png");
```

```
if (addNewPage) pdf.addPage();
pdf.drawImage(imgData, "PNG", 0, 0, width, height);
```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

```

};

await processPage(page1Ref);    // перша сторінка
await processPage(page2Ref, true); // друга сторінка

pdf.save("report.pdf");
};

return (
  <div className="container mt-5">
    <div
      className="card shadow-sm p-4"
      style={{
        minWidth: "700px",
        maxWidth: "850px",
        margin: "0 auto",
        borderRadius: "16px",
      }}
    >
      <p className="mb-4 text-center text-muted"
        style={{ fontSize: "1.5rem", fontStyle: "italic", fontWeight: 500 }}>
        {t("report.info_text")}
      </p>
      <div className="d-flex justify-content-center gap-3 flex-wrap">
        <Select
          options={periodOptions}
          value={periodOptions.find(opt => opt.value === period)}
          onChange={(selected) => setPeriod(selected.value)}
          isSearchable={false}
          styles={customStyles}
        />
        <button
          onClick={() => setShowReport(true)}
          className="btn btn-success"
        >
          {t("generate_report")}
        </button>
      </div>
    </div>
    {showReport && (
      <div id="report-content" style={{
        backgroundColor: "#fff",

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

padding: "40px 60px",
margin: "40px auto",
width: "calc(100% - 60px)",
maxWidth: "850px",
fontFamily: "'Segoe UI', Tahoma, Geneva, Verdana, sans-serif",
lineHeight: "1.7",
color: "#222",
borderRadius: "12px",
boxShadow: "0 0 10px rgba(0,0,0,0.05)"
}}
>
<div className="container mt-4">

<div id="report-content" style={{ padding: "40px 60px" }}>
<div ref={page1Ref} id="report-page-1">
  <h3 style={{ fontSize: "1.8rem", fontWeight: "600", marginBottom:
"1.5rem" }}>{t("report.title")}</h3>
  <p>
    <img
      src={calendarDate}
      alt="icon"
      style={iconStyle}
    />
    {t("report.generated_on")}: {new Date().toLocaleDateString()}</p>
  <p>
    <img
      src={profileUser}
      alt="icon"
      style={iconStyle}
    />
    {t("report.user_name")}: {user.first_name} {user.last_name}</p>
  <p>
    <img
      src={debitCard}
      alt="icon"
      style={iconStyle}
    />
    {t("overview.balance")}: {formatCurrency(parsedBalance)}
  </p>
  <p>
    <img
      src={chartDown}
      alt="icon"
      style={iconStyle}
    />

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		95

```

        {t("overview.expenses")}: {totalExpenses.toFixed(2)} ₴
    </p>
    <p>
        <img
            src={chartUp}
            alt="icon"
            style={iconStyle}
        />
        {t("overview.income")}: {totalIncome.toFixed(2)} ₴
    </p>
    <p>
        <img
            src={calendar}
            alt="icon"
            style={iconStyle}
        />
        {t("overview.average_daily_expenses")}: {avgPerDay.toFixed(2)}
    </{t("day")}>
    </p>

    <h5>
        <img
            src={badge}
            alt="icon"
            style={iconStyle}
        />
        {t("overview.top_categories")}
    </h5>
    <ul style={{ fontSize: "1.1rem", lineHeight: "1.8" }}>
        {topCategories.map(cat => (
            <li key={cat.name}>{cat.name}: {cat.value.toFixed(2)} ₴</li>
        ))}
    </ul>

    <h6 className="mt-4" style={{ fontSize: "1.25rem", fontStyle: "italic",
marginTop: "2.5rem", marginBottom: "1.5rem" }}>{t("report.pie_title")}</h6>
    <div style={{ height: 300, marginBottom: 40 }}>
    <ResponsiveContainer>
        <PieChart>
            <Pie
                data={allCategoryData}
                dataKey="value"
                nameKey="name"
                outerRadius={90}
                labelLine={true}

```

```

        label=(({ percent }) => `${(percent * 100).toFixed(1)}%`)
      >
      {allCategoryData.map((entry, index) => (
        <Cell key={`cell-${index}`} fill={COLORS[index %
COLORS.length]} />
      ))}
    </Pie>
    <Tooltip formatter={(val) => `${val.toFixed(2)} ₾`} />
    <Legend />
  </PieChart>
</ResponsiveContainer>
</div>
</div>

<div ref={page2Ref} id="report-page-2">
  <h6 className="mt-5" style={{ fontSize: "1.25rem", fontStyle: "italic",
marginTop: "2.5rem", marginBottom: "1.5rem"}}>{t("report.line_title")}</h6>
  <div style={{ height: 300, marginBottom: 40 }}>
    <ResponsiveContainer>
      <LineChart data={lineChartData}>
        <CartesianGrid strokeDasharray="3 3" />
        <XAxis dataKey="date" />
        <YAxis />
        <Tooltip
          formatter={(value, name) => {
            const translatedName = name === "income" ? t("income") :
t("expense");
            return [value, translatedName];
          }}
        />
        <Legend
          formatter={(value) => {
            return value === "income" ? t("income") : t("expense");
          }}
        />
        <Line
          type="monotone"
          dataKey="income"
          stroke="#82ca9d"
          dot={{ r: 3 }}
          strokeWidth={2}
        />
        <Line
          type="monotone"
          dataKey="expense"

```

```

        stroke="#ff4c4c"
        dot={{ r: 3 }}
        strokeWidth={2}
      />
    </LineChart>
  </ResponsiveContainer>
</div>
</div>

</div>
</div>
</div>
<div className="text-center mt-4">
  <button onClick={handleDownloadAfterPreview} className="btn btn-
success" style={{marginBottom: "1rem"}}>
    {t("download_pdf")}
  </button>
</div>
</>
)}
</div>
));

```

Додаток В

```

App.py:
import re
from fastapi import FastAPI, UploadFile, File
from fastapi.middleware.cors import CORSMiddleware
from fastapi.responses import JSONResponse
import pytesseract
import os
import subprocess
import tempfile
import re
from categorize_smart_router import categorize_smart_router
from categories import categorize_router

app = FastAPI()

app.add_middleware(
    CORSMiddleware,

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

allow_origins=["http://localhost:3000"], # або ["*"] на час розробки
allow_credentials=True,
allow_methods=["*"],
allow_headers=["*"],
)

# Підключаємо маршрут категоризація
app.include_router(categorize_router)
app.include_router(categorize_smart_router, prefix="/api")

# Вказуємо шлях до tesseract
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
tessdata_dir = os.path.abspath(
    os.path.join(BASE_DIR, "..", "client", "public", "tessdata")
)

def is_garbage_line(text):
    if len(text) < 10:
        return True
    suspicious_symbols = r"[\w\sa-яА-ЯіїїєЄґґ\-.]"
    if len(re.findall(suspicious_symbols, text)) > 6:
        return True
    if sum(1 for ch in text if ch in "[]{}()|/\\" ) >= 3:
        return True
    return False

# Функція для обробки покупок

def extract_purchase_lines(text):
    lines = text.splitlines()
    purchases = []
    last_item = None

    for line in lines:
        print(f'Processing line: {line}') # Логування кожного рядка
        line = line.strip()
        if not line:
            continue

        # Використовуємо додаткову логіку для виявлення сміття
        if is_garbage_line(line):
            print(f'Видалено як is_garbage_line: {line}')
            continue

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				99
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# Додаткові евристики
if len(line) < 6 and not re.search(r"\d", line):
    print(f"Видалено як занадто короткий шум: {line}")
    continue

if not re.search(r"[аєіііоуяюаєіоу]", line, re.IGNORECASE):
    print(f"Видалено як рядок без голосних: {line}")
    continue

if len(re.findall(r"[a-zA-Za-яА-Яііє]", line)) < 3 and len(line) > 4:
    print(f"Видалено як нерозбірливий: {line}")
    continue
if len(re.findall(r"^[w\s]", line)) >= 2 and len(re.findall(r"[a-zA-Za-яА-Яііє]", line)) <= 3:
    print(f"Видалено як рядок із символів і малої кількості букв: {line}")
    continue

# Ключові "сміттєві" слова та патерни, які треба прибрати
ignore_patterns = [
    r"штрих[-\s]*код", r"\бкаса\b", r"\бсупермаркет\b", r"\бсільпо\b",
    r"\бат\b",
    r"автора", r"картк[аи]", r"безготівк", r"готівк", r"транз", r"платіжн",
    r"оплата",
    r"чек", r"знижка", r"знихка", r"по картці", r"\бціна зі знижкою\b",
    r"@w+", # email або згадки
    r"^[0-9\s]{6,}$", # рядок лише з цифр
    r"^\d{4,}$", # довгі числа без нічого
    r"^[—=_\s]{2,}$", # лінії: тире, дефіси, крапки
    r"\бкод\b", # слово "код"
    r"\бкоа\b", # кодові аббревіатури типу "КОА АВТ"
    r"\бавт\b", r"%25", # АВТ, як у 'КОА АВТ. 394259'
    r"[Xx]{5,}", # багато X (зашифровані карти,
    ідентифікатори)
    r"[{}[\]\'\"]", r"^[^w\s]{2,}$", r"^[w\s]{0,5}\W{2,}$",
]
# Якщо рядок співпадає з будь-яким з ігнорованих патернів —
пропустити
if any(re.search(pattern, line, re.IGNORECASE) for pattern in
ignore_patterns):
    print(f"Видалено як сміття: {line}")
    continue

# Перевірка на наявність ціни в середині назви товару
match = re.match(r"(.+?)s+(\d{1,3}\.\d{1,2})s*(грн|A)?$", line)
if match:

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		100

```

print(f"Match found: {line}") # Логування знайдених товарів
if last_item:
    last_item['name'] = last_item['name'] + " " + match.group(1)
    last_item['price'] = match.group(2)
    last_item['price_info'] = None # Оновлюємо price_info
else:
    purchases.append({
        'name': match.group(1),
        'price': match.group(2),
        'price_info': None
    })
    continue

# Альтернативне розбиття рядка на два товари, якщо ціна посередині
middle_price_match = re.match(r"(.+?)\s+(\d{1,3}\.\d{1,2})\s+([А-ЯА-
ZІЇЄ].+)", line)
if middle_price_match:
    print(f"Розділено як два товари: {line}")
    purchases.append({
        'name': middle_price_match.group(1).strip(),
        'price': middle_price_match.group(2).replace(".", ","),
        'price_info': None
    })
    purchases.append({
        'name': middle_price_match.group(3).strip(),
        'price': None,
        'price_info': None
    })
    continue

# Перевірка на наявність маси і ціни
mass_price_match =
re.match(r'(\d{1,3}[\.,]?\d{1,2})\s*[Xx]?s*(\d{1,3}[\.,]?\d{1,2})', line)
if mass_price_match:
    print(f"Mass price match: {line}") # Логування маси та ціни
    if last_item:
        mass = mass_price_match.group(1)
        price = mass_price_match.group(2)
        total_price = float(mass) * float(price)
        last_item['price'] = round(total_price, 2)
    continue

# Якщо рядок не містить ціну, зберігаємо назву товару
last_item = {"name": line, "price": None, "price_info": None}
purchases.append(last_item)

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				101
Змн.	Арк.	№ докум.	Підпис	Дата		

```

final_purchases = []
i = 0
while i < len(purchases):
    current = purchases[i]
    next_item = purchases[i + 1] if i + 1 < len(purchases) else None

    # Виправлення "хибної" малої ціни у першому рядку
    if (
        current['price'] is not None and next_item and next_item['price'] is not
None
        and float(current['price']) < 10.0
        and float(next_item['price']) > 20.0
        and len(current['name'].split()) >= 3
    ):
        combined_name = current['name'].strip() + " " + next_item['name'].strip()
        final_purchases.append({
            'name': combined_name,
            'price': next_item['price'],
            'price_info': None
        })
        i += 2
        continue

    # 1. Об'єднання: поточна мала ціна, наступна більша — справжня
    if (
        current['price'] is not None and next_item and next_item['price'] is not
None
        and float(current['price']) < 10.0 and float(next_item['price']) > 25.0
        and len(current['name'].split()) <= 5
        and len(next_item['name'].split()) >= 2
    ):
        combined_name = current['name'].strip() + " " + next_item['name'].strip()
        final_purchases.append({
            'name': combined_name,
            'price': next_item['price'],
            'price_info': None
        })
        i += 2
        continue

    # 2. Об'єднання: ціна лише в другому рядку
    if current['price'] is None and next_item and next_item['price'] is None:
        price_match = re.search(r"(\d{1,3}\.\d{1,2})", next_item['name'])
        if price_match:

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				102
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        combined_name = current['name'] + " " + next_item['name']
        price = price_match.group(1)
        final_purchases.append({
            'name': combined_name.strip(),
            'price': price,
            'price_info': None
        })
        i += 2
        continue

    final_purchases.append(current)
    i += 1

# 4. Inline ціна в name, якщо price досі None
if current['price'] is None:
    price_match = re.search(r"(\d{1,3}\.\d{1,2})", current['name'])
    if price_match:
        price = price_match.group(1)
        name_cleaned = re.sub(r"\s*\d{1,3}\.\d{1,2}(?=\s|$)", "",
current['name']).strip()
        final_purchases.append({
            'name': name_cleaned,
            'price': price,
            'price_info': None
        })
        i += 1
        continue

return final_purchases

@app.post("/parse-check/")
async def parse_check(file: UploadFile = File(...)):
    try:
        contents = await file.read()
        print("Файл прочитано, байтів:", len(contents))

        # Тимчасовий файл для збереження вмісту
        with tempfile.NamedTemporaryFile(suffix=".png", delete=False) as temp:
            temp.write(contents)
            temp.flush()
            temp_path = temp.name

        # Використовуємо tesseract для розпізнавання тексту з зображення
        output_txt = temp_path + "_out"

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		103

```

try:
    result = subprocess.run(
        [
            pytesseract.pytesseract.tesseract_cmd,
            temp_path,
            output_txt,
            "-l", "ukr+eng",
            "--tessdata-dir", tessdata_dir,
            "--oem", "3",
            "--psm", "4"
        ],
        stderr=subprocess.PIPE
    )

    if result.returncode != 0:
        error_message = result.stderr.decode("utf-8")
        print("Tesseract stderr:", error_message)
        return {"error": error_message}

    with open(output_txt + ".txt", "r", encoding="utf-8", errors="ignore") as f:
        raw_text = f.read()

    # Обробка витягнутого тексту
    structured_lines = extract_purchase_lines(raw_text)

    # Повертаємо відповідь у вигляді JSON
    return {"items": structured_lines}

finally:
    os.remove(temp_path)
    if os.path.exists(output_txt + ".txt"):
        os.remove(output_txt + ".txt")

except Exception as e:
    import traceback
    traceback.print_exc()
    return JSONResponse(status_code=500, content={"error": str(e)})

@app.get("/")
def read_root():
    return {"message": "Сервер працює!"}

Parser-service/Category.app:
from fastapi import APIRouter
from pydantic import BaseModel

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		104

```

import openai
import os
from openai import OpenAI
from dotenv import load_dotenv
from pathlib import Path
import re
from categorize_smart_router import categorize_smart_router

load_dotenv(dotenv_path=Path(__file__).resolve().parent.parent / "server" /
".env")

client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

categorize_router = APIRouter()
categorize_router.include_router(categorize_smart_router)

class ItemRequest(BaseModel):
    name: str

## ОЧИЩЕННЯ ПЕРЕД ЗЧИТУВАННЯ ЧАТУ
def clean_ocr_text(text: str) -> str:
    text = text.lower()
    text = re.sub(r'^[а-яієґа-з0-9\s]', "", text) # прибрати спецсимволи
    text = re.sub(r'\b[йн]{1}/?[а-за-я]*\b', "", text) # прибрати "й", "н/..." залишки
    text = re.sub(r'\s+', ' ', text) # замінити багато пробілів на один
    return text.strip()

@categorize_router.post("/ai-category")
async def categorize_item(req: ItemRequest):
    try:
        prompt = f'До якої з категорій належить товар '{req.name}''?\nКатегорії:
Їжа, Медицина, Транспорт, Комуналка, Інше.\nВідповідь тільки категорією:'
        response = client.chat.completions.create(
            model="gpt-3.5-turbo",
            messages=[{"role": "user", "content": prompt}],
            temperature=0
        )
        category = response.choices[0].message.content.strip()
        return {"category": category}
    except openai.APIError as e:
        logger.error(" GPT API error: %s", e)
        return JSONResponse(
            status_code=status.HTTP_503_SERVICE_UNAVAILABLE,
            content={"error": "GPT API недоступний"}
        )

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		105

```
except Exception as e:
    logger.error(" Загальна помилка GPT: %s", e)
    return {"category": "Інше"}
```

Parser-service/Categorize_smart_route.app:

```
from fastapi import APIRouter, HTTPException
from pydantic import BaseModel
from openai import OpenAI
from dotenv import load_dotenv
from pathlib import Path
import os, re
import requests
```

```
load_dotenv(dotenv_path=Path(__file__).resolve().parent.parent / "server" /
".env")
```

```
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))
categorize_smart_router = APIRouter()
```

```
SERVER_BASE_URL = "http://localhost:5000"
```

```
class ItemRequest(BaseModel):
    name: str
```

```
BAD_KEYWORDS = {
    "продукт", "товар", "стручок", "карамельна", "вологі", "пресовані",
    "солодкий",
    "та", "з", "на", "г", "кг", "шт", "грн", "pack", "упаковка", "light", "saleks"
}
```

```
NOUN_LIKE_SUFFIXES = ("о", "а", "я", "е", "ик", "ість", "тво", "ія", "ина",
"ина", "мак", "ель")
```

```
def is_likely_noun(word):
    return word.endswith(NOUN_LIKE_SUFFIXES)
```

```
def is_valid_keyword(word, position_in_line=0):
    return (
        word
        and word.isalpha()
        and len(word) > 2
        and word.lower() not in BAD_KEYWORDS
        and (
            is_likely_noun(word.lower()) or position_in_line == 0
        )
        and not re.match(r"^[a-zA-Z]{3,}$", word)
```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				106
Змн.	Арк.	№ докум.	Підпис	Дата		

```

)

def extract_keywords(text: str) -> list:
    prompt = (
        f"У цьому рядку: '{text}' — знайди одне слово, яке чітко вказує, що це за товар "
        f"(наприклад: 'молоко', 'батончик', 'йогурт', 'печиво', 'вафлі', 'вайвлі'). "
        f"Ігноруй слова на кшталт 'пресовані', 'карамельна', 'стручок', 'вологі', бренди, вагу, прикметники та числові дані. "
        f"Не використовуй загальні слова на кшталт 'продукт', 'товар'. "
        f"Відповідь: лише одне слово — без опису і зайвих символів."
    )

    try:
        response = client.chat.completions.create(
            model="gpt-3.5-turbo",
            messages=[{"role": "user", "content": prompt}],
            temperature=0
        )
        content = response.choices[0].message.content.strip()
        keywords = [kw.strip() for kw in content.split(",") if kw.strip()]
        return [kw for kw in keywords if is_valid_keyword(kw)]
    except Exception as e:
        print("!! GPT error in extract_keywords:", e)
        return []

@categoryize_smart_router.post("/api/ai-category-smart")
async def categorize_item_smart(req: ItemRequest):
    try:
        original_name = req.name
        keywords = extract_keywords(original_name)

        results = []

        for keyword in keywords:
            cleaned = keyword.lower()

            # 1. Перевірка кешу
            cache_check = requests.get(
                f'{SERVER_BASE_URL}/api/check-category',
                params={"cleaned_name": cleaned}
            )

            if cache_check.status_code == 200:
                category = cache_check.json().get("category")

```

		Прокопчук О.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		107

```

        if category:
            results.append({"name": cleaned, "category": category})
            continue

# 2. GPT-категоризація
prompt = f'До якої з категорій належить товар '{cleaned}'?\nКатегорії:
Їжа, Медицина, Транспорт, Комуналка, Інше.\nВідповідь тільки категорією:"
try:
    gpt_response = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": prompt}],
        temperature=0
    )
    category = gpt_response.choices[0].message.content.strip()
except Exception as e:
    print("!! GPT error in categorize:", e)
    category = "Інше"

# 3. Збереження в кеш
save_res = requests.post(f'{SERVER_BASE_URL}/api/save-category",
json={
    "cleaned_name": cleaned,
    "category": category
})

if save_res.status_code not in [200, 201]:
    print(f'!! Не вдалося зберегти категорію: {cleaned} - {category}')

results.append({"name": cleaned, "category": category})

return {"results": results}

except Exception as e:
    print("!! ERROR в categorize_item_smart:", e)
    raise HTTPException(status_code=500, detail="Помилка категоризації")

```