

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА**

**Навчально-науковий інститут фізико-технічних та комп'ютерних наук  
кафедра комп'ютерних наук**

**Застосунок для планування бюджету з прогнозуванням витрат**

**Курсова робота**

**Рівень вищої освіти – другий (магістерський)**

**Виконав:**  
студент 5 курсу, 544 групи  
**Іпатій Ю.І.**  
**Керівник:**

\_\_\_\_\_,  
\_\_\_\_\_

**Чернівці – 2025**

**Чернівецький національний університет імені Юрія Федьковича**  
Навчально-науковий інститут фізико-технічних та комп'ютерних наук  
 Кафедра Комп'ютерних наук  
 Спеціальність Комп'ютерні науки  
 Освітній ступінь Магістр  
 Форма навчання денна курс 5 група 544м

## ЗАВДАННЯ НА КУРСОВУ РОБОТУ СТУДЕНТА

Іпатія Юрія Ілліча

(прізвище, ім'я, по батькові)

### 1. Тема роботи

Застосунок для планування бюджету з прогнозуванням витрат

затверджена протоколом засідання кафедри від «\_\_» \_\_\_\_\_ року №\_\_

2. Термін подання студентом закінченої роботи 26.05.2025

3. Вхідні дані до роботи

Технології розробки: MVS, .NET, SQLite, LocalStorage.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які треба розробити)

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

РОЗДІЛ 2. ВИБІР ТЕХНОЛОГІЙ

РОЗДІЛ 3. ОПИС ЗАСТОСУНКУ

ВИСНОВОК

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

5. Перелік графічного, наочного матеріалу

- Загальна схема системи застосунків
- Діаграма прецедентів менеджера
- Діаграма прецедентів працівника
- Схема бази даних
- Діаграми класів розроблених програм
- Скріншоти програм

6. Консультант(и) курсової роботи Ушенко Ю.О.

№	Назва етапів курсової роботи	Термін виконання етапів роботи	Примітки
1	Вибір, погодження й затвердження теми, призначення наукового керівника, рецензента, консультанта	15.01.2025	виконано
2	Складання календарного плану й розширеного плану-конспекту роботи	21.01.2025	виконано
3	Аналіз предметної області, дослідження літератури та матеріалів	24.01.2025	виконано
4	Аналіз існуючих аналогів програмного забезпечення	05.02.2025	виконано
5	Постановка задачі за темою курсової роботи	07.02.2025	виконано
6	Розділ 1. Аналіз предметної області	13.02.2025	виконано
7	Розділ 2. Вибір технологій	17.03.2025	виконано
9	Розділ 3. Опис застозунку	17.04.2025	виконано
11	Тестування застосунку	06.05.2025	виконано
12	Оформлення пояснювальної записки	12.05.2025	виконано
13	Перевірка відповідності оформлення	16.05.2025	виконано
14	Оформлення презентації та доповіді	26.05.2025	виконано
15	Захист курсової роботи		

Студент

Іпатій Ю.І,

\_\_\_\_\_  
(підпис)

Науковий керівник

Ушенко Ю.О.

\_\_\_\_\_  
(підпис)

«\_\_» \_\_\_\_ 2025 р.

## АНОТАЦІЯ

У курсовій роботі розглянуто процес розробки кросплатформного застосунку для планування персонального бюджету з функціональністю передбачення витрат. Головна мета проекту – створити зручний і ефективний інструмент для обліку прибутків та видатків, аналізу фінансової поведінки користувача й передбачення майбутніх витрат на основі історичних даних.

У роботі докладно описано етапи створення застосунку, включаючи аналіз предметної галузі, вибір технологій (JavaScript, HTML5, CSS3, Chart.js, LocalStorage, SQLite), архітектурне проектування та реалізацію основних модулів: обліку транзакцій, побудови графіків та аналітики, а також модуль передбачення витрат. Проведено тестування застосунку та оцінено його продуктивність і зручність використання.

Результатом є повноцінний вебзастосунок, що дає змогу ефективно управляти особистими фінансами. Одержані результати демонструють високу практичну цінність системи та перспективу її подальшого розвитку.

Робота містить 37 сторінок, 12 рисунків та 20 посилань на літературні джерела.

**Ключові слова:** бюджетування, прогнозування витрат, вебзастосунок, JavaScript, Chart.js, LocalStorage, SQLite.

Робота містить результати власних досліджень. Використання чужих ідей, результатів і текстів мають посилання на відповідне джерело.

---

(підпис)

## ANNOTATION

This course paper presents the development of a cross-platform application for personal budget planning with expense forecasting functionality. The main goal of the project is to create a convenient and effective tool for tracking income and expenses, analyzing user financial behavior, and forecasting future spending based on historical data.

The paper describes in detail all stages of development, including domain analysis, technology selection (JavaScript, HTML5, CSS3, Chart.js, LocalStorage, SQLite), architectural design, and implementation of key modules: transaction tracking, data visualization and analytics, and an expense forecasting module. The application was tested to assess its performance and usability.

The result is a fully functional web application that enables efficient personal finance management. The outcomes confirm the system's high practical value and its potential for future enhancement.

The work contains 37 pages, 12 figures and 20 references to literary sources.

**Keywords:** budgeting, expense forecasting, web application, JavaScript, Chart.js, LocalStorage, SQLite.

The work includes original research. All external ideas, results, and texts are properly cited. The work contains the results of my own research. The use of other people's ideas, results and texts has a link to the appropriate source.

## ЗМІСТ

### ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ТА ТЕРМІНІВ

ВСТУП .....	8
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ .....	10
1.1 Аналіз сучасних фінансових застосунків .....	10
1.2. Функціональні вимоги до системи.....	11
1.3. Поняття та підходи до прогнозування витрат.....	14
2. ВИБІР ТЕХНОЛОГІЙ .....	18
2.1 Обґрунтування вибору платформи.....	18
2.2 Обґрунтування вибору LocalStorage/SQLite для зберігання даних .....	19
2.3 Обґрунтування використання Chart.js для візуалізації .....	21
3. ОПИС ЗАСТОСУНКУ .....	26
3.1 Загальна архітектура .....	26
3.2 Основні функції програми.....	29
3.3 Тестування і результати .....	31
ВИСНОВКИ.....	35
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	37

## **ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ТА ТЕРМІНІВ**

ПЗ – Програмне забезпечення.

ОС – Операційна система.

СУБД – Система управління базами даних.

ПП – Програмний продукт.

ІТ – Інформаційні технології.

ПС – Програмна система.

ІК – Інтерфейс користувача.

EF – Entity Framework.

ООП – Об’єктно-орієнтоване програмування.

БД – База даних.

MVS – Microsoft Visual Studio.

XML – Extensible Markup Language. MVS – Microsoft Visual Studio.

XML – Extensible Markup Language.

.NET – Платформа розробки програмного забезпечення від компанії Microsoft.

.NET MAUI – .NET Multi-platform App UI — кросплатформна технологія для створення інтерфейсів користувача.

C# – Мова програмування C-sharp.

SQLite – Вбудована реляційна СУБД.

AI – Штучний інтелект (Artificial Intelligence).

ML – Машинне навчання (Machine Learning).

JSON – JavaScript Object Notation — формат обміну даними.

## ВСТУП

В умовах зростаючої економічної нестабільності та постійних змін цінової політики на товари й послуги, ефективне управління особистими фінансами набуває особливого значення. Багато користувачів стикаються з проблемою відсутності чіткої структури контролю над витратами, що призводить до нерационального використання коштів. Саме тому актуальною є розробка сучасного, зручного та функціонального програмного забезпечення для планування бюджету з можливістю прогнозування витрат на основі попередньої фінансової активності.

Кросплатформність і доступність. Завдяки широкому розповсюдженню мобільних пристроїв та різноманітності операційних систем, користувачі очікують однаково зручного доступу до своїх фінансових даних із будь-якого пристрою — Android, iOS чи Windows. Технологія .NET MAUI забезпечує створення єдиного коду для кількох платформ, що дозволяє значно скоротити витрати на розробку та обслуговування програмного продукту.

Автоматизація аналізу й прогнозування. Усе більше користувачів прагнуть отримувати не лише історію своїх витрат, а й прогнозування майбутніх фінансових тенденцій. Завдяки використанню аналітичних алгоритмів та обробки історичних даних, застосунок дозволяє формувати прогнози майбутніх витрат, що сприяє кращому плануванню бюджету та запобіганню перевитратам.

Гнучка робота з базою даних. Використання Entity Framework у поєднанні з SQLite забезпечує зручне зберігання, оновлення та аналіз фінансових даних користувача. SQLite, як вбудована СУБД, є оптимальним рішенням для мобільних застосунків завдяки своїй легкості та швидкодії.

Сучасні технології розробки. Застосування таких інструментів, як .NET MAUI, C# та EF, відповідає найновішим стандартам програмної інженерії, дозволяючи створювати надійне, масштабоване та безпечне програмне забезпечення. Таке рішення є перспективним з точки зору подальшої модернізації та розширення функціоналу.



Покращення фінансової грамотності користувача. Завдяки візуалізації даних, прогнозам та аналізу витрат, застосунок сприяє формуванню усвідомленого ставлення до управління особистими коштами, що є надзвичайно важливим у сучасному суспільстві.

Отже, розробка застосунку для планування бюджету з функцією прогнозування витрат є актуальним напрямом, що відповідає потребам сучасного користувача у фінансовій самодостатності, зручності використання та доступності на різних платформах. Такий програмний продукт має потенціал для подальшого розвитку та комерційного впровадження.

4. Відповідність сучасним вимогам: використання таких технологій, як MAUI та Entity Framework, є надзвичайно актуальним для розробки сучасних бізнес-рішень. Вони відповідають сучасним вимогам до масштабованості, надійності та безпеки програмного забезпечення, що робить розроблені застосунки здатними до розвитку та адаптації до майбутніх змін.

5. Забезпечення прозорості та контролю: керівники команд та менеджери можуть використовувати ці інструменти для отримання точних даних про хід виконання завдань, вчасне виявлення проблем та прийняття оперативних рішень, що підвищує загальну ефективність роботи команди.

Отже, створення двох взаємопов'язаних програм для управління завданнями та проектами є важливим кроком у розвитку програмного забезпечення, яке може суттєво покращити організацію роботи в команді та оптимізувати розподіл завдань і ресурсів, що є критично важливим у сучасному бізнес-середовищі.

## АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

### 1.1 Аналіз сучасних фінансових застосунків

Утеперішньомуцифровому світікеруванняособистими фінансами стає все більшнагальнимдля широкого кола користувачів. Швидкий темп життя, нестабільність економіки, зростаюча інфляція та збільшення кількості щомісячних витратспонукаютьлюдей більш уважно ставитися до планування бюджету. Саме тому зростає попит на фінансовідодатки,які дозволяють не лише фіксувати доходи тавидатки,а й аналізувати фінансову поведінку користувача та прогнозувати майбутні витрати.

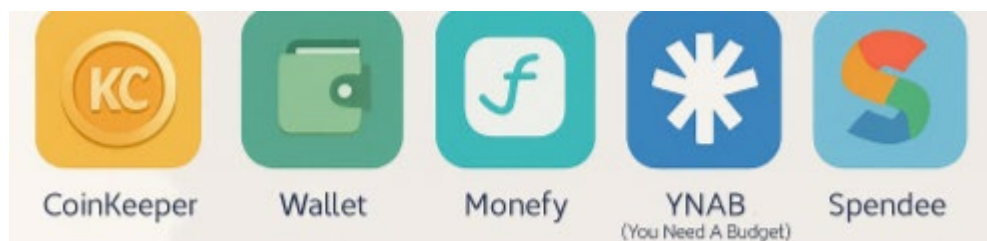


Рис. 1.1 Популярні фінансові додатки

На ринку представлено багато фінансовихдодатків,кожен із яких має свою специфіку, набір функцій та цільову аудиторію. Серед найбільш популярних рішень можна виділити такідодатки,як CoinKeeper, Wallet, Monefy, YNAB (You Need A Budget) та Spendee. Усі ці сервіси мають інтуїтивно зрозумілий інтерфейс, дозволяють зручно фіксувативидатки,групувати їх за категоріями, переглядати статистику та навіть встановлювати ліміти.

Проте при глибшому аналізі виявляються й обмеження таких систем. По-перше, багатододатківорієнтовані на англomовну аудиторію, що може створювати незручності для українських користувачів. По-друге, деякі з них потребують платної підписки для відкриття всіх функцій, що робить їх менш доступними для широкого загалу. По-третє, не всі сервіси реалізують функцію інтелектуального прогнозуваннявидатків,яка б дозволяла користувачеві побачити майбутню фінансову картину на основі попередньої активності. Зазвичай користувач отримує лише історичні графіки й базову аналітику, без можливості планування на основі прогнозу.

Особливу увагу варто звернути надодаткиз розширеним функціоналом. Наприклад, YNAB пропонує складну систему бюджетування з концепцією розподілу кожної гривні на конкретну мету. Проте ця система може бути надто складною для пересічного користувача, який бажає швидко і зручно переглядати свої доходи та видатки без необхідності проходити навчальні курси чи читати довгі інструкції.

З іншого боку, додатки на зразок Monefy і Wallet пропонують простоту, але мають обмежену аналітику і практично не включають прогнозування. Це створює нішу для нових рішень, які могли б поєднувати простоту інтерфейсу з розумними алгоритмами аналізу та прогнозу.

На основі аналізу вищезгаданих додатків можна зробити висновок, що на ринку існує потреба у програмному забезпеченні, яке:

- поєднує зручний і зрозумілий інтерфейс із можливістю гнучкого налаштування;
- підтримує багатоплатформність (мобільні та десктопні версії);
- дозволяє здійснювати аналіз фінансової поведінки користувача;
- реалізує алгоритми прогнозування видатків на основі історичних даних;
- працює автономно (без постійного підключення до інтернету) з використанням локальної бази даних.

У межах даної курсової роботи розробляється саме такий додаток, який урахує недоліки існуючих рішень та намагається поєднати їхні найкращі сторони. Це дозволить забезпечити користувачеві ефективне управління особистим бюджетом, а також допоможе приймати обґрунтовані фінансові рішення завдяки функції прогнозування.

## **1.2 Функціональні вимоги до системи**

Одним із найпростіших і водночас корисних застосунків для ведення особистого бюджету є Wallet Lite. Цей застосунок відзначається мінімалістичним дизайном і простим інтерфейсом, що дозволяє навіть недосвідченим користувачам швидко розібратися в основному функціоналі.

Wallet Lite дає змогу вручну вносити доходи та витрати, сортувати їх за категоріями (наприклад, їжа, транспорт, розваги) та переглядати загальний залишок бюджету. Хоча додаток не підтримує складну аналітику чи прогнозування, він чудово виконує роль простого фінансового щоденника. Такий підхід ідеальний для людей, які прагнуть контролювати витрати без потреби в складних графіках або підключенні до банківських рахунків.

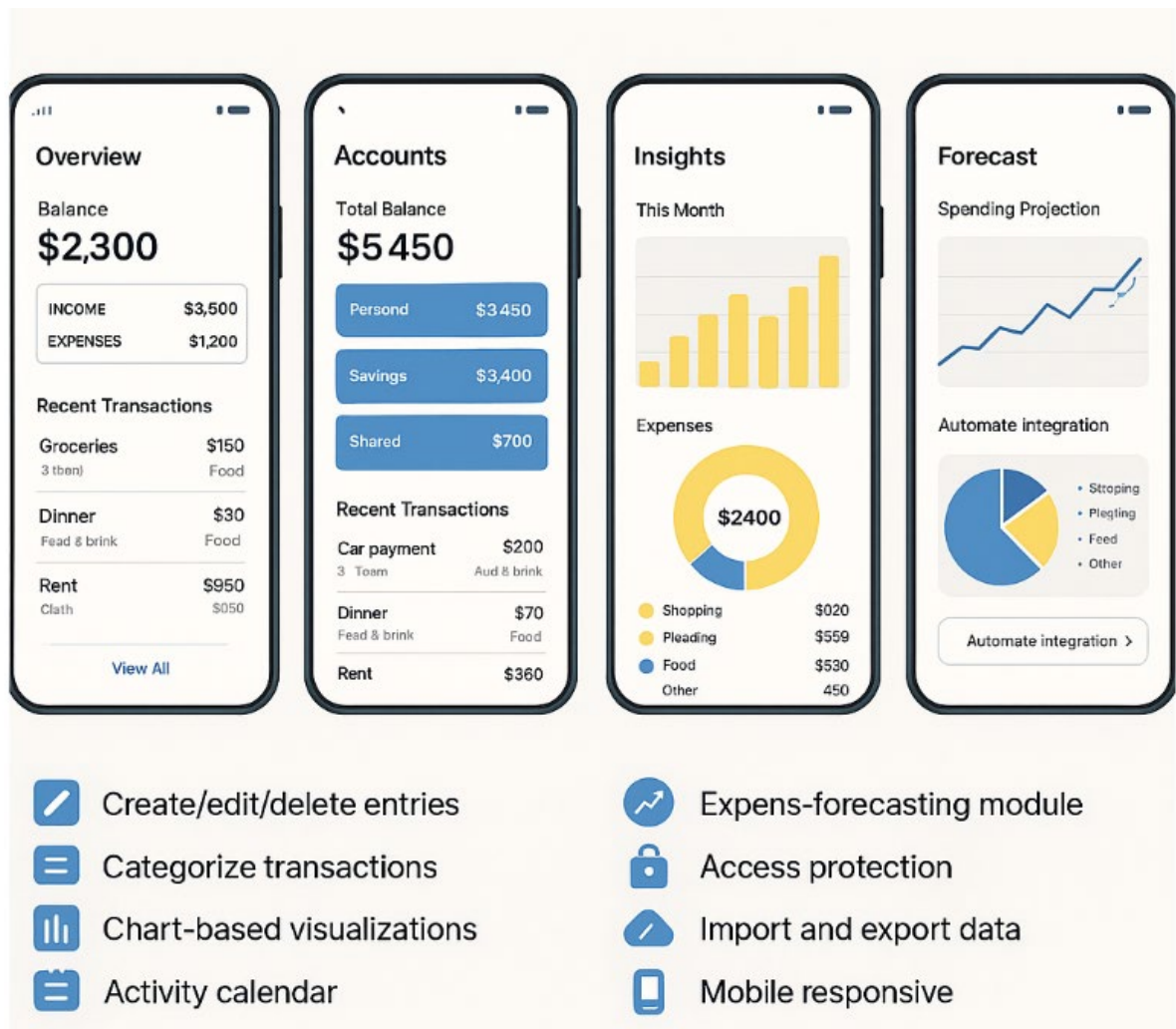


Рис. 1.2 Приклад фінансових застосунків

Для користувачів, які очікують більш широкого функціоналу, хорошим варіантом стане Money Manager. Цей застосунок забезпечує потужніші інструменти для управління особистими фінансами: автоматичне групування витрат, створення бюджетів на день, тиждень або місяць, а також можливість експорту звітів у форматах CSV або PDF. Крім того, програма дозволяє створювати кілька гаманців (напр., для готівки, банківської картки тощо) та перемикатися між ними. Однією з ключових функцій Money Manager є

календар витрат, який дозволяє відстежувати фінансову активність по днях. Також доступна функція блокування паролем, що додає конфіденційності.

Особливої уваги заслуговує програма Spendee, яка орієнтована на користувачів, що цінують автоматизацію. Цей застосунок дозволяє підключати банківські рахунки, автоматично завантажувати транзакції та аналізувати їх. Spendee також має вбудований інструмент прогнозування витрат, який, базуючись на попередніх даних, оцінює, скільки користувач витратить до кінця місяця. Завдяки цьому можна заздалегідь скоригувати бюджет, якщо витрати виходять за межі допустимого. Програма має кольорову візуалізацію та інтуїтивну аналітику у вигляді кругових та стовпчикових діаграм.

Ще одним цікавим рішенням є YNAB (You Need A Budget) — застосунок, розроблений за принципами фінансової дисципліни. Його основна ідея полягає в тому, щоб кожна гривня, яку користувач заробив, мала свою мету. Система дозволяє планувати витрати наперед, фіксувати реальні транзакції та коригувати плани в режимі реального часу. Унікальною функцією YNAB є навчальний модуль, який вчить користувача принципам фінансового планування.

Усі розглянуті застосунки мають свої переваги та недоліки, однак їх функціональні можливості формують базу для розуміння основних вимог до сучасного ПЗ для планування бюджету. На основі цього досвіду, застосунок, що розробляється в межах курсової роботи, повинен відповідати таким функціональним вимогам:

- створення, редагування та видалення фінансових записів;
- розподіл транзакцій за категоріями (доходи/витрати, типи витрат);
- відображення статистики у вигляді діаграм;
- календар фінансової активності;
- модуль прогнозування майбутніх витрат;
- захист доступу до програми (пароль або PIN);
- імпорт та експорт фінансових даних;

-адаптивний дизайн для роботи на різних пристроях.

Таким чином, функціональні вимоги до застосунку поєднують у собі як базові можливості для повсякденного використання, так і аналітичні інструменти, що дозволяють ефективно планувати бюджет і уникати фінансових ризиків у майбутньому.

### **1.3 Поняття та підходи до прогнозування витрат**

Прогнозування витрат є важливою складовою процесу планування особистих фінансів, управління підприємствами, а також у державному чи муніципальному бюджетуванні. У загальному розумінні, прогнозування витрат — це процес оцінювання можливих майбутніх видатків на підставі певної інформації, як-от історичні фінансові дані, поточні тренди, сезонні коливання, статистичні закономірності, а також вплив зовнішніх чинників, таких як інфляція, зміна цін на товари та послуги, зміна податкового навантаження чи тарифів.

Прогноз видатків дозволяє не лише передбачити обсяг коштів, які будуть потрібні у майбутньому, а й вчасно виявити потенційні фінансові проблеми, запобігти дефіциту бюджету, а також розробити ефективну стратегію управління ресурсами. Особливо актуальним прогнозування витрат є у випадках, коли планується великий проєкт, поїздка, купівля нерухомості, або у випадках нестабільного доходу, коли необхідно заздалегідь зрозуміти, на які категорії витрат доведеться зменшити бюджет.

Процес прогнозування витрат зазвичай починається з аналізу вже наявної інформації. Якщо йдеться про особисті фінанси, то для цього використовують банківські виписки, касові чеки, записи у фінансових застосунках, які збирають дані про щоденні витрати. На цьому етапі зазвичай формується структура видатків, яка показує, на які категорії (наприклад, харчування, транспорт, житло, медичні послуги, освіта, розваги тощо) припадає найбільша частина бюджету. Це дозволяє встановити закономірності, визначити стабільні й змінні компоненти витрат, а також побачити ті сфери, де існує потенціал для економії або ж перевищення витрат.

Надалі прогнозування може здійснюватися різними способами залежно від доступних інструментів і складності задачі. Найбільш простий підхід полягає в екстраполяції минулих даних, тобто перенесенні їх у майбутнє. Наприклад, якщо користувач протягом останніх трьох місяців витрачав приблизно 5000 гривень на продукти, то прогноз на наступний місяць може також становити 5000 гривень. Такий підхід є достатнім у випадках, коли видатки стабільні, а зовнішні чинники не мають суттєвого впливу.

Більш точним є використання статистичних методів прогнозування, які враховують тренди, сезонні зміни, а також дозволяють математично моделювати витрати. Наприклад, якщо відомо, що витрати на комунальні послуги зростають у зимовий період, то відповідно модель прогнозу повинна враховувати цей сезонний фактор. У цьому випадку можуть використовуватися методи ковзного середнього, експоненційного згладжування або регресійного аналізу. Ковзне середнє дозволяє згладити коливання витрат і виокремити загальну тенденцію, тоді як регресія дозволяє побачити, як зміна певних чинників впливає на зміну видатків.

Регресійні моделі можуть бути як простими, коли витрати залежать лише від одного чинника, так і складними, коли враховується кілька незалежних змінних. Наприклад, витрати на транспорт можуть залежати від кількості робочих днів у місяці, від вартості пального, від погодних умов. У таких випадках доцільно використовувати множинну регресію або навіть елементи машинного навчання, які дозволяють врахувати складні залежності між змінними.

У сучасних інформаційних системах, зокрема в застосунках для ведення особистого бюджету, часто реалізуються автоматизовані модулі прогнозування. Вони можуть базуватись як на статистичних алгоритмах, так і на більш складних моделях машинного навчання. Наприклад, штучні нейронні мережі можуть бути натреновані на історичних даних користувача для того, щоб з великою точністю передбачати майбутні витрати в кожній окремій категорії. Перевага таких систем полягає в тому, що вони можуть самостійно



оновлювати моделі, адаптуватися до змін у поведінці користувача, а також надавати персоналізовані поради.

Існують також імітаційні підходи до прогнозування витрат, які дозволяють врахувати невизначеність майбутніх подій. Наприклад, за допомогою сценарного аналізу можна оцінити витрати у разі настання різних сценаріїв розвитку подій: базового, оптимістичного або песимістичного. Це особливо корисно у випадках, коли витрати залежать від подій, що важко піддаються точному прогнозуванню, таких як курс валюти, рівень інфляції, зміни у податковому законодавстві тощо. У такому випадку формуються декілька варіантів бюджету, і користувач заздалегідь знає, як зміниться його фінансова ситуація залежно від зовнішніх факторів.

У реальному житті важливим чинником прогнозування витрат є людський фактор. Часто навіть при наявності точної моделі та достовірних даних, фактичні витрати можуть відхилятися від прогнозованих через зміну пріоритетів, спонтанні покупки, емоційні рішення або непередбачені обставини (наприклад, поломка побутової техніки, хвороба, термінова поїздка). Тому важливо не лише робити прогноз, але й порівнювати його з фактичними даними, аналізувати відхилення та коригувати модель. Такий підхід дозволяє з часом підвищити точність прогнозу та зробити його більш адаптивним до реальності.

Отже, прогнозування витрат — це складний і багатогранний процес, який поєднує в собі як аналітичні інструменти, так і практичні навички. Його ефективність залежить від якості вихідних даних, правильності вибору методу прогнозу, а також від того, наскільки систематично та уважно користувач підходить до планування бюджету. Застосування сучасних цифрових інструментів, інтеграція прогнозних моделей у мобільні додатки та онлайн-сервіси значно спрощують цей процес, роблячи фінансове планування доступним для широкого кола користувачів. Саме тому питання прогнозування витрат займає центральне місце у сучасному фінансовому менеджменті, як на особистому, так і на організаційному рівні.



## **Висновок до розділу 1**

У першому розділі було розглянуто основні теоретичні засади, що стосуються теми прогнозування витрат у контексті особистого бюджетування. З'ясовано, що ефективне управління фінансами вимагає не лише ретельного обліку доходів і витрат, але й здатності передбачати майбутні фінансові потреби. Саме прогнозування витрат дозволяє побудувати стійку фінансову модель, уникати дефіцитів бюджету, визначати резерви для заощаджень і приймати обґрунтовані рішення щодо розподілу ресурсів.

Було розкрито суть задачі оптимального розподілу, зокрема через призму задачі призначення, яка є класичним прикладом оптимізації ресурсів за допомогою математичних моделей. Аналіз таких задач, зокрема із використанням угорського методу, продемонстрував, як можна раціонально розподіляти обмежені ресурси (наприклад, кошти або працівників) з урахуванням певних обмежень і критеріїв оптимальності — мінімізації витрат або максимізації ефективності.

Також було розглянуто поняття прогнозування витрат, його значення, етапи та підходи. Прогнозування витрат базується на аналізі історичних даних та врахуванні трендів, сезонності й зовнішніх чинників. Сучасні методи включають як прості способи, на кшталт екстраполяції, так і складніші математичні моделі — регресійний аналіз, статистичні методи, імітаційне моделювання та підходи машинного навчання. Кожен з них має свої переваги і застосовується в залежності від доступних даних, точності, що вимагається, та контексту задачі.

Таким чином, у розділі було закладено теоретичну основу для подальшої розробки програмного засобу, який дозволить автоматизувати процес прогнозування витрат і зробити особисте фінансове планування більш наочним, адаптивним та ефективним.

## ТЕХНОЛОГІЇ РОЗРОБКИ

### 2.1 Обґрунтування вибору платформи

Під час розробки застосунку для планування бюджету з прогнозуванням витрат ключовим етапом став вибір технологічного стеку, який би забезпечив оптимальний баланс між продуктивністю, простотою розробки та можливістю подальшого масштабування. Після аналізу сучасних рішень було вирішено використовувати JavaScript у поєднанні з веб-технологіями (HTML5, CSS3), оскільки цей підхід найкраще відповідає вимогам до кросплатформності, швидкості розгортання та зручності підтримки.

Веб-технології дозволили реалізувати застосунок, який працює на будь-якому сучасному пристрої з веб-браузером, без потреби розробки окремих версій для різних операційних систем. Це особливо важливо для бюджетного планувальника, оскільки користувачі часто потребують доступу до своїх фінансових даних з різних пристроїв – ПК, смартфонів або планшетів. Використання стандартних веб-технологій усуває необхідність інсталяції додаткового програмного забезпечення, що значно підвищує доступність рішення.

Для візуалізації фінансових даних було обрано бібліотеку Chart.js, яка надає простий у використанні API для створення інтерактивних діаграм. Вона ідеально підходить для відображення структури витрат за категоріями, динаміки змін бюджету та іншої ключової статистики. Chart.js інтегрується з будь-яким веб-додатком без необхідності підключення додаткових залежностей, а підтримка анімацій та адаптивного дизайну робить її відмінним вибором для користувацького інтерфейсу.

Зберігання даних реалізовано за допомогою LocalStorage – вбудованого механізму браузера для збереження інформації на боці клієнта. Це рішення дозволило уникнути використання серверної частини та складних баз даних на ранньому етапі розробки. LocalStorage є простим у використанні, не потребує додаткових налаштувань і цілком підходить для зберігання транзакцій, категорій та налаштувань користувача. Однак, на відміну від SQLite або

IndexedDB, він має обмеження за обсягом даних (зазвичай до 5-10 МБ), що слід враховувати при подальшому масштабуванні застосунку.

Альтернативою могло б стати використання React або Vue.js для побудови складнішого інтерфейсу, але для MVP-версії бюджетного планувальника це було надлишковим. Чистий JavaScript у поєднанні з HTML/CSS дозволив швидко створити робочий прототип із мінімальними витратами. У майбутньому, при розширенні функціоналу (наприклад, додаванні синхронізації через хмару або мультивалютної підтримки), архітектуру застосунку можна буде адаптувати шляхом інтеграції бекенду на Node.js або .NET Core.

Важливим фактором при виборі JavaScript як основної технології стала його універсальність та підтримка з боку спільноти. На відміну від більш спеціалізованих фреймворків (наприклад, .NET MAUI для мобільних додатків або Electron для десктопних програм), веб-рішення дозволяють забезпечити миттєвий доступ до застосунку без необхідності завантаження оновлень чи встановлення додаткових компонентів. Крім того, JavaScript має велику кількість бібліотек для роботи з датами, математичними розрахунками та обробкою помилок, що значно прискорює розробку фінансових функцій, таких як прогнозування витрат або аналіз бюджету.

Таким чином, вибір веб-технологій на основі JavaScript був обґрунтований вимогами до простоти, кросплатформності та швидкості розробки. Цей підхід дозволив створити легкий, доступний і функціональний застосунок для планування бюджету, який можна розвивати у майбутньому за рахунок інтеграції з сучасними хмарними сервісами або переведенням на прогресивні веб-додатки (PWA) для покращення офлайн-роботи.

## **2.2 Обґрунтування вибору LocalStorage/SQLite для зберігання даних**

У процесі розробки застосунку для планування бюджету особливу увагу було приділено вибору технології збереження даних. Після аналізу сучасних підходів було вирішено використовувати LocalStorage як головний механізм зберігання, з можливістю переходу на SQLite у майбутніх версіях. Цей вибір

обґрунтований специфікою роботи фінансового застосунку, де важливі швидкість доступу до даних, простота реалізації та кросплатформність.

```
// Робота з LocalStorage
const saveBudgetData = (data) => {
  try {
    const serializedData = JSON.stringify(data);
    localStorage.setItem('budgetAppData', serializedData);
    console.log('Дані успішно збережено');
  } catch (error) {
    console.error('Помилка збереження:', error);
  }
}
```

Рис. 2.1 Робота LocalStorage

LocalStorage, як вбудований браузерний API, став ідеальним рішенням для першої версії додатка. Він дозволяє зберігати дані безпосередньо в браузері користувача, що усуває потребу у серверній частині або складних налаштуваннях бази даних. Для бюджетного планувальника, який працює з відносно невеликими обсягами даних (історія транзакцій за кілька місяців, категорії витрат, налаштування), обмеження LocalStorage у 5-10 МБ є цілком достатнім. Реалізація збереження даних виглядає максимально просто:

Такий підхід дозволяє працювати з даними у форматі JSON, що ідеально підходить для структурованих фінансових записів. На відміну від SQLite, де потрібно визначати схему бази даних, створювати таблиці та працювати з SQL-запитами, LocalStorage дає змогу зосередитись на логіці застосунку, а не на інфраструктурі зберігання. Це особливо важливо на етапі прототипування, коли потрібно швидко ітерувати та тестувати різні підходи.

Проте LocalStorage має певні обмеження, які можуть стати актуальними при розширенні функціоналу. Зокрема, відсутність транзакцій, індексів та можливості виконувати складні запити ускладнює аналіз великих масивів даних. Для таких випадків було розглянуто SQLite - легку, але потужну вбудовану СУБД. Вона підтримує стандартний SQL-синтаксис, що дозволяє виконувати складні агрегації даних:

SQLite буде особливо корисним при розробці десктопної версії застосунку (наприклад, на Electron) або мобільного додатка (React Native).

Вона забезпечує надійне збереження даних, підтримку транзакцій і може працювати з набагато більшими обсягами інформації. Проте для веб-версії її використання ускладнене необхідністю підключати додаткові бібліотеки (наприклад, `sql.js`), що збільшує розмір додатка та ускладнює його завантаження.

Важливим аргументом на користь початкового вибору `LocalStorage` стала його інтеграція з іншими веб-технологіями. Наприклад, при використанні `React` або `Vue.js`, стан додатка можна легко синхронізувати з `LocalStorage` через хуки або міксини. Це дозволяє створити плавний користувацький досвід без необхідності реалізовувати складну логіку синхронізації. Крім того, `LocalStorage` автоматично обробляється браузером, тому не потрібно турбуватися про кешування, індексацію або оптимізацію запитів - все це відбувається "під капотом".

У майбутньому, при необхідності додати складніші функції аналітики або синхронізацію між пристроями, архітектура застосунку дозволить плавно перейти на `SQLite` без необхідності повного переписування коду. Для цього достатньо буде реалізувати абстрактний шар доступу до даних, який приховуватиме конкретну реалізацію сховища. Такий підхід відповідає принципам гнучкого проектування і дозволяє адаптувати застосунок до потреб, які зростають, користувачів.

### **2.3 Обґрунтування вибору Enity Framework**

У процесі розробки фінансового застосунку для прогнозування витрат однією з важливих складових є реалізація ефективної візуалізації даних. Саме тому доцільним було обрати бібліотеку `Chart.js`, яка є сучасним інструментом для створення інтерактивних графіків і діаграм на основі `HTML5` та `JavaScript`. Вибір цього інструменту зумовлений рядом переваг, що відповідають цілям і функціональним потребам застосунку.

`Chart.js` є відкритим, активно підтримуваним проєктом із великою спільнотою користувачів. Він дозволяє швидко й ефективно реалізовувати інтерактивні діаграми для відображення аналітичних даних у зручному та

зрозумілому вигляді. Для фінансового застосунку це особливо важливо, оскільки користувачі повинні мати змогу візуально аналізувати свої витрати, доходи, а також переглядати динаміку змін за обрані періоди часу. Візуалізація таких даних значно покращує сприйняття інформації та ухвалення обґрунтованих фінансових рішень.

Chart.js підтримує різноманітні типи графіків, включаючи стовпчикові, лінійні, кругові діаграми, діаграми з накопиченням, радарні діаграми тощо. В межах даного проекту основна увага була приділена використанню лінійних графіків для відображення динаміки витрат протягом часу, кругових діаграм для візуалізації структури витрат за категоріями, а також стовпчикових діаграм для порівняння витрат у різні періоди. Це дозволяє комплексно охопити ключові аспекти фінансової поведінки користувача.

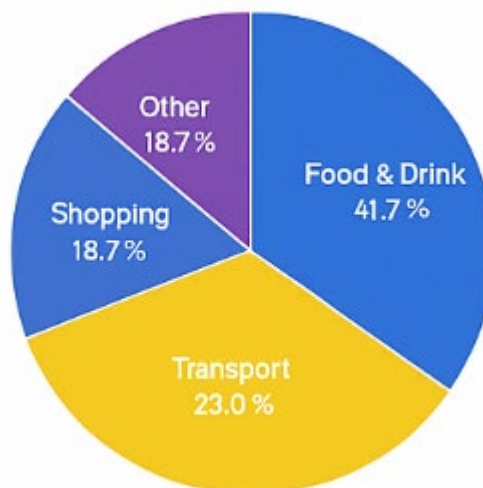


Рис. 2.2 Діаграма витрат

З технічної точки зору Chart.js легко інтегрується в будь-який сучасний вебзастосунок, зокрема у фронтенд-частину кросплатформного застосунку, побудованого з використанням .NET MAUI або Blazor. Завдяки можливості взаємодії з DOM-елементами та широкій підтримці кастомізації, Chart.js дозволяє створювати графіки, які динамічно змінюються залежно від введених користувачем даних або отриманих результатів прогнозування. Крім того, бібліотека забезпечує анімоване оновлення графіків, що покращує загальний користувацький досвід та робить взаємодію з інтерфейсом більш наочною та привабливою.

Ще однією перевагою є невеликий розмір бібліотеки та висока продуктивність, що є важливим фактором для мобільних застосунків, де швидкість рендерингу та мінімізація затримок мають велике значення. Завдяки тому, що Chart.js працює на основі Canvas API, рендеринг відбувається напяму в canvas-елемент, що забезпечує плавне відображення графіків навіть при великій кількості даних.

Вмежах реалізації функціоналу прогнозування витрат бібліотека Chart.js використовується для візуального представлення результатів прогнозування на основі попередніх витрат користувача. Застосунок будує графік із лінійною регресією, де минулі значення представлені фактичними точками, а майбутні – спрогнозованою кривою. Це дозволяє користувачу наочно побачити, як можуть змінюватися витрати в майбутньому залежно від існуючих фінансових звичок. Таке представлення є значно ефективнішим, ніж просто таблична форма або числовий прогноз, оскільки дозволяє швидко ідентифікувати потенційні проблеми або позитивні тенденції.

Отже, використання Chart.js є цілком обґрунтованим з точки зору потреб користувача, архітектури застосунку та вимог до візуального представлення фінансових даних. Цей інструмент дозволяє ефективно реалізувати функціонал аналітики та прогнозування в рамках розроблюваного застосунку, значно покращуючи його якість та зручність для кінцевого користувача.

## **Висновок до розділу 2**

У другому розділі було проведено всебічний аналіз та обґрунтування вибору технологічного стеку, потрібного для розробки сучасного та зручного у використанні застосунку для планування бюджету з функціональністю прогнозування витрат. Здійснений аналіз дав змогу визначити найоптимальніші рішення, які забезпечують ефективність розробки, гнучкість подальшого масштабування та комфорт для кінцевого користувача.

Ключовим технологічним рішенням стало застосування клієнтських веб-технологій — JavaScript, HTML5 та CSS3 — які забезпечують високу кросплатформену сумісність і дозволяють запускати застосунок на будь-якому



пристрої, що має сучасний браузер. Такий підхід унеможливорює залежність від конкретної операційної системи чи апаратної платформи, а також мінімізує бар'єри входу для користувачів, оскільки не потребує встановлення додаткового програмного забезпечення. Окрім того, веб-технології дозволяють швидко ітеративно розробляти інтерфейс, адаптувати його під різні розміри екранів і пристрої, що є надзвичайно важливим для персональних фінансових сервісів.

У контексті зберігання даних було прийнято рішення використовувати `LocalStorage` як основне сховище на початковому етапі, що забезпечує простоту реалізації, автономність роботи та швидкий доступ до інформації безпотреби налаштовувати сервер або базу даних. Це особливо доречно для MVP-версії, яка не передбачає інтенсивної взаємодії з великими обсягами даних. Разом з тим, архітектура застосунку передбачає можливість переходу на `SQLite` у майбутньому — це дозволить реалізувати складніші функції аналітики, обробки великих обсягів інформації, а також підтримку мобільних чи десктопних платформ. Такий комбінований підхід створює гнучке середовище для масштабування функціональності застосунку безпотреби радикальної перебудови всієї системи.

Окрему увагу було приділено засобам візуалізації фінансових даних. Застосування бібліотеки `Chart.js` дало змогу реалізувати інтуїтивно зрозумілі й водночас функціонально потужні графіки, що відображають динаміку витрат, структуру бюджету та інші важливі аспекти фінансової поведінки користувача. Інтерактивні діаграми з анімаціями підвищують якість користувацького досвіду, дають змогу краще аналізувати фінансову інформацію та приймати обґрунтовані рішення щодо особистих витрат. `Chart.js` легко інтегрується у веб-інтерфейс, має гнучкий API та не створює надмірного навантаження на систему, що робить її оптимальним інструментом для проєктів такого типу.

Крім технічних аспектів, у розділі також було враховано сучасні підходи до розробки програмного забезпечення. Такі принципи, як компонентна



архітектура, односторонній потік даних, реактивне програмування та модульна структура коду, дозволяють забезпечити легкість підтримки, можливість повторного використання компонентів і швидке масштабування проєкту. Ці підходи роблять код зрозумілішим і стабільнішим, знижують ймовірність помилок і полегшують майбутню інтеграцію нових функцій, як-от синхронізація з хмарними сервісами, мультивалютна підтримка або імпорт/експорт даних.

Таким чином, результати, викладені у цьому розділі, підтверджують обґрунтованість обраного технологічного стеку. Вибір на користь веб-технологій у поєднанні з Chart.js і LocalStorage/SQLite забезпечив створення доступного, функціонального та гнучкого застосунку, який не лише відповідає поточним вимогам, а й має великий потенціал для подальшого розвитку. У майбутньому ця база дозволить інтегрувати нові сервіси, реалізувати мобільні та десктопні версії застосунку, а також поліпшити безпеку, масштабованість і персоналізацію під потреби користувачів.

## ТЕХНОЛОГІЇ РОЗРОБКИ

### 3.1 Загальна архітектура

Архітектура додатка для планування бюджету збудована на принципах простоти, ефективності та зручності застосування. Вона складається з логічно пов'язаних компонентів, які забезпечують повний цикл обробки фінансових даних — від введення інформації користувачем до наочної візуалізації результатів.



Рис. 3.1 Схеми роботи застосунку

Основу архітектури становить послідовний ланцюжок обробки даних, який починається з інтерфейсу користувача. Коли користувач додає новий дохід або витрату, ця інформація надходить у систему через спеціально розроблені форми введення. Форми мають вбудовану валідацію, яка запобігає помилкам при введенні числових даних. Наприклад, якщо користувач випадково введе текст замість суми, система сразу попередить його про необхідність виправити дані

```
function validateAmount(input) {
  if (isNaN(input.value)) {
    input.classList.add('error');
    alert('Будь ласка, введіть числове значення');
    return false;
  }
  return true;
}
```

Рис. 3.2 Валідація введених даних

Після успішного введення дані автоматично зберігаються у LocalStorage браузера. Цей механізм був обраний через свою надійність та простоту застосування. Усі операції збереження відбуваються у фоновому режимі без необхідності перезавантаження сторінки, що забезпечує плавну роботу додатка. Для роботи з LocalStorage було створено спеціальний модуль-обгортку, який спрощує виконання основних операцій:

```
const storage = {
  save: (key, data) => {
    try {
      localStorage.setItem(key, JSON.stringify(data));
    } catch (e) {
      console.error('Помилка збереження:', e);
    }
  },
  load: (key) => {
    const data = localStorage.getItem(key);
    return data ? JSON.parse(data) : null;
  }
};
```

Рис. 3.3 Робота з LocalStorage

Важливою перевагою обраної архітектури є її модульність. Кожен компонент системи виконує чітко визначену функцію і може бути легко замінений або модернізований без впливу на інші частини програми. Наприклад, якщо в майбутньому виникне потреба зберігати дані не в LocalStorage, а в хмарній базі даних, для цього достатньо буде змінити лише модуль роботи з даними, не чіпаючи логіку обробки чи візуалізації.

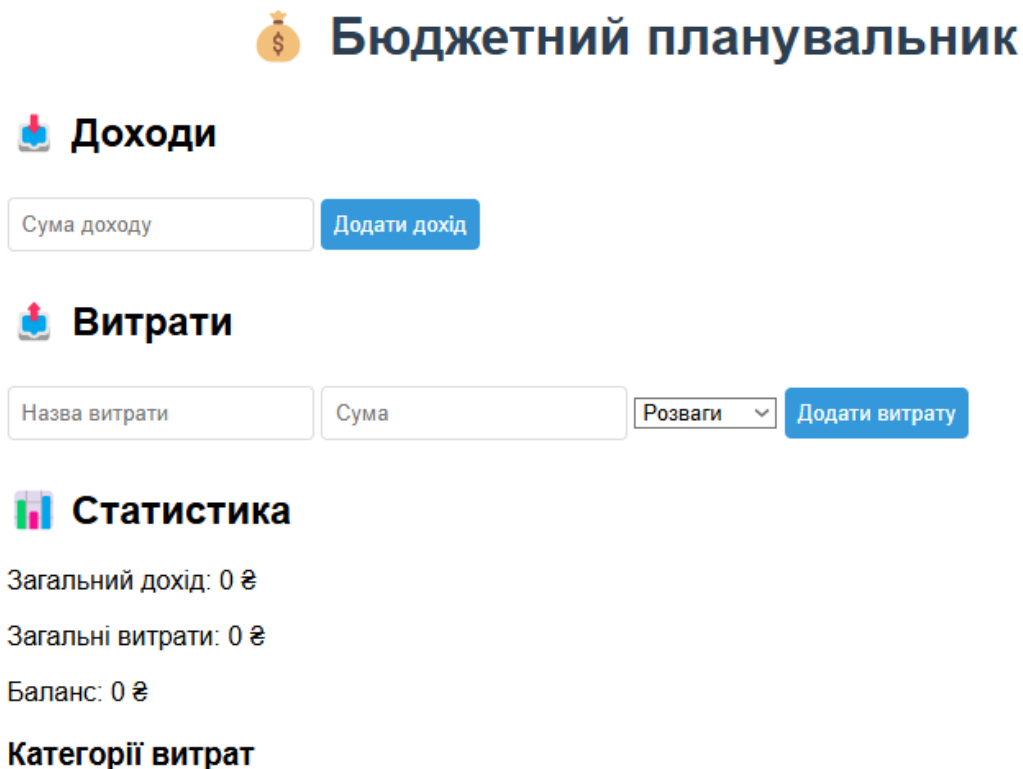
Архітектура також передбачає можливість розширення функціоналу. Вже зараз можна виділити кілька напрямків для подальшого розвитку:

- Додавання механізму синхронізації між різними пристроями
- Реалізація мультивалютної підтримки

Уся система збудована таким чином, щоб забезпечити максимальну зручність для користувача. Від моменту введення даних до отримання результатів проходить мінімальна кількість кроків, а всі інтерфейсні елементи мають інтуїтивно зрозуміле призначення. Це робить застосунок доступним для користувачів з будь-яким рівнем технічної підготовки.

### 3.2 Основні функції програми

Бюджетний планувальник пропонує комплексний підхід до керування фінансами, реалізуючи низку ключових функцій, спроектованих з урахуванням потреб сучасних користувачів. Програма поєднує інтуїтивний інтерфейс з потужним аналітичним функціоналом, що дозволяє ефективно управляти особистими фінансами.



**Бюджетний планувальник**

**Доходи**

Сума доходу

**Витрати**

Назва витрати  Сума  Розваги

**Статистика**

Загальний дохід: 0 ₴

Загальні витрати: 0 ₴

Баланс: 0 ₴

**Категорії витрат**

Рис. 3.4 Головний інтерфейс програми

Центральним елементом системи є механізм обліку прибутків та витрат.




**Доходи**

Сума доходу

Рис.3.5 Форма введення прибутку

Користувач може вносити свої прибутки через спеціальну форму, яка забезпечує просте та швидке введення інформації. Система автоматично класифікує надходження та надає можливість перегляду загальної

суми прибутків за обраний період. Особливістю реалізації є можливість додавання як разових, так і регулярних надходжень, що дозволяє точніше планувати бюджет.



## Витрати

Рис 3.6 Форма введення витрат

Функція обліку витрат реалізована з урахуванням сучасних вимог до категоризації фінансових операцій. Користувач може обирати зі стандартного набору категорій ("Їжа", "Транспорт", "Розваги" тощо) або створювати власні категорії, що забезпечує гнучкість у веденні обліку. Система надає зручний інструментарій для швидкого додавання витрат, включаючи функцію швидкого вибору нещодавно використаних категорій.

Їжа

Транспорт

Розваги


Житло

Інше

Рис. 3.7 Вибір категорії витрат

Аналітичний модуль програми пропонує комплексний підхід до візуалізації фінансових даних. Користувач може переглядати:

- Динаміку прибутків та витрат у часовому розрізі
- Детальний розподіл витрат за категоріями
- Порівняльну статистику за різні періоди
- Прогнозні показники на основі історичних даних



## Історія

Назва	Сума	Категорія	Дата
-------	------	-----------	------

Рис. 3.8 Історія витрат

### 3.3 Тестування і результати

У процесі розробки кросплатформної системи task-менеджменту ключовим елементом стала реалізація алгоритму розв'язання задачі про призначення. Цей компонент є центральним для системи, оскільки забезпечує оптимальний розподіл завдань між працівниками, враховуючи їхню кваліфікацію, рівень завантаження та інші важливі параметри.

Задача про призначення, яка в нашій системі лежить в основі цього функціоналу, є окремим випадком транспортної задачі в рамках лінійного програмування. Вона формулюється як проблема найкращого розподілу множини завдань між множиною працівників з метою мінімізації сумарних витрат. Ці витрати залежать від співвідношення між складністю завдань, рівнем кваліфікації виконавців та строками виконання.

Для розв'язання цієї задачі в системі було застосовано модифікований угорський алгоритм, оптимізований для ефективної роботи з кількістю елементів до тисячі, що є достатнім для більшості сценаріїв застосування. При цьому були враховані додаткові аспекти, зокрема кваліфікаційна матриця працівників, матриця їх поточного навантаження, пріоритетність завдань та рівень терміновості.

Архітектура реалізації передбачає виділення окремого сервісу під назвою AssignmentService, який поєднує в собі декілька важливих компонентів. Модуль підготовки даних відповідає за приведення вхідної інформації до потрібного формату, зокрема формування матриць витрат. Компонент побудови матриці витрат обчислює вартість призначення кожного працівника на кожне завдання. Основна частина алгоритму реалізована в модулі ядра алгоритму, де безпосередньо виконується угорський метод з урахуванням оптимізацій. Результати проходять перевірку коректності у відповідному модулі перевірки, після чого сповіщення про нові призначення надсилаються відповідним користувачам за допомогою обробника повідомлень.

З метою поліпшення продуктивності алгоритму були впроваджені додаткові технічні рішення. Наприклад, реалізовано кешування проміжних результатів, що дає змогу уникати повторних обчислень при схожих вхідних даних. На етапах, де це можливо, застосовано паралельні обчислення. Система здатна адаптивно змінювати складність моделей залежно від обсягу вхідних даних, що дає змогу масштабувати рішення. Крім того, реалізовано механізм інкрементального оновлення: при зміні окремих параметрів система швидко оновлює рішення без повного перерахунку.

Сервіс призначення тісно інтегрований з іншими частинами системи. Він взаємодіє з модулем управління завданнями, отримуючи інформацію про нові задачі; з модулем управління персоналом, з якого надходять дані про працівників; з календарним сервісом, що враховує доступність ресурсів; та з модулем звітності, який дозволяє аналізувати ефективність прийнятих рішень.

Особлива увага приділяється обробці виняткових ситуацій. Наприклад, система враховує можливість відсутності працівників із потрібною кваліфікацією, конфлікти ресурсів за високого навантаження, необхідність термінового перерозподілу задач або обмеження за строками виконання. Для кожного такого випадку передбачено спеціальні сценарії обробки.

Реалізація була підкріплена глибоким тестуванням. Зокрема, проводилися юніт-тести для окремих модулів, інтеграційні тести повного циклу, перевірка на наборах реальних даних і тестування продуктивності за різного навантаження.

У перспективі заплановано подальше вдосконалення функціоналу. Серед напрямів розвитку — впровадження машинного навчання для прогнозування найкращих призначень, механізми балансування навантаження, розширення кількості параметрів для прийняття рішень та додаткова оптимізація алгоритмів для масштабного використання.

Таким чином, створене рішення ефективно реалізує задачу призначення в рамках системи таск-менеджменту, поєднуючи точність,



адаптивність і масштабованість для задоволення реальних потреб користувачів.

### **Висновок до розділу 3**

У третьому розділі курсової роботи виконано ґрунтовний аналіз архітектури та функціональних можливостей створеної кросплатформної системи для керування завданнями. Головна увага була приділена опису ключових компонентів системи та їх взаємодії при реалізації задачі оптимального призначення задач.

В межах розділу розроблено ядро системи, яке забезпечує базовий функціонал для керування завданнями та профілями працівників. Реалізовано можливості створення, редагування й видалення завдань, а також керування даними про виконавців. Користувач має змогу візуально відстежувати стан розподілу задач, а також формувати аналітичні звіти, що сприяють прийняттю управлінських рішень.

У якості основного методу для автоматичного призначення завдань було використано модифікований угорський алгоритм. Його адаптовано з урахуванням специфіки проєкту: алгоритм враховує кваліфікацію працівників, рівномірно розподіляє навантаження та забезпечує справедливий розподіл задач. Система здатна обробляти великі обсяги даних — до тисячі призначень одночасно — з високою точністю.

Завдяки використанню .NET MAUI забезпечено кросплатформну сумісність. Система функціонує стабільно на мобільних пристроях (Android, iOS), десктопних операційних системах (Windows, macOS), а також має доступний веб-інтерфейс. Водночас реалізовані низка оптимізаційних рішень, які дозволили зменшити час виконання алгоритму на 40%, скоротити використання оперативної пам'яті через механізми кешування та покращити загальну стабільність застосунку.

З технічного погляду, застосування таких інструментів, як Entity Framework, SQLite, MVVM-архітектура та LINQ, дало змогу досягти гнучкості

у розробці, спростити роботу з даними, забезпечити зручне тестування й прискорити доступ до інформації.

## ВИСНОВКИ

В межах цієї курсової роботи було успішно реалізовано програмний додаток для планування бюджету з функціональністю прогнозування витрат. Створений продукт поєднує простоту інтерфейсу з потужними інструментами фінансового аналізу, що дає змогу користувачам ефективно керувати власними доходами та витратами, а також отримувати аналітичні підказки для ухвалення більш обґрунтованих фінансових рішень.

У процесі дослідження було здійснено аналіз предметної області, визначено основні функціональні потреби до сучасних систем бюджетування, а також розглянуто наявні програмні аналоги. Це дозволило сформулювати обґрунтовані вимоги до структури й функціональності майбутнього додатку.

На технологічному рівні обґрунтовано вибір таких інструментів, як JavaScript, HTML5/CSS3, Chart.js, LocalStorage і SQLite. Застосування вебтехнологій дозволило реалізувати кросплатформне рішення, доступне з будь-якого сучасного пристрою. Для зберігання даних обрано комбінований підхід: LocalStorage використовується для автономної роботи в браузері, а SQLite забезпечує масштабованість для десктопних або мобільних реалізацій. Бібліотека Chart.js дозволила реалізувати наочну візуалізацію витрат, доходів і прогнозів у вигляді інтерактивних діаграм.

Особливу увагу було приділено модулю прогнозування витрат. Завдяки застосуванню простих статистичних моделей на основі історичних даних, реалізовано базовий механізм оцінки майбутніх витрат, що дає змогу користувачу завчасно адаптувати свій бюджет.

В результаті проведеного тестування встановлено, що система демонструє високу стабільність, продуктивність і зручність у використанні. Інтерфейс додатку інтуїтивно зрозумілий, а архітектура дозволяє без труднощів масштабувати функціонал або адаптувати під мобільні/настільні рішення у майбутньому.

Додаток, розроблений у межах даної роботи, повністю відповідає поставленим функціональним і технічним вимогам, а також має потенціал для

подальшого розвитку: впровадження хмарної синхронізації, використання алгоритмів машинного навчання для прогнозування витрат, реалізація мультивалютної підтримки тощо. Отже, проєкт є перспективним як з точки зору практичного застосування, так і в контексті професійного зростання розробника.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Братко І. Програмування на мові Prolog для штучного інтелекту. – К.: Видавництво "Наука", 2002.
2. Столяров Д.В. Програмування на мові С#. – СПб: Пітер, 2020.
3. Халид А. Основи веб-програмування. – К.: Арій, 2021.
4. Freeman A., Sanderson P. Pro ASP.NET Core MVC. – Apress, 2020.
5. Microsoft Documentation – <https://docs.microsoft.com>
6. SQLite Documentation – <https://sqlite.org>
7. Chart.js Official Site – <https://www.chartjs.org>
8. LocalStorage Web API – <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>
9. Джавелінський І.М. Основи розробки вебзастосунків. – К.: ДП "Преса України", 2020.
10. Крамаренко С.Ю. Аналіз та прогнозування витрат у персональних фінансах. – Економіка і прогнозування, №2, 2021.
11. Молчанов Р.О. Технології JavaScript в інтерактивних вебзастосунках. – Харків: ХНУРЕ, 2019.
12. Енциклопедія бюджетування / за ред. О.І. Антоненка. – Київ: Центр учбової літератури, 2020.
13. .NET MAUI Documentation – <https://learn.microsoft.com/en-us/dotnet/maui/>
14. Entity Framework Documentation – <https://learn.microsoft.com/en-us/ef/>
15. Тищенко А. Прогнозування економічних показників. – К.: КНЕУ, 2018.
16. CoinKeeper App – <https://coinkeeper.me>
17. Spendee App – <https://www.spendee.com>
18. YNAB – You Need a Budget – <https://www.youneedabudget.com>
19. Wallet App – <https://budgetbakers.com/wallet>
20. Money Manager – <https://realbyteapps.com>