МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ
ФАКУЛЬТЕТ №2
Кафедра інформаційних технологій

РОЗРОБЛЕННЯ WEB-ДОДАТКУ ДЛЯ КЕРУВАННЯ ФІНАНСАМИ ПРАВООХОРОННИХ ПІДРОЗДІЛІВ

Кваліфікаційна робота
здобувача вищої освіти
4 курсу денної форми навчання
Василини НАЦЯК

Науковий керівник:
доцент, канд. фіз.-мат. наук
Тетяна МАГЕРОВСЬКА

Рецензент:
професор, доктор фіз.-мат. наук
Ігор ДЕМКІВ

Кваліфікаційна робота допущена до захисту
«___» _____ 2026 р., протокол № ____

Завідувач кафедри інформаційних технологій
_____ Олег ЗАЧЕК
(підпис)

Львів
2026

СПИСОК УМОВНИХ СКОРОЧЕНЬ

API	Application Programming Interface (інтерфейс програмування застосунків)
DOM	Document Object Model (модель об'єкта документа)
SQL	Structured query language (мова структурованих запитів)
UI	User Interface (інтерфейс користувача)
БД	База даних
ІС	Інформаційна система
МК	Мікроконтролер
ПЗ	Програмне забезпечення
СУБД	Система управління базами даних

АНОТАЦІЯ

НАЦЯК В. Розроблення Web-додатку для керування фінансами правоохоронних підрозділів. – Рукопис.

Дослідження на здобуття освітнього ступеня «бакалавр» за спеціальністю F6 – «інформаційні системи та технології». – Львівський державний університет внутрішніх справ, МВС України. Львів. 2026.

У кваліфікаційній роботі спроєктовано та розроблено Web-додаток для моніторингу фінансової діяльності правоохоронних підрозділів, який передбачає поєднання обліку, контролю й аналізу їх фінансової діяльності. З цією метою у роботі, на основі сучасних досягнень у галузі проєктування інформаційних систем, обрано інструментарій для проєктування Web-додатку, визначено технології та принципи його реалізації. За результатами апробації спроєктованого Web-додатку встановлено, що його функціонал забезпечує зручне керування фінансовою діяльністю правоохоронного підрозділу, зокрема завдяки застосуванню інструментів категоризації й аналізу фінансових операцій. Окрім цього, під час апробації Web-додатку окреслено подальші перспективи його удосконалення шляхом розширення можливостей підсистем аналітики та прогнозування.

Ключові слова: Web-додаток, фінансовий моніторинг, керування фінансами, база даних, СУБД Mongo DB, React, Node.js.

ABSTRACT

NATSYAK V. Development of a Web-application for managing the finances of law enforcement units. – Manuscript. Research for the degree of Bachelor in specialty F6 – “Information Systems and Technologies”. – Lviv State University of Internal Affairs, Ministry of Internal Affairs of Ukraine. Lviv. 2026.

In the qualification work, a Web application for monitoring the financial activities of law enforcement units was designed and developed, which involves a combination of accounting, control and analysis of their financial activities. To this

end, in the work, based on modern achievements in the field of information systems design, a toolkit for designing a Web application was selected, technologies and principles of its implementation were determined. According to the results of testing the designed Web application, it was established that its functionality provides convenient management of the financial activities of a law enforcement unit, in particular through the use of categorization tools and analysis of financial transactions. In addition, during the testing of the Web application, further prospects for its improvement were outlined by expanding the capabilities of the analytics and forecasting subsystems.

Keywords: Web application, financial monitoring, financial management, database, Mongo DB DBMS.

ЗМІСТ

ВСТУП

В умовах сьогодення фінансовий менеджмент займає важливе місце з огляду досягнення фінансової стабільності та вирішення суспільних питань. Здатність моніторити дохідну та витратну частини бюджету, його балансувати, проводити аналізування фінансових потоків сприяє формуванню економічної грамотності, прийняттю обґрунтованих рішень. Значне сприяння в управлінні фінансами може забезпечити застосування відповідних програмних продуктів, наприклад, Web-додатку для обліку фінансових операцій. Такий застосунок здатен стати зручним інструментом, що забезпечить зручну діяльність у напрямі контролю за фінансами. Необхідність застосування такого програмного додатку обумовлюється широким впровадженням децентралізаційних процесів у державі та цифровізацією діяльності нинішнього суспільства. Потребу у використанні такого додатку можуть мати не лише окремі особи, а й структури, підрозділи приватних і державних органів. Не є виключенням і підрозділи Національної поліції. Так, наприклад, його застосування в окремих підрозділах Національної поліції забезпечить дієвий та інтуїтивно зрозумілий спосіб контролю й аналізування їх фінансових операцій.

Володіючи комбінованим набором функцій (облік доходів, облік витрат, планування витрат, візуалізація фінансових відомостей) цей програмний продукт покликаний надати допомогу окремим особам, підрозділам у здійснені контролю над виділеними коштами та забезпечити досягнення фінансової мети.

Із викладеного випливає **актуальність теми роботи** – зростаюча потреба державних органів і структур (у тому числі правоохоронних органів) в одержанні зручного та функціонального програмного засобу для керування та контролю за фінансовими ресурсами.

Розроблення описаного Web-додатку передбачає поєднання обліку, контролю, аналізування фінансових витрат і є необхідним рішенням, яке сприятиме підвищенню ефективності діяльності у фінансової сфері як окремих працівників (користувачів), так і окремих груп (підрозділів). Таке рішення дозволить забезпечити, зрештою, фінансову стабільність.

Аналіз останніх досліджень і публікацій. Дослідженню питань проєктування Web-додатків для моніторингу фінансових операцій присвячено чимало наукових робіт, підготовлених українськими та закордонними ученими. Прикладами можуть слугувати праці: О. Ткаченка та М. Рачкова [1], О. П’ятикопа та ін. [2], К. Гояла та ін. [3], С. Хелмі та ін. [4], А. Пахса [5], Т. Стефанова та С. Варбанової [6] тощо. Однак, розвиток сучасних технологій, пов’язаних із проєктуванням Web-додатків, вимагає проведення подальших удосконалень з огляду функціоналу, дизайну, програмних рішень тощо. Тому питання проєктування Web-додатків для моніторингу фінансових операцій залишається надалі актуальним.

Метою роботи є розроблення ефективного Web-додатку, що надасть можливість правоохоронним підрозділам здійснювати керування власними фінансовими потоками.

Для досягнення поставленої мети слід виконати наступні **завдання**:

- провести аналіз потреб у сфері керування фінансами підрозділу;
- спроєктувати інтерфейс, що забезпечить швидку та зручну реєстрацію фінансових операцій;
- забезпечити реалізацію функціоналу обліку фінансових операцій з поділом на категорії;
- розробити інструмент для формування бюджету і застосування обмежень за категоріями витрат;
- забезпечити можливість інтеграції з банками з метою автоматизації імпорту транзакцій;

- реалізувати можливість фінансового прогнозування на основі історії трансакцій.

Об'єктом дослідження у роботі є проєктування Web-додатку для моніторингу фінансових потоків підрозділу із можливістю їх аналізу.

Предмет дослідження – засоби проєктування Web-додатку для моніторингу фінансових операцій на основі СУБД Mongo DB, з використанням інструментів «Express.js», «React», «Node.js».

Методи досліджень. Під час проєктування Web-додатку застосовано ряд загальноприйнятих методів дослідження, які дозволили у повній мірі досягти мету проєкту та вирішення завдань розробки. Зокрема: бібліометричний метод дозволив проаналізувати наявні наукові праці у даній сфері діяльності; використовуючи різні філософські методи пізнання (діалектичний, синергетичний, аксіологічний тощо) проведено всебічне аналізування предмету дослідження; аналіз, синтез, індуктивний та дедуктивний методи забезпечили можливість досягнення вище зазначених завдань.

Архітектура Web-додатку забезпечує поділ клієнтської та серверної складової, що у майбутньому надасть можливість розширювати його функціональні можливості та забезпечити інтегрування розроблювальних сервісів без суттєвого змінювання існуючого коду.

Продуктивність додатку досягається внаслідок впровадження сучасних методик оптимізування обчислювальних процесів, зокрема, застосування серверного кешування (зменшує навантаження на БД), клієнтського кешування (пришвидшує завантаження сторінок).

Структура роботи. Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел, додатків. Обсяг основного тексту роботи складає 53 сторінки, 13 рисунків, 2 таблиці, 7 додатків і 23 бібліографічних джерела. Загальний обсяг роботи – 73 сторінки.

Розділ 1.
ТЕОРЕТИЧНІ ЗАСАДИ ОБРАННЯ ІНСТРУМЕНТАРІЮ ДЛЯ СТВОРЕННЯ WEB-ДОДАТКУ

1.1. Огляд предметної області

Використання Web-додатку для фінансового моніторингу це інноваційний засіб, призначення якого полягає у комплексному керуванні фінансовими потоками. У даній роботі здійснюється розроблення програмного продукту, що має забезпечити не тільки контроль за фінансовими операціями, а й їх аналізування з ціллю забезпечення формування оптимальних фінансових планів. Основними аспектами Web-додатку, який належить розробити, є:

- створення функціоналу для фіксування, класифікації і контролю транзакцій;
- забезпечення аналітичного функціоналу, включно із візуалізацією фінансових операцій у вигляді діаграм;
- створення функціоналу для автоматизованого і ручного опрацювання дохідної частини та витрат;
- проєктування зручного та зрозумілого інтерфейсу користувача.

Згідно основних аспектів Web-додатку, який належить розробити, метою роботи є створення програмного продукту із зручним, зрозумілим та ефективним функціоналом для моніторингу фінансових операцій підрозділу правоохоронного органу.

1.2. Аналіз функціоналу популярних Web-додатків для керування фінансовими операціями

Нині на ринку можна знайти велику кількість програмних продуктів, основна мета яких облік й аналіз фінансових операцій. До популярних

рішень належать: Buddy (https://buddy.download/), Goodbudget (https://goodbudget.com), Money Manager (https://moneymanager.com.ua), Money Lover (https://web.moneylover.me), Monobudget (https://monobudget.com.ua), Saldo (https://saldoapps.com), Spendee (https://www.spendee.com), Wallet (https://wallet.google/intl/uk_ua/). Дані програмні продукти володіють значним функціоналом у напрямі фінансового менеджменту. Загалом цей функціонал направлений на покращення моніторингу над фінансами (особистими чи підрозділу). Практично головною особливістю названих Web-додатків є моніторинг дохідної і витратної частини бюджету. Тобто, користувачі здатні контролювати фінансові дії за певними категоріями. Окремі програмні продукти мають можливість синхронізації із банківськими рахунками.

У деякі застосунки закладена функція планування, тобто вони дають можливість формувати бюджет, лімітувати витрати за відповідними категоріями, моніторити трансакції. В окремих випадках у Web-додатках застосовано засіб «конверт», що забезпечує наочніший фінансовий розподіл. Сутність цього методу полягає у тому, що наявний кошторис розподіляється за певною категорією витрат. Це схоже на розподіл фінансів в окремі конверти, які мають різне призначення.

Окрім цього, деякі програмні застосунки володіють функціоналом для проведення інвестиційного аналізування, хоч для такої мети застосовують, як правило, спеціалізовані програмні продукти. Значна кількість із перелічених вище програмних продуктів інтегруються із банківським рахунком (кредитною карткою). Такий підхід забезпечує автоматизацію процесів оновлення і, відповідно, практичність у використанні. Також серед перелічених вище рішень можна віднайти застосування функцій прогнозування, які на підставі наявних інформаційних відомостей надають можливість спрогнозувати фінансову ситуацію у найближчому майбутньому. Слід для об’єктивності зазначити, що такий інструмент нині ще не досягнув відповідної популярності.

1.3. Порівняння основних компонентів популярних Web-додатків керування фінансовими операціями

З метою усвідомлення функціоналу різних існуючих популярних Web-додатків для проведення моніторингу фінансових операцій необхідно здійснити їх аналіз методом порівняння. З цією метою доцільно описати можливості кожного програмного продукту у вигляді таблиці (див. табл. 1.1).

Аналіз даних у таблиці дозволяє констатувати наступні загальні особливості: загалом, усі розглянуті застосунки передбачають можливості моніторингу фінансових операцій, містять інструментарій проведення бюджетування та підготовку фінансової звітності. Поряд з цим існують також і відмінності, особливо щодо наявності такого функціоналу, як автоматичне інтегрування із банківською системою, наявність інструментарію щодо аналізу інвестиційної привабливості.

Перевагою автоматичного інтегрування із банківською системою є практичність використання такого застосунку та економія часу. Однак, у цьому випадку на перше місце може виходити проблема щодо забезпечення безпекових питань. Уведення даних у ручному режимі дозволяє забезпечити кращий контроль, однак створює певні незручності у використанні такого застосунку. Застосування різних підходів до бюджетування (за категоріями, за цілями, за методом «конвертів») сприяють задоволенню різноманітних потреб користувачів.

Аналіз таблиці 1.1. показує, що під час проєктування Web-додатків для моніторингу фінансових потоків розробники здійснюють спроби надати користувачам найзручніший і максимально функціональний інструмент. Проаналізовані програмні продукти містять у собі певні можливості моніторингу надходжень і витрат, володіють певним функціоналом щодо аналітики, інтегрування із банківською системою та відображенням інформації.

Таблиця 1.1. Порівняльний аналіз функціоналу Web-додатків керування фінансами

Застосунок	**Облік фінансів**							**Аналітика**					**Банківська інтеграція**			**Платформи**			**Вартість**		
	Автоматичний облік	Ручний облік	Базовий облік	Спільний облік	Цільове бюджетування	Бюджетування за категоріями	Бюджетування за конвертами	Детальні звіти	Базові звіти	Статистика	Візуалізація	Аналіз витрат, планування	Так	Ні	Часткова (імпорт виписок)	Web	iOS	Android	Безкоштовно	Підписка	Платна
Buddy				■					■				■				■		■		■
Goodbudget		■					■					■			■	■	■	■	■	■	
Money Lover	■					■		■			■		■			■	■	■	■	■	
Money Manager			■			■		■		■			■				■	■	■		
Monefy		■				■			■					■			■	■	■		
Saldo	■								■				■				■	■		■	
Spendee	■					■					■		■				■	■	■	■	
Wallet	■				■			■					■				■	■	■		

Констатуємо, що усі розглянуті Web-додатки містять сильні аспекти, які здатні задовольнити різних користувачів, наприклад, таких, що віддають перевагу ручному моніторингу, і таких, які схильні до автоматизації й аналітики. Загальний же тренд визначається поступовим переходом до персоналізованіших і складніших рішень, які враховують економічні питанні підрозділу та всезростаючі вимоги до покращеного, зручного у використанні інтерфейсу.

1.4. Вибір інструментарію для проєктування Web-додатку для керування фінансами

Розроблення Web-додатків для керування фінансовими потоками вимагає використання сучасних підходів, за допомогою яких можна досягти ефективності їх роботи, забезпечити захист даних, надати гнучкість для подальшого розвитку та, зрештою, забезпечити комфортним інструментом користувачів.

З метою створення серверної логіки використаємо платформу «Node.js», яка дозволяє проєктувати швидкісні й масштабні Web-сервери.

Для розроблення клієнтської частини проєкту застосуємо мову програмування «Java Script» із відповідними бібліотеками, що дозволить забезпечити проєктування динамічного користувацького інтерфейсу.

Функціональні можливості для прогнозування реалізуємо з використанням технології «TensorFlow.js». «TensorFlow.js» – це потужна бібліотека машинного навчання. Вона застосовується під час роботи безпосередньо у браузері.

Для зберігання інформації, як на нашу думку, слід використати Mongo DB. Це забезпечить зручне керування неструктурованими даними та зручність у масштабуванні.

Інтегроване застосування названого вище інструментарію сприятиме створенню сучасного, надійного і інтелектуального Web-додатку із значним функціоналом.

Визначившись із інструментарієм розроблення нашого Web-додатку для проведення моніторингу фінансів, розглянемо детальніше функціонал згаданих вище засобів.

Платформа «Node.js». «Node.js» являє собою середовище програмування на мові Java Script, яке дозволяє проєктувальникам виконувати код Java Script безпосередньо на сервері. «Node.js» можна використовувати разом із фреймворком «Express.js», який є зручним та потужним інструментом для створення Web-додатку у названому середовищі. До основних переваг використання такої комбінації засобів слід віднести:

- застосування однієї мови програмування для розроблення Web-додатку;
- висока ефективність проєктування, що визначається забезпеченням спрощеного обміну між командою розробників;
- забезпечення високошвидкісного опрацювання запитів, що є дуже важливим аспектом під час роботи Web-додатків, з яким працює велика кількість користувачів одночасно;
- доступ до значної кількості бібліотек та інструментарію, що забезпечує суттєве підвищення ефективності процесу розроблення Web-застосунку, а також сприяє розширенню його функціональних спроможностей;
- наявність фреймворку забезпечує відповідний інструментарій для пришвидшеного проєктування Web-додатку, надаючи програмістам свободу під час обрання архітектурного рішення.

Поряд з цим, платформа «Node.js» має також і певні потенційні недоліки, зокрема, оброблення інтенсивного потоку операцій процесором здатне привести до блокування головного потоку операцій. При цьому слід зауважити, що нині є способи та методологія для вирішення цієї обмежувальної проблеми.

Розглянута платформа є найкращим (на нашу думку) рішенням для проєктування серверного блоку Web-додатку керування фінансами, яка здатна забезпечити необхідну продуктивність і зручність проєкту, зокрема через використання Java Script на усіх рівнях програмування.

Мова програмування Java Script. Java Script є популярною та сучасною мовою для створення Web-додатку. Вона дозволяє забезпечувати інтерактивність і динамічність під час наповнення Web-додатків. Під час створення інтерфейсу користувача можна скористатися бібліотекою «React», яка здатна використовувати компонентно-орієнтовану архітектуру з метою ефективного оновлення. Компонентно-орієнтована архітектура забезпечує повторне використання програмного коду та покращує процедуру керування інтерфейсом, що є позитивним аспектом під час проєктування фінансових Web-додатків. Можливості аналізування фінансів підрозділу чи певного колективу забезпечуватиметься за допомогою різних розділів, а структура бібліотеки сприятиме організації та підтримці кодової основи.

До основних переваг використання мови програмування Java Script разом із бібліотекою «React» можна віднести наступне:

- можливість проєктування ефективного інтерфейсу, що забезпечує необхідний рівень взаємозв'язку з користувачем;
- фреймування інтерфейсу на окремі компоненти, що надає можливість застосування повторно програмного коду, сприяє розробленню Web-додатку, забезпечує підтримання складних структур;
- висока продуктивність роботи Web-додатку завдяки оптимізуванню оновлень інтерфейсу;
- велика кількість доступних бібліотек та інструментарію дозволяє забезпечити зручність процесу проєктування Web-додатку.

Разом із перевагами слід зазначити і певні недоліки обраного інструменту, наприклад, через складність проєкту, кінцевий продукт може

бути доволі великим, що впливатиме на тривалість першопочаткового завантаження.

Бібліотека «TensorFlow.js». «TensorFlow.js» обрано нами для проєктування у Web-додатку функціональної можливості прогнозування. Ця бібліотека, розроблена Google, має відкритий вихідний код, що дозволяє програмістам перевіряти модель машинного навчання у браузері. Таким чином, вона забезпечує інструментарієм для розгортання такої моделі безпосередньо у Web-додатку. Впровадження можливості штучного інтелекту у Web-додаток для контролю фінансових потоків дозволяє розширити його функціональні можливості щодо прогнозування бюджету. Для цього Web-додаток використовуватиме дані щодо фінансової історії, на підставі якої виявлятимуться закономірності й тенденції, які надаватимуть користувачу певні інтелектуальні пропозиції. Слід зазначити, що для ефективного застосування такої методики для проєктуваня й навчання моделі прогнозу є необхідність у наявності великих обсягів достовірних даних.

Mongo DB. Mongo DB – це потужна документальна база даних SQL, яка забезпечує зберігання відомостей. Вона популярна завдяки своїй масштабованості, високій продуктивності й простоті у використанні. Гнучкість даної БД забезпечує легку адаптацію до можливих змін у відомостях, що може стати корисною особливістю у Web-додатках управління фінансами, у якому можна буде здійснювати моніторинг різних банківських операцій чи розглядуваних вище фінансових категорій упродовж певного часу [7]. На противагу реляційним БД, що мають жорстко визначені зв'язки, така база даних забезпечує певні варіації без застосування необхідності складного мігрування чи змін у структурі. Це сприяє її використанню як зручного інструмента для ефективного розроблення та масштабування функціональних можливостей залежно від потреб користувачів.

Розділ 2.
ОБРАННЯ ТЕХНОЛОГІЙ І ПРИНЦИПІВ РОЗРОБЛЕННЯ WEB-ДОДАТКУ

2.1. Визначення функціональних можливостей Web-додатку для керування фінансами

Web-додаток для моніторингу та управління фінансами підрозділу складається із клієнтської та серверної частин. Також містить базу даних, яка направлена на реалізацію певних можливостей для ефективності подальшого фінансового прогнозування. Головно, основні можливості Web-додатку передбачають:

– реєстрацію та авторизацію користувачів з урахуванням вимог безпеки та конфіденційності;

– ведення обліку доходів і витрат із можливістю їх редагування та розподілу за категоріями;

– формування персональних бюджетів, встановлення фінансових цілей та відстеження їх досягнення;

– перегляд узагальненої фінансової інформації за допомогою інтерактивної аналітичної панелі;

– отримання прогнозів щодо майбутнього фінансового стану на основі попередньої активності користувача;

– автоматичне оновлення інформації про транзакції для зменшення ручної роботи користувача;

– адаптивний інтерфейс, що забезпечує комфортну взаємодію з платформою на різних пристроях.

На противагу розглянутим аналогам, розроблюваний Web-додаток має поєднувати у собі сучасні досягнення у галузі інформаційних технологій

необхідну функціональність, масштабованість тощо. Сутність розроблюваного Web-додатку полягає у інтегруванні можливостей прогнозу на основі моніторингу фінансових операцій, що забезпечить не лише відображення фінансового стану, а й на економічну спроможність кожного окремого підрозділу.

Також слід реалізувати підсистему збереження історії проведених фінансових операцій за певний визначений період, що дозволяє здійснювати процеси аналізування динаміки фінансових операцій і їх прогнозування на перспективу. Такі функції дозволяють Web-додатку стати не лише засобом моніторингу фінансових операцій, а й програмним засобом для обґрунтованих економічних рішень.

2.2. Архітектурне рішення для проєктування Web-додатку

Архітектурно Web-додаток для моніторингу та керування фінансами підрозділу проєктується на основі класичної трирівневої моделі. Цим досягається чітке розмежування між різними блоками інформаційної системи, а також забезпечується її масштабованість та підтримка. Кожний рівень забезпечує виконання власної специфічної ролі у досягненні функціоналу [8].

Клієнтська частина розроблювального додатку є основною складовою Web-додатку, яка, головно, направлена на забезпечення безпосередньої взаємодії із користувачами. Ця частина додатку розробляється за допомогою бібліотеки «React». Це забезпечує можливість створити швидкодіючий інтерфейс, включаючи здатність швидкого оновлення сторінки без потреби повного перезавантаження [9]. Це значно сприяє покращенню користувацькому досвіду та забезпечує підвищення функціоналу під час використання Web-додатку.

Для керування Web-додатком застосовується «Redux» (JavaScript-бібліотека, направлена на спрощення керування станом Web-додатку), що забезпечить централізоване зберігання відомостей і сприяє ефективному

координуванню взаємодій між компонентами інформаційної системи. Такий підхід забезпечує спрощення опрацювання існуючих сценаріїв активності користувачів, передбачає дії Web-додатку та значно поліпшує його підтримування й масштабування.

Функціонал клієнтської складової Web-додатку містить ряд важливих розділів. До них входять:

- сторінка реєстрації й авторизації користувача;
- дашборд з відображенням фінансових операцій;
- інтерфейси для моніторингу надходжень та витрат;
- налагодження профілю користувача;
- перегляд результатів аналізу й прогнозування.

Вказані особливості дозволяють забезпечити активну взаємодію користувача із інформаційною системою, а також сприяють ефективності управлінню фінансовими потоками. Завдячуючи зручному інтерфейсу й графічному відображенню відомостей, користувач Web-додатку здатен миттєво зорієнтуватися у фінансових операціях підрозділу та, за потреби, прийняти відповідні рішення.

Клієнтська частина Web-додатку забезпечує обмін інформацією із серверною частиною. Це сприяє надсиланню запитів для одержання оновлень та зберігання даних. Отримані дані опрацьовуються та візуалізуються у зручному вигляді.

Важливим фактором є створення адаптивного дизайну. Це забезпечує коректність під час візуалізації графічного інтерфейсу на різних дивайсах.

Серверна частина забезпечує виконання ролі центрального логічного вузла Web-додатку. Вона опрацьовує запити, які поступають від клієнтської частини, та реалізує основну бізнес-логіку Web-додатку. Серверна частина спроєктована на основі «Node.js». Основним фреймворком для створення API застосовано «Express.js», завдяки якому спрощується маршрутизація й опрацювання HTTP-запитів.

Програмна частина з боку серверу досягає критично важливого функціоналу, головні серед яких:

- автентифікація та авторизація;
- опрацювання фінансових потоків;
- управління бюджетом;
- збирання й агрегація відомостей;
- реалізування функції прогнозування.

Окрім названого, сервер здатен підтримувати можливість автоматизованого імпортування фінансових операцій із зовнішніх джерел. Це забезпечує зменшення навантаження на особу, яка управляє фінансами, та збільшує актуальність фінансових відомостей.

У роботі особливу увагу акцентовано на захисті даних. Зокрема, використані відомі сучасні способи захисту персональних даних та фінансових операцій, у тому числі із використанням токенів та хешуванням паролів. Серверна частина під час роботи є надійним посередником між інтерфейсом користувача та БД. При цьому забезпечується цілісність, безперервність і логіка всіх операцій, що відбуваються в інформаційній системі.

Для забезпечення надійності та гнучкості збереження фінансових відомостей у Web-додатку застосовується документальна БД Mongo DB. У цій базі зберігається головна інформація щодо:

- облікових записів користувачів;
- відомості про фінансові операції;
- налагодження профіля користувача;
- проведені транзакції.

Завдяки значній продуктивності згадана документальна БД може забезпечити ефективне опрацювання значних обсягів даних в режимі онлайн, при цьому надаючи швидкий і стабільний доступ до них [10].

2.3. Проєктування бази даних

Проєктуючи БД для Web-додатку нами враховувалась специфіка предметної галузі та потреба у забезпеченні гнучкості, масштабованості й ефективності моделі збереження інформації. Архітектурно БД спроєктована таким способом, щоб підтримати динаміку структури даних, забезпечити миттєвий доступ до вірогідних відомостей та залишити можливості для наступного розширення функціоналу не провадячи суттєві зміни у структурі сховища. Концептуально організацію відомостей орієнтовано на простоту опрацювання, логічну цілісність та здатність до гнучкого адаптування у випадку застосування різних сценаріїв використання ІС.

Під час розроблення БД акцентовано увагу на логічній структурі полів бази даних і зв’язків між ними. Цей аспект дозволяє забезпечити ефективну роботу під час здійснення типових операцій, наприклад, під час зберігання, пошуку, фільтрування й агрегації даних. Під час проєктування БД необхідно також врахувати проблеми забезпечення інформаційної безпеки та вірогідності інформаційних ресурсів, що зберігаються. Це забезпечує можливість проведення наступної оптимізаційної діяльності щодо роботи зі значними обсягами відомостей. Таким чином, під час проєктування БД забезпечено стабільний фундамент для надійної роботи Web-додатку, враховуючи питання продуктивності, масштабованості й гнучкості з метою подальшого розвитку ІС.

Опис полів та їх атрибутів. У структурі БД Web-додатку зберігаються відомості, що вказують на ключові фактори взаємодії користувача з ІС. Такі відомості формуються у різних інформаційних площинах, які охоплюють як загальні інформаційні ресурси, так і аналітичні, включаючи інтеграційні механізми. Для забезпечення цілісності, логічності й масштабованості інформаційної моделі, слід концептуально визначити головні структурні одиниці інформаційних відомостей, з якими має працювати система.

Кожне поле має у собі деяку цілісну одиницю відомостей, якій притаманні унікальні ознаки, певну автономність і можливість до встановлення логічного зв’язку з іншими полями. Наприклад, незалежно від додатку, поле має мати чітку ідентифікацію, описуватися визначеним набором властивостей та виконувати певну роль у структурі даних [11].

Встановлення зв’язків між полями здійснюється з використанням системи посилань, що забезпечує ефективне структурування складних фінансових операцій і забезпечує цілісність даних під час їх модифікування. Кожне поле також володіє власним життєвим циклом з відповідно визначеним станом та умовами переходу між ними. Це забезпечує коректне опрацювання бізнес-логіки додатку.

Формалізація полів є фундаментом створення цілісної моделі даних, що надає можливість системі бути узгодженою та ефективно опрацьовувати запити користувачів. Цим самим створюються передумови щодо можливості розширення функціоналу.

Опис полів у БД описані у додатках 1–6.

Створення зв’язків між полями бази даних. На рис. 2.1. відображені зв’язки між полями спроєктованої бази даних.

На даній схемі показано не лише логічну структуру збереження інформаційних ресурсів, а й взаємозв’язки між полями, які, зрештою забезпечують роботу інформаційної системи. Основна мета, яка переслідувалась під час створення БД – це проєктування гнучкої, масштабованої, захищеної архітектури, яка здатна опрацьовувати значну кількість фінансових відомостей, забезпечуючи користувачів зручними інструментами для здійснення фінансової діяльності і планування фінансових операцій.

Головною таблицею усієї структури БД є таблиця users. У ній розміщуються відомості про реєстрацію користувачів Web-додатку. Дана таблиця є базовою для побудови зв’язків з іншими полями. У кожній

наступній таблиці зберігаються специфічні дані, які пов’язані із конкретним користувачем. З метою захисту даних автентифікаційні відомості збережені у вигляді хешів. Такий захист дозволяє виключити можливість їх одержання у випадку компрометації даних.

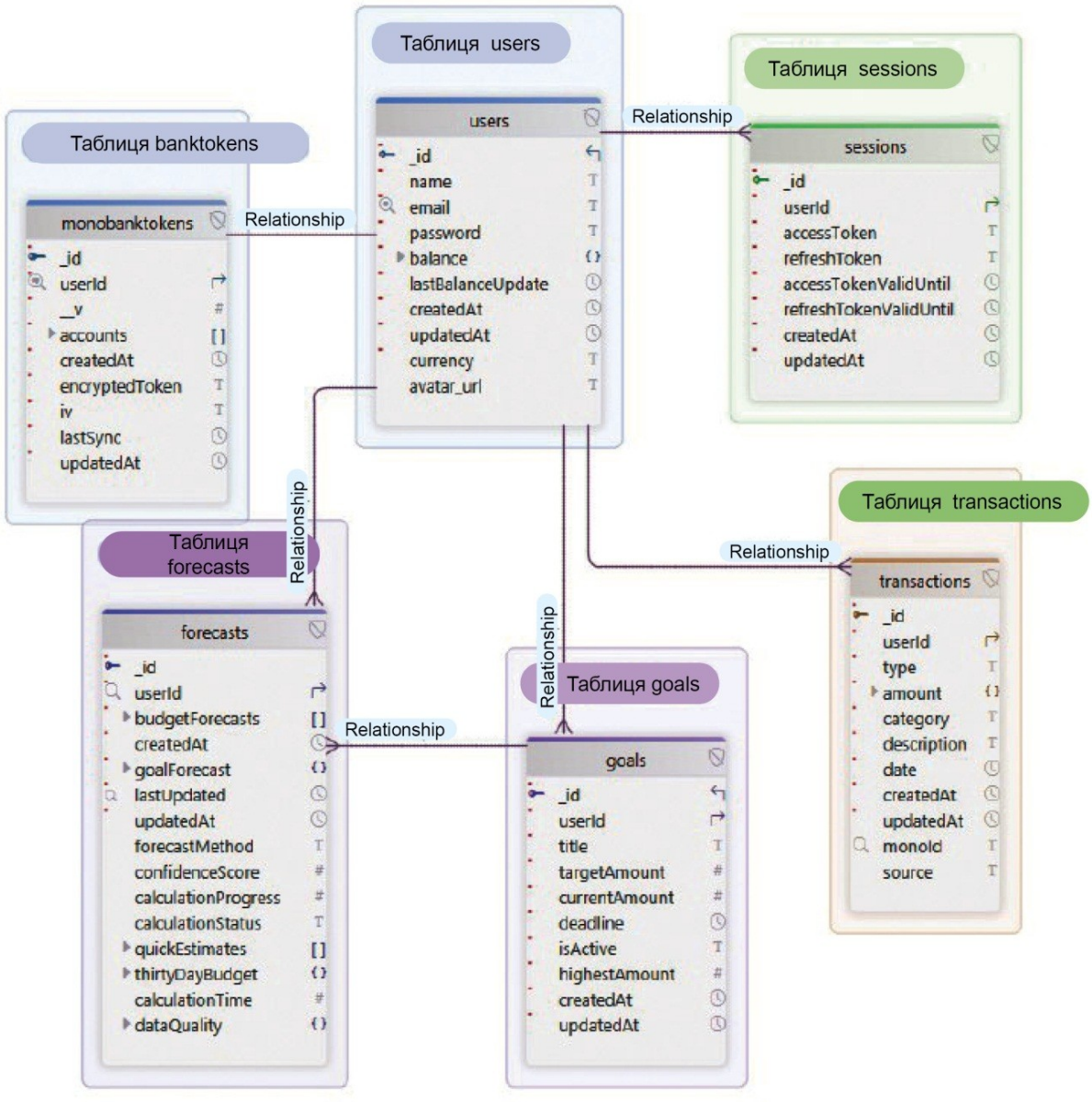

Рис. 2.1. Структура таблиць бази даних Web-додатку

Також критичне значення у функціонуванні Web-додатку має таблиця transactions. Дана таблиця містить інформацію про фінансові операції користувачів. Усі записи пов’язані із користувачем за допомогою поля userId. Це дозволяє проводити персоналізований облік фінансових операцій. Для

аналітичного опрацювання фінансових транзакції, кожна з них класифікується за визначеними категоріями, які зрештою збереженні у вигляді стрічки поля. Однак, загальна БД надає можливість внесення категорій до окремої таблиці з метою ієрархічної класифікації та детальнішого аналізування фінансових операцій у наступних періодах.

Для розширення можливості економічного планування у схемі даних передбачено додаткова таблиця. Так, таблиця goals застосовується з метою збереження фінансової мети користувача, наприклад, накопичення визначеної суми фінансів до визначеного часу або зниження витрат за певною категорією. Разом із цією колекцією працює таблиця forecast. Вона має прогнозтичні відомості, зокрема, такі як очікувані накопичування й витрати. Знову ж таки, дані колекції мають тісний зв'язок із користувачем за допомогою поля userId. Це забезпечує реалізацію персоналізованої аналітики і динамічні рекомендації стосовно фінансового менеджменту.

Головним (ключовим) функціоналом поєктованого Web-додатку є інтегрування з певним банком, що забезпечує автоматичний імпорт фінансових транзакцій. З цією метою у представленій схемі зв’язків застосовано колекцію banktokens, у якій зберігаються шифровані токени, за якими здійснюється доступ до банківського рахунку користувача. З метою захисту даних кожний токен збережено у шифрованому вигляді. Це забезпечує захист персональних даних від витоку. Окрім цього, вказана таблиця містить додаткові відомості про рахунки, що забезпечує зв'язок фінансових операцій з певним джерелом. Для підтримування механізму автентифікації й керування сесіями у БД наявна таблиця sessions. У ній зберігаються дані про активну сесію користувача, у тому числі, токени доступу, оновлення й строки їх тривалості. Такі відомості забезпечують безпеку і зручне функціонування Web-додатку, продовжуючи діючі сеанси, не викликаючи потреби у користувача повторно заходити до інформаційної системи [12].

Важливим аспектом, який взято до уваги під час розроблення схеми зв’язків, було забезпечення безпеки фінансових та персональних даних. З цією метою усі критично важливі відомості, зокрема, пароль, токени доступу до банківського рахунку, зберігатимуться із застосуванням надійного алгоритму хешування чи шифрування. Окрім цього, зв’язки між таблицями реалізуються за допомогою ідентифікатора користувача. Це забезпечує ефективне виконання запиту та допомагає об’єднати відомості, зберігши під час цього чіткість у логічному розмежуванні між відомостями різних користувачів. Також додатково реалізовано обмеження доступу до даних на рівні API, які забезпечують перевірку авторизації кожного наданого запиту з використанням токена, тим самим унеможливлюючи несанкціоноване одержання або модифікацію даних.

Додатково для забезпечення цілісності та захисту даних реалізовано засоби валідування на рівні моделі. Такий спосіб запобігає зберіганню некоректно введених чи неповних даних. Для підвищення захисту персональних даних, наприклад, облікових записів користувачів, для їх зберігання застосовується шифрування. Окрім цього, розробкою передбачається здійснення регулярного резервного копіювання бази даних, що сприяє мінімізації ризиків втрати даних під час можливих збоїв у інформаційно-комунікаційній системі. Також система логування змін дозволяє відстежувати історію дій користувача й адміністраторів, що забезпечує прозорість та спрощує аудит.

Архітектуру БД спроєктовано з врахуванням подальшого удосконалення інформаційної системи у майбутньому. Зокрема, у разі необхідності реалізування детальнішого виокремлення категорій транзакцій чи додавання певного типу фінансової аналітики, є можливість створити нові таблиці або додати інші атрибути до уже існуючих. Це забезпечує інформаційну систему гнучкістю і адаптивністю щодо подальших потреб користувачів. Також завдяки застосуванню Mongo DB є можливість

здійснювати масштабування сховища «горизонтально», що забезпечує перевагу у разі збільшення активності користувачів і обсягів інформаційних ресурсів, які необхідно зберігати.

Зв'язки бази даних підтримують їх версіонування, що забезпечує безболісне оновлення форматів відомостей без загрози втрати даних зареєстрованих користувачів. У застосунку реалізується система індексування з метою оптимізації опрацювання складних запитів з фільтруванням за часом, датою, категорією та сумами транзакцій. Це особливо важливий аспект під час опрацювання значних обсягів фінансових відомостей.

Вибір СУБД. Вибір системи управління базою даних є ключовим архітектурним рішенням під час проєктування Web-додатку для моніторингу фінансових операцій. У нашій роботі з цією метою обрано СУБД Mongo DB – документальну SQL БД, яка, на наш погляд, щонайкраще підходить як за поточними технічними вимогами, так і за майбутніми удосконаленнями, які сприятимуть розвитку Web-додатку. Обрання саме цієї СУБД дозволяє під час проєктування забезпечити гнучкість і масштабованість інформаційної системи, що є необхідною умовою успішної роботи сервісу.

Mongo DB характеризується гнучкістю у питанні зберігання даних (на відміну від інших реляційних систем управління базами даних, наприклад, Microsoft Access). У цій СУБД відомості зберігаються не у таблицях із зафіксованими зв'язками, а у вигляді документів (формат .bson). Це забезпечує збереження різноструктурованих даних, що є доволі важливо для Web-додатку, у якому постійно опрацьовуються фінансові відомості. Через можливість варіювання транзакцій та додавання нових функціональних можливостей без попереднього застосування строгих зв'язків, дана БД забезпечує можливість уникнення складних міграцій БД, які можуть доволі часто виникати у реляційних системах управління базою даних під час зміни структури.

Важливим аспектом цієї СУБД є її можливість горизонтально масштабуватися. Володіючи архітектурою, яка підтримує шардування, відомості можуть розподілятися між різними серверами. Це забезпечує зберігання й опрацювання великих обсягів інформаційних ресурсів без впливу на продуктивність. Це доволі важливий аспект для розроблюваного Web-додатку такого типу, адже він може потенційно опрацьовувати тисячі чи мільйони запитів користувачів.

Також слід зазначити, що Mongo DB технологічно дуже добре поєднується з іншими складовими ІС. Завдячуючи застосуванню Java Script на фронтенді, серверна складова та база даних забезпечують цілісність і швидкість проєктування. При цьому забезпечується мінімізація помилок під час трансформації інформації між складовими Web-додатку. Забезпечення збереження різних типів транзакцій, формування категорій витрат і надходжень, підтримання моделей прогнозування із непостійною структурою – це можливості гнучкої СУБД Mongo DB.

Таким чином, обрання Mongo DB для проєктування Web-додатку керування фінансовими операціями є обґрунтованим з огляду гнучкості, зручності розроблення, а також відповідає нинішнім вимогам до використання динамічних Web-рішень. Таке технологічне рішення не тільки дозволяє оптимізувати процеси збереження й опрацювання інформаційних ресурсів, а й забезпечити можливості для подальшого розвитку функціоналу Web-додатку.

2.4. Проєктування серверної складової Web-додатку

На серверну складову Web-додатку покладається головна роль щодо забезпечення його функціональної, стабільної та безпечної роботи. Серверна частина виконує функцію посередника між користувачами і БД, опрацьовує запити, забезпечує бізнес-логіку та відповідає за керування інформаційними ресурсами. Під час розроблення серверної складової Web-додатку головну

увагу приділено структурі організації коду, визначеному поділу відповідальності між складовими, а також забезпеченню ефективного та безпечного механізму взаємодії із клієнтською складовою.

До головного функціоналу серверної складової належать опрацювання запитів до бази даних, керування сесіями, валідування вхідних і вихідних відомостей, проведення розрахунків відповідно до правил Web-додатку, збереження й опрацювання відомостей у БД.

Важливу увагу надано розробленню механізму автентифікації й авторизації. Це забезпечить можливість ідентифікації користувача й обмежить доступність до захищених даних. Серверна частина здатна інтегруватися із іншими мікросервісами. Вдало спроєктована серверна частина дозволяє забезпечити високу продуктивність, надійність та адаптивність розроблюваного Web-додатку [13].

Під час реалізування серверної логіки, як правило, застосовують існуючі фреймворки, які здатні спростити процеси опрацювання запитів і підключень до БД. Такі підходи сприяють підтримуванню модульності, повторному використанню коду, а також полегшенню перевірки функціональності Web-додатку. Також, забезпечення принципів безпечного функціонування Web-додатку є невід’ємною вимогою під час проєктування архітектури.

Вибір технології для проєктування серверної складової. Для проєктування серверної складової Web-додатку нами запропоновано обрати середовище «Node.js» із фреймворком «Express.js», що забезпечує надійність проєктування API. Обране середовище володіє зручним інструментарієм для опрацювання маршрутизації, запитів і проміжного ПЗ, що сприяє стабільності й гнучкості під час розроблення та випробовування Web-додатку. Під’єднання змінних у середовищі здійснюється за допомогою бібліотеки «dotenv». Такий підхід дозволяє забезпечити окреме збереження конфіденційних даних від основного коду.

Збереження інформаційних ресурсів виконується з використанням бібліотеки «mongoose», яка володіє зручним інтерфейсом для розроблення документальної структури, валідування відомостей тощо. Процеси авторизування, автентифікації й хешування паролів проводяться з використанням «cookie-parser» і «bcrypt», які опрацьовують куки користувача. З метою захищення взаємодії між доменами використовується «Cors». Це дозволяє проводити опрацювання запитів, надісланих із клієнтської складової системи, яка розміщується в іншому домені.

З метою валідування вхідної інформації у запитах застосовано бібліотеку «Joi», яка надає можливість проєктувати надійні схеми для перевірки даних. Зовнішні інтерфейси прикладного програмування, у тому числі банківські послуги, можна інтегрувати з використанням бібліотеки «Axios», що сприяє швидкому і зрозумілому способу надсилання HTTP-запитів. У випадку застосування авторизації через Google використовується бібліотека «Google-auth-library», яка реалізовує OAuth 2.0.

Для логування використовується комбінація «pino» і «pino-pretty». Це дозволяє зафіксувати історію фінансових подій та проводити відповідне їх форматування для процесу аналізування. Для опрацювання дат та часу застосовано бібліотеку «date-fns». Опрацювання помилок стандартизовано за допомогою «http-errors», що дозволяє надавати HTTP-помилкам відповідний статус-код.

Особливістю серверної складової розроблювального Web-додатку є реалізування функції прогнозу фінансової діяльності користувача з використанням бібліотеки «tensorflow /tfjs». Завдяки застосуванню машинного навчання виявилось за можливе проводити інтелектуальний прогноз фінансового балансу, прибутків і витрат. Створені моделі здійснюють аналіз історичних відомостей (фінансових операцій) користувача і будують прогноз, який сприяє прийняттю обґрунтованих фінансових рішень.

Опис компонентів серверної частини. Серверна складова Web-додатку є ядром усієї інформаційної системи. З її допомогою опрацьовуються запити, що надходять із клієнтської складової, здійснюється бізнес-логіка, забезпечується взаємодія з БД і зовнішніми підпрограмами, контролюються процеси авторизації й автентифікації користувача, перевіряється достовірність введеної інформації та створюються звіти у визначеному форматі. Код сервера структурується згідно з принципами відкритої архітектури з розмежуванням відповідальності між структурними складовими: МК, маршрутизаторами, службами, утилітами, моделями тощо. Це забезпечує простоту у процесах масштабування проєкту, можливість впровадження нового функціоналу і підтримування діючих функції без ускладнень.

У табл. 2.1 описано головні програмні складові (компоненти) серверної частини.

Забезпечення функціональності інтерфейсу прикладного програмування серверної складової. Серверна складова програмного Web-додатку реалізується на основі архітектури REST API. Це дозволяє забезпечити гнучкість інтерфейсу взаємодії між клієнтським програмним забезпеченням та сервером.

Таке проєктування дозволяє реалізувати ендпоінти (кінцеві точки доступу до мережевих ресурсів, які використовуються для обміну даними між різними програмами або системами). У Додатку 7 наведені ендпоінти, які згруповано згідно до функціонального призначення.

Таблиця 2.1. Головні програмні складові (компоненти) серверної частини

Компонент	Опис компоненту
1	2
Маршрутизація (routers)	Файли у папці routers відповідають за опрацювання HTTP-запитів до відповідних частин системи. Кожен модуль (наприклад, balance .js, goal. js, bank .js) містить маршрути для відповідних функцій таких як, опрацювання даних про баланс, цілі, транзакції, тощо. Вони передають запити до відповідних контролерів
Контролери (controllers)	Дані компоненти реалізують логіку опрацювання запитів. Вони виступають посередниками між маршрутизаторами та сервісами. Наприклад, контролер forecast .js опрацьовує запити, пов'язані із прогнозуванням витрат і балансу, викликаючи відповідний сервіс з TensorFlow
Сервіси (services)	Сервіси містять бізнес-логіку, що реалізує основну функціональність веб-застосунку. Наприклад, AI_ForecastService .js відповідає за прогнозування майбутнього фінансового стану користувача на основі історичних даних. Інші сервіси, як goal. js або transactions .js, реалізують CRUD-операції та обчислення
Моделі (models)	Описують структуру документів для таких сутностей, як User, Transaction, Goal, Session, Forecast тощо. Саме моделі забезпечують взаємодію з БД та її цілісність
Проміжне програмне забезпечення (middleware)	Папка middlewares містить функції для опрацювання запитів перед тим, як вони дійдуть до маршрутизаторів. Наприклад, authenticate .js перевіряє, чи має користувач дійсний токен доступу, перш ніж дозволити доступ до захищених маршрутів
Валідація (validation)	Файли у папці validation відповідають за перевірку вхідних даних від користувача. Вони використовують бібліотеку Joi для перевірки, чи відповідають надані дані очікуваним схемам. Це важливо для безпеки та надійності веб-застосунку

Таблиця 2.1. (продовження)

1	2
Утиліти (utils)	Допоміжні функції, такі як googleOAuth2.js для автентифікації через Google, categoryMapper. js для роботи з категоріями транзакцій, або ctrlWrapper .js для обгортання асинхронних контролерів із опрацюванням помилок, розміщені у папці utils. Вони використовуються повторно в різних частинах системи
Інтеграція штучного інтелекту	Модуль AI_ForecastService.js реалізує прогнозування майбутнього балансу користувача. Він використовує TensorFlow .js для створення моделей машинного навчання, що аналізують попередні доходи, витрати та фінансові цілі. На основі цих даних система може формувати прогнозовані значення майбутнього стану бюджету, надаючи користувачу корисні фінансові підказки
Інтеграція із зовнішніми сервісами	Реалізована можливість підключення користувачів до особистого кабінету банку шляхом передачі токена. Після авторизації сервер отримує доступ до історії транзакцій користувача та може синхронізувати ці дані з власною базою. Для цього призначено окремі сервіси, контролери і моделі. Також веб-застосунок підтримує авторизацію через обліковий запис Google. Інтеграція реалізована через бібліотеку google-auth-library і допоміжний файл googleOAuth2 .js
Конфігурації та інші компоненти	Dotenv використовується для збереження чутливих даних у .env файлі (ключі, URI), pino та pino-pretty забезпечують ефективне логування, cookie-parser опрацьовує cookies у запитах, http-errors генерує HTTP-помилки

2.5. Розроблення клієнтської складової Web-додатку

Клієнтська складова Web-додатку відповідає за взаємодію із користувачем, візуалізацію інформації, одержаної із серверної складової, й ініціалізування запитів до певних інтерфейсів прикладного програмування. Ця частина Web-додатку передбачає створення інтерфейсу, за допомогою якого користувачі «спілкуються» із функціоналом Web-додатку, здійснюють управління власною інформацією, аналізують її й отримують зворотний відгук. Головною ціллю під час проєктування клієнтської складової є розроблення простого, функціонального й естетичного інтерфейсу користувача [14].

Проєктуючи клієнтську складову Web-додатку слід врахувати ряд принципів: адаптивність, модульність, можливість повторного застосування програмного коду. Web-інтерфейс розробляється із урахуванням структури відомостей, які пересилаються серверною складовою, чим досягається необхідний зв’язок між всіма компонентами Web-додатку. При цьому необхідно застосувати існуючі методи управління станом, маршрутизацією й валідацією форм.

Окрім візуалізації, клієнтська складова управляє логікою взаємодії із користувачем, опрацюванням трансакцій, заповненням форм і зміною стану інтерфейсу, візуалізацією інформації стосовно помилок у роботі, внесення відомостей у фоновому режимі й інтеграцією із зовнішніми службами за допомогою API [15–18]. Захист персональних відомостей досягається завдяки безпечному передаванню токенів автентифікації.

Обрання технологій для розроблення клієнтської складової. За основу розробки Web-додатку взято бібліотеку «React», яка забезпечує динамічне його проєктування, можливість створення компонентних інтерфейсів із швидким оновленням інформації. Завдяки «React» спрощується створення коду та управління інтерфейсом залежно від стану Web-додатку.

Для глобального управління застосунком використовується «Redux Toolkit» – офіційний інструмент для «Redux». З його допомогою забезпечується керування конфігурацією та зменшується розмір шаблонного коду. Разом із «React» «Redux» забезпечує ефективний зв’язок логіки стану із візуальними складовими. Для зберігання міжсесійних станів використовується «Redux-Persist», що забезпечує автоматичний запис та відновлення із браузера. Попередньо згадувана бібліотека «Axios», дозволяє забезпечити зручність під час виконання http-запитів до серверної складової. Цим досягається коректна робота під час відправлення та одержання фінансової інформації користувача. «React Router DOM» надає можливість забезпечити маршрутизацію у середині Web-додатку на рівні клієнта, що дозволяє досягти швидкої навігації між вкладками без перезавантаження. З метою проєктування інтерактивності форм і валідування використано бібліотеку «Formik» у комбінації з «Yup». Це забезпечило можливість створення складних форм із встановленими правилами контролю за введенням даних. Це, зокрема, зручно для форм входження до застосунку, налаштування профілю, введення інформації про транзакцію.

Для створення діаграм, візуалізації певної аналітичної інформації у Web-додатку застосовано «Recharts», що забезпечує інтегрування інтерактивних діаграм і графіків із динамічними оновленнями фінансових відомостей. Інші бібліотеки, наприклад, такі як «React-Hot-Toast», надають можливість досягти зручної візуалізації повідомлень, що видається застосунком під час роботи, бібліотека «Swiper» застосовується під час проєктування, наприклад, слайдерів, а для створення адаптивних іконок застосовується бібліотека «Lucide-React».

Для запуску проєкту використовується «Vite», за допомогою якого досягається швидке розроблення та оптимізація проєктування. За допомогою плагіну «vitejs/plugin-react-swc» досягається більша продуктивність через швидку трансляцію JSX.

Для досягнення якості коду застосовується «ESLint» у комбінації з плагінами для «React». Цим досягається можливість виявлення помилок на початкових етапах проєкту.

Структура Web-сторінок та елементів інтерфейсу. Базою архітектури клієнтської складової є можливість розподілу коду на окремі елементи, які дозволяють сформувати необхідні функціональні блоки інтерфейсу. Усі ці складові групуються в спеціальну папку components. Окремий модуль забезпечує виконання визначеного завдання, при цьому він незалежний від вмісту Web-сторінки.

В окремі папки вносяться наступні елементи: фінансовий баланс, витрати, прогноз, графік транзакцій. Для опрацювання складніших операцій, наприклад, візуалізація прогнозу, використовуємо окремі компоненти, наприклад «BankConnect». Компонент «ErrorBoundary» відповідає за опрацювання помилок. Це забезпечує стабільну і зручну роботу додатка. Завдячуючи можливостям бібліотеки «Formik», всі спроєктовані форми у Web-додатку здатні інтегруватися із компонентом валідування та управління введенням інформації.

Глобалізація Web-додатку реалізується із застосуванням «Redux Toolkit». Його стан поділяється на логічні домени, які внесені до окремих категорій (goals, bank, transactions…). У кожному домені є слайси, за допомогою яких структуруються дані, редуктори, які забезпечують оновлення статусу сторінки, та асинхронні функції опрацювання запитів. «Redux» знаходиться у бібліотеці «Store.js», у якій об’єднуються усі редуктори, та реалізовано інтегрування із «Redux-persist», що дозволяє забезпечити зберігання міжсесійного стану. Таким способом досягається централізація управляння даними, що забезпечує ефективну взаємодію із бекендом та спрощує тестування функціоналу Web-додатку.

Глобалізація стану активно застосовується для синхронізації інформації між різними складовими інтерфейсу. Наприклад, під час

авторизування дані про користувача зберігаються в auth-слайсері і таким чином є доступними іншим модулям Web-додатку без потреби надсилання повторного запиту. Таким чином підвищується ефективність роботи застосунку, зменшується серверне навантаження й забезпечується узгодженість візуалізації даних в UI.

На вітальній сторінці розміщуються назви заголовків, рисунки, інформація про застосунок і кнопки переходу до вікон реєстрації та входу. Вітальна сторінка не має перенавантаження контентом, оскільки забезпечує функцію презентації Web-додатку і має заманити користувача до його встановлення. Початкова сторінка Web-додатку (макет) представлена на рис. 2.2.

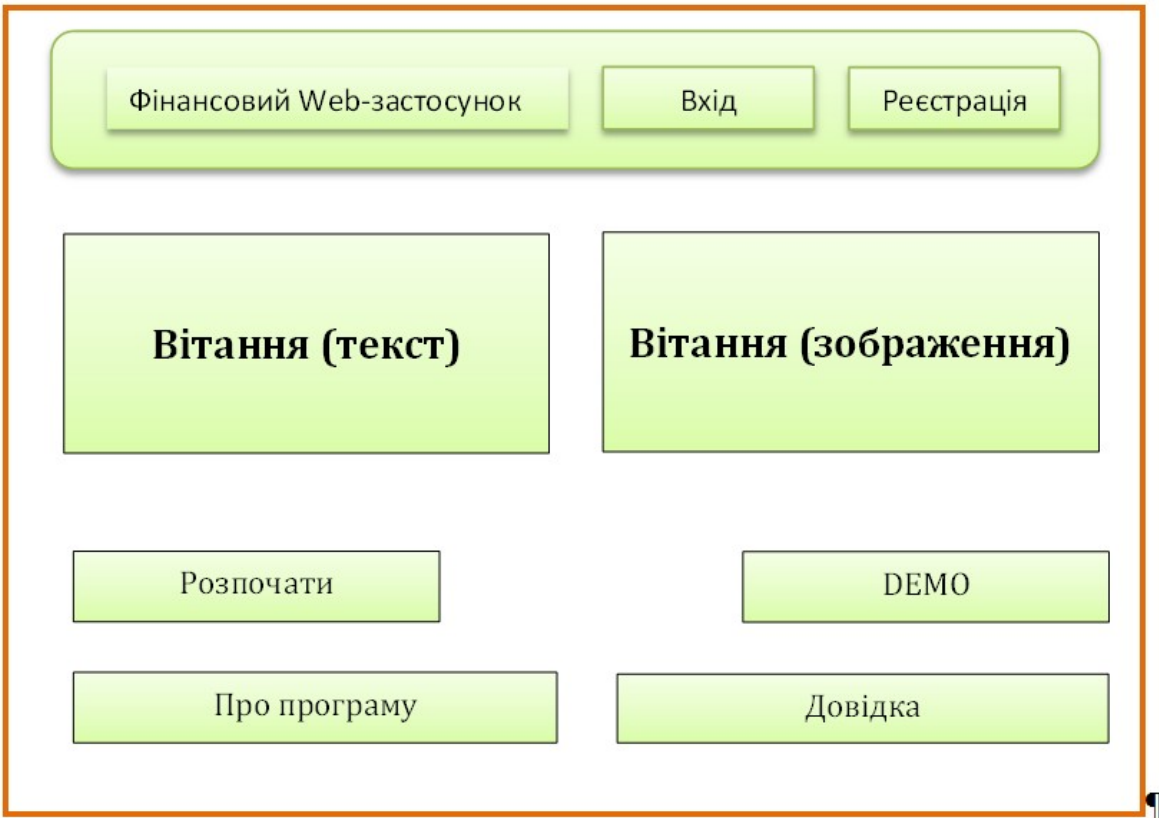

Рис. 2.2. Початкова сторінка Web-додатку (макет)

Вхід у Web-додаток передбачає використання сторінок авторизації або реєстрації. Дані сторінки спроєктовані як звичайні форми (використано «Formik»), що реалізує функції зручної валідації та обробки уведеної інформації. У випадку успішності валідації статус користувача оновлюється

у «Redux», при цьому здійснюється перенаправлення користувача на головну сторінку Web-додатку. Якщо відомості про користувача уведено невірно (наприклад, невірний логін або пароль), він одержує відповідну інформацію про помилку входу до застосунку. Також передбачено опрацювання певних серверних помилок, наприклад, відсутність мережі чи зв'язку із сервером. Макет сторінки входу до Web-додатку показано на рис. 2.3.

Рис. 2.3. Макет сторінки входу до Web-додатку

Найнасиченішою сторінкою Web-додатку за кількістю функціональних елементів є головна сторінка. Тут зосереджується основні фінансові відомості – баланс, витрати, курс валют тощо. Важливим аспектом є інтегрування у додаток модального вікна прогнозування, у якому візуалізується прогноз про фінансові тенденції. Вигляд головної сторінки створений за принципами побудови дашборду. Це забезпечує користувача можливістю отримувати повну інформацію про свій фінансовий стан. Усі елементи головної сторінки мають коректно відображатися на різних типах і розмірах моніторів (екранів). У мобільному дивайсі структурні елементи

сторінки автоматично групуються у вертикальній площині із оптимальними розмірами. Макет головної сторінки показано на рис. 2.4.

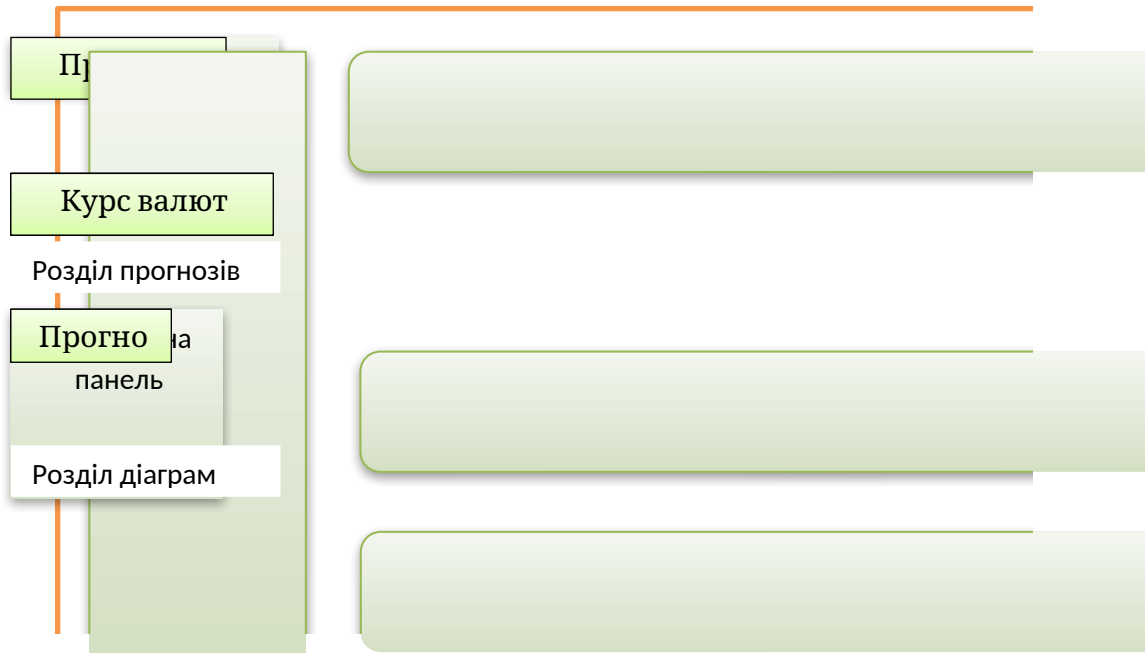

Рис. 2.4. Макет головної сторінки

Для аналізування фінансових доходів і витрат спроєктовано сторінку аналітики, у якій з використанням бібліотеки «Recharts» реалізовано візуалізацію діаграм за відповідними категоріями. Таке представлення сприяє глибшому аналізуванню фінансової поведінки та надає можливість визначати тренди. Макет даної сторінки показано на рис 2.5.

Налаштування профілю користувача виконується на сторінці спроєктованій у вигляді форми, яка забезпечує можливість зміни ім’я користувача, паролю й вибору валют. Макет сторінки показано на рис. 2.6.

Окремий функціонал виконує сторінка трансакцій. У ній об’єднуються усі фінансові операції, які провів користувач, є можливість додавання нових операцій у ручному режимі та закладається функціонал під'єднання до банку. Тобто, досягається повний контроль за фінансовими операціями на одній

сторінці, а інтегрування із банком забезпечує автоматизацію значної кількості процесів.

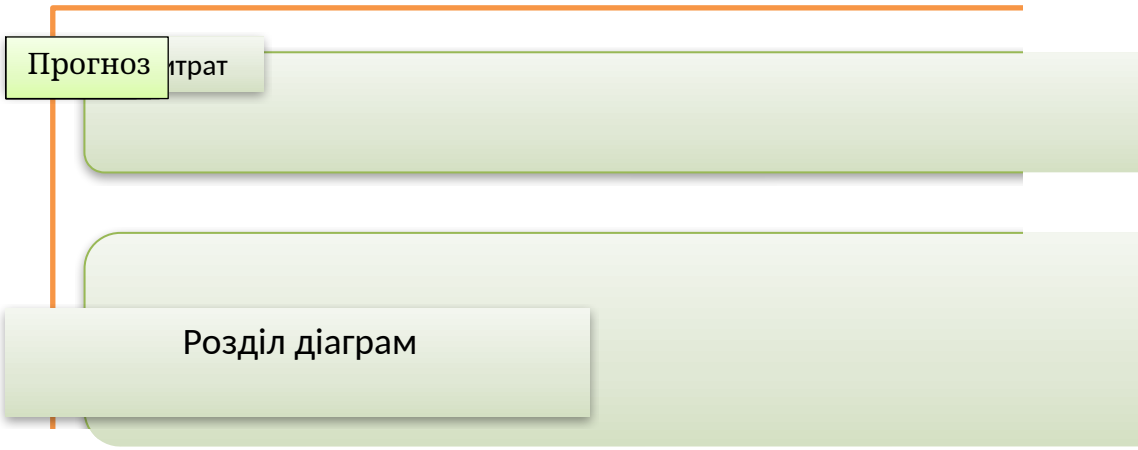

Рис. 2.5. Макет сторінки аналітики

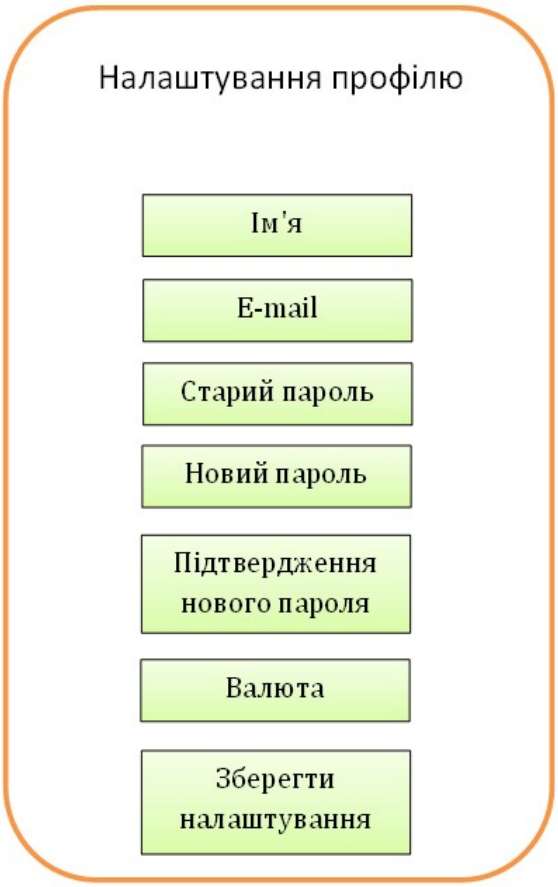

Рис. 2.6. Макет сторінки налаштування профілю користувача

Окрім цього, дана сторінка здійснює підтримку автоматичного оновлення інформації після додавання нової або редагування існуючої транзакцій. Завдячуючи адаптивності у дизайні, сторінка зручно візуалізується на будь-яких дивайсах. Перспективним виглядає можливість розширення аналізу транзакцій саме на даній Web-сторінці, що забезпечить користувача можливістю одержувати швидкий зворотній зв'язок стосовно стану рахунку [19].

Для оптимізування продуктивності під час опрацювання великого обсягу транзакцій застосовуються відповідні запити до БД та процедура кешування часто використовуваних даних. Макет сторінки трансакцій показано на рис. 2.7.

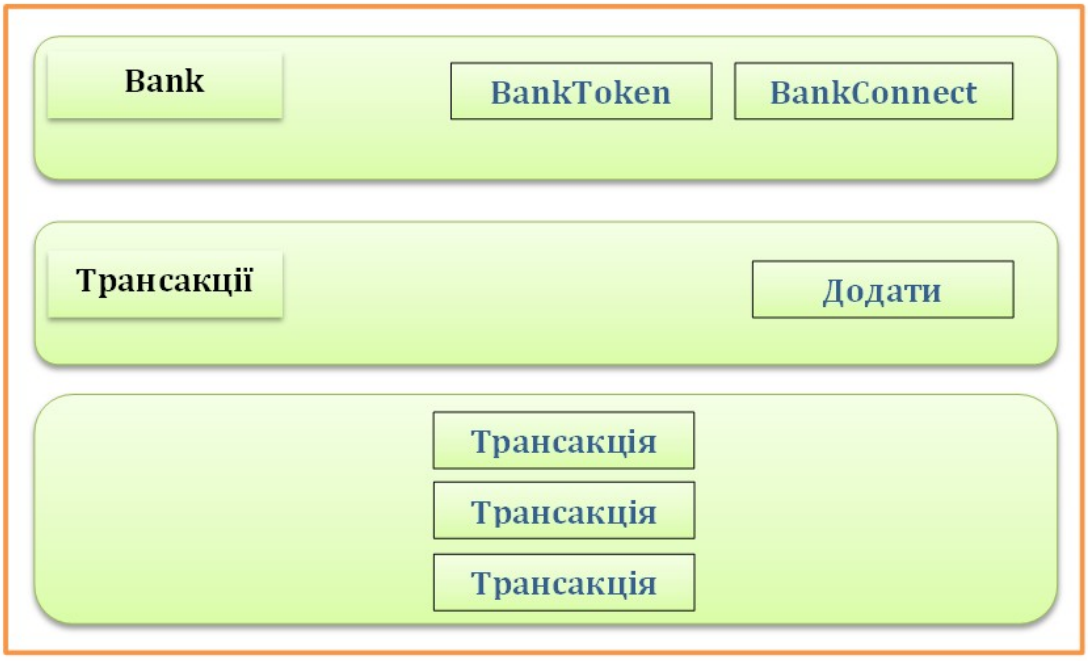

Рис. 2.5. Макет сторінки трансакцій

Проєктування інтерфейсу користувача. Під час проєктування інтерфейсу користувача значну увагу приділено забезпеченню необхідної взаємодії користувача із системою. Пріоритетним завданням було забезпечення швидкого доступу до головного функціоналу застосунку при

застосуванні мінімальної кількості операцій та логічної навігації, що сприяє зручному орієнтуванню навіть для недосвідчених користувачів.

Інтерфейс спроєктовано з врахуванням принципу «основне функціональність, а не форма». Тут кожний структурний елемент містить функціональне призначення. Забезпечено створення уніфікованої системи макетів Web-сторінок і шаблонів різних елементів (форм, таблиць, діаграм). Такий підхід забезпечив послідовність графічного інтерфейсу включаючи застосування розширення функціональності.

Ергономічність у створенні дизайну була головним чинником. Розміри іконок, які використовуються для управління застосунком, сприяють зручності під час користування на різних дивайсах. Єдиний дизайнерський підхід забезпечує єдність у сприйнятті усього інтерфейсу. Інтерактивні елементи значно сприяють візуалізації інформації та забезпечують простоту під час взаємодії користувача та інформаційної системи, а спроєктовані діаграми показують зміну у фінансовому прогнозі згідно з проведеними трансакціями.

Додаток має багатоступеневий захист від помилок. Це і клієнтська валідація у формі реального часу, і серверна валідація під час важливих операцій, і візуалізація помилок валідації. Повідомлення під час виявлення проблеми створюються без застосування наукових чи технічних термінів і є зрозумілими для користувача [20].

Оптимізація продуктивності інтерфейсу виконана за допомогою компоненту «Lazy loading», кешованих обчислень, оптимізації графічних даних і кешування статичних відомостей.

У результаті спроєктовано гнучкий, зручний інтерфейс, що дозволяє забезпечити потреби користувачів та ефективність у роботі з інформацією, володіючи при цьому простотою та візуальною привабливістю.

Розділ 3.
ПРОЄКТУВАННЯ WEB-ДОДАТКУ

3.1. Налаштування середовища проєктування Web-додатку

Для проєктування клієнтської складової Web-додатку для керування фінансами підрозділу правоохоронного органу обрано осучаснений та зручний інструментарій. Основою для підготовки програмних кодів обрано редактор «VS Code», який володіє гнучкістю, широким набором розширень та високою продуктивністю, що дозволяє максимально спростити процеси проєктування додатку, зокрема, інтерфейсу користувача. Крім даного редактора, для підготовки середовища проєктування використовувались: «Vite» – зручний інструмент, за допомогою якого проводиться швидка збірка проєкту; «ESLint» – інструмент для дотримання якості коду [21]; «Git» – забезпечує збереження історії створення проєкту, дозволяє проєктувати новий функціонал і запобігати конфліктам. Такий інструментарій забезпечує оптимізацію процесу розроблення клієнтської складової Web-додатку.

Особливу увагу приділено підготовці середовища для проєктування додатку з огляду підтримання можливостей мови програмування Java Script. Таким чином вдалось адаптувати систему для автоматизованого розпізнавання компонентів, шляхів імпортування й інтегрування із статичним аналізуванням типу даних. Для додержання «чистоти коду» підготовлено лінтери та форматери, які самостійно форматують файли та витримують єдиний стиль.

На початковій стадії проєкту створено набір папок, у які за логічним принципом розподіляються усі складові додатку (Web-сторінки, утиліти, типи інформаційних ресурсів тощо). Це дало змогу полегшити навігацію під час створення додатку та підготовити базу для його подальшого проєктування й удосконалення.

У подальшому проведено розроблення скриптів, які забезпечують швидкий старт Web-застосунку під час проведення його попередньої апробації, об'єднання компонентів та тестування коду на відповідність вимогам [22].

У результаті, підготовчі заходи дозволили створити необхідну інфраструктуру, яка сприяє впорядкованому та ефективному проєктуванню клієнтської складової Web-додатку.

3.2. Проєктування клієнтської складової Web-додатку

Головним аспектом клієнтської складової Web-додатку є базовий компонент «App». Він є відповідальним за маршрутизацію й авторизацію, під'єднання служб, наприклад, системи опрацювання помилок. Перше завантаження програми здійснює перевірку токена авторизації (виконується запит до серверної складової), що дозволяє відновити діючу сесію користувача, якщо вона залишалася активною.

Навігація між Web-сторінками організована за допомогою маршрутизації. Доступ до тієї чи іншої Web-сторінки залежить від встановленого режиму доступу користувача. Для встановлення режиму доступу використовуються «PublicRoute» та «PrivateRoute». Загальна маршрутизація відкриває Web-сторінки реєстрації і входження до додатку, а індивідуальна – захищає функціонал користувача від несанкціонованого доступу [23].

Для забезпечення швидкодії застосунку використано завантаження Web-сторінок за допомогою функціоналу «Lazy». При цьому компонент «Suspense» забезпечує візуалізацію процесу завантаження.

Структурно Web-додаток також складається із компонентів верхнього рівня: «Layout» – приватних, «PublicLayout» – публічних. Це забезпечує чітке розділення інтерфейсу й спрощення маршрутизації. У лістингу 3.1 відображено основну конфігурацію маршрутизації у Web-додатку.

Лістинг 3.1 Основна конфігурація маршрутизації застосунку:

```
<ErrorBoundary>
<Suspense fallback={<LoadingSpinner />}>
<Routes>
<Route index element={<Navigate to="/landing" />} />
{/* Public routes with PublicLayout */}
<Route element={<PublicLayout />}>
<Route path="/landing" element={<PublicRoute component={<Suspense
fallback={<LoadingSpinner />}><LandingPage /></Suspense>}
redirectTo="/home" />} />
<Route path="/signup" element={<PublicRegisterRoute
component={<Suspense fallback={<LoadingSpinner />}><SignupPage
/></Suspense>} redirectTo="/signin" />} />
<Route path="/signin" element={<PublicRoute component={<Suspense
fallback={<LoadingSpinner />}><SinginPage /></Suspense>}
redirectTo="/home" />} />
</Route>
{/* Private routes */}
<Route path="/" element={<PrivateRoute component={<Layout />}
redirectTo="/signin" />}>
<Route path="home" element={<Suspense fallback={<LoadingSpinner
/>}><MainPage /></Suspense>} />
```

Проєктування інтерфейсу авторизації. Інтерфейс авторизації реалізовувався у вигляді універсального модуля, який автоматизовано встановлює режим роботи в залежності від маршрутизації.

Керування базовою формою відбувається за допомогою «Formik», що дозволяє забезпечити управління полями введення, опрацювання помилок і надсилання даних. На цьому етапі передбачається, що під час режиму реєстрації рендереться додатковий запис для підтвердження паролю, тоді, як

під час входження до застосунку заповняються лише поля електронної пошти і паролю. Валідування даних реалізуються з використанням бібліотеки «Yup», за допомогою якої встановлюються правила створення паролю, визначається формат e-mail і перевірка полів паролів на збігання.

З метою зручності використання додатку передбачається окремий елемент поля паролю, який дозволяє перемкнути у режим видимості під час його уведення. Процес опрацювання авторизування організовано за допомогою «Redux». Після відправляння форми активізується відповідна дія «LogIn» чи «Register» в залежності від режиму. Для опрацювання результату застосовано методику .unwrap(), яка забезпечує безпосередньо отримання успішної відповіді чи опрацьовує помилку.

Окрім звичної форми для авторизування, передбачається спосіб входження до додатку за допомогою акаунта Google. З цією метою запроваджений обробник «HandleGoogleSignIn», який дозволяє виконувати запит і перенаправити на Web-сторінку авторизування Google.

Помилки, які можуть виникнути внаслідок реєстрації чи входження до Web-додатку, візуалізуються безпосередньо біля відповідних полів введення, що дозволяє оперативно інформувати користувача про причини невдалої операції.

Проєктування головної сторінки. Під час проєктування головної Web-сторінки додатку за допомогою компонента «Dashboard» виконувалось завдання отримати логічну, гнучку архітектуру. Основна ціль – досягнути якнайшвидшого завантаження основних елементів інтерфейсу, відомостей про користувача; ефективного управління станами; зручної взаємодії з інтерфейсом.

Для організації потрібного потоку відомостей, насамперед, закладено запити, які мають виконуватись відразу після завантаження Web-сторінки. Сюди віднесено: баланс, транзакції, прогнози. З цією метою на початковому етапі розроблення Web-застосунку створювались відповідні сервіси на

«Redux». Під час монтування компонента за допомогою «UseEffect» відбувається поступовий запуск потрібних запитів. Їх порядок показано на лістингу 3.2.

Лістинг 3.2 Запити при монтуванні для отримання даних:

```
useEffect(() => {
  dispatch(fetchGoals());
  dispatch(fetchTransactions());
  dispatch(fetchBalance());
  dispatch(calculateBudgetForecast());
  dispatch(calculateGoalForecast());
  dispatch(fetchGoalForecasts());
  dispatch(fetchCategoryForecasts());
}, [dispatch]);
```

Окремо проєктувалось гнучке управління локальними станами. Для прикладу, для відкривання вкладок прогнозування за цілю або категорією слід перемикнути їх у модальному вікні.

Для управління модальними вікнами спроєктовано два основні параметри. Параметр відкривання модального вікна прогнозування приймає функцію вкладки за замовчуванням (у нашому випадку це вкладка «goals»), визначає активну вкладку і відкриває модальне вікно за допомогою стану «SetIsForecastModalOpen» у показник «True». Параметр закривання модального вікна встановлюється за допомогою стану видимості модального вікна в показник «False» через «SetIsForecastModalOpen». Цим досягається його приховування.

Під час проєктування структури інтерфейсу використовувалась концепція розділу відповідальності, тому кожний тип відомостей представлявся окремим компонентом. Зокрема, дохід відображається «Баланс», витрати «Витрати», курси валют «Курс валют». З метою перегляду

прогнозів спроєктовано секцію прогнозів, а аналізування витратної складової реалізовано за допомогою вікна трансакцій.

Додатково акцент було зроблено на розширеній аналітиці через компонент «DetailedForecastModal», який приймає необхідні пропси для контролю стану та відображення потрібної вкладки.

Проєктування модуля аналізу фінансового балансу. Проєктування модуля аналізу фінансового балансу є пріоритетною частиною розроблення інструменту, що не лише зберігає фінансові транзакції, а й забезпечує можливість здійснювати аналіз фінансових тенденцій. Попередньо розроблений інтуїтивний інтерфейс (його стиль) став базою Web-сторінки аналізу. Він створює можливості для оцінювання розподілу витратної і доходної частин за категоріями як на підставі аналізу наявних даних, так і на основі прогнозування на майбутні періоди. Функціонально модуль має подвійний режим візуалізації відомостей: «факт» та «прогноз». З цією метою застосовано можливість переключення між даними, які враховують як проведені фінансові транзакції, так і спрогнозовані показники, що згенеровані за допомогою системи прогнозування.

Спочатку нами спроєктовано структуру, яка забезпечує отримання транзакцій і прогнозів за допомогою «Redux». З цією метою застосовувалися функції «UseSelector», що забезпечує доступ до потрібних складових стану Web-додатку. Разом з цим організовано асинхронний запит до сервера з метою загрузки прогнозованих відомостей з використанням «Dispatch» (застосовується разово при монтуванні компоненту).

Основною трудністю під час створення проєкту було вирішення питання досягнення правильного опрацювання відомостей для побудови діаграм. Так як за сутністю фінансові трансакції та прогнозні дані володіють різною структурою, для кожного з них розроблено відповідні функції опрацювання. Для реальних відомостей застосовується групування фінансових операцій за категоріями із сумуванням. Для прогнозування – отримання передбачуваних

сум на поточний місяць. Тобто, названі типи даних піддаються конвертації до формату, який зручний для наступної візуалізації. Механізм групування відомостей за категоріями показано на лістингу 3.3.

Лістинг 3.3. Механізм групування реальних даних за категоріями:

```
const processTransactionsByCategory = () => {
  if (!transactions?.length) return [];
  const categoriesMap = transactions.reduce((acc, transaction) => {
    if (transaction.type === viewType) {
      acc[transaction.category] = (acc[transaction.category] || 0) +
transaction.amount;
    }
    return acc;
  }, {});
  return Object.entries(categoriesMap).map(([category, amount]) =>
({ category, amount }));
};
```

Для візуалізації аналізу фінансових операцій використано елемент «BarChart» бібліотеки «Recharts», який надає можливість створювати діаграми із налаштуванням зовнішнього вигляду. Залежно від режиму, на діаграмі можуть відображатися не лише підсумовані дані за категоріями, а й реальні та прогнозовані значення у комбінації. З метою поліпшення взаємодії із клієнтом спроєктовано окремий елемент підказки, який під час наведення на стовпчик діаграми візуалізує інформацію про загальне значення фактичних витрат чи значення, яке прогнозується.

Для забезпечення безперервності користувацького досвіду враховано ймовірні ситуації завантаження чи відсутності відомостей. Якщо прогнозовані відомості у процесі завантаження, то на екрані про це показується повідомлення. Якщо відомостей для прогнозування недостатньо відображається визначений текст (замість діаграми).

Проєктування модуля керування фінансовими операціями та інтеграція із зовнішніми сервісами. Робота із фінансовими операціями (транзакціями) здійснюється компонентом «Transactions». Під час завантаження Web-сторінки ініціалізація інформації здійснюється в автоматичному режимі. Тобто: завантажуються усі фінансові операції та наявний у підрозділу баланс з використанням функцій «FetchTransactions» і «FetchBalance» за допомогою «Redux-thunk». З цією метою задіються хуки «UseEffect».

Відповідальний за роботу із застосунком може додати нову фінансову операцію за допомогою модального вікна, яке викликається натисканням кнопки «Додати». Під час цієї операції здійснюється перевірка готовності балансу. За умови, якщо його ще не завантажено, додатково викликається «FetchBalance».

Для інтегрування із зовнішнім фінансовим сервісом спороєктовано елемент BankConnect, що забезпечує під'єднання рахунку підрозділу до визначеного банку. За наявності такого під'єднання відбувається отримання доступу до власних рахунків, синхронізування фінансових операцій. Інтерфейс під'єднання спроєктовано так, що дані про рахунки та можливості синхронізації візуалізуються у окремому фреймі, який можна згортати або відкривати. Це реалізується за допомогою стану внутрішнього компонента й опрацювання події кліком.

І на завершення, спроєктовано засіб для перевірки правильності уведеного токена, який забезпечує уникнення помилки під час під'єднання й безпечний обмін інформацією. У випадку втрати під'єднання токена ІС в автоматичному режимі сповіщає про потребу повторного підключення.

3.3. Презентація роботи Web-додатку

Реєстрація і налаштування профілю. Під час першого використання Web-додатку користувач направляється на головну Web-сторінку застосунку,

у якій наведено відомості про сервіс. Головно, тут знаходяться іконки для доступу до Web-сторінок авторизування та створення акаунту (рис. 3.1). Інтерфейс головної сторінки Web-додатку може працювати і з використанням мобільних дивайсів. Окрім цього, на Web-сторінці розташовуються іконки, які вказують на головні переваги Web-додатку. Проста й інтуїтивно зрозуміла навігація дозволяє користувачеві швидко зорієнтуватися та віднайти потрібні відомості.

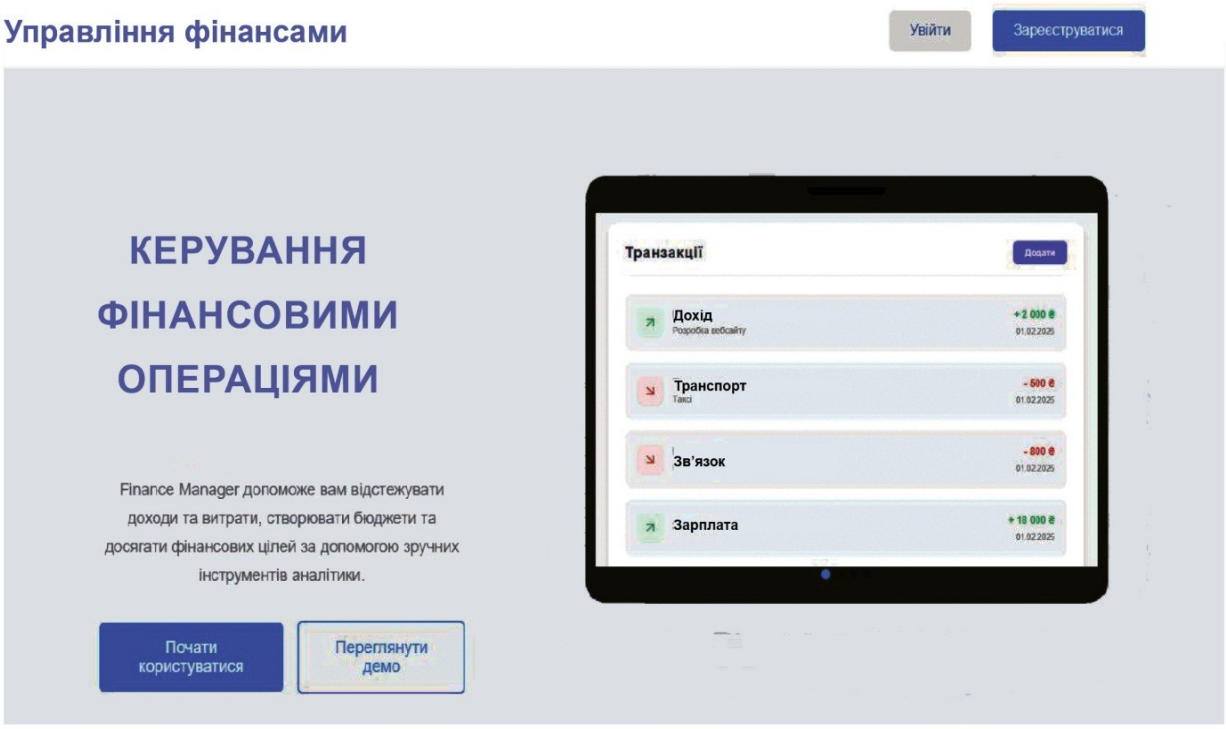

Рис. 3.1. Головна сторінка Web-додатку

Під час активування відповідної команди відбувається перехід на Web-сторінку авторизування (рис. 3.2), за допомогою якої можна увійти до власного аканту, чи на Web-сторінку реєстрації (рис. 3.3), де можна створити власний профіль. Форми передбачають перевірку уведеної інформації в автоматичному режимі: формату електронної пошти, довжини паролю, збіг уведених паролів для підтвердження.

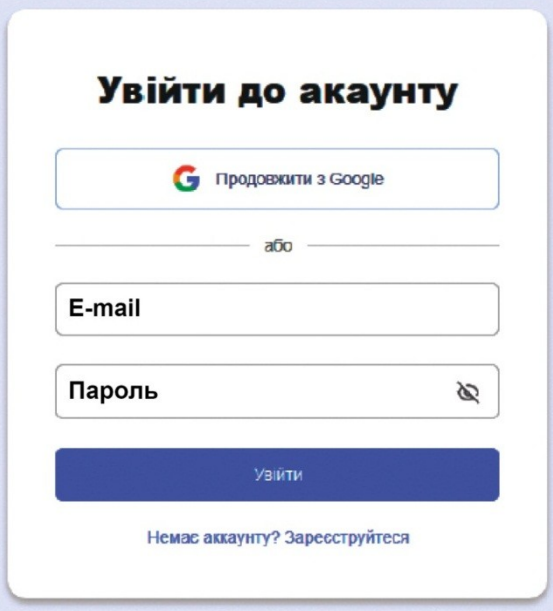

Рис. 3.2. Web-сторінка для авторизації користувача

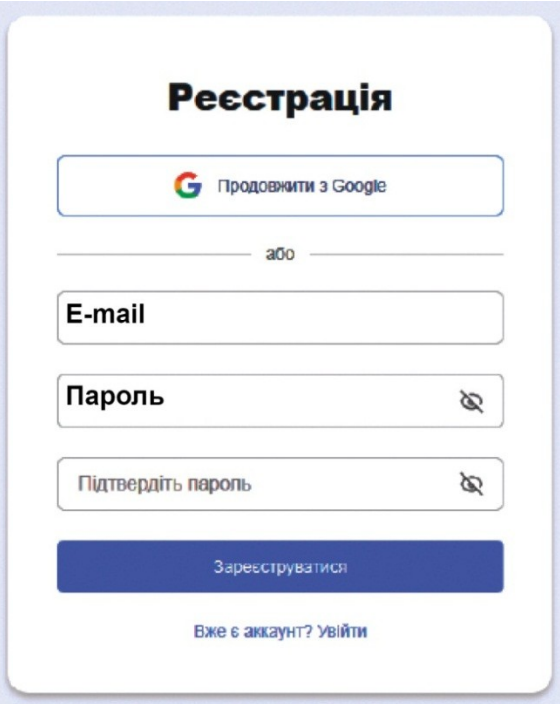

Рис. 3.3. Web-сторінка для реєстрації користувача

У випадку невірно уведених даних біля поля уведення з’являється інформація, яка допомагає виправити невідповідність. Якщо користувач здійснює спробу зареєструватися за вже існуючим емейлом ІС проінформує про це та запросить увійти до акаунту. За успішної реєстрації користувач попаде на Web-сторінку авторизації. Також забезпечено можливість відображення паролю з метою запобігання помилок під час уведення.

Валідування емейлу виконується з використанням виразів, що здійснюють перевірку наявності знаку «@», домена та структури адреси. Перевірка пароля передбачає перевірку довжини, присутність спеціальних знаків та чисел. ІС одночасно визначає кількість невдалих спроб під час входження до аканту. У випадку невдалих 3-х спроб відбувається тимчасове блокування доступу до акаунту.

Важливим аспектом є швидка взаємодія Web-додатку і користувача. Так Web-додаток інформує користувача про успішність проведених трансакцій чи про помилки, які сталися під час заповнення форм. Проєктування таких підходів дозволяє зменшити кількість помилок і підвищити задоволеність під час використання застосунку.

Окрім реєстрації за допомогою вікна авторизації передбачено спосіб авторизування за допомогою облікового запису Google. Для цього слід використати іконку «Продовжити з Google», що призводить до переходу на Web-сторінку підтвердження. У разі успішного авторизування відбувається автоматичний перехід до Web-додатку. Під час першого використання даного способу входження до Web-додатку створюється профіль користувача із збереженням адреси пошти (береться з облікового запису Google) та встановлюються налаштування за замовчуванням. Після успішної реєстрації чи входу відбувається перехід до головної сторінки Web-додатку.

Якщо у новоствореному акаунті не записано нік, візуалізується повідомлення із пропозицією перейти до налаштувань профіля (рис. 3.4). У ньому можна встановити ім’я, змінювати паролі чи встановити валюту для моніторингу фінансових операцій. Усі внесенні дані зберігаються у БД в автоматичному режимі, а введені оновлення відразу синхронізуються з інтерфейсом Web-додатку, що забезпечує актуальність візуалізованої інформації. Передбачено також вихід із акаунту за допомогою відповідної іконки в налаштуваннях, чим досягається необхідний рівень безпеки, особливо під час користування Web-додатком на спільних дивайсах.

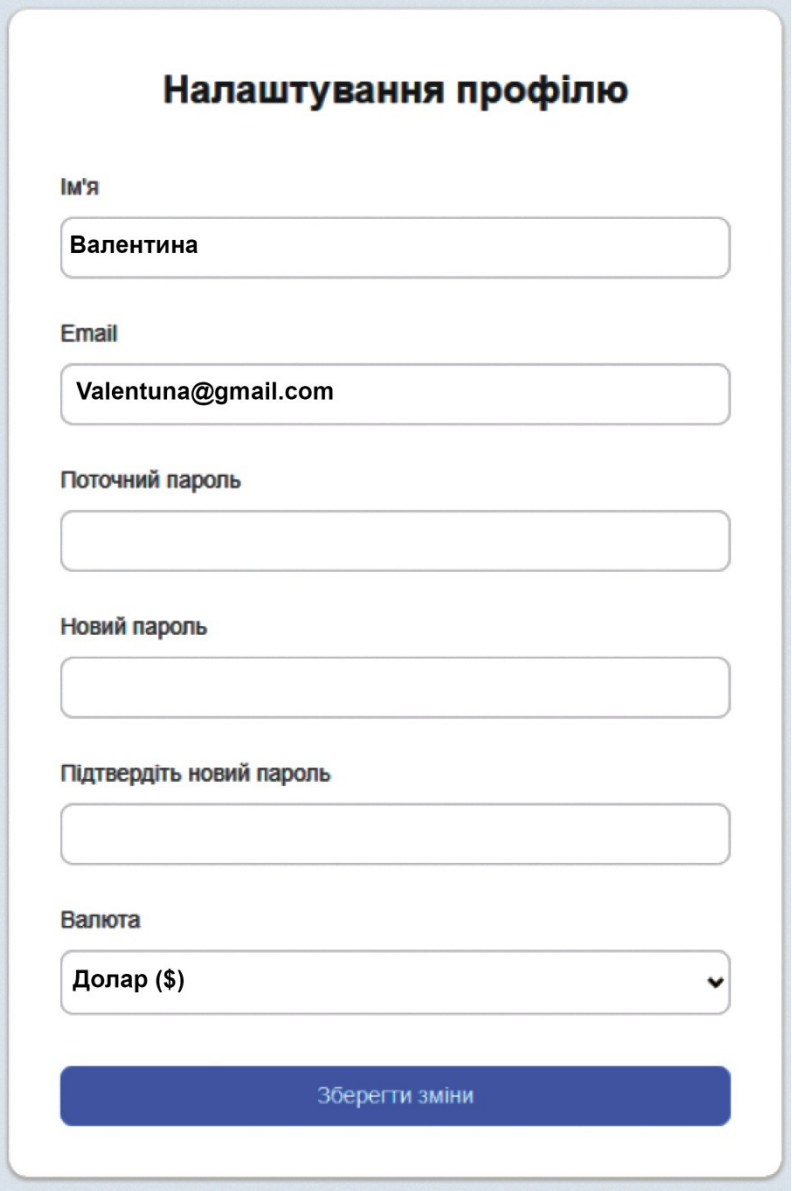

Рис. 3.4. Форма налаштувань профіля користувача

Основні функції керування фінансовими операціями. Після завершення налаштувань відбувається перенаправлення на головну сторінку Web-додатку (рис. 3.5), у якій відразу візуалізується актуальна фінансова ситуація. З верху відображається поточний баланс та витрати. Усі ці відомості завантажуються в автоматичному режимі з бекенду в результаті ініціалізації Web-сторінки за допомогою слайсів «Redux». Інтерфейс спроєктовано так, щоб можна було проаналізувати фінансовий стан без потреби переходу до інших Web-сторінок.

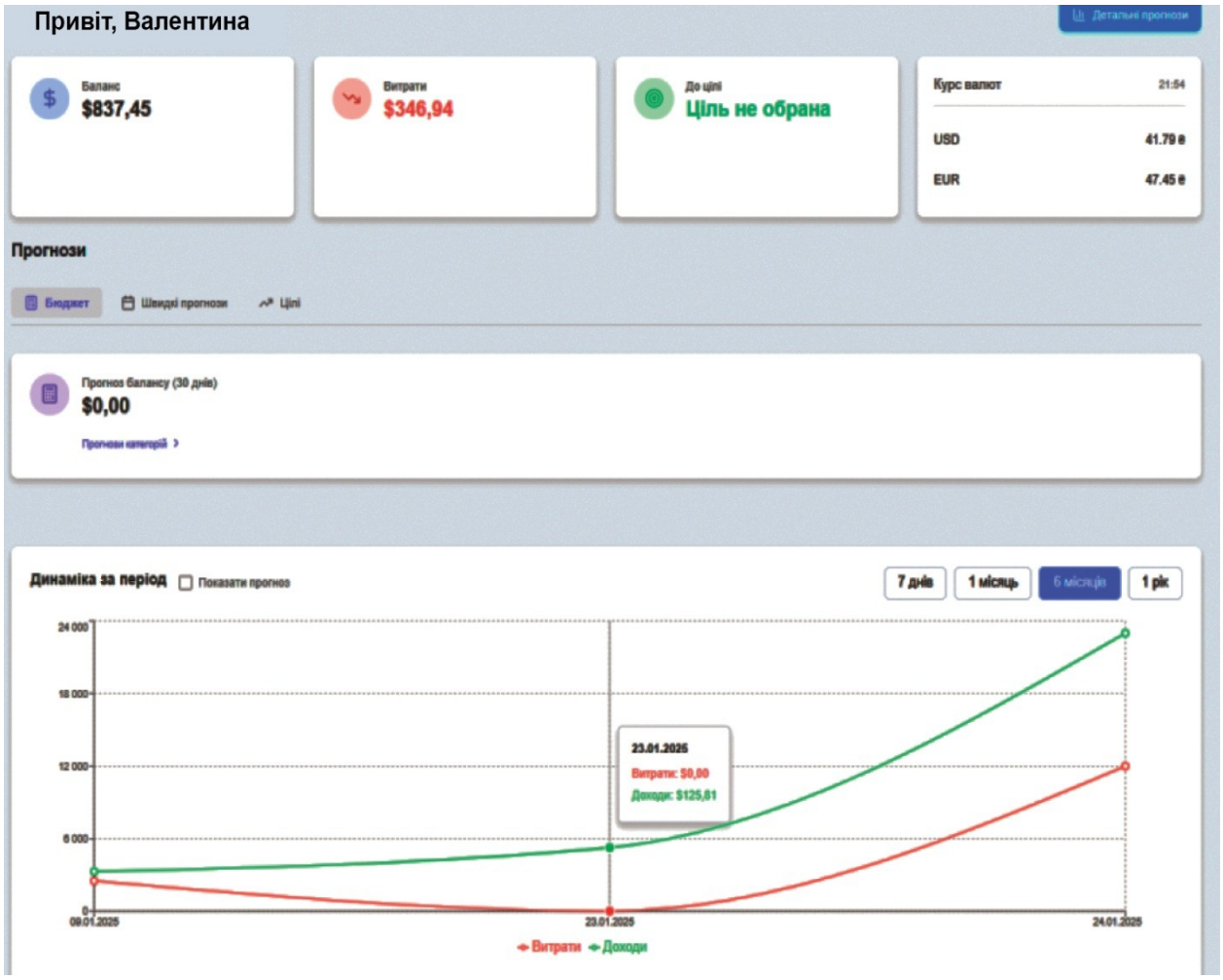

Рис. 3.5. Особиста сторінка користувача Web-додатку

У нижній частині головної Web-сторінки розташовується графік прогнозування, на основі якого можна визначити очікувані зміни фінансового балансу за деякий часовий період. Завдяки інтерактивності, даний графік відображає прогнозоване значення доходів і витрат, а це своєю чергою надає можливість визначити фінансову тенденцію. З метою поглибленого аналізування фінансового балансу спроєктовано окрему Web-сторінку. У ній можна аналізувати можливості перерозподілу дохідної та витратної частин бюджету за категоріями за допомогою діаграм. Користувачу доступна функція переключення між реальними та прогнозними відомостями (рис. 3.6), що спрощує процеси формування бюджету.

Також для користувача доступною є Web-сторінка з інформацією про проведені транзакції (рис. 3.7), у якій можна переглянути історію фінансових

операцій, а також у ручному режимі додати потрібні записи у вікні (рис. 3.8) чи здійснити синхронізацію даних з банком.

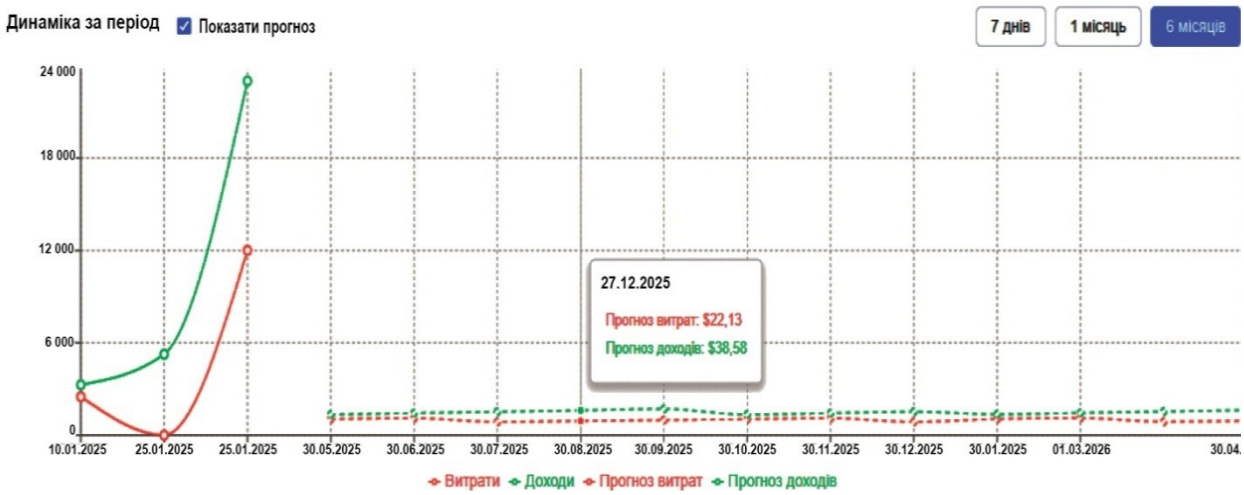

Рис. 3.6. Прогнозування фінансового балансу

Рис. 3.7. Вікно трансакцій

Функціональні можливості Web-додатку забезпечують повсякденний контроль фінансової діяльності і перспективне планування фінансових операцій, що надає йому універсальності у питаннях керування фінансовою діяльністю підрозділу.

Окрім наведених можливостей Web-додаток надає потужний інструментарій для аналітики: візуалізація фінансових потоків за визначений

час на основі діаграм (графіків) чи побудованих зведених таблиць. Це надає можливість вчасно визначити найвитратніші категорії та визначити заходи для фінансової економії.

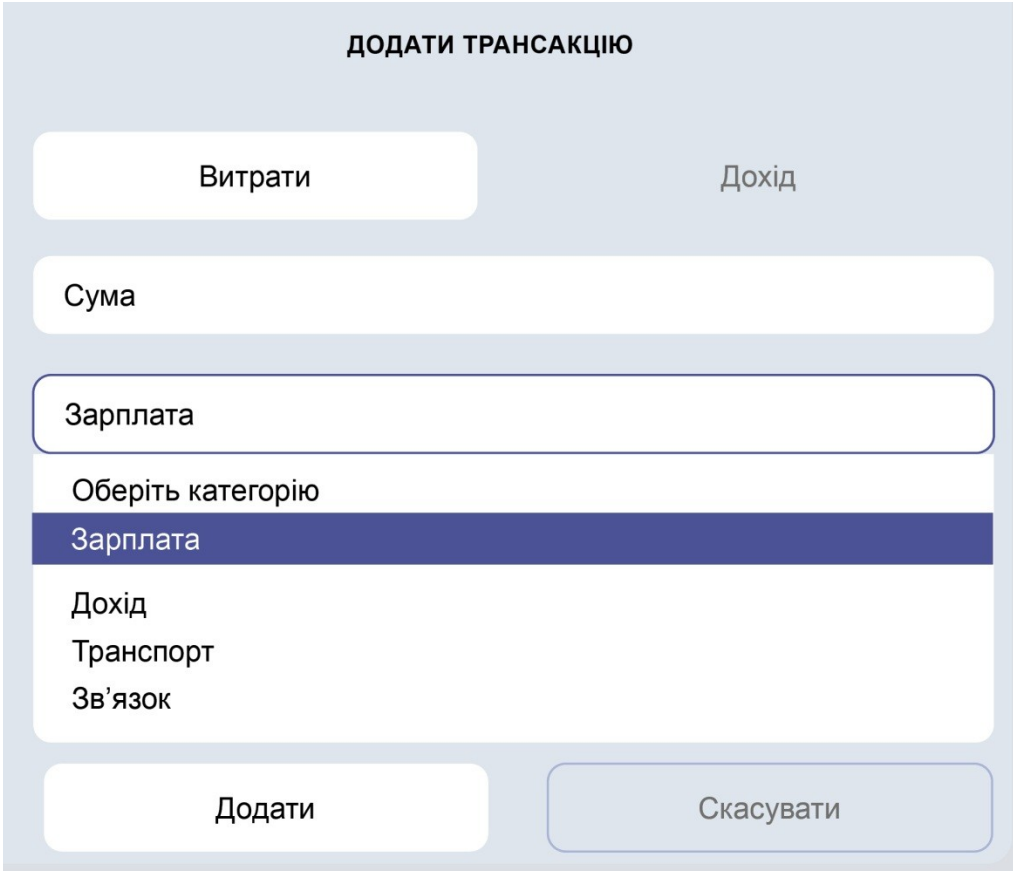

Рис. 3.8. Віно для введення трансакцій

3.4. Перспективи удосконалення Web-додатку

На етапі завершення проєктування Web-додатку можемо констатувати те, що нині він уже забезпечений зручним інтерфейсом і функціональним інструментарієм для проведення фінансового моніторингу підрозділу правоохоронного органу. Завдячуючи застосуванню сучасних досягнень у галузі комп'ютерної графіки і Web-дизайну, інтегрування із акантами Google та наявності аналітичних інструментів, спроєктований Web-додаток суттєво різниться від аналогічних додатків.

Разом з цим, спроєктований Web-додаток володіє потужним потенціалом для перспектив удосконалення. Зокрема, у перспективі доцільно

поступово розширити функціонал, що забезпечить конкурентоспроможність додатку.

Одним із таких можливих аспектів удосконалення функціоналу Web-додатку є запровадження можливості групового керування фінансовими операціями. Це забезпечить можливість здійснювати спільні фінансові операції, спільне планування бюджету, колективне його аналізування та відповідно, прийняття правильних колективних управлінських рішень. При цьому необхідно передбачити можливість встановлення необхідних прав доступу до фінансових операцій кожному окремому користувачеві.

Також важливо у подальшому розширити можливості засобів аналітики та прогнозування, наприклад, за допомогою запровадження до Web-додатку штучного інтелекту. Це спростить для користувачів підготовку фінансових звітів загалом чи за певними видами діяльності. А покращення візуалізації результатів аналітичної діяльності та прогнозування (удосконалення методик створення графіків і діаграм) надасть можливість менеджменту правоохоронного підрозділу приймати правильні управлінські рішення, пов’язані із фінансуванням діяльності чи розподілу бюджетних коштів.

Окрім запропонованого функціоналу можна також запровадити систему інформаційних повідомлень, за допомогою якої можна буде встановити нагадування про регулярний платіж (наприклад, мобільний зв'язок, орендна плата), про потребу здійснення аналізу трансакцій тощо. Уведення такого функціоналу до Web-додатку сприятиме його розвитку та популярності.

ВИСНОВКИ

У кваліфікаційній роботі спроєктовано та розроблено Web-додаток, який надасть можливість правоохоронним підрозділам здійснювати керування власними фінансовими потоками.

Під час реалізації проєкту виконані наступні завдання:

- проведено аналіз потреб у сфері керування та контрою за фінансами правоохоронного підрозділу на основі функціоналу існуючих Web-додатків для проведення моніторингу фінансових операцій;
- спроєктовано інтерфейс Web-додатку, що забезпечує швидку та зручну реєстрацію фінансових операцій;
- реалізовано функціонал обліку фінансових операцій з поділом на категорії;
- розроблено інструменти для формування бюджету і застосування обмежень за категоріями витрат;
- забезпечено можливість інтеграції Web-додатку з банками з метою автоматизації імпорту транзакцій;
- реалізовано можливість фінансового прогнозування на основі історії трансакцій.

Під час реалізації вказаних завдань використано такий основний інструментарій: платформа – СУБД Mongo DB, із використанням мови програмування JavaScript та основних інструментів (бібліотек): «Express.js», «React», «Node.js».

Функціональні можливості спроєктованого Web-додатку забезпечують повсякденний контроль фінансової діяльності і перспективне планування фінансових операцій, що надає йому універсальності у питаннях керування фінансовою діяльністю правоохоронного підрозділу. Окрім цього він надає потужний інструментарій для аналітики: візуалізація фінансових потоків за

визначений час на основі діаграм (графіків) чи побудованих зведених таблиць.

Результат тестування Web-додатку для керування фінансовими операціями показав коректність та надійність його роботи. Поряд з цим, результати апробації дали змогу означити перспективи подальшого удосконалення Web-додатку, які сприятимуть його популяризації.

В результаті виконання проєкту отримано нові знання і практичні навики із проєктування Web-додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ткаченко, О. & Рачкова, М. (2024). Система моніторингу фінансової діяльності підприємства. *Цифрова платформа: інформаційні технології в соціокультурній сфері*, 7(2), 259–273. https://doi.org/10.31866/2617-796X.7.2.2024.317735

2. П’ятикоп, О., Гніденко, В. & Воротнікова, З. (2024). Мобільний застосунок аналізу та прогнозування особистих витрат. *Вісник Приазовського Державного Технічного Університету. Серія: Технічні науки*, 1(49), 28–35. https://doi.org/10.31498/2225-6733.49.1.2024.321181

3. Goyal, K., Kumar, S. & Xiao, J. J. (2021). Antecedents and consequences of Personal Financial Management Behavior: a systematic literature review and future research agenda. *International Journal of Bank Marketing,* 39(7), 1166–1207. https://doi.org/10.1108/IJBM-12-2020-0612

4. Helmi, S., Adelia, Trisninawati, Rianawati, D. & Wedadjati, R. S. (2024). The Role Of Mobile Banking In Improving Service Efficiency To Customers In The Digital Era. Data, *Journal of Information Systems and Management*, 2(2), 62–71. https://doi.org/10.61978/data.v2i3.285

5. Pahsa, A. (2024). Financial technology decision support systems. *Journal of Electrical Systems and Inf Technol*, 11, 5. https://doi.org/10.1186/s43067-023-00130-0

6. Stefanov, T., Varbanova, S., Stefanova, M. & Alinski, I. (2024). Mobile App Prototype Supporting People With Special Needs. *TEM Journal*, 13(3), 1871–1880. https://doi.org/10.18421/TEM133-15

7. Step-by-Step Guide: Connecting MongoDB with React.js for Seamless Full Stack Development. URL: https://medium.com/@kaklotarrahul79/step-by-step-guide-connecting-mongodb- with-react-js-for-seamless-full-stack-development-db51c34da282

8. React project structure for scale: decomposition, layers and hierarchy. URL: https://www.developerway.com/posts/react-project-structure

9. React Reference Overview. URL: https://react.dev/reference/react

10. Welcome to the MongoDB Docs. URL: https://www.mongodb.com/docs/

11. Гороховатський, В. О. & Творошенко, І. С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посіб. Харків. ХНУРЕ, 92 с. URL: https://openarchive.nure.ua/handle/document/15868

12. Morgan, N. (2024). JavaScript Crash Course: A Hands-On, Project-Based Introduction to Programming. San Francisco: No Starch Press, 376 p. URL: https://www.oreilly.com/library/view/javascript-crash-course/9781098168797/

13. Herman, D. (2012). Effective JavaScript: 68 Specific Ways to Harness the Power of JavaScript. Boston: Addison-Wesley Professional, 240 p. URL: https://pepa.holla.cz/wp-content/uploads/2016/08/Effective-JavaScript.pdf

14. Gupta, Ashish. (2025). Revolutionizing Web Development: The Power of the MERN Stack. *International Journal of Scientific Research in Engineering and Management*, 9. 1–9. https://doi.org/10.55041/IJSREM46471

15. Shah, H. (2017). Node.js Challenges in Implementation. *Global Journal of Computer Science and Technology*, 17(2), 73–84. URL: https://gjcst.com/index.php/gjcst/article/view/669

16. Khan, S. M. (2023). Clean Architecture Used in Software Development. URL: https://dev.to/sardarmudassaralikhan/clean-architecture-used-in-software-development-5gpc

17. Lahute, S. V. & Jadhav, S. P. (2024). React Js - A Javascript Library. *International Research Journal of Modernization in Engineering Technology and Science*, 6(4), 1200–1203. https://www.doi.org/10.56726/IRJMETS52186

18. Bapna, S., Shrivastava, V., Pandey, A. & Sharma, A. (2024). React JS – A Frontend JavaScript Library. *International Journal of Research Publication and*

Reviews, 5(3), 1703–1705. URL: https://ijrpr.com/uploads/V5ISSUE3/IJRPR23525.pdf

19. Env Variables and Modes. URL: https://vite.dev/guide/env-and-mode

20. Marcos, Dias. (2023). Adding ESLint and Prettier to a ViteJS React project. URL: https://dev.to/marcosdiasdev/adding-eslint-and-prettier-to-a-vitejs-react-project-2kkj

21. Sistla, S. (2023). Domain-Driven Design in Modern Software Architecture Best Practices and Patterns. *Journal of Mathematical & Computer Applications*, 2(1), 1–4. https://www.doi.org/10.47363/JMCA/2023(2)E130.

22. Hawkar, J. & Khalid, E. (2024). Trends in Node.js Framework Evolution. URL: http://www.diva-portal.org/smash/get/diva2:1889284/FULLTEXT01.pdf

23. Allen, J. & Kelleher, C. (2023). React example viability for efficient API learning (REVEAL): A tool to help programmers utilize incompatible code examples in React.js. *Journal of Computer Languages*, 75, 101201 https://doi.org/10.1016/j.cola.2023.101201

ДОДАТКИ

Додаток 1

Огляд полів таблиці users

Назва поля	Тип даних	Опис поля
_id	ObjectId	Унікальний ідентифікатор користувача у БД
name	String	Ім’я користувача (за замовчуванням порожній рядок)
email	String	Електронна адреса користувача, використовується для входу до системи, має бути унікальною та відповідати шаблону e-mail
password	String	Хешований пароль користувача для безпечного зберігання
currency	String	Обрана користувачем валюта для відображення фінансових даних (за замовчуванням UAH)
lastCurrencyUpdate	Date	Дата останнього оновлення курсу валюти (за замовчуванням поточна дата)
balance	Number	Поточний баланс
lastBalanceUpdate	Date	Дата останнього оновлення балансу (за замовчуванням поточна дата)
createdAt	Date	Дата та час створення запису про користувача
updatedAt	Date	Дата останнього оновлення запису про користувача

Додаток 2

Огляд полів таблиці transactions

Назва поля	Тип даних	Опис поля
_id	ObjectId	Унікальний ідентифікатор транзакції у базі даних
userId	ObjectId	Ідентифікатор користувача, якому належить транзакція (посилання на колекцію users)
type	String	Тип транзакції («income» – дохід, «expense» – витрата), обов'язкове поле
createdAt	Date	Дата та час створення запису про транзакцію
updatedAt	Date	Дата останнього оновлення запису про транзакцію
amount	Number	Сума транзакції, обов'язкове поле
category	String	Категорія транзакції, обов'язкове поле (наприклад, «Зарплата», «Транспорт»)
description	String	Додатковий опис транзакції
date	Date	Дата проведення транзакції (за замовчуванням поточна дата)
source	String	Джерело транзакції (введено вручну, імпортовано)

Додаток 3

Огляд полів таблиці forecast

Назва поля	Тип даних	Опис поля
_id	ObjectId	Унікальний ідентифікатор прогнозу у базі даних
userId	ObjectId	Ідентифікатор користувача, якому належить транзакція (посилання на колекцію users)
budgetForecasts	Array	Масив прогнозів бюджету на різні періоди (містить дату, прогнозовані доходи/витрати/баланс, прогнози за категоріями, рівень впевненості та оцінку ризику)
quickEstimates	Array	Масив швидких прогнозів на найближчі місяці для швидкого відображення (містить місяць, прогнозовані доходи/витрати/баланс, рівень впевненості та дату розрахунку)
thirtyDayBudget	Object	Прогноз бюджету на наступні 30 днів (містить прогнозовані доходи/витрати/баланс, рівень впевненості та дату розрахунку)
calculationStatus	String	Статус процесу розрахунку прогнозу («pending», «in_progress», «completed», «failed»)
calculationProgress	Number	Прогрес розрахунку прогнозу
goalForecast	Object	Прогноз досягнення фінансової цілі (містить ідентифікатор цілі, очікувану/оптимістичну/песимістичну кількість місяців, щомісячні заощадження, ймовірність досягнення, фактори ризику та швидку оцінку)
lastUpdated	Date	Дата останнього оновлення прогнозу
forecastMethod	String	Метод, що використовувався для створення прогнозу (за замовчуванням «Advanced-AI-Enhanced-v4»)
confidenceScore	Number	Загальний показник впевненості прогнозу (від 0 до 100)
calculationTime	Number	Час, витрачений на розрахунок прогнозу (у мілісекундах)
dataQuality	Object	Показники якості даних, що використовувалися для прогнозування (кількість транзакцій, кількість місяців з даними, повнота даних у відсотках)
createdAt	Date	Дата та час створення запису про прогноз
updatedAt	Date	Дата останнього оновлення запису про прогноз

Додаток 4

Огляд полів таблиці goals

Назва поля	Тип даних	Опис поля
_id	ObjectId	Унікальний ідентифікатор фінансової цілі у базі даних
userId	ObjectId	Ідентифікатор користувача, який поставив ціль (посилання на колекцію users)
title	String	Назва фінансової цілі, обов'язкове поле
targetAmount	Number	Цільова сума, яку користувач хоче досягти, обов'язкове поле, мінімальне значення 0
currentAmount	Number	Поточна накопичена сума для досягнення цілі (за замовчуванням 0, мінімальне значення 0)
deadline	Date	Кінцевий термін досягнення цілі, обов'язкове поле
isActive	Boolean	Статус активності цілі (true – активна, false – неактивна, за замовчуванням false)
highestAmount	Number	Найвища сума, яка була досягнута у процесі накопичення цілі (за замовчуванням 0)
createdAt	Date	Дата та час створення запису про фінансову ціль
updatedAt	Date	Дата та час останнього оновлення запису про фінансову ціль

Додаток 5

Огляд полів таблиці bank

Назва поля	Тип даних	Опис поля
_id	ObjectId	Унікальний ідентифікатор запису про токен банку в базі даних
userId	ObjectId	Ідентифікатор користувача, якому належить токен
encryptedToken	String	Зашифрований токен доступу до банку, обов'язкове поле (шифрування забезпечує безпеку)
iv	String	Унікальний вектор ініціалізації, що використовується для шифрування токена, обов'язкове поле
lastSync	Date	Дата та час останньої успішної синхронізації транзакцій з MonoBank (може бути null, якщо синхронізація ще не проводилась)
accounts	Array	Масив об'єктів, що містять інформацію про рахунки користувача в банку
createdAt	Date	Дата та час створення запису про токен банку
updatedAt	Date	Дата та час останнього оновлення запису про токен банку

Додаток 6

Огляд полів таблиці sessions

Назва поля	Тип даних	Опис поля
_id	ObjectId	Унікальний ідентифікатор сесії користувача в базі даних
userId	ObjectId	Ідентифікатор користувача, якому належить сесія (посилання на колекцію users), обов'язкове поле
accessToken	String	Токен доступу, що використовується для аутентифікації запитів користувача, обов'язкове поле
refreshToken	String	Токен оновлення, що використовується для отримання нового токена доступу без повторної авторизації, обов'язкове поле
accessToken ValidUntil	Date	Дата та час закінчення терміну дії токена доступу, обов'язкове поле
refreshToken ValidUntil	Date	Дата та час закінчення терміну дії токена оновлення, обов'язкове поле
createdAt	Date	Дата та час створення запису про сесію

Додаток 7

Ендпоінти серверної частини

а) автентифікація й авторизація

Метод	Ендпоінт	Функціональність	Валідація
POST	/register	Реєстрація нового користувача	Валідація тіла запиту за схемою registerUserSchema
POST	/login	Автентифікація користувача та видача токенів	Валідація тіла запиту за схемою loginUserSchema
POST	/logout	Завершення сеансу користувача	-
GET	/get-oauth-url	Отримання URL для OAuth авторизації через Google	-

б) управління користувачами

Метод	Ендпоінт	Функціональність	Автентифікація
GET	/current	Отримання даних поточного користувача	Потрібна
PATCH	/settings	Оновлення налаштувань користувача	Потрібна

в) керування балансом

Метод	Ендпоінт	Функціональність	Автентифікація
GET	/	Отримання поточного балансу користувача	Потрібна
PUT	/	Оновлення поточного балансу	Потрібна

г) керування фінансовою метою

Метод	Ендпоінт	Функціональність	Автентифікація
POST	/	Створення нової фінансової цілі	Потрібна
GET	/	Отримання списку фінансових цілей	Потрібна
PATCH	/:goalId/activate	Активація фінансової цілі	Потрібна
PATCH	/:goalId/deactivate	Деактивація фінансової цілі	Потрібна
DELETE	/:goalId	Видалення фінансової цілі	Потрібна

д) керування трансакціями

Метод	Ендпоінт	Функціональність	Автентифікація	Валідація
POST	/	Додавання нової транзакції	Потрібна	Валідація тіла запиту за схемою transactionValidationSchema
GET	/	Отримання списку транзакцій користувача	Потрібна	-

е) прогнозування

Метод	Ендпоінт	Функціональність	Автентифікація
GET	/	Отримання загальних прогнозів	Потрібна
GET	/quick	Отримання швидких прогнозів	Потрібна
GET	/categories	Отримання прогнозів за категоріями	Потрібна
GET	/goals	Отримання прогнозів	Потрібна

е) інтегрування з банком

Метод	Ендпоінт	Функціональність	Автентифікація	Валідація
POST	/connect	Підключення API Монобанку	Потрібна	Валідація тіла запиту за схемою monobankTokenSchema
DELETE	/disconnect	Відключення API Монобанку	Потрібна	-
POST	/sync	Примусова синхронізація транзакцій	Потрібна	-
GET	/status	Перевірка статусу підключення до Монобанку	Потрібна	-