Полтавський університет економіки і торгівлі

Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___»_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

«РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ОПТИМІЗАЦІЇ ОСОБИСТИХ ФІНАНСІВ СТУДЕНТІВ»

зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня бакалавра

Виконавець роботи Люлька Владислав Володимирович
_____«____»_____ 202_ р.
(підпис)

Науковий керівник к. ф.-м. н., доцент, Ольховська Олена Володимирівна
_____«____»_____ 202_ р.
(підпис)

Рецензент _____

ПОЛТАВА 2026

ЗМІСТ

ВСТУП 3

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ ОПТИМІЗАЦІЇ ОСОБИСТИХ ФІНАНСІВ СТУДЕНТСЬКОЇ МОЛОДІ 6

1.1 Особливості управління студентським бюджетом в цифровому середовищі 6

1.2 Чинники формування фінансової поведінки здобувачів освіти 9

1.3 Мобільні рішення для контролю доходів і витрат 11

Висновки до розділу 1 17

РОЗДІЛ 2. ПРОЄКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ОПТИМІЗАЦІЇ ОСОБИСТИХ ФІНАНСІВ СТУДЕНТІВ 18

2.1 Формування вимог до функціональних можливостей програмного продукту 18

2.2 Обґрунтування вибору кросплатформного середовища розробки 22

2.3 Архітектура мобільного додатку для персонального фінансового обліку 25

2.4 Моделювання взаємодії користувача з основними екранами системи 32

Висновки до розділу 2 38

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ СТУДЕНТСЬКОГО ФІНАНСОВОГО ПЛАНУВАННЯ 39

3.1 Реалізація структури застосунку засобами мобільної розробки 39

3.2 Інтерфейсне оформлення екранів обліку особистих фінансів 44

3.3 Перевірка роботи функцій контролю витрат і планування бюджету 53

3.4 Оцінювання результатів використання розробленого програмного продукту 59

Висновки до розділу 3 64

ВИСНОВКИ 66

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 68

ДОДАТОК А 71

ВСТУП

Актуальність теми. Розвиток цифрових сервісів фінансового планування посилює значення інструментів, орієнтованих на щоденне управління особистими коштами. Для студентів, питання раціонального використання доходів набуває особливого значення через обмежені фінансові ресурси, нерегулярність надходжень, залежність від стипендійної підтримки, допомоги родини, підробітку або інших джерел. Недостатній рівень фінансової дисципліни часто призводить до неконтрольованих витрат, відсутності резерву, складнощів з плануванням навчальних та побутових потреб.

Актуальність розробки мобільного додатку для оптимізації особистих фінансів студентів зумовлена потребою поєднання простого обліку доходів та витрат з практичними засобами бюджетного планування. Більшість універсальних фінансових сервісів орієнтується на широку аудиторію, тому не завжди враховує специфіку студентського способу життя, характер регулярних витрат, потребу в швидкому внесенні операцій і зрозумілій аналітиці.

Метою кваліфікаційної роботи є розробка мобільного додатку для оптимізації особистих фінансів студентів, який забезпечує облік доходів та витрат, планування бюджету, відображення фінансової статистики та підтримку прийняття рішень щодо раціонального використання коштів.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Проаналізувати особливості управління особистими фінансами студентської молоді.
2. Дослідити чинники, що впливають на фінансову поведінку здобувачів освіти.
3. Розглянути сучасні мобільні рішення для контролю доходів та витрат.
4. Сформувати вимоги до функціональних можливостей програмного продукту.

5. Обґрунтувати вибір технологічного стеку для створення мобільно орієнтованого вебзастосунку.

6. Спроєктувати архітектуру програмного продукту та логіку взаємодії користувача з основними екранами.

7. Реалізувати мобільний додаток для обліку особистих фінансів студентів.

8. Перевірити роботу основних функцій та оцінити результати використання розробленого програмного продукту.

Об’єктом дослідження є процес управління особистими фінансами студентів з застосуванням цифрових інструментів.

Предметом дослідження є методи та засоби проєктування та розробка мобільного додатку для оптимізації студентського бюджету.

Методи дослідження. В кваліфікаційній роботі використано метод аналізу наукових та практичних джерел для визначення особливостей фінансової поведінки студентів, метод порівняння для вивчення наявних сервісів фінансового планування, метод системного аналізу для формування вимог до програмного продукту, метод моделювання для побудови структури застосунку, а також, метод експериментальної перевірки для оцінки працездатності реалізованих функцій.

Практичне значення одержаних результатів полягає в створенні програмного продукту, який може бути використаний студентами для щоденного контролю доходів і витрат, визначення залишку бюджету, аналізу фінансових звичок і планування подальших витрат. Розроблений додаток може застосовуватись як демонстраційний програмний засіб під час вивчення питань фінансової грамотності та прикладної веброзробки.

Структура кваліфікаційної роботи складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. В першому розділі розглянуто теоретичні аспекти оптимізації особистих фінансів студентської молоді. В другому розділі виконано проєктування мобільного додатку для оптимізації особистих фінансів студентів. В третьому розділі наведено

програмну реалізацію, описано інтерфейс, перевірено роботу функцій та подано оцінювання результатів використання розробленого програмного продукту.

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ ОПТИМІЗАЦІЇ ОСОБИСТИХ ФІНАНСІВ СТУДЕНТСЬКОЇ МОЛОДІ

1.1 Особливості управління студентським бюджетом в цифровому середовищі

Управління студентським бюджетом є важливою складовою особистої фінансової організації, оскільки період навчання часто супроводжується обмеженими доходами, нерівномірними надходженнями коштів та високою залежністю від поточних побутових потреб. Студентський бюджет формується під впливом навчального графіку, місця проживання, наявності стипендії, підтримки родини, підробітку, транспортних витрат, харчування, оплати зв'язку, навчальних матеріалів та дозвілля.

Невеликі щоденні платежі можуть суттєво впливати на загальний залишок коштів наприкінці тижня або місяця. За відсутності системного обліку студент часто не бачить реального співвідношення між доходами, обов'язковими витратами та сумою, доступною для вільного використання [1]. Основні характеристики студентського бюджету узагальнено в таблиці 1.1.

Таблиця 1.1 – Основні характеристики студентського бюджету

Характеристика	**Прояв у фінансовій поведінці студента**	**Вплив на бюджетне планування**
Обмежені доходи	Основним джерелом коштів може бути стипендія, допомога родини або нерегулярний підробіток	Потрібен постійний контроль залишку коштів та планування витрат на короткі періоди
Нерівномірність надходжень	Грошові надходження можуть мати різну періодичність і різний обсяг	Виникає потреба у резервуванні частини коштів для обов'язкових платежів
Переважання дрібних витрат	Значна частина платежів пов'язана з харчуванням, транспортом, зв'язком і навчальними потребами	Необхідне швидке внесення операцій без складної процедури заповнення форми
Сезонність фінансового	На початку семестру або перед сесією збільшуються витрати на	Варто передбачати планування витрат за періодами

Характеристика	Прояв у фінансовій поведінці студента	Вплив на бюджетне планування
навантаження	матеріали, проїзд чи проживання	навчального року
Залежність від цифрових платежів	Оплата часто здійснюється банківськими картками, онлайн-сервісами або мобільними гаманцями	Зростає значення мобільного інтерфейсу для оперативного контролю бюджету

Наведені характеристики свідчать, що студентський бюджет має динамічну структуру та потребує інструментів поточного контролю. На відміну від стабільного сімейного або корпоративного бюджету, фінанси студента часто змінюються залежно від навчального навантаження, побутових обставин і можливості тимчасового заробітку.

Цифрове середовище змінює характер управління особистими фінансами, оскільки більшість фінансових дій може виконуватись через смартфон або браузер. Студент отримує змогу вносити витрати одразу після покупки, переглядати залишок бюджету, аналізувати категорії витрат і визначати найбільш навантажені напрями використання коштів [2]. Напрями використання цифрових інструментів для управління студентським бюджетом наведено в таблиці 1.2.

Таблиця 1.2 – Напрями використання цифрових інструментів для управління студентським бюджетом

Напрям використання	Функціональне значення	Очікуваний результат для користувача
Облік доходів	Фіксація надходжень від стипендії, родини, підробітку або інших джерел	Формування реальної суми доступного бюджету
Облік витрат	Запис повсякденних платежів із зазначенням суми та категорії	Виявлення основних напрямів витрачання коштів
Категоризація операцій	Розподіл витрат за групами, пов’язаними з навчанням, харчуванням, транспортом і дозвіллям	Спрощення аналізу фінансової поведінки
Бюджетні обмеження	Встановлення допустимого рівня витрат на певний період	Зниження ризику перевищення доступної

Напрям використання	**Функціональне значення**	**Очікуваний результат для користувача**
		суми коштів
Візуальна аналітика	Відображення підсумків у вигляді діаграм, карток і показників	Швидке розуміння фінансового стану без складних розрахунків
Локальне збереження інформації	Збереження операцій в пам’яті браузера або пристрою	Можливість користування застосунком без складної реєстрації

Цифрові інструменти мають особливе значення для студентів через звичку користуватись мобільними пристроями в навчанні, комунікації та побуті. Якщо фінансовий облік інтегрований в зручний мобільний інтерфейс, користувач частіше вносить операції та отримує повнішу картину власного бюджету. Наявність зрозумілих категорій витрат допомагає не лише фіксувати платежі, але й поступово змінювати фінансову поведінку, оскільки студент бачить наслідки власних рішень в числовому та графічному вигляді.

Порівняння традиційного та цифрового способу ведення студентського бюджету дає змогу визначити переваги програмної реалізації фінансового обліку. Ручні записи можуть бути корисними на початковому етапі формування фінансової дисципліни, однак вони мають обмежену наочність та потребують додаткового часу для підрахунків. Цифровий формат забезпечує автоматизацію підсумків, швидке оновлення даних і зручне відображення результатів [3]. Порівняння традиційного та цифрового ведення студентського бюджету наведено в таблиці 1.3.

Таблиця 1.3 – Порівняння традиційного та цифрового ведення студентського бюджету

Критерій порівняння	Традиційне ведення бюджету	Цифрове ведення бюджету
Швидкість внесення операцій	Запис потребує ручного оформлення та подальших підрахунків	Операція додається через коротку форму, підсумки оновлюються автоматично
Зручність аналізу	Для оцінки витрат потрібно самостійно групувати записи	Категорії та показники формуються засобами застосунку
Доступність	Записи залежать від наявності блокнота, файлу або окремого документа	Дані доступні через браузер на пристрої користувача
Наочність результатів	Переважає текстова або таблична форма без автоматичної візуалізації	Можливе використання карток, графіків і підсумкових індикаторів
Стійкість фінансової дисципліни	Регулярність залежить від особистої мотивації користувача	Інтерфейс може стимулювати постійне оновлення фінансових записів

Перевага цифрового ведення бюджету полягає в тому, що студент отримує не лише журнал операцій, але й інструмент для швидкої оцінки власного фінансового стану. Після внесення доходу або витрати користувач одразу бачить зміну залишку, рівень витрат і структуру платежів [4].

Управління студентським бюджетом в цифровому середовищі повинно поєднувати облік операцій, планування витрат та наочний аналіз фінансових результатів. Ефективність програмного продукту залежить від здатності швидко фіксувати фінансові дії, зберігати інформацію, відображати актуальний залишок та допомагати користувачу бачити напрями найбільшого фінансового навантаження.

1.2 Чинники формування фінансової поведінки здобувачів освіти

Студентський період пов’язаний з переходом до більш самостійного прийняття рішень, тому молодь поступово набуває досвіду розподілу власних коштів, визначення пріоритетів витрачання, реагування на нестачу ресурсу та планування майбутніх платежів. Рівень організованості особистого бюджету

залежить не лише від обсягу доходів, а й від здатності користувача оцінювати фінансові наслідки щоденних рішень [5].

Студент може отримувати стипендію, матеріальну підтримку родини, оплату за часткову зайнятість або випадкові доходи від тимчасових робіт. Кожне джерело має власну періодичність, стабільність та прогнозованість [6]. При наявності регулярних надходжень легше планувати витрати на місяць, тоді як нерегулярний дохід посилює потребу в резервуванні коштів та контролі щоденних платежів [7]. Економічні чинники фінансової поведінки студентів узагальнено в таблиці 1.4

Таблиця 1.4 – Економічні чинники фінансової поведінки студентів

Чинник	**Зміст впливу**
Рівень доходу	Визначає обсяг коштів, доступних для навчальних, побутових і особистих потреб
Регулярність надходжень	Впливає на можливість планування витрат за тиждень або місяць
Обов’язкові платежі	Охоплюють оплату проживання, зв’язку, транспорту, навчальних матеріалів
Непередбачувані витрати	Виникають через побутові потреби, медичні послуги, поїздки або навчальні обставини
Наявність заощаджень	Зменшує фінансову напруженість і підвищує стійкість бюджету

Економічні чинники визначають базові можливості студента щодо споживання та накопичення. Обмежений дохід не завжди означає низьку якість фінансового управління, оскільки важливим є співвідношення між сумою надходжень, обов’язковими витратами та особистими пріоритетами [8].

Соціальне середовище має значний вплив на фінансові рішення студентів. Навчальна група, друзі, родина, умови проживання та рівень фінансової культури оточення формують уявлення про допустимий рівень витрат. Частина студентів витрачає кошти під впливом групових активностей, спільного дозвілля або бажання підтримувати певний стиль життя [9].

Психологічні чинники пов’язані з рівнем самоконтролю, здатністю відкладати миттєве задоволення, ставленням до накопичення та сприйняттям фінансового ризику. Для багатьох студентів характерна орієнтація на короткострокові потреби, оскільки навчальний період супроводжується активним соціальним життям та змінним розкладом. Відсутність попереднього досвіду самостійного бюджетування може спричинити імпульсивні покупки та недооцінку сукупного впливу дрібних платежів [10]. Психологічні чинники фінансової поведінки здобувачів освіти наведено в таблиці 1.5.

Таблиця 1.5 – Психологічні чинники фінансової поведінки здобувачів освіти

Чинник	**Прояв у поведінці користувача**
Самоконтроль	Здатність обмежувати витрати відповідно до доступного бюджету
Імпульсивність	Схильність до незапланованих покупок
Фінансова тривожність	Відчуття невпевненості через нестачу коштів
Мотивація до заощадження	Бажання накопичити кошти на навчальні або особисті потреби
Звичка до обліку	Регулярність внесення доходів і витрат

Взаємодія всіх груп чинників визначає вимоги до майбутнього програмного продукту. Для студентів важливо не лише записувати доходи та витрати, але й отримувати зрозумілу підтримку при прийнятті фінансових рішень. Застосунок повинен допомагати бачити поточний стан бюджету, виділяти пріоритетні напрями витрачання, контролювати необов’язкові платежі та поступово формувати фінансову дисципліну.

1.3 Мобільні рішення для контролю доходів і витрат

Ринок цифрових інструментів для контролю особистих фінансів представлений значною кількістю мобільних сервісів, які орієнтовані на облік доходів, фіксацію витрат, бюджетне планування, аналітику фінансових операцій та формування звичок ощадливого використання коштів. Для

студентської аудиторії, інтерес становлять рішення, які мають простий інтерфейс, швидке внесення операцій, зрозумілий розподіл платежів за категоріями та можливість перегляду підсумкових показників без складних налаштувань. Аналіз наявних застосунків дає змогу визначити функціональні можливості, які варто врахувати під час розробки мобільного додатку для оптимізації особистих фінансів студентів.

Серед поширених сервісів персонального фінансового обліку можна виділити Money Lover, Wallet by BudgetBakers, Monefy та Spendee. Вибір для аналізу зумовлений орієнтацією даних рішень на облік витрат, бюджетування, візуальне представлення фінансових результатів та підтримку мобільного користування. Кожен сервіс має власну логіку взаємодії з користувачем, різний рівень деталізації фінансової інформації та відмінний підхід до представлення статистики.

Money Lover є сервісом для керування особистими фінансами, який поєднує облік доходів, витрат, бюджетів і фінансових звітів. Його функціональність орієнтована на користувачів, які хочуть контролювати повсякденні платежі, планувати майбутні витрати та аналізувати структуру бюджету [11]. Перевагою є наявність категорій, підсумкових показників та можливості швидкого перегляду фінансового стану. Недоліком є надлишкова кількість можливостей, яка може викликати труднощі при роботі. Головна сторінка сервісу Money Lover наведена на рисунку 1.1.

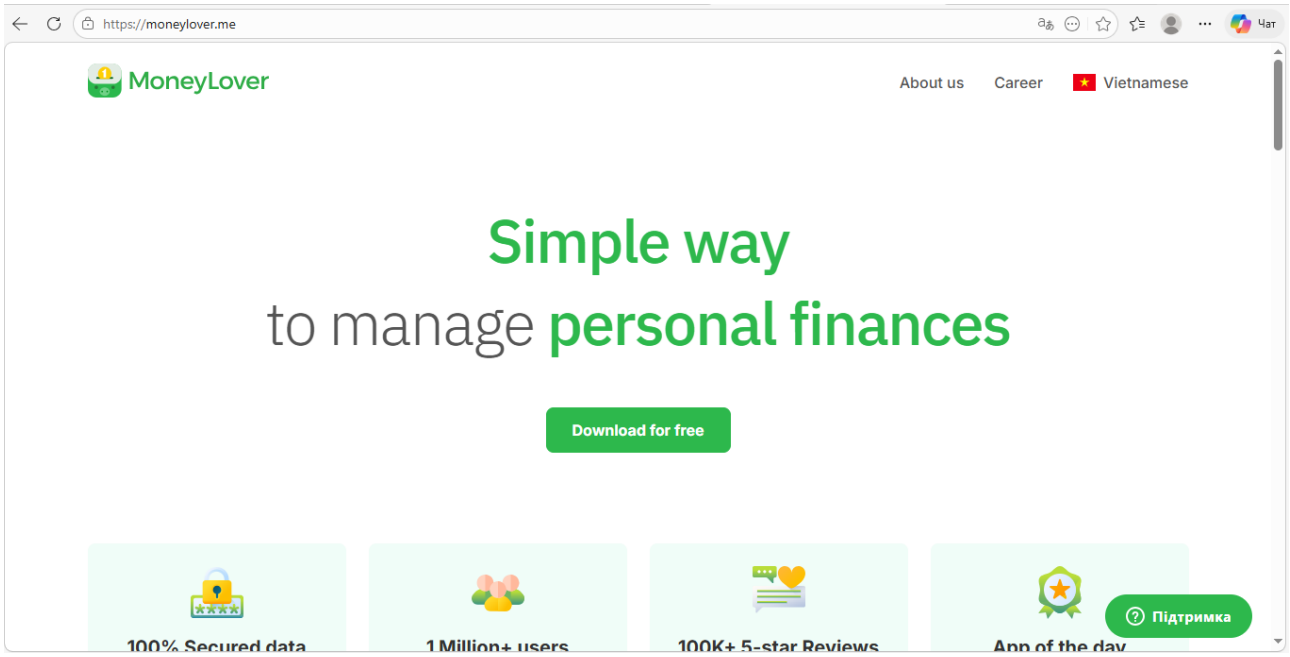

Рисунок 1.1 – Головна сторінка сервісу Money Lover

Wallet by BudgetBakers орієнтований на комплексне управління фінансами користувача. Сервіс підтримує побудову бюджетів, перегляд звітів, синхронізацію з банківськими рахунками та автоматичне групування операцій. Його перевагою є розвинена аналітика та можливість роботи з фінансовою інформацією на різних пристроях [12]. Для студентської аудиторії корисними можуть бути бюджетні ліміти, огляд витрат та структуроване представлення фінансових операцій. Водночас глибока інтеграція з банківськими сервісами та розширена функціональність не завжди є необхідною для студента, оскільки головним завданням виступає доступний облік власних коштів. Головна сторінка сервісу Wallet by BudgetBakers наведена на рисунку 1.2.

Monefy позиціонується як простий інструмент для відстеження витрат та контролю бюджету. Його концепція ґрунтується на швидкому додаванні операцій, що є важливим для користувачів, які не бажають витрачати багато часу на заповнення форм. Для студентів перевагою є зрозуміла логіка внесення платежів та візуальне представлення витрат за категоріями [13]. Головна сторінка сервісу Monefy наведена на рисунку 1.3.

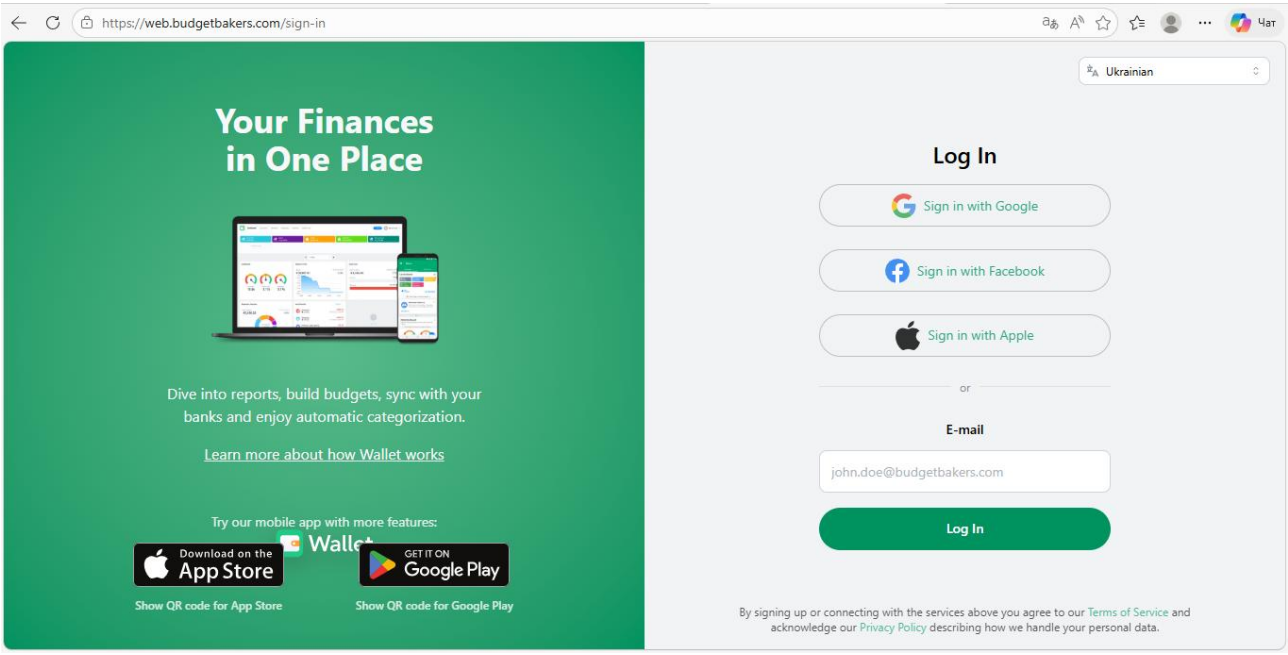

Рисунок 1.2 – Головна сторінка сервісу Wallet by BudgetBakers

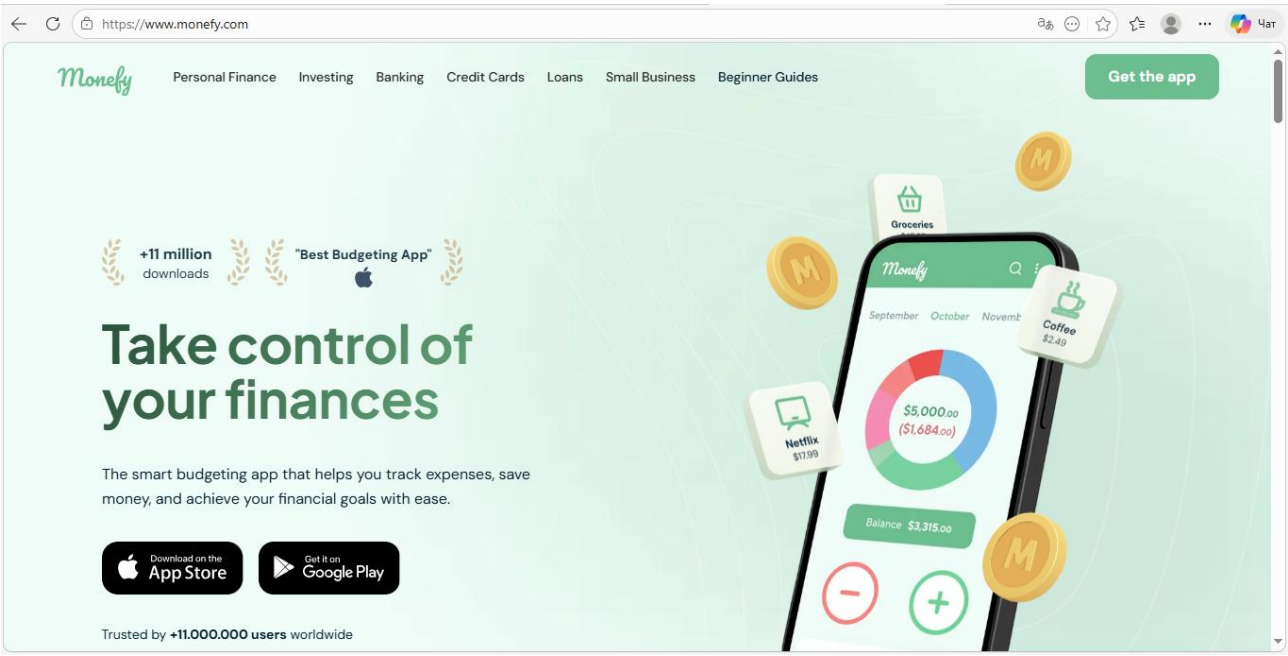

Рисунок 1.3 – Головна сторінка сервісу Monefy

Spendee є сервісом для керування доходами, витратами, бюджетами та фінансовими звичками. Його інтерфейс орієнтований на наочне представлення фінансової інформації, аналіз поведінки користувача та контроль виконання бюджетних обмежень [14]. Головна сторінка сервісу Spendee наведена на рисунку 1.4.

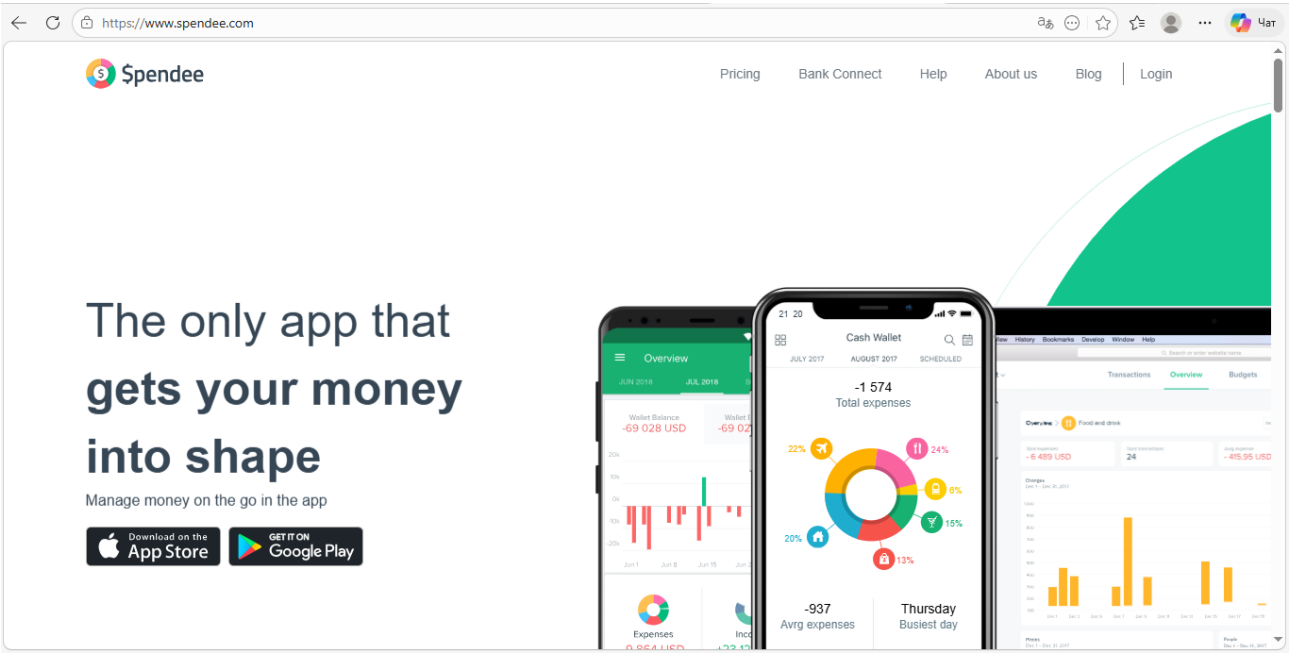

Рисунок 1.4 – Головна сторінка сервісу Spendee

Проведений огляд засвідчує, що більшість сучасних фінансових сервісів поєднує облік операцій, категоризацію платежів, бюджетування та візуальну аналітику [15]. Відмінності полягають в глибині функціональності, складності інтерфейсу та рівні орієнтації на щоденне використання [16]. Для студентів найбільш важливими є не максимальна кількість можливостей, а швидкість роботи, зрозуміла структура, відсутність зайвих етапів внесення операції та наочне відображення залишку коштів [17]. Порівняльна характеристика мобільних рішень для контролю доходів і витрат наведена в таблиці 1.6.

Таблиця 1.6 – Порівняльна характеристика мобільних рішень для контролю доходів і витрат

Критерій порівняння	**Money Lover**	**Wallet by BudgetBakers**	**Monefy**	**Spendee**
Основне призначення	Облік витрат, планування бюджету та перегляд фінансових звітів	Комплексне керування фінансами з аналітикою та синхронізацією	Швидке внесення витрат і контроль бюджету	Аналіз фінансових звичок і бюджетне планування

Критерій порівняння	**Money Lover**	**Wallet by BudgetBakers**	**Monefy**	**Spendee**
Зручність для студентів	Висока завдяки категоріям і звітам	Середня через широку функціональність	Висока завдяки простому введенню операцій	Висока завдяки наочній аналітиці
Облік доходів і витрат	Передбачений	Передбачений	Передбачений	Передбачений
Категоризація платежів	Наявна	Наявна з автоматизованими можливостями	Наявна	Наявна
Бюджетні обмеження	Підтримуються	Підтримуються	Підтримуються в базовій логіці контролю	Підтримуються
Візуальна аналітика	Наявна	Розвинена	Спрощена	Розвинена
Робота через браузер	Передбачена вебверсія	Передбачена вебверсія	Основна орієнтація на мобільний застосунок	Передбачений вебдоступ
Рівень складності інтерфейсу	Середній	Вищий через широку функціональність	Низький	Середній
Придатність ідей для майбутньої розробки	Швидкий огляд бюджету та категорії витрат	Підсумкова аналітика й бюджетні ліміти	Просте додавання операцій	Графічне подання фінансових результатів

Аналіз мобільних рішень для контролю доходів та витрат показав, що ефективний фінансовий застосунок повинен поєднувати простий облік операцій, категоризацію платежів, бюджетне планування та зрозумілу аналітику [18]. Для студентської аудиторії найбільш важливими є швидке додавання доходу або витрати, наочне відображення залишку бюджету,

адаптація інтерфейсу до мобільного екрана та відсутність складних налаштувань.

Висновки до розділу 1

В першому розділі кваліфікаційної роботи розглянуто теоретичні аспекти оптимізації особистих фінансів студентської молоді. Студентський бюджет має специфічні ознаки, пов’язані з обмеженим обсягом доходів, нерегулярністю надходжень, наявністю обов’язкових побутових витрат та потребою в постійному контролі залишку коштів. Для здобувачів освіти особливе значення має не складна фінансова аналітика, а простий та доступний інструмент щоденного обліку доходів та витрат.

На прийняття фінансових рішень впливають економічні умови, соціальне оточення, психологічні особливості користувача та цифрові звички. Обмежені доходи, вплив студентського середовища, імпульсивність покупок та постійне використання смартфона створюють потребу в мобільному рішенні, яке забезпечує швидке внесення операцій, зрозумілий розподіл витрат за категоріями та оперативне відображення фінансового стану.

Порівняльна характеристика сервісів фінансового планування дала змогу визначити функціональні орієнтири для майбутньої розробки. До найбільш значущих належать облік доходів і витрат, категоризація платежів, контроль бюджетного ліміту, відображення залишку коштів та базова візуальна аналітика. Мобільний додаток для оптимізації особистих фінансів студентів має поєднувати доступність, адаптацію до невеликого екрану, локальне збереження інформації та зрозумілу логіку взаємодії з користувачем.

РОЗДІЛ 2. ПРОЄКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ОПТИМІЗАЦІЇ ОСОБИСТИХ ФІНАНСІВ СТУДЕНТІВ

2.1 Формування вимог до функціональних можливостей програмного продукту

Формування вимог до програмного продукту є початковим етапом проєктування мобільного додатку для оптимізації особистих фінансів студентів. На даному етапі визначається коло завдань, які має виконувати застосунок, уточнюються потреби майбутнього користувача, описуються інформаційні потоки та встановлюються обмеження щодо роботи інтерфейсу.

Майбутній програмний продукт розглядається як мобільно орієнтований вебзастосунок, який відкривається в браузері та має інтерфейс, адаптований до екрану смартфона. Зазначена форма реалізації дає змогу поєднати зручність мобільного користування з доступністю браузерного запуску. Користувач не повинен встановлювати окремий інсталяційний пакет або налаштовувати складне серверне середовище. Головна увага приділяється швидкому виконанню базових фінансових дій, а саме внесенню доходів, додаванню витрат, перегляду залишку бюджету та аналізу структури платежів.

Під час формування вимог потрібно враховувати реальні сценарії студентського користування. Застосунок має бути корисним в повсякденних умовах, коли користувач швидко фіксує оплату харчування, транспорту, навчальних матеріалів або побутових потреб. Надмірна кількість полів у формах введення може знизити регулярність користування, тому основні операції мають виконуватись через просту та зрозумілу послідовність дій. Спрощення не повинно позбавляти програмний продукт можливості надавати підсумкову фінансову інформацію [19]. Потреби студентської аудиторії та вимоги до програмного продукту наведено в таблиці 2.1.

Таблиця 2.1 – Потреби студентської аудиторії та вимоги до програмного продукту

Потреба користувача	**Практичний зміст потреби**	**Вимога до реалізації**
Швидке внесення витрат	Студент фіксує платіж одразу після покупки або оплати послуги	Форма витрати має містити суму, категорію, дату та короткий опис
Контроль залишку коштів	Користувач хоче бачити суму, доступну після виконаних операцій	Головний екран має відображати доходи, витрати та поточний баланс
Поділ платежів за напрямами	Витрати пов’язані з харчуванням, транспортом, навчанням, житлом і дозвіллям	Потрібна система категорій з можливістю групування операцій
Обмеження витрат	Студенту важливо не перевищити заплановану суму на певний період	Має бути передбачений бюджетний ліміт і показник його використання
Перегляд фінансової динаміки	Користувач оцінює, куди витрачається найбільше коштів	Потрібна сторінка статистики з графічним або табличним поданням
Простий доступ	Застосунок використовується на смартфоні або комп’ютері без інсталяції	Інтерфейс має коректно відкриватися у браузері на різних екранах

Функціональні вимоги визначають набір дій, які має виконувати програмний продукт. Для мобільного додатку фінансового планування базовими є операції з додавання, перегляду, редагування та видалення фінансових записів. Кожен запис повинен містити тип операції, суму, категорію, дату та необов’язковий коментар. За рахунок структурування даних, користувач зможе не лише фіксувати факти платежів, але й отримувати узагальнене уявлення про власний бюджет.

Окремою функціональною вимогою є розрахунок фінансових підсумків. Після внесення операції застосунок має автоматично оновлювати суму доходів, суму витрат та залишок коштів. Відображення результату на головному екрані

зменшує потребу в ручних підрахунках та допомагає користувачу швидко приймати рішення щодо подальших витрат [19]. Функціональні вимоги до мобільного додатку наведено в таблиці 2.2.

Таблиця 2.2 – Функціональні вимоги до мобільного додатку

Функціональна вимога	Опис реалізації	Очікуваний результат
Додавання доходу	Користувач вводить суму, джерело надходження та дату	В бюджеті збільшується загальна сума доходів
Додавання витрати	Користувач зазначає суму, категорію, дату й короткий опис платежу	В бюджеті оновлюється сума витрат і поточний баланс
Перегляд операцій	Застосунок відображає журнал фінансових записів	Користувач бачить історію доходів та витрат
Редагування запису	Користувач може змінити суму, категорію або опис операції	Помилково внесені дані можна виправити без очищення журналу
Видалення запису	Користувач прибирає операцію з історії	Підсумкові показники перераховуються автоматично
Категоризація витрат	Платежі розподіляються за основними напрямами	Структура витрат стає зрозумілою для аналізу
Встановлення бюджету	Користувач задає допустиму суму витрат на період	Застосунок показує рівень використання ліміту
Формування статистики	Дані групуються за категоріями та періодами	Користувач отримує наочну оцінку фінансової поведінки

Важливим елементом вимог є визначення інформаційної структури застосунку. Для роботи програмного продукту необхідно зберігати записи про доходи, витрати, категорії, бюджетний ліміт та підсумкові значення. Оскільки застосунок орієнтований на браузерний запуск, можна використати локальне збереження даних в середовищі браузера.

Інформаційні вимоги мають забезпечувати цілісність фінансових розрахунків. Сума операції повинна бути числовим значенням, тип операції має відповідати доходу або витраті, а категорія повинна допомагати групувати

записи. Дата операції потрібна для відображення історії та формування статистики за періодами. Коментар до операції може залишатись необов'язковим, оскільки не кожен платіж потребує детального пояснення [19]. Інформаційні об'єкти програмного продукту представлено в таблиці 2.3.

Таблиця 2.3 – Інформаційні об'єкти програмного продукту

Інформаційний об'єкт	Призначення	Основні дані
Фінансова операція	Збереження факту доходу або витрати	Тип, сума, категорія, дата, опис
Категорія	Групування платежів за напрямами	Назва категорії, вид операції
Бюджетний ліміт	Контроль максимально допустимої суми витрат	Сума ліміту, період дії
Підсумковий баланс	Відображення поточного фінансового стану	Доходи, витрати, залишок
Статистичний показник	Узагальнення фінансових даних	Частка категорії, сума за період, рівень використання бюджету
Налаштування інтерфейсу	Збереження базових параметрів користування	Обрана валюта, формат відображення, режим перегляду

Окрім функціональних вимог, потрібно визначити вимоги до зручності використання. Для студентської аудиторії інтерфейс має бути лаконічним, адаптивним та зрозумілим без попереднього навчання. Головний екран повинен одразу показувати ключові показники, а навігація має забезпечувати швидкий перехід до журналу операцій, форми додавання запису, бюджетного блоку та статистики. Нижня навігаційна панель є доречною для мобільного формату, оскільки вона відповідає поширеній логіці користування смартфоном.

Вимоги до візуального оформлення мають підтримувати сприйняття фінансової інформації. Доходи та витрати варто розділяти на рівні карток, підписів і підсумкових блоків. Статистичний екран повинен містити графічне або табличне відображення структури витрат, щоб користувач міг швидко

побачити категорії з найбільшим фінансовим навантаженням. Нефункціональні вимоги до мобільно-орієнтованого вебзастосунку наведено в таблиці 2.4.

Таблиця 2.4 – Нефункціональні вимоги до мобільно орієнтованого вебзастосунку

Група вимог	Зміст вимоги	Практичне значення
Адаптивність	Інтерфейс коректно відображається на смартфоні, планшеті та комп'ютері	Забезпечується зручність доступу з різних пристроїв
Продуктивність	Основні екрани відкриваються без помітних затримок	Користувач може швидко внести операцію
Зрозумілість	Елементи інтерфейсу мають очевидне призначення	Зменшується потреба в інструкціях
Надійність розрахунків	Підсумкові значення оновлюються після кожної зміни даних	Підвищується довіра до фінансової інформації
Локальне збереження	Дані залишаються доступними після перезавантаження сторінки	Забезпечується безперервність роботи з бюджетом
Безпека демонстраційних даних	Застосунок не передає інформацію на сторонні сервери	Особисті записи залишаються в середовищі браузера
Доступність інтерфейсу	Текст, кнопки й форми мають бути зручними для перегляду	Полегшується робота на малому екрані

Визначені вимоги формують основу для подальшого проєктування мобільного додатку. Функціональна логіка має забезпечити повний цикл роботи з особистими фінансами студента: внесення доходу, фіксацію витрати, перегляд історії, контроль бюджетного ліміту та аналіз фінансових результатів.

2.2 Обґрунтування вибору кросплатформного середовища розробки

Вибір технологічного стеку для мобільного додатку по оптимізації особистих фінансів студентів має враховувати функціональні вимоги, визначені на попередньому етапі проєктування, особливості студентської аудиторії та умови демонстрації програмного продукту.

Для розробки програмного продукту обрано стек React, Vite, CSS і Web Storage API. Вибір зазначених технологій пов’язаний з потребою створити легкий, зрозумілий та зручний в тестуванні застосунок без складної серверної частини. React забезпечує побудову інтерфейсу на основі окремих компонентів, що спрощує створення екранів фінансового обліку [20]. Vite використовується для швидкого запуску середовища розробки та формування готової збірки. CSS відповідає за адаптивне оформлення, а Web Storage API дає змогу зберігати дані користувача в браузері між сеансами роботи. Загальна характеристика обраного технологічного стеку наведена в таблиці 2.5.

Таблиця 2.5 – Загальна характеристика обраного технологічного стеку

Технологія	**Призначення в програмному продукті**
React	Побудова екранів і компонентів користувацького інтерфейсу
Vite	Організація середовища розроблення та збірки
CSS	Оформлення сторінок і адаптація до різних розмірів екрана
Web Storage API	Локальне збереження фінансових записів
JavaScript	Реалізація логіки обробки даних
HTML	Структурування вмісту сторінок

React є доречним вибором для реалізації інтерфейсу, оскільки фінансовий застосунок складається з багатьох повторюваних елементів. До них належать картки доходів і витрат, форма додавання операції, список транзакцій, блок статистики, бюджетний індикатор та навігаційна панель.

Важливою перевагою React є можливість динамічного оновлення інтерфейсу після зміни даних. Коли користувач додає витрату або дохід, підсумкові показники мають змінюватись без ручного перезавантаження сторінки. Завдяки реактивній логіці роботи, інтерфейс може одразу відображати оновлений баланс, суму витрат, рівень використання бюджету та структуру платежів [20].

Vite обрано як інструмент створення та запуску фронтенд-проєкту. Його використання спрощує організацію структури застосунку, підключення React та

перевірку результатів в браузері. Vite формує готову збірку, яку можна розмістити як статичний вебзастосунок [21].

CSS використовується для формування візуального стилю та адаптивності мобільного інтерфейсу. Головний екран повинен містити помітні підсумкові картки, кнопки додавання доходів і витрат, а також, навігацію між основними розділами. За допомогою CSS, можна налаштувати розташування блоків, розміри елементів, відступи, шрифти, стан кнопок та поведінку інтерфейсу на різних екранах [22].

Мобільна адаптація є ключовою умовою реалізації, оскільки застосунок орієнтований на роботу зі смартфона. Інтерфейс має бути зручним для вертикального перегляду, а основні дії повинні виконуватись без потреби збільшувати масштаб сторінки. Нижня навігаційна панель, великі кнопки, компактні картки та лаконічні форми відповідають типовій логіці мобільного користування.

Web Storage API є важливим засобом збереження даних, оскільки фінансова інформація зберігається локально в браузері. Для програмного продукту, який не потребує реєстрації та серверної синхронізації, локальне сховище дає змогу забезпечити збереження доходів, витрат, категорій та бюджетних обмежень після перезавантаження сторінки. Користувач може закрити браузер, а під час наступного відкриття побачити раніше внесені записи [23].

Локальне збереження спрощує архітектуру, зменшує кількість залежностей і не потребує розроблення окремого серверного модуля. Обмеженням вказаного варіанту є прив’язка даних до конкретного браузера та пристрою, однак для навчального продукту, орієнтованого на показ основної функціональності, вказане обмеження є прийнятним.

Браузерний формат з мобільним інтерфейсом можна розглядати як основу для PWA-логіки. PWA дає змогу створити застосунок на вебтехнологіях з користувацьким досвідом, наближеним до мобільного програмного продукту.

Порівняно з нативною мобільною розробкою, веборієнтований стек є раціональним для поставленого завдання. Нативні технології потребують окремих інструментів для Android або iOS, складнішого тестування та інколи додаткової підготовки середовища. Браузерний запуск дозволяє показати роботу застосунку на різних пристроях і спрощує передачу проєкту для перевірки.

Обраний стек відповідає функціональним та нефункціональним вимогам, сформованим для мобільного додатку з оптимізації особистих фінансів студентів. React забезпечує гнучку побудову інтерфейсу, Vite спрощує запуск і збірку, CSS створює адаптивний мобільний вигляд [24], а Web Storage API підтримує локальне збереження фінансових даних. Разом вони формують технологічну основу, достатню для реалізації обліку доходів і витрат, бюджетного контролю, журналу операцій і візуальної статистики.

2.3 Архітектура мобільного додатку для персонального фінансового обліку

Для застосунку, орієнтованого на студентську аудиторію, архітектура має бути достатньо простою для реалізації та водночас придатною для підтримки основних функцій бюджетного контролю. До таких функцій належать додавання доходів та витрат, збереження операцій, розрахунок балансу, групування платежів за категоріями та відображення статистичних показників.

Обраний технологічний стек передбачає побудову клієнтської архітектури без окремої серверної частини. Основна логіка застосунку виконується безпосередньо в браузері користувача, а фінансова інформація зберігається за допомогою локального сховища.

Архітектура мобільного додатку формується на основі кількох взаємопов’язаних рівнів. Перший рівень відповідає за відображення інтерфейсу та взаємодію користувача з екранами. Другий рівень охоплює логіку обробки фінансових операцій, розрахунок підсумкових значень та фільтрацію записів.

Третій рівень забезпечує збереження інформації в браузері та відновлення записів після повторного відкриття застосунку [25]. Рівні архітектури мобільного додатку узагальнено в таблиці 2.6.

Таблиця 2.6 – Рівні архітектури мобільного додатку

Рівень архітектури	**Призначення**	**Основні елементи**
Рівень інтерфейсу	Забезпечує відображення екранів і приймання дій користувача	Головний екран, форма операції, журнал записів, статистика, навігація
Рівень логіки	Виконує обробку фінансових операцій і розрахунок показників	Функції додавання, редагування, видалення, обчислення балансу, групування витрат
Рівень збереження	Забезпечує запис і відновлення фінансової інформації	Web Storage API, локальні записи операцій, бюджетний ліміт, налаштування
Рівень візуалізації	Формує наочне представлення підсумкових значень	Картки балансу, індикатор бюджету, діаграма витрат, статистичні блоки

Інтерфейсний рівень має найбільше значення для користувача, оскільки саме через нього студент виконує фінансові дії. Головний екран повинен відображати поточний баланс, суму доходів, суму витрат та рівень використання бюджету. Форма операції використовується для внесення доходу або витрати, а журнал записів дає змогу переглядати історію фінансових дій. Екран статистики повинен надавати узагальнену інформацію про структуру витрат, щоб користувач міг визначити напрями найбільшого фінансового навантаження.

Рівень логіки відповідає за правильність розрахунків. Після додавання нової операції застосунок повинен оновити масив фінансових записів, перерахувати суму доходів, суму витрат і залишок коштів. Якщо користувач редагує або видаляє запис, підсумкові показники мають змінюватись

автоматично. Логічний рівень виконує групування витрат за категоріями, що потрібне для побудови статистичних блоків та діаграм [25].

Рівень збереження реалізується через Web Storage API. Оскільки застосунок не має серверної частини, всі записи зберігаються в браузері користувача. Під час запуску програмний продукт зчитує раніше збережену інформацію та відображає актуальний фінансовий стан. Після будь-якої зміни записів, оновлена інформація знову передається до локального сховища. Завдяки цьому, користувач може продовжити роботу після закриття вкладки або перезавантаження сторінки. Компонентна архітектура мобільного додатку для персонального фінансового обліку наведена на рисунку 2.1.

Компонентна схема демонструє, що користувач взаємодіє з програмним продуктом через навігаційну панель та основні екрани. Кожен екран звертається до логіки розрахунків, яка виконує обробку фінансових записів і передає оновлену інформацію до інтерфейсу. Керування сховищем виконує роль посередника між логічним рівнем та локальним сховищем браузера [25].

Архітектура програмного продукту передбачає наявність кількох основних модулів. Модуль головного екрана відповідає за відображення поточного фінансового стану. Модуль операцій забезпечує внесення доходів і витрат. Модуль журналу дає змогу переглядати історію записів. Модуль статистики групує інформацію та формує візуальне представлення витрат. Модуль сховища відповідає за запис інформації в браузер і відновлення даних під час запуску. Модулі програмного продукту та їх призначення узагальнено в таблиці 2.7.

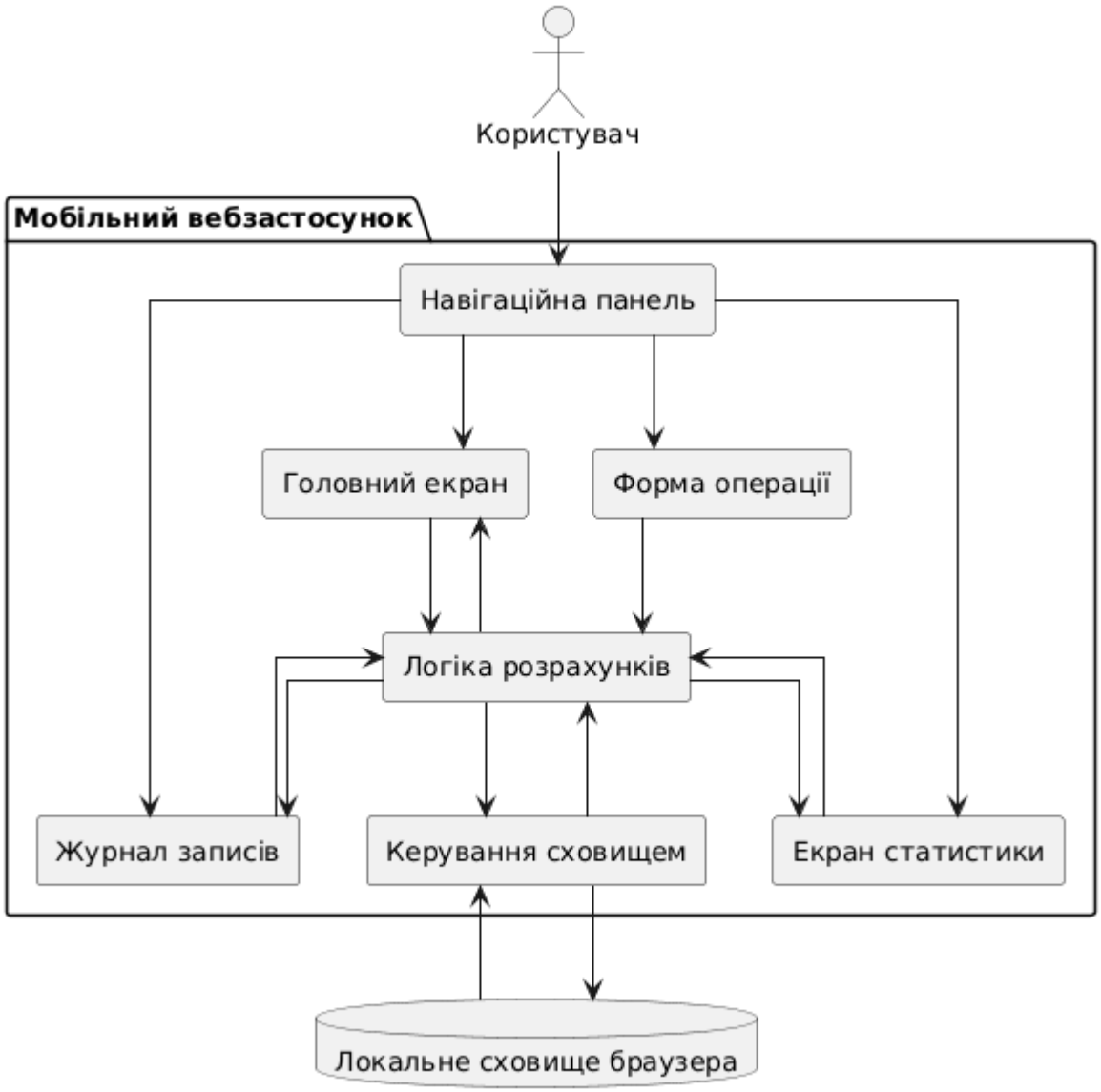

Рисунок 2.1 – Компонентна архітектура мобільного додатку для персонального фінансового обліку

Таблиця 2.7 – Модулі програмного продукту та їх призначення

Модуль	**Призначення**	**Очікуваний результат роботи**
Модуль головного екрана	Відображає баланс, доходи, витрати та стан бюджету	Користувач бачить узагальнену картину особистих фінансів
Модуль фінансових операцій	Забезпечує додавання, редагування й видалення записів	Журнал доходів і витрат залишається актуальним
Модуль категорій	Групує платежі за напрямами студентських витрат	Користувач розуміє структуру використання коштів

Модуль	Призначення	Очікуваний результат роботи
Модуль бюджетного ліміту	Зберігає планову суму витрат і контролює її використання	Студент бачить рівень наближення до встановленого обмеження
Модуль статистики	Формує підсумки за категоріями та періодами	Фінансова поведінка подається у зрозумілому вигляді
Модуль локального сховища	Зберігає записи в браузері та відновлює їх після запуску	Інформація не втрачається після закриття сторінки
Модуль навігації	Забезпечує перехід між екранами застосунку	Робота з додатком залишається зручною на смартфоні

Інформаційна структура застосунку базується на фінансовій операції як основному об’єкті. Кожна операція містить тип запису, суму, категорію, дату та опис. Тип визначає, чи збільшує операція бюджет, чи зменшує його. Сума використовується для розрахунку балансу. Категорія потрібна для групування платежів, а дата дає змогу впорядкувати журнал та виконувати аналіз за періодами. Опис допомагає уточнити призначення платежу, проте його заповнення не повинно бути обов’язковою умовою [26].

Для забезпечення коректної роботи застосунку, фінансові записи мають зберігатись в структурованому форматі. Найбільш придатним варіантом для браузерного застосунку є масив об’єктів JavaScript, який перетворюється на текстовий формат перед передаванням до локального сховища. Після повторного запуску застосунку інформація зчитується, перетворюється назад в масив об’єктів та використовується для побудови екранів.

Категорії витрат повинні відповідати типовим потребам студентської аудиторії. Необхідно передбачити категорії, пов’язані з харчуванням, транспортом, навчанням, проживанням, зв’язком, здоров’ям та дозвіллям. Даний набір охоплює основні напрями використання коштів і дає змогу формувати зрозумілу статистику. Для доходів можна передбачити категорії, пов’язані зі стипендією, підтримкою родини, підробітком та іншими надходженнями.

Потік інформації в застосунку починається з дії користувача. Студент відкриває форму, вводить фінансову операцію та підтверджує збереження.

Після перевірки введених значень, запис додається до загального набору операцій. Далі застосунок перераховує підсумкові показники, оновлює головний екран, журнал і статистику. Оновлена інформація передається до локального сховища браузера. При наступному відкритті сторінки, застосунок відновлює попередній стан та продовжує роботу з наявними записами [26]. Потік обробки фінансової операції в мобільному додатку представлено на рисунку 2.2.

Представлена схема відображає послідовність дій від моменту відкриття застосунку до оновлення фінансових показників. Важливим етапом є зчитування записів з локального сховища, оскільки саме воно забезпечує відновлення стану після попереднього сеансу. Після додавання операції, застосунок виконує кілька пов’язаних дій, а саме, оновлює журнал, перераховує баланс, формує статистику та зберігає змінену інформацію. Завдяки цьому всі екрани працюють з актуальними даними.

В майбутньому до програмного продукту можна додати експорт фінансових записів, цілі накопичення, нагадування про регулярні платежі або авторизацію користувача. Поточна структура не перешкоджає подальшому розвитку, оскільки основні модулі вже мають окреме призначення. Розширення функціональності може виконуватись через додавання нових компонентів без суттєвого порушення базової логіки роботи [26].

Запропонована архітектура охоплює інтерфейсний рівень, логіку обробки операцій, локальне збереження інформації та візуальне представлення підсумків. Завдяки клієнтській реалізації програмний продукт може відкриватися в браузері без складного налаштування, а адаптивне оформлення забезпечує зручність використання на смартфоні.

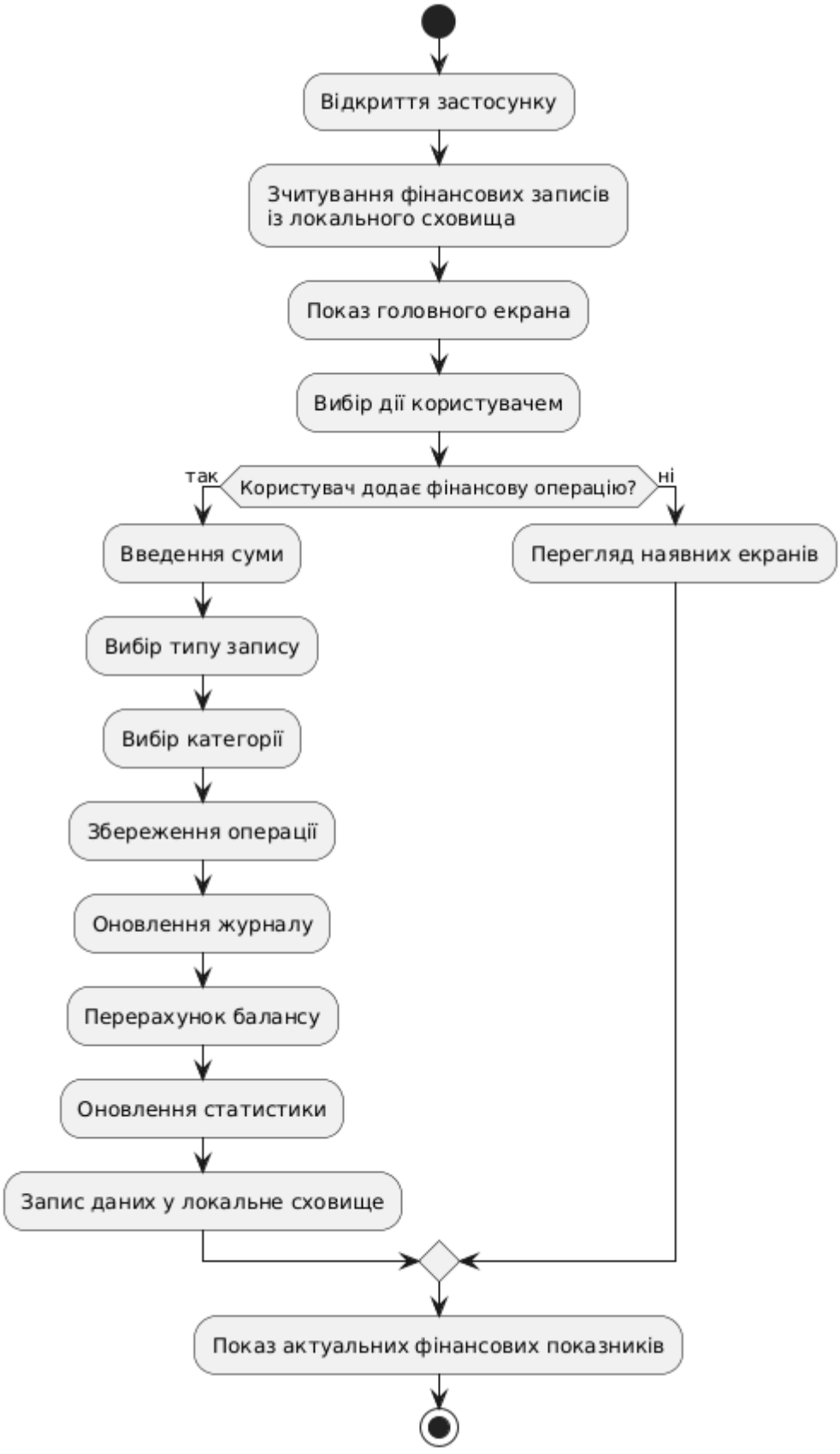

Рисунок 2.2 – Потік обробки фінансової операції в мобільному додатку

2.4 Моделювання взаємодії користувача з основними екранами системи

Мобільний інтерфейс програмного продукту має орієнтуватись на щоденне використання. Користувач відкриває застосунок для перевірки залишку коштів, внесення доходу або витрати, перегляду історії записів, контролю бюджетного ліміту та аналізу структури витрат. Основні екрани повинні бути пов’язані між собою через нижню навігаційну панель, оскільки зазначене розміщення елементів є зручним для роботи зі смартфона. Завдяки цьому, користувач може швидко перейти до потрібної функції без пошуку дій в складному меню. Основні екрани мобільного додатку та їх призначення представлено в таблиці 2.8.

Таблиця 2.8 – Основні екрани мобільного додатку та їх призначення

Екран застосунку	**Призначення**	**Основні дії користувача**
Головний екран	Відображення поточного фінансового стану	Перегляд балансу, доходів, витрат і рівня використання бюджету
Екран операції	Внесення доходу або витрати	Заповнення суми, вибір типу запису, вибір категорії, збереження операції
Журнал фінансів	Перегляд історії доходів і витрат	Аналіз записів, редагування операції, видалення помилкового запису
Екран бюджету	Контроль планової суми витрат	Встановлення ліміту, перегляд використаної частини, оцінка залишку
Екран статистики	Узагальнення фінансових даних	Перегляд категорій витрат, аналіз частки платежів, оцінка фінансової поведінки
Екран налаштувань	Визначення базових параметрів роботи	Вибір валюти, очищення демонстраційних даних, перегляд інформації про застосунок

Головний екран виконує роль початкової точки взаємодії. Після відкриття застосунку, користувач повинен одразу побачити поточний баланс, суму доходів, суму витрат і стан бюджетного ліміту. Інформація подається у вигляді коротких карток, що дає змогу швидко оцінити фінансове становище без

переходу до додаткових сторінок. З головного екрану користувач може перейти до додавання операції, журналу, бюджету або статистики [27].

Екран внесення операції повинен бути побудований за принципом мінімальної кількості полів. Користувач зазначає суму, тип операції, категорію, дату та за потреби короткий опис. Після натискання кнопки збереження, система перевіряє коректність введених значень, додає запис до журналу, оновлює баланс і передає дані до локального сховища браузера. Якщо сума не вказана або має некоректне значення, застосунок повинен показати зрозуміле повідомлення без втрати вже введеної інформації.

Журнал фінансів забезпечує перегляд всіх раніше внесених операцій. Для мобільного формату найзручнішим є представлення записів у вигляді карток, де зазначено суму, тип операції, категорію, дату та короткий опис. Користувач має можливість редагувати або видалити запис, якщо він був внесений помилково. Після будь-якої зміни, підсумкові показники автоматично перераховуються, що забезпечує актуальність фінансової інформації.

Екран бюджету призначений для контролю витрат відносно встановленого ліміту. Користувач задає суму, яку планує витратити протягом обраного періоду, а застосунок відображає використану частину бюджету. За рахунок візуального індикатора, студент може оцінити наближення до обмеження та скоригувати подальші витрати [28].

Екран статистики формує аналітичну інтерпретацію фінансових записів. Його завданням є виявлення категорій з найбільшою часткою витрат та надання загального співвідношення між доходами та витратами. Для спрощення сприйняття інформації можна використовувати діаграму структури платежів, підсумкові картки та таблицю категорій. Аналітичний екран не повинен бути перевантажений великою кількістю показників, оскільки головною метою є швидке розуміння фінансової поведінки. Сценарії взаємодії користувача з застосунком наведено в таблиці 2.9.

Таблиця 2.9 – Сценарії взаємодії користувача з застосунком

Сценарій	**Початкова дія**	**Результат для користувача**
Перевірка фінансового стану	Відкриття головного екрана	Користувач бачить баланс, доходи, витрати й стан бюджету
Додавання витрати	Перехід до форми операції та введення даних	В журналі з’являється новий запис, а баланс зменшується
Додавання доходу	Вибір типу доходу та збереження запису	Загальна сума доходів і залишок коштів збільшуються
Редагування операції	Вибір запису в журналі та зміна значень	Підсумкові показники перераховуються після збереження
Видалення операції	Натискання дії видалення біля запису	Помилковий запис прибирається з журналу
Встановлення бюджету	Введення планової суми витрат	Система показує рівень використання встановленого ліміту
Перегляд статистики	Перехід до аналітичного екрана	Користувач бачить структуру витрат за категоріями

Для формального опису ролі користувача та доступних функцій необхідно використати діаграму варіантів використання. Вона відображає основні дії, які студент може виконувати в системі. Діаграма допомагає узгодити функціональні вимоги з майбутньою реалізацією інтерфейсу та показує, які операції мають бути доступними з боку користувача. Діаграма варіантів використання мобільного додатку наведена на рисунку 2.3.

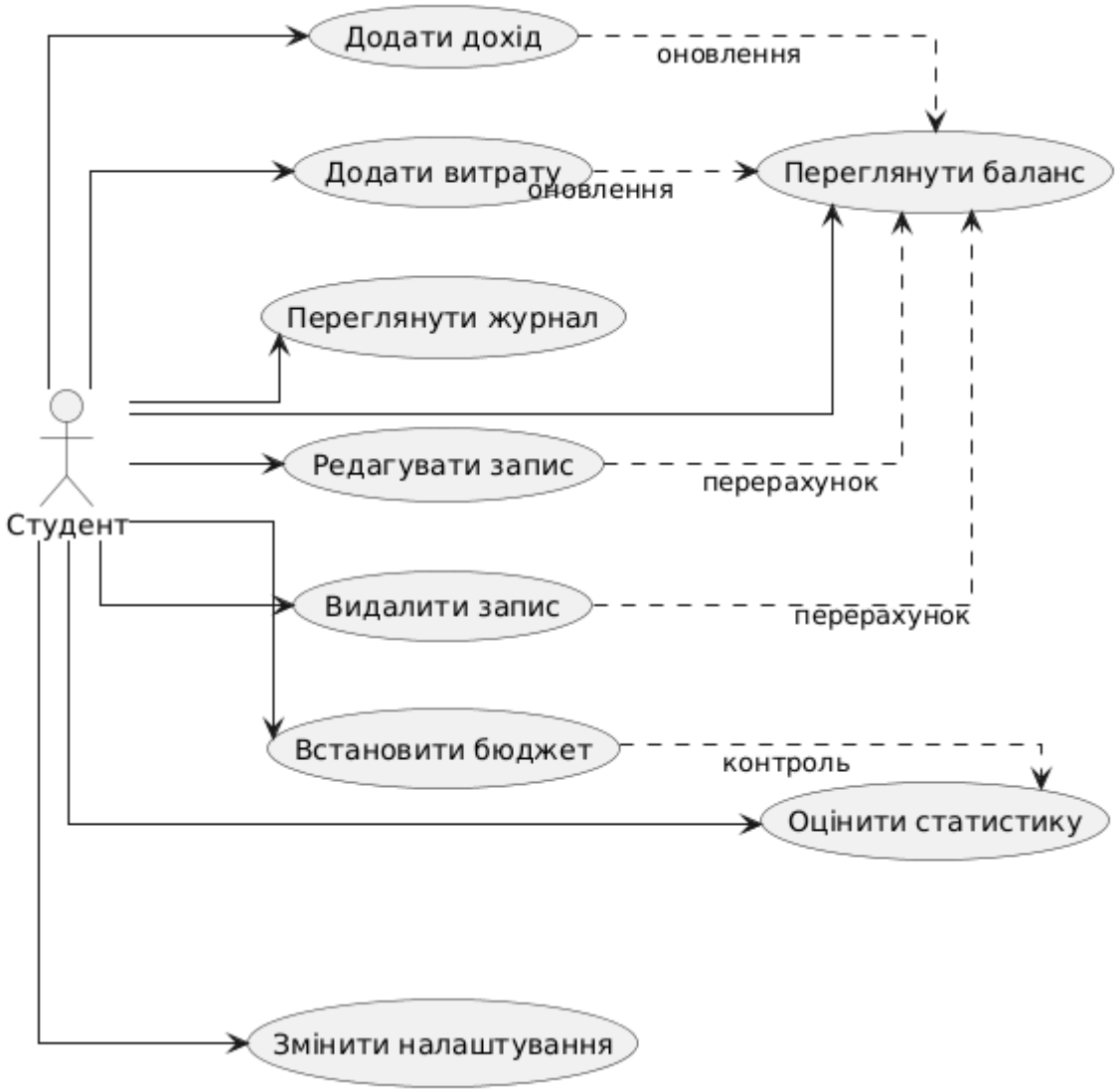

Рисунок 2.3 – Діаграма варіантів використання мобільного додатку

Діаграма варіантів використання показує, що студент є основним користувачем програмного продукту. Він може переглядати фінансовий стан, додавати операції, працювати з журналом, задавати бюджетний ліміт та аналізувати статистику. Зв’язки між варіантами використання відображають залежність підсумкових показників від дій з фінансовими записами. Після додавання, редагування або видалення операції, система повинна оновити баланс та аналітичні дані.

Після завершення дії застосунок повертає користувача до головного екрана або до журналу, якщо операція редагується з історії записів. Побудова інтерфейсу за вказаною логікою забезпечує простоту користування та зменшує ризик втрати орієнтації під час роботи зі смартфона [28].

Особливо важливим сценарієм є додавання фінансової операції. Саме він найчастіше повторюється під час щоденного користування. Для забезпечення зручності потрібно, щоб користувач виконував мінімальну кількість дій: відкрив форму, ввів суму, обрав тип і категорію, після чого зберіг запис. Система перевіряє коректність інформації, додає запис до журналу, оновлює баланс і статистику, а потім зберігає змінений набір даних у браузері. Послідовність додавання фінансової операції представлена на рисунку 2.4

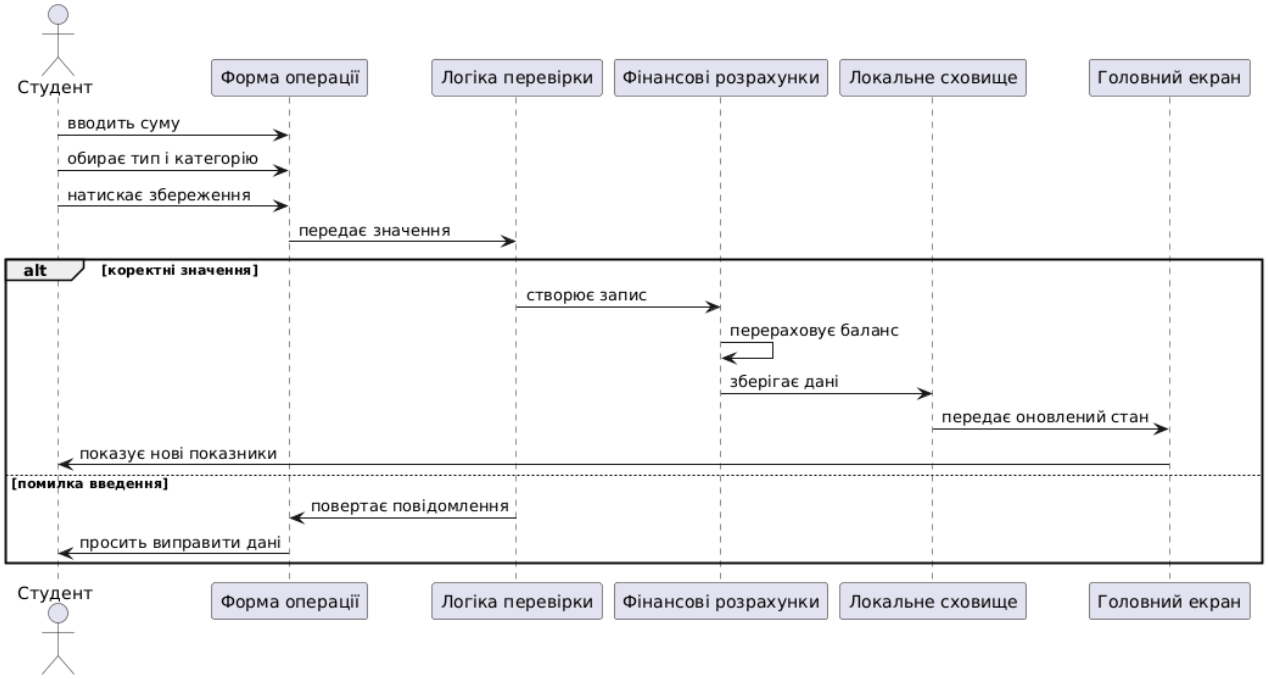

Рисунок 2.5 – Послідовність додавання фінансової операції

Діаграма послідовності показує взаємодію між користувачем, формою, логікою перевірки, розрахунковим блоком, локальним сховищем і головним екраном. У разі коректного введення застосунок створює фінансовий запис та перераховує підсумкові значення. Якщо введено некоректну суму або не обрано обов’язковий параметр, користувач отримує повідомлення про потребу виправлення.

Сценарій перегляду статистики починається з переходу до аналітичного екрану, після чого система групує витрати за категоріями, визначає суму для кожної групи та формує наочне подання результатів. Студент бачить, які

напрями витрат мають найбільшу частку, і може прийняти рішення щодо зміни фінансової поведінки.

Модель взаємодії повинна враховувати не лише успішні сценарії, але й помилки користувача. Типовими проблемами можуть бути порожнє поле суми, введення від’ємного значення, відсутність категорії або спроба зберегти неповний запис. Для таких випадків інтерфейс має надавати короткі повідомлення, які пояснюють причину помилки та не блокують подальшу роботу. Важливо, щоб користувач міг швидко виправити значення без повторного введення всієї операції [28].

Поведінка інтерфейсу повинна залишатись послідовною на всіх екранах. Якщо операція збережена, користувач має отримати оновлені фінансові показники. Якщо дію скасовано, стан журналу не повинен змінюватись. Якщо змінено бюджетний ліміт, екран бюджету та головний екран мають показати нове співвідношення між витратами та плановою сумою. Послідовність реакцій формує довіру до програмного продукту та полегшує його використання без додаткових пояснень.

Важливим результатом моделювання є визначення вимог до розміщення елементів на екранах. Для смартфона потрібно передбачити вертикальну структуру, зверху розміщуються ключові показники, в центральній частині подаються записи або форма, а знизу розташовується навігаційна панель. Кнопки повинні мати достатній розмір для натискання пальцем, а текстові підписи мають бути стислими. Для збереження зручності на комп’ютері ті самі екрани можуть мати обмежену ширину, що імітує мобільний формат.

Моделювання взаємодії користувача з основними екранами системи підтверджує, що майбутній мобільний додаток має спиратись на короткі сценарії, зрозумілу навігацію та автоматичне оновлення фінансових підсумків. Основними екранами виступають головна сторінка, форма операції, журнал фінансів, бюджетний блок, статистика та налаштування. Кожен екран виконує окрему роль, але всі вони працюють з єдиним набором фінансових записів, який зберігається в браузері користувача.

Висновки до розділу 2

В другому розділі кваліфікаційної роботи виконано проєктування мобільного додатку для оптимізації особистих фінансів студентів. На основі результатів теоретичного аналізу сформовано вимоги до функціональних можливостей програмного продукту, зокрема до обліку доходів і витрат, категоризації платежів, контролю бюджетного ліміту, перегляду журналу фінансових операцій та формування статистичних показників. Застосунок має бути орієнтований на швидке виконання повсякденних дій і зручне використання на екрані смартфона.

Обґрунтовано вибір веборієнтованого технологічного стеку для реалізації мобільного інтерфейсу. Для розробки програмного продукту запропоновано використати React, Vite, CSS, JavaScript і Web Storage API. Обрані технології забезпечують компонентну побудову інтерфейсу, швидкий запуск в браузері, адаптивне оформлення екранів та локальне збереження фінансової інформації.

Визначено основні рівні програмного продукту, серед яких інтерфейсний рівень, логіка обробки фінансових операцій, рівень локального збереження та блок візуалізації підсумкових показників. Описано модулі головного екрана, фінансових операцій, категорій, бюджетного ліміту, статистики, локального сховища та навігації. Запропонована архітектура забезпечує взаємозв’язок між діями користувача, обробкою записів та оновленням фінансових результатів.

Результати проєктування створюють основу для програмної реалізації мобільного додатку, здатного забезпечити облік студентського бюджету, оперативне оновлення показників та наочне представлення фінансової інформації.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ СТУДЕНТСЬКОГО ФІНАНСОВОГО ПЛАНУВАННЯ

3.1 Реалізація структури застосунку засобами мобільної розробки

Практична реалізація мобільного додатку для оптимізації особистих фінансів студентів передбачає створення програмного продукту з браузерним запуском та мобільно адаптованим інтерфейсом. Обрана форма реалізації дає змогу забезпечити доступність застосунку для користувача без встановлення окремого інсталяційного пакета. Основна увага приділяється побудові зручної структури інтерфейсу, організації фінансових записів, локальному збереженню інформації та автоматичному оновленню підсумкових показників після кожної операції.

Структура програмного продукту формується за принципом розподілу функціональності між окремими файлами й компонентами. Головний файл застосунку відповідає за зберігання стану, передачу даних між екранами та підключення основних функцій. Окремі компоненти призначені для відображення фінансових карток, форми операції, журналу записів, бюджетного блоку, статистики та навігації. Завдяки компонентній організації кожна частина інтерфейсу має власне призначення, що спрощує подальше редагування програмного коду.

Головний компонент застосунку повинен зберігати основні стани, пов’язані з фінансовими операціями, бюджетним лімітом, активним екраном і налаштуваннями відображення. Під час запуску, застосунок отримує дані з локального сховища браузера. Якщо записи вже існують, вони завантажуються до стану програми та відображаються на відповідних екранах. Якщо збережених даних немає, користувач отримує порожній журнал і може розпочати ведення бюджету з першої операції.

Для зручності роботи застосунок поділяється на кілька інтерфейсних зон. Верхня частина головного екрану містить фінансові підсумки, центральна

частина використовується для журналу або форми, а нижня частина містить навігацію. Розміщення елементів орієнтоване на вертикальний екран смартфона. При відкритті на комп'ютері ширина основної області може бути обмежена, щоб зберегти мобільну логіку перегляду та забезпечити коректне відображення під час демонстрації. Основні компоненти інтерфейсу застосунку представлено в таблиці 3.1.

Таблиця 3.1 – Основні компоненти інтерфейсу застосунку

Компонент	**Призначення**	**Дані, з якими працює компонент**
Header	Відображення назви застосунку та короткого опису	Назва програмного продукту, поточний екран
SummaryCards	Показ фінансових підсумків	Доходи, витрати, баланс, залишок бюджету
OperationForm	Додавання або редагування фінансового запису	Тип операції, сума, категорія, дата, опис
TransactionList	Перегляд історії доходів і витрат	Масив фінансових операцій
BudgetPanel	Контроль бюджетного ліміту	Ліміт витрат, сума витрат, відсоток використання
StatisticsView	Відображення аналітичних показників	Витрати за категоріями, підсумкові значення
BottomNavigation	Перехід між основними екранами	Активний екран і доступні розділи
SettingsPanel	Налаштування базових параметрів	Валюта, очищення даних, службова інформація

Компонент SummaryCards має важливе значення для користувача, оскільки саме через нього студент одразу бачить поточний фінансовий стан. Після додавання доходу або витрати картки оновлюють показники без перезавантаження сторінки. Компонент OperationForm забезпечує внесення нових записів і повинен містити мінімальну кількість полів, необхідних для коректного обліку. Компонент TransactionList відображає журнал операцій у вигляді компактних карток, зручних для вертикального прокручування на смартфоні.

Логіка роботи застосунку ґрунтується на єдиному наборі фінансових операцій. Кожна операція містить ідентифікатор, тип, суму, категорію, дату та опис. Після додавання запису масив операцій оновлюється, а функції розрахунку визначають загальну суму доходів, загальну суму витрат і поточний баланс. Для статистичного екрана операції групуються за категоріями, що дає змогу показати основні напрями витрачання коштів. Бюджетний блок використовує суму витрат і встановлений ліміт для обчислення відсотку використання доступного ресурсу. Логічні функції програмного продукту представлено в таблиці 3.2.

Таблиця 3.2 – Логічні функції програмного продукту

Функція	**Призначення**	**Результат виконання**
addTransaction	Додавання нового фінансового запису	Операція з’являється в журналі та враховується в підсумках
updateTransaction	Зміна наявного запису	Дані операції оновлюються, баланс перераховується
deleteTransaction	Видалення помилкового запису	Операція прибирається з журналу, показники змінюються
calculateIncome	Обчислення суми доходів	Формується загальний показник надходжень
calculateExpenses	Обчислення суми витрат	Формується загальний показник платежів
calculateBalance	Визначення поточного залишку	Користувач бачить доступну суму коштів
groupByCategory	Групування витрат за категоріями	Статистика отримує дані для візуального відображення
saveToStorage	Збереження інформації в браузері	Дані залишаються доступними після перезавантаження
loadFromStorage	Відновлення інформації під час запуску	Застосунок відкривається з попереднім станом

Реалізація логічних функцій повинна забезпечувати узгодженість всіх екранів. Якщо користувач додає витрату, зміна має відобразитись на головному

екрані, у журналі, бюджетному блоці та статистиці. Якщо запис видалено, усі залежні показники повинні бути перераховані.

Локальне збереження інформації виконується через Web Storage API. Для збереження масиву операцій застосунок перетворює дані в текстовий формат і записує їх в браузерне сховище. Під час повторного запуску інформація зчитується, перетворюється в структуру JavaScript та передається до стану застосунку. Окремо можуть зберігатись бюджетний ліміт, обрана валюта та інші базові параметри. Вибір локального сховища відповідає демонстраційній логіці програмного продукту, оскільки не потребує реєстрації користувача, серверної бази даних або підключення зовнішніх сервісів. Дані, які зберігаються в локальному сховищі браузера наведено в таблиці 3.3.

Таблиця 3.3 – Дані, які зберігаються в локальному сховищі браузера

Назва даних	**Зміст**	**Призначення**
transactions	Набір доходів і витрат користувача	Відновлення журналу операцій після запуску
budgetLimit	Планова сума витрат на період	Контроль використання бюджету
currency	Позначення валюти	Коректне відображення фінансових сум
selectedPeriod	Обраний період перегляду	Фільтрація записів і статистики
appSettings	Базові параметри інтерфейсу	Збереження користувацьких налаштувань

Структура даних має бути достатньо простою, щоб забезпечити швидку роботу застосунку. Для кожної операції використовується об’єкт з фіксованими полями. Ідентифікатор потрібен для редагування та видалення запису, тип операції відокремлює доходи від витрат, сума бере участь в розрахунках, категорія потрібна для статистики, дата забезпечує впорядкування журналу, а опис надає користувачу можливість уточнити призначення платежу. Стандартизація запису зменшує ризик помилок під час обробки інформації. Структура реалізації мобільно орієнтованого вебзастосунку представлена на рисунку 3.1.

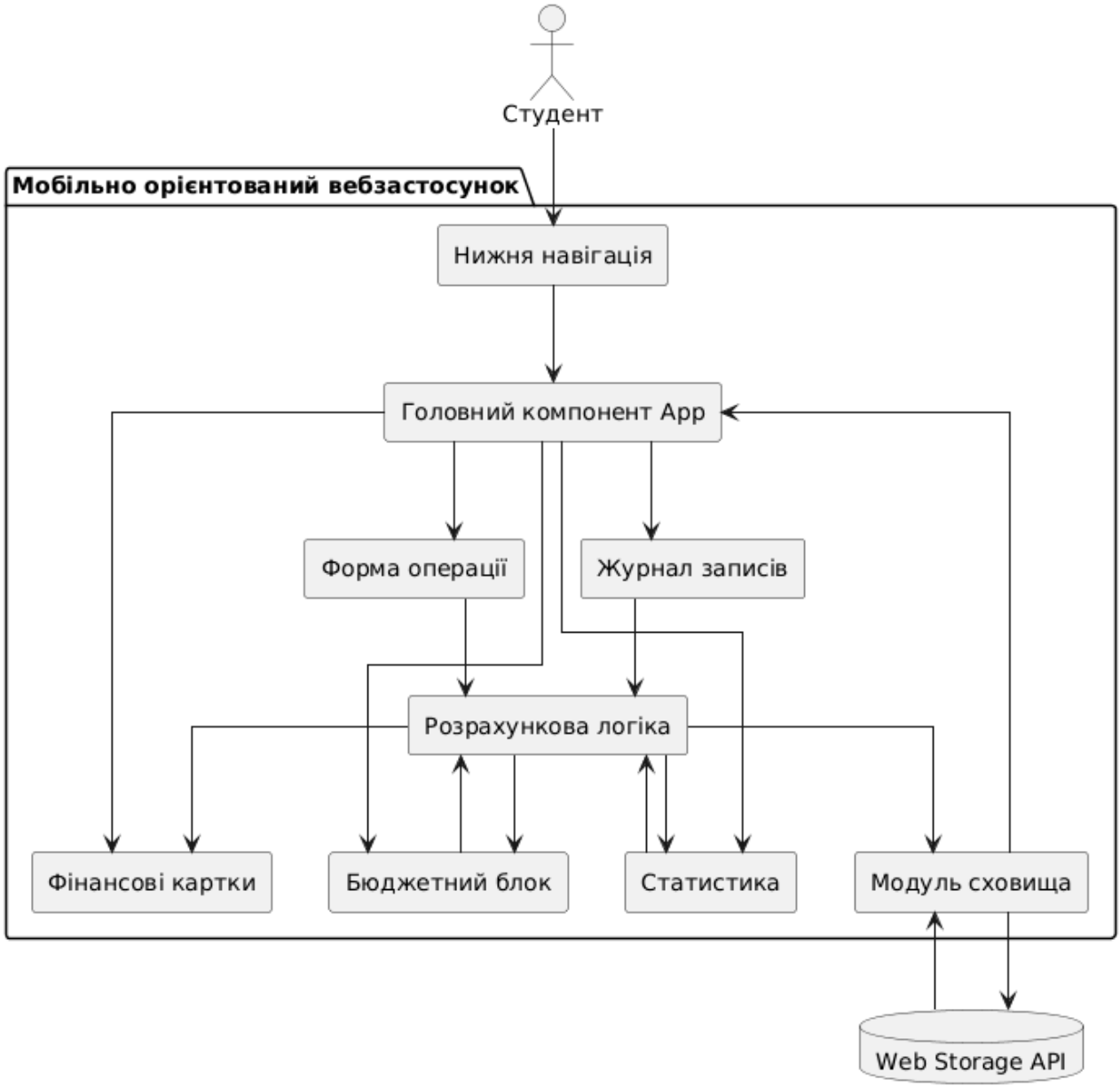

Рисунок 3.1 – Структура реалізації мобільно орієнтованого вебзастосунку

Користувач взаємодіє з застосунком через нижню навігацію та основні екрани. Форма операції, журнал, бюджетний блок і статистика звертаються до розрахункової логіки, яка оновлює фінансові показники. Модуль сховища забезпечує зв’язок з Web Storage API та відповідає за збереження інформації між сеансами роботи.

Під час реалізації мобільного інтерфейсу важливо врахувати компонування екранів. Головний екран повинен містити підсумкові показники у верхній частині, короткий огляд останніх операцій в центральній зоні та

нижню панель переходів. Форма операції має бути лаконічною, щоб студент міг швидко додати платіж після покупки. Журнал фінансів повинен підтримувати вертикальний перегляд записів, а статистика має подавати дані у візуально зрозумілій формі. Налаштування розміщуються окремо, щоб не перевантажувати основні екрани.

Підготовка застосунку до запуску в браузері виконується через стандартні команди середовища Vite. Після створення проєкту встановлюються необхідні залежності, запускається локальний сервер розробки, а готовий інтерфейс відкривається за локальною адресою.

Обрана структура забезпечує взаємозв’язок між фінансовими операціями, локальним збереженням і візуальним відображенням результатів. Завдяки цьому програмний продукт може виконувати основні завдання студентського фінансового планування, а саме, вести журнал доходів і витрат, показувати баланс, контролювати бюджетний ліміт і формувати зрозумілу статистику.

3.2 Інтерфейсне оформлення екранів обліку особистих фінансів

Основною метою оформлення є забезпечення швидкого доступу до фінансових показників, зручного внесення доходів і витрат, перегляду журналу операцій, контролю бюджетного ліміту та аналізу структури витрат. Візуальна організація екранів побудована так, щоб користувач міг працювати з застосунком на смартфоні без масштабування сторінки та без переходу через складне меню.

Під час створення інтерфейсу застосовано карткову структуру, нижню навігаційну панель, короткі форми введення та підсумкові блоки. Картки дають змогу виокремити головні фінансові показники, а навігаційна панель забезпечує швидкий перехід між розділами. В програмному продукті передбачено екрани головної сторінки, журналу, нового запису, бюджету, статистики та налаштувань. Кожен екран має окреме призначення, але всі вони працюють з єдиним набором фінансових операцій, який зберігається в браузері користувача

[29]. Призначення основних екранів реалізованого застосунку наведено в таблиці 3.4.

Таблиця 3.4 – Призначення основних екранів реалізованого застосунку

Екран	**Функціональне призначення**	**Основні елементи інтерфейсу**
Головна сторінка	Відображення поточного фінансового стану	Баланс, доходи, витрати, ліміт, резерв, форма швидкої операції
Журнал	Перегляд збережених фінансових записів	Фільтр типу операції, сортування, картки доходів і витрат
Запис	Додавання або редагування фінансової операції	Сума, тип операції, категорія, дата, опис, кнопка збереження
Бюджет	Контроль планової суми витрат	Поле ліміту, індикатор використання, фінансова підказка
Статистика	Аналіз структури витрат	Діаграма категорій, підсумкова таблиця витрат
Параметри	Зміна службових налаштувань	Вибір валюти, демонстраційні записи, очищення інформації

Головна сторінка виконує роль центрального екрану застосунку. У верхній частині відображається назва StudentFinance, кнопка зміни теми та поточний баланс. Нижче розміщено картки доходів, витрат, ліміту та резерву. Користувач одразу після відкриття бачить основний фінансовий стан і може оцінити доступний залишок коштів. Головна сторінка мобільного додатку StudentFinance наведена на рисунку 3.2.

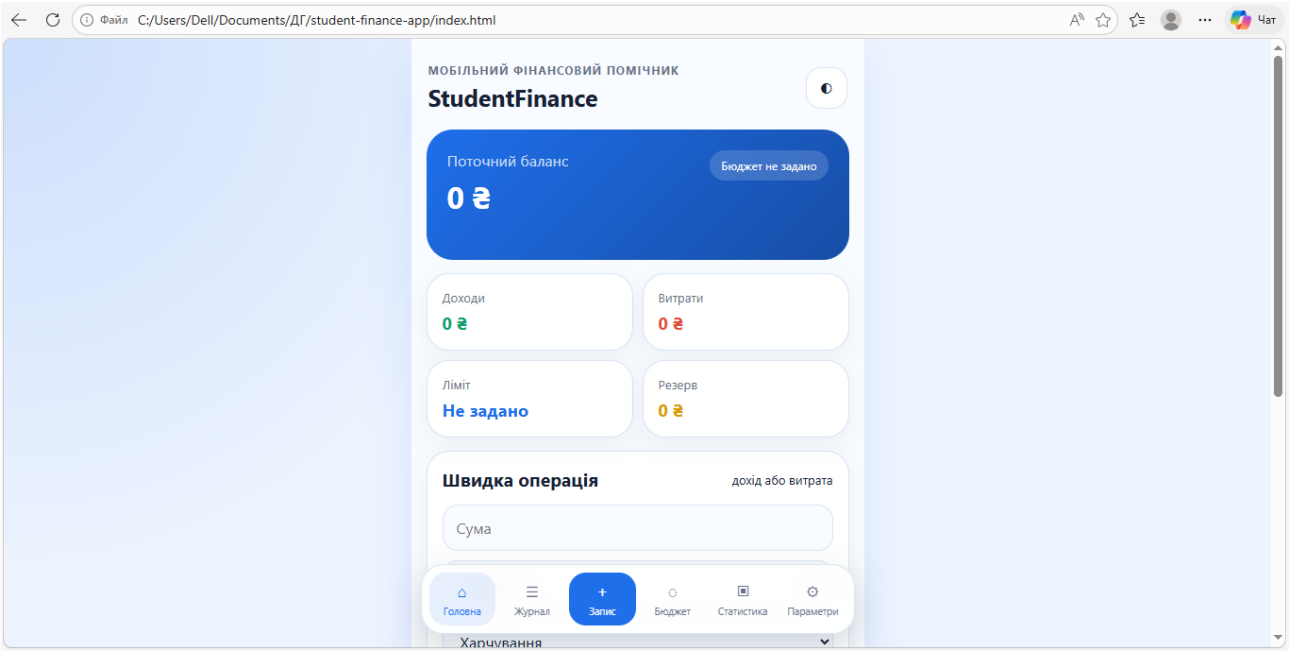

Рисунок 3.2 – Головна сторінка мобільного додатку StudentFinance

Головна сторінка також містить блок швидкої операції. Він дає змогу внести дохід або витрату без переходу на окремий екран запису. Користувач вводить суму, обирає тип операції, категорію, за потреби додає короткий опис і натискає кнопку додавання. Після збереження, фінансові показники автоматично оновлюються, а операція з’являється в журналі та в блоці останніх записів.

Перевага розміщення форми швидкої операції на головному екрані полягає в скороченні кількості дій користувача. Якщо додавання платежу займає небагато часу, користувач частіше веде облік і отримує повнішу картину власного бюджету. Блок швидкого додавання фінансової операції наведено на рисунку 3.3.

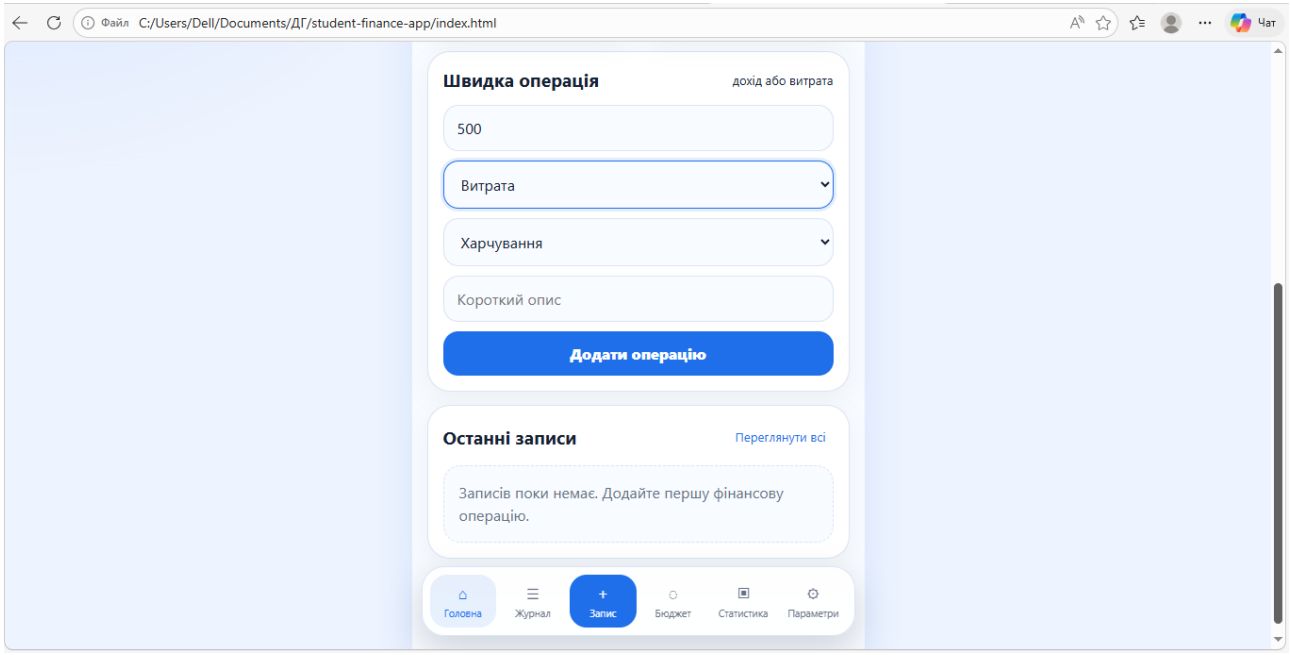

Рисунок 3.3 – Блок швидкого додавання фінансової операції

Екран журналу призначений для перегляду всіх внесених доходів і витрат. Записи відображаються у вигляді окремих карток, що містять суму, категорію, дату та опис операції. Для зручності користувача передбачено фільтр за типом операції та режим сортування. Студент може переглянути лише доходи, лише витрати або всі записи, а також, змінити порядок відображення за новизною чи розміром суми.

Журнал фінансів має важливе значення для контролю правильності введених даних. Якщо користувач помилково зазначив суму або категорію, запис можна відредагувати. Якщо операція була створена випадково, її можна видалити. Після редагування або видалення застосунок перераховує баланс, витрати, доходи, резерв і статистику, що забезпечує узгодженість всіх екранів. Екран журналу фінансових операцій наведено на рисунку 3.4.

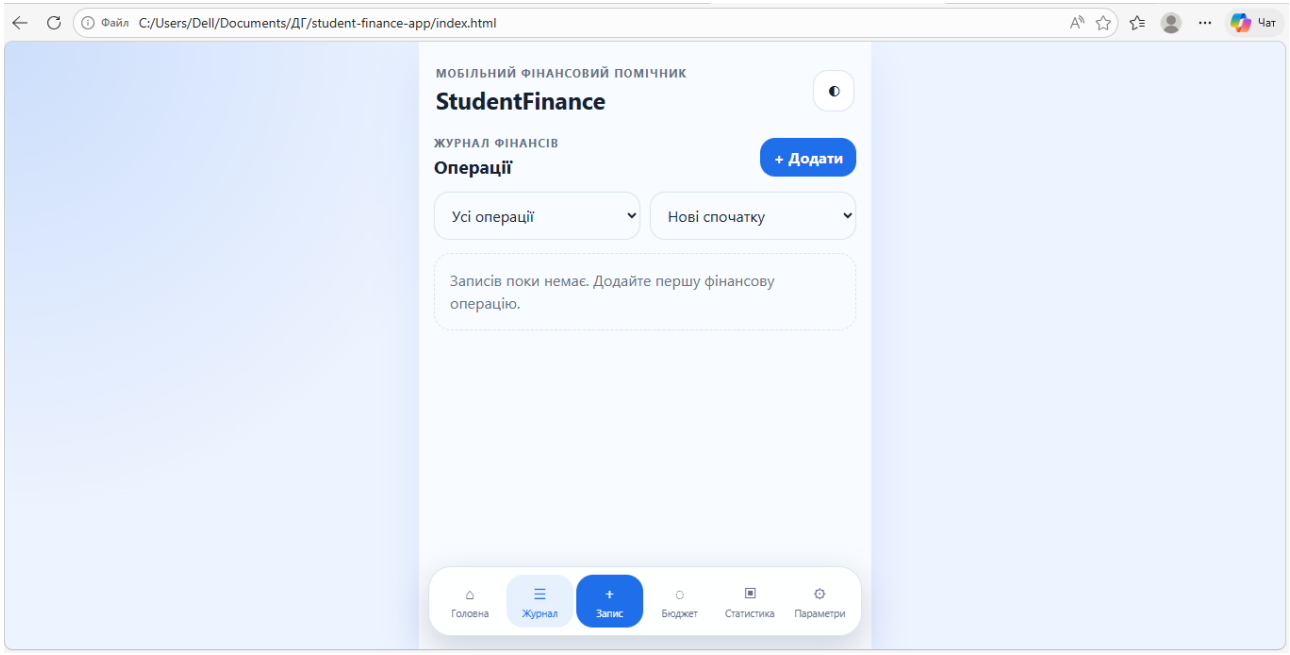

Рисунок 3.4 – Екран журналу фінансових операцій

Окремий екран запису використовується для детального внесення фінансової операції. На ньому розміщено поля для суми, типу операції, категорії, дати та опису. На відміну від швидкої форми на головній сторінці, екран запису дає більше простору для введення інформації, тому він зручний для внесення платежів з поясненням або для редагування вже наявної операції.

Логіка оформлення форми спрямована на зменшення помилок користувача. Поле суми приймає числове значення, тип операції обирається зі списку, а категорія залежить від обраного типу. Якщо користувач додає витрату, застосунок пропонує категорії, пов’язані з харчуванням, транспортом, навчанням, проживанням та іншими напрямами. Для доходів доступні категорії стипендії, підробітку, допомоги родини та інших надходжень. Екран створення нового фінансового запису наведено на рисунку 3.5.

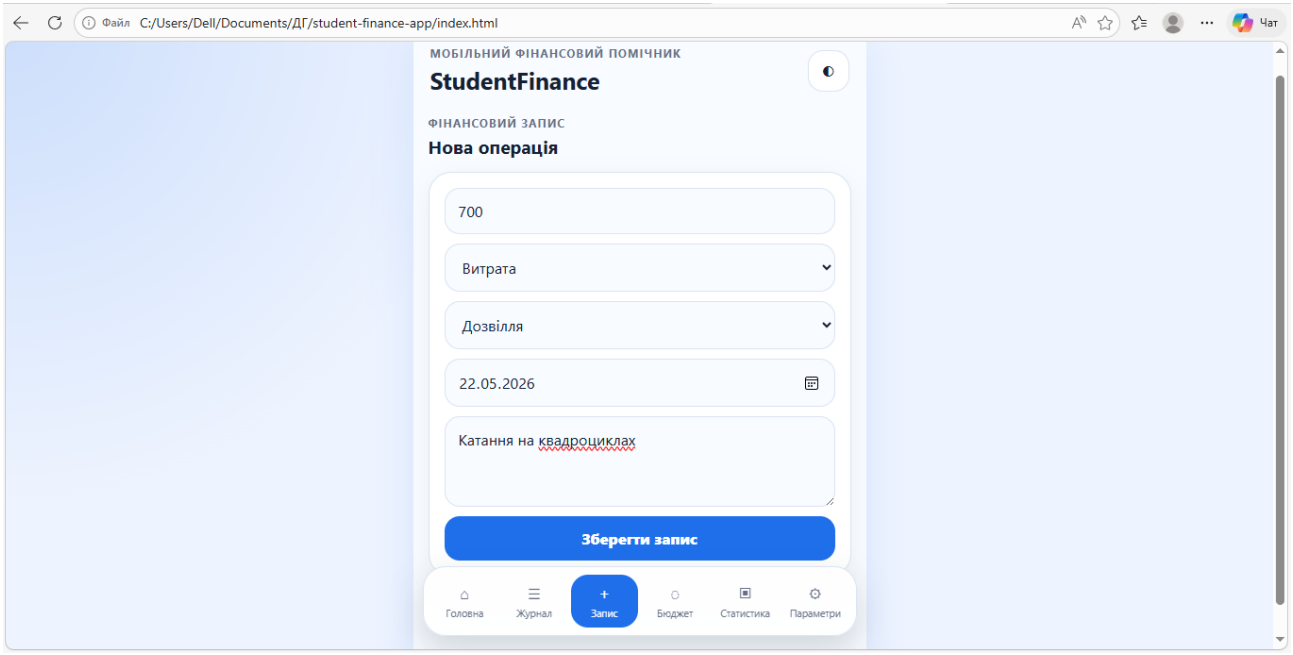

Рисунок 3.5 – Екран створення нового фінансового запису

Екран бюджетного ліміту забезпечує планування максимально допустимої суми витрат. Користувач вводить планову суму, натискає кнопку збереження, після чого застосунок розраховує відсоток використання бюджету. На екрані також відображається індикатор прогресу, який показує, яку частину ліміту вже витрачено. Візуальний індикатор полегшує сприйняття фінансового стану та допомагає швидко оцінити ризик перевищення запланованої суми.

Фінансова підказка на екрані бюджету змінюється залежно від стану витрат. Якщо бюджет використано частково, користувач отримує повідомлення про стабільний стан. Якщо витрати наближаються до встановленої суми або перевищують її, застосунок подає попередження щодо потреби уважнішого контролю. Для студентів така функція є корисною, оскільки вона підтримує регулярну оцінку поточного фінансового навантаження. Екран контролю бюджетного ліміту наведено на рисунку 3.6.

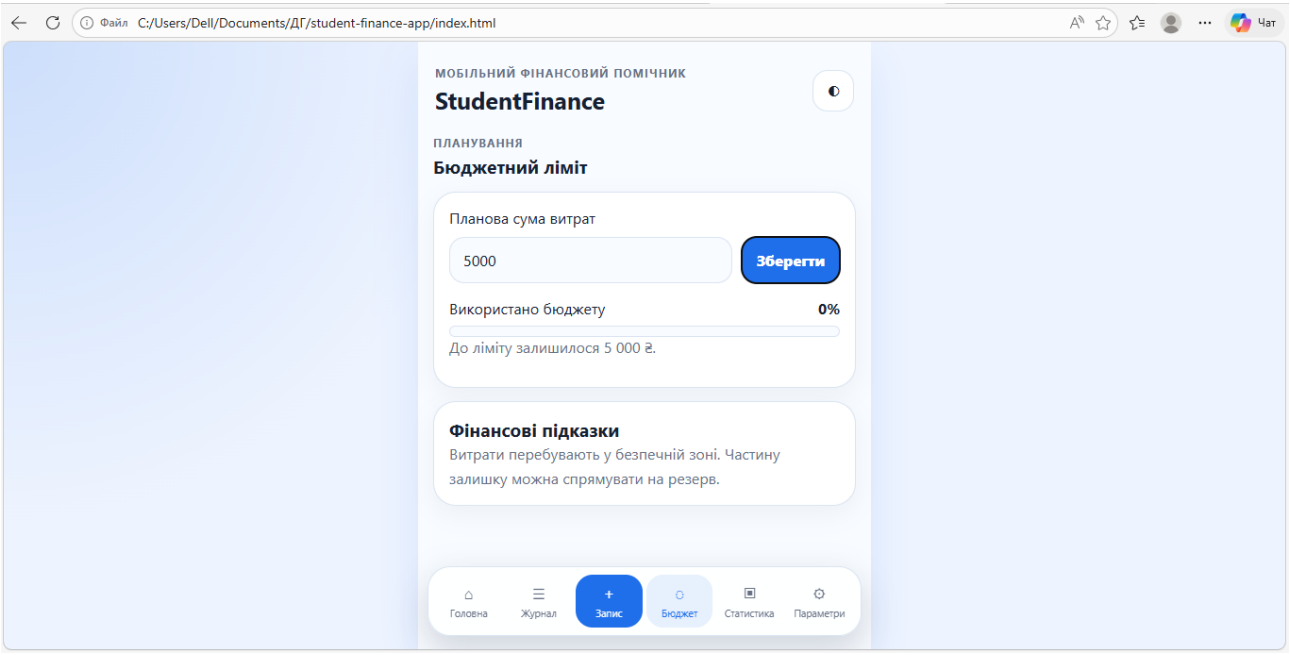

Рисунок 3.6 – Екран контролю бюджетного ліміту

Екран статистики призначений для аналітичного представлення витрат. В ньому реалізовано блок структури витрат та підсумкову таблицю за категоріями. Діаграма відображає співвідношення між напрямами платежів, а таблиця показує суму витрат і частку кожної категорії. Завдяки цьому, користувач може визначити, які витрати мають найбільший вплив на бюджет.

Статистичний екран не перевантажено складними фінансовими показниками, оскільки застосунок орієнтований на повсякденний студентський облік. Основна мета полягає в швидкому виявленні домінуючих категорій витрат. Якщо найбільша частка припадає на харчування або дозвілля, студент може переглянути власну фінансову поведінку та скоригувати майбутні платежі. Екран статистики витрат за категоріями наведено на рисунку 3.7.

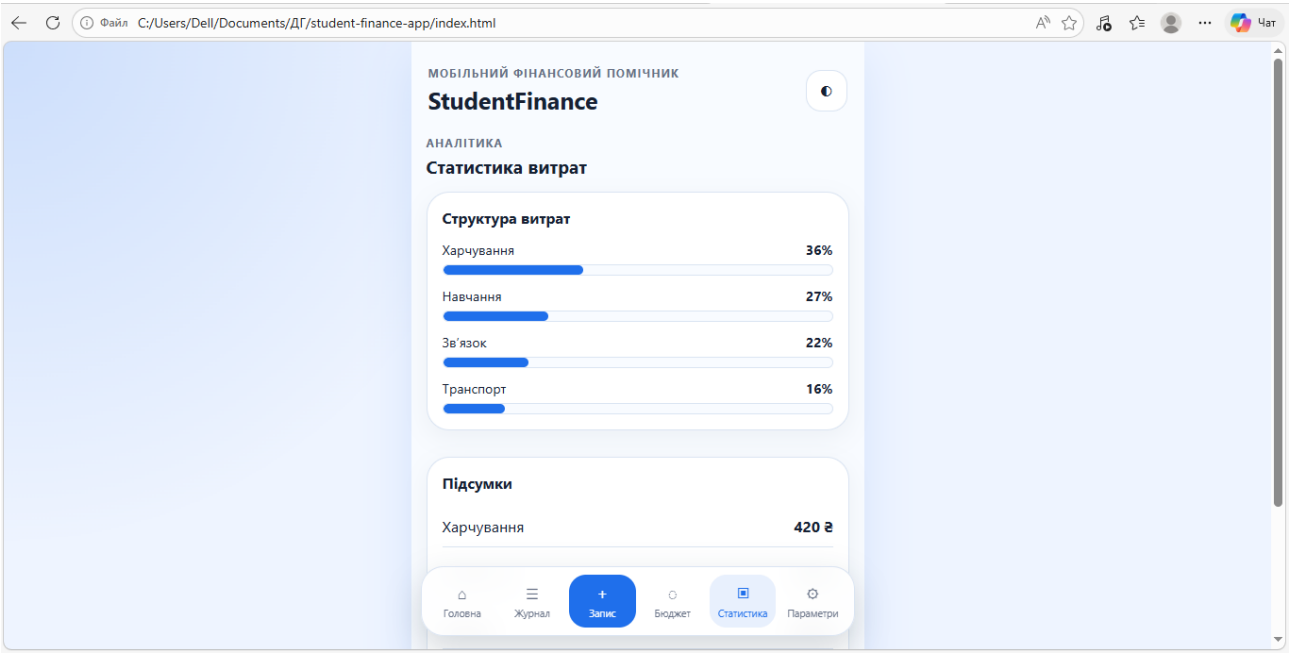

Рисунок 3.7 – Екран статистики витрат за категоріями

Екран параметрів містить службові можливості, які допомагають підготувати застосунок до демонстрації та змінити базові налаштування. Користувач може обрати валюту відображення, додати демонстраційні записи або очистити всі локальні дані. Наявність демонстраційного наповнення є корисною для перевірки роботи інтерфейсу, оскільки після натискання відповідної кнопки автоматично заповнюються журнал, головний екран, бюджетні показники та статистика.

Кнопка очищення даних використовується для повернення застосунку до початкового стану. Вона потрібна під час тестування, коли необхідно повторити сценарій роботи без попередніх операцій. Вибір валюти впливає на відображення сум в картках, журналі, бюджеті та статистиці. Завдяки цьому користувач може адаптувати фінансову інформацію до зручного формату перегляду. Екран параметрів мобільного додатку наведено на рисунку 3.8.

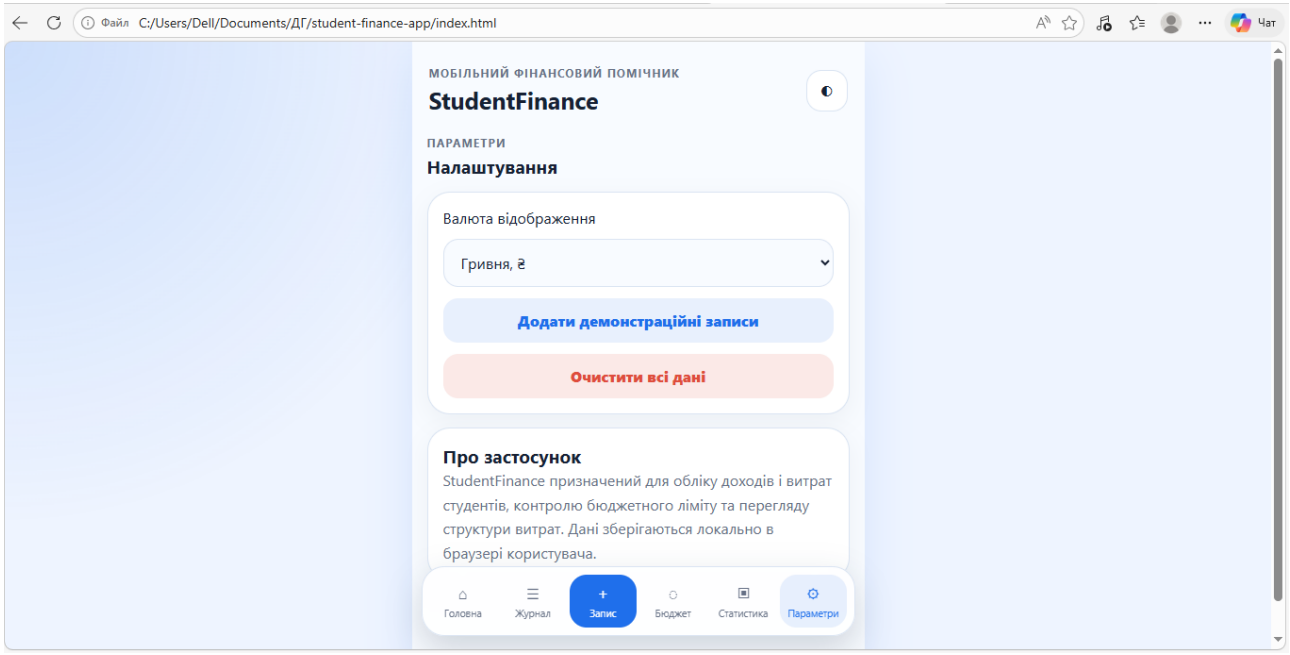

Рисунок 3.8 – Екран параметрів мобільного додатку

Окремою перевагою інтерфейсу є підтримка зміни теми оформлення. У верхній частині застосунку розміщено кнопку перемикання світлого та темного вигляду. Наявність темного режиму підвищує зручність користування за умов недостатнього освітлення та дає змогу продемонструвати гнучкість візуального оформлення. Перемикання теми не впливає на фінансові записи, оскільки змінюється лише зовнішній вигляд елементів. Відображення застосунку в темному режимі наведено на рисунку 3.9.

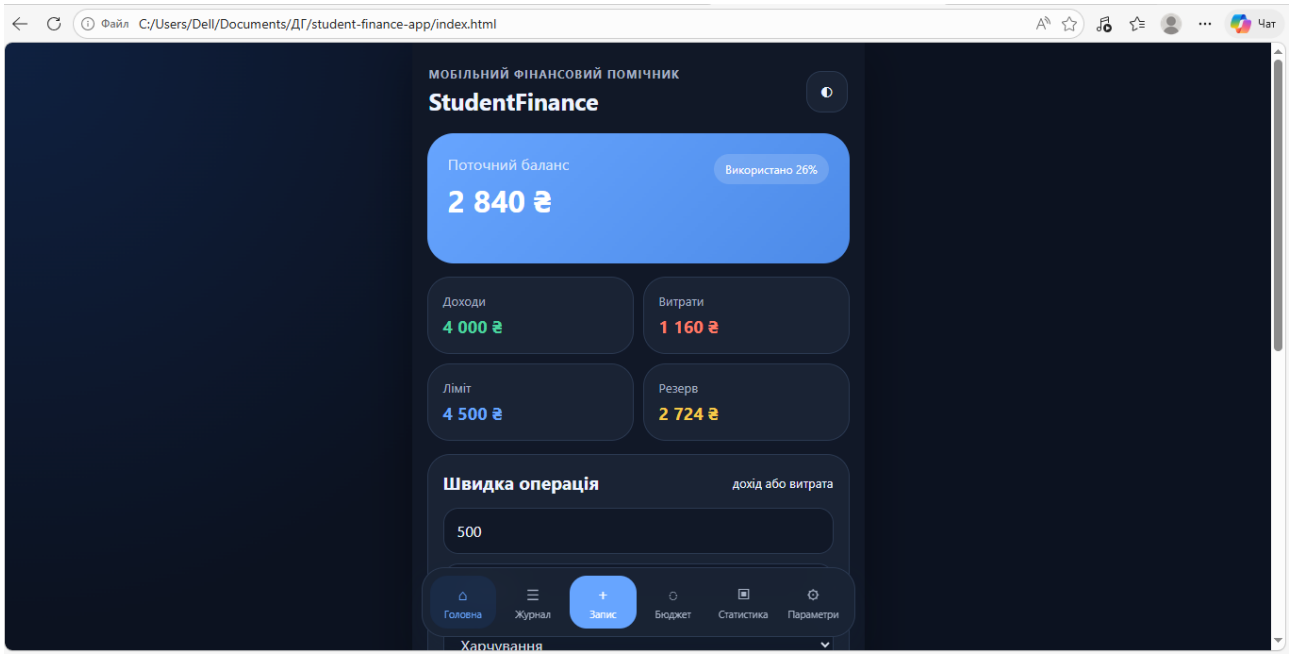

Рисунок 3.9 – Відображення застосунку в темному режимі

Інтерфейсне оформлення реалізованого застосунку відповідає вимогам, сформованим у другому розділі дипломної роботи. Програмний продукт має мобільну структуру, зрозумілу навігацію, карткове представлення фінансових показників і зручні форми введення операцій. Кожен екран виконує окреме завдання, але всі елементи працюють узгоджено через спільну логіку обробки та локального збереження даних.

3.3 Перевірка роботи функцій контролю витрат і планування бюджету

Перед початком перевірки застосунок відкривається через файл index.html. Для отримання заповненого стану можна використати кнопку додавання демонстраційних записів на екрані параметрів. Після її натискання в журналі з’являються приклади доходів і витрат, встановлюється бюджетний ліміт, а головний екран, бюджетний блок і статистика отримують дані для подальшого аналізу. Наявність демонстраційного набору прискорює перевірку, оскільки користувач одразу може оцінити роботу всіх основних модулів.

Після додавання демонстраційних записів застосунок показує оновлений баланс, суму доходів, суму витрат, бюджетний ліміт і резерв. Вказаний результат підтверджує, що дані з параметрів передаються до загального набору фінансових операцій, після чого виконуються автоматичні розрахунки. Важливо, що зміни відображаються без ручного оновлення сторінки, адже логіка інтерфейсу реагує на зміну даних одразу після виконання дії.

Для перевірки додавання витрати використано форму нового запису. Користувач переходить на екран «Запис», вводить суму операції, обирає тип «Витрата», задає категорію, встановлює дату та додає короткий опис. Після натискання кнопки збереження, операція додається до журналу, а загальна сума витрат і поточний баланс змінюються. Вказана перевірка демонструє працездатність форми, валідації введення та автоматичного оновлення підсумкових показників. Повідомлення про успішне збереження нової витрати наведено на рисунку 3.10.

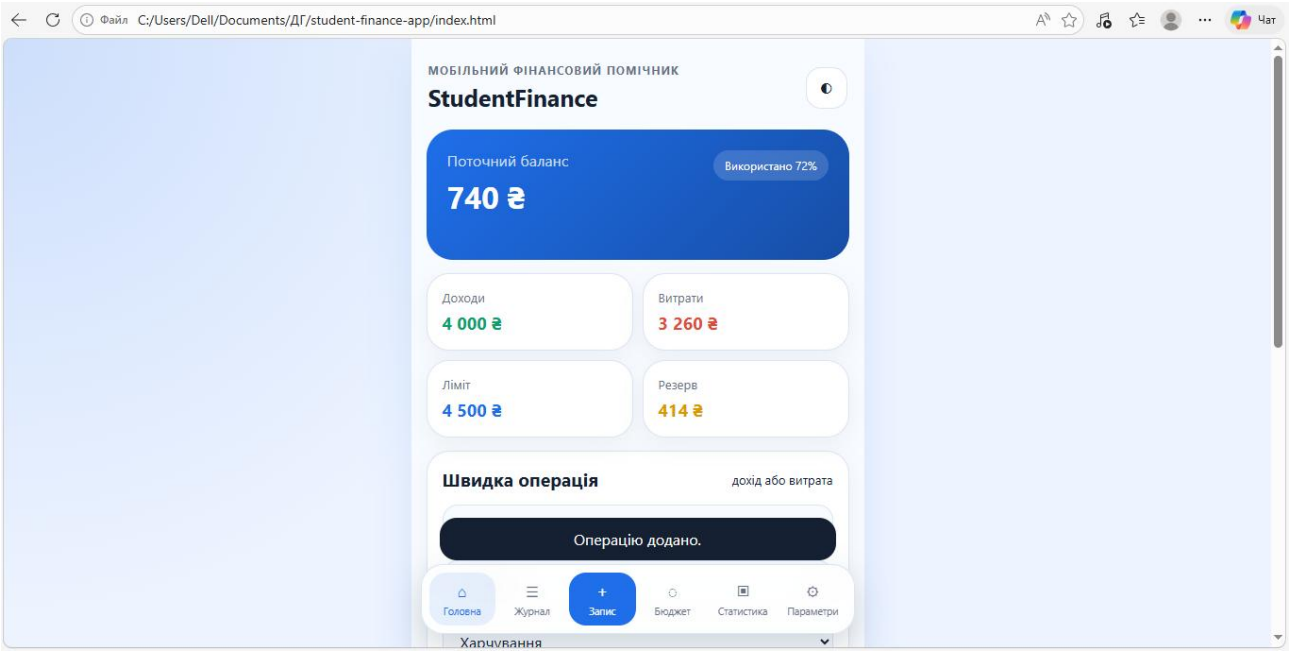

Рисунок 3.10 – Повідомлення про успішне збереження нової витрати

Перевірка додавання доходу виконується аналогічно, однак в полі типу операції обирається значення «Дохід». Після зміни типу застосунок автоматично змінює перелік категорій, пропонуючи варіанти, пов’язані зі стипендією, підробітком, допомогою родини або іншим надходженням. Після збереження дохід збільшує загальну суму надходжень і поточний баланс. Зміна переліку категорій підтверджує, що форма враховує відмінність між доходами та витратами.

Окремо перевірено роботу журналу фінансових операцій. Після створення кількох записів користувач може фільтрувати інформацію за типом операції та змінювати порядок сортування. Для контролю витрат важливим є режим відображення лише витрат, оскільки він дає змогу швидко оцінити платежі без змішування з доходами. Сортування за найбільшими сумами допомагає виявити фінансові операції, які найбільше впливають на студентський бюджет. Фільтрація журналу за витратами та найбільшими сумами наведена на рисунку 3.11.

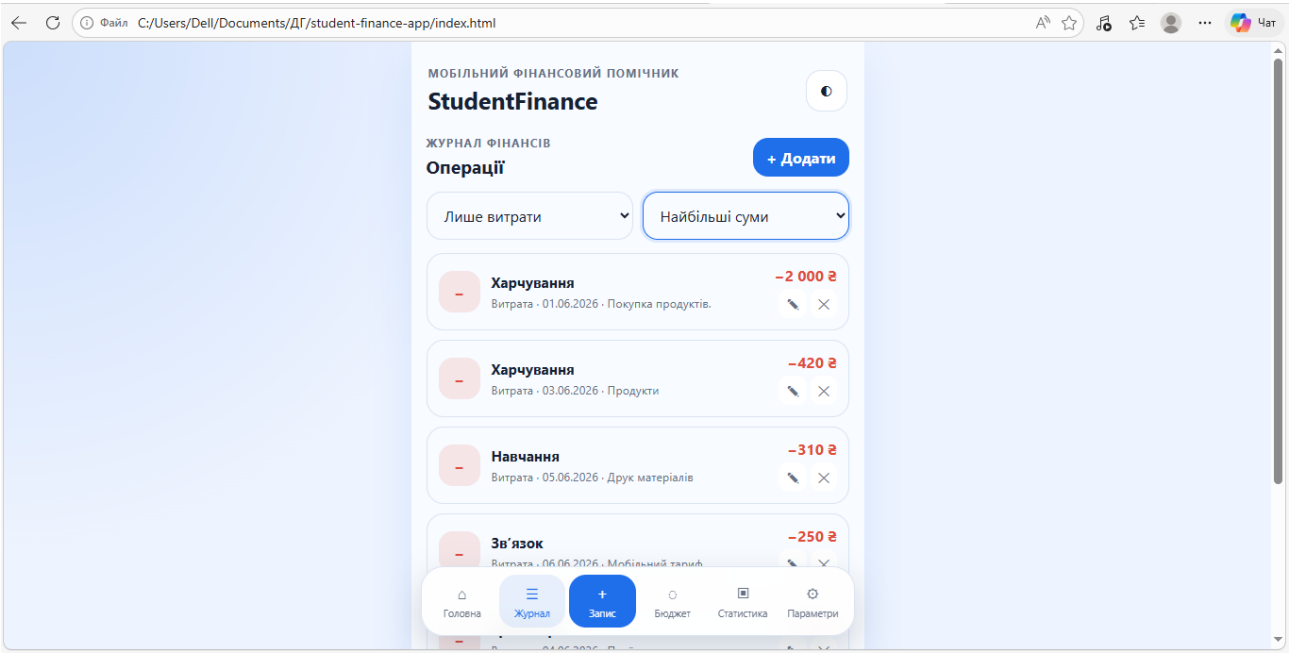

Рисунок 3.11 – Фільтрація журналу за витратами та найбільшими сумами

Фільтрація журналу підтверджує, що застосунок може працювати не лише з повним набором записів, але й з окремими групами операцій. В режимі «Лише витрати» користувач бачить платежі, які зменшують бюджет. В режимі сортування за найбільшою сумою записи розміщуються за спаданням фінансового навантаження. Завдяки цьому студент може швидко виявити найбільш значні витрати та прийняти рішення щодо їх скорочення.

Редагування фінансового запису перевірено через кнопку з символом олівця в журналі. Після її натискання застосунок відкриває екран запису, де поля форми заповнюються значеннями обраної операції. Користувач може змінити суму, категорію, дату або опис, після чого натискає кнопку оновлення. Після збереження нових значень журнал та фінансові підсумки перераховуються. Режим редагування фінансової операції представлено на рисунку 3.12.

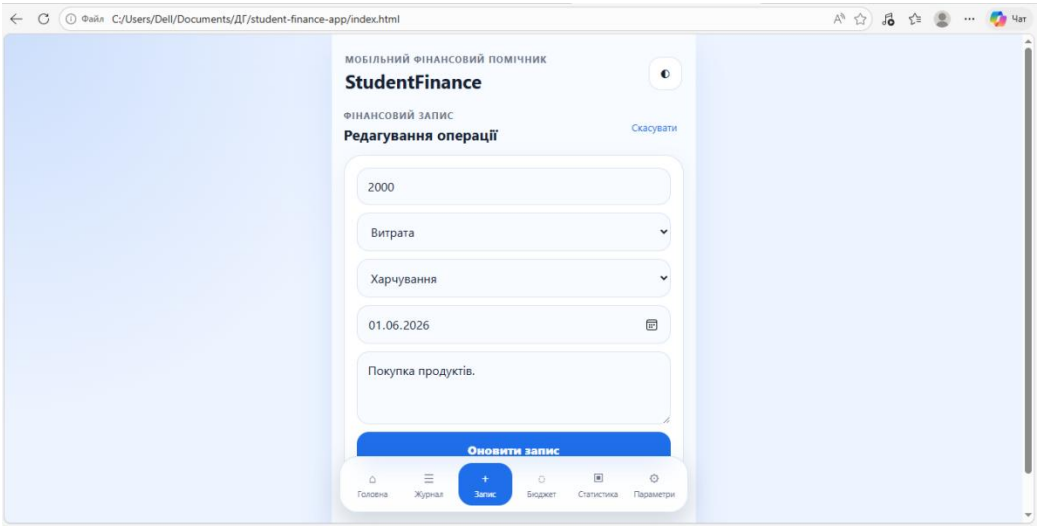

Рисунок 3.12 – Режим редагування фінансової операції

Режим редагування є важливим для практичного користування, оскільки помилки під час введення сум або категорій трапляються доволі часто. Без можливості виправлення користувач був би змушений видаляти запис та створювати його повторно. Наявність заповненої форми редагування скорочує кількість дій та підвищує точність ведення фінансового журналу.

Функція видалення перевіряється через кнопку з символом хрестика біля запису. Перед остаточним видаленням браузер показує вікно підтвердження. Якщо користувач погоджується, операція зникає з журналу, а баланс і статистика перераховуються. Наявність підтвердження зменшує ризик випадкового видалення фінансової інформації. Підтвердження видалення фінансового запису наведено на рисунку 3.13.

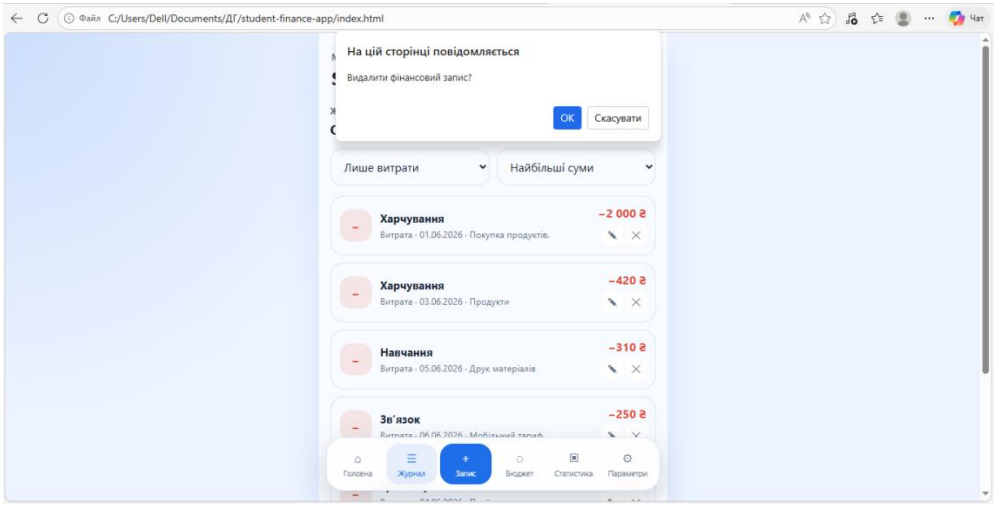

Рисунок 3.13 – Підтвердження видалення фінансового запису

Перевірка планування бюджету здійснюється через екран «Бюджет». Користувач вводить планову суму витрат і натискає кнопку збереження. Після цього застосунок порівнює суму фактичних витрат з встановленим лімітом, показує відсоток використання бюджету, заповнює індикатор прогресу та формує текстову фінансову підказку. Якщо витрати незначні порівняно з лімітом, користувач отримує повідомлення про безпечну зону. Якщо витрати наближаються до ліміту або перевищують його, підказка змінюється. Відображення бюджетного попередження після зміни ліміту наведена на рисунку 3.14.

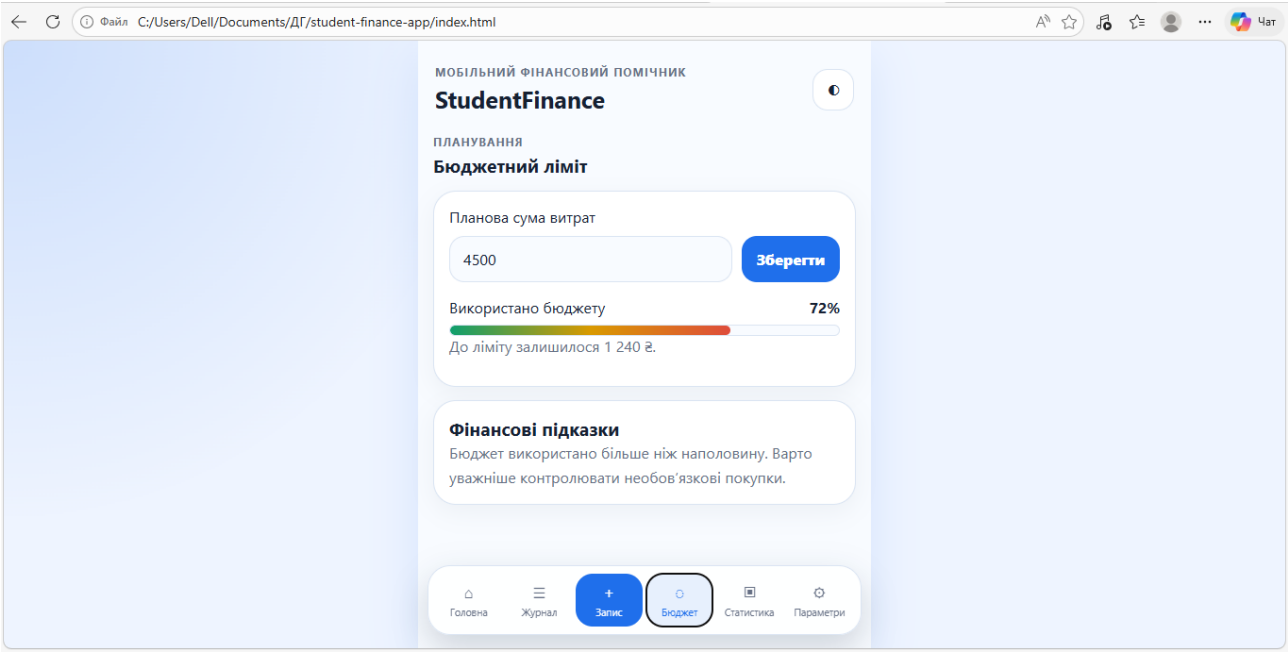

Рисунок 3.14 – Відображення бюджетного попередження після зміни ліміту

Бюджетний блок перевіряє не лише факт збереження ліміту, але й здатність застосунку реагувати на зміну фінансового стану. Якщо користувач після встановлення ліміту додає нову витрату, відсоток використання бюджету має змінитися. Це підтверджує зв’язок між журналом операцій, розрахунковою логікою та екраном планування.

Перевірка статистики виконується після створення кількох витрат в різних категоріях. Застосунок групує витрати за категоріями, визначає суму кожної групи та розраховує частку від загальної суми витрат. На екрані

статистики відображаються горизонтальні індикатори, які дають змогу швидко побачити домінуючі напрями платежів. Підсумкова таблиця дублює числові значення, що підвищує точність аналізу. Оновлення статистики після додавання витрат різних категорій наведено на рисунку 3.15.

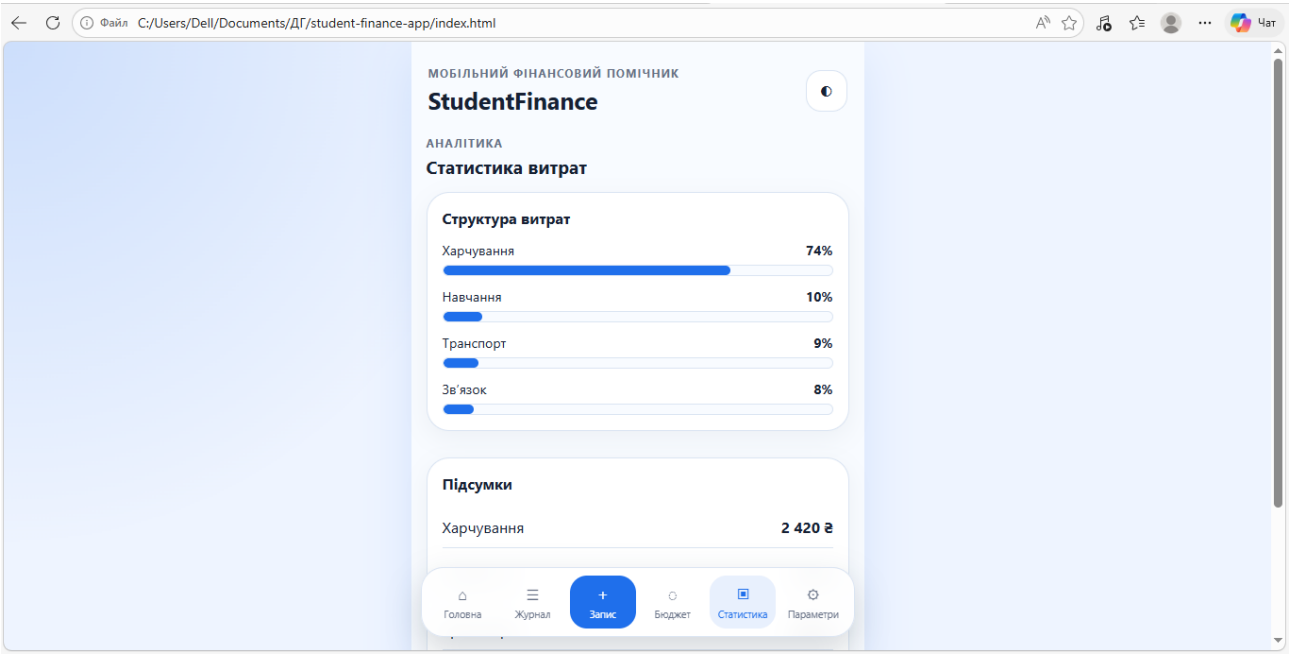

Рисунок 3.15 – Оновлення статистики після додавання витрат різних категорій

Статистика підтверджує, що застосунок не обмежується простим журналом записів, а виконує базовий аналіз фінансової поведінки. Для студента результат має практичне значення, оскільки він бачить категорії з найбільшою сумою витрат. Якщо найбільша частка припадає на необов'язкові платежі, користувач може скоригувати майбутній фінансовий план.

Локальне збереження даних перевірено через перезавантаження сторінки після додавання операцій і встановлення бюджетного ліміту. Після повторного відкриття або оновлення сторінки фінансові записи, обрана валюта, тема оформлення та бюджетний ліміт залишаються доступними. Це підтверджує коректну роботу Web Storage API та правильне відновлення стану застосунку під час запуску.

Проведена перевірка засвідчила, що реалізований мобільний додаток виконує основні функції контролю витрат та планування бюджету. Користувач може додавати доходи та витрати, переглядати журнал, редагувати записи,

видаляти помилкові операції, встановлювати бюджетний ліміт та аналізувати структуру витрат. Підсумкові показники оновлюються після кожної зміни, а інформація зберігається локально в браузері.

3.4 Оцінювання результатів використання розробленого програмного продукту

Для оцінювання використано контрольний сценарій, наближений до реальної поведінки студента. Користувач відкриває застосунок в браузері, додає кілька доходів, фіксує витрати на харчування, транспорт, навчання та дозвілля, встановлює бюджетний ліміт, переглядає журнал операцій та аналізує структуру витрат. Після завершення дій перевіряється оновлення балансу, стан бюджету, правильність відображення журналу, робота статистики та збереження інформації після перезавантаження сторінки [30]. Критерії оцінювання результатів використання застосунку наведено в таблиці 3.5

Таблиця 3.5 – Критерії оцінювання результатів використання застосунку

Критерій оцінювання	**Зміст критерію**	**Очікуваний результат**
Повнота фінансового обліку	Наявність дій для додавання доходів і витрат	Користувач може вести базовий журнал студентського бюджету
Актуальність підсумків	Перерахунок балансу після зміни записів	Фінансові показники відповідають журналу операцій
Зручність навігації	Доступ до розділів через нижню панель	Перехід між екранами виконується без складного меню
Контроль бюджету	Порівняння витрат із плановим лімітом	Користувач бачить рівень використання бюджету
Аналітичність	Групування витрат за категоріями	Визначаються напрями найбільшого фінансового навантаження
Збереження інформації	Відновлення записів після перезавантаження	Дані залишаються доступними у браузері
Демонстраційна придатність	Можливість швидко підготувати приклади	Застосунок можна показати під час захисту роботи

Проведене оцінювання показало, що програмний продукт охоплює повний базовий цикл роботи з особистими фінансами. Користувач може створити фінансовий запис, зберегти його, переглянути в журналі, відредагувати, видалити, оцінити вплив на баланс і побачити зміну бюджетного ліміту. Вказана послідовність відповідає вимогам, сформованим на етапі проєктування, та демонструє практичну завершеність основних модулів застосунку.

Одним з важливих результатів є можливість швидкого переходу від порожнього стану до заповненого фінансового журналу. Порожній стан застосунку потрібен для початку нового облікового періоду, очищення демонстраційних даних або повторної перевірки роботи програми. Після очищення інформації головний екран має показувати нульові показники, а журнал і статистика мають бути порожніми. Початковий стан застосунку після очищення фінансових даних наведено на рисунку 3.16.

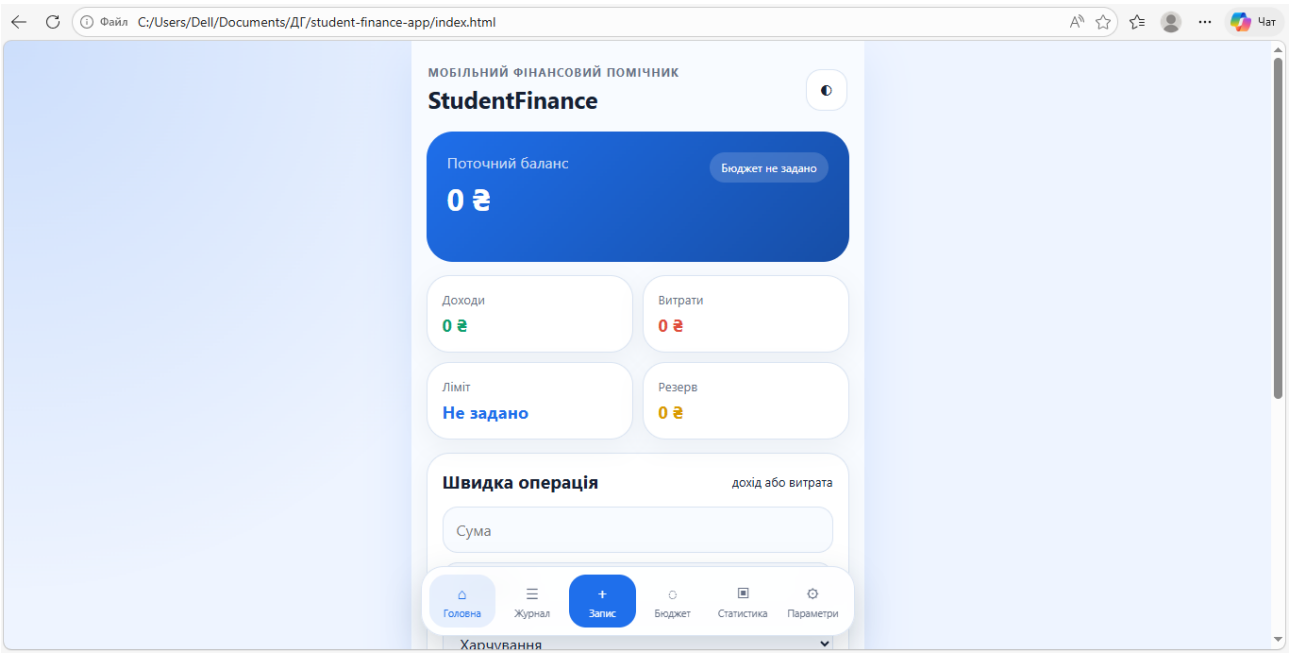

Рисунок 3.16 – Початковий стан застосунку після очищення фінансових даних

Початковий стан після очищення даних підтверджує, що застосунок може коректно скидати локальне сховище та повертатись до вихідного вигляду.

Оцінювання швидкості виконання основних дій здійснено за кількістю кроків, потрібних користувачу для досягнення результату. Для студентів важливо, щоб додавання витрат або перевірка балансу не вимагали значної кількості переходів. В реалізованому застосунку перегляд балансу доступний одразу після відкриття, а додавання швидкої операції може виконуватись з головного екрана.

Бюджетний блок оцінювався через встановлення ліміту, який перевищує поточні витрати. В такому стані застосунок повинен показати, що витрати перебувають в прийнятних межах, а користувач має резерв для подальших платежів. Даний результат відрізняється від попередньої перевірки попередження про перевищення ліміту, оскільки демонструє позитивний сценарій бюджетного планування. Результат контролю бюджету при безпечному рівні витрат представлено на рисунку 3.17.

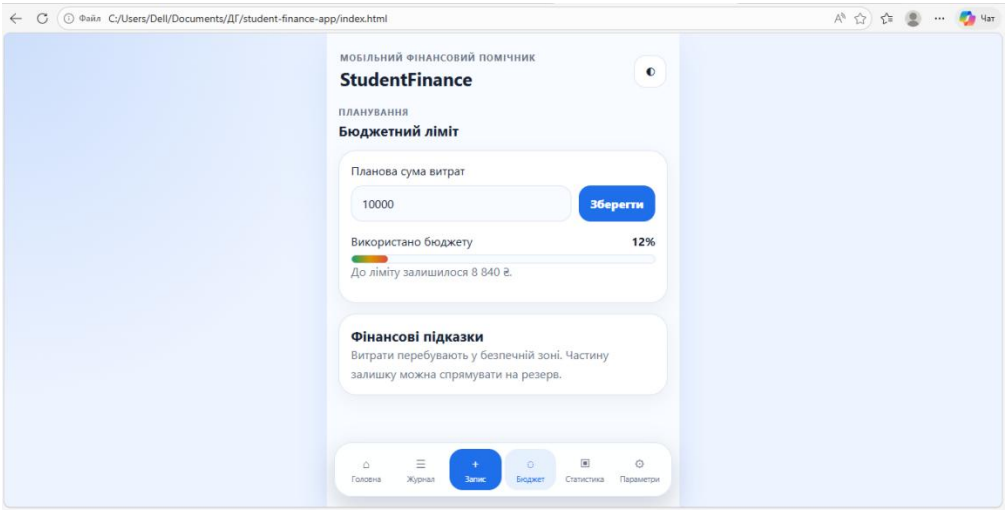

Рисунок 3.17 – Результат контролю бюджету за безпечного рівня витрат

Стан безпечного рівня витрат показує, що застосунок може не лише попереджати про перевищення бюджету, але й фіксувати прийнятну фінансову ситуацію.

Оцінювання журналу виконано після коригування контрольного запису. Для цього користувач змінює суму або категорію однієї операції, повертається до журналу та перевіряє оновлений запис. Фінансові підсумки після

редагування мають відповідати новому значенню. Важливим результатом є те, що виправлення помилки не потребує створення нового запису, а журнал залишається цілісним. Журнал фінансів після коригування контрольної операції наведено на рисунку 3.18.

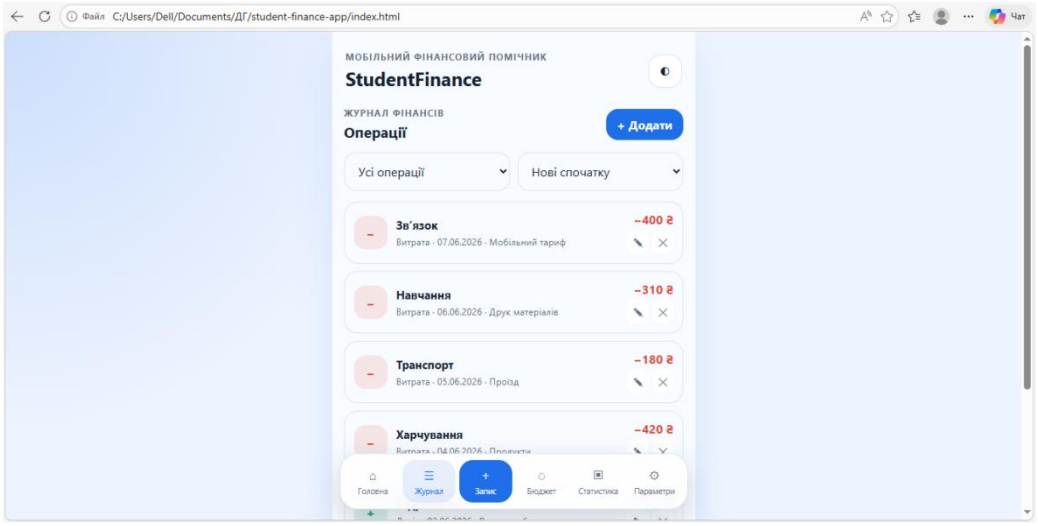

Рисунок 3.18 – Журнал фінансів після коригування контрольної операції

Журнал після коригування контрольної операції підтверджує працездатність редагування в завершеному сценарії. Користувач бачить вже не форму редагування, а результат зміни в історії фінансових записів.

Додатково оцінено зміну валюти відображення. В параметрах користувач може обрати інше позначення валюти, після чого суми на екранах застосунку відображаються з новим символом. Вказана функція не змінює арифметичні значення операцій, однак підвищує гнучкість представлення фінансової інформації. Відображення фінансових показників після зміни валюти наведено на рисунку 3.19.

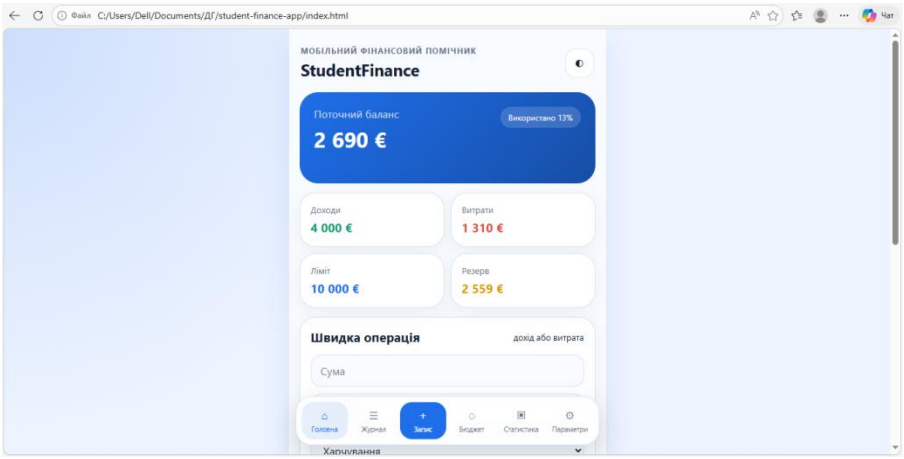

Рисунок 3.19 – Відображення фінансових показників після зміни валюти

Оцінювання аналітичного результату виконано через порівняння фінансових показників до та після введення контрольного набору операцій. Після додавання доходу, кількох витрат і бюджетного ліміту застосунок формує баланс, суму витрат, резерв і структуру категорій. На основі даних показників користувач отримує можливість оцінити власну фінансову поведінку без ручних розрахунків.

Наведені результати демонструють, що застосунок може бути використаний як інструмент швидкої оцінки студентського бюджету. Користувач отримує не лише перелік операцій, але й узагальнені значення, які допомагають приймати фінансові рішення. Показник резерву до ліміту є особливо важливим, оскільки він вказує на доступний простір для подальших витрат.

Окремо варто оцінити обмеження розробленого програмного продукту. Застосунок призначений для навчального використання, тому не містить банківської синхронізації, облікових записів, хмарного сховища та автоматичного імпорту транзакцій. Всі операції вводяться вручну, а дані зберігаються в конкретному браузері. Переваги та обмеження розробленого застосунку представлено в таблиці 3.6.

Таблиця 3.16 – Переваги та обмеження розробленого застосунку

Характеристика	**Позитивний результат**	**Обмеження**
Браузерний запуск	Застосунок відкривається без інсталяції	Дані прив’язані до конкретного браузера
Локальне збереження	Інформація не передається на зовнішні сервери	Немає синхронізації між пристроями
Мобільний інтерфейс	Екрани зручні для вертикального перегляду	Немає нативних функцій смартфона
Ручне введення операцій	Користувач повністю контролює записи	Відсутній автоматичний імпорт банківських платежів
Демонстраційні записи	Перевірка функцій виконується швидко	Демонстраційні дані не замінюють реального фінансового періоду
Категоризація витрат	Статистика показує напрями платежів	Набір категорій має базовий характер

StudentFinance може використовуватися студентом для щоденного контролю власних коштів. Його функції дають змогу бачити залишок, фіксувати невеликі платежі, планувати ліміт витрат і визначати категорії, які найбільше впливають на бюджет. Завдяки локальному збереженню користувач може продовжити роботу після повторного відкриття сторінки, а демонстраційний режим дозволяє швидко показати можливості програми під час захисту.

Висновки до розділу 3

В третьому розділі кваліфікаційної роботи виконано практичну реалізацію мобільного додатку StudentFinance, призначеного для оптимізації особистих фінансів студентів. Програмний продукт створено як мобільно орієнтований вебзастосунок з браузерним запуском, що забезпечує доступність використання без встановлення окремого інсталяційного пакету. В процесі реалізації застосовано React, Vite, JavaScript, CSS і Web Storage API, що дало

змогу побудувати адаптивний інтерфейс, організувати обробку фінансових записів та забезпечити локальне збереження інформації.

Проведено перевірку роботи функцій контролю витрат і планування бюджету. Результати тестування підтвердили коректність додавання доходів і витрат, редагування записів, видалення операцій, фільтрації журналу, встановлення бюджетного ліміту, формування статистики та відновлення даних після перезавантаження сторінки. Окремо оцінено результати використання застосунку за критеріями повноти фінансового обліку, актуальності підсумків, зручності навігації, аналітичності та демонстраційної придатності.

Розроблений мобільний додаток StudentFinance відповідає поставленим вимогам і може застосовуватись як практичний інструмент для щоденного контролю студентського бюджету. Програмний продукт забезпечує облік фінансових операцій, автоматичний перерахунок балансу, контроль витрат відносно встановленого ліміту та візуальне представлення основних напрямів використання коштів.

ВИСНОВКИ

В кваліфікаційній роботі вирішено практичне завдання, пов’язане з розробкою мобільного додатку для оптимізації особистих фінансів студентів. Актуальність обраної теми зумовлена потребою студентської молоді в зручних цифрових інструментах для контролю доходів, витрат, бюджетного ліміту та фінансових звичок. Обмеженість доходів, нерівномірність надходжень та наявність регулярних побутових витрат потребують простого програмного засобу, який допомагає швидко оцінювати поточний фінансовий стан і приймати обґрунтовані рішення щодо подальшого використання коштів.

В процесі виконання роботи проаналізовано особливості управління студентським бюджетом та визначено основні чинники, що впливають на фінансову поведінку здобувачів освіти. На розподіл коштів впливають економічні умови, соціальне середовище, рівень самоконтролю, цифрові звички та досвід самостійного планування. На основі проведеного аналізу обґрунтовано потребу в мобільному рішенні, яке забезпечує оперативне внесення фінансових операцій, зрозумілий перегляд залишку коштів і наочне представлення структури витрат.

Проведено огляд сучасних сервісів для контролю особистих фінансів, зокрема Money Lover, Wallet by BudgetBakers, Monefy та Spendee. Порівняння їх можливостей дало змогу визначити функції, важливі для майбутнього програмного продукту. До найбільш значущих належать облік доходів і витрат, категоризація платежів, бюджетне планування, журнал операцій, візуальна статистика та зручний мобільний інтерфейс. Водночас встановлено, що універсальні фінансові сервіси часто мають надмірну функціональність, яка не завжди відповідає повсякденним потребам студентів.

Для реалізації програмного продукту обрано веборієнтований технологічний стек, який включає React, Vite, JavaScript, CSS і Web Storage API. Застосування вказаних технологій дало змогу створити мобільно орієнтований

вебзастосунок з браузерним запуском, адаптивним інтерфейсом та локальним збереженням інформації. Приведена організація програмного продукту забезпечує доступність використання без встановлення окремого інсталяційного пакета та спрощує демонстрацію роботи застосунку.

Розроблений додаток StudentFinance забезпечує основні функції персонального фінансового обліку. Користувач може додавати доходи та витрати, обирати категорії, переглядати журнал операцій, редагувати або видаляти записи, встановлювати бюджетний ліміт, змінювати валюту відображення, аналізувати витрати за категоріями та використовувати демонстраційні записи для швидкої перевірки роботи. Дані зберігаються локально в браузері, тому після перезавантаження сторінки фінансові записи залишаються доступними.

Перевірка працездатності застосунку підтвердила коректність реалізованих функцій. Після додавання, редагування або видалення фінансової операції показники доходів, витрат, балансу, резерву та бюджетного ліміту оновлюються автоматично. Журнал дає змогу контролювати історію платежів, а статистика допомагає визначати категорії з найбільшим фінансовим навантаженням. Наявність бюджетного індикатора сприяє своєчасному виявленню ризику перевищення планової суми витрат.

Практичне значення отриманих результатів полягає в створенні доступного цифрового інструменту, який може застосовуватись студентами для щоденного контролю особистих фінансів. Програмний продукт допомагає впорядковувати фінансові записи, аналізувати структуру витрат, оцінювати залишок коштів і планувати подальше використання бюджету.

Подальший розвиток програмного продукту може бути пов’язаний з додаванням авторизації користувача, хмарного збереження, експорту фінансових записів, нагадувань про регулярні платежі, цілей накопичення та розширеної аналітики. Водночас, реалізована версія повністю виконує поставлене завдання, оскільки забезпечує базову оптимізацію студентського бюджету через облік операцій, бюджетне планування та наочний аналіз витрат.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Національний банк України. Фінансова грамотність, фінансова інклюзія та фінансовий добробут в Україні у 2021 році : звіт за результатами дослідження. Київ : НБУ, 2021. 83 с. URL: https://bank.gov.ua/admin_uploads/article/Research_Financial_Literacy_Inclusion_Welfare_2021.pdf
2. Письменний В. В. Фінансова грамотність (рівень стандарту) : підручник для 10 (11) класу закладів загальної середньої освіти. Тернопіль : Астон, 2023. 240 с. URL: https://elartu.tntu.edu.ua/bitstream/lib/45889/1/%D0%A4%D1%96%D0%BD%D0%B0%D0%BD%D1%81%D0%BE%D0%B2%D0%B0%20%D0%B3%D1%80%D0%B0%D0%BC%D0%BE%D1%82%D0%BD%D1%96%D1%81%D1%82%D1%8C.pdf
3. Фінансова грамотність. Фінанси. Що? Чому? Як? : навчальний посібник / керівники авторських колективів: Т. С. Грищенко, Ю. В. Вітка, Т. О. Смовженко, А. Я. Кузнєцова. Київ : ДВНЗ «Університет банківської справи», 2019. 272 с. URL: https://events.bank.gov.ua/gmw2021/src/files/Financial_Literacy_Textbook_Aug2019.pdf
4. Фінанси : підручник / В. П. Горин, В. Г. Дем'янишин, О. П. Кириленко та ін. ; за ред. А. І. Крисоватого. Тернопіль : ЗУНУ, 2024. 632 с. URL: https://dspace.wunu.edu.ua/items/63f8ea7b-c00e-45bb-b36d-32c1e7646343
5. Національна стратегія розвитку фінансової грамотності до 2030 року. Київ, 2024. 44 с. URL: https://me.gov.ua/download/aea9bd7a-0491-4004-b5ba-f84892effc1d/file.pdf
6. Кукель Г. С. Конспект лекцій з дисципліни «Управління особистими доходами та витратами» для студентів спеціальності 075 «Маркетинг».

Вінниця : ВНТУ, 2023. 74 с. URL: https://pdf.lib.vntu.edu.ua/books/2024/Kukel_2023_74.pdf

7. Нікіфоров П. О., Марич М. Г. Поведінкові фінанси : навчальний посібник. Чернівці : Чернівецький національний університет імені Юрія Федьковича, 2024. 181 с. URL: https://archer.chnu.edu.ua/bitstream/handle/123456789/10710/%D0%9F%D0%BE%D0%B2%D0%B5%D0%B4%D1%96%D0%BD%D0%BA%D0%BE%D0%B2%D1%96%20%D1%84%D1%96%D0%BD%D0%B0%D0%BD%D1%81%D0%B8.pdf?isAllowed=y&sequence=3
8. Гончаренко І. Г., Бережна Л. В., Березіна О. Ю. Інтерактивний посібник з фінансової грамотності. Черкаси : Видавець Третяков О. М., 2024. 120 с. URL: https://er.chdtu.edu.ua/handle/ChSTU/5471 (дата звернення: 07.06.2026).
9. Державна служба статистики України. Витрати і ресурси домогосподарств України в 2021 році : статистичний бюлетень. URL: https://www.gusrv.gov.ua/statinform/menu-vitraty.htm
10. Міністерство освіти і науки України. Навчальні матеріали «Фінансова грамотність. Фінанси. Що? Чому? Як?» для учнів 10, 11 класів. URL: https://mon.gov.ua/osvita-2/zagalna-serednya-osvita/navchalni-materiali-programi-pidruchniki-ta-navchalno-metodichni-posibniki/navchalni-materiali-finansova-gramotnist-finansi-shcho-chomu-yak-dlya-uchniv-10-11-klasiv
11. Money Lover. No. 1 Expense Manager Budget Planner. URL: https://moneylover.me/
12. BudgetBakers. Wallet Features – Complete Financial Management Tool. URL: https://budgetbakers.com/en/products/wallet/features/
13. Monefy. Budget & Track Your Money. URL: https://monefy.com/
14. Spendee. Money Manager & Budget Planner. URL: https://www.spendee.com/
15. Money Lover: Money Manager. Google Play. URL: https://play.google.com/store/apps/details?id=com.bookmark.money&hl=en

16. BudgetBakers Help Center. What is the Wallet app? URL: https://support.budgetbakers.com/hc/en-us/articles/12212428113810-What-is-the-Wallet-app
17. Monefy: Money Tracker. App Store. URL: https://apps.apple.com/nl/app/monefy-money-tracker/id1212024409?l=en-GB
18. Spendee Help Center. What is Spendee? URL: https://help.spendee.com/article/114-what-is-spendee
19. ISO/IEC/IEEE 29148:2018. Systems and software engineering – Life cycle processes – Requirements engineering. URL: https://www.iso.org/standard/72089.html
20. React. Create user interfaces from components. URL: https://react.dev/
21. Vite. Getting Started. URL: https://vite.dev/guide/
22. MDN Web Docs. JavaScript Guide. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
23. MDN Web Docs. HTML elements reference. URL: https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements
24. MDN Web Docs. Using media queries. URL: https://developer.mozilla.org/en-US/docs/Web/CSS/Guides/Media_queries/Using
25. MDN Web Docs. Window: localStorage property. URL: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
26. MDN Web Docs. Progressive web apps. URL: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
27. ISO 9241-210:2019. Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems. URL: https://www.iso.org/standard/77520.html
28. Nielsen Norman Group. Basic Patterns for Mobile Navigation: A Primer. URL: https://www.nngroup.com/articles/mobile-navigation-patterns/
29. Nielsen Norman Group. Cards: UI-Component Definition. URL: https://www.nngroup.com/articles/cards-component/

30. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation – System and software quality models. URL: https://www.iso.org/standard/35733.html

ДОДАТОК А

Код основного файлу мобільного додатку StudentFinance

```
const STORAGE_KEY = 'studentFinance.transactions.v1';
const SETTINGS_KEY = 'studentFinance.settings.v1';

const incomeCategories = ['Стипендія', 'Підробіток', 'Допомога родини', 'Інше
надходження'];
const expenseCategories = ['Харчування', 'Транспорт', 'Навчання', 'Проживання',
'Зв’язок', 'Дозвілля', 'Здоров’я', 'Інші витрати'];

const defaultSettings = {
  currency: '₴',
  budgetLimit: 0,
  theme: 'light',
};

let transactions = loadTransactions();
let settings = loadSettings();
let activeScreen = 'home';
let editId = null;

const $ = (selector) => document.querySelector(selector);
const $$ = (selector) => Array.from(document.querySelectorAll(selector));

const formatDateInput = (date = new Date()) => date.toISOString().slice(0, 10);

function formatMoney(value) {
  const number = Number(value) || 0;
  return `${number.toLocaleString('uk-UA', { maximumFractionDigits: 2 })}
${settings.currency}`;
}

function loadTransactions() {
  try {
    const data = localStorage.getItem(STORAGE_KEY);
    return data ? JSON.parse(data) : [];
  } catch (error) {
    return [];
  }
}

function saveTransactions() {
  localStorage.setItem(STORAGE_KEY, JSON.stringify(transactions));
}

function loadSettings() {
  try {
    const data = localStorage.getItem(SETTINGS_KEY);
    return data ? { ...defaultSettings, ...JSON.parse(data) } : {
...defaultSettings };
  } catch (error) {
    return { ...defaultSettings };
  }
}

function saveSettings() {
  localStorage.setItem(SETTINGS_KEY, JSON.stringify(settings));
}

function showToast(message) {
  const toast = $('#toast');
```

```
  toast.textContent = message;
  toast.classList.add('show');
  window.setTimeout(() => toast.classList.remove('show'), 2200);
}

function setTheme() {
  document.body.classList.toggle('dark', settings.theme === 'dark');
}

function updateCategoryOptions(typeSelect, categorySelect) {
  const currentType = typeSelect.value;
  const categories = currentType === 'income' ? incomeCategories :
expenseCategories;
  categorySelect.innerHTML = categories.map((category) => `<option
value="${category}">${category}</option>`).join('');
}

function getTotals() {
  const income = transactions.filter((item) => item.type ===
'income').reduce((sum, item) => sum + item.amount, 0);
  const expenses = transactions.filter((item) => item.type ===
'expense').reduce((sum, item) => sum + item.amount, 0);
  const balance = income - expenses;
  const reserve = Math.max(balance - expenses * 0.1, 0);
  return { income, expenses, balance, reserve };
}

function getExpenseGroups() {
  const groups = {};
  transactions
    .filter((item) => item.type === 'expense')
    .forEach((item) => {
      groups[item.category] = (groups[item.category] || 0) + item.amount;
    });
  return Object.entries(groups)
    .map(([category, amount]) => ({ category, amount }))
    .sort((a, b) => b.amount - a.amount);
}

function renderSummary() {
  const { income, expenses, balance, reserve } = getTotals();
  $('#balanceValue').textContent = formatMoney(balance);
  $('#incomeValue').textContent = formatMoney(income);
  $('#expenseValue').textContent = formatMoney(expenses);
  $('#limitValue').textContent = settings.budgetLimit ?
formatMoney(settings.budgetLimit) : 'Не задано';
  $('#reserveValue').textContent = formatMoney(reserve);

  if (!settings.budgetLimit) {
    $('#budgetBadge').textContent = 'Бюджет не задано';
  } else {
    const percent = Math.round((expenses / settings.budgetLimit) * 100);
    $('#budgetBadge').textContent = `Використано ${Math.min(percent, 999)}%`;
  }
}

function renderTransactions(listElement, items, limit = null) {
  const visibleItems = limit ? items.slice(0, limit) : items;
  listElement.innerHTML = '';
  listElement.classList.toggle('empty-state', visibleItems.length === 0);

  visibleItems.forEach((item) => {
    const article = document.createElement('article');
    article.className = 'transaction-item';
    const sign = item.type === 'income' ? '+' : '−';
```

```
    const typeLabel = item.type === 'income' ? 'Дохід' : 'Витрата';
    article.innerHTML = `
      <div class="transaction-icon ${item.type}">${sign}</div>
      <div class="transaction-meta">
        <strong>${item.category}</strong>
        <span>${typeLabel} · ${new Date(item.date).toLocaleDateString('uk-UA')}
· ${item.note || 'без опису'}</span>
      </div>
      <div class="transaction-sum">
        <strong class="${item.type}">${sign}${formatMoney(item.amount)}</strong>
        <div class="action-row">
          <button type="button" data-edit="${item.id}">✎</button>
          <button type="button" data-delete="${item.id}">✕</button>
        </div>
      </div>
    `;
    listElement.appendChild(article);
  });
}

function getFilteredHistory() {
  const filter = $('#filterType').value;
  const sort = $('#sortMode').value;
  let items = [...transactions];

  if (filter !== 'all') {
    items = items.filter((item) => item.type === filter);
  }

  items.sort((a, b) => {
    if (sort === 'oldest') return new Date(a.date) - new Date(b.date);
    if (sort === 'largest') return b.amount - a.amount;
    return new Date(b.date) - new Date(a.date);
  });

  return items;
}

function renderBudget() {
  const { expenses } = getTotals();
  $('#budgetLimit').value = settings.budgetLimit || '';
  const percent = settings.budgetLimit ? Math.round((expenses /
settings.budgetLimit) * 100) : 0;
  $('#budgetPercent').textContent = `${Math.min(percent, 999)}%`;
  $('#budgetProgress').style.width = `${Math.min(percent, 100)}%`;

  if (!settings.budgetLimit) {
    $('#budgetHint').textContent = 'Встановіть планову суму витрат, щоб
контролювати студентський бюджет.';
    $('#financeTip').textContent = 'Додайте кілька операцій і встановіть ліміт,
щоб отримати персональну підказку.';
    return;
  }

  const remain = settings.budgetLimit - expenses;
  $('#budgetHint').textContent = remain >= 0
    ? `До ліміту залишилося ${formatMoney(remain)}.`
    : `Ліміт перевищено на ${formatMoney(Math.abs(remain))}.`;

  if (percent < 50) {
    $('#financeTip').textContent = 'Витрати перебувають у безпечній зоні.
Частину залишку можна спрямувати на резерв.';
  } else if (percent < 85) {
```

```
    $('#financeTip').textContent = 'Бюджет використано більше ніж наполовину.
Варто уважніше контролювати необов’язкові покупки.';
  } else if (percent <= 100) {
    $('#financeTip').textContent = 'Витрати наближаються до встановленого
ліміту. Доцільно скоротити платежі на дозвілля або інші непершочергові
потреби.';
  } else {
    $('#financeTip').textContent = 'Бюджетний ліміт перевищено. Потрібно
переглянути категорії з найбільшими витратами.';
  }
}

function renderStats() {
  const groups = getExpenseGroups();
  const total = groups.reduce((sum, item) => sum + item.amount, 0);
  const chart = $('#categoryChart');
  const table = $('#categoryTable');

  chart.innerHTML = '';
  table.innerHTML = '';

  if (!groups.length) {
    chart.classList.add('empty-state');
    table.classList.add('empty-state');
    return;
  }

  chart.classList.remove('empty-state');
  table.classList.remove('empty-state');

  groups.forEach((item) => {
    const percent = Math.round((item.amount / total) * 100);
    const row = document.createElement('div');
    row.className = 'bar-row';
    row.innerHTML = `
      <div class="bar-
top"><span>${item.category}</span><strong>${percent}%</strong></div>
      <div class="bar-track"><div class="bar-fill"
style="width:${percent}%"></div></div>
    `;
    chart.appendChild(row);

    const line = document.createElement('div');
    line.className = 'category-line';
    line.innerHTML =
`<span>${item.category}</span><strong>${formatMoney(item.amount)}</strong>`;
    table.appendChild(line);
  });
}

function renderAll() {
  renderSummary();
  const recentSorted = [...transactions].sort((a, b) => new Date(b.date) - new
Date(a.date));
  renderTransactions($('#recentList'), recentSorted, 3);
  renderTransactions($('#historyList'), getFilteredHistory());
  renderBudget();
  renderStats();
  $('#currencySelect').value = settings.currency;
}

function navigate(screen) {
  activeScreen = screen;
  $$('.screen').forEach((element) => element.classList.toggle('active',
element.dataset.screen === screen));
```

```
  $$('.nav-btn').forEach((button) => button.classList.toggle('active',
button.dataset.nav === screen));
  renderAll();
  window.scrollTo({ top: 0, behavior: 'smooth' });
}

function resetOperationForm() {
  editId = null;
  $('#formTitle').textContent = 'Нова операція';
  $('#saveOperation').textContent = 'Зберегти запис';
  $('#cancelEdit').classList.add('hidden');
  $('#operationForm').reset();
  $('#date').value = formatDateInput();
  $('#type').value = 'expense';
  updateCategoryOptions($('#type'), $('#category'));
}

function submitOperation(formPrefix = '') {
  const amountEl = $(`#${formPrefix}amount`) || $('#quickAmount');
  const typeEl = $(`#${formPrefix}type`) || $('#quickType');
  const categoryEl = $(`#${formPrefix}category`) || $('#quickCategory');
  const noteEl = $(`#${formPrefix}note`) || $('#quickNote');
  const dateEl = $(`#${formPrefix}date`) || $('#date');

  const amount = Number(amountEl.value);
  if (!amount || amount <= 0) {
    showToast('Введіть коректну суму операції.');
    return false;
  }

  const payload = {
    id: editId || Date.now(),
    amount,
    type: typeEl.value,
    category: categoryEl.value,
    date: dateEl ? dateEl.value : formatDateInput(),
    note: noteEl.value.trim(),
  };

  if (editId) {
    transactions = transactions.map((item) => item.id === editId ? payload :
item);
    showToast('Операцію оновлено.');
  } else {
    transactions.push(payload);
    showToast('Операцію додано.');
  }

  saveTransactions();
  renderAll();
  resetOperationForm();
  return true;
}

function editTransaction(id) {
  const item = transactions.find((transaction) => transaction.id === id);
  if (!item) return;

  editId = id;
  navigate('add');
  $('#formTitle').textContent = 'Редагування операції';
  $('#saveOperation').textContent = 'Оновити запис';
  $('#cancelEdit').classList.remove('hidden');
  $('#amount').value = item.amount;
  $('#type').value = item.type;
```

```
  updateCategoryOptions($('#type'), $('#category'));
  $('#category').value = item.category;
  $('#date').value = item.date;
  $('#note').value = item.note;
}

function deleteTransaction(id) {
  if (!confirm('Видалити фінансовий запис?')) return;
  transactions = transactions.filter((item) => item.id !== id);
  saveTransactions();
  renderAll();
  showToast('Запис видалено.');
}

function seedDemoData() {
  const today = new Date();
  const shiftDate = (days) => {
    const date = new Date(today);
    date.setDate(today.getDate() - days);
    return formatDateInput(date);
  };

  transactions = [
    { id: Date.now() + 1, type: 'income', amount: 2200, category: 'Стипендія',
date: shiftDate(8), note: 'Щомісячне надходження' },
    { id: Date.now() + 2, type: 'income', amount: 1800, category: 'Підробіток',
date: shiftDate(5), note: 'Разова робота' },
    { id: Date.now() + 3, type: 'expense', amount: 420, category: 'Харчування',
date: shiftDate(4), note: 'Продукти' },
    { id: Date.now() + 4, type: 'expense', amount: 180, category: 'Транспорт',
date: shiftDate(3), note: 'Проїзд' },
    { id: Date.now() + 5, type: 'expense', amount: 310, category: 'Навчання',
date: shiftDate(2), note: 'Друк матеріалів' },
    { id: Date.now() + 6, type: 'expense', amount: 250, category: 'Зв’язок',
date: shiftDate(1), note: 'Мобільний тариф' },
  ];
  settings.budgetLimit = 4500;
  saveTransactions();
  saveSettings();
  renderAll();
  showToast('Демонстраційні записи додано.');
}

function bindEvents() {
  $$('[data-nav]').forEach((button) => {
    button.addEventListener('click', () => navigate(button.dataset.nav));
  });

  $('#quickType').addEventListener('change', () =>
updateCategoryOptions($('#quickType'), $('#quickCategory')));
  $('#type').addEventListener('change', () => updateCategoryOptions($('#type'),
$('#category')));

  $('#quickForm').addEventListener('submit', (event) => {
    event.preventDefault();
    const saved = submitQuickOperation();
    if (saved) $('#quickForm').reset();
  });

  $('#operationForm').addEventListener('submit', (event) => {
    event.preventDefault();
    const saved = submitOperation('');
    if (saved) navigate('home');
  });
```

```
  $('#cancelEdit').addEventListener('click', () => {
    resetOperationForm();
    navigate('history');
  });

  $('#historyList').addEventListener('click', (event) => {
    const editButton = event.target.closest('[data-edit]');
    const deleteButton = event.target.closest('[data-delete]');
    if (editButton) editTransaction(Number(editButton.dataset.edit));
    if (deleteButton) deleteTransaction(Number(deleteButton.dataset.delete));
  });

  $('#recentList').addEventListener('click', (event) => {
    const editButton = event.target.closest('[data-edit]');
    const deleteButton = event.target.closest('[data-delete]');
    if (editButton) editTransaction(Number(editButton.dataset.edit));
    if (deleteButton) deleteTransaction(Number(deleteButton.dataset.delete));
  });

  $('#filterType').addEventListener('change', renderAll);
  $('#sortMode').addEventListener('change', renderAll);

  $('#budgetForm').addEventListener('submit', (event) => {
    event.preventDefault();
    settings.budgetLimit = Number($('#budgetLimit').value) || 0;
    saveSettings();
    renderAll();
    showToast('Бюджетний ліміт збережено.');
  });

  $('#currencySelect').addEventListener('change', (event) => {
    settings.currency = event.target.value;
    saveSettings();
    renderAll();
    showToast('Валюту змінено.');
  });

  $('#themeToggle').addEventListener('click', () => {
    settings.theme = settings.theme === 'dark' ? 'light' : 'dark';
    saveSettings();
    setTheme();
  });

  $('#seedData').addEventListener('click', seedDemoData);

  $('#clearData').addEventListener('click', () => {
    if (!confirm('Очистити всі доходи, витрати та налаштування?')) return;
    transactions = [];
    settings = { ...defaultSettings };
    saveTransactions();
    saveSettings();
    setTheme();
    resetOperationForm();
    updateCategoryOptions($('#quickType'), $('#quickCategory'));
    renderAll();
    navigate('home');
    showToast('Дані очищено.');
  });
}

function submitQuickOperation() {
  const amount = Number($('#quickAmount').value);
  if (!amount || amount <= 0) {
    showToast('Введіть коректну суму операції.');
    return false;
```

```
  }

  transactions.push({
    id: Date.now(),
    amount,
    type: $('#quickType').value,
    category: $('#quickCategory').value,
    date: formatDateInput(),
    note: $('#quickNote').value.trim(),
  });

  saveTransactions();
  renderAll();
  showToast('Швидку операцію додано.');
  return true;
}

function init() {
  setTheme();
  $('#date').value = formatDateInput();
  updateCategoryOptions($('#quickType'), $('#quickCategory'));
  updateCategoryOptions($('#type'), $('#category'));
  bindEvents();
  renderAll();
}

init();
```