Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка автоматизованої системи обліку витрат та прогнозування фінансової поведінки на основі штучного інтелекту
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-1
В'ячеслав Іскендеров
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)

Керівник ст. викл. Олександр Стьопін
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Іскендерову В'ячеславу Ігоревичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка автоматизованої системи обліку витрат та прогнозування фінансової поведінки на основі штучного інтелекту

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 червня 20 25 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проєктів документація Node.js, Next.js, FastAPI, Supabase, Python, міжнародні стандарти фінансової звітності, закон України «Про захист персональних даних», REST API

4. Перелік питань, що потрібно опрацювати в роботі
1) Актуальність проєкту та аналіз наявних рішень
2) Теоретичні засади розробки системи
3) Імплементація автоматизованої системи
4) Тестування і оцінювання ефективності системи

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	21.05.2025	виконано
3	Огляд методів обробки природньої мови	25.05.2025	виконано
4	Огляд моделей для виявлення кібербулінгу та	25.05.2025	виконано
	їх метрик		
5	Програмна реалізація	25.05.2025	виконано
6	Написання пояснювальної записки	26.05.2025	виконано
7	Перевірка на академічний плагіат	29.05.2025	виконано
8	Нормоконтроль	30.05.2025	виконано
9	Підготовка презентації та доповіді	07.06.2025	виконано
10	Попередній захист	10.06.2025	виконано
11	Рецензування	13.06.2025	виконано
12	Захист перед ЕК	18.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

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

РЕФЕРАТ

Пояснювальна записка: 55 с., 7 рис., 4 табл., 1 дод., 12 джерел.

ГРАДІЄНТНИЙ БУСТИНГ, ДОХІД, ЗВІТНІСТЬ, ПРОГНОЗУВАННЯ, ТРАНЗАКЦІЯ, ЧАСОВИЙ РЯД, ШТУЧНИЙ ІНТЕЛЕКТ, LIGHT GBM.

Об'єкт дослідження – процес цифрового обліку та аналізу фінансової активності користувача в інформаційних системах.

Предмет дослідження – методи та інструменти обробки, аналізу й прогнозування фінансових транзакцій за допомогою машинного навчання в контексті повноцінного вебзастосунка.

Мета роботи – розробка інформаційної системи для автоматизованого обліку фінансових транзакцій та реалізація моделі прогнозування витрат користувача на основі історичних даних із використанням методів машинного навчання.

Методи дослідження – аналіз літературних джерел і предметної області, обробка та агрегація фінансових даних, методи машинного навчання для прогнозування витрат.

ABSTRACT

Bachelor’s thesis contains: 55 pp., 7 fig., 4 tabl., 1 ann., 12 references.

ARTIFICIAL INTELLIGENCE, FORECASTING, GRADIENT BOOSTING, LIGHT GBM, REPORTING, REVENUE, TIME SERIES, TRANSACTION.

The object of the study is the process of digital accounting and analysis of the user's financial activity in information systems.

The subject of the study is methods and tools for processing, analyzing and forecasting financial transactions using machine learning in the context of a full-fledged web application.

The purpose of the work is to develop an information system for automated accounting of financial transactions and implement a model for forecasting user expenses based on historical data using machine learning methods.

Research methods are analysis of literary sources and the subject area, processing and aggregation of financial data, machine learning methods for forecasting expenses.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

RL – Reinforcement Learning – навчання з підкріплення:

SL – Supervised Learning – навчання з вчителем;

SPA – Single Page Application – односторінкові вебдодатки;

UI – User Interface – інтерфейс користувача;

UL – Unsupervised Learning – навчання без вчителя;

UX – User Experience – досвід користувача.

ВСТУП

У сучасному світі ефективне управління особистими фінансами стає ключовим фактором фінансової стабільності та незалежності. Усе більш поширеною стає тенденція нераціональних витрат людьми, особливо сильно ця тенденція стосується людей молодшого віку, які зростали в епосі інтернету. Причин, що пояснюють зростання витрат, – багато, але однією з них є відсутність якісних інструментів контролю своїх затрат.

Користувачі наявних рішень часто стикаються зі складнощами в плануванні своїх витрат через непередбачуваність і недостатність відповідних інструментів прогнозування. Наявні рішення, хоча й мають певні функціональні можливості, часто не враховують персоналізовані особливості поведінки користувача та не пропонують інтегрованих автоматизованих засобів для передбачення майбутніх витрат. Іншим вада наявних додатків є не інтуїтивний візуальний інтерфейс, що відлякує користувачів або потребує від них витрат у часі для навчання користування цими інструментами.

Метою даної роботи становить розробка автоматизованої системи, що дозволяє вести облік доходу та витрат користувачів, та надає можливість отримати прогнозування стосовно майбутніх витрат на основі історичних даних витрат даного користувача за допомогою інтегрованої моделі машинного навчання.

Розроблена система може використовуватися у повсякденному житті громадян для підвищення фінансової грамотності, оптимізації планування свого бюджету та підвищення точності прогнозування витрат. Іншим способом користування додатком є ведення бюджету не тільки для окремих лиць, але й для сімей або малого бізнесу, де прогнозування витрат є значно більш критичним. Розроблена система може стати основою під час розробки більш комплексного фінансового сервісу.

1 АКТУАЛЬНІСТЬ ПРОЄКТУ ТА АНАЛІЗ НАЯВНИХ РІШЕНЬ

1.1 Постановка проблеми

Ефективне управління особистими фінансами сьогодні є важливим чинником, що сприяє на фінансову стабільність і добробут людини. Однак, багато людей, особливо це стосується молодшого покоління (23–28 років), стикаються із труднощами через тенденцію до необдуманих та ірраціональних витрат, що особливо яскраво проявляється в умовах інформаційного суспільства. Однією з ключових причин цієї проблеми можна вважати відсутність одночасно доступних, простих у використанні, персоналізованих і водночас ефективних інструментів контролю фінансових витрат.

Причини даної тенденції – тема для соціальних досліджень, але одними із найактивніших факторів, що на це впливає є широка доступність товарів через інтернет та глобалізацію ринків. Категорія людей, що зросла у часи цифровізації та в умовах широкого доступу до онлайн-магазинів і мобільних додатків, не знаходить необхідним займатися розвитком своєї фінансової грамотності. Тільки 27% з опитаних 7138 молодих людей віком від двадцяти трьох до двадцяти восьми років змогли відповісти на питання пов’язаних з основами фінансової грамотності [1]. Відсутність обізнаності у своїх фінансових рішеннях призводить до накопичення боргів, неефективного використання доходів і фінансової нестабільності.

Наявні рішення у сфері особистих фінансів, хоча й надають певні можливості ведення обліку доходів і витрат, переважно орієнтовані на стандартні сценарії використання і не враховують унікальні елементи поведінки користувача. Наслідком стає відмова користувачів використати наявні інструменти через складність, необхідність постійного ручного введення даних або недостатньо зрозумілий інтерфейс, що робить складнішим для людей формувати здорові фінансові звички.

1.2 Аналіз наявних рішень

Важливо визначитися із двома термінами, що будуть використовуватися у наступному розділі, для повного розуміння наявних проблем сучасних рішень:

– персоналізація системи – функціонал фінансової системи, спрямований на адаптування своєї логіки (хай то візуальна адаптація чи зміна обробки запитів) залежно від поведінки користувача;

– прогнозування витрат – окремий випадок персоналізації системи, що буде розглядатися у ході розробки системи, що фокусується на передбаченні витрат користувача залежно від поведінкових характеристик.

На сьогодні вже існує чимало сервісів задля розв'язання поставленої проблеми. Найпоширенішими інструментами керування особистими фінансами є:

– Mint;

– YNAB (You Need a Budget);

– Monobank;

– Spendee.

Кожен із перелічених додатків має свої особливості та функціональні можливості, проте, вони характеризуються певними обмеженнями, що не дозволяють повною мірою розв'язувати проблеми, із якими стикаються користувачі.

Почнемо аналіз із Mint. Це найпоширеніша у США система ведення персонального обліку, що пропонує зручний інтерфейс, інтегруючи інформацію з банківських рахунків та кредитних карток. Вона пропонує функціонал для створення бюджетів та отримання повідомлень про перевищення витрат. Основними перевагами є автоматизація збору даних і аналітика категорій витрат. Однак двома великі вадами цього додатку є:

– локалізація – орієнтованість виключно на американський ринок, інтеграція лише із банками у межах США, що робить його непридатним для користувачів у інших країнах;

– відсутність персоналізації – сервіс не враховує поведінкові шаблони користувача, тобто додаток не може запропонувати прогноз майбутніх витрат або порад на основі індивідуальних фінансових звичок користувача.

YNAB (You Need a Budget) використовує концепцію бюджетування з нуля, що має як вади, так і переваги. Ця модель дає більший контроль над керуванням категорій витрат, тобто користувач сам визначає, які витрати треба вносити до додатка, а які – ні. Даний сервіс немає можливості прогнозування фінансових витрат, але навіть якщо додати цю можливість, існуватиме велика ймовірність на помилку цього прогнозу через вибірковість внесених до додатка даних користувачем.

Monobank – це не просто сервіс менеджменту фінансами, а українська віртуальна банкова установа. Явною перевагою є легкість ведення своїх доходів та витрат завдяки тому, що це є банк, а значить має фізичні картки. Конкурувати з цим немає сенсу, якщо немає бажання відкривати новий банк. Проте, залишається недолік у відсутності можливостей для персоніфікованого аналізу та передбачення своїх витрат.

Spendee, своєю чергою, має дуже простий та інтуїтивно зрозумілий дизайн, який дозволяє швидко освоїтися навіть новачкам. На відміну від попередніх аналогів, цей додаток пропонує широкий набір утиліт для детального аналізу фінансових потоків, зручну категоризацію витрат та наочну візуалізацію даних у вигляді графіків і діаграм. Проте, попри всі переваги, можливість прогнозування майбутніх витрат реалізована не на належному рівні. Цей процес вимагає значних зусиль і ручного втручання з боку користувача, хоча його цілком можна було б автоматизувати.

Проаналізуємо перелічені недоліки через таблицю 1.1, де можна побачити, що наявні сервіси уже зробили, а якого функціоналу їм бракує.

Таблиця 1.1 – Порівняльна таблиця переваг та недоліків наявних сервісів

Назва	Локалізація	Автоматичний облік	Персоналізація	Прогнозування
Mint	США	Так	Ні	Ні
YNAB	Глобально	Ні	Частково	Ні
Monobank	Україна	Так	Ні	Ні
Spendee	Глобально	Частково	Ні	Ні

Необхідно зазначити, що жодна система не пропонує функціоналу із прогнозування, що спонукає користувачів піддаватися своїм імпульсивним бажанням. За відсутності персоналізації, будь-яка трата вважається за одноразову, тобто людина не очікує схожих витрат наступного місяця, хоча, статистично, такі витрати перетворюються у регулярні, але користувач не має змоги це проаналізувати.

Таким чином, після ретельного аналізу наявних розв'язань запропонованої задачі, можна побачити, що жодна з наявних систем не пропонує жодного функціоналу щодо прогнозування витрат користувача, та майже жодна система не пропонує персоналізації у повному форматі. Виходячи з аналізу, можна побачити тенденцію, що автоматичний облік досягається лише через специфічну локалізацію системи.

1.3 Обґрунтування розробки власної системи

На основі ретельного аналізу наявних рішень, можна зробити висновок, що розробка нової автоматизованої системи, яка може задовольнити потреби сучасних користувачів, є перспективною задачею. Отже, основними недоліками, що запропонована система повинна вирішувати – це недостатня персоналізація, відсутність автоматизованого прогнозування та складність у використанні.

Складність використання – аспект, що дуже залежить від інтерфейсу додатка. Важливість UI та UX були визначені у дослідницькій роботі Юнхі Чон, де вона зазначає, що покращення UI/UX дизайну на основі потреб користувачів та підвищення доступності вебінтерфейсів суттєво підвищують коефіцієнт конверсії на e-commerce платформах завдяки зменшенню точок тертя та підвищенню задоволеності користувачів [2]. Попри те, що авторка аналізувала комерційну платформу, можна використати таку ж логіку для додатка керування фінансами. Простота і зручність у використанні будуть забезпечені завдяки адаптивному дизайну, мінімізації ручного введення та автоматизації більшості процесів.

Ще однією перевагою запропонованої системи, що виходячи з аналізу буде виділяти її серед інших, – персоналізація прогнозування витрат. Завдяки інтеграції сучасних алгоритмів машинного навчання, система матиме змогу застосовувати аналіз фінансових звичок та поведінку користувачів індивідуально. Це дозволить створювати більш точні та релевантні прогнози, що розв'язувати проблему з ефективним веденням особистого або сімейного бюджету з урахуванням непередбачених витрат.

Як результат із двох основних зобов’язань системи, що були перелічені, виходить третє – користувачі повинні мати змогу вести облік своїх даних та отримувати прогнози щодо своїх затрат настільки просто та зручно, щоб користувачі, що ніколи не використовували подібних додатків, могли зрозуміти, як його використовувати, без додаткової допомоги.

Отже, розробка автоматизованої системи керування особистими фінансами та інтеграція системи прогнозування на основі алгоритмів штучного навчання є актуальною задачею, що дозволить підвищити фінансову грамотність населення, забезпечити більш ефективне управління особистими або сімейними бюджетами та зробити процес фінансового планування більш точним і зручним.

1.4 Постановка задачі

Метою даної кваліфікаційної роботи є розробка автоматизованої системи керування особистими фінансами з інтеграцією модулю прогнозування витрат на основі методів машинного навчання та принципів UX/UI-дизайну. Очікується, що результати дослідження дозволять підвищити доступність, точність та ефективність фінансового планування для широкого кола користувачів.

В постановки задачі біло виділено такі завдання:

– здійснити огляд існуючих рішень у сфері додатків для обліку та аналізу особистих фінансів, виявити їх сильні та слабкі сторони;

– дослідити можливості застосування алгоритмів машинного навчання для персоналізованого прогнозування витрат на основі індивідуальної фінансової поведінки користувачів;

– розробити концепцію системи, яка мінімізує ручне введення даних, автоматизує ключові процеси та забезпечує адаптивність до потреб користувача;

– реалізувати систему з використанням обраних підходів до дизайну та алгоритмів прогнозування.

2 ТЕОРЕТИЧНІ ЗАСАДИ РОЗРОБКИ СИСТЕМИ

2.1 Теоретичні основи прогнозування за допомогою методів машинного навчання

Перед тим як перейти до аналізу методів прогнозування за допомогою машинного навчання, доцільно окреслити основні теоретичні терміни, що будуть систематично використовуватися у ході розробки запропонованої системи:

– фінансове прогнозування – процес передбачення майбутніх грошових потоків, витрат або доходів на основі історичних даних;

– модель прогнозування – окремий алгоритм, що маючи попередні спостереження, здатен робити оцінку майбутніх значень цільової змінної. У контексті даного дослідження, цільовою змінною є сума витрат;

– fullstack розробка (з англ. «розробка повного стаку») – підхід до веброзробки, що полягає у розробці та клієнтської частини (або фронт енд частини), і серверної частини (або бекенд частини).

У сучасному житті машинне навчання стало важливим інструментом розв'язання багатьох задач, зокрема прогнозування, завдяки його здатності виявляти складні взаємозв’язки у великих об’ємах даних. Основною ідеєю машинного навчання є побудова моделей, здатних «навчатися» на історичних даних, щоб робити точні прогнози для нових, невідомих раніше ситуацій

Є декілька різних підходів до навчання математичних моделей:

– навчання без вчителя (unsupervised learning);

– навчання з підкріпленням (reinforcement learning);

– навчання з вчителем (supervised learning).

Навчання без вчителя – найкращий вибір, коли є необхідність виявити приховані структури у даних без наявності міток або цільових змінних. У фінансовому контексті цей вид навчання може бути корисним для

кластеризації користувачів за схожими фінансовими звичками або для виявлення аномальних транзакцій. Наприклад, алгоритм K-Means був застосований американською фінансовою компанією MLQ для кластеризації компаній на основі рухів їхніх акцій протягом двох років, що дозволило групувати компанії з подібною динамікою цін [3].

Навчання з підкріпленням (RL) – підхід, що навчається приймати рішення через взаємодію із середовищем та отримання зворотного зв’язку у вигляді винагород. У сфері фінансів RL використовується з метою оптимізації торгових стратегій та управління портфелем. Згідно із дослідженням, сучасні алгоритми машинного навчання перевершують традиційні методи прогнозування, особливо в умовах високої волатильності та складних динамічних патернів [4].

Одним із найбільш поширених підходів у машинному навчанні є навчання з вчителем (SL). Цей підхід передбачає використання наборів даних, у яких вже відомі як вхідні змінні (features), так і вихідні змінні (labels або target variables). Модель «навчається», знаходячи відповідність між цими змінними, щоб потім застосовувати отримані закономірності для прогнозування майбутніх значень. У контексті фінансового прогнозування SL дає змогу моделі аналізувати попередні фінансові транзакції користувача, та на цій основі прогнозувати майбутні витрати.

У даному дослідженні акцент задачі переноситься на прогнозування часових рядів (time series regression), де історичні дані про фінансові транзакції використовуються для передбачення наступних витрат. Цей вид прогнозування має свої особливості: важливо враховувати послідовність спостережень та їхню залежність одне від одного. Тематика даної роботи вимагає надання особливої уваги таким залежностям, оскільки витрати користувача можуть мати сезонні, щомісячні або тижневі цикли, які модель повинна мати змогу розпізнавати та аналізувати.

Історичні транзакції користувачів є основним джерелом інформації для прогнозування майбутніх витрат. Модель має враховувати різноманітні

характеристики цих транзакцій, такі як категорії витрат, розмір витрат, час та регулярність здійснення платежів. Ці характеристики дозволяють побудувати ефективні персоналізовані прогнози.

Не можна забувати про нормалізацію та підготовку даних перед навчанням моделі. Задля забезпечення максимальної ефективності машинного навчання також необхідно враховувати особливості фінансових даних, таких як кількість нульових значень (дні без витрат) або різкі стрибки витрат, які можуть бути сезонними або зв’язані із непередбачуваними обставинами.

Перевагою SL є можливість вибору між різними типами алгоритмів, такими як лінійна регресія, дерева рішень, градієнтний бустинг або нейронні мережі. Вибір конкретного алгоритму буде залежати від специфіки задачі та характеристик наявних даних.

2.2 Вибір моделі прогнозування

Для ефективного прогнозування витрат необхідно обрати відповідну модель машинного навчання. Вибір моделі є ключовим етапом проєктування, бо від її можливостей залежить точність прогнозування, стабільність при використанні, швидкість навчання, а також масштабованість системи.

У контексті даної задачі було сформульовано наступні вимоги до моделі:

– здатність обробляти як числові, так і категоріальні ознаки – транзакції мають такі ознаки як «сума», але також мають унікальні ідентифікатори категорій витрат;

– підтримка нелінійних залежностей – поведінка користувача із фінансовими операціями зазвичай не є лінійною;

– стійкість до аномалій і пропущених значень;

– висока точність прогнозу.

Почнемо аналіз із найпростішого алгоритму – лінійна регресія, що визначається за формулою:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon, \quad (2.1)$$

де y – цільова змінна;

x_i – вхідні ознаки;

β_i – коефіцієнти регресії;

ϵ – залишок моделі.

Головними перевагами лінійної регресії вважаються простота реалізації, висока швидкість роботи та зрозумілість отриманих результатів. Однак, головним недоліком, що виходить навіть із назви цього алгоритму, є неможливість опрацювання даних з нелінійними зв'язками, що, ймовірно, негативно вплине на якість результатів під час роботи із даними фінансової природи походження.

Іншим популярним методом є дерево рішень. Його математичний опис робиться через розбиття множини ознак на підмножини за критерієм, що мінімізує ентропію або похибку:

$$Entropy\,(D) = -\sum_{i=1}^{m} p_i log_2(p_i), \quad (2.2)$$

де D – вибірка;

m – кількість унікальних класів у вибірці;

p_i – ймовірність належності випадкового елемента до класу i.

Дерева рішень добре інтерпретується, працюють з категоріальними даними та стійкі до аномалій, але часто страждають від перенавчання.

Нейронні мережі дозволяють моделювати складні нелінійні залежності, що може бути гарним вибором для моделей, що працюють із фінансовими системами. Головним недоліком моделі – є потреба у значних

ресурсах для тренування: об’ємні дата сети та машинні потужності, – іншим недоліком – є важкість інтерпретації результатів.

Ефективним сучасним підходом є метод градієнтного бустингу, який мінімізує похибку через ітеративне навчання ансамблю слабкіших моделей. Описується цей метод через формулами 2.3 та 2.4.

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x), \tag{2.3}$$

де $F_m(x)$ – поточна версія моделі;

$F_{m-1}(x)$ – ансамблева модель на попередньому кроці;

γ_m – коефіцієнт, який масштабує внесок нової моделі;

$h_m(x)$ – нова модель.

$$\gamma_m = argmin_\gamma \sum_{i=1}^{n} L\big(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\big), \tag{2.4}$$

де γ_m – оптимальне значення коефіцієнта для поточного кроку;

L – функція втрат;

y_i – реальне значення відповіді;

$F_{m-1}(x_i)$ – прогноз попередньої моделі для прикладу i;

$h_m(x_i)$ – прогноз нової базової моделі для прикладу i.

Градієнтний бустинг відзначається високою точністю прогнозів, здатністю до ефективного опрацювання великих наборів даних та хорошою роботою з категоріальними та числовими ознаками [5]. Популярними реалізаціями даного алгоритму, такі як XGBoost, LightGBM або CatBoost, широко використовуються під час розв'язання складних задач прогнозування у фінансовому секторі.

Після аналізу всіх вище перелічених методів навчання з учителем, для реалізації запропонованої системи було обрано алгоритм градієнтного бустингу – LightGBM. Обрана модель характеризується високою

ефективністю роботи з великими обсягами даних, швидкістю навчання та точністю прогнозів, особливо на складних фінансових наборах даних [6]. Отже, вибір методу градієнтного бустингу є оптимальним рішенням для реалізації поставленої задачі, що забезпечить високу якість прогнозування та зручність використання результатів моделі користувачами додатка.

2.3 Вимоги до моделі прогнозування

Запропонована система вимагає від моделі градієнтного бустингу можливості прогнозування частки витрат на певну категорію фінансових транзакцій (де категорія – група фінансових списань, об'єднаних спільною метою).

Формально, цільову змінну поставленої задачі можна визначити за формулою:

$$target = \frac{NextMonthCategory'sSpending}{NextMonthSheet'sSpending}, \quad (2.5)$$

де $Next\ Month\ Category's\ Spending$ – витрати у даній категорії наступного місяця;

$Next\ Month\ Sheet's\ Spending$ – витрати у даному листі наступного місяця.

Таким чином, проблема є задачею регресії, де необхідно передбачити значення у межаж від 0 до нескінченності.

Для ефективного прогнозування фінансових операцій необхідно визначитися із задачами моделі. У контексті запропонованої системи, модель машинного навчання повинна передбачати суму грошей, що користувач може витратити на одну якусь категорію витрат. З цього випливає, що модель повинна вміти:

– аналізувати вплив минулих значень та їх вплив на поточні;

– захоплювати тренди, сезонність або мінливість;

– порівнювати значення між собою в межах одного часового періоду;

– розпізнавати закономірності пов’язані із часом.

Виходячи з вимог до моделі були визначені типи ознак, що необхідно використати під час тренування моделі.

Для врахування впливу минулих значень на поточні було використано лагові ознаки. Лагові ознаки – це значення змінної, зміщені назад у часі на певну кількість періодів (лагів).

Для захоплення трендів було використано два види ознак: ковзні статистики для короткострокових тенденцій та експоненційне згладжування для довгострокових. Перший тип ознак можна представити формулою:

$$RollingMean(x)_t = \frac{1}{\omega} \sum_{i=t-\omega+1}^{t} x_i \,, \tag{2.6}$$

де t – поточний момент часу, для якого обчислюється ковзне середнє;

ω – розмір ковзового вікна;

x_i – значення змінної x у період часу i.

Також важливою є ковзне стандартне відхилення, що оцінює варіабельність статистики та допомагає вловлювати не тільки рівень, але й стабільність витрат у часі.

Експоненційного згладжування надає змогу «забувати» застарілі дані та надавати значення новішим. Математично описати цей вид ознак можна наступним чином:

$$ExpMean(x)_t = \alpha \cdot x_t + (1 - \alpha) \cdot ExpMean(x)_{t-1}, \tag{2.7}$$

де t – поточний момент часу;

$\alpha \in (0, 1)$ – коефіцієнт згладжування;

x_i – значення змінної x у період часу i.

Були додано агрегати на основі середніх витрат по листах та категоріях, що дозволяє виявляти «загальну поведінку» користувача. Цей тип ознак класифікується як «крос-секційна ознака».

На відміну від спеціалізованих моделей (таких як Arima чи LSTM), LightGBM не вміє працювати із часовими рядами. Вона очікує, що ознаки кожного прикладу вже містять усю інформацію про залежності. Щоб модель могла передбачити майбутні значення, дата сет повинен зберігати порядок спостережень (тобто не можна змішувати випадковим чином записи) та перетворити часову залежність у вигляді числових характеристик через вище описані ознаки. Саме якість побудованих ознак визначає наскільки добре модель «розуміє», що дані мають часову структуру.

2.4 Теоретичні основи вебтехнологій

Розробка вебдодатків – одна із найконкурентніших сфер у розробці IT-продуктів, але й у той самий час, одна із найскладніших, бо є багато різних браузерів та різного розміру пристроїв, що повинні однаково гарно відображати один і той самий додаток [7]. Окрім цього, завжди необхідно враховувати потужності пристроїв та швидкість інтернет-з’єднання. Саме з цих причин під час розробки сучасних вебдодатків важливо використовувати технології, що забезпечують високу ефективність, зручність підтримки та масштабованість.

На цей час є декілька різних бібліотек та фреймворків, що дозволяють легше писати легкий у підтримці та ефективний код. Найпопулярнішими з них вважаються: React (створений командою компанії Meta), Vue (створений вільним автором) та Angular (створений командою компанії Google). Хоча немає однозначного вибору щодо кращого фреймворку,

згідно із Stack Overflow Developer Survey, React – найпопулярніша бібліотека серед SPA (Single Page Application).

Single Page Application – додатки, що працюють лише на одному JavaScript, мають 3 недоліки:

– швидкість першої завантажування вебсторінки – браузеру необхідно завантажити весь JavaScript файл, що може досягати декількох мегабайтів;

– через те, що SPA мають один HTML файл, браузер не може правильно розпізнавати навігацію за сайтом;

– оптимізація для пошукових систем – набагато складніше налаштувати додаток для гарної SEO (Search Engine Optimization).

Кожний із перелічених недолік має своє окреме рішення, але гарною практикою буде використовувати фреймворк Next.js, що побудований на React, але із підтримкою серверного рендерингу, що вирішує всі 3 недоліки чистого React. Крім того, ця технологія має вбудовану підтримку TypeScript, що значно спрощує процес розробки та підвищує надійність додатка.

Наступним важливим рішенням є вибір бази даних. Додатки розроблені для роботи із фінансовою сферою, як правило, мають добре структуровані дані, тож реляційні бази даних будуть кращим вибором за нереляційні. Однією із найпопулярніших реляційних баз даних із відкритим кодом та великою підтримкою суспільства розробників – PostgreSQL, тож було обрано використовувати базу даних – Supabase, що побудована на основі PostgreSQL та надає також функціонал для авторизації та зберігання файлів, а також має корисну панель розробника для простішої розробки та менеджменту даними.

Щодо серверної частини додатка, ми можемо поділити логіку додатка на 2 частини: опрацювання бази даних та робота із моделлю машинного навчання. За першу частину у нас відповідає Next.js та його серверні компоненти. Однак найбільш популярні бібліотеки для роботи із машинним навчанням отримала мова програмування Python, тож кращим вибором – є

написання ML логіки окремим сервером за допомогою фреймворку FastAPI мовою Python.

Комбінація Next.js, Supabase та FastAPI дозволяє створити повноцінний вебдодаток із чітким розподілом обов'язків між клієнтською частиною, серверною логікою та базою даних. Такий підхід забезпечує максимальну продуктивність, зручність розробки та простоту підтримки, що є критично важливими для системи управління фінансами, де важлива як надійність, так і швидкість роботи.

2.5 Архітектура повного стека

Висока продуктивність додатка – пріоритетна характеристика системи, задля досягнення якої необхідно обрати правильну архітектуру повного стека. В сучасних вебдодатках найчастіше використовується клієнт-серверна модель, де чітко визначено ролі та взаємодію між фронтендом, бекендом та базою даних [8].

Запропонована архітектура передбачає, що клієнтська частина відповідатиме за взаємодію з користувачем, обробку і відображення отриманих даних. Серверна частина забезпечує логіку додатка, обробку запитів клієнта, доступи та взаємодію із базами даних, та інтеграцію із зовнішніми сервісами, включаючи машинне навчання.

Клієнто-серверна архітектура використовує REST API (Representational State Transfer Application Programming Interface) як основний засіб взаємодії між фронтендом і бекендом. Цей підхід передбачає використання базових методів HTTP для обміну даних: GET, POST, PUT, PATCH, DELETE, – що дозволяє простіше організовувати взаємодію між частинами додатка. REST API також забезпечує прозорість, зрозумілість структури запитів та відповідей, що значно спрощує процес розробки, тестування та підтримки сервісу [9].

Запропонована архітектура додатка передбачає чітке розділення шарів на фронт енд, бекенд і модель машинного навчання. Такий поділ дозволяє більш ефективно організувати процес розробки, підтримки та масштабування додатка, а також зменшити вірогідність з’явлення помилок.

Фронтенд-частина, що відповідає за інтерфейс користувача та написана за допомогою фреймворку Next, забезпечує швидке завантаження та інтерактивність. Бекенд-додаток, серверна частина додатка написаного на Next, забезпечує виконання бізнес-логіки та інтеграцію з базою даних Supabase. Окремою частиною є ML API, написаний за допомогою Python фреймворку – Fast API, що дозволяє легше працювати з моделлю машинного навчання. Вибір даного фреймворку зумовлений швидкістю та високою продуктивністю, що можна досягти за допомогою асинхронних операцій, що FastAPI підтримує.

На етапі проєктування БД було застосовано принципи реляційної нормалізації для забезпечення логічної цілісності, усунення надлишковості та підвищення ефективності обробки запитів. В результаті було виділено чотири ключові сутності: користувач, лист, транзакція та категорія транзакції. Як уже було зазначено, система використовує БД Supabase, що уже надає необхідне API для роботи з авторизацією, тож таблиця користувачів вже була готова для роботи. Інші таблиці необхідно було проєктувати самостійно.

Усі таблиці спроєктованої бази даних відповідають першій нормальній формі, оскільки не містять повторюваних груп або множин. Кожне поле містить атомарне значення.

Було усунено часткові залежності: усі атрибути кожної таблиці повністю залежать від первинного ключа, а зв’язки між таблицями реалізовані за допомогою зовнішніх ключів, що дозволило чітко відокремити логічні одиниці даних та уникнути надмірного дублювання.

Схеми таблиць указано на рисунку 2.1. Слід зазначити, що таблиця користувачів, реалізована Supabase, має набагато більше стовпців, але для даної системи на даному етапі вони значення не мають.

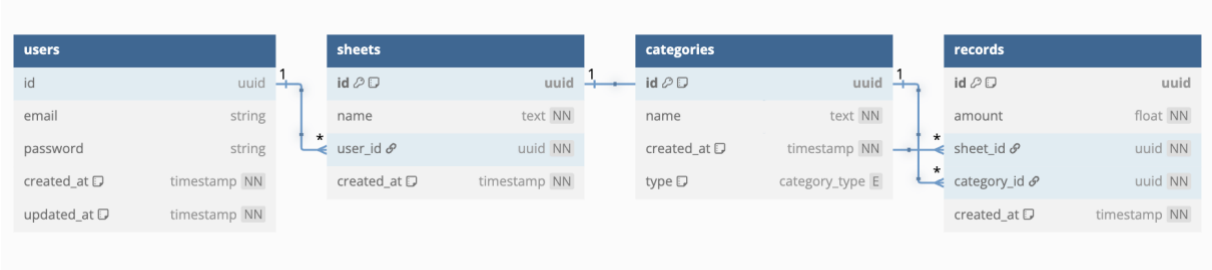

Рисунок 2.1 – Схема бази даних

Іншою важливою характеристикою дата сету – мати достатню кількість ознак, щоб модель могла виконувати прогнозування коректно. Модель повинна працювати із даними фінансової природи, тож повинна мати зазначені у розділі 2.3 ознаки:

– експоненційне згладжування;
– лагові ознаки;
– ковзні середні;
– часові ряди.

Отже, виходячи зі структури бази даних, ми можемо створити дата сет зі структурою описаною у таблиці 2.1, але попередньо ми повинні згрупувати всі транзакції за трьома характеристиками: лист, категорія та дата (рік та місяць).

Таблиця 2.1 – Структура датасету для моделі навчання LightGBM

Назва ознаки	Тип даних	Опис ознаки
sheet_id	UUID (Universally Unique Identifier)	Унікальний ідентифікатор листа, якому належить конкретна категорія

Продовження таблиці 2.1

Назва ознаки	Тип даних	Опис ознаки
category_id	UUID (Universally Unique Identifier)	Унікальний ідентифікатор категорії
year_month	String (формату «{рік}-{місяць}»)	Рік та місяць
total_spent_in_sheet_last_month	Float	Витрати у листі за попередній місяць
total_spent_in_sheet_last_3_months	Float	Витрати у листі за попередні 3 місяці
average_spent_in_sheet_last_6_months	Float	Витрати у листі за попередні 6 місяців
sheet_spending_trend	Int (-1 |0 | 1)	Тенденція на зміну у витратах, де -1 значить, що витрати зменшилися попередньо із минулими місяцями, 1 – збільшились, а 0 – залишились на тому ж рівні
total_spent_in_category_last_month	Float	Витрати у категорії за попередній місяць
total_spent_in_category_last_3_months	Float	Витрати у категорії за попередні 3 місяці
average_spent_in_category_last_6_months	Float	Витрати у категорії за попередні 6 місяців
category_spending_trend	Int (-1 | 0 | 1)	Тенденція на зміну у витратах, де -1 значить, що витрати зменшилися попередньо із минулими місяцями, 1 – збільшились, а 0 – залишились на тому ж рівні

Продовження таблиці 2.1

Назва ознаки	Тип даних	Опис ознаки
category_spent_percentage_last_month	Float	Процентне відношення витрат у категорії до витрат у листі загалом
category_spent_percentage_trend	Float	Тенденція на зміну у витратах, де -1 значить, що витрати зменшилися попередньо із минулими місяцями, 1 – збільшились, а 0 – залишились на тому ж рівні
most_frequent_spending_category_in_sheet	UUID (Universally Unique Identifier)	Категорія витрат у листі, яка зустрічається найчастіше
most_expensive_category_last_month_in_sheet	UUID (Universally Unique Identifier)	Категорія витрат, що виявилася найдорожчою
least_expensive_category_last_month_in_sheet	UUID (Universally Unique Identifier)	Категорія витрат, що виявилася найдешевшою

Таким чином, була спроєктована архітектура для повноцінної роботи системи фінансового прогнозування із розділенням на 3 основні компоненти: клієнтська частина, серверна частина та серверна частина моделі прогнозування.

3 ІМПЛЕМЕНТАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ

3.1 Архітектура системи

Перед початком розробки будь-якої складної системи необхідно створити її загальну архітектурну схему, яка буде описувати основні компоненти системи, їх роль та взаємодію один між одним. Правильна архітектура дозволить розробити додаток, що буде легко розширюватися та підтримуватися, а також забезпечить надійність розроблюваного програмного забезпечення.

Запропонована система, як уже було зазначено, складається із трьох основних частин:

– клієнтська частина (Frontend);
– серверна частина (Backend);
– сервер машинного навчання (ML API).

Клієнтська частина реалізована за допомогою фреймворка Next.js. Вона відповідає за інтерфейс користувача, за обробку взаємодій із користувачем, а також введених їм даних. Фронтенд також займається відправленням запитів до бекенду через REST API, на відповідь у які, генерує таблиці, графіки та забезпечує введення нових записів витрат та доходів.

Серверна частина реалізована у рамках фреймворку Next.js за допомогою серверних компонентів. Сервер обробляє HTTP-запити, відповідає за аутентифікацію користувача за допомогою Supabase, здійснює взаємодію з базою даних PostgreSQL та перенаправляє запити на сервер машинного навчання у випадках, коли потрібне прогнозування майбутніх витрат [10].

Сервер машинного навчання розроблений на базі фреймворку FastAPI та за допомогою мови програмування Python. Цей сервер має лише одну задачу – залежно від зібраних даних користувача, надавати прогноз щодо

затрат користувача на наступний період часу. Логіка прогнозування засновується на моделі машинного навчання (LightGBM). Сервер працює незалежно від основного, що дозволяє досягти кращої масштабованості обробки запитів на прогнозування за потребою.

Взаємодія між усіма трьома компонентами (рисунок 3.1) відбувається методом HTTP запитів між собою.

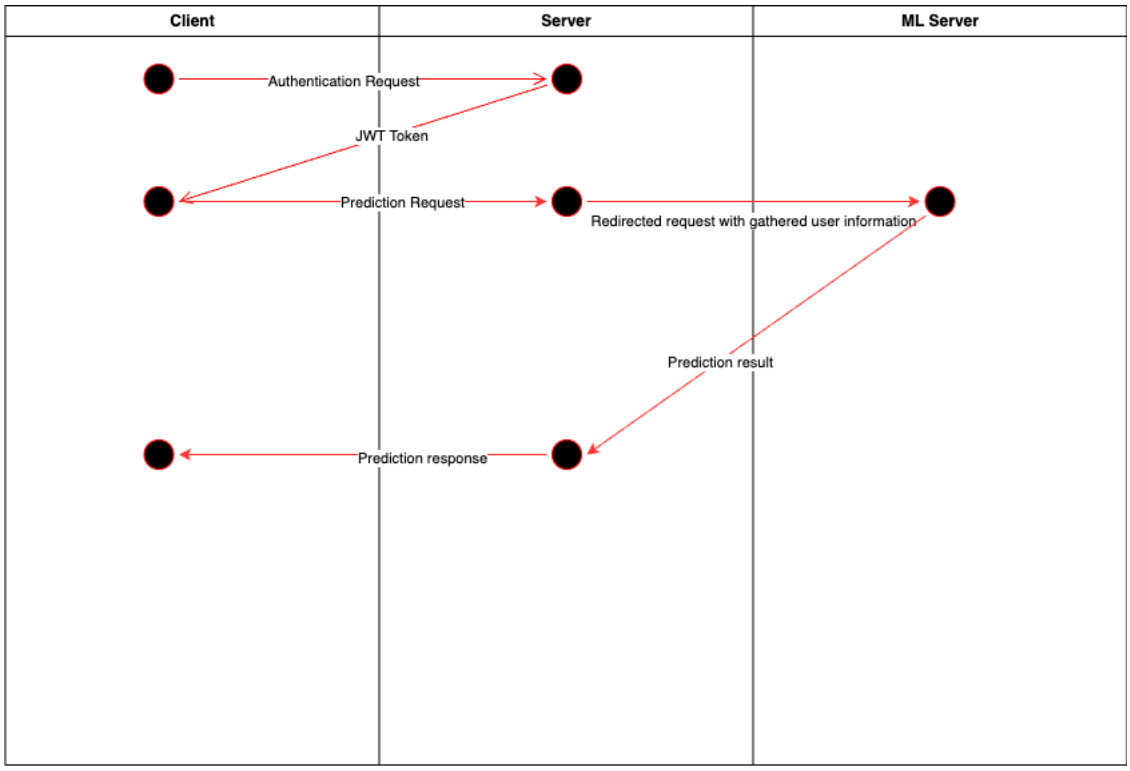

Рисунок 3.1 – Схема взаємодії трьох компонентів

На прикладі запиту на отримання прогнозу витрат користувача на наступний проміжок часу, розглянемо базово алгоритм дій:

– клієнт робить запрос на аутентифікацію, щоб отримати доступ до додатку;

– після успішного входу у сервіс, користувач робить запит на отримання прогнозу на наступний проміжок часу до серверу Next.js;

– серверні компоненті збирають усі необхідні для прогнозу дані, та надсилає запит до серверу машинного навчання;

– сервер машинного навчання надає прогноз за допомогою моделі LightGBM та повертає відповідь;

– основний сервер обробляє відповідь та надсилає її до користувача;

– користувач відрисовує відповідь.

Такий розподіл забезпечує модульність системи, що спрощує її розробку та підтримку, та надає можливість незалежно масштабувати кожен із компонентів відповідно до навантаження.

3.2 Реалізація клієнтської частини

Клієнтська частина реалізована із використанням фреймворку Next.js, що використовує бібліотеку React та дозволяє поєднати її з перевагами серверного рендерингу. Під час розробки додатків на запропонованому фреймворку важливо правильно визначити які компоненти мають бути клієнтськими, а які серверними. Визначення типу компонентів можна зробити після порівняння їх властивостей та можливостей наведених у таблиці 3.1.

Таблиця 3.1 – Порівняння клієнтських компонентів та серверних

	Серверний комп’ютер	Клієнтський комп’ютер
Мета	Відмальовування не інтерактивного інтерфейсу, збір даних або важкі обчислювальні операції	Інтерактивний інтерфейс (кнопки, форми ітд.)
Розмір	Немає, бо не являється частиною бандлу	Залежить від компоненту
Доступ до серверних елементів (наприклад до БД)	Так	Ні
Відмальовування	Подається як HTML документ	Інтерактивний JavaScript

Отже, необхідно розглянути, якого типу компоненти необхідні для реалізації запропонованої системи:

– макетні компоненти (від англ. «Layout») – у Next.js слугує як файл для спільної логіки між компонентами одного направлення;

– сторінки;

– інтерактивні компоненти (наприклад форми, кнопки, поля для вводу тощо);

– компоненти, що завантажують інформацію із серверу;

– статичні компоненти – компоненти, що не мають важкої логіки, а лише тільки відображають інформацію.

Ідеальне рішення мало би за мету знайти компроміс між використанням серверних компонентів та клієнтських через те, що використання тільки жодного типу компонентів має багато своїх недоліків.

Задля досягнення оптимізованого рішення було вирішено всі макетні та сторінкові компоненти зробити серверними, щоб користувач мав змогу бачити інтерактивні елементи замість білого екрану браузеру, поки інформація завантажується.

Клієнтськими компонентами було зроблено ті елементи, що повинні мати взаємодію з користувачем: форми, кнопки, поля для вводу тощо.

Окремої уваги заслуговують компоненти, що не є інтерактивними, але завантажують інформацію. Такі компоненти однаково можуть бути зроблені як серверними, так і клієнтськими. Тим не менш, було прийнято рішення робити їх клієнтськими з наступних причин:

– користувач швидше побачить стан прогрузки, бо не має важких операцій на серверній частині;

– простіше оновлення завантажених даних (для прикладу візьмемо список фінансових операцій користувача. Коли користувач додає новий запис до списку, серверні компоненти повинні бути оновлені повністю, що робить сторінку менш інтерактивною; клієнтські компоненти

дозволять оновити стан списку не блокуючи взаємодію із вже намальованими компонентами).

Важливо розуміти до яких функцій користувачі можуть мати дозвіл. Однією із поставлених цілей даного додатку – є зручний UX. Гарний досвід має за мету надати користувачеві зручний інтерфейс та можливості використовувати функції сервісу без додаткових дій. Один із важливих способів досягнення поставленої мети – є зручна навігація. Користувач повинен мати змогу використовувати браузерні функції історії та навігації під час знаходження у додатку. Згідно зі схемою на рисунку 3.2, було реалізовано 6 сторінок.

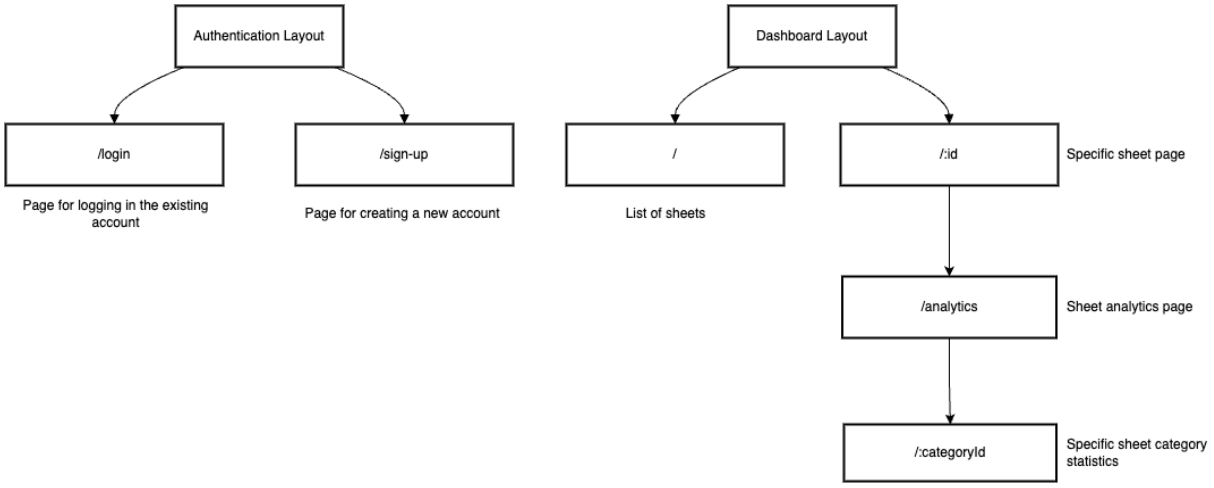

Рисунок 3.2 – Схема навігації додатка

Сторінки поділені на 2 типи: публічні та ті, що потребують авторизації. Усі сторінки «Authentication Layout» дають доступ неавторизованим користувачам, сторінки «Dashboard Layout» навпаки – дають доступ тільки за умови успішної авторизації.

Сторінки так званого «Dashboard» мають достатньо просту мету – одна сторінка відповідає за перелік усіх листів («Sheet») користувача, а інша дає змогу переглядати фінансові операції та модифікувати їх. Існує ще дві сторінки:

– /analytics – дає змогу користувачеві переглядати список існуючих категорій;

– /:categoryId – дає змогу подивитись у графічній формі співвідношення затрат та доходів користувача щомісяця та отримати прогноз на наступний за допомогою ML.

Важливим аспектом запропонованої системи – є безпека. Так як додаток пов’язаний із фінансами користувачів, не можна ставити під збереження даних під сумнів.

Неможливість для неавторизованих користувачів доступитися до даних інших людей – важлива складова безпеки додатку. Система авторизації полягається на сервіс Supabase, що пропонує API для готової та безпечної аутентифікації користувача. Supabase Authentication API бере відповідальність за збереження даних користувача та видавання JWT токенів, тож на стороні фронтенду залишається питання вирішення доступу до інформації. Рішення полягає у 2 кроках:

– використання Next.js middleware та модифікація middleware, що пропонує Supabase;

– перевірка на авторизацію у Dashboard Layout.

Middleware у Next.js мають більше навігаційну функцію, тобто покладатися лише на авторизаційні middleware – небезпечно (наприклад, коли неавторизований користувач робить запит на захищену сторінку, middleware зробить переадресування запиту на публічну сторінку, але об’єкт запиту на сторінку може бути модифікований зловмисниками, і middleware не можуть це помітити). Тож, найбезпечнішим методом захисту вважається перевірка на авторизацію вже під час рендерингу сторінки. Оскільки компоненти макетів (Layout) відпрацьовують кожний раз, коли був зроблений запит на сторінку, це є гарним місцем для верифікації на авторизацію.

Отже, запропонована система має надійну клієнтську частину, що дозволяє робити запити та оновлювати дані у зручний для користувача

спосіб, що не викликає надлишкових запитів та вимальовує актуальний стан додатка без перезавантажень сторінок. Також була досягнута надійна система навігації із публічними та захищеними сторінками за допомогою Next.js middleware та макетних компонентів.

3.3 Реалізація серверного застосунку (Next.js)

Серверна частина додатка реалізована на базі Next.js та серверних компонентів. Основною задачею серверної частини є обробка бізнес-логіки системи, взаємодія із базою даних Supabase та забезпечення авторизації.

Як вже було зазначено у розділі 3.2, надійна авторизація була досягнута за допомогою функціоналу, що пропонує Supabase Authentication. Тож, усе ще залишається серверним компонентам – перевіряти чи авторизований користувач при кожному запиті. Функція подана у лістингу 3.1 була реалізована спеціально із метою перевірки стану користувача при кожному запиті.

Лістинг 3.1 – Програмний код функції отримання стану авторизації користувача

```
export const getServerUser = async (): Promise<User> => {
  const supabase = await createClient();
  const {
    data: { user },
    error,
  } = await supabase.auth.getUser();

  if (error || !user) {
    redirect("/login");
  }

  return user;
};
```

Уся взаємодія із базою даних була реалізована за допомогою ORM (Object-relational mapping) Drizzle ORM. Це дозволяє підтримувати та масштабувати додаток у майбутньому легше. У випадку, якщо PostgreSQL буде замінена на іншу СУБД, модифікація системи буде полягати лише у змінні визначення схеми таблиць бази даних у вигляді функцій відповідних бібліотек, що пропонує використана ORM.

Важливою частиною роботи із СУБД – безпека доступу до даних, тобто тільки користувач із необхідним доступом повинен мати змогу отримувати відповідні дані із бази даних. Описана мета може бути досягнута як на стороні серверного додатка (у цьому випадку серверні функції Next.js можуть забезпечити це), так і на стороні СУБД (у такому випадку Supabase з PostgreSQL). Гарною практикою, що дозволить досягти найбільших показників безпеки даних, є поєднання двох методів.

Серверні функції виконують поставлену задачу завдяки попередньої авторизації, повторним перевіркам на авторизацію користувача перед запитами до СУБД через функцію зазначену у лістингу 3.1 «getServerUser», а також повноцінними фільтраційними умовами під час запитами (наприклад, як зазначено на рисунку 3.3, таблиця листів має поле «user_id», тож під час виконання запиту на отримання списку листів, відповідна функція отримує унікальний ідентифікатор авторизованого користувача та передає його як фільтраційну умову під час запиту).

Однак, використання лише тільки вище описаного методу забезпечення безпеки даних може бути не ефективним. Наприклад, візьмемо таблицю «records», яка згідно зі схемою описаною на рисунку 2.1, немає жодної прямої залежності до користувача, тому щоб зробити перевірку на користувача, необхідно виконувати операцію «JOIN» під час запита до СУБД, що може бути менше ефективним через необхідність робити додаткові операції на стороні бази даних. Розв'язком проблеми є використання RLS («Row Level Security» – з англійської «Безпека на рівні рядку»). RLS – додатковий фільтр, що можна додати до таблиці бази даних.

Під час проєктування таблиць СУБД, було визначено наступні правила для таблиці «sheets»:

– видаляти записи можуть тільки власники цих записів;

– переглядати записи можуть тільки власники цих записів;

– користувач може створювати нові записи можуть тільки для того ж користувача.

Для таблиць «records» та «categories» були створені аналогічні правила, але додано ще одно: оновлювати записи можуть тільки власники цих записів.

Бізнес-логіка додатка потребує декілька CRUD («Create-Read-Update-Delete») серверних функцій. Тобто, серверні компоненти повинні мати можливість створювати, оновлювати, видаляти та зчитувати інформацію о фінансових операціях користувача. Логіка роботи з транзакціями була досягнута за допомогою так званих «server action» (з англ. «серверних дій») – функцій, що виконуються на сервері Next.js додатку та викликаються або із серверних компонентів, або із клієнтських (але тоді, вони оброблюються як HTTP запити).

3.4 Інтеграція моделі машинного навчання

Інтеграція машинного навчання до запропонованої системи – один із ключових етапів, що дозволяє здійснювати прогнозування майбутніх витрат користувачеві.

3.4.1 Тренування моделі LightGBM

Забезпечення високої якості прогнозування витрат – основна функція даної моделі, для виконання якої важливим є врахування часових залежностей у даних [11]. У межах розробки алгоритму тренування було реалізовано низку специфічних ознак для часових рядів.

Лагові ознаки – значення певних змінних за попередні періоди, які дозволяють моделі враховувати історичну динаміку витрат. Під час тренування моделі були створені лагові ознаки витрат за минулі 1, 2, 3, 6 та 12 місяців як на рівні окремих категорій витрат, так і на загальному рівні листа.

Імплементація лагових ознак, як можна побачити на лістингу 3.4, була реалізована за допомогою зсуву відповідних часових рядів за допомогою функцією групування.

Лістинг 3.4 – Функція створення лагових ознак

```
    def create_lag_features(df: pd.DataFrame, lag_periods:
List[int]) -> pd.DataFrame:
        for lag in lag_periods:
            df[f'category_spent_lag_{lag}'] =
df.groupby(['sheet_id', 'category_id'])[
                'total_spent'].shift(lag)
            df[f'sheet_spent_lag_{lag}'] = df.groupby(

['sheet_id'])['total_spent_in_sheet_last_month'].shift(lag)
        return df
```

Наступним кроком було додавання ковзних середніх, що згладжують довгострокові коливання та дають змогу отримати більш стабільну оцінку довгострокової тенденції. Розрахунок середніх було здійснено згідно з формулою 2.4 та розраховано з вікнами на 3, 6 та 12 місяців для категорій та загально листів. З метою досягти простого у написанні, підтримки та функціонально надійного та коректного коду, було використано бібліотеку pandas, що є Python бібліотекою, що надає ряд уже реалізованих математичних функцій. Програмний код описаної функції подано у лістингу 3.5.

Лістинг 3.5 – Програмний код розрахунку середніх ковзних для набору даних

```
    def create_rolling_statistics(df: pd.DataFrame, windows: List[int]) -> pd.DataFrame:
        for window_size in windows:
            df[f'category_rolling_mean_{window_size}m'] = df.groupby(['sheet_id', 'category_id'])['total_spent'].transform(
                lambda x: x.rolling(window_size, min_periods=1).mean()
            )
            df[f'category_rolling_std_{window_size}m'] = df.groupby(['sheet_id', 'category_id'])['total_spent'].transform(
                lambda x: x.rolling(window_size, min_periods=1).std()
            )
            df[f'sheet_rolling_mean_{window_size}m'] = df.groupby(['sheet_id'])['total_spent_in_sheet_last_month'].transform(
                lambda x: x.rolling(window_size, min_periods=1).mean()
            )
        return df
```

Важливим кроком перед налаштуванням роботи із часовими рядами стало експоненційне згладжування, що надає більшу вагу останнім записам, а меншу – старішим. Налаштування даної ознаки потребувало коректних коефіцієнтів задля кращого згладжування, тож були обрані коефіцієнти 0.2, 0.4 та 0.6.

Перед останнім кроком стало налаштування крос-секційних ознак, що як можна побачити у лістингу 3.6, вираховує загальні середні значення по

листу та категоріям, а також середні значення у заданих вікнах, що у цьому випадку були 3, 6 та 12 місяців.

Лістинг 3.6 – Програмний код вирахування крос-секційних ознак

```
    def create_cross_cutting_features(df: pd.DataFrame,
windows: List[int]) -> pd.DataFrame:
        df['sheet_category_avg'] = df.groupby(['sheet_id',
'year_month'])[
            'total_spent'].transform('mean')

        df['category_sheet_avg'] = df.groupby(['category_id',
'year_month'])[
            'total_spent'].transform('mean')

        for window_size in windows:
            df[f'sheet_category_rolling_avg_{window_size}m'] =
df.groupby('sheet_id')['sheet_category_avg'].transform(
                lambda x: x.rolling(window_size,
min_periods=1).mean()
            )

            df[f'category_sheet_rolling_avg_{window_size}m'] =
df.groupby('category_id')['category_sheet_avg'].transform(
                lambda x: x.rolling(window_size,
min_periods=1).mean()
            )

        return df
```

Нарешті, фінальним кроком було задання часових ознак. Завдяки уже налаштованим ознакам, достатньо було додати ті часові ознаки, що є важливими для фінансових транзакцій. Оскільки модель повинна прогнозувати на наступний місяць, необхідно було обрати часові ознаки

таким чином, щоб вони могли описувати природу грошової операції, тому були обрані наступні ознаки:

- рік;
- місяць;
- фінансовий квартал.

Важливою частиною створення моделі навчання – є інтерфейс вхідних даних, з яких модель може робити прогноз. Дані, що будуть отримані на вхід, повинні бути достатньо легко сформовані сервером Next.js та БД. Після ретельного аналізу та практичного виявлення найефективніших ознак, було створено список із наступних ознак:

- витрати по листу;
- витрати по листу за останній місяць;
- витрати по листу за останні 3 місяці;
- витрати по листу за останні 6 місяців;
- тенденція листу на збільшення/зменшення;
- витрати по категорії за останній місяць;
- витрати по категорії за останні 3 місяці;
- витрати по категорії за останні 6 місяців;
- тенденція категорії на збільшення/зменшення;
- процентне співвідношення витрат категорії до витрат по листу за останній місяць;
- тенденція категорії у процентному співвідношенні до збільшення/зменшення.

3.4.2 Налаштування ML API

Як уже було зазначено під час створення архітектури системи, для створення інтерфейсу для комунікації між сервером Next.js та моделлю, було зроблено рішення використати Python фреймворк Fast API.

Задачею заданого сервера є отримання указаних у розділі 3.4.2 вхідних до моделі даних та виконання основної функції моделі – прогнозування. Для цього був створений ендпоінт /predict із методом POST, що очікує від запиту даних формату вказаного у лістингу 3.7 у тілі запита та перевіряє ці дані.

Лістинг 3.7 – Код класу для валідації тіла запита на прогнозування

```
class PredictionInput(BaseModel):
    total_spent: float
    total_spent_in_sheet_last_month: float
    total_spent_in_sheet_last_3_months: float
    average_spent_in_sheet_last_6_months: float
    sheet_spending_trend: float
    total_spent_in_category_last_month: float
    total_spent_in_category_last_3_months: float
    average_spent_in_category_last_6_months: float
    category_spending_trend: float
    category_spent_percentage_last_month: float
    category_spent_percentage_trend: float
```

На запуск додатку, сервер завантажує до пам’яті модель, що потім виконується під час роботі ендпоінту. Код функції прогнозування наведено у лістингу 3.8.

Лістинг 3.8 – Програмний код функції-оброботчика ендпоінту /predict

```
@app.post("/predict")
def predict(input: PredictionInput):
    features = np.array([[
        input.total_spent,
        input.total_spent_in_sheet_last_month,
        input.total_spent_in_sheet_last_3_months,
        input.average_spent_in_sheet_last_6_months,
        input.sheet_spending_trend,
```

Продовження лістингу 3.8.

```
            input.total_spent_in_category_last_month,
            input.total_spent_in_category_last_3_months,
            input.average_spent_in_category_last_6_months,
            input.category_spending_trend,
            input.category_spent_percentage_last_month,
            input.category_spent_percentage_trend,
        ]])

        prediction_percentage = booster.predict(features)[0]
        prediction = input.total_spent_in_category_last_month
* prediction_percentage
        return {"predicted_next_month_spent":
round(float(prediction), 2),
"predicted_next_month_spent_percentage":
round(float(prediction_percentage), 2)}
```

Модель була натренована передбачати процентне відношення заради кращої роботи із роботи із «шумом». Тренувальний датасет містив рядки із екстремально високими значеннями, але на фоні більш здержаних записів, вони виявились несуттєвими. Передбачення процентного відношення дозволяє краще працювати із випадками, коли користувач має незвично високі для моделі значення.

З погляду проєктування серверів, надзвичайно важливо мати надійні механізми для постійного відстежування стану серверу. Це дозволяє своєчасно виявляти будь-які збої або критичні помилки, які можуть негативно вплинути на роботу системи в цілому. Наявність таких механізмів допомагає уникнути тривалих простоїв, оперативно реагувати на несправності та підтримувати стабільність і надійність сервісу для користувачів. Саме з цією метою було розроблено спеціальний ендпоінт для перевірки «здоров’я» серверу – GET /health.

3.4.3 Інтеграція до сервера Next.js

Архітектура комунікації двох або більше серверів – важливе питання, яке підіймається при розробці багатомовних серверних додатків. Існує 2 варіанти комунікації між серверами:

– HTTP запити, наприклад, стандартні запити чи технології, що використовують HTTP у своїй реалізації, як-от gRPC;

– event-driven комунікація.

Обидва підходи мають свої переваги та недоліки. У додатках великих масштабів заведено використовувати event-driven варіант комунікації, але це потребує налаштування додаткових сервісів як: Redis, RabbitMQ, Kafka чи інші; цей варіант також біль важкий не тільки у реалізації, а й у відновленні помилок теж. HTTP запити, з іншої сторони, простіше у реалізації, але працюють повільніше.

Для системи було обрано використовувати метод HTTP запитів. Next.js та Fast API сервери не потребують важкої логіки комунікації, та швидкість запиту не є принциповим питанням у такому випадку.

Отже, було розроблено серверну функцію на Next.js, що робить запит до СУБД, зчитує та агрегує інформацію необхідну для прогнозування та запитує Fast API сервер через метод fetch.

4 ТЕСТУВАННЯ І ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 Методологія тестування

Результатом розробленої системи є веб-додаток, яким можуть користуватися будь-які користувачі в інтернеті, тому важливо провести тестування усіх компонентів системи, щоб зменшити кількість помилок під час її роботи та покращити досвід користувача під час користування системою [12].

Тестування сервісу було поділено на 4 частини:

– ручне тестування – базова перевірка працездатності додатку;

– модульне тестування – перевірка окремих функцій;

– інтеграційне тестування – перевірка взаємодії клієнтської частини із серверною;

– end-to-end тестування (E2E) – перевірка системи шляхом моделювання реальних сценаріїв використання додатку.

Під час тестування особлива увага приділялася наступним аспектам:

– валідація форм;

– безперервність навігації між сторінками додатку;

– стабільність отримання прогнозу витрат;

– стабільність роботи при великій кількості даних у БД.

4.1.1 Тестування візуального інтерфейсу

Найважливішим елементом розробляємої системи – є візуальний інтерфейс. Це той компонент системи, що бачать користувачі, та з яким вони взаємодіють. Неважливо, якщо серверна логіка та модель прогнозування працюють ідеально, якщо користувачі не можуть достатньо просто це використовувати. Як уже було зазначено, одним із методів тестування було – ручне тестування. Воно ж було зроблено першим. Як можна

побачити на рисунку 4.1, форма авторизації дає зрозуміти користувачу, що введені їм дані – некоректні.

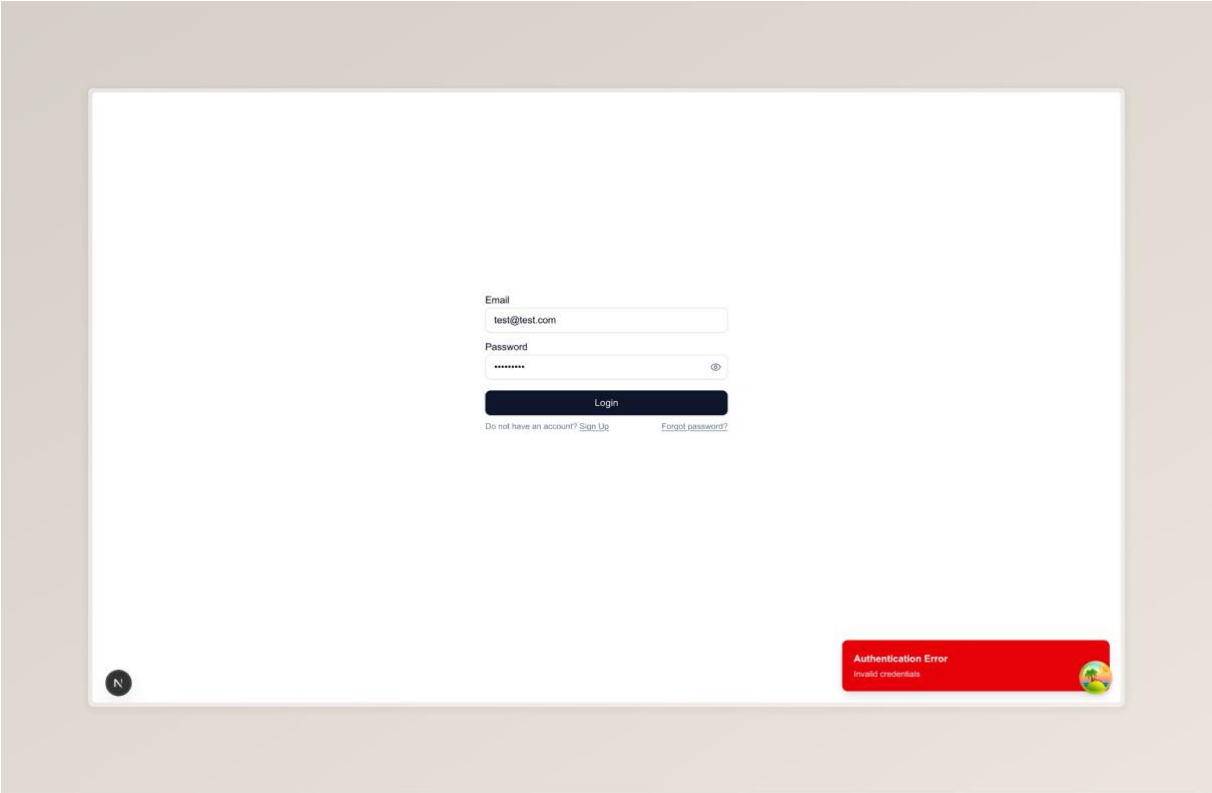

Рисунок 4.1 – Форма авторизації із некоректно введеними даними

Іншим важливим моментом, щоб при створені нового листа, неможливо було відправити запит без заданого імені. Форму можна побачити на рисунку 4.2. Говорячи про форми запитів, важливо вказати те, що жодна форма не повинна дозволяти бути відправлено більше одного разу за короткий проміжок часу: більш конкретним прикладом є ненавмисне клацання кнопки підтвердження запиту більше одного разу для однієї форми. Це важливо, бо користувачі можуть ненавмисно нажати декілька разів через погане інтернет з’єднання, чи малу потужність пристрою, з якого відбувається взаємодія із системою. Отже, в результаті тестування, що можна побачити на рисунку 4.3, було перевірено, щоб інтерфейс давав візуальний індикатор загрузки.

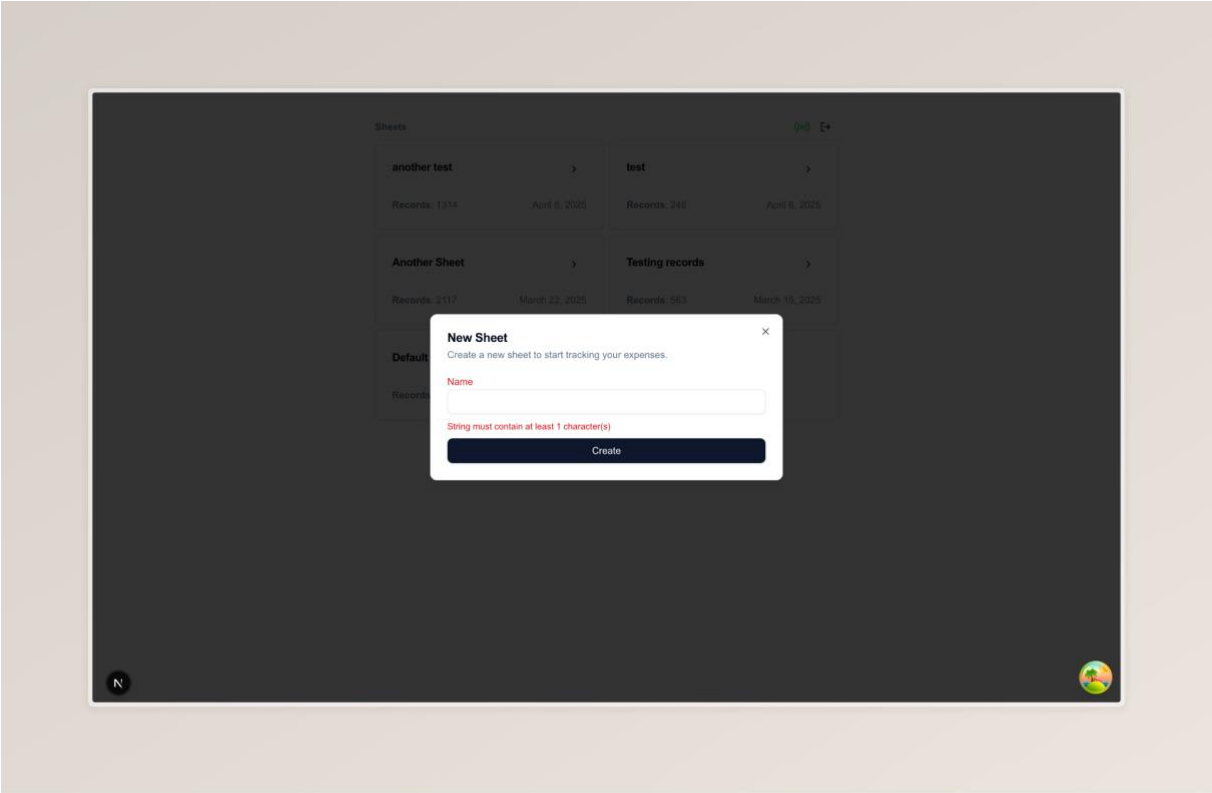

Рисунок 4.2 – Форма створення листу

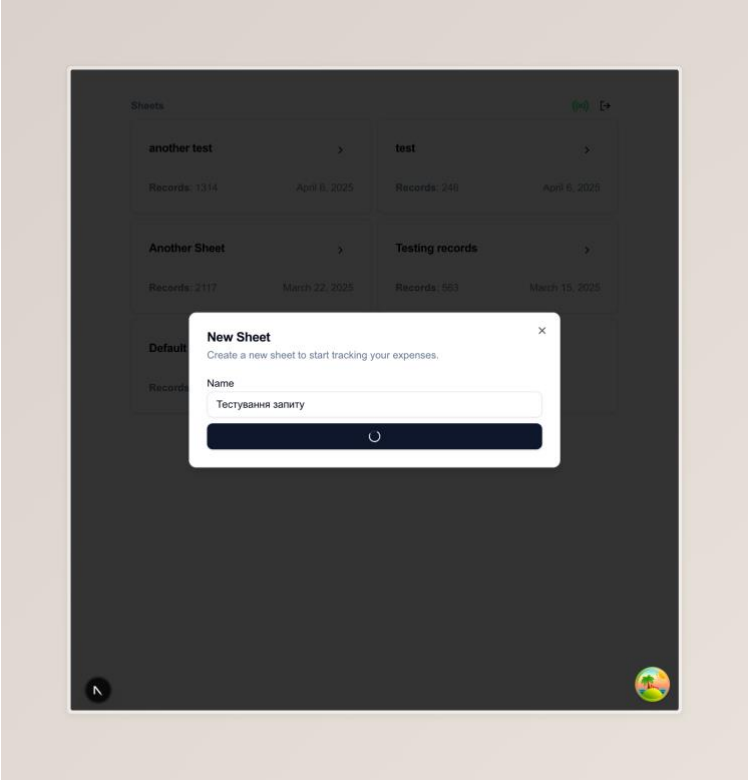

Рисунок 4.3 – Форма створення листу під час довгої загрузки

Важливим елементом системи була перевірка прогнозування витрат користувача за категорією. Отже, коли користувач переходив за посиланням «:id/analytics/:categoryId», де:

– :id – унікальний ідентифікатор листа;

– :categoryId – унікальний ідентифікатор категорії.

Була перевірена логіка прогнозування за необхідною категорією. Наприклад, при переході до категорії «Salary» (з англ. «Зарплатня») можна було побачити передбачення, що зображено на рисунку 4.4.

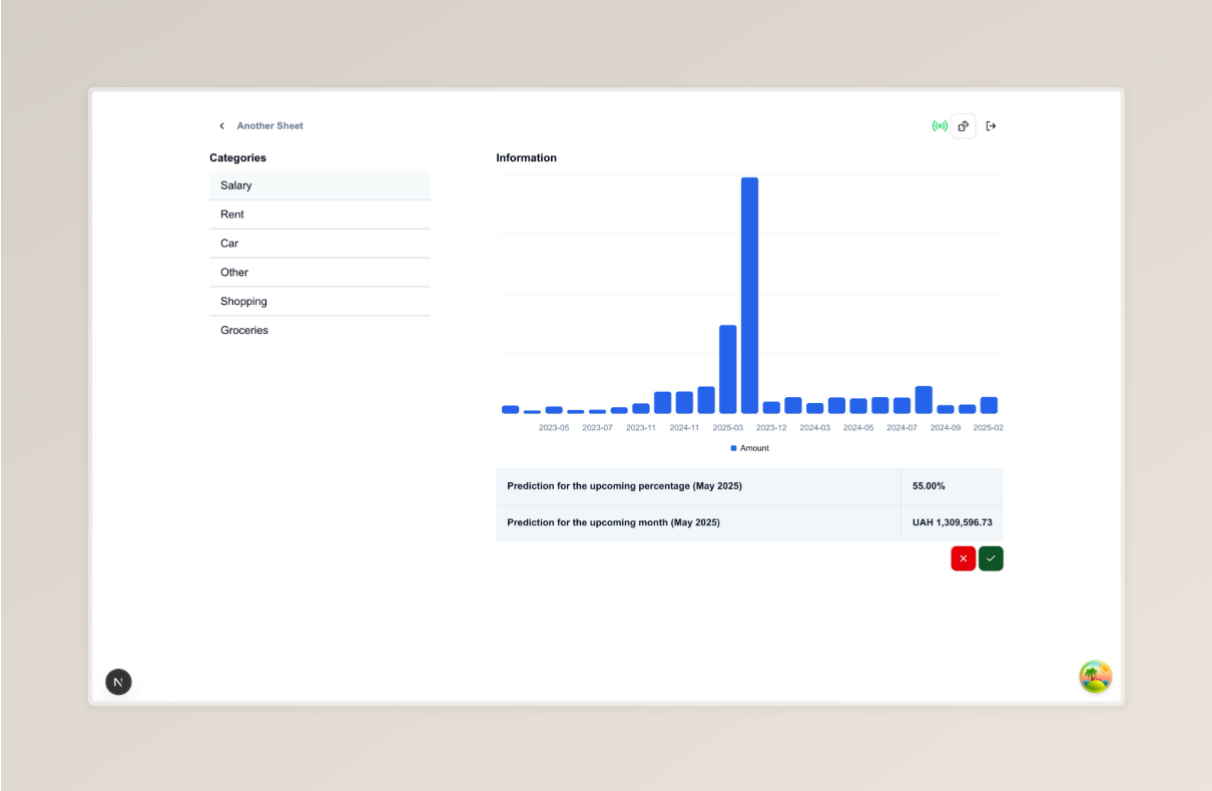

Рисунок 4.4 – Прогнозування доходів користувача

4.2 Підготовка даних для тренування моделі прогнозування

У контексті датасету із фінансовими операціями частою проблемою є брак історичних даних. Через чутливість даних даної природи, майже

неможливо знайти історії транзакцій справжніх людей у відкритих даних, тому є тільки 2 варіанти знаходження необхідної інформації:

– зв’язатися з банковими установами для партнерської роботи;

– генерування синтетичних даних та тренування на справжніх даних користувачів після старту роботи сервісу.

У ході розробки запропонованої системи задля простоти реалізації та відсутності жодних зобов'язань перед банковими структурами, було прийнято рішення використовувати синтетичні дані.

Синтетичні дані – це штучно створені дані, що повинні бути схожі на справжні датасети. На жаль, відкритих синтетичних фінансових наборів даних, що підходять за кількістю інформації до задач даної моделі не було знайдено, тому задля знаходження балансу між кількістю записів та їх можливість нагадувати природну поведінку людей із фінансами, було прийнято рішення згенерувати датасет на 10000 записів датасету за допомогою LLM (Large Language Model) Chat GPT o4 створеної та розповсюдженої американською компанією OpenAI.

Завдяки природі принципів роботи мовних моделей, Chat GPT o4 створило достатньо велику історичних даних, що нагадують поведінку різних людей з фінансами у проміжку великої кількості часу.

4.2 Оцінювання точності моделі прогнозування

Модель прогнозування – важливий компонент системи, що повинен виділяти запропонований додаток серед конкурентів, тому необхідно було ретельно проаналізувати ефективність відповідей моделі.

Під час оцінювання точності роботи моделі використовувалися такі основні метрики:

– MAE (Mean Absolute Error) – середня абсолютна похибка, що обчислює середнє відхилення прогнозованих значень від реальних;

– RMSE (Root Mean Squared Error) – корінь середньоквадратичної похибки, що сильніше штрафує відхилення;

– R^2 (коефіцієнт детермінації) – показник, що демонструє, яку частку варіації реальних даних пояснює модель.

Результат оцінки точності моделі на тестових даних у порівнянні із базовим методом прогнозування: середнім значенням витрат за попередні місяці, – продемонстровано у таблиці 4.1.

Таблиця 4.1 – Таблиця порівняння результатів методів прогнозування

	Модель	Базовий метод
MAE	≈ 92.5 грн	≈ 180.8 грн
RMSE	≈ 120.3 грн	≈ 185.7 грн
R^2	≈ 0.84	≈ 0.57

Таким чином, використання моделі LightGBM для персоналізованого прогнозування витрат дає суттєво кращі результати у порівнянні із простими статистичними методами.

Окремо варто зазначити, що точність прогнозу залежить від обсягу історичних даних користувача. Чим більше транзакцій доступно для аналізу, тим точніше модель може виявити закономірності

ВИСНОВКИ

У ході виконання бакалаврської роботи було проведено дослідження, спрямоване на аналіз фінансових транзакцій користувача та персоналізоване прогнозування витрат за допомогою методів машинного навчання. У процесі роботи було вивчено сучасні підходи до побудови фінансових вебсистем, алгоритми машинного навчання для прогнозування фінансової поведінки, а також методи обробки часових рядів.

Було розроблено концепцію повнофункціонального вебзастосунка для обліку особистих фінансів, що включає систему ведення фінансових записів (або транзакцій), статистику витрат за категоріями, а також модуль прогнозування на основі історичних даних. Особисту увагу було приділено побудові зручного інтерфейсу та реалізації надійного сервера з урахуванням принципів безпеки та масштабованості.

Дослідження підтвердило доцільність використання машинного навчання, зокрема моделей градієнтного бустингу, для прогнозування витрат, що дало змогу підвищити якість фінансового планування. Необхідно зазначити важливість якісної попередньої обробки даних: очищення, агрегації та нормалізації, – для високої точності та ефективної роботи моделі прогнозування.

Узагальнюючи, можна зазначити, що результати дослідження мають як теоретичне, так і практичне значення. Вони можуть бути використані для створення персональних фінансових асистентів, систем планування бюджетів, а також як приклад інтеграції моделей машинного навчання у вебзастосунок.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Jeong Y. An analysis of the impact of UX/UI improvements and web accessibility enhancements on user experience and conversion rates in e-commerce platforms. *ResearchGate*. URL: https://www.researchgate.net/publication/386383076_An_analysis_of_the_impact_of_UXUI_improvements_and_web_accessibility_enhancements_on_user_experience_and_conversion_rates_in_e-commerce_platforms (дата звернення: 28.04.2025).

2. Mitchell O. S., Lusardi A., Curto V. Financial Literacy Among the Young: Evidence and Implications for Consumer Policy. *SSRN Electronic Journal*. 2009. URL: https://doi.org/10.2139/ssrn.1459141 (date of access: 28.04.2025).

3. Соммервілль І. Інженерія програмного забезпечення. Київ : Діалектика, 2013. 816 с.

4. Фаулер М. Архітектура корпоративних додатків. Київ : Діалектика, 2020. 528 с.

5. Філдінг Р. Архітектурні стилі та проектування програмного забезпечення. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm (дата звернення: 30.04.2025).

6. Foy P. Unsupervised Learning: Stock Market Clustering with K-Means. *MLQ.ai*. URL: https://blog.mlq.ai/stock-market-clustering-with-k-means/ (date of access: 29.04.2025).

7. Yu Y. A Survey of Deep Reinforcement Learning in Financial Markets. *Atlantis Highlights in Computer Sciences*. Dordrecht, 2024. P. 188–194. URL: https://doi.org/10.2991/978-94-6463-419-8_24 (date of access: 29.04.2025).

8. Мартін Р. Чистий код: створення, аналіз та рефакторинг. Київ : Фабула, 2019. 416 с.

9. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Incorporated, 2022.

10. Raschka S. Python Machine Learning. 3rd ed. Birmingham : Packt Publishing, 2019. 770 p.

11. Natekin A., Knoll A. Gradient boosting machines, a tutorial. URL: https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2013.00021/full (date of access: 30.04.2025).

12. Дукен М. JavaScript. Повний посібник. Київ : Вид-во Старого Лева, 2022. 560 с.