

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ФІНАНСОВОГО ПЛАНУВАННЯ НА ОСНОВІ ПОВЕДІНКОВИХ

ПАТЕРНІВ КОРИСТУВАЧА

RESEARCH AND SOFTWARE IMPLEMENTATION OF FINANCIAL
PLANNING METHODS BASED ON USER BEHAVIORAL PATTERNS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-111

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

СКОРИК С.С.

(ПРИЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ЗЕЛІК О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

СКОРИКА Сергія Сергійовича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів фінансового планування на основі поведінкових патернів користувача. Research and Software Implementation of Financial Planning Methods Based on User Behavioral Patterns

керівник проєкту (роботи) к.т.н., доцент ЛЬОВКІН Валерій Миколайович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково–пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Проєктування системи. 4. Розробка програми 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1–5 Основна частина	ЛЬОВКІН В.М., доцент		
Нормоконтроль	БСЛОВА А.В., асистент		

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Матеріали і методи.	2 тиждень	Розділ 2
4	Проектування системи.	2 тиждень	Розділ 3
5	Розробка програми.	3–4 тижні	Розділ 4
6	Експлуатація, тестування та експериментальне дослідження програми.	5 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Сергій СКОРИК
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Валерій ЛЬОВКІН
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 115 с., 23 табл., 42 рис., 3 дод., 28 джерел.

ФІНАНСОВЕ ПЛАНУВАННЯ, ПРОГНОЗУВАННЯ ВИТРАТ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, МАШИННЕ НАВЧАННЯ.

Об'єкт дослідження – процес фінансового планування. Предмет дослідження – методи фінансового планування на основі поведінкових патернів користувача. Мета роботи – розробка програмного забезпечення фінансового планування на основі поведінкових патернів користувача для надання можливостей прогнозування витрат користувача з використанням машинного навчання.

Матеріали, методи та технічні засоби: мова програмування Python, фреймворк Django, бібліотеки pandas, scikit-learn, операційна система Ubuntu Linux, контейнеризація за допомогою Docker.

Результати. Розроблено програму, що формує персоналізовані прогнози витрат на основі історії транзакцій. Наукова новизна роботи полягає в запропонованому методі прогнозування витрат на основі поведінкових патернів користувачів, що базується на багатоступеневій обробці та відборі ознак із подальшим застосуванням гібридних нейронних архітектур, що дозволяє виконувати розподілення витрат за категоріями, забезпечуючи гнучкість, точність і адаптивність прогнозування.

Висновки. Розроблена модель автоматизує фінансове планування, адаптуючись до поведінки користувача. Система може використовуватись у персональних фінансових сервісах для ухвалення рішень щодо витрат.

Галузь використання – персональні фінансові сервіси, банківські застосунки, цифрові гаманці.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 115 pages, 23 tables, 42 figures, 3 appendixes, 28 sources.

FINANCIAL PLANNING, COST FORECASTING, SOFTWARE, MACHINE LEARNING.

Object of study is the process of financial planning. Subject of study are methods of financial planning based on user behavioral patterns. The purpose of the study is to develop financial planning software based on user behavioral patterns to provide opportunities for forecasting user expenses using machine learning.

Materials, methods and tools: Python programming language, Django framework, pandas libraries, scikit-learn, Ubuntu Linux operating system, containerization using Docker.

Results. A program has been developed that, based on the analysis of the user's spending history, generates individual forecasts of future expenses. The scientific novelty of the work is as follows the proposed method of cost forecasting based on behavioral patterns based on multi-stage processing and selection of features with followed by the use of hybrid neural architectures, which allows to perform categorization of costs, providing flexibility, accuracy and adaptability forecasting.

Conclusions. The developed model automates financial planning by adapting to user behavior. The system can be used in personal financial services to make spending decisions.

The field of use includes personal financial services, banking applications, and digital wallets.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	9
Вступ.....	10
1 Аналіз предметної області.....	12
1.1 Огляд предметної області.....	12
1.2 Огляд сучасного стану предметної області.....	13
1.3 Потреба програмних рішень	13
1.4 Огляд аналогів	14
1.4.1 YNAB	14
1.4.2 PocketGuard.....	15
1.4.3 Quicken Simplifi	16
1.4.4 Порівняльний аналіз аналогів.....	17
1.5 Висновки до першого розділу.....	20
2 Матеріали і методи.....	21
2.1 Актуальні методи інтелектуального аналізу даних.....	21
2.1.1 Лінійні моделі та дерева рішень	21
2.1.2 Нейронні мережі.....	22
2.1.3 Гібридні підходи	23
2.2 Аналіз набору даних	23
2.3 Метод прогнозування витрат на основі поведінкових патернів користувачів	25
2.4 Висновки до другого розділу	28
3 Проєктування системи.....	29
3.1 Вибір засобів проєктування і розробки	30
3.2 Визначення функціональності системи	33
3.3 Проєктування структури бази даних.....	35
3.4 Архітектура програмного забезпечення	40
3.5 Висновки до третього розділу	42

4 Розробка програми	44
4.1 Алгоритм функціонування застосунку	44
4.2 Особливості програмної реалізації.....	46
4.2.1 Підготовка даних.....	47
4.2.2 Побудова ознак.....	48
4.2.3 Відбір ознак	50
4.2.4 Розділення даних.....	51
4.2.5 Прогнозування та оцінка моделі	52
4.3 Опис модулів розроблюваної системи.....	53
4.3.1 Клас DataPreprocessor	53
4.3.2 Клас DataSplitter	53
4.3.3 Клас FeatureEngineer	54
4.3.4 Клас FeatureSelector	56
4.3.5 Клас CustomHyperModel.....	56
4.3.6 Клас ModelTrainer	57
4.3.7 Клас Predictor.....	57
4.4 Висновки до четвортго розділу.....	58
5 Експлуатація, тестування та експериментальне дослідження програми	59
5.1 Призначення й умови застосування програми.....	59
5.1.1 Призначення програми	59
5.1.2 Функції програми.....	59
5.1.3 Умови застосування програми.....	60
5.2 Умови виконання програми	61
5.3 Експлуатація програми.....	63
5.4 Методика та результати тестування.....	64
5.5 Тестування застосунку та його результати	70
5.6 Висновки до п'ятого розділу.....	81
Висновки	82
Перелік джерел посилання	83
Додаток А Технічне завдання	86

Додаток Б Текст програми	92
Додаток В Слайди презентації.....	107

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;
CSS – Cascading Stylesheets;
CSV – Comma-Separated Values;
GRU – Gated Recurrent Unit;
HTML – Hypertext Markup Language;
LSTM – Long Short-Term Memory;
MAE – Mean Absolute Error;
MASE – Mean Absolute Squared Error;
ORM – Object Relational Mapping;
REST – Representational State Transfer;
RMSE – Root Mean Square Error;
SQL – Structured Query Language;
TCN – Temporal Convolutional Network;
WSGI – Web Server Gateway Interface;
YNAB – You Need A Budget;
СКБД – Система Керування Базами Даних.

ВСТУП

У сучасному світі фінансові послуги набувають дедалі більшого значення, оскільки люди стикаються з необхідністю ефективно управляти своїми фінансами. Бюджетування та контроль витрат є важливими аспектами особистих фінансів, однак без належних інструментів, що допомагають обробляти великі обсяги фінансової інформації, цей процес може бути досить складним. Технології та програмні рішення, що дозволяють прогнозувати витрати на основі історичних даних, стають надзвичайно корисними для підтримки в цьому процесі.

З розвитком інформаційних технологій та зростанням популярності персональних фінансових застосунків стало можливим застосування методів машинного навчання для більш точного прогнозування витрат на основі минулого поведінкового патерну користувача. Прогнозування витрат дає можливість значно покращити процес планування бюджету, сприяючи більш зручному та ефективному управлінню особистими фінансами.

Ідея розробки програми для прогнозування витрат є актуальною, оскільки вона відповідає потребам сучасних користувачів, які бажають оптимізувати свої фінансові витрати, а також адаптувати свої витрати відповідно до персональних потреб. Така програма може стати надійним інструментом для прогнозування майбутніх витрат, що базується на даних про попередні транзакції та забезпечує наочність і зручність у представленні результатів.

Розробка цього інструменту передбачає вирішення низки ключових завдань, серед яких:

- аналіз предметної області управління персональними фінансами та підходів до прогнозування витрат на основі історичних даних;
- експериментальне дослідження ефективності моделей машинного навчання для прогнозування витрат у різних категоріях;

- проектування та розробка програмного забезпечення з інтегрованим вебінтерфейсом для взаємодії з користувачем;
- проведення комплексного тестування програмного забезпечення для перевірки коректності, стабільності та зручності використання.

Успішне виконання цих завдань дозволить створити ефективний інструмент для фінансового планування, що відповідатиме вимогам сучасного користувача та забезпечить ефективне управління особистими фінансами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У сучасних умовах цифрової економіки особливої актуальності набувають інструменти для фінансового планування, які допомагають користувачам ефективно керувати особистими коштами. Основою для таких систем є аналіз транзакцій, на основі яких можна виявляти типові сценарії витрат, прогнозувати майбутні фінансові потреби та рекомендувати бюджетні обмеження. Це дозволяє користувачам краще розуміти структуру своїх витрат, контролювати фінансову дисципліну та своєчасно реагувати на потенційні фінансові ризики.

Крім того, розвиток технологій та доступ до великих обсягів даних дозволяють застосовувати більш складні методи аналізу, такі як машинне навчання та штучний інтелект, для створення більш точних фінансових прогнозів. Вони допомагають не тільки виявляти закономірності в транзакціях, а й адаптувати планування до змінюваних умов, таких як сезонні коливання доходів чи непередбачувані витрати. Такий підхід забезпечує більш персоналізовані та ефективні стратегії управління коштами, що робить фінансове планування доступнішим і кориснішим для широкого кола користувачів [1].

1.1 Огляд предметної області

У сучасному фінансовому середовищі, що динамічно розвивається, ефективне управління особистими коштами стало особливо важливим. Замість традиційних електронних таблиць все частіше використовуються вебзастосунки для фінансового планування, які дозволяють в реальному часі відстежувати витрати, формувати бюджети та прогнозувати фінансові потреби. Такі інструменти забезпечують користувачам структурований підхід до керування фінансами, підвищують прозорість і спрощують прийняття рішень [2].

Крім того, користувачі додатків з управління бюджету знижують свої необов'язкові витрати на 15–20% вже протягом перших трьох місяців регулярного використання [3]. Це свідчить про те, що сучасні інструменти фінансового планування не лише зручні, а й дієві у зміні фінансової поведінки.

1.2 Огляд сучасного стану предметної області

Сучасні програмні рішення для фінансового планування охоплюють широкий спектр функцій – від базового обліку витрат до прогнозування фінансового стану на основі транзакцій. Нижче наведено приклади популярних застосунків, що активно використовуються у сфері персонального фінансового менеджменту:

- You Need A Budget (YNAB) – базується на методиці нульового бюджету, допомагає користувачам планувати кожен долар і формувати стабільні фінансові звички;
- PocketGuard – автоматично синхронізується з банківськими рахунками та показує, скільки грошей доступно для витрат після урахування рахунків і цілей. Пріоритетом додатка є запобігання надмірним витратам;
- Quicken Simplifi – фокусується на простоті використання та гнучкому налаштуванні бюджету.

1.3 Потреба програмних рішень

Попри наявність різноманітних програм для фінансового планування, значна частина з них не враховує важливих аспектів взаємодії з користувачем, зокрема – простоти інтерфейсу, персоналізації та фокусування на конкретних завданнях. Підвищити залученість користувачів можна шляхом спрощення дизайну, орієнтації на чітко визначені фінансові потреби, використання ігрових механік і персоналізованих підходів до взаємодії [4].

Виходячи з цього, існує потреба у створенні спеціалізованого програмного рішення, що поєднує простоту використання з гнучкою функціональністю. Така система має підтримувати імпорт транзакцій з файлів, здійснювати автоматизоване прогнозування витрат, візуалізувати результати у вигляді графіків, забезпечувати фільтрацію категорій і дат. Реалізація таких можливостей сприятиме підвищенню користувацької активності, зменшенню бар'єрів у фінансовому плануванні та кращому розумінню персонального фінансового стану.

1.4 Огляд аналогів

Існує кілька популярних програмних рішень для фінансового планування, кожне з яких має свої особливості та обмеження. Однак більшість з них не надають достатнього рівня персоналізації або простоти у використанні, що може впливати на ефективність користування ними, особливо для осіб з конкретними фінансовими цілями.

1.4.1 YNAB

You Need a Budget – це онлайн-програма персонального бюджетування, заснована на системі конвертів, розроблена приватною американською компанією з однойменною назвою [5].

Програмне забезпечення розроблене з акцентом на методику нульового бюджету, яка полягає в тому, що кожен зароблений долар має бути заздалегідь призначений для певної мети. Додаток допомагає користувачам розподіляти кошти між обов'язковими витратами, накопиченнями та фінансовими цілями, стимулюючи відповідальне ставлення до бюджету.

YNAB пропонує зручний інтерфейс для ручного введення або імпорту банківських транзакцій, а також надає щоденні підказки щодо управління витратами. Він не фокусується на автоматичному прогнозуванні на основі

історичних даних, проте має потужний інструментарій для планування майбутніх витрат, гнучкої перебалансування бюджету та відстеження фінансового прогресу. Програма доступна для Windows, macOS, Android, iOS, а також як вебзастосунок. Головна сторінка вебсайту зображена на рис. 1.1.

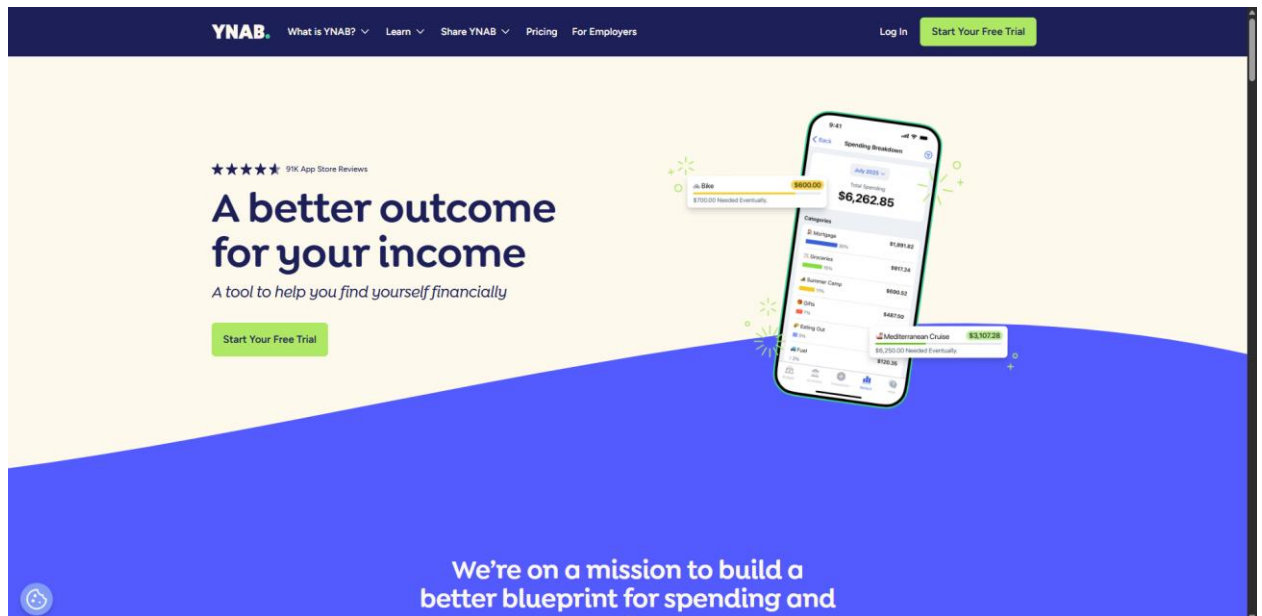


Рисунок 1.1 – Головна сторінка сайту YNAB

1.4.2 PocketGuard

PocketGuard – це мобільний застосунок для керування особистими фінансами, орієнтований на простоту використання та автоматизований контроль витрат. Він автоматично синхронізується з банківськими рахунками користувача, аналізує регулярні надходження й платежі, після чого розраховує суму, яку можна безпечно витратити без ризику перевищення бюджету [6].

Додаток допомагає уникнути імпульсивних витрат, показуючи в режимі реального часу, скільки коштів залишилося після врахування рахунків, підписок та фінансових цілей. PocketGuard не потребує ручного введення даних, однак не забезпечує гнучкого прогнозування майбутніх витрат на основі користувацьких поведінкових моделей. Застосунок доступний на Android та iOS. Головна сторінка вебсайту зображена на рис. 1.2.

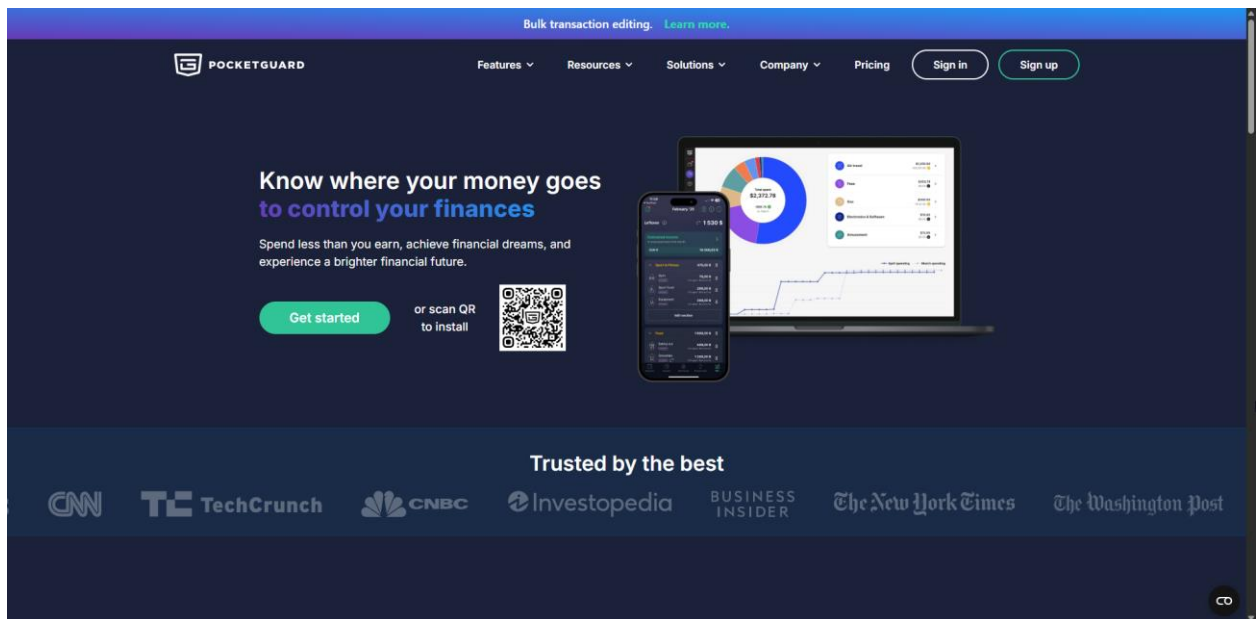
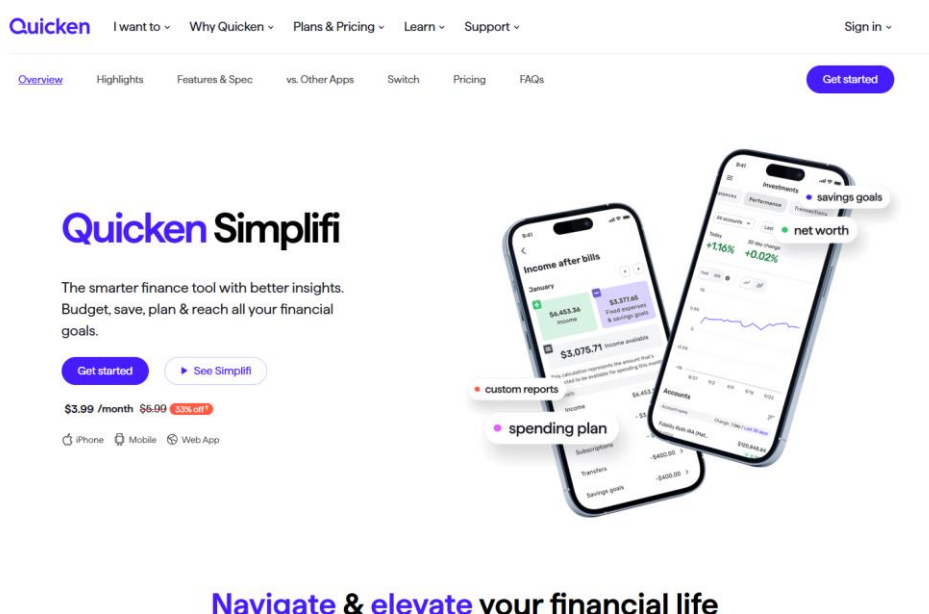


Рисунок 1.2 – Головна сторінка сайту PocketGuard

1.4.3 Quicken Simplifi

Quicken Simplifi – це застосунок для управління особистими фінансами, розроблений компанією Quicken Inc. Його основна мета – забезпечити користувачам простий та зручний спосіб контролювати бюджет, доходи й витрати в одному інтерфейсі [7].

Програма вирізняється мінімалістичним дизайном і підтримує автоматичну синхронізацію з банківськими рахунками, що дозволяє отримувати актуальні дані в реальному часі. Simplifi пропонує налаштовувані фінансові цілі, сповіщення про перевитрати та категоризацію витрат, що спрощує аналіз фінансової поведінки. Додаток також формує прогнози залишку коштів до кінця місяця на основі типових витрат, проте не надає розширених функцій сценарного аналізу чи ручного імпорту великих обсягів транзакцій з файлів. Головна сторінка вебсайту зображена на рис.1.3.



Navigate & elevate your financial life

Рисунок 1.3 – Головна сторінка сайту Quicken Simplifi

1.4.4 Порівняльний аналіз аналогів

Для прийняття подальших рішень необхідно було виконати порівняння існуючих аналогів, що проводилось за такими критеріями: компанія, платформи, основні функції, основні функції, прогнозування витрат, інтерфейс, індивідуальні налаштування. Порівняння існуючих програм для фінансового планування представлено у вигляді табл. 1.1.

Таблиця 1.1 – Порівняння існуючих аналогів

Критерій	YNAB	PocketGuard	Quicken Simplifi	Розроблюваний застосунок
1	2	3	4	5
Компанія	You Need A Budget, LLC	PocketGuard, Inc.	Quicken Inc.	Власна розробка в межах кваліфікаційної роботи

Продовження таблиці 1.1

1	2	3	4	5
Платформи	Windows, macOS, iOS, Android, Вебдодаток	iOS, Android	iOS, Android, Вебдодаток	Вебдодаток
Основні функції	Методика нульового бюджету, гнучкість категорій витрат	Автоматичне синхронізу- вання з банками, запобігання надмірним витратам	Спрощене управління бюджетом, цілі для заощаджень, інтеграція з банками	Завантажен- ня транзакцій, фільтрація, збереження прогнозів
Прогнозуван- ня витрат	Прогноз на базі попередніх даних	Орієнтовано на поточний залишок	Оцінка залишку коштів, формування цілей	Прогнозуван ня бюджету на основі історичних даних із фільтрами категорій і дат
Інтерфейс	Гнучкий, налаштову- ваний, орієнтований на користувача	Лаконічний, спрощений	Мінімаліс- тичний, зручний для початківців	Зручний із вибором категорій, відображен- ням графіків і списком прогнозів

Кінець таблиці 1.1

1	2	3	4	5
Індивідуальні налаштування	Глибока адаптація категорій та фінансових цілей	Базові опції керування витратами	Помірна гнучкість у плануванні та звітах	Фільтрація транзакцій за категоріями та датами, можливість врахування особистих транзакцій у прогнозі

На основі порівняння можна зробити такі висновки.

YNAB – найфункціональніший серед трьох проаналізованих додатків. Він забезпечує повний контроль над бюджетом за допомогою методики нульового бюджету, підтримує глибоку персоналізацію та має гнучкий інтерфейс. Це найкращий варіант для користувачів, які активно керують фінансами та мають складну структуру витрат.

RocketGuard – найпростіший у використанні додаток, орієнтований на автоматизацію контролю витрат. Завдяки лаконічному інтерфейсу та автоматичному підключенню до банківських рахунків він добре підходить для початківців або тих, хто хоче мінімального залучення.

Quicken Simplifi – компромісний варіант, який поєднує простоту інтерфейсу та базові функції прогнозування і планування. Підходить для користувачів, які потребують зрозумілого інструменту з помірною гнучкістю та цільовим плануванням.

Власна розробка представляє сучасний інструмент для персонального фінансового менеджменту з розширеною функціональністю завантаження та обробки транзакцій у популярних форматах. Вона забезпечує прогнозування бюджету на основі історичних даних із можливістю гнучкої фільтрації

категорій та періодів, що дозволяє користувачу отримувати адаптовані та інформативні фінансові прогнози. Інтерфейс застосунку поєднує простоту і зручність із широкими можливостями для кастомізації, включаючи ручне додавання транзакцій та збереження результатів прогнозування. Такий комплексний підхід робить розробку конкурентоспроможною на ринку фінансових застосунків і дає змогу задовольнити потреби як досвідчених користувачів, так і новачків, що прагнуть ефективно планувати свої фінанси.

1.5 Висновки до першого розділу

У першому розділі було проведено огляд сучасного стану та актуальних рішень у сфері фінансового планування та прогнозування витрат. Проаналізовано особливості популярних додатків, таких як YNAB, PocketGuard і Quicken Simplifi, що дозволило виявити їх сильні й слабкі сторони. На основі порівняння сформовано перелік функцій, які мають бути реалізовані в рамках проєкту: гнучке налаштування категорій, імпорт транзакцій, інтерактивне прогнозування бюджету, візуалізація даних та збереження результатів. Це заклало підґрунтя для створення функціональності майбутньої системи, орієнтованої на потреби користувача, простоту взаємодії та ефективність аналізу фінансової інформації.

2 МАТЕРІАЛИ І МЕТОДИ

У цьому розділі представлені основні методи інтелектуального аналізу фінансових даних, аналіз набору даних й власний запропонований метод прогнозування витрат, як необхідна частина під час розробки програмного продукту. Описані матеріали та методи визначають загальний підхід до виконання роботи, а також характеризують ключові етапи використані для збору та обробки даних.

2.1 Актуальні методи інтелектуального аналізу даних

2.1.1 Лінійні моделі та дерева рішень

Лінійні моделі є одними з найперших і водночас найпоширеніших методів машинного навчання. Їх популярність пояснюється поєднанням простоти, швидкодії та інтерпретованості результатів [8]. У фінансовій аналітиці лінійні моделі часто виступають стартовим інструментом, що дозволяє швидко оцінити вплив тих чи інших факторів на результат [9].

Принцип роботи лінійних моделей базується на припущенні, що зв'язок між вхідними ознаками та результатом є лінійним, тобто зміна значення будь-якого вхідного параметра призводить до пропорційної зміни вихідного результату.

Переваги:

- інтерпретованість та прозорість результатів [8];
- швидке навчання [8];
- ефективність при великій кількості ознак.

Обмеження:

- не здатні моделювати нелінійні зв'язки, що частор трапляються у фінансових даних [9];
- чутливість до викидів та шумів у даних.

Дерева рішень – це ще одна популярна категорія алгоритмів машинного навчання, яка надає більшу гнучкість у моделюванні залежностей між змінними. В основі їхньої роботи лежить логіка умовних операторів "якщо... то...", що дозволяє будувати ієрархічну структуру рішень [9].

Переваги:

- здатність моделювати складні, нелінійні взаємозв'язки;
- вбудована обробка взаємодії між ознаками.

Недоліки:

- без належного налаштування схильні до перенавчань;
- не враховують залежності в часі без створення лагів та агрегатів.

2.1.2 Нейронні мережі

Нейронні мережі, зокрема їх рекурентні архітектури – Long Short-Term Memory (LSTM) та Gated Recurrent Unit (GRU), спеціалізуються на моделюванні послідовностей і залежностей у часі, що робить їх природним вибором для задач прогнозування фінансових витрат [10], [11]. Вони ефективно працюють із нерівномірними та розрідженими часовими рядами, характерними для фінансових транзакцій [9], [11].

Переваги:

- моделі здатні "пам'ятати" вплив попередніх днів на поточну поведінку, що критично для виявлення трендів, сезонності, повторюваних шаблонів;
- добре працюють із нерівномірними та розрідженими часовими рядами, що характерно для фінансових транзакцій.

Недоліки:

- довге навчання;
- складність налаштування;
- чутливість до гіперпараметрів;

– «чорна скринька», важко зрозуміти, чому був зроблений саме такий прогноз.

2.1.3 Гібридні підходи

Гібридні моделі, що комбінують LSTM та Temporal Convolutional Network (TCN), набувають популярності для аналізу часових рядів, зокрема фінансових даних [12], [13].

Переваги:

– LSTM фокусується на довготривалих трендах, TCN – на локальних паттернах;

- поєднання дає змогу гнучко виявляти різні часові закономірності;
- підходить для задач з розрідженими та нерівномірними витратами.

Недоліки:

- вимагають великого обсягу даних і значних обчислювальних ресурсів;
- складність архітектури;
- потреба у якісному аналізі даних для вибору оптимальних гіперпараметрів.

2.2 Аналіз набору даних

Набір даних [14], використаний у роботі, був взятий з платформи Kaggle та знаходиться у вільному доступі. Згідно з описом автора, це особисті записи транзакцій бюджету, зібрані протягом тривалого періоду. Дані охоплюють категорії, такі як Restaurants, Markets, Coffee тощо, і містять дату кожної транзакції та суму витрат. Набір даних орієнтований на навчальні цілі та може бути використаний для аналізу фінансів, вивчення поведінкових моделей витрат, побудови прогнозів та візуалізацій [14].

Загалом у наборі міститься 3608 транзакцій, які охоплюють 794 унікальні дні та 22 категорії витрат.

Більшість категорій демонструють високу кількість нульових днів, тобто днів, коли витрати були відсутні. Наприклад, категорії Market і Coffee мають понад 500 активних днів, тоді як Motel, Rent car, Film/Enjoyment зустрічаються лише кілька разів. Кількість нульових значень за категоріями у вигляді графіку зображено на рис. 2.1.

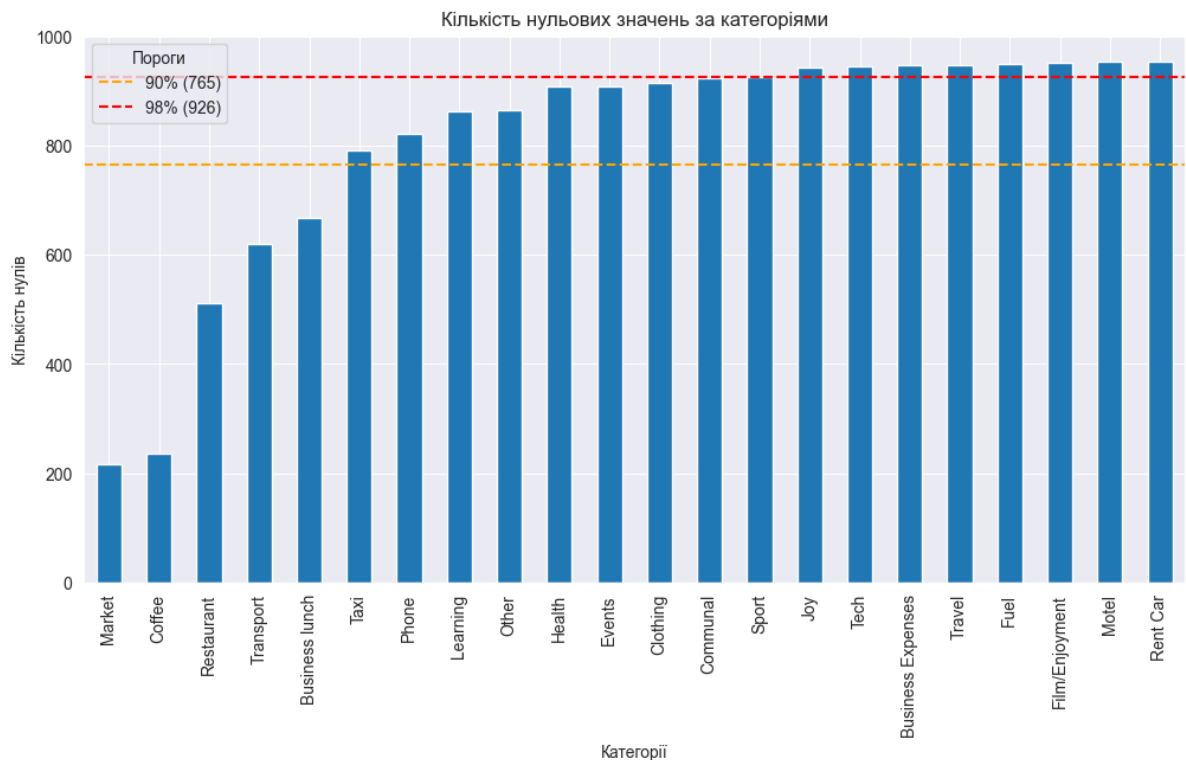


Рисунок 2.1 – Кількість нульових значень за категоріями

У середньому щодня витрати фіксуються лише у 3–4 категоріях, що свідчить про розрідженість вибірки. Кількість категорій витрат по днях зображено на рис. 2.2.



Рисунок 2.2 – Кількість унікальних категорій витрат щодня

Крім того, у даних спостерігаються циклічні коливання та поведінкові шаблони: деякі категорії є типовими для буднів, інші – для вихідних. Витрати мають нерівномірну амплітуду – здебільшого це невеликі суми, проте трапляються окремі високі транзакції, що формують довгий хвіст у розподілі.

Також виявлено орфографічні помилки в назвах деяких категорій (наприклад, «Restuarant», «Coffe»), які були нормалізовані для забезпечення узгодженості.

Таким чином, вибірка є поєднанням регулярних і рідкісних подій, дрібних і великих витрат, що вимагає гнучкого підходу до подальшого прогнозування – з урахуванням як факту витрат, так і їхніх обсягів, а також календарного контексту.

2.3 Метод прогнозування витрат на основі поведінкових патернів користувачів

Враховуючи можливі особливості вибірки, які можуть включати і у випадку вибірки [14] включають розрідженість даних, наявність сезонних та нерегулярних витрат, а також різноманітність категорій – одне універсальне

рішення для прогнозування не може бути достатньо ефективним. Тому метод побудовано як комплексний послідовний процес, що включає кілька етапів трансформації та підготовки даних для подальшого комбінованого моделювання.

На першому етапі має бути виконано приведення даних до структурованого вигляду. Для цього необхідно забезпечити зчитування вхідних файлів користувача. Такі вхідні дані зокрема можуть бути даними у форматі CSV або Excel. Зчитані дані приводяться до структурованого вигляду: кожна транзакція повинна містити дату, суму та категорію. Далі передбачено формування календарних ознак на основі дати транзакції – зокрема, визначення дня тижня, місяця, року, а також створення синусоїдальних і косинусоїдальних перетворень для відображення циклічної природи даних.

На цьому етапі також необхідно виконати класифікацію категорій витрат за частотою їх появи. Для цього потрібно провести аналіз щільності (частки днів із транзакціями в межах кожної категорії) з метою поділу категорій на три групи:

- регулярні категорії характеризуються високою щільністю, тобто наявністю транзакцій майже щодня;
- нерегулярні категорії з середньою щільністю, що відповідає відсутності транзакцій у понад 80% днів;
- неперіодичні категорії з низькою щільністю, коли транзакції спостерігаються дуже рідко.

Наступним етапом є побудова ознак. Зокрема, формуються лагові значення витрат за попередні дні, ковзні середні, медіани, стандартні відхилення, а також ознаки, які відображають відстань до останньої транзакції. Для кожної категорії необхідно обчислити кількість днів із ненульовими витратами у ковзних вікнах розміром 30, 60 та 90 днів. Це дозволяє оцінити інтенсивність витрат на різних часових масштабах і врахувати коротко- та середньострокову активність користувача. Відсутні значення заповнюються нулями.

Після побудови повного набору ознак на наступному етапі реалізується відбір ознак за допомогою статистичних методів оцінки інформативності та виключення слабо корельованих ознак. Здебільшого необхідно застосовувати кореляційний аналіз для оцінки лінійної залежності між ознаками та цільовими змінними, а також методи обчислення міри взаємної інформації для виявлення нелінійних залежностей.

Після завершення етапу відбору інформативних ознак здійснюється розділення підготовленого набору даних на навчальну та тестову підвибірки. Розділення виконується у хронологічному порядку, що дозволяє зберегти часову структуру даних і уникнути витоків інформації з майбутнього в минуле. Такий підхід є критично важливим для забезпечення коректної оцінки якості прогнозної моделі в умовах, наближених до реального сценарію використання. Навчальна вибірка використовується виключно для побудови та калібрування моделі, тоді як тестова – для незалежної перевірки її узагальнюючої здатності.

На завершальному етапі побудована гібридна модель, що складається з LSTM для захоплення довготривалих залежностей та TCN для моделювання локальних патернів. Архітектура моделі включає послідовне застосування кількох рекурентних шарів із варіативною кількістю нейронів, а також регуляризацию для запобігання перенавчанню. На вхід моделі подаються параметри, які включають розмір вікна спостережень, кількість вхідних ознак, горизонт прогнозування та кількість вихідних категорій. Проведено пошук оптимальних гіперпараметрів для налаштування архітектури та процесу навчання, з вибором алгоритму оптимізації серед Adam, AdamW та RMSprop. Модель навчена на тренувальній вибірці з обраними параметрами.

Така комбінація забезпечує гнучкість і точність прогнозів для різних категорій витрат. Якість прогнозів контролюється стандартними метриками, що дає змогу своєчасно коригувати модель.

2.4 Висновки до другого розділу

У другому розділі розглянуто основні методи інтелектуального аналізу фінансових даних: лінійні моделі, дерева рішень, нейронні мережі та гібридні підходи, їхні переваги та обмеження у контексті задачі прогнозування витрат. Було проведено комплексний аналіз вибірки фінансових транзакцій, що дозволив виявити ключові особливості даних, зокрема сезонність, розрідженість і різноманітність категорій витрат. Враховуючи складність і специфіку вибірки, було обґрунтовано необхідність використання комбінованих моделей, здатних враховувати як довготривалі, так і локальні залежності у часі.

Запропонований метод базується на багатоступеневій обробці та відборі ознак із подальшим застосуванням гібридних нейронних архітектур, що забезпечує гнучкість, точність і адаптивність системи прогнозування. Такий комплексний метод створює міцну основу для подальшої розробки і вдосконалення моделей прогнозування фінансових витрат.

3 ПРОЄКТУВАННЯ СИСТЕМИ

Процес розробки програмного забезпечення для прогнозування витрат користувача базується на ретельно спланованому етапі проєктування, який передбачає вибір відповідних інструментів, визначення функціональних вимог, створення архітектури бази даних та програмної системи загалом. У даному розділі описано ключові технічні аспекти побудови системи, яка поєднує сучасні методи штучного інтелекту та вебтехнології для реалізації інструменту фінансового планування.

Під час проєктування було обрано набір технологій, які забезпечують ефективну обробку та зберігання даних за допомогою бази даних PostgreSQL, побудову серверної логіки із застосуванням вебфреймворку Django. Для надійного розгортання застосовано Unicorn та Nginx. Розробка здійснювалась із урахуванням функціональних вимог технічного завдання, яке передбачає підтримку завантаження даних користувача, виконання прогнозування витрат, візуалізацію результатів і можливість гнучкої фільтрації інформації.

У цьому розділі розглянуто:

- вибір інструментів для реалізації системи;
- формалізацію функціональних вимог до програмного продукту;
- побудову логічної структури бази даних;
- опис архітектури програмного забезпечення з урахуванням взаємодії компонентів системи.

Проведене проєктування є критично важливим етапом, що закладає фундамент для надійної, масштабованої та зручної в користуванні системи, яка допоможе користувачам планувати витрати на основі аналізу власної фінансової активності.

3.1 Вибір засобів проєктування і розробки

Для реалізації програмного забезпечення, що виконує прогнозування витрат користувача, було обрано набір сучасних та перевірених технологій, кожна з яких виконує чітко визначену роль у загальній архітектурі системи. Основними критеріями вибору стали: призначення, шаблони для рендерингу, підтримка REST API, асинхронність, Object-relational mapping (ORM), безпека.

Мова програмування – Python.

Python використано як основну мову розробки серверної частини завдяки її простоті та потужним бібліотекам для аналізу даних (pandas, numpy) і машинного навчання (TensorFlow, Keras). Модель прогнозування реалізовано на основі гібридної нейронної мережі, що поєднує LSTM та TCN, що дозволяє ефективно враховувати як довготривалі залежності, так і локальні часові патерни у фінансових даних, підвищуючи точність прогнозів і адаптивність моделі [15]-[19].

Фреймворк для веброботки – Django.

Django обрано для створення бекенду завдяки його високій швидкості розробки, що особливо важливо при швидкому прототипуванні ітеративних змін у проєкті. Вбудована ORM спрощує роботу з базами даних, дозволяючи ефективно керувати моделями та знижуючи кількість ручного SQL-коду [20].

Мова програмування JavaScript та бібліотеки для фронтенду.

JavaScript застосовувався для реалізації логіки на стороні клієнта. Зокрема, за його допомогою реалізовано:

- динамічну побудову графіків прогнозованих витрат у реальному часі з використанням відповідної бібліотеки Chart.js [21];
- логіку інформування користувача через сповіщення, що забезпечує зручну взаємодію з системою.

Система керування базами даних – PostgreSQL.

Для зберігання фінансових транзакцій, параметрів прогнозування та результатів було використано PostgreSQL – надійну реляційну систему керування базами даних (СКБД) з підтримкою складних запитів, транзакцій та ефективного індексування [22].

Розгортання – Gunicorn і Nginx.

Вебдодаток розгорнуто з використанням Web Server Gateway Interface (WSGI) сервера Gunicorn, що ефективно обробляє запити до Django-додатку. Вебсервер Nginx використовується як зворотний проксі, а також обробляє статичні ресурси й забезпечує балансування навантаження [23], [24].

Система контейнеризації – Docker.

Для забезпечення гнучкості та незалежності від середовища розгортання було використано Docker. Контейнери дозволяють ізолювати середовище виконання, що полегшує налаштування, тестування й масштабування системи [25], [26].

Таким чином, комбінація вищезгаданих технологій забезпечує повноцінну функціональність застосунку: від аналізу даних і побудови прогнозів до зручного інтерфейсу з інтерактивною візуалізацією та системою зворотного зв'язку з користувачем.

Серед можливих альтернатив для реалізації бекенд-частини також розглядалися інші популярні фреймворки на Python – Flask та FastAPI [27], [28]. Для аргументованого вибору було проведено порівняльний аналіз основних характеристик трьох вебфреймворків, що подано у табл. 3.1.

Таблиця 3.1 – Порівняння фреймворків для бекенду

Критерій	Django	FastAPI	Flask
1	2	3	4
Призначення	Повноцінний фреймворк «все–в–одному»	Сучасний асинхронний API-фреймворк	Мікрофреймворк для простих застосунків

Кінець таблиці 3.1

1	2	3	4
Шаблони для рендерингу	Вбудовані потужні шаблони для рендерингу HTML	Потрібно інтегрувати сторонні рішення (Jinja2, Mako)	Вбудовані шаблони (Jinja2)
Підтримка REST API	Через Django REST Framework	Вбудована нативна підтримка	Потрібно реалізовувати вручну або через Flask–RESTful
Асинхронність	Часткова підтримка	Повноцінна асинхронна підтримка	Лімітована, через сторонні бібліотеки
ORM	Вбудована потужна ORM	Сторонні бібліотеки	Сторонні бібліотеки
Безпека	Високий рівень (вбудовані механізми)	Менше готових рішень	Потрібно реалізовувати самостійно
Легкість використання	Багато вбудованих інструментів	Вимагає додаткових бібліотек та більше ручної роботи	Вимагає додаткових бібліотек та більше ручної роботи

З огляду на складність і масштаб розроблюваного застосунку, потребу у використанні вбудованої ORM, надійної системи аутентифікації, використанні вбудованих шаблонів для рендерингу, підтримці масштабування та інтеграції з базами даних, було прийнято рішення на користь Django. Його можливості в плані шаблонізації та масштабування роблять його найбільш оптимальним вибором для розробки вебзастосунку.

3.2 Визначення функціональності системи

Функціональність визначає основні можливості та операції, які має забезпечувати система для досягнення поставленої мети – розробити програмне забезпечення для фінансового планування з прогнозуванням витрат користувача на основі його поведінкових патернів з використанням машинного навчання. Відповідно до технічного завдання, основними складовими функціональності програмного забезпечення є:

- імпорт транзакцій – користувач повинен мати змогу завантажити файл з фінансовими транзакціями у форматах CSV або Excel. Завантажені дані повинні автоматично оброблятися, зберігатися у базі даних та використовуватися для подальшого аналізу;

- прогнозування витрат – користувач може ініціювати запуск моделі прогнозування. Модель повинна повертати прогнозовані витрати по категоріях на заданий період часу з урахуванням історичних даних;

- візуалізація прогнозу – результати прогнозування мають бути подані у вигляді графіку реалізованого за допомогою JavaScript, що дозволяє користувачу інтуїтивно сприймати майбутні витрати за різними категоріями;

- інтерактивність графіка – користувач має змогу приховувати або показувати окремі категорії витрат на графіку, що дозволяє зосередитися на найбільш важливих аспектах прогнозу;

- фільтрація даних – користувач повинен мати можливість фільтрувати транзакції за категоріями та часовими рамками перед запуском прогнозу;

- збереження прогнозів – користувач може зберігати результати прогнозування для подальшого аналізу або порівняння з майбутніми результатами;

- перегляд історії прогнозів – користувач повинен мати доступ до раніше збережених прогнозів з можливістю їх перегляду у зручному форматі;

- ручне редагування транзакцій – користувач може додавати або видаляти транзакції вручну для точнішого аналізу;
- перегляд і фільтрація таблиці транзакцій – можливість перегляду особистих транзакцій із фільтрацією за датою, сумою та категорією;
- врахування користувацьких транзакцій у прогнозі – автоматичне включення доданих вручну транзакцій у процес прогнозування.

Реалізація зазначених функціональних вимог дозволяє досягти високого рівня автоматизації процесу фінансового планування, підвищити зручність користування системою та забезпечити точність отримуваних прогнозів завдяки застосуванню сучасних методів машинного навчання.

Основні сценарії взаємодії користувача з системою зображено на рис. 3.1.

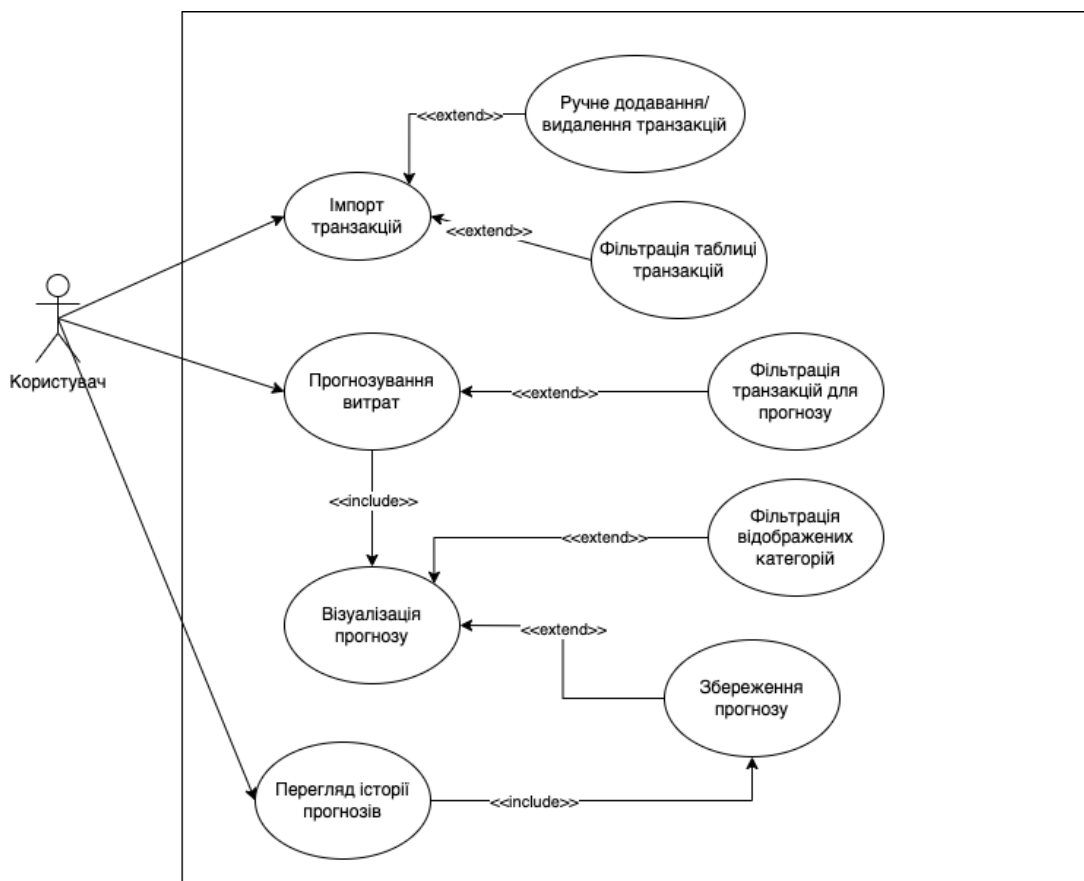


Рисунок 3.1 – Діаграма прецендентів роботи застосунку

3.3 Проектування структури бази даних

Проектування бази даних є ключовим етапом у розробці системи прогнозування витрат, оскільки саме база забезпечує зберігання, цілісність та доступність усіх необхідних даних – як користувацьких, так і аналітичних.

У рамках реалізованого програмного забезпечення була спроектована реляційна база даних, що реалізована за допомогою ORM-системи Django. Структура бази охоплює кілька взаємопов'язаних сутностей, кожна з яких виконує чітко визначену роль у функціонуванні системи. Основні таблиці та їх призначення описані нижче:

Таблиця `accounts_customuser` – зберігає інформацію про користувачів: ім'я, прізвище, пароль, дату останнього входу та інші атрибути. Використовується для автентифікації та авторизації. Структуру колонок наведено у табл. 3.2.

Таблиця 3.2 – Колонки таблиці `accounts_customuser`

Назва колонки	Тип даних
<code>id</code>	<code>BIGINT</code>
<code>email</code>	<code>VARCHAR</code>
<code>first_name</code>	<code>VARCHAR</code>
<code>last_name</code>	<code>VARCHAR</code>
<code>is_active</code>	<code>BOOLEAN</code>
<code>is_staff</code>	<code>BOOLEAN</code>
<code>password</code>	<code>VARCHAR</code>
<code>last_login</code>	<code>TIMESTAMP</code>
<code>is_superuser</code>	<code>BOOLEAN</code>

Таблиця `predictions_userfileupload` – містить метадані про завантажені користувачами файли транзакцій (ім'я файлу, дата завантаження, посилання на користувача).

Структуру колонок таблиці predictions_userfileupload наведено у табл. 3.3.

Таблиця 3.3 – Колонки таблиці predictions_userfileupload

Назва колонки	Тип даних
id	BIGSERIAL
user_id	BIGINT
file	VARCHAR
uploaded_at	TIMESTAMP

Таблиця predictions_transaction –містить дату, суму, категорію витрат та посилання на користувача транзакції. Структуру колонок наведено у табл. 3.4.

Таблиця 3.4 – Колонки таблиці predictions_transaction

Назва колонки	Тип даних
id	BIGSERIAL
name	BIGINT
date	DATE
category_id	BIGINT
amount	NUMERIC
created_at	TIMESTAMP
updated_at	TIMESTAMP
data_source	INTEGER

Таблиця predictions_transactioncategory – таблиця категорій транзакцій, які використовуються для класифікації витрат (наприклад, харчування, транспорт, медицина тощо). Структура колонок наведена у табл. 3.5.

Таблиця 3.5 – Колонки таблиці predictions_transactioncategory

Назва колонки	Тип даних
id	BIGINT
name	VARCHAR

Таблиця predictions_userprediction – зберігає записи про запуски прогнозування, включно з назвою сесії та часом створення. Структуру колонок наведено у табл. 3.6.

Таблиця 3.6 – Колонки таблиці predictions_userprediction

Назва колонки	Тип даних
id	BIGSERIAL
name	VARCHAR
user_id	BIGINT
created_at	TIMESTAMP

Таблиця predictions_predictionday –містить окремі дні прогнозу з прив’язкою до загального прогнозу. Структуру колонок наведено у табл. 3.7.

Таблиця 3.7 – Колонки таблиці predictions_predictionday

Назва колонки	Тип даних
id	BIGSERIAL
prediction_id	BIGINT
order	INTEGER

Таблиця predictions_predictionalgorithm – зберігає загальну інформацію про можливі методи прогнозування. Структуру колонок наведено у табл. 3.8.

Таблиця 3.8 – Колонки таблиці predictions_predictionalgorithm

Назва колонки	Тип даних
id	BIGSERIAL
name	VARCHAR
description	TEXT
created_at	TIMESTAMP WITH TIME ZONE
updated_at	TIMESTAMP WITH TIME ZONE

Таблиця predictions_predictioncategory – таблиця, що містить інформацію про витрати по категорії за певний день. Структуру колонок наведено у табл. 3.9.

Таблиця 3.9 – Колонки таблиці predictions_predictioncategory

Назва колонки	Тип даних
id	BIGSERIAL
prediction_day_id	BIGINT
category_id	BIGINT
amount	DOUBLE

Таблиці auth_* (auth_group, auth_permission тощо) – стандартні таблиці, що використовуються фреймворком Django для реалізації системи прав доступу та груп користувачів. Структуру колонок наведено у табл. 3.10-3.11.

Таблиця 3.10 – Колонки таблиці auth_group

Назва колонки	Тип даних
id	SERIAL
name	VARCHAR

Таблиця 3.11 – Колонки таблиці auth_permission

Назва колонки	Тип даних
id	SERIAL
name	VARCHAR
content_type_id	INTEGER
codename	VARCHAR

Таблиця django_admin_log – журнал змін, що відображає адміністративні події, пов’язані з об’єктами системи. Структуру колонок наведено у табл. 3.12.

Таблиця 3.12 – Колонки таблиці django_admin_log

Назва колонки	Тип даних
id	SERIAL
action_time	TIMESTAMP
user_id	INTEGER
content_type_id	INTEGER
object_id	TEXT
object_repr	VARCHAR
action_flag	INTEGER
change_message	TEXT

На рис. 3.2 наведено повну ER–модель бази даних, створену в процесі розробки.

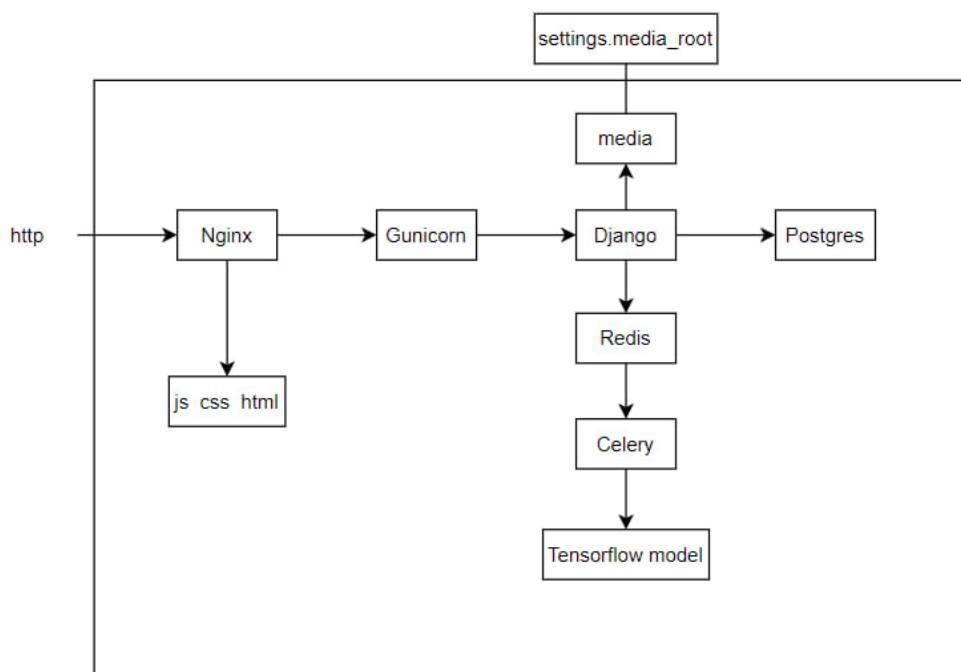


Рисунок 3.3 – Загальна архітектура застосунку

Основні компоненти системи:

- Nginx – вебсервер, що виконує роль зворотного проксі. Він отримує HTTP-запити від користувачів і передає їх до Gunicorn. Також обслуговує статичні ресурси (JavaScript, Cascading Stylesheets (CSS), Hypertext Markup Language (HTML)).
- Gunicorn – WSGI-сервер, який виконує запуск Django-додатку та обробку запитів від Nginx;
- Django – основний вебфреймворк, що забезпечує логіку обробки запитів, управління даними, авторизацію користувачів та формування відповідей;
- PostgreSQL – СКБД, яка зберігає всю інформацію про користувачів, транзакції, категорії витрат, прогнози тощо;
- settings.MEDIA_ROOT / media – директорія для зберігання завантажених користувачами файлів транзакцій, які надалі використовуються для аналізу та побудови прогнозів;

- Redis – брокер повідомлень, який використовується разом із Celery для асинхронного виконання задач побудови прогнозу;
- Celery – система розподілених задач, яка дозволяє виконувати довготривалі чи важкі задачі у фоновому режимі, не блокуючи основний потік програми;
- TensorFlow model – модель машинного навчання, яка викликається Celery-завданням для побудови прогнозу.

Вся система контейнеризована за допомогою Docker, що забезпечує ізольоване середовище для кожного компонента та спрощує процес розгортання. Кожен з елементів архітектури – Nginx, Gunicorn та Django, PostgreSQL, Redis, Celery – розгортається у власному контейнері, що дозволяє легко масштабувати систему, оновлювати окремі модулі та переносити її між різними середовищами без зміни конфігурації. Використання Docker Compose забезпечує автоматизовану оркестрацію сервісів.

Ця архітектура забезпечує високу продуктивність та масштабованість застосунку. Завдяки використанню Redis і Celery фонові задачі, пов'язані з машинним навчанням, не перевантажують вебсервер і можуть оброблятися незалежно. Крім того, завдяки розділенню логіки між Nginx, Gunicorn і Django забезпечується модульність та гнучкість у розгортанні системи.

3.5 Висновки до третього розділу

У даному розділі було описано реалізацію вебзастосунку для прогнозування витрат користувачів із використанням контейнеризованої архітектури на базі Docker. Обґрунтовано вибір Django як основного фреймворку серверної частини, а також використання JavaScript для реалізації фронтенд-логіки, включно з динамічною побудовою графіків та системою нотифікацій для підвищення зручності користування.

Було розроблено структуру бази даних на основі PostgreSQL, що забезпечує ефективне зберігання даних про користувачів, транзакції, категорії, файли та результати прогнозування.

Також представлено архітектуру програмного забезпечення, яка включає компоненти Nginx, Gunicorn, Django, PostgreSQL, Redis, Celery та модель TensorFlow. Така багаторівнева архітектура забезпечує високу продуктивність обробки запитів, асинхронне виконання задач та інтеграцію машинного навчання створення прогнозів.

Загалом, реалізація програмного забезпечення охоплює як серверну, так і клієнтську логіку, забезпечуючи інтерактивність, зручність користування та масштабованість, що є важливими характеристиками сучасного вебзастосунку.

4 РОЗРОБКА ПРОГРАМИ

4.1 Алгоритм функціонування застосунку

Процес побудови прогнозної моделі реалізовано у вигляді послідовного пайплайну з п'яти основних етапів, кожен з яких виконує специфічні трансформації даних та забезпечує модульність системи.

Перший етап – підготовка даних.

На початковому етапі виконується завантаження вхідних даних користувача з файлів формату CSV або Excel. Зчитані транзакції приводяться до уніфікованого вигляду з обов'язковими полями: дата, сума, категорія. Далі здійснюється генерація календарних ознак на основі дати транзакції – визначаються місяць, день тижня, рік, а також додаються синусоїдальні та косинусоїдальні перетворення для відображення циклічної структури часу.

Після цього формується зведена таблиця витрат користувача за категоріями в розрізі дат. Для кожної категорії розраховується частка днів, у які спостерігались транзакції, що дозволяє класифікувати категорії за частотою:

- регулярні категорії з високою частотою появи;
- нерегулярні категорії, де транзакції відсутні більше ніж у 80% днів;
- неперіодичні категорії з пропусками у понад 95% днів.

Для зменшення розмірності та зниження впливу шуму суми витрат у неперіодичних категоріях агрегуються. Також додатково генеруються допоміжні ознаки – зокрема, ті, що мають високу кореляцію з витратами в регулярних категоріях.

Другий етап – побудова ознак.

Наступним кроком є побудова розширеного набору ознак, що включає обчислення лагових значень, ковзних середніх, медіан і стандартних відхилень для виявлення локальних трендів і сезонності. Крім того, додаються календарні та циклічні ознаки, що відображають особливості часу, а також домен-специфічні показники, зокрема частку витрат кожної категорії у

загальних витратах і врахування важливих подій. Особливу увагу приділено ознакам, що відображають інтенсивність витрат, зокрема кількість днів із ненульовими витратами в ковзних вікнах розміром 30, 60 та 90 днів, що дозволяє оцінити коротко- та середньострокову активність користувача. Відсутні значення заповнюються нулями. В результаті формується повний набір ознак, готових для подальшого моделювання.

Третій етап – відбір ознак.

Після формування повного набору ознак проводиться комплексний відбір ключових ознак. Спершу оцінюється значущість календарних змінних, потім здійснюється фільтрація ознак за рівнем розрідженості та варіативності, щоб виключити малоінформативні або надмірно розріджені дані. Далі виконується аналіз міри взаємної інформації між ознаками та цільовими змінними для виявлення як лінійних, так і нелінійних залежностей. На основі отриманих результатів формується оптимальний набір ознак, який максимально впливає на якість прогнозування.

Четвертий етап – розділення даних.

Дані розділяються на тренувальну та тестову вибірки у хронологічному порядку, що імітує реальний сценарій прогнозування. Відокремлюються вхідні ознаки та цільові змінні. Забезпечується відсутність перетину за часом між вибірками для коректної оцінки моделі.

П'ятий етап – прогнозування.

Реалізується навчання моделей машинного навчання на підготовлених ознаках, з використанням рекурентних мереж LSTM та TCN і автоматичним пошуком гіперпараметрів.

На рис. 4.1 зображено діаграму діяльності, яка ілюструє послідовний процес побудови моделі.



Рисунок 4.1 – Процес побудови моделі прогнозування

4.2 Особливості програмної реалізації

У цьому пункті розглядаються технічні аспекти реалізації пайплайну побудови моделі прогнозування. Описано структуру коду, а також підходи до автоматизації процесів підготовки даних, побудови ознак, відбору ознак, розділення даних та навчання моделей.

4.2.1 Підготовка даних

На початковому етапі здійснюється завантаження даних користувача та парсинг дат транзакцій із подальшою генерацією базових часових ознак – таких як місяць, день тижня та рік. Ці ознаки необхідні для врахування сезонності та циклічності витрат, що є характерними для особистих фінансів.

Далі формується зведена таблиця, в якій витрати агрегуються за категоріями у часовому розрізі. Така трансформація дозволяє перетворити розрізнені записи в упорядкований часовий ряд, що є зручним для аналізу та подальшого моделювання.

На наступному кроці категорії витрат класифікуються за часткою нульових значень у їхніх часових рядах. Категорії поділяються на три типи:

- регулярні (з часткою пропущених значень менше 20%);
- періодичні (з пропущеними значеннями від 80% до приблизно 95%);
- рідкісні (з пропущеними значеннями понад 95%).

Пороги для класифікації вибрані з урахуванням унікальних періодів даних і пропущених значень, що дозволяє моделі сфокусуватися на найбільш стабільних патернах поведінки користувача.

Для врахування циклічності часу додаються синусоїдальні та косинусоїдальні перетворення для дня тижня і місяця. Це дозволяє моделі інтерпретувати час як неперервну змінну, уникаючи штучних розривів між кінцем і початком періоду (наприклад, між неділею та понеділком).

Після цього відбувається формування агрегованих ознак сумарних витрат за періодичними та рідкісними категоріями. Такий підхід знижує розмірність даних і зменшує вплив шуму від малозначущих чи випадкових витрат, що сприяє кращому фокусуванню моделі на загальних трендах.

На фінальному етапі виконується пошук допоміжних ознак, які мають високий рівень кореляції з регулярними витратами. Кореляція визначається за допомогою коефіцієнта Пірсона, що допомагає відібрати найбільш релевантні змінні для побудови прогнозової моделі.

Код визначення допоміжних ознак за кореляцією наведений нижче:

```
all_expense_columns_for_corr = [col for col in (regular_cols +
periodical_cols + occasional_cols) if col in self.df.columns]

if not all_expense_columns_for_corr:
    correlated_aux_features = []
else:
    df_for_correlation_analysis =
self.df[all_expense_columns_for_corr].copy()
    valid_regular_cols_for_corr = [col for col in regular_cols if col in
df_for_correlation_analysis.columns]

    if not valid_regular_cols_for_corr:
        correlated_aux_features = []
    else:
        correlated_aux_features = self._find_correlated_aux_features(
            df_corr_analysis=df_for_correlation_analysis,
            target_columns=valid_regular_cols_for_corr,
            correlation_threshold=correlation_threshold_for_aux_features,
            method='pearson'
        )
```

4.2.2 Побудова ознак

Першим кроком побудови ознак є агрегування витрат за типами категорій дозволяє моделі працювати не лише з окремими транзакційними ознаками, а й з узагальненими патернами поведінки користувача. Зокрема наступні.

IrregularSpendings – сума всіх категорій, класифікованих як «нерегулярні». Якщо таких категорій немає, значення дорівнює нулю. Це виділяє витрати з непередбачуваним характером.

NonPeriodicSpendings – агрегує категорії з рідкісними транзакціями, які мають ще нижчу регулярність. Це допомагає відокремити шум від стабільних витрат.

AllListedSpendings – сумує всі категорії: цільові, нерегулярні та неперіодичні, формуючи загальну оцінку щоденних витрат як агрегований вхідний сигнал для моделі.

Таке агрегування зменшує розмірність даних, зберігаючи ключову інформацію про структуру витрат, що підвищує інтерпретованість моделі та

дозволяє враховувати контекст навіть при дефіциті даних у конкретних категоріях.

Далі виконується перевірка неперервності часових даних – пропущені дати заповнюються нульовими рядками, що є необхідним для коректної роботи алгоритмів обробки часових рядів із рівномірним кроком.

Для нормалізації розподілу витрат застосовується логарифмічна трансформація $\log 1p$, а також проводиться тест Дікі-Фуллера для перевірки стаціонарності часових рядів, що є важливим для подальшого аналізу і прогнозування.

Потім розраховуються календарні (день тижня, місяць, квартал) і циклічні ознаки ($\sin/\cos$ від дня тижня, місяця, дня року), що допомагають моделі коректно інтерпретувати час як неперервний параметр без штучних розривів.

Для врахування минулої поведінки створюються лаг-ознаки – зсуви витрат на кілька попередніх днів, а також ковзні статистики на основі рухомих вікон. Вони включають середнє, медіану, суму, стандартне відхилення, асиметрію, ексцес, квартилі, інтерквартильний розмах, середнє абсолютне відхилення і обрізане середнє. Ці ознаки допомагають моделі вловлювати локальні тренди, сезонність і мінливість у витратах.

Окрім цього, додаються ознаки, що відображають час від останньої витрати та частоту транзакцій у вікнах 30, 60 і 90 днів, що надає моделі інформацію про регулярність і ритм фінансової активності користувача.

Завершальним кроком є генерація домен-специфічних ознак, таких як частка категорій у загальних витратах (ролінгові співвідношення), прапор великої витрати вчора та сума нетаргетних витрат за останні дні. Це дозволяє враховувати вплив значущих подій і нетипових операцій.

Перед поверненням результату виконується перевірка відсутності пропущених значень і, у разі потреби, їх заповнення нулями. В результаті формується повний набір якісних ознак, готових для подачі у модель прогнозування.

Код алгоритму тесту Дікі-Фуллера наведений нижче:

```
for col_name in cols_to_test_adf:
    series_to_test = self.df[col_name].dropna()
    if len(series_to_test) < 10:
        continue
    try:
        p_value = adfuller(series_to_test)[1]
        if p_value < 0.05:
            self.logger.info(f"'{col_name}' likely stationary (p-value:
{p_value:.4f})")
        else:
            self.logger.info(f"'{col_name}' likely NON-stationary (p-value:
{p_value:.4f})")
    except Exception as e:
        self.logger.error(f"ADF test error for '{col_name}': {e}")
```

4.2.3 Відбір ознак

Для забезпечення побудови ефективної моделі прогнозування використовується відбір ознак шляхом зменшення розмірності простору ознак, усунення надлишкових чи шумових змінних, покращення якості даних та зменшення ризику перенавчання.

На першому етапі виконується аналіз календарних ознак, таких як день тижня, місяць, квартал. Для цього застосовується непараметричний тест Краскала-Уолліса з метою перевірки, чи мають ці змінні статистично значущий вплив на цільові змінні. Ознаки, які не демонструють такого впливу, відкидаються як неінформативні.

Другим кроком є фільтрація ознак за розрідженістю та варіативністю. Тут обчислюється частка ненульових значень і дисперсія лише для непорожніх елементів. Ознаки, які або занадто рідкісні, або практично не змінюються з часом, вилучаються з подальшого аналізу.

На наступному етапі здійснюється оцінка взаємної інформації (mutual information) між кожною ознакою і цільовими змінними. Це дозволяє визначити, наскільки кожна ознака містить інформацію, корисну для передбачення. Залишаються лише ті ознаки, чия взаємна інформація перевищує певний статистичний поріг.

Після цього застосовується метод Permutation Importance, заснований на навчанні моделі Random Forest. Визначається внесок кожної ознаки у якість прогнозу шляхом вимірювання зниження точності при випадковому перемішуванні її значень. Ознаки з низькою значущістю вилучаються.

В результаті формується оптимальний підмножина ознак, яка забезпечує ефективну, інтерпретовану й стабільну модель машинного навчання.

4.2.4 Розділення даних

Для розділення даних на тренувальну та тестову вибірку спочатку здійснюється визначення формату цільових змінних: за потреби використовуються логарифмовані значення. Це дозволяє зробити розподіл даних ближчим до нормального, що підвищує ефективність моделей, чутливих до відхилень та асиметрії.

Далі перевіряється коректність вибраних ознак та індексу: гарантується, що всі вказані ознаки реально присутні у підготовленому датасеті, а індексом є часові мітки, необхідні для хронологічного поділу.

Основним етапом є хронологічне розділення даних на train та test за заданою датою. Це імітує реальний сценарій, коли модель навчається на історичних даних і робить прогноз на майбутнє. Такий підхід дозволяє уникнути витоку інформації.

Після поділу даних відокремлюються вхідні ознаки та цільові змінні.

У разі наявності пропущених значень у цільових змінних, вони заповнюються нулями, аби уникнути помилок при навчанні моделі.

Загалом, реалізований підхід забезпечує коректне, стабільне та гнучке формування навчальних і тестових вибірок для моделі машинного навчання, що є критично важливим для точного прогнозування, особливо в задачах, пов'язаних з часовими рядами.

4.2.5 Прогнозування та оцінка моделі

На завершальному етапі реалізується процес побудови та навчання гібридної моделі, яка поєднує архітектури LSTM та TCN. LSTM використовується для моделювання довготривалих часових залежностей, тоді як TCN дозволяє ефективно виявляти локальні патерни в межах фіксованого вікна спостережень. Модель побудована як послідовність рекурентних і згорткових шарів із варіативною кількістю нейронів, із застосуванням регуляризаційних технік для зниження ризику перенавчання.

На вхід моделі подається набір параметрів, що включає розмір вікна, кількість вхідних ознак, горизонт прогнозування та кількість цільових категорій. Попередньо здійснюється масштабування даних для стабільності навчання. Параметри моделі налаштовуються шляхом перебору гіперпараметрів, зокрема вибору оптимізатора серед Adam, AdamW і RMSprop, а також підбору розміру шарів та швидкості навчання. Навчання моделі здійснюється на тренувальній вибірці із використанням механізмів ранньої зупинки для запобігання перенавчанню. Прогнозування реалізується рекурсивно, коли кожен наступний крок базується на попередньому, що відповідає сценаріям реального використання.

Код для підбору гіперпараметрів наведений нижче:

```
def fit(self, hp, model, *args, **kwargs):
    batch_size = hp.Int('batch_size',
                        min_value=self.batch_size_min,
                        max_value=self.batch_size_max,
                        step=self.batch_size_step)
    return model.fit(*args, batch_size=batch_size, **kwargs)
```

4.3 Опис модулів розроблюваної системи

4.3.1 Клас DataPreprocessor

Клас DataPreprocessor призначений для обробки даних про витрати стосовно підготовки, класифікації категорій (табл. 4.1).

Таблиця 4.1 – Методи класу DataPreprocessor

Методи класу	Опис
<code>__init__</code>	Ініціалізація екземпляру класу
<code>_create_pivot_table</code>	Створення зведеної таблиці із сумами витрат за категоріями
<code>_engineer_temporal_features</code>	Створення циклічних часових ознак ($\sin$, $\cos$) та кодування року
<code>_engineer_aggregated_spend_features</code>	Створення агрегованих ознак для періодичних та рідкісних категорій
<code>_find_correlated_aux_features</code>	Пошук допоміжних ознак з кореляцією вище заданого порогу
<code>process_spending_data</code>	Основний метод обробки даних: завантаження, трансформація, інженерія ознак, класифікація та пошук корельованих ознак

4.3.2 Клас DataSplitter

Клас DataSplitter призначений для розділення DataFrame на тренувальні та тестові набори (табл. 4.2).

Таблиця 4.2 – Методи класу DataSplitter

Методи класу	Опис
<code>__init__</code>	Ініціалізація екземпляру класу
<code>split_data</code>	Розділення даних на тренувальні та тестові набори X і y

4.3.3 Клас FeatureEngineer

Клас FeatureEngineer призначений для розширеного аналізу часових рядів витрат і генерації ознак (табл. 4.3).

Таблиця 4.3 – Методи класу FeatureEngineer

Методи класу	Опис
1	2
<code>__init__</code>	Ініціалізація екземпляру класу
<code>_check_datetime_index_continuity</code>	Перевірка неперервності індексу дат у таблиці
<code>_check_features</code>	Перевірка наявності необхідних ознак у таблиці
<code>_aggregate_spending_categories</code>	Агрегація витрат за заданими категоріями
<code>_fill_missing_dates</code>	Заповнення відсутніх дат у таблиці
<code>_apply_log_transformation</code>	Застосування логарифмічного перетворення до числових стовпців витрат

Кінець таблиці 4.3

1	2
_check_stationarity	Перевірка стаціонарності часових рядів тестом Дікі-Фулера
week_of_month	Обчислення номера тижня в межах місяця
_create_calendar_features	Створення циклічних (синус/косинус) ознак для часових компонентів
_create_lag_features	Створення лагових ознак для цільових стовпців
_rolling_mad	Обчислення медіанного абсолютного відхилення в скользячому вікні
_rolling_trimmed_mean	Обчислення усіченого середнього в скользячому вікні
_create_rolling_window_features	Створення ознак з використанням скользячих статистик для цільових стовпців
_create_time_since_and_frequency_features	Створення ознак часу від останньої активності та частоти витрат
_create_domain_specific_features	Створення специфічних ознак на основі доменної логіки
engineer_features	Головний метод для створення повного набору ознак

4.3.4 Клас FeatureSelector

Клас FeatureSelector призначений для виконання відбору ознак з використанням статистичних тестів, взаємної інформації, кореляційного аналізу та важливості ознак (Permutation Importance) з RandomForest (табл. 4.4).

Таблиця 4.4 – Методи класу FeatureSelector

Методи класу	Опис
<code>__init__</code>	Ініціалізація екземпляру класу
<code>_perform_calendar_feature_analysis</code>	Аналіз календарних ознак за заданими групами ознак і цільових колонок
<code>_filter_by_sparsity_and_variance</code>	Фільтрація ознак на основі їх розрідженості та дисперсії
<code>_calculate_mutual_information</code>	Обчислення взаємної інформації між ознаками та цільовими змінними для визначення важливості
<code>_get_permutation_importance</code>	Оцінка важливості ознак за допомогою методу перестановок на основі зв'язку з цільовими змінними
<code>select_features</code>	Метод для відбору релевантних ознак з урахуванням оброблених даних, цілей та параметрів тестового набору

4.3.5 Клас CustomHyperModel

Клас CustomHyperModel призначений для інкапсуляції побудови та навчання моделі через Keras Tuner (табл. 4.5).

Таблиця 4.5 – Методи класу CustomHyperModel

Методи класу	Опис
<code>__init__</code>	Ініціалізація екземпляру класу
<code>build</code>	Побудова моделі на основі гіперпараметрів
<code>fit</code>	Навчання моделі з використанням тюнеру

4.3.6 Клас ModelTrainer

Клас ModelTrainer відповідає за фінальне навчання моделі (табл. 4.6).

Таблиця 4.6 – Методи класу ModelTrainer

Методи класу	Опис
<code>__init__</code>	Ініціалізація екземпляру класу
<code>tune_hyperparameters</code>	Пошук гіперпараметрів з Keras Tuner
<code>load_model</code>	Завантаження раніше збереженої моделі

4.3.7 Клас Predictor

Клас Predictor призначений для створення та оцінки прогнозів (табл. 4.7).

Таблиця 4.7 – Методи класу Predictor

Методи класу	Опис
<code>__init__</code>	Ініціалізація екземпляру класу
<code>predict_one_day</code>	Виконує прогноз на один крок
<code>predict_recursively</code>	Робить рекурсивні прогнози на вказану кількість кроків
<code>calculate_mae</code>	Обчислення середньої абсолютної помилки
<code>calculate_mase</code>	Обчислення середньої абсолютної помилки масштабу
<code>evaluate_one_day_prediction</code>	Оцінка прогнозу одного або кількох кроків
<code>evaluate_on_test_set</code>	Оцінка моделі на тестовому наборі

4.4 Висновки до четвертого розділу

У четвертому розділі було зосереджено увагу на реалізації програмного забезпечення та докладному описі архітектури моделі прогнозування. Розглянуто основні компоненти, що забезпечують обробку вхідних даних, побудову ознак, запуск моделі та відображення результатів користувачу.

Особливу увагу приділено структурі моделі машинного навчання, яка реалізує багатокрокове прогнозування витрат за категоріями. Було описано алгоритм побудови ознак, включно з лагами, статистиками за ковзними вікнами, контекстними та частотними характеристиками, що забезпечують моделі інформаційне підґрунтя для точного прогнозу.

Результати розробки свідчать про цілісність і функціональну завершеність архітектурного рішення, що забезпечує ефективну роботу системи в межах поставлених завдань.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення й умови застосування програми

5.1.1 Призначення програми

Призначенням розробленої програми є автоматизоване прогнозування особистих витрат користувача, яке базується на аналізі історії його фінансових операцій. Застосунок створений для допомоги користувачам здійснювати ефективне управління своїми фінансами, що включає не лише контроль витрат, а й планування на майбутнє на основі точних прогнозів.

Однією з ключових функцій програми є автоматичне прогнозування витрат по кожній категорії, що дозволяє користувачам отримати чітке уявлення про свої фінансові потреби в майбутньому. Крім того, для зручності сприйняття результатів, система надає користувачеві графічну візуалізацію прогнозованих витрат, що значно полегшує сприйняття складних фінансових даних та дозволяє користувачам бачити закономірності і тенденції своїх витрат.

Таким чином, програма надає значну допомогу користувачам, які хочуть ефективно планувати свій бюджет на основі аналізу минулих витрат, без необхідності постійного введення нових даних чи ручного моніторингу фінансових операцій.

5.1.2 Функції програми

Розроблене програмне забезпечення надає низку функціональних можливостей, які дозволяють користувачам ефективно працювати з фінансовими даними та отримувати точні прогнози щодо майбутніх витрат. Система орієнтована на простоту використання та гнучкість, забезпечуючи наступні основні функції:

- користувач може імпортувати файл з фінансовими транзакціями у форматі CSV або Excel;
- після завантаження даних запускається процес прогнозування майбутніх витрат на основі виявлених у минулому закономірностей;
- результати прогнозування виводяться у вигляді графіка, що дозволяє наочно оцінити витрати по категоріях;
- користувач має змогу приховати окремі категорії на графіку для зосередження уваги на потрібних даних;
- перед прогнозуванням можна обрати лише ті категорії витрат, які потрібно враховувати в розрахунках;
- фільтрація транзакцій за певними датами, що дозволяє обмежити період аналізу;
- результати прогнозування можна зберегти для подальшого аналізу чи порівняння;
- додавання та видалення транзакцій вручну;
- перегляд доданих користувачем транзакцій у табличному вигляді;
- фільтрація таблиці транзакцій за датою, сумою та категорією;
- врахування користувацьких транзакцій у процесі прогнозування;
- збережені прогнози залишаються доступними для перегляду у будь-який момент.

Ці функціональні можливості забезпечують ефективне та зручне використання програми для прогнозування витрат, допомагаючи користувачам не лише в плануванні своїх витрат, але й у гнучкому налаштуванні прогнозів під індивідуальні потреби.

5.1.3 Умови застосування програми

Програма призначена для використання користувачами, які мають попередньо зібрані дані про власні фінансові операції у вигляді транзакцій із зазначенням дати, суми та категорії витрат. Для коректної роботи системи

необхідно, щоб дані були структуровані відповідно до встановленого формату, який підтримується програмою (CSV або Excel-файл із відповідними стовпцями).

Застосунок реалізовано як вебінтерфейс, що не вимагає встановлення додаткового програмного забезпечення – користувачеві достатньо мати сучасний браузер і стабільне інтернет-з'єднання. Усі обчислення виконуються на стороні сервера після запуску алгоритму прогнозування, тому потужність локального пристрою не має критичного значення.

Для забезпечення коректності результатів рекомендується, щоб вхідні дані охоплювали значний період часу (наприклад, кілька місяців) і містили реальні, несуперечливі записи про витрати. У випадку відсутності достатнього обсягу даних або їх низької якості, точність прогнозу може бути зниженою.

Програма орієнтована на індивідуальне використання й не потребує технічних знань зі сторони користувача. Увесь процес взаємодії з системою зведено до завантаження файлу, внесення транзакцій та перегляду візуалізованого прогнозу, що робить застосунок доступним навіть для користувачів без досвіду роботи з фінансовими або аналітичними інструментами.

Крім того, оскільки система не вимагає збереження даних на сервері, рівень конфіденційності для користувача залишається високим. Такий підхід дозволяє мінімізувати ризики, пов'язані з обробкою персональної інформації, і водночас гарантує стабільну та безпечну роботу сервісу в різних умовах експлуатації.

5.2 Умови виконання програми

Для стабільної роботи вебзастосунку необхідно дотримуватись визначених технічних вимог, які гарантують надійну обробку даних, вчасне формування прогнозів та безперебійну взаємодію між компонентами системи. Оскільки система використовує сучасні інструменти контейнеризації, зокрема

Docker, до апаратного та програмного забезпечення висуваються помірні, але чітко сформульовані вимоги.

Мінімальні системні вимоги дозволяють запускати програму на базових машинах з обмеженими ресурсами. До них належать:

- процесор: щонайменше двоядерний з тактовою частотою 2.0 ГГц;
- оперативна пам'ять: не менше 4 Гб, необхідні для запуску контейнерів і базової обробки даних;
- дисковий простір: 500 Мб вільного місця для зберігання образів Docker, проміжних даних та логів;
- операційна система: будь-який дистрибутив Linux, який підтримує Docker (наприклад, Debian, Ubuntu, Fedora);
- графічний інтерфейс: базова або інтегрована відеокарта, що забезпечує роботу з вебінтерфейсом;
- мережа: доступ до Інтернету для початкового завантаження контейнерів і залежностей.

Для більш комфортної роботи, особливо при одночасній обробці великих обсягів даних або паралельному доступі кількох користувачів, рекомендовані системні характеристики включають:

- процесор: сучасний чотириядерний процесор з частотою 3.0 ГГц і більше;
- оперативна пам'ять: 8 Гб або більше для забезпечення достатнього буфера при обробці великих наборів транзакцій;
- дисковий простір: мінімум 1 Гб для зберігання історичних даних, конфігурацій, результатів прогнозування та логів;
- операційна система: рекомендовано Ubuntu 20.04 LTS або новіша, як стабільна і широко підтримувана платформа;
- відеокарта: дискретна відеокарта з підтримкою CUDA та не менш ніж 4 Гб відеопам'яті;
- мережа: стабільне з'єднання з Інтернетом для безперебійної роботи, оновлень та інтеграції з зовнішніми сервісами.

Окрім апаратних параметрів, застосунок вимагає попередньої установки Docker для ізоляції середовища виконання та спрощення розгортання та Docker Compose, що полегшує управління взаємодією між окремими контейнерами (серверна частина, інтерфейс, модель машинного навчання тощо).

При дотриманні зазначених умов забезпечується не лише стабільна технічна робота системи, а й зручність для кінцевого користувача – без тривалих очікувань, з мінімальними ресурсними витратами та можливістю швидкої адаптації до нових середовищ розгортання.

5.3 Експлуатація програми

Експлуатація програмного забезпечення побудована з урахуванням принципів зручності, передбачуваності та розділення відповідальностей між етапами взаємодії з системою. Основна ідея реалізації полягає у створенні простого, інтуїтивно зрозумілого процесу, який дозволяє користувачеві здійснювати фінансовий прогноз у зручний для нього момент, без потреби негайної обробки щойно завантажених даних.

Після входу до системи користувач має змогу завантажити файл з власними транзакціями. Формат вхідного файлу передбачає наявність базових колонок: дата операції, категорія витрат та сума. Після завантаження даних система зберігає їх для подальшої обробки, але не виконує жодних обчислювальних дій автоматично. Такий підхід дозволяє користувачеві самостійно обрати час, коли буде запущено прогноз, і уникнути зайвого навантаження на систему у випадку помилково завантажених або неповних даних.

Для ініціації прогнозування користувач повинен перейти до відповідного розділу інтерфейсу – сторінки прогнозування, де відображаються доступні параметри для запуску моделі. На поточному етапі реалізації система підтримує лише один алгоритм прогнозування витрат, однак структура

інтерфейсу спроектована з урахуванням майбутньої розширюваності. У перспективі користувач зможе обирати серед кількох моделей, які можуть ґрунтуватися на різних методах машинного навчання (наприклад, дерева рішень, рекурентні нейронні мережі тощо).

Крім вибору моделі, користувач також може зазначити категорії транзакцій, які мають бути враховані у прогнозі, а також вибрати часовий період, за який необхідно побудувати прогноз. Це дозволяє адаптувати результати аналітики до конкретних запитів користувача.

Запуск прогнозу здійснюється натисканням відповідної кнопки. Після обробки вхідних даних і побудови прогнозу, результат відображається у вигляді графіка. Візуалізація дозволяє швидко виявити потенційні пікові навантаження або зниження активності в окремих категоріях витрат. Під час перегляду графіка користувач може приховувати окремі категорії для кращої наочності або порівняння, а також має змогу зберегти побудований прогноз для подальшого перегляду без потреби повторного запуску.

Застосунок також дозволяє повторно запускати прогнозування без необхідності повторного імпорту даних, що значно підвищує зручність роботи із системою.

Загалом, такий підхід до експлуатації дозволяє забезпечити гнучкий, зручний сценарій використання додатку, який не перевантажує користувача зайвими діями, водночас надаючи йому повний контроль над кожним етапом процесу.

5.4 Методика та результати тестування

Тестування програми було проведено вручну, з акцентом на перевірку функціональних та нефункціональних можливостей системи. Метою тестування було оцінити ефективність роботи основних функцій, забезпечення точності прогнозів та зручність взаємодії користувача з інтерфейсом програми, а також стабільність і продуктивність застосунку.

Під час тестування було виконано наступні перевірки:

- тестування функціональності: Основна увага була приділена перевірці коректності завантаження файлів, процесу обробки транзакцій, ініціації прогнозування та коректності відображення результатів. Перевірялась правильність функціонування механізму фільтрації даних та запуску моделі прогнозування;

- користувацьке тестування: Оцінка зручності інтерфейсу та інтуїтивності взаємодії з програмою. Аналізувалися такі аспекти, як зрозумілість навігації, можливість легкого завантаження даних, доступність параметрів прогнозування та простота візуалізації результатів. Особливу увагу приділено тестуванню різних сценаріїв взаємодії користувача з програмою, зокрема завантаження даних та запуску прогнозування з мінімальними зусиллями з боку користувача;

- нефункціональне тестування: Оцінка продуктивності програми, зокрема швидкості завантаження та обробки великих файлів, а також стабільності її роботи в умовах різних конфігурацій обладнання. Перевірялися також параметри сумісності з різними браузерами.

Для того, щоб почати роботу з програмою, користувачам необхідно пройти процедуру реєстрації або авторизації. Ці етапи є важливими для збереження персональних налаштувань і даних про витрати, а також для доступу до прогнозів та їх збереження.

Користувачі можуть зареєструвати новий обліковий запис або увійти в уже існуючий, ввівши відповідні облікові дані. Інтерфейс сторінок реєстрації та авторизації зображено на рис. 5.1-5.2.

Budget Planner

Login Register

Register

Email

First name

Last name

Password

Password confirmation

Sign Up

[Already have an account? Login here.](#)

Рисунок 5.1 – Сторінка реєстрації

Budget Planner

Login Register

Login

Email

Password

Login

[Don't have an account? Register here.](#)

Рисунок 5.2 – Сторінка авторизації

Після успішної авторизації користувач автоматично перенаправляється на головну сторінку застосунку. На цій сторінці відображається останній збережений прогноз витрат, що дає змогу одразу переглянути результати попереднього аналізу. У верхній навігаційній панелі користувач має доступ до основних розділів програми, зокрема: головної сторінки, завантаження файлів, прогнозування та вихід з облікового запису. Внизу сторінки користувач може

додати транзакцію вручну за допомогою відповідної форми, а також переглядати та фільтрувати додані ним транзакції у вигляді таблиці. Така структура інтерфейсу забезпечує швидкий і зручний перехід між основними етапами роботи із застосунком. Інтерфейс головної сторінки показано на рис. 5.3-5.4.

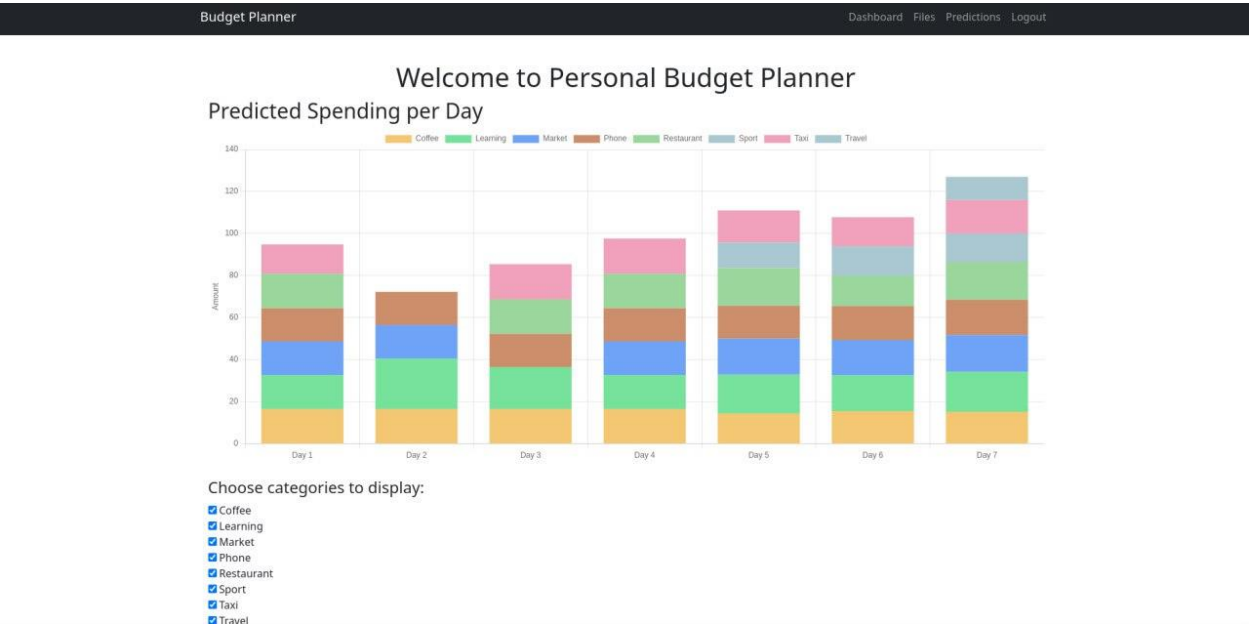


Рисунок 5.3 – Головна сторінка з графіком останнього прогнозування

Add Transaction

Date:

mm/dd/yyyy

Category:

Amount:

Add

Date from:

mm/dd/yyyy

Date to:

mm/dd/yyyy

Category:

All categories

Amount min:

Amount max:

Filter

Clear

Your Transactions

Date	Category	Amount	Actions
Feb. 15, 2025	Communal	20.00	Delete
Feb. 11, 2025	Rent Car	12.00	Delete

Рисунок 5.4 – Головна сторінка з формою для додавання транзакцій

Сторінка завантаження файлів призначена для додавання нових вхідних даних, на основі яких виконуватиметься прогнозування. Основним елементом інтерфейсу є кнопка, що дозволяє користувачу обрати та завантажити файл з локального пристрою. Після успішного завантаження дані автоматично зберігаються в системі, а файл відображається в списку завантажених. Інтерфейс сторінки зображено на рис. 5.5.

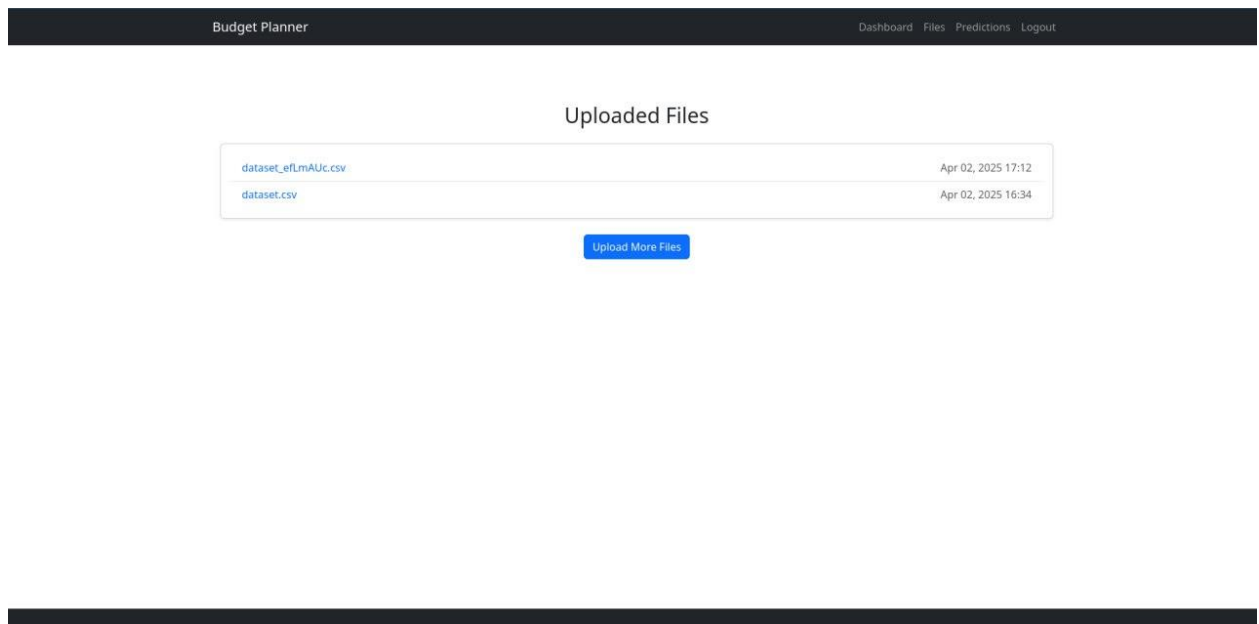


Рисунок 5.5 – Сторінка завантаження файлів

Сторінка прогнозування дозволяє користувачеві налаштувати параметри для прогнозу витрат. Є форма для вибору категорій витрат, два календарі для вибору дат транзакцій і список з радіокнопками для вибору алгоритму прогнозування. Нижче відображається список збережених прогнозів, де можна переглянути попередні результати прогнозування. Після завершення прогнозування, користувач отримує сповіщення про завершення обчислень. Інтерфейс сторінки прогнозування та сповіщення зображено на рис. 5.6–5.7

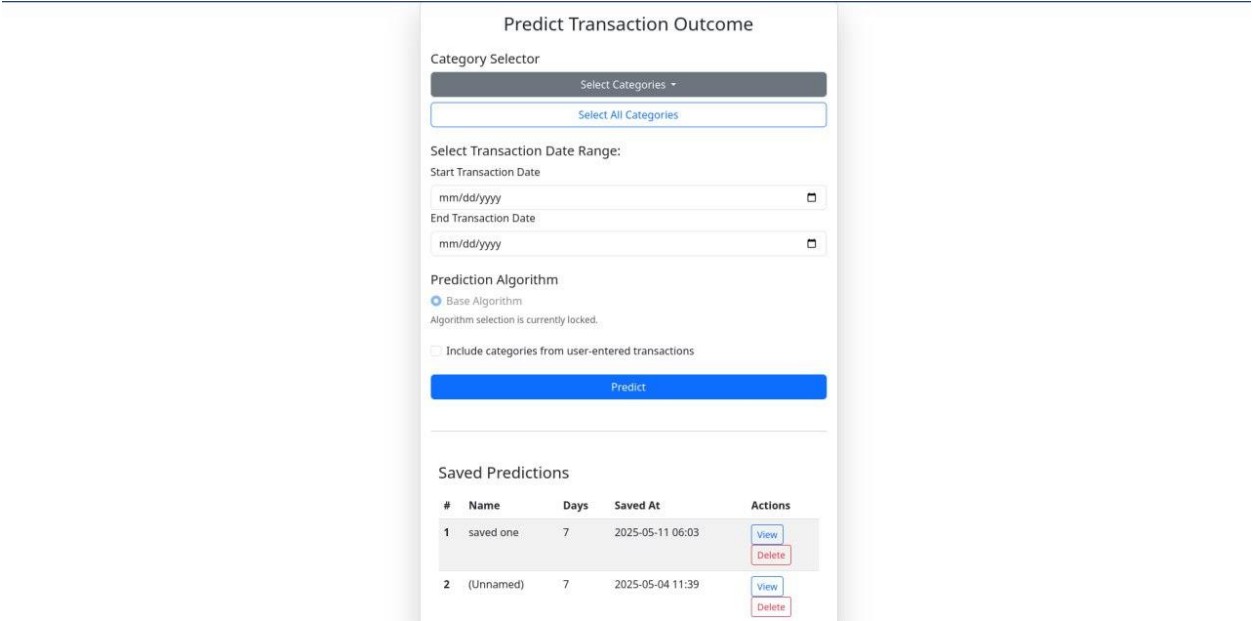


Рисунок 5.6 – Сторінка завантаження файлів

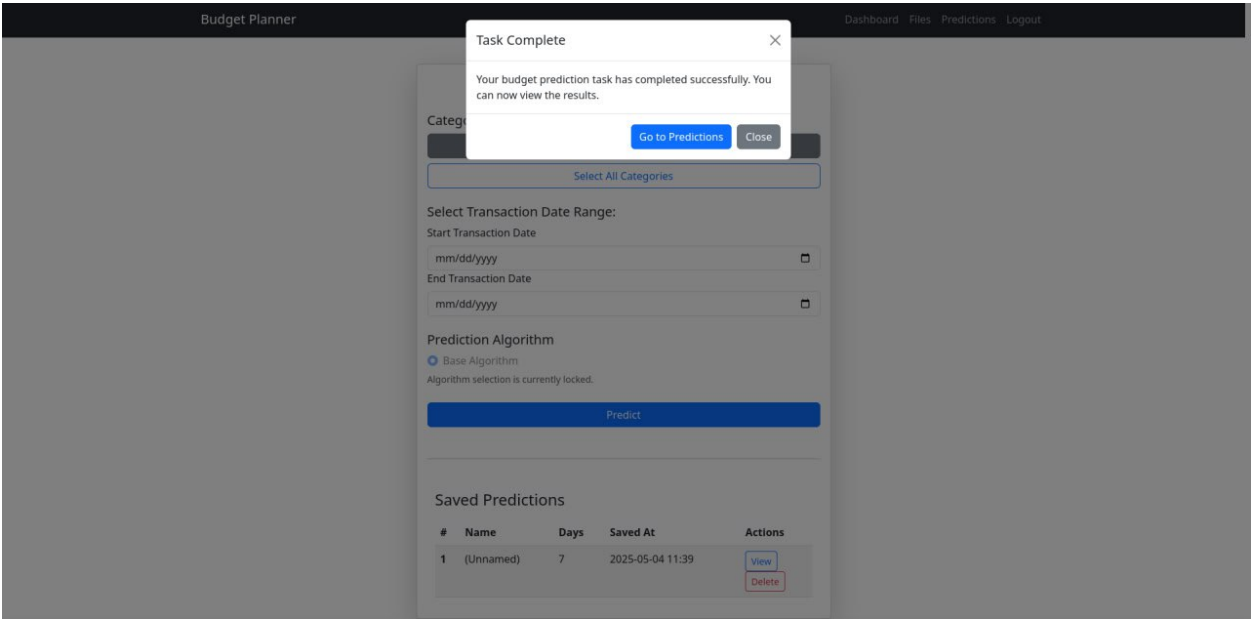


Рисунок 5.7 – Сповіщення про завершення прогнозування

Після переходу за посиланням із нотифікації користувач потрапляє на сторінку з результатами прогнозування. У центральній частині сторінки відображається графік прогнозованих витрат по категоріях. Для зручності передбачена можливість приховати окремі категорії з графіка, щоб зосередитись на найважливіших показниках. У нижній частині сторінки

розміщена форма для збереження поточного прогнозу для подальшого аналізу. Інтерфейс сторінки зображено на рис. 5.8.



Рисунок 5.8 – Сторінка результатів прогнозування і форма для збереження результатів

5.5 Тестування застосунку та його результати

У цьому розділі розглядається процес тестування розробленого застосунку, метою якого було перевірити коректність функціонування, стабільність роботи та обробку виняткових ситуацій. Особлива увага приділялася перевірці реакції системи на некоректні дії користувача та наявність необхідних вхідних даних. Результати тестування демонструють ефективність реалізованих механізмів валідації та обробки помилок, що забезпечує зручність і надійність роботи застосунку.

У випадку введення некоректних облікових даних інтерфейс відображає відповідне повідомлення про помилку, що інформує користувача про неможливість входу до системи. Приклади таких ситуацій наведено на рис. 5.9-5.10.

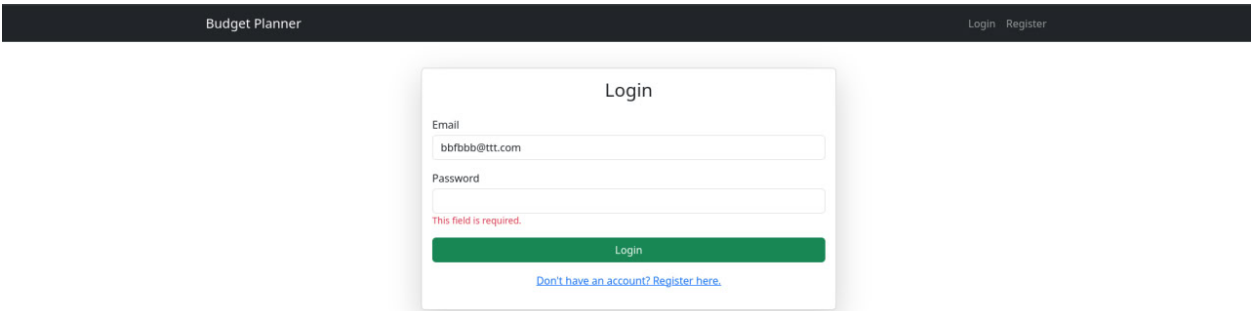


Рисунок 5.9 – Обробка випадку надсилання форми авторизації без введення пароля

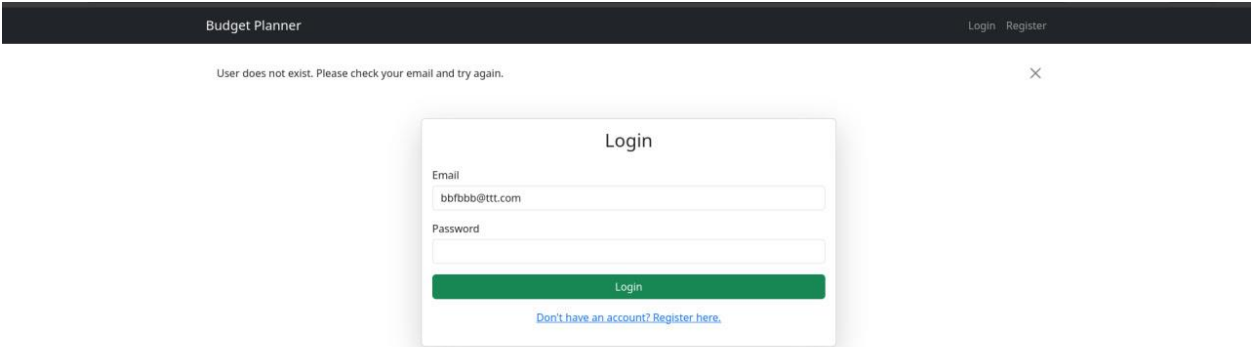


Рисунок 5.10 – Обробка випадку авторизації з неіснуючим обліковим записом

Помилки, що виникають під час реєстрації нового користувача, ілюструються на рис. 5.11-5.12. Зокрема, показано повідомлення у випадках, коли заповнено не всі обов’язкові поля або використано некоректний формат даних.

Budget Planner

Login Register

Register

Email

foo@example.com

First name

Anton

Last name

Antonov

Password

Password confirmation

The two password fields didn't match.

Sign Up

[Already have an account? Login here.](#)

Рисунок 5.11 – Обробка ситуації, коли під час реєстрації введені паролі не збігаються

Budget Planner

Login Register

Register

Email

test@example.com

Custom user with this Email already exists.

First name

Anton

Last name

Antonov

Password

Password confirmation

The two password fields didn't match.

Sign Up

[Already have an account? Login here.](#)

Рисунок 5.12 – Обробка ситуації, коли користувач намагається зареєструватися повторно

Головна сторінка відображає порожній екран у разі, якщо користувач ще не створив жодного прогнозу. Така поведінка показана на рис. 5.13.

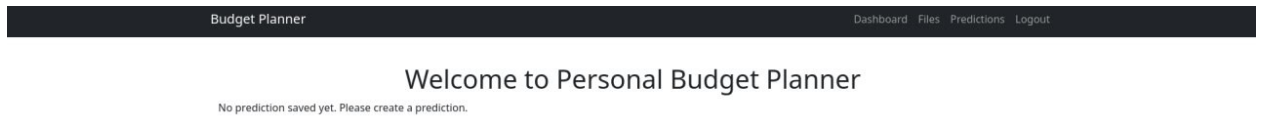


Рисунок 5.13 – Порожній екран, якщо користувач ще не створив жодного прогнозу

Кнопка прогнозування стає неактивною, якщо у користувача відсутні транзакції, що запобігає виконанню прогнозу на неповних даних. Це показано на рис. 5.14.

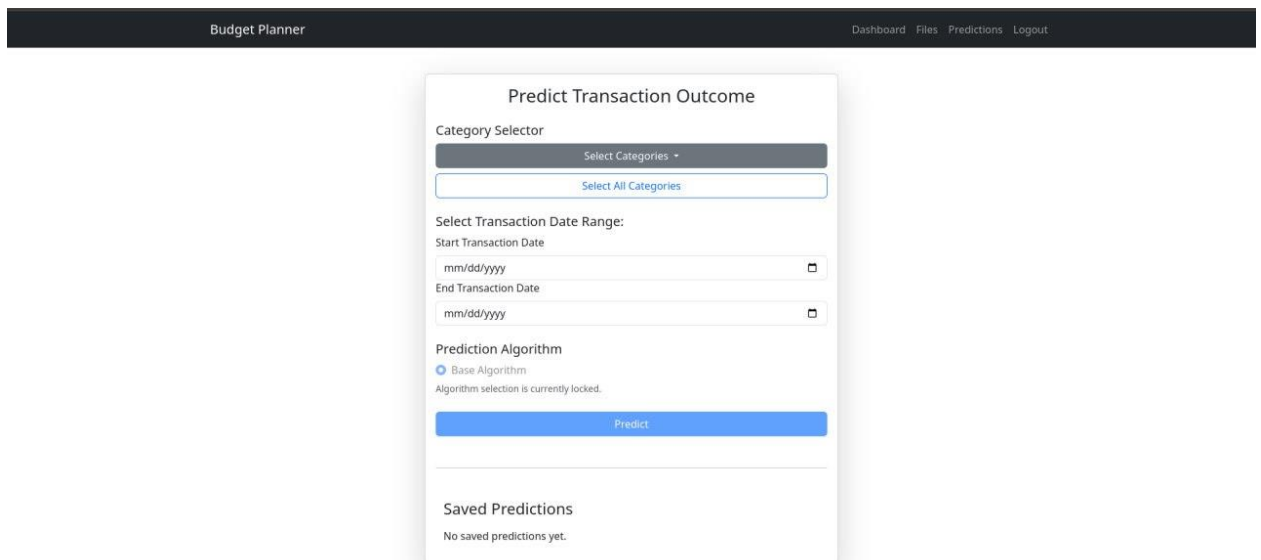


Рисунок 5.14 – Вимкнена кнопка прогнозування у разі відсутності транзакцій користувача

Перехоплення помилки відбувається у випадку, коли користувач не завантажив жодного файлу, і система відображає порожній список. Це продемонстровано на рис. 5.15.

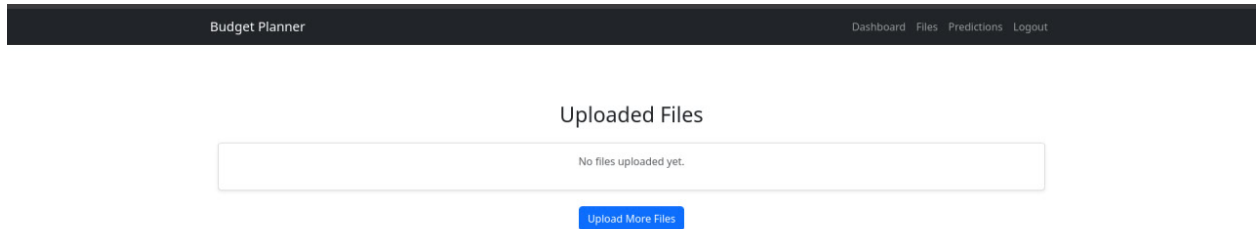


Рисунок 5.15 – Порожній список, якщо користувач не завантажив жодного файлу

Під час введення нової транзакції застосунок виконує перевірку заповнення обов'язкових полів форми. Якщо деякі поля залишаються незаповненими, система перехоплює цю помилку і відображає відповідне повідомлення для користувача. Цей процес показано на рисунку 5.16.

☒ Sport

☒ Taxi

☒ Travel

Return

Add Transaction

Date:

mm/dd/yyyy

Category:

Please fill out this field.

Amount:

Add

Date from:

02/05/2025

Date to:

02/12/2025

Category:

All categories

Amount min:

Amount max:

Filter

Clear

Your Transactions

Date	Category	Amount	Actions
Feb. 11, 2025	Rent Car	12.00	Delete

Рисунок 5.16 – Обробка ситуації, коли форма додавання транзакцій має незаповнені поля

В табл. 5.1 наведений чек-лист тестування програми.

Таблиця 5.1 – Результати тестування програми

Тестування	Результат	Коментарі
1	2	3
Реєстрація користувача	Успішно	Новий користувач
Авторизація користувача	Успішно	Зареєстрований користувач
Імпорт файлу транзакцій	Успішно	Файл імпортовано без помилок, всі транзакції коректно зчитані
Ініціація процесу прогнозування майбутніх витрат	Успішно	Прогнозування розпочато без помилок, модель отримала коректні вхідні дані

Продовження таблиці 5.1

1	2	3
Ініціація прогнозування при відсутності вхідних даних	Успішно	Кнопка створення прогнозування стає неактивною, якщо у користувача нема транзакцій
Відображення результатів прогнозування у вигляді графіків за категоріями витрат	Успішно	Графіки відображають правильні значення прогнозованих витрат по категоріях
Виведення повідомлення після завершення прогнозування	Успішно	Користувач отримує повідомлення про успішне завершення процесу прогнозування
Можливість приховувати окремі категорії на графіку для фокусування на потрібних даних	Успішно	Категорії успішно приховані на графіку без порушення інших даних
Фільтрація категорій витрат, які будуть враховані при прогнозуванні	Успішно	Модель враховує лише вибрані категорії
Фільтрація транзакцій за датами для обмеження періоду аналізу	Успішно	Модель враховує транзакції лише за обраний період

Кінець таблиці 5.1

1	2	3
Збереження результатів прогнозування для подальшого аналізу та порівняння	Успішно	Прогнози збережені коректно, дані доступні для подальшого використання
Перегляд доданих користувачем транзакцій у табличному вигляді	Успішно	Таблиця відображає всі додані транзакції коректно і повністю
Створення транзакції через відповідну форму	Успішно	Транзакція успішно створена і відображається у таблиці
Відправка пустої форми для створення транзакції	Успішно	Форма не відправлена, поява повідомлень, що треба заповнити дані

Оцінювання точності побудованої моделі здійснювалося на основі стандартних метрик для регресійних задач: Mean Absolute Error (MAE), Root Mean Scaled Error (RMSE) та Mean Absolute Scaled Error (MASE). Аналіз проводився окремо для кожної категорії витрат, а також усереднено по всіх категоріях для загальної оцінки. Загальні метрики прогнозування зображені на рис. 5.17.

```

Restaurant : 2.9000307, Transport : 0.09102104, Business lunch : 0.370972007}
Metrics for all 7 recursive steps (averaged):
Market: {'MAE': 5.662328156062535, 'RMSE': 6.473257575323223, 'MASE': 0.6418967518385263}
Coffee: {'MAE': 8.050545147487096, 'RMSE': 8.067823646730304, 'MASE': 0.8504443020457466}
Restaurant: {'MAE': 15.844567469188146, 'RMSE': 25.99632061068587, 'MASE': 0.877954447012187}
Transport: {'MAE': 1.0775634880576814, 'RMSE': 1.230363101993339, 'MASE': 0.6829627741210658}
Business lunch: {'MAE': 0.3771321049758366, 'RMSE': 0.3771362848049151, 'MASE': 0.11800347239493139}
Average: {'MAE': 6.2024272731542585, 'RMSE': 8.42898024390753, 'MASE': 0.6342523494824914}

```

Рисунок 5.17 – Загальні значення MAE, RMSE та MASE при рекурсивному прогнозуванні для гібридної моделі LSTM та TCN

Табл. 5.2 відображає загальні значення MAE, RMSE та MASE, представлені на рис. 5.17, для гібридної моделі LSTM та TCN, з метою зручності порівняння.

Таблиця 5.2 – Загальні значення метрик MAE, RMSE та MASE для рекурсивного прогнозування

Категорія	MAE	RMSE	MASE
Market	5.662	6.473	0.641
Coffee	8.051	8.068	0.850
Restaurant	15.845	25.996	0.878
Transport	1.078	1.230	0.683
Business lunch	0.377	0.377	0.118
Average	6.202	8.429	0.634

За результатами тестування на 7 кроків наперед (рекурсивне прогнозування), модель показала стабільно добру якість передбачень. Усі значення MASE були меншими за 1, що вказує на перевагу моделі над наївним прогнозом ($MASE = 1$). Найнижчі значення помилок були зафіксовані у категоріях Business lunch ($MASE \approx 0.12$) та Transport ($MASE \approx 0.68$), що свідчить про високу точність моделі для регулярних та стабільних витрат. Також добрі результати отримано в категоріях Market та Coffee ($MASE \approx 0.64$ – 0.85), які є типовими щоденними витратами користувача. Найбільші

відхилення спостерігалися у категорії Restaurant, що може бути пов'язано з високою варіативністю сум у цих транзакціях.

Оцінка на внутрішньому тестовому наборі (direct one-step evaluation) підтвердила узагальнюючу здатність моделі: середнє значення MASE склало ≈ 0.83 . Це означає, що модель забезпечує прийнятну точність навіть без адаптації на попередні кроки прогнозу, що є важливим показником її надійності.

Для обґрунтування доцільності використання гібридного підходу було проведено порівняльне прогнозування з використанням лише LSTM-моделі. Це дало змогу продемонструвати переваги поєднання LSTM і TCN, зокрема покращення точності завдяки одночасному врахуванню довготривалих і локальних залежностей у даних. Загальні метрики прогнозування LSTM моделі зображені на рис. 5.18.

```
Metrics for all 7 recursive steps (averaged):
Market: {'MAE': 6.784392267274845, 'RMSE': 7.524381689157416, 'MASE': 0.813215430942182}
Coffee: {'MAE': 8.691754926108235, 'RMSE': 9.204683512391276, 'MASE': 0.950142380159235}
Restaurant: {'MAE': 18.316790582237451, 'RMSE': 28.725168934820435, 'MASE': 1.049682395152873}
Transport: {'MAE': 1.302376427945732, 'RMSE': 1.482924760183727, 'MASE': 0.854374190519340}
Business lunch: {'MAE': 0.441682350128765, 'RMSE': 0.453912770384127, 'MASE': 0.167920584628374}
Average: {'MAE': 7.107399710338206, 'RMSE': 9.278214713587796, 'MASE': 0.767066196280801}
```

Рисунок 5.18 – Загальні значення MAE, RMSE та MASE при рекурсивному прогнозуванні для моделі LSTM

Табл. 5.3 відображає загальні значення MAE, RMSE та MASE, представлені на рис. 5.18, для моделі LSTM, з метою зручності порівняння.

Таблиця 5.3 – Загальні значення метрик MAE, RMSE та MASE для рекурсивного прогнозування

Категорія	MAE	RMSE	MASE
1	2	3	4
Market	6.784	7.524	0.813

Кінець таблиці 5.3

1	2	3	4
Coffee	8.691	9.204	0.950
Restaurant	18.316	28.725	1.049
Transport	1.302	1.482	0.854
Business lunch	0.441	0.453	0.167
Average	7.107	9.278	0.767

Порівняння результатів двох моделей – базової LSTM та гібридної LSTM та TCN засвідчує перевагу останньої за всіма ключовими метриками. Гібридна архітектура забезпечила зниження помилок MAE, RMSE та MASE відповідно на 12,7%, 9,2% та 17,3% порівняно з чистою LSTM.

Зокрема, найбільше покращення спостерігається в категоріях з вищими витратами, як-от Restaurant (MAE зменшено з 18.31 до 15.84, RMSE з 28.73 до 26.00), що свідчить про кращу здатність гібридної моделі враховувати як глобальні тренди, так і локальні флуктуації. Також покращення показників MASE свідчить про більш стабільне узгодження з наївними базовими прогнозами.

Таким чином, поєднання LSTM для моделювання довгострокових залежностей і TCN для виявлення локальних патернів виявилось ефективним підходом, що підтверджує доцільність використання гібридної архітектури у задачах багатокрокового рекурсивного прогнозування витрат. Отримані результати демонструють, що побудована модель забезпечує високу точність прогнозування, особливо в категоріях зі сталою поведінкою, і може бути успішно застосована в реальному середовищі для підтримки прийняття фінансових рішень користувачем.

5.6 Висновки до п'ятого розділу

У результаті проведеного дослідження було підтверджено працездатність програмного продукту відповідно до визначених функціональних вимог. Застосунок успішно реалізує механізм завантаження вхідних даних, додавання і видалення транзакцій вручну, запуску прогнозування та відображення результатів у зручному графічному форматі.

Під час тестування перевірено стабільність роботи, коректність обробки даних та зручність користування інтерфейсом. Застосунок демонструє стабільну роботу при типових сценаріях використання, а також адекватно реагує на можливі виняткові ситуації.

Окрему увагу приділено аналізу умов експлуатації, зокрема процедурі запуску прогнозування та взаємодії з користувачем. Реалізований функціонал дозволяє гнучко керувати параметрами аналізу та отримувати результати у зрозумілому вигляді. Загалом, програмне забезпечення відповідає заявленим цілям і готове до практичного використання.

Якість побудованих прогнозів проаналізовано за допомогою метрик MAE, RMSE та MASE як для прямого однокрокового прогнозування, так і для рекурсивного багатоетапного. Отримані значення MASE нижчі за 1 в усіх категоріях витрат, що свідчить про кращу точність моделі порівняно з наївним підходом. Додаткове порівняння з LSTM-моделлю показало перевагу гібридної архітектури LSTM та TCN за всіма метриками, що підтверджує доцільність її використання. Це підтверджує ефективність обраного підходу до моделювання часових рядів і придатність розробленого застосунку для реального використання в задачах фінансового планування. Загалом, програмне забезпечення відповідає заявленим цілям і готове до практичного використання.

ВИСНОВКИ

У ході виконання дипломної кваліфікаційної роботи бакалавра було розроблено програмне забезпечення для прогнозування особистих витрат на основі транзакцій користувача. Для досягнення поставленої мети було виконано ряд завдань:

- аналіз предметної області управління персональними фінансами та підходів до прогнозування витрат на основі історичних даних;
- експериментальне дослідження ефективності моделей машинного навчання для прогнозування витрат у різних категоріях;
- проектування та розробка програмного забезпечення з інтегрованим вебінтерфейсом для взаємодії з користувачем;
- реалізація функціоналу завантаження, обробки та візуалізації транзакцій, запуску прогнозування та роботи з результатами;
- проведення комплексного тестування програмного забезпечення для перевірки коректності, стабільності та зручності використання.

Розроблений застосунок повністю відповідає вимогам технічного завдання, забезпечує точне прогнозування витрат та може використовуватись як основа для персональних фінансових асистентів або аналітичних платформ у сфері фінансового планування.

Наукова новизна роботи полягає в запропонованому методі прогнозування витрат на основі поведінкових патернів користувачів, який базується на багатоступеневій обробці та відборі ознак із подальшим застосуванням гібридних нейронних архітектур, що дозволяє виконувати розподілення витрат за категоріями, забезпечуючи гнучкість, точність і адаптивність прогнозування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ResearchGate. Digital Economy and Smart Financial Management [Electronic resource] : Using AI Tools for Financial Literacy. – Access mode : https://www.researchgate.net/publication/390725459_Digital_Economy_and_Smart_Financial_Management_Using_AI_Tools_for_Financial_Literacy.
2. Financial Health Network. Consumer Financial Health Study [Electronic resource]. – 2022. – Access mode : <https://finhealthnetwork.org/research/consumer-financial-health-study>
3. Lukas, M. The Influence of Budgets on Consumer Spending [Electronic resource] / M. Lukas, R. Howard // Journal of Consumer Research. – 2022. – Vol. 49, No. 1. – P. 74–91. – Access mode : <https://doi.org/10.1093/jcr/ucac024>.
4. French, D. Personal finance apps and low-income households [Electronic resource] / D. French, D. McKillop, C. Stewart. – 2021. – Access mode : https://www.researchgate.net/publication/353109750_Personal_finance_apps_and_low-income_households.
5. YNAB. YNAB Features [Electronic resource]. – Access mode : <https://www.ynab.com/features>
6. PocketGuard. PocketGuard Features [Electronic resource]. – Access mode : <https://pocketguard.com/>
7. Quicken. Quicken Simplifi [Electronic resource]. – Access mode : <https://www.quicken.com/simplifi>
8. Investopedia. Regression: Definition, Analysis, Calculation, and Example [Electronic resource]. – Access mode : <https://www.investopedia.com/terms/r/regression.asp>
9. Forecasting of Personal Financial Expenses Using Machine Learning Techniques. Decision Support Systems [Electronic resource]. – Access mode: <https://www.sciencedirect.com/science/article/abs/pii/S1062940820301480>
10. Min, J. Financial Market Trend Forecasting and Performance Analysis Using LSTM [Electronic resource] / J. Min. – 2020. – Access mode :

https://www.researchgate.net/publication/340452280_Financial_Market_Trend_Forecasting_and_Performance_Analysis_Using_LSTM

11. Gao, Y. Stock Prediction Based on Optimized LSTM and GRU Models [Electronic resource] / Y. Gao, R. Wang, E. Zhou // Scientific Programming. – 2021. – Access mode : https://www.researchgate.net/publication/354944046_Stock_Prediction_Based_on_Optimized_LSTM_and_GRU_Models

12. Masini, R. P. Machine Learning Advances for Time Series Forecasting [Electronic resource] / R. P. Masini, M. C. Medeiros, E. F. Mendes. – 2020. – Access mode : https://www.researchgate.net/publication/352956452_Machine_Learning_Advances_for_Time_Series_Forecasting

13. Mohammed, A. Leveraging Hybrid Deep Learning Models for Enhanced Multivariate Time Series Forecasting [Electronic resource] / A. Mohammed, A. Mohammed // Neural Processing Letters. – 2024. – Access mode : https://www.researchgate.net/publication/383343470_Leveraging_Hybrid_Deep_Learning_Models_for_Enhanced_Multivariate_Time_Series_Forecasting

14. Personal Budget Transactions Dataset [Electronic resource]. – Access mode : <https://www.kaggle.com/datasets/ismetsemedov/personal-budget-transactions-dataset>.

15. Python documentation [Electronic resource]. – Access mode : <https://www.python.org/doc>.

16. Pandas documentation [Electronic resource]. – Access mode : <https://pandas.pydata.org>.

17. NumPy documentation [Electronic resource]. – Access mode : <https://numpy.org/doc>.

18. TensorFlow documentation [Electronic resource]. – Access mode : <https://www.tensorflow.org>.

19. TCN for Keras [Electronic resource]. – Access mode : <https://github.com/philipperemy/keras-tcn>.

20. Django documentation [Electronic resource]. – Access mode :
<https://docs.djangoproject.com>

21. Chart.js documentation [Electronic resource]. – Access mode :
<https://www.chartjs.org>.

22. PostgreSQL documentation [Electronic resource]. – Access mode :
<https://www.postgresql.org/docs>.

23. Gunicorn documentation [Electronic resource]. – Access mode :
<https://gunicorn.org>.

24. Nginx documentation [Electronic resource]. – Access mode :
<https://nginx.org/en/docs/>.

25. Docker documentation [Electronic resource]. – Access mode :
<https://docs.docker.com>.

26. Celery documentation [Electronic resource]. – Access mode :
<https://docs.celeryq.dev/en/stable/>.

27. Flask documentation [Electronic resource]. – Access mode :
<https://flask.palletsprojects.com>.

28. FastAPI documentation [Electronic resource]. – Access mode :
<https://fastapi.tiangolo.com>.

ДОДАТОК А
Технічне завдання

Вступ

Розроблюваний проєкт за темою «Дослідження та програмна реалізація методів фінансового планування на основі прогнозування витрат користувача» передбачає створення програмного забезпечення для автоматизації процесу фінансового планування. Метою проєкту є розробка програмного забезпечення фінансового планування на основі поведінкових патернів користувача для надання можливостей прогнозування витрат користувача з використанням машинного навчання.

Це дасть змогу більш ефективно планувати бюджет, уникати несподіваних витрат та приймати обґрунтовані фінансові рішення.

Актуальність розробки зумовлена зростаючою потребою в персональних фінансових сервісах, які допомагають користувачам управляти своїми витратами в умовах сучасної економічної ситуації. Прогнозування витрат на основі аналізу даних дає можливість отримати точнішу картину майбутніх фінансових потреб, що є важливим інструментом для прийняття фінансових рішень.

Очікуваним результатом є створення програмного продукту, здатного автоматично прогнозувати майбутні витрати користувача на основі його фінансової активності, що сприятиме кращому плануванню бюджету та підвищенню фінансової грамотності.

A.1 Підстава для розробки

Основна підстава для виконання роботи сформульована у відповідності з Наказом № 209 від 28 квітня 2025 року, виданим у Національному університеті «Запорізька політехніка», де для виконання даної дипломної кваліфікаційної роботи бакалавра визначено «Дослідження та програмна реалізація методів фінансового планування на основі поведінкових патернів користувача» як тему роботи.

A.2 Основні вимоги до програми, що розробляється

A.2.1 Вимоги до функціональних характеристик

Функціональні вимоги визначають основні можливості, які має надавати програмне забезпечення, щоб задовольнити потреби користувача. Вони є критично важливими для створення ефективного та зручного інтерфейсу програми, який дозволить користувачеві зручно працювати з фінансовими даними.

Функціональні можливості програми повинні забезпечувати реалізацію таких дій користувача:

- завантаження файлу з транзакціями – можливість завантажити файл, що містить фінансові транзакції, у підтримуваних форматах (CSV, Excel), для подальшого імпорту даних про витрати та їх аналізу;
- прогнозування бюджету – ініціювання процесу прогнозування бюджету на основі історичних даних, що включає оцінку майбутніх витрат з урахуванням трендів, виявлених у попередніх транзакціях;
- перегляд прогнозу у вигляді графіку – результат прогнозування надається у вигляді графіку, який наочно демонструє прогнозовані витрати по різних категоріях;
- приховування категорій на графіку – можливість приховати певні категорії витрат на графіку для зручнішого перегляду та фокусування на найбільш актуальних категоріях;
- фільтрація категорій витрат при прогнозуванні – можливість застосування фільтрів для вибору лише тих категорій витрат, які будуть враховані при прогнозуванні, що дозволяє адаптувати модель до індивідуальних потреб;
- фільтрація дат транзакцій при прогнозуванні – можливість встановлення фільтрів для вибору конкретного періоду дат транзакцій, які братимуть участь у прогнозуванні;

- збереження результатів прогнозування – можливість зберегти результати прогнозування для подальшого перегляду та аналізу, що дозволяє порівнювати прогнози та оцінювати зміни у фінансовому стані;
- можливість додавати і видаляти транзакції вручну;
- перегляд особисто доданих транзакцій у таблиці;
- можливість фільтрувати таблицю транзакцій за датою, сумою транзакції та категорією;
- можливість враховувати користувацькі транзакції у прогнозі;
- перегляд збережених прогнозів – доступ до збережених прогнозів та можливість перегляду більш старих прогнозів.

A.2.2 Вимоги до інтерфейсу програми

Інтерфейс програми повинен забезпечувати такі можливості:

- інтуїтивно зрозумілий та зручний інтерфейс із чітко структурованими елементами, що забезпечує швидкий доступ до основних функцій без зайвих труднощів;
- мінімалістичний дизайн для уникнення перевантаження зайвими інструментами та елементами;
- спрощена навігація завдяки логічному групуванню елементів за їх функціональним призначенням;
- читабельність тексту та контенту, з використанням чітких шрифтів і контрастних кольорів для забезпечення зручності читання;
- наявність інформаційних повідомлень, щоб користувач був поінформований про стан операцій.

A.2.3 Вимоги до надійності

Програмне забезпечення має забезпечувати стабільну та безперебійну роботу в умовах штатного використання. У випадку некоректного введення

даних або помилкових дій користувача програма повинна обробляти виняткові ситуації без аварійного завершення роботи.

З метою підвищення надійності необхідно врахувати:

- програмне забезпечення повинно забезпечувати стабільну та безперебійну роботу навіть при високих навантаженнях;
- у разі некоректного введення даних програма повинна надавати чітке повідомлення про помилку без аварійного завершення роботи;
- всі функції програми повинні бути захищені від потенційних збоїв через некоректні дії користувача, щоб забезпечити безпеку даних.

A.2.4 Вимоги до складу та параметрів технічний засобів

Для коректної роботи програми рекомендується наступна конфігурація апаратного забезпечення.

Мінімальні вимоги:

- процесор: 2-ядерний, 2.0 ГГц або аналогічний;
- оперативна пам'ять: 4 Гб;
- місце на диску: 500 МБ вільного простору для контейнерів та даних;
- операційна система: Linux (будь-яка дистрибуція, що підтримує Docker);
- відеокарта: вбудована або базова відеокарта, що підтримує роботу з графічними елементами;
- мережа: підключення до мережі для завантаження образів Docker.

Рекомендовані вимоги:

- процесор: 4-ядерний, 3.0 ГГц або вище;
- оперативна пам'ять: 8 Гб або більше;
- місце на диску: 1 Гб вільного простору для зберігання даних, контейнерів та логів;
- операційна система: Linux (рекомендовано Ubuntu 20.04 або новіша);

- відеокарта: дискретна відеокарта з підтримкою CUDA та не менш ніж 4 ГБ відеопам'яті;
- мережа: стабільне підключення до інтернету для роботи з Docker та оновленнями.

Додаткові вимоги:

- Docker: підтримка Docker для створення та керування контейнерами;
- Docker Compose: для зручного налаштування та керування багатоконтейнерними додатками.

А.3 Порядок контролю та приймання

Сумарно 8 тижнів виділено на виконання дипломної кваліфікаційної роботи бакалавра, розподіл між якими має виконуватися за календарним планом, визначеним у завданні на роботі, з забезпеченням результатів, які мають бути досягнуті та які зазначені в календарному плані.

У підсумку робота оцінюється керівником зі створенням відгуку, рецензентом зі створенням рецензії та екзаменаційною комісією за наперед затвердженим графіком.

ДОДАТОК Б
Текст програми

Б.1 Текст файлы data_preparation.py

```

class DataPreprocessor:
    """
    Class for processing spending data, including data preparation,
    feature engineering, and category analysis.
    """

    def __init__(self, logger: logging.Logger | None = None):
        self.logger = logger if logger else get_logger(self.__class__.__name__,
                                                         log_file_name='data_formatting.log')
        self.df: pd.DataFrame = pd.DataFrame()

    def _load_and_prepare_initial_df(self, init_data_file_path: pathlib.Path) ->
    tuple[pd.DataFrame, list[str]]:
        self.logger.info(f"Loading initial dataset from: {init_data_file_path}")
        try:
            initial_df = pd.read_csv(init_data_file_path)
        except FileNotFoundError:
            self.logger.error(f"File not found: {init_data_file_path}")
            raise
        except Exception as e:
            self.logger.error(f"Error reading CSV file {init_data_file_path}: {e}")
            raise

        self.logger.info("Creating temporal features from date...")
        initial_df['date'] = pd.to_datetime(initial_df['date'])
        initial_df['month'] = initial_df['date'].dt.month
        initial_df['day_of_week'] = initial_df['date'].dt.dayofweek # Monday=0, Sunday=6
        initial_df['week_of_month'] = (initial_df['date'].dt.day - 1) // 7 + 1
        initial_df['year'] = initial_df['date'].dt.year
        initial_df['day'] = initial_df['date'].dt.day

        initial_df['month_period'] = initial_df[['month', 'year']].apply(lambda x: f'{x.iloc[0]}-{x.iloc[1]}', axis=1)

        self.logger.info(f"Years: {initial_df['year'].nunique()} ({','.join(map(str, sorted(initial_df['year'].unique()))})")
        self.logger.info(f"Sequential months: {initial_df['month_period'].nunique()}")

        categories = list(initial_df["category"].unique())
        self.logger.info(f"Found categories: {len(categories)}")
        self.logger.info(f"Initial dataframe length: {len(initial_df)}")
        return initial_df, categories

    def _create_pivot_table(self, initial_df: pd.DataFrame) -> pd.DataFrame:
        """
        Creates a pivot table with expense sums by category.
        """
        self.logger.info("Creating pivot table...")
        initial_df_copy = initial_df.copy() # Work with a copy to avoid SettingWithCopyWarning

```

```

initial_df_copy['date_agg'] = initial_df_copy['date'].dt.date

# Use all temporal features in the index to retain them
df_pivoted = initial_df_copy.pivot_table(
    index=['date_agg', 'year', 'month', 'week_of_month', 'day_of_week'],
    columns='category',
    values='amount',
    aggfunc='sum'
).reset_index()

df_pivoted.rename(columns={'date_agg': 'date'}, inplace=True)

# Add 'month_period' again if lost during pivot_table
df_pivoted['month_period'] = df_pivoted[['month', 'year']].apply(lambda x: f'{x.iloc[0]}-{x.iloc[1]}', axis=1)

df_pivoted.fillna(0, inplace=True)
self.logger.info(f"Number of unique dates in pivot table: {len(df_pivoted['date'].unique())}")
self.logger.info(f"Pivot table length: {len(df_pivoted)}")
return df_pivoted

def _classify_spending_categories(self, df_pivoted: pd.DataFrame, categories: list[str],
                                initial_month_period_nunique: int) -> tuple[
    dict[str, list[str]], list[str], list[str], list[str]]:
    """
    Classifies spending categories into regular, periodic, and rare
    based on the number of zero values.
    """
    self.logger.info("Classifying spending categories by frequency...")
    threshold_80 = 0.80 * len(df_pivoted)
    threshold_95 = len(df_pivoted) - initial_month_period_nunique + SKIPPED_PERIODS

    zero_values = {}
    for category in categories:
        if category in df_pivoted.columns:
            zero_values[category] = len(df_pivoted.loc[df_pivoted[category] == 0])
        else:
            zero_values[category] = len(df_pivoted)

    zero_values_series = pd.Series(zero_values).sort_values(ascending=True)

    above_95_nan_cols = zero_values_series.loc[zero_values_series >
threshold_95].index.to_list()
    above_80_nan_cols = zero_values_series.loc[
        zero_values_series.between(threshold_80, threshold_95, inclusive='right')].index.to_list()
    regular_cols = zero_values_series.loc[zero_values_series < threshold_80].index.to_list()

    self.logger.info(f"Regular categories (<80% zeros): {len(regular_cols)}")
    self.logger.info(f"Periodic categories (80–95% zeros): {len(above_80_nan_cols)}")
    self.logger.info(f"Rare categories (>95% zeros): {len(above_95_nan_cols)}")

```

```

classified_cols = {
    'regular': regular_cols,
    'above_80_nan': above_80_nan_cols,
    'above_95_nan': above_95_nan_cols
}
return classified_cols, regular_cols, above_80_nan_cols, above_95_nan_cols

def _engineer_temporal_features(self, df_target: pd.DataFrame) -> list[str]:
    """
    Creates cyclical temporal features (sin, cos) and encodes the year.
    Modifies the DataFrame `df_target` in place.
    """
    self.logger.info("Creating cyclical temporal features...")
    new_temporal_cols: list[str] = []
    for col, period in TEMPORAL_COLUMNS_CONFIG.items():
        if col not in df_target.columns:
            self.logger.warning(f"Column '{col}' not found in DataFrame. Skipping cyclical
features for it.")
            continue

        values = df_target[col]
        sin_col_name = f'{col}_sin'
        cos_col_name = f'{col}_cos'
        df_target[sin_col_name] = np.sin(2 * np.pi * values / period)
        df_target[cos_col_name] = np.cos(2 * np.pi * values / period)
        new_temporal_cols.extend([sin_col_name, cos_col_name])

    if 'year' in df_target.columns:
        self.logger.info("Encoding year...")
        df_target['encoded_year'] = LabelEncoder().fit_transform(df_target['year'])
        new_temporal_cols.append('encoded_year')
    else:
        self.logger.warning("Column 'year' not found for encoding.")

    self.logger.info(f"New temporal columns created: {new_temporal_cols}")
    return new_temporal_cols

def _engineer_aggregated_spend_features(self, df_target: pd.DataFrame, periodical_cols:
list[str],
                                occasional_cols: list[str]) -> list[str]:
    """
    Creates aggregated features for periodic and rare expenses.
    Modifies the DataFrame `df_target` in place.
    """
    self.logger.info("Creating aggregated spending features...")
    subtotals_columns: list[str] = []

    if periodical_cols:
        df_target['periodical_sum'] = df_target[periodical_cols].sum(axis=1)
        subtotals_columns.append('periodical_sum')
        self.logger.info("'periodical_sum' created.")
    else:

```

```

df_target['periodical_sum'] = 0
subtotals_columns.append('periodical_sum')
self.logger.info("No columns for 'periodical_sum', zero-filled column created.")

if occasional_cols:
    df_target['occasional_sum'] = df_target[occasional_cols].sum(axis=1)
    subtotals_columns.append('occasional_sum')
    self.logger.info("'occasional_sum' created.")
else:
    df_target['occasional_sum'] = 0
    subtotals_columns.append('occasional_sum')
    self.logger.info("No columns for 'occasional_sum', zero-filled column created.")

return subtotals_columns

def _find_correlated_aux_features(
    self,
    df_corr_analysis: pd.DataFrame,
    target_columns: list[str],
    correlation_threshold: float = 0.5,
    method: str = 'pearson'
) -> list[str]:
    """
    Finds auxiliary features that have correlation (by absolute value)
    above the given threshold with any of the target columns.
    """
    self.logger.info(f"Searching for correlated auxiliary features with threshold
{correlation_threshold}...")
    if not isinstance(df_corr_analysis, pd.DataFrame):
        self.logger.error("'df_corr_analysis' must be a pandas DataFrame.")
        raise ValueError("'df_corr_analysis' must be a pandas DataFrame.")
    if not isinstance(target_columns, list) or not all(isinstance(col, str) for col in
target_columns):
        self.logger.error("'target_columns' must be a list of strings.")
        raise ValueError("'target_columns' must be a list of strings.")
    if not 0 <= correlation_threshold <= 1:
        self.logger.error("'correlation_threshold' must be in [0, 1].")
        raise ValueError("'correlation_threshold' must be in [0, 1].")

    valid_target_columns = [col for col in target_columns if col in df_corr_analysis.columns]
    if not valid_target_columns:
        self.logger.warning("None of the specified target columns found in correlation
DataFrame.")
        return []

    aux_candidate_columns = [col for col in df_corr_analysis.columns if col not in
valid_target_columns]

    if not aux_candidate_columns:
        self.logger.warning("No auxiliary numeric columns for correlation analysis.")
        return []

```

```

try:
    corr_matrix = df_corr_analysis.corr(method=method)
except Exception as e:
    self.logger.error(f"Correlation matrix computation failed: {e}")
    return []

    correlations_with_targets = corr_matrix.loc[aux_candidate_columns,
valid_target_columns].abs()
    highly_correlated_mask = (correlations_with_targets > correlation_threshold).any(axis=1)
    result_features = correlations_with_targets[highly_correlated_mask].index.tolist()

    self.logger.info(f"Found {len(result_features)} highly correlated auxiliary features.")
    return sorted(result_features)

def process_spending_data(self, file_path: pathlib.Path,
                           correlation_threshold_for_aux_features: float = 0.1) -> tuple[
    pd.DataFrame, dict[str, list[str]], list[str]]:
    """
    Main method for processing spending data.
    Loads data, cleans, transforms, engineers features,
    and returns processed DataFrame, column classification and correlated features.
    """
    self.logger.info(f"Starting data processing for file: {file_path}")

    initial_df, all_categories = self._load_and_prepare_initial_df(file_path)
    initial_month_period_nunique = initial_df["month_period"].nunique()

    self.df = self._create_pivot_table(initial_df)

    classified_categories, regular_cols, periodical_cols, occasional_cols = \
        self._classify_spending_categories(self.df, all_categories, initial_month_period_nunique)

    self._engineer_temporal_features(self.df)
    self._engineer_aggregated_spend_features(self.df, periodical_cols, occasional_cols)

    all_expense_columns_for_corr = [col for col in (regular_cols + periodical_cols +
occasional_cols) if
                                   col in self.df.columns]

    if not all_expense_columns_for_corr:
        self.logger.warning("No expense columns found for correlation analysis. Returning
empty list.")
        correlated_aux_features = []
    else:
        df_for_correlation_analysis = self.df[all_expense_columns_for_corr].copy()
        valid_regular_cols_for_corr = [col for col in regular_cols if col in
df_for_correlation_analysis.columns]

        if not valid_regular_cols_for_corr:
            self.logger.warning("No valid regular target columns for correlation found. Returning
empty list.")
            correlated_aux_features = []

```

```

    else:
        correlated_aux_features = self._find_correlated_aux_features(
            df_corr_analysis=df_for_correlation_analysis,
            target_columns=valid_regular_cols_for_corr,
            correlation_threshold=correlation_threshold_for_aux_features,
            method='pearson'
        )

    cols_to_drop = [col for col in RAW_TEMPORAL_COLUMNS_TO_DROP if col in
self.df.columns]
    processed_df = self.df.drop(columns=cols_to_drop)

    self.logger.info("Data processing completed successfully.")
    return processed_df, classified_categories, correlated_aux_features

def main():
    print("Running test scenario for SpendingDataProcessor...")

    file_path_data_folder = pathlib.Path(__file__).parents[3].resolve() / 'data'
    file_path = file_path_data_folder / 'dataset.csv'

    processor = DataPreprocessor()

    try:
        final_df, column_categories, correlated_features = processor.process_spending_data(
            file_path)

        if not (file_path_data_folder / 'intermediate').exists():
            pathlib.Path(file_path_data_folder / 'intermediate').mkdir(parents=True)

        pd.to_pickle(final_df, file_path_data_folder / 'intermediate/data_prepare.pkl')

        with open(file_path_data_folder / 'intermediate/columns_cat.json', 'w') as f:
            json.dump(column_categories, f)

        with open(file_path_data_folder / 'intermediate/correlated_features.txt', 'w') as f:
            f.writelines(correlated_features)

        print(f"DataFrame shape: {final_df.shape}")

        print(f"\n--- Classified categories ---")
        for key, val in column_categories.items():
            print(f"{key}: {val}")

        print(f"\n--- Correlated auxiliary features (with regular expenses, threshold 0.1) ---")
        print(correlated_features)

        # Check for expected columns in the final DataFrame
        print(f"\n--- Columns in final DataFrame ---")
        print(final_df.columns.tolist())

    except Exception as e:

```

```

print(f"An error occurred during the test run: {e}")

if __name__ == '__main__':
    main()

```

Б.2 Текст файла views.py

```

import json
import os

import pandas as pd
from django.contrib.auth.mixins import LoginRequiredMixin
from django.contrib.sites import requests
from django.core.exceptions import ValidationError, PermissionDenied
from django.http import JsonResponse, HttpResponseRedirect, HttpResponseBadRequest
from django.shortcuts import render, redirect, get_object_or_404
from django.urls import reverse_lazy, reverse
from django.views import View
from django.views.generic import FormView, ListView, DeleteView
from django.views.generic import TemplateView

from .forms import UserFileUploadForm, PredictionForm
from .models import UserFileUpload, TransactionCategory, Transaction, UserPrediction,
PredictionDay, PredictionCategory, PredictionAlgorithm
from predictions.tasks import train_and_predict_task
from celery.result import AsyncResult
from django.http import JsonResponse
from personal_budget_planner.celery import app

def prediction_task_status(request, task_id):
    result = AsyncResult(task_id, app=app)
    if result.ready():
        prediction = result.result
        # Replace None or missing values with a default value
        cleaned_prediction = [
            {key: (value if value is not None else 0) for key, value in day.items()}
            for day in prediction
        ]

        request.session['prediction_data'] = cleaned_prediction
        return JsonResponse({'status': 'done', 'result': cleaned_prediction})
    return JsonResponse({'status': 'pending'})

class HomeView(LoginRequiredMixin, TemplateView):
    template_name = "home.html"
    login_url = "login"
    redirect_field_name = "next"

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

```

```

latest_prediction = UserPrediction.objects.filter(user=self.request.user).order_by('-
created_at').first()
context['readonly'] = True
if latest_prediction:
    days_data = []
    for day in latest_prediction.days.order_by('order'):
        day_dict = {cat.category.name: float(cat.amount) for cat in day.categories.all()}
        days_data.append(day_dict)
    context['prediction'] = json.dumps(days_data)
    context['prediction_length'] = len(days_data)

else:
    context['prediction'] = None
    context['prediction_length'] = 0
return context

```

```

class FileUploadView(LoginRequiredMixin, FormView):
    template_name = 'file_upload.html'
    form_class = UserFileUploadForm
    success_url = reverse_lazy('file-list')

    def form_valid(self, form):
        file_instance = form.save(commit=False)
        file_instance.user = self.request.user
        file_path = file_instance.file.path
        file_extension = file_path.split('.')[-1].lower()

        if file_extension not in ['csv', 'xls', 'xlsx']:
            form.add_error('file', 'Unsupported file type. Please upload a CSV or XLSX file.')
            return self.form_invalid(form)

        file_instance.save()
        status = self.process_uploaded_file(file_instance)
        if status == 0:
            form.add_error('file', 'There was an error processing the file.')
            return self.form_invalid(form)

        return super().form_valid(form)

    def process_uploaded_file(self, file_instance):
        file_path = file_instance.file.path
        file_extension = file_path.split('.')[-1].lower()

        if file_extension == 'csv':
            df = pd.read_csv(file_path)
        elif file_extension in ['xls', 'xlsx']:
            df = pd.read_excel(file_path)
        else:
            return 0

        default_category = TransactionCategory.objects.filter(name='Other').first()

```

```

if not default_category:
    raise ValidationError("'Other' category does not exist in the database.")
required_columns = ['category', 'date', 'amount']
missing_columns = [col for col in required_columns if col not in df.columns]

if missing_columns:
    raise ValidationError(f"Missing required columns: {'', '.join(missing_columns)}")
for index, row in df.iterrows():
    category_name = row.get('category', 'Other')
    category = TransactionCategory.objects.filter(name=category_name).first()
    if not category:
        category = default_category

    try:
        transaction_date = pd.to_datetime(row['date'], errors='raise').date()
    except Exception:
        raise ValidationError(f"Invalid date format at row {index + 1}.")
    try:
        amount = float(row['amount'])
    except ValueError:
        raise ValidationError(f"Invalid amount format at row {index + 1}.")

    transaction = Transaction(
        user=file_instance.user,
        date=transaction_date,
        category=category,
        amount=amount
    )
    transaction.save()

return 1

```

```

class FileListView(LoginRequiredMixin, ListView):
    model = UserFileUpload
    template_name = 'file_list.html'
    context_object_name = 'files'

    def get_queryset(self):
        return UserFileUpload.objects.filter(user=self.request.user).order_by('-uploaded_at')

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        for file in context['files']:
            file.filename = os.path.basename(file.file.name)
        return context

```

```

class PredictionFormView(LoginRequiredMixin, FormView):
    template_name = 'prediction_form.html'
    form_class = PredictionForm

    def get(self, request, *args, **kwargs):

```

```

    form = self.form_class()
    saved_predictions = UserPrediction.objects.filter(user=request.user) \
        .prefetch_related('days') \
        .order_by('-created_at')
    algorithms = PredictionAlgorithm.objects.all()
    has_transactions = Transaction.objects.filter(user=request.user).exists()
    return self.render_to_response({
        'form': form,
        'saved_predictions': saved_predictions,
        'algorithms': algorithms,
        'has_transactions': has_transactions,
    })

def form_valid(self, form):
    categories = form.cleaned_data['category']
    date_from = form.cleaned_data.get('date_from')
    date_to = form.cleaned_data.get('date_to')

    transactions = Transaction.objects.filter(category__in=categories, user=self.request.user)

    if date_from:
        transactions = transactions.filter(date__gte=date_from)
    if date_to:
        transactions = transactions.filter(date__lte=date_to)

    transaction_ids = list(transactions.values_list('id', flat=True))
    task = train_and_predict_task.delay(self.request.user.id, transaction_ids, 7)

    # Store task ID in session for later checking in JS
    self.request.session['task_id'] = task.id

    return redirect('prediction_form')

def form_invalid(self, form):
    saved_predictions =
UserPrediction.objects.filter(user=self.request.user).prefetch_related('days').order_by('-
created_at')
    return self.render_to_response({
        'form': form,
        'saved_predictions': saved_predictions
    })

class PredictionResultView(LoginRequiredMixin, TemplateView):
    template_name = 'prediction_result.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        prediction_id = self.kwargs.get('pk')
        context['readonly'] = bool(self.kwargs.get('pk'))

```

```

if prediction_id:
    prediction = get_object_or_404(UserPrediction, pk=prediction_id, user=self.request.user)
    days_data = []
    for day in prediction.days.order_by('order'):
        day_dict = {cat.category.name: float(cat.amount) for cat in day.categories.all()}
        days_data.append(day_dict)
    else:
        days_data = self.request.session.get('prediction_data', [])

    context['prediction'] = json.dumps(days_data)
    context['prediction_length'] = len(days_data)

    return context

class SavePredictionView(LoginRequiredMixin, View):
    def post(self, request):
        data_raw = request.POST.get('data', "")
        if not data_raw:
            return HttpResponseBadRequest("No prediction data provided.")

        try:
            data = json.loads(data_raw)
        except json.JSONDecodeError:
            return HttpResponseBadRequest("Invalid JSON in prediction data.")

        name = request.POST.get('name', "").strip()

        user_prediction = UserPrediction.objects.create(user=request.user, name=name)
        days = [PredictionDay(prediction=user_prediction, order=i + 1) for i in range(len(data))]
        PredictionDay.objects.bulk_create(days)

        days = list(user_prediction.days.all())
        categories = []
        for i, day_data in enumerate(data):
            for category_name, amount in day_data.items():
                category = TransactionCategory.objects.filter(name=category_name).first()
                if category:
                    categories.append(PredictionCategory(
                        prediction_day=days[i],
                        category=category,
                        amount=amount
                    ))
        PredictionCategory.objects.bulk_create(categories)

        # Delete prediction data from session after saving
        request.session.pop('prediction_data', None)

        return redirect('home')

class DeletePredictionView(LoginRequiredMixin, View):
    def post(self, request, *args, **kwargs):

```

```

        prediction = get_object_or_404(UserPrediction, pk=kwargs['pk'],
user_id=self.request.user.id)

        if prediction.user != request.user:
            raise PermissionDenied("You do not have permission to delete this prediction.")

        prediction.delete()
        return redirect('prediction_form')

class TransactionCreateView(LoginRequiredMixin, CreateView):
    model = Transaction
    form_class = TransactionForm
    success_url = reverse_lazy('home')

    def form_valid(self, form):
        form.instance.user = self.request.user
        form.instance.data_source = 2
        return super().form_valid(form)

class TransactionDeleteView(LoginRequiredMixin, View):
    def post(self, request, *args, **kwargs):
        pk = kwargs.get('pk')
        transaction = Transaction.objects.filter(user=request.user, data_source=2, pk=pk).first()
        if transaction:
            transaction.delete()
        return HttpResponseRedirect(reverse_lazy('home'))
        return redirect('prediction_form')

```

Б.3 Текст файла data_splitter.py

```

import json
import pathlib

import pandas as pd
import logging

from predictions.services.main_model.utils import get_logger

from personal_budget_planner.constants import *

class DataSplitter:
    """
    Class for splitting a DataFrame into training and test sets (X, y).
    """

    def __init__(self, logger: logging.Logger | None = None):
        """
        Initialize DataSplitter.
        """

```

```

self.logger = logger if logger else get_logger(self.__class__.__name__,
                                              log_file_name="data_splitter.log")

def split_data(self,
               full_df: pd.DataFrame,
               selected_feature_names: list[str],
               target_names: list[str],
               test_set_size_days: int = TEST_SIZE_DAYS,
               use_log_transformed_targets: bool = False,
               ) -> tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    """
    Splits the data into training and test sets X and y.
    """
    self.logger.info("Starting data split into training and test sets...")
    df = full_df.copy()

    # Determine actual target columns to use
    actual_targets_to_use = [f'{col}_log' for col in target_names] if
    use_log_transformed_targets else target_names
    actual_targets_to_use = [t for t in actual_targets_to_use if t in df.columns]

    if not actual_targets_to_use:
        self.logger.error("No target columns found in DataFrame.")
        empty_df = pd.DataFrame()
        return empty_df, empty_df, empty_df, empty_df

    # Validate selected features
    valid_selected_features = [f for f in selected_feature_names if f in df.columns]
    if not valid_selected_features:
        self.logger.error("Selected features list is empty or not found in DataFrame.")
        empty_df = pd.DataFrame()
        return (
            empty_df, empty_df,
            df[actual_targets_to_use] if actual_targets_to_use else empty_df,
            df[actual_targets_to_use] if actual_targets_to_use else empty_df
        )

    # Check if index and data are valid
    if df.index.max() is None or len(df) == 0:
        self.logger.error("DataFrame is empty or has no valid datetime index.")
        empty_df = pd.DataFrame()
        return empty_df, empty_df, empty_df, empty_df

    # Perform the split
    split_date = df.index.max() - pd.Timedelta(days=test_set_size_days)
    df_train = df[df.index <= split_date].copy()
    df_test = df[df.index > split_date].copy()

    self.logger.info(
        f'Data split at: {split_date}. Training samples: {len(df_train)}, test samples: {len(df_test)}.'
    )

```

```

X_train = df_train[valid_selected_features]
y_train = df_train[actual_targets_to_use]
X_test = df_test[valid_selected_features]
y_test = df_test[actual_targets_to_use]

# Fill NaNs in targets with 0
y_train = y_train.fillna(0)
y_test = y_test.fillna(0)

self.logger.info(
    f"Shapes: X_train={X_train.shape}, y_train={y_train.shape}, X_test={X_test.shape},
    y_test={y_test.shape}"
)
self.logger.info("Data split completed.")
return X_train, X_test, y_train, y_test

def main():
    file_path_data_folder = pathlib.Path(__file__).parents[3].resolve() / 'data'

    df = pd.read_pickle(file_path_data_folder / 'intermediate/engineered_df.pkl')

    with open(file_path_data_folder / 'intermediate/selected_features.json', 'r') as f:
        selected_features = json.load(f)['selected_features']

    with open(file_path_data_folder / 'intermediate/features_dict.json', 'r') as f:
        target_cols = json.load(f)['target']

    X_train, X_test, y_train, y_test = DataSplitter().split_data(df,
                                                                selected_features,
                                                                target_names=target_cols)

    X_train.to_pickle(file_path_data_folder / 'intermediate/X_train.pkl')
    y_train.to_pickle(file_path_data_folder / 'intermediate/y_train.pkl')
    X_test.to_pickle(file_path_data_folder / 'intermediate/X_test.pkl')
    y_test.to_pickle(file_path_data_folder / 'intermediate/y_test.pkl')

if __name__ == '__main__':
    main()

```

ДОДАТОК В
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ Кафедра програмних засобів НУ "Запорізька політехніка"		
	ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ФІНАНСОВОГО ПЛАНУВАННЯ НА ОСНОВІ ПОВЕДІНКОВИХ ПАТЕРНІВ КОРИСТУВАЧА	
	Автор - студент групи КНТ-111 Сергій СКОРИК Керівник - к.т.н., доцент Валерій ЛЬОВКІН	
	2025 р.	

Рисунок В.1 – Слайд 1

	Об'єкт, предмет та мета роботи
	Об'єкт дослідження – процес фінансового планування. Предмет дослідження – методи фінансового планування на основі поведінкових патернів користувача. Мета роботи – розробка програмного забезпечення фінансового планування на основі поведінкових патернів користувача для надання можливостей прогнозування витрат користувача з використанням машинного навчання.

Рисунок В.2 – Слайд 2

Порівняння існуючих аналогів

Критерій	YNAB	PocketGuard	Quicken Simplifi	Розроблюваний застосунок
Компанія	You Need A Budget, LLC	PocketGuard, Inc.	Quicken Inc.	Власна розробка в межах кваліфікаційної роботи
Платформи	Windows, macOS, iOS, Android, Веб-додаток	iOS, Android	iOS, Android, Веб-додаток	Веб-додаток
Основні функції	Методика нульового бюджету, фінансові звички	Автоматична синхронізація з банками	Спрощене управління бюджетом	Завантаження транзакцій, фільтрація, збереження прогнозів
Прогнозування витрат	Прогноз на базі попередніх даних	Орієнтовано на поточний залишок	Оцінка залишку коштів	Прогнозування бюджету на основі історичних даних
Інтерфейс	Гнучкий	Лаконічний, спрощений	Мінімалістичний	Зручний із вибором категорій та графіками

Рисунок В.3 – Слайд 3

Аналіз набору даних

- У середньому 3–4 активні категорії щодня
- Багато днів без витрат у більшості категорій
- Витрати мають циклічність (будні/вихідні)
- виправлено орфографічні помилки в категоріях
- Розподіл із домінуванням малих сум і поодинокими піками
- Потрібен адаптивний підхід із урахуванням часу, суми, частоти

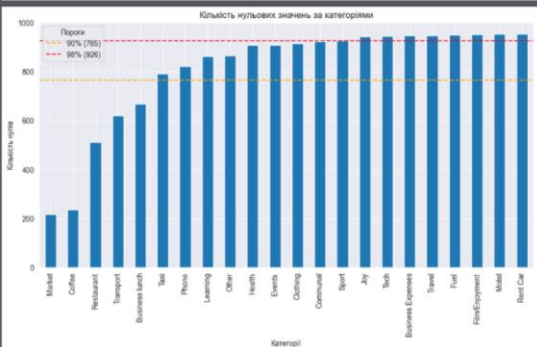
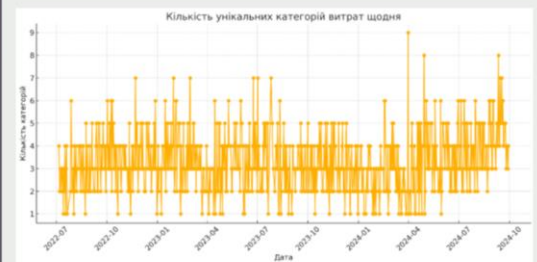


Рисунок В.4 – Слайд 4

Порівняння фреймворків для бекенду

Критерій	Django	FastAPI	Flask
Призначення	Повноцінний фреймворк «все-в-одному»	Сучасний асинхронний API-фреймворк	Мікрофреймворк для простих застосунків
Шаблони для рендерингу	Вбудовані шаблони	Потрібно інтегрувати сторонні рішення	Вбудовані шаблони
Підтримка REST API	Через Django REST Framework	Вбудована нативна підтримка	Потрібно реалізовувати вручну або через Flask-RESTful
Асинхронність	Часткова підтримка	Повноцінна асинхронна підтримка	Лімітована, через сторонні бібліотеки
ORM	Вбудована	Сторонні бібліотеки	Сторонні бібліотеки
Безпека	Вбудовані механізми	Сторонні бібліотеки	Потрібно реалізовувати самостійно
Легкість використання	Багато вбудованих інструментів	Вимагає додаткових бібліотек та більше ручної роботи	Вимагає додаткових бібліотек та більше ручної роботи

Рисунок В.5 – Слайд 5

Діаграма прецедентів застосунку

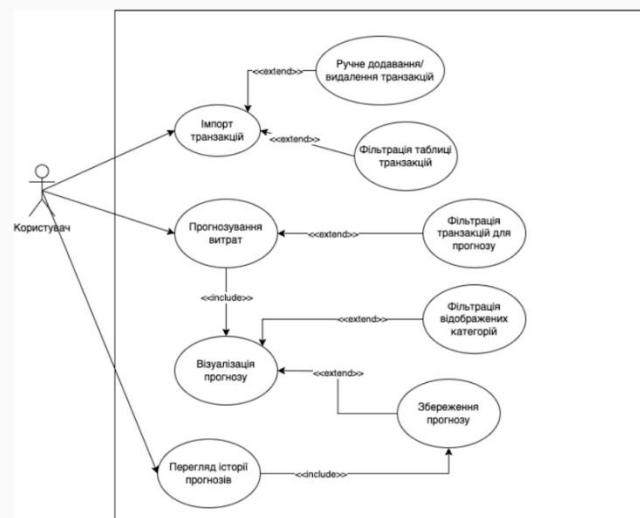


Рисунок В.6 – Слайд 6

Структурна схема бази даних

Рисунок В.7 – Слайд 7

Архітектурна схема застосуку

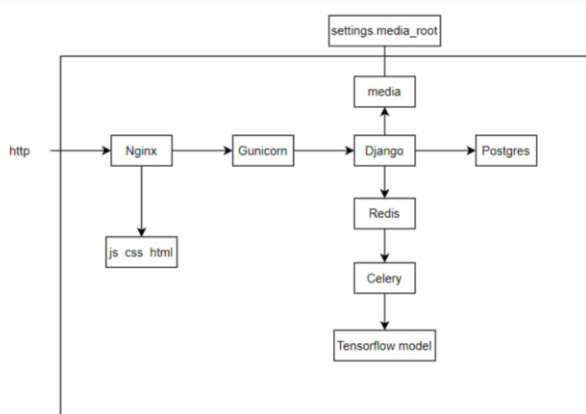


Рисунок В.8 – Слайд 8

Схема побудови моделі прогнозування



Рисунок В.9 – Слайд 9



Результати роботи програми

Рисунок В.10 – Слайд 10

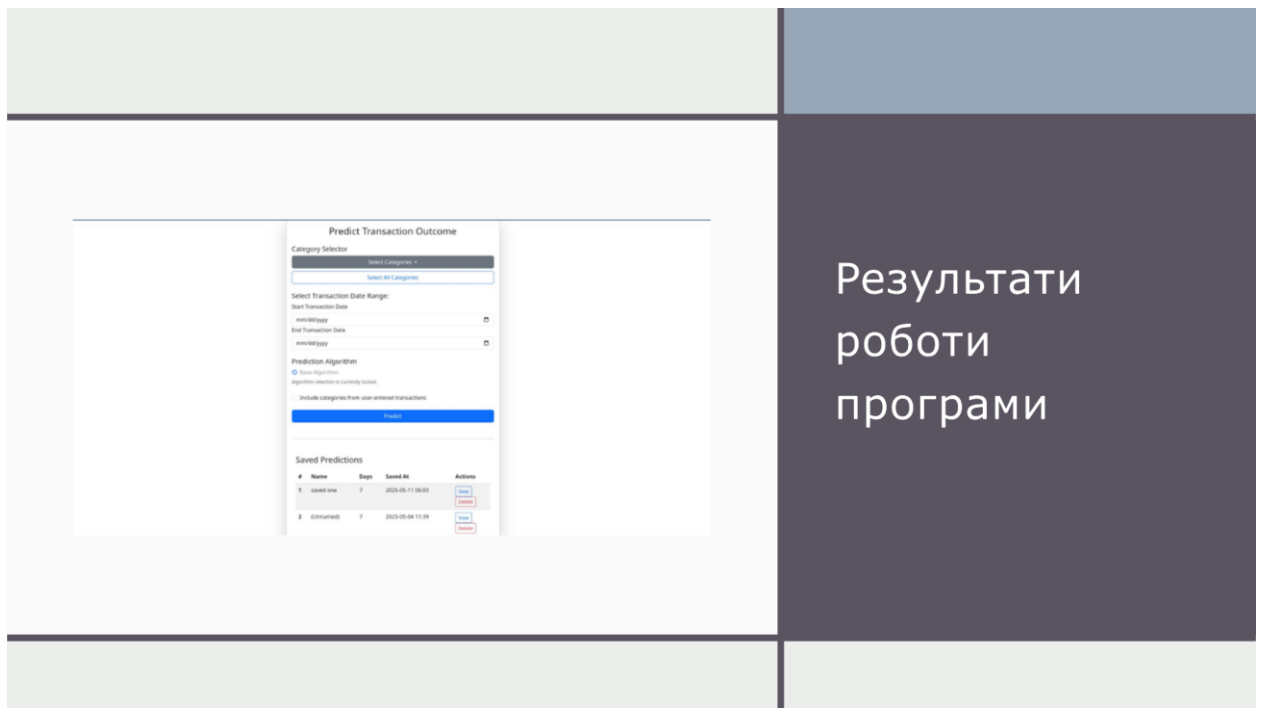


Рисунок В.11 – Слайд 11

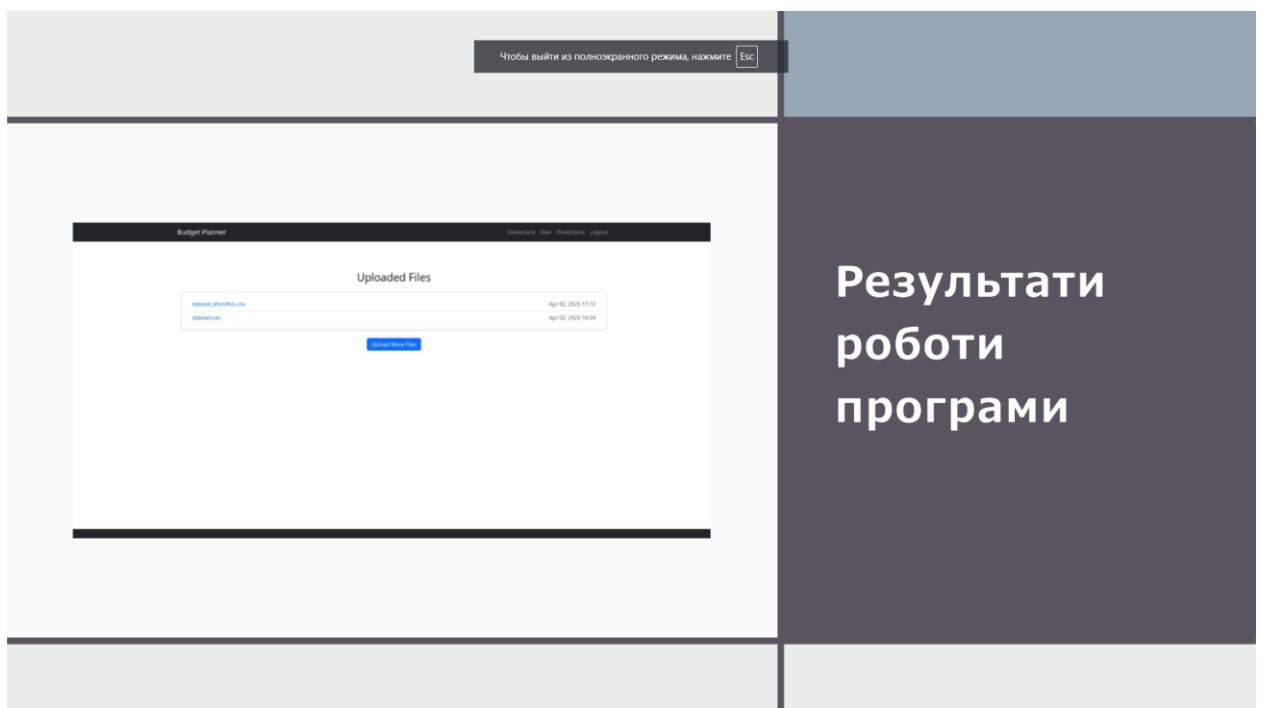


Рисунок В.12 – Слайд 12

Порівняння моделей прогнозування

Kateropія	MAE	RMSE	MASE
Market	5.662	6.473	0.641
Coffee	8.051	8.068	0.850
Restaurant	15.845	25.996	0.878
Transport	1.078	1.230	0.683
Business lunch	0.377	0.377	0.118
Average	6.202	8.429	0.634

Гібридна модель LSTM + TCN

Kateropія	MAE	RMSE	MASE
Market	6.784	7.524	0.813
Coffee	8.691	9.204	0.950
Restaurant	18.316	28.725	1.049
Transport	1.302	1.482	0.854
Business lunch	0.441	0.453	0.167
Average	7.107	9.278	0.767

LSTM модель

Рисунок В.13 – Слайд 13

Висновки

У ході виконання дипломної кваліфікаційної роботи бакалавра було розроблено програмне забезпечення для прогнозування особистих витрат. Для досягнення поставленої мети було виконано ряд завдань, зокрема аналіз предметної області, експериментальне дослідження ефективності моделей машинного навчання для прогнозування витрат у різних категоріях, проектування та розробка програмного забезпечення, реалізація функціоналу завантаження, обробки та візуалізації транзакцій, проведення комплексного тестування програмного забезпечення.

Розроблений застосунок повністю відповідає вимогам технічного завдання, забезпечує точне прогнозування витрат та може використовуватись як основа для персональних фінансових асистентів або аналітичних платформ у сфері фінансового планування.

Наукова новизна роботи полягає в запропонованому методі прогнозування витрат на основі поведінкових патернів користувачів, який базується на багатоступеневій обробці та відборі ознак із подальшим застосуванням гібридних нейронних архітектур, що дозволяє виконувати розподілення витрат за категоріями, забезпечуючи гнучкість, точність і адаптивність прогнозування.

Рисунок В.14 – Слайд 14

Публікації

Скорик С. С. Програмне забезпечення стискання фотографій на основі засобів кластеризації [Текст] / С. С. Скорик, В. М. Льовкін // Free and open source software : XV Міжнародна науково-практична конференція. 13-14 лютого 2024 р., Харків, Україна. - Харків - Харківський національний університет імені Семена Кузнеця, 2024 р. - С. 96 - 97.

Рисунок В.15 – Слайд 15