ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ УНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: "РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ КЕРУВАННЯ ФІНАНСАМИ ROCKETPENNY МОВОЮ PYTHON"

Виконав: студен 4 курсу, групи ПД–41
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)
Диндар А.В.
(прізвище та ініціали)
Керівник Трінтіна Н.А.
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ УНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти – "Бакалавр"
Спеціальність підготовки – 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____Негоденко О.В_____
"___"_____2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

_____ДИНДАР АЛЬБЕРТА ВАЛЕНТИНОВИЧА_____
(прізвище, ім'я, по батькові)

1. Тема роботи:___"Розробка мобільного додатку для керування фінансами PocketPenny мовою Python"_____
   Керівник роботи:___Трінтіна Н.А. к.т.н., доц._____
   (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
   Затверджені наказом вищого навчального закладу від" 24 " лютого 2023 року № 26
2. Строк подання студентом роботи" 1 " червня 2023 року
3. Вхідні дані до роботи:
   3.1 Інструменти розробки програмного забезпечення
   3.2 Інструменти обробки та візуалізації даних
   3.3 Методи обробки та візуалізації даних
   3.4 Науково–технічна література, пов'язана з розробкою
4. Зміст розрахунково–пояснювальної записки (перелік питань, які потрібно розробити).

4.1 Аналіз предметної області

4.2 Засоби програмної реалізації

4.3 Проектування та розробка програмного забезпечення

4.4 Тестування програмного забезпечення

4.5 Висновки

5. Перелік демонстраційного матеріалу.

5.1 Титульний слайд

5.2 Мета, об’єкт та предмет дослідження

5.3 Порівняння аналогів

5.4 Програмні засоби реалізації

5.5 Діаграми

5.6 Інтерфейс користувача

5.7 Апробація результатів дослідження

5.8 Висновки

6. Дата видачі завдання " 25 " лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково–технічної літератури	01.03.2023	Виконано
2	Аналіз та дослідження існуючих аналогів	12.04.2023	Виконано
3	Дослідження програмних засобів	16.04.2023	Виконано
4	Проектування системи	20.04.2023	Виконано
5	Створення програмного продукту	29.04.2023 – 15.04.2023	Виконано
6	Вступ, висновки, реферат	16.05.2023	Виконано
7	Розробка обов’язкових демонстраційних матеріалів	17.05.2023	Виконано
8	Попередній захист роботи та перевірка на плагіат	22.05.2023	
9	Здача роботи	01.06.2023	

Студент_____Диндар А.В._____

(підпис) (прізвище та ініціали)

Керівник роботи_____Трінтіна Н.А.___

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 76 с., 43 рис., 1 табл., 15 джерел.

Об'єкт дослідження – процес керування особистими фінансами.

Предмет дослідження – мобільний додаток для керування особистими фінансами.

Мета роботи – спростити процес керування особистими фінансами за рахунок використання мобільного додатку PocketPenny.

Методи дослідження – методи обробки, зберігання та відображення інформації, методи розробки програмного забезпечення, методи графічного відображення інформації.

Даний додаток може використовуватися будь–яким користувачем для персональних цілей, кращого розуміння власних фінансів та їх ефективного управління.

Галузь використання – додаток можуть завантажити користувачі смартфонів. Додаток допоможе підвищити навички фінансової грамотності.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ……..……...…….…..8
ВСТУП……..……...……9
1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ…….……...……...10
1.1 Поняття та огляд фінансового менеджменту.....12
1.2 Методи ведення бюджету.....14
1.2.1 Методи ведення бюджету в компаніях.....14
1.2.2 Методи ведення особистого бюджету ……..……..……15
1.3 Огляд найближчіх аналогів.....18
1.3.1 Додатки аналоги.....18
1.3.2 Переваги та недоліки.....28
2. АНАЛІЗ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ……...…..……..29
2.1 Засоби розробки.....29
2.1.1 Мова програмування Python для розробки додатків.....29
2.1.2 Середовище розробки PyCharm.....30
2.1.3 Огляд бібліотек та інструментів Python для мобільної розробки.....31
2.1.4 Огляд фреймворків для мобільної розробки.....32
3. ПРОЕКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ…….....33
3.1 Планування розробки ПЗ.....33
3.2 Діаграма використання додатку.....35
3.3 Діаграма класів.....37
3.4 Схема функціонування програми.....38
3.5 Проектування графічного інтерфейсу програми.....39
3.6 Реалізація функціональної частини.....45
3.6.1 Створення кнопок "Категорії" та "Рахунки".....45
3.6.2 Створення показників рахунків та балансу.....46
3.6.3 Створення кнопки "Діаграма".....47
3.6.4 Створення кнопок "Плюс" та "Мінус".....50
3.7 Проведення тестування.....54
ВИСНОВКИ……..……...……..55

ПЕРЕЛІК ПОСИЛАНЬ……..56
ДОДАТОК А……..……...57
ДОДАТОК Б…….……..……...…65

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

ПК – персональний комп’ютер

macOS – Macintosh Operating System

ПО – програмне забезпечення

США – Сполучені Штати Америки

Inc – Incorporated

ВСТУП

Актуальність теми. На сьогоднішній день все більше людей хочуть бути більш досвідченими та свідомими у своїх фінансах, тому приділяють увагу до особитстих витрат, прибутків та заощаджень. В такій відповідальній справі важко утримати все в голові, тому попит на програмні продукти для фінансових задач стає частішим.

У сучасному світі багато людей мають проблеми з керуванням своїми фінансами. Це призводить до того, що вони можуть опинитися в ситуації, коли їм бракує коштів на будь–що. Також через невміння керувати своїми фінансами люди часто відмовляються від речей, які вони хочуть придбати, або обмежені в подорожах та своєму духовному розвитку.

Також, невміння керувати своїми фінансами створює суттєву перешкоду на шляху досягнення своїх матеріальних мрій. Іноді люди, які не вміють керувати своїми фінансами, можуть опинитися у боргах і мати проблеми з їх погашенням, і в такому випадку про досягнення мрій мови йти не може.

Але що, якщо у людей буде додаток на смартфоні, який допоможе відстежувати і контролювати їх фінанси? Тоді вони зможуть легше досягнути мети щодо заощаджень і поліпшувати своє фінансове становище. PocketPenny – мобільний додаток, розроблений мовою програмування Python, який може стати надійним помічником в керуванні фінансами. Він дозволяє відстежувати витрати і прибутки, розподіляти кошти на різні категорії, встановлювати бюджет і контролювати його дотримання.

На відміну від додатків конкурентів на ринку PocketPenny відрізняється від них своєю унікальною функціональністю, що враховує досвід користувачів попередніх додатків, а також інноваційними можливостями, які роблять керування фінансами ще більш зручним та ефективним.

Таким чином, PocketPenny може бути ідеальним варіантом для тих, хто хоче навчитися керувати своїми фінансами і досягати фінансової стабільності.

Об’єкт дослідження – ефективне управління особистими фінансами та їх контроль.

Предмет дослідження – мобільний додаток для керування фінансами.

Мета роботи – підвищення ефективності додатку для спрощення ведення бюджету.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Проаналізувати переваги та недоліки існуючих аналогів.

2. Проаналізувати технічні засоби, що будуть використовуватися для розробки.

3. Спроектувати та розробити мобільний додаток на основі потреб користувачів.

4. Провести тестування додатку.

Галузь використання – додаток можуть завантажити користувачі смартфонів. Додаток допоможе підвищити навички фінансової грамотності.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття та огляд фінансового менеджменту

Фінансовий менеджмент – це процес планування, контролю та моніторингу фінансових ресурсів компанії, або фізичної особи. Фінансовий менеджмент є одним з ключових елементів управління будь–якої компанії, організації, а також особистих фінансів. Простіше кажучи – це те, що допоможе людині грамотно розпоряджатися своїми прибутками, та контролювати витрати. Для досягнення поставлених фінансових цілей фінансовий менеджмент є обов'язковим у цьому процесі.

Таким чином можна сказати, що фінансовий менеджмент може бути поділений на декілька видів, наприклад державний сектор, бізнес, чи особистий. Кожний із цих видів має свої особливості, і таким чином може бути ефективним для певної галузі. Також слід розуміти, що важливими для фінансового менеджменту є планування, аналіз, та контроль.

Фінансове планування – процес розробки планів з використання фінансових ресурсів організації на майбутнє, з урахуванням її цілей та потреб.

Фінансовий аналіз – це процес оцінки фінансової стійкості організації, що передбачає аналіз фінансових звітів та даних, а також оцінку фінансових ризиків.

Фінансовий контроль – процес моніторингу та контролю за використанням фінансових ресурсів, що передбачає визначення стандартів та процедур для контролю за бюджетом та фінансовою діяльністю організації.

Фінансовий менеджмент компанії – це процес управління фінансовими ресурсами компанії з метою забезпечення ефективності та досягнення поставлених фінансових цілей, наприклад збільшити капітал компанії, чи накопичити кошти для відкриття нового відділення. Основні аспекти фінансового менеджменту компанії включають бюджетування, управління кредитами та інвестиціями, фінансовий аналіз та планування.

Особистий фінансовий менеджмент – це процес управління особистими фінансами з метою досягнення фінансової стабільності та встановленої мети,

наприклад придбати автомобіль. Основні аспекти особистого фінансового менеджменту включають управління бюджетом, планування витрат та за бажанням інвестиції.

Для ефективного управління фінансами та ведення бюджету необхідна фінансова грамотність. Фінансова грамотність означає розуміння фінансових понять та принципів, здатність до аналізу фінансових даних та управління ресурсами. Фінансова грамотність є важливою для кожної людини.

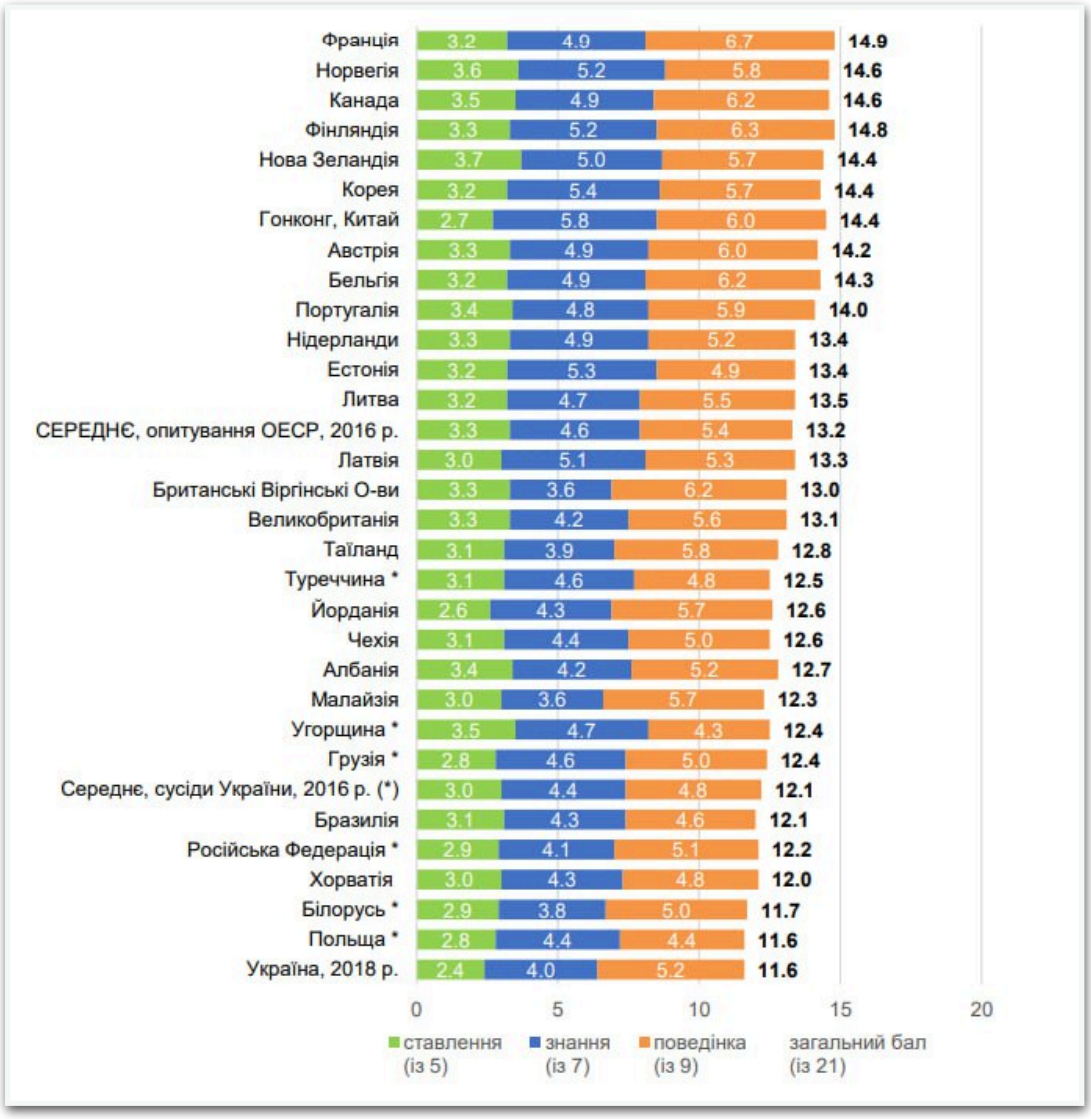


Рисунок 1.1 – Рейтинг країн з найбільш фінансово грамотним населенням

Це результати дослідження проведеного у 2018 році, і в якому середній рейтинг фінансової грамотності українців становить 11.6 пунктів з 21. Рейтинг встановлювався за показниками фінансової поведінки населення та рівня фінансових знань. Порівняно з 2017 роком ситуація покращилася на рівні зміни фінансової поведінки громадян. Саме фінансова поведінка громадян прямо впливає на розвиток фінансового ринку країни, рівня тіньової економіки та ставлення населення до реформ та ініціатив влади.

1.2 Методи ведення бюджету

1.2.1 Методи ведення бюджету в компаніях

Для ефективного управління бюджетними силами компанії потрібно добре знати і розуміти найвигідніші методи, які підходять під ситуацію або ціль компанії. Розглянемо методи ведення бюджету для компаній і корпорацій.

Бюджетування на основі активності (Activity–Based Budgeting) – цей метод базується на ідентифікації активностей та їх ресурсів, які необхідні для досягнення поставлених цілей компанії, та визначенні витрат на кожну активність. Цей метод дає можливість орієнтувати діяльність компанії на досягнення стратегічних цілей та успіху. Замість встановлення бюджетів на основі попередніх років або прогнозів, цей метод дозволяє визначити необхідні ресурси залежних від активностей для досягнення цілей компанії.

Бюджетування на основі нульової бази (Zero–Based Budgeting) – цей метод передбачає початок бюджетування з нуля, тобто всі витрати повинні бути виправдані і затверджені заново на кожен рік. Це допомагає уникнути непотрібних витрат і зосередитися на необхідних для компанії напрямках. Цей метод зазвичай використовують при формуванні бюджетів великих корпорацій, де велика кількість витрат, і вони є різного рівня важливості, і через це складно відслідковувати їх зміни з часом. Застосування даного методу дозволяє компанії краще розуміти вартість кожної програми та діяльності та допомагає приймати більш обґрунтовані рішення.

Бюджетування на основі прибуткової мети (Profit–Based Budgeting) – це метод ведення бюджету, коли прибуток є основним критерієм планування. Цей метод використовується для досягнення прибуткової мети, яку встановлює компанія на певний період часу. У цьому методі компанія прогнозує свій прибуток на майбутній період і планує свої витрати на основі цього прогнозу. Для досягнення цієї мети важливо ретельно вивчити фактори, що впливають на прибуток компанії, такі як продажі, ціни, конкуренція, інфляція та інші. Для використання методу бюджетування на основі прибуткової мети, компанії повинні мати достатньо інформації про свої фінансові показники, а також про фінансові ринки та тренди. Крім того, потрібно відстежувати витрати та їх відношення до прибутку, щоб забезпечити оптимальне використання ресурсів компанії. Це може допомогти компанії розробити стратегію збільшення прибутку, зниження витрат і оптимізації бізнес–процесів.

Бюджетування на основі ресурсів (Resource–Based Budgeting) – цей метод виходить з того, що ресурси компанії, такі як людські ресурси, матеріали та обладнання, є обмеженими, і передбачає розрахунок бюджету на основі цих обмежень. При використанні цього методу, компанія визначає обсяг ресурсів, які потрібні для реалізації певних проектів, або цілей, і встановлює бюджет на основі цих вимог. Наприклад, якщо компанія планує розширити свої маркетингові зусилля, вона визначить, скільки ресурсів потрібно для реклами та рекламної кампанії, і на цій основі встановить свій бюджет на маркетинг. Цей метод дозволяє компаніям більш точно співвідносити реальні витрати на ресурси, що необхідні для реалізації їхніх цілей. Він також дозволяє компаніям здійснювати керування своїми ресурсами більш ефективно, оскільки вони мають більш точне уявлення про те, які ресурси потрібні для досягнення поставленої мети, або результату.

1.2.2 Методи ведення особистого бюджету

Для досягнення поставленої мети можна використовувати вже існуючі методи ведення бюджету. Розглянемо декілька методів ведення бюджету для фізичної особи.

Класичний метод полягає в тому, щоб заздалегідь розпланувати свої витрати на місяць та записувати всі транзакції які ви робите, тобто і витрати, і прибутки. Для цього можна використовувати звичайний зошит, або створити таблицю у програмі Excel. В цьому методі не беруться до уваги непередбачувані витрати, ми записуємо їх як звичайні.

Метод 50/30/20 розроблений Сенатором з США Елізабет Воррен та базується на тому, щоб розподілити свій дохід на три частини: 50% – необхідні витрати, 30% – особисті витрати та розваги, 20% – заощадження. Таким чином покриваються всі необхідні витрати, залишаються кошти на розваги, і поповнюється скарбничка, тобто відкладення. Таким чином цей метод є простим та ефективним щоб задовільнити усі потреби та бажання.

Метод "конвертів" полягає в тому, щоб розподілити свій бюджет на кілька категорій, які можна самому створювати та редагувати. Далі розмістити кожну категорію в окремому "конверті". Наприклад, окремий "конверт" на продукти харчування, на оплату комунальних послуг, чи оренду житла, або на розваги, тощо. Після отримання доходу, наприклад отримали зарплатню, ви берете необхідну кількість грошей з кожного "конверту" на місяць та тільки цими грошима розраховуєтеся за свої витрати. Перевищувати ліміт кожного "конверта" неможна, інакше втрачається головна ідея і в якійсь категорії може бути нестача коштів.

Метод онлайн–банкінгу полягає в тому, щоб використовувати онлайн–банкінг для відстеження своїх доходів та витрат. Попередньо необхідно мати відкритий рахунок в банку. Далі потрібно завантажити програму банку на смартфон та відслідковувати свої фінанси, планувати бюджет. Ви будете отримувати сповіщення про транзакції на свій телефон, і таким чином нічого не упустите. Але в такому методі ми не відслідковуємо транзакції з готівкою.

Метод нульового бюджету полягає в тому, що кожна витрата повинна бути обґрунтована та має бути запланована з самого початку. Основна ідея полягає в тому, щоб використовувати збалансований підхід до витрат. Завдяки цьому методу ми можемо уникнути зайвих витрат та планувати їх з точністю.

Метод попереднього бюджетування полягає в тому, що витрати плануються на основі минулих витрат. Цей метод може бути корисним для тих, хто має стабільний дохід та витрати, і в такому випадку буде легко завжди триматися стабільності та спокійно відкладати на свою мрію, тим самим наближаючи її. Але у разі виникнення проблем з роботою, або при її зміні, цей метод не буде працювати, і необхідним кроком буде нове планування на основі склавшихся обставин.

Метод сімейного бюджетування полягає в тому, що сім'я планує витрати разом та використовує спільний бюджет, або створюють тимчасовий спільний бюджет на якусь витрату. Таким чином не обов’язково що обидва партнери відкладали у спільний "конверт". Замість цього можна притримуватись способу виділяти кошти навпіл, кожен зі свого бюджету, на якусь спільну покупку, чи поїздку. Цей метод допомагає уникнути зайвих витрат та сприяє співпраці в сім’ї, або пари.

Метод бюджетування за цілями, також відомий як метод "розумних" цілей, полягає в тому, що перед тим як складати бюджет, людина визначає свої фінансові цілі на певний період часу. При цьому важливо, щоб ці цілі були конкретними, досяжними, та часово обмеженими. Особливістю цього методу є те, що він дозволяє людині зосередитися на цілі і розставляти пріоритети для досягнення бажаних результатів.

Методи ведення особистого бюджету включають різні підходи до контролю та управління фінансами. Деякі методи базуються на ручному запису доходів та витрат, інші використовують автоматизовані програми та додатки для відстеження фінансових операцій. Вибір методу залежить від індивідуальних потреб та зручності для користувача.

1.3 Огляд найближчіх аналогів

1.3.1 Додатки аналоги

– Monefy

Це додаток для керування особистими фінансами, створений українським розробником Андрієм Барановим. В ньому зручно вести облік своїх доходів та витрат. Додаток простий у використанні та інтуїтивно зрозумілий, що дозволяє швидко і легко ввести інформацію про всі транзакції. В додатку є можливість придбати Про версію за певну суму. В ній користувачу стануть доступні деякі додаткові функції, яких немає у безкоштовній версії програми.

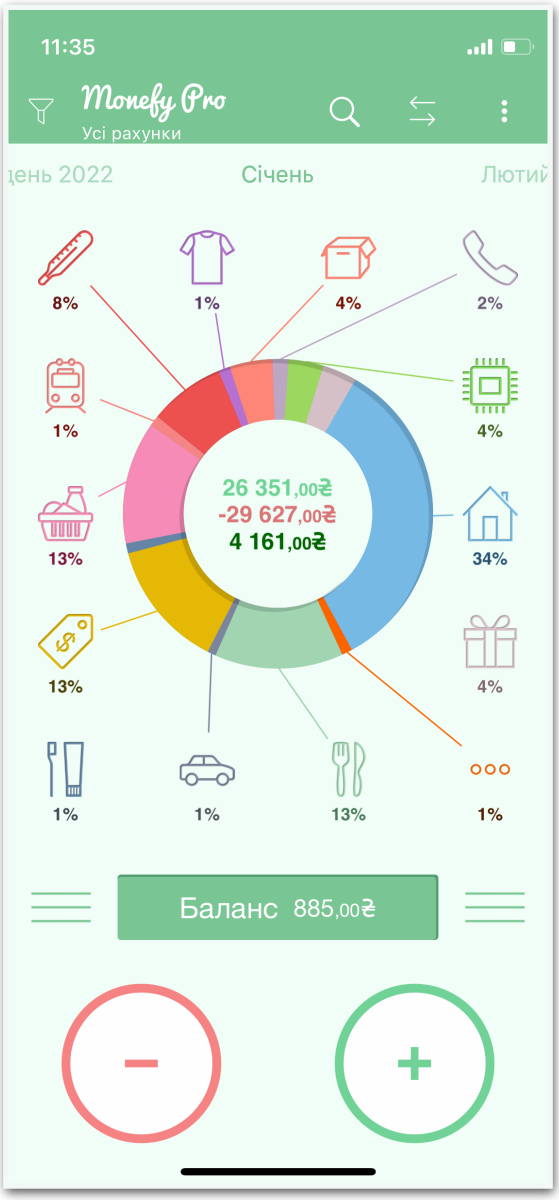


Рисунок 1.2 – Скріншот головного екрану Monefy Pro

Також він має категорії витрат, які можна налаштувати на основі індивідуальних потреб. Наприклад, витрати можна розділити на їжу, транспорт, одяг, платежі, дім, подарунки і т.д. Крім того, користувачі можуть створювати власні категорії витрат, або видаляти існуючі.

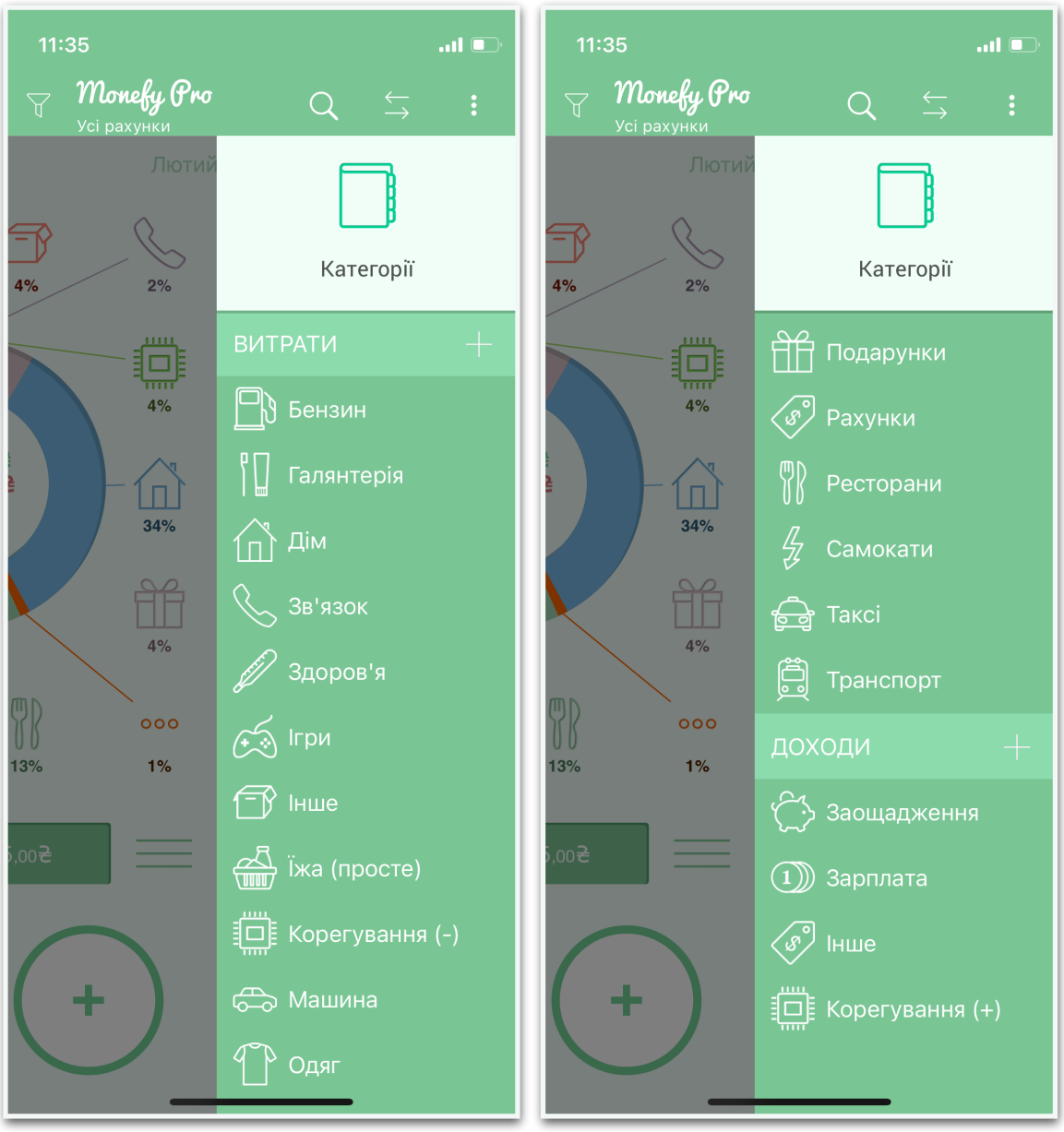


Рисунок 1.3 – Скріншот категорій у додатку

Додаток показує графіки та статистику витрат, де одразу видно, яка категорія скільки відсотків займає від загального кола. Це дозволяє користувачам легко відстежувати свої витрати та бачити, де вони витрачають більше грошей.

Також додаток також має функцію бекапу, що дозволяє зберегти дані в обліковому записі Google Drive або Dropbox. Ця функція точно не є зайвою, бо так дані користувача завжди будуть доступні, навіть після зміни девайсу.

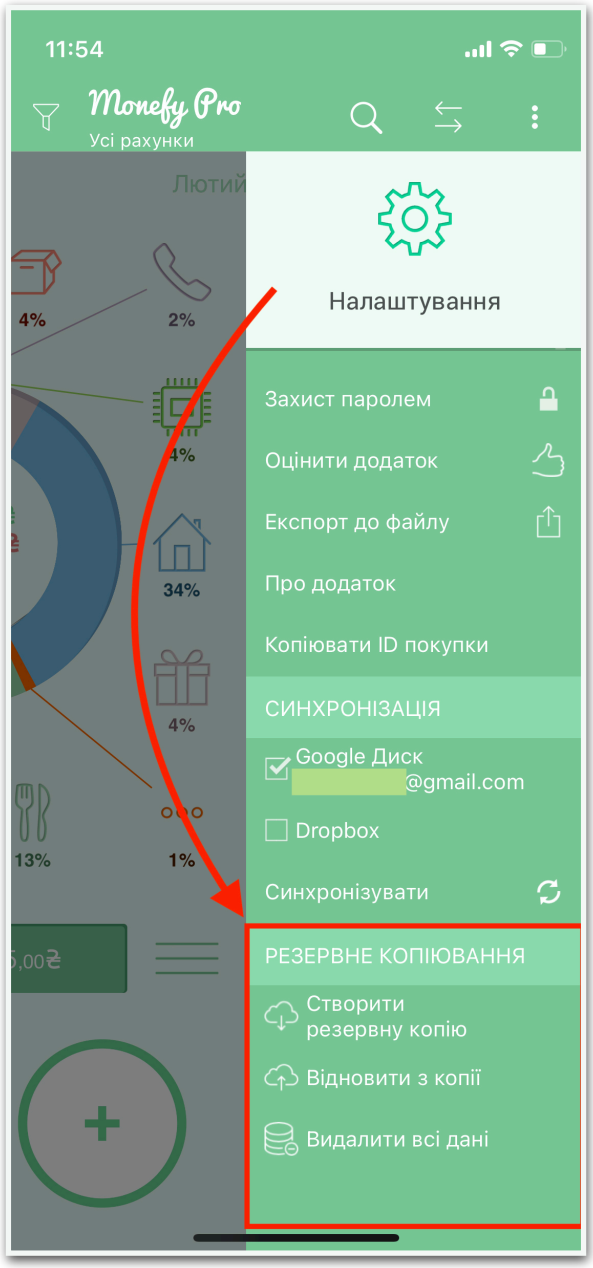


Рисунок 1.4 – Скріншот можливості резервного копіювання

Додаток доступний на платформах Android та iOS, і доступний для завантаження з Play Market та App Store.

– CoinKeeper: облік витрат

Ще одна програма для контролю фінансів, розробником якої є компанія "Disrapp", яка на момент створення додатку, а саме 2011 рік, знаходилася у місті Львів. Реліз додатку у магазини для користувачів відбувся у 2012 році. На відміну від попереднього додатку, CoinKeeper має інтуїтивний інтерфейс та більше налаштувань. В додатку також присутні категорії, функції трекінгу прибутків та витрат.

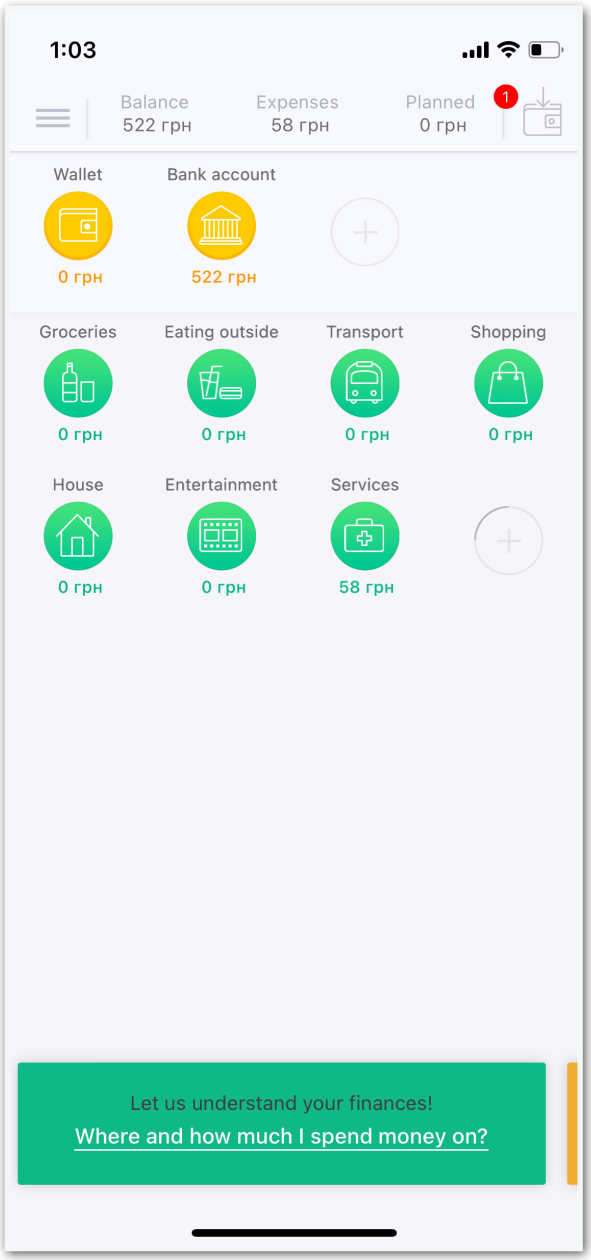


Рисунок 1.5 – Скріншот головного екрану CoinKeeper

На головному екрані ми маємо категорії, поточний баланс та деякі кнопки налаштувань, такі як додати категорію та додати фінансове середовище.

Програма також має платні функції, доступ до яких можна отримати заплативши певну суму. На відміну від Monefy, додаток веде агресивну рекламу своєї платної версії, що насправді дратує користувача і відштовхує. При першому запуску нас зустрічають купа банерів щодо платної версії.

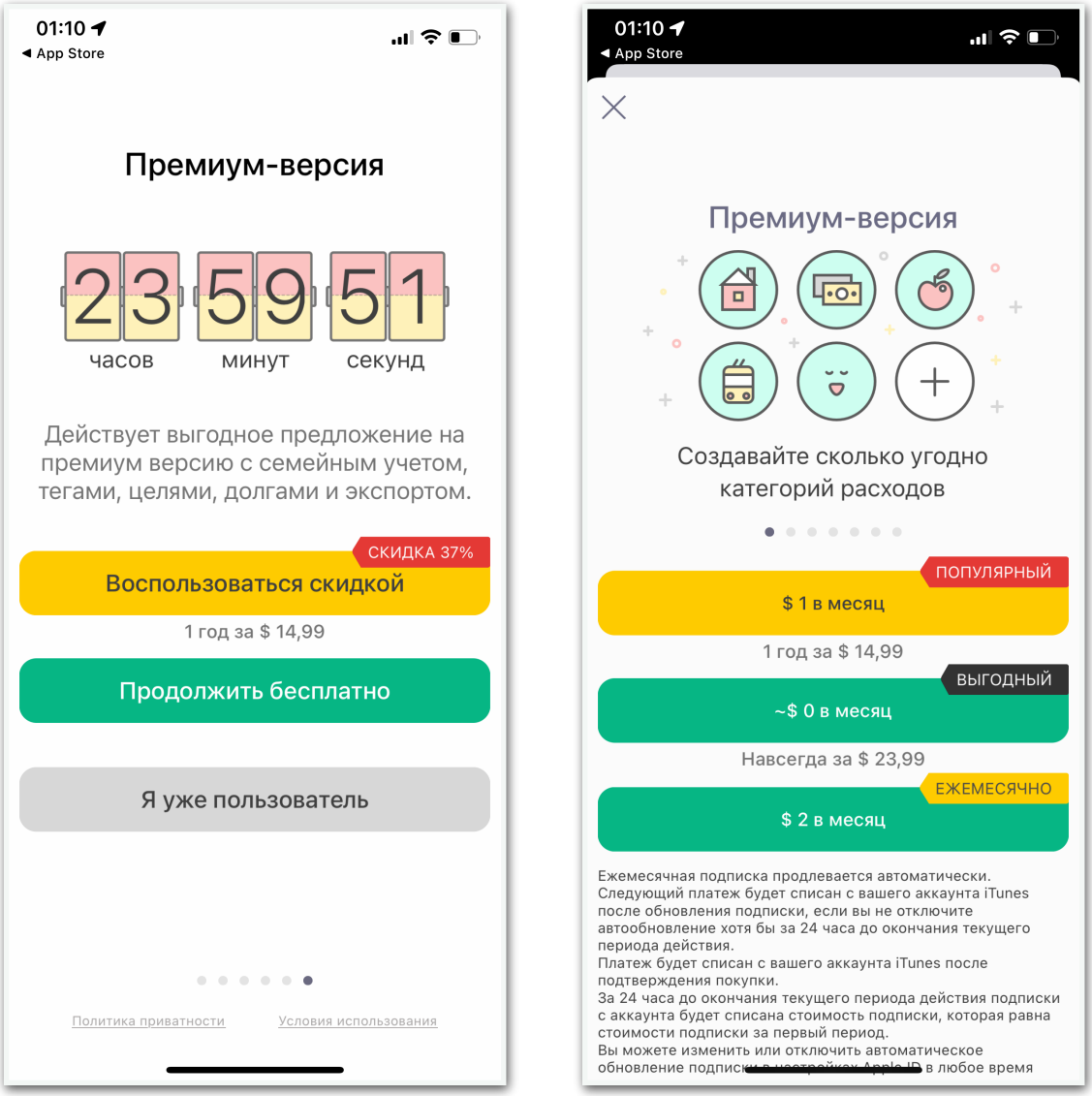


Рисунок 1.6 – Скріншот пропозиції платної версії програми

Додаток має загальний графік, де користувач може побачити всі свої транзакції у вигляді таблиці. Але є "але" – це також платна функція. Резервне копіювання також є платною функцією, що вже ставить під сумнів вибір користувача в бік саме цього додатку для особистого фінансового менеджменту.

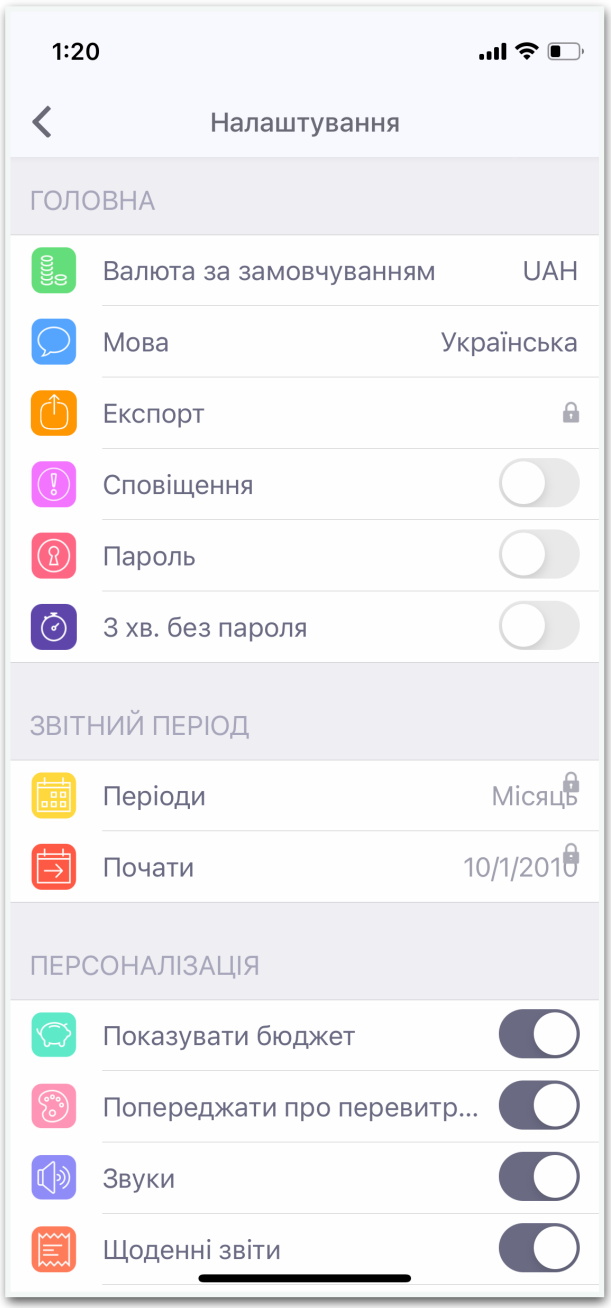


Рисунок 1.7 – Скріншот налаштувань програми та платні функції з іконкою замочку

Варто віддати увагу можливості поставити пароль на додаток. Таким чином дані користувача про його транзакції та особисті рахунки будуть недоступні стороннім особам. Присутня функція автоблокування додатку через певний час, заданий користувачем, на випадок якщо CoinKeeper залишиться на довгий час без нагляду власника.

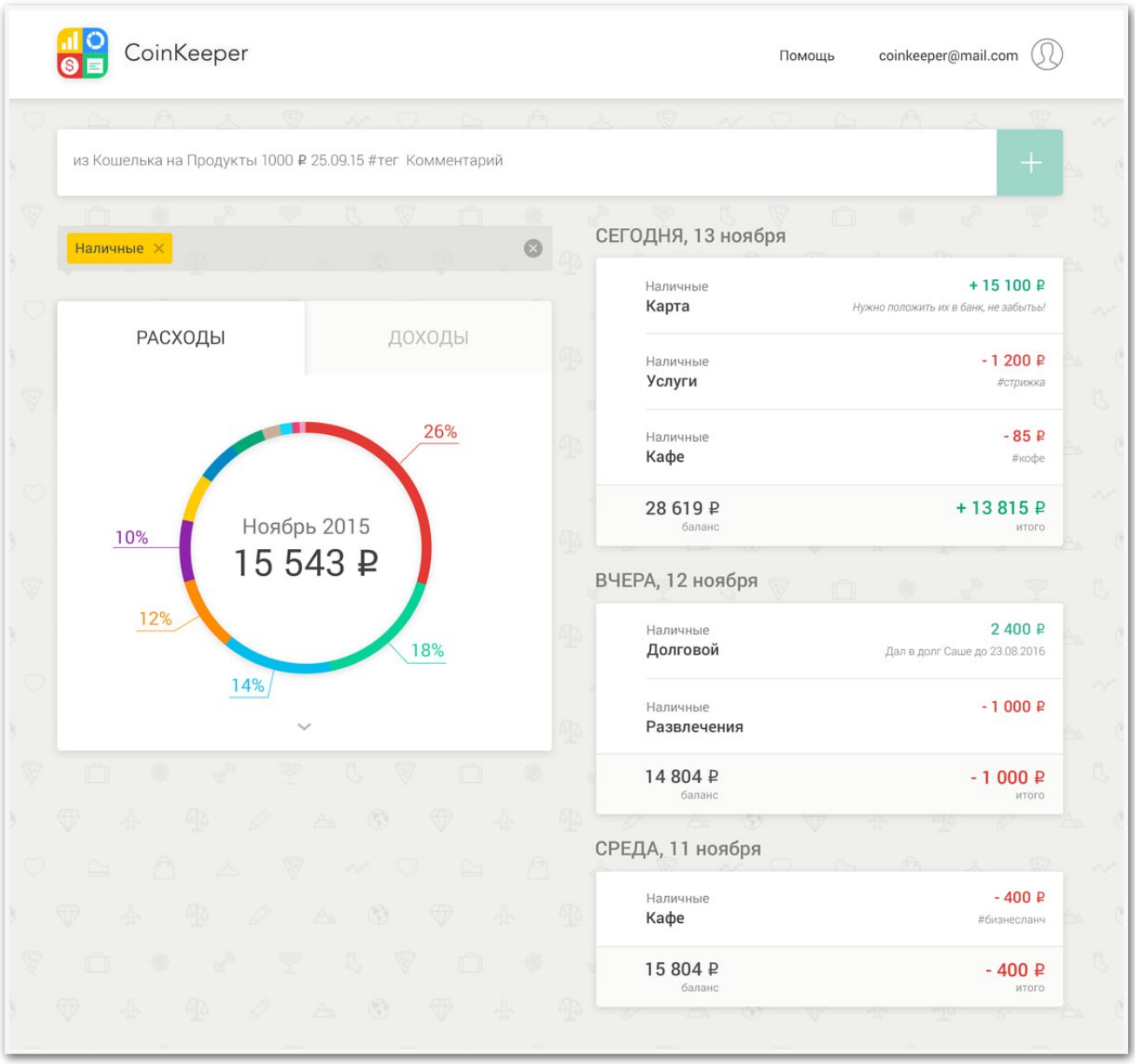


Рисунок 1.8 – веб версія CoinKeeper

Додаток CoinKeeper доступний на платформах Android та iOS, а також є веб–версія, що є перевагою і дає можливість розробникам охопити більшу аудиторію. Нажаль, наразі веб версія CoinKeeper недоступна на невизначений термін.

– Money manager, expense tracker

Це мобільний додаток для ведення фінансового обліку, розроблений компанією Realbyte Inc. Реліз додатку відбувся у 2014 році і одразу себе зарекомендував та має високу оцінку в App Store. Програма має інтуїтивно зрозумілий інтерфейс, через це нею легко користуватися та відстежувати свої фінансові операції. Цікавою функцією є додавання фото до кожної витрати чи прибутку, на відміну від програм аналогів.

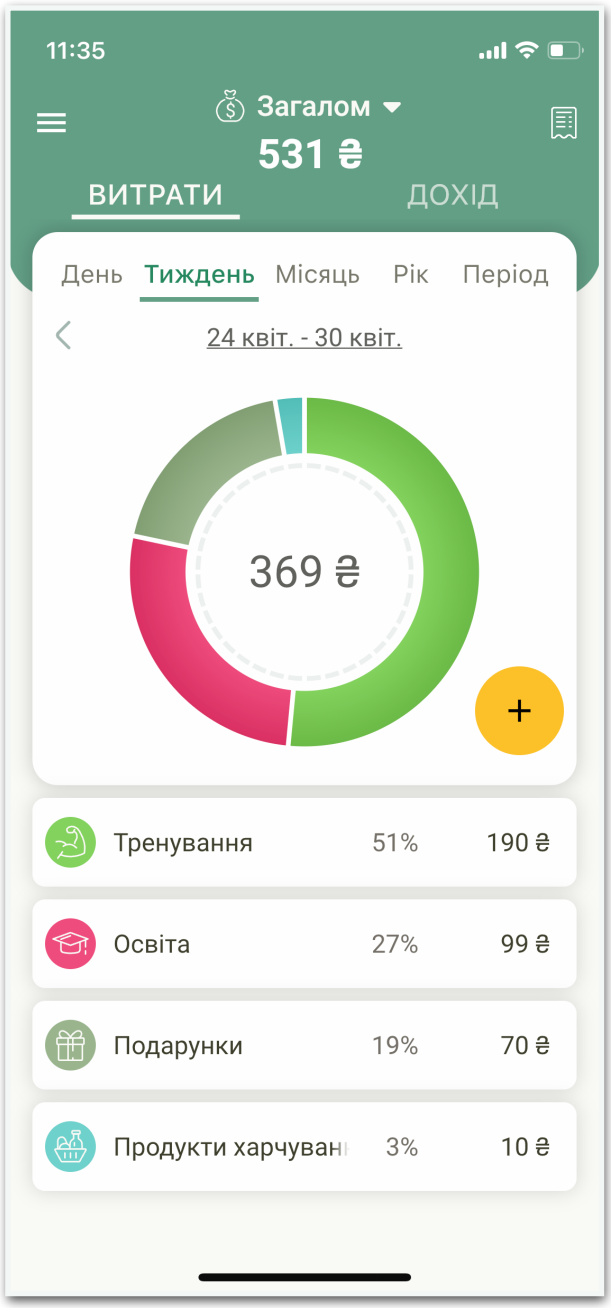


Рисунок 1.9 – Скріншот головного екрану Money manager, expense tracker

Основні функції додатку включають створення категорій витрат і доходів, налаштування повторюваних транзакцій, відстеження балансу рахунків, генерацію звітів та графіків транзакцій, встановлення бюджетів і багато іншого. Також користувач може експортувати свої транзакції у форматі програми Excel.

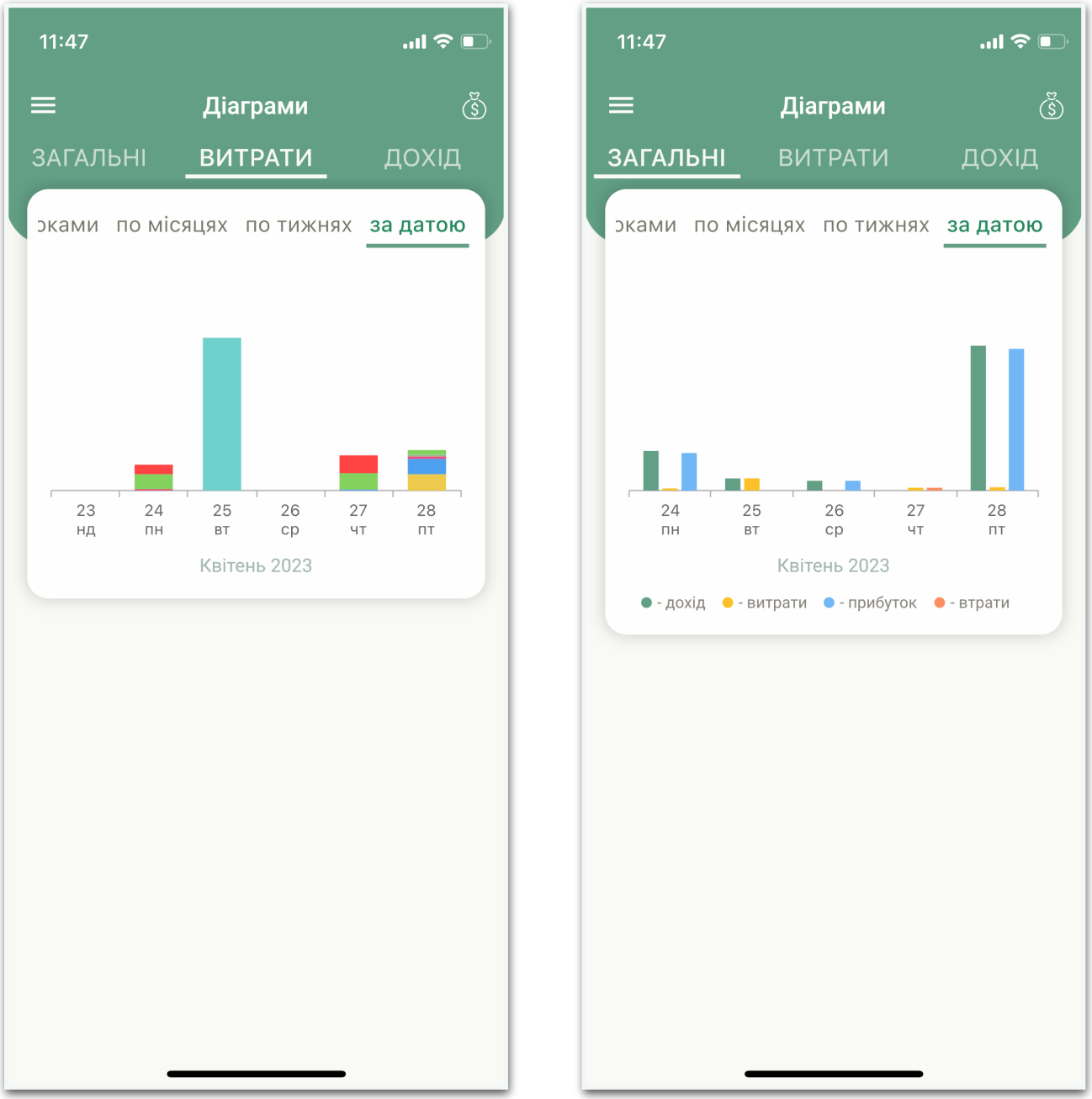


Рисунок 1.10 – Діаграми витрат та загальна діаграма

Варто звернути увагу на функцію автоматичної зміни теми додатку на денну тему, або нічну тему. Це налаштування залежить від поточної теми на вашому смартфоні. Дана функція є безкоштовною, що не скаже про аналоги. В Money manager, expense tracker корисною функцією також є встановлення пароля на додаток.

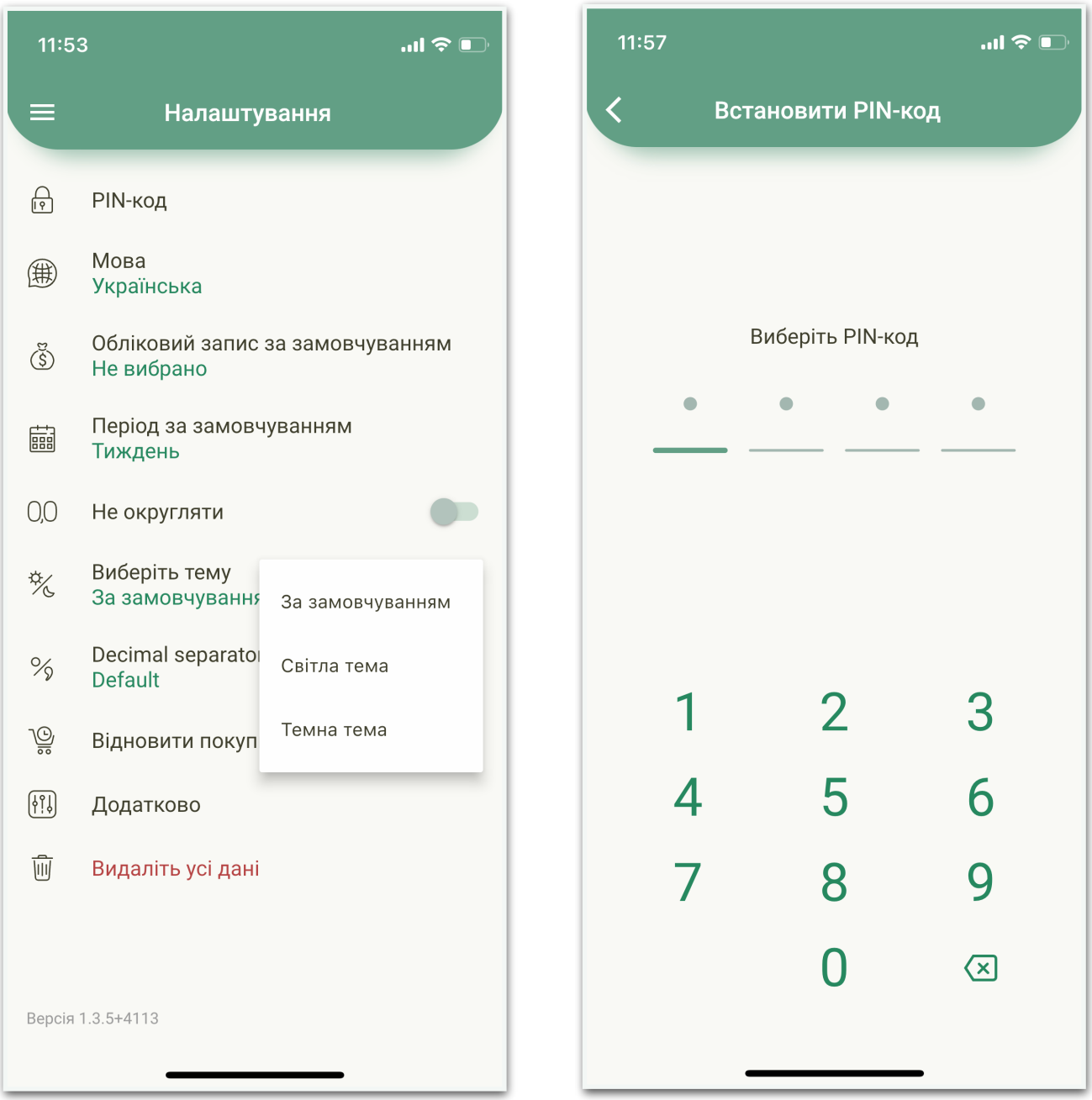


Рисунок 1.11 – Скріншот налаштувань програми та встановлення пін–коду

Додаток є кросплатформеним та працює на таких операційних системах як Android, iOS та на ПК з операційною системою Windows та macOS.

1.3.2 Переваги та недоліки

Таблиця 1.1 – Результати аналізу розглянутих мобільних додатків фінансового менеджменту.

Назва	Переваги	Недоліки
Monefy	● Інтуїтивний інтерфейс ● Резервна копія на Google Disk ● Немає нав'язливої реклами ● Можливість придбати Про версію	● Застарілий дизайн ● Немає автозміни теми денна/нічна ● Найкращі функції тільки у Про версії
CoinKeeper	● Приємні звуки при роботі з програмою ● Гарні анімації іконок ● Таблиця та дашбоард ● Пароль на додаток та автозамикання	● Не юзерфрендлі інтерфейс, новачок може заплутатись ● Майже весь функціонал платний ● Нав'язлива реклама Про версії
Money manager, expense tracker	● Інтуїтивний інтерфейс ● Пароль на додаток ● Автозміна денної/нічної теми додатку ● Великий вибір мов програми ● Весь функціонал безкоштовний	● Не юзерфрендлі інтерфейс, новачок може заплутатись ● Присутні фрізи в додатку
PocketPenny	● Інтуїтивний інтерфейс ● Немає реклами ● Діаграма та статистика	● Немає паролю на додаток ● Відсутня зміна теми додатку

Аналіз особливостей програм аналогів в категорії фінансового трекінгу дозволяє сформулювати вимоги до майбутнього програмного продукту. Програма PocketPenny має бути реалізована для мобільної платформи та мати наступні функції:

- можливість проводити трекінг особистих фінансів
- можливість працювати з категоріями
- можливість відстежувати поточні баланси
- україномовний інтерфейс

2. АНАЛІЗ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Засоби розробки

2.1.1 Мова програмування Python для розробки додатків

Для розробки додатку PocketPenny було обрано мову програмування Python оскільки вона є загального призначення, а також відома своєю простотою, ефективністю та читабельністю.

Для мови програмування Python існує велика кількість бібліотек і фреймворків, що дає змогу розробляти та створювати додатки на високому рівні абстракції та скорочує час розробки. Також Python підходить для операційної системи Android, яка є дуже розповсюдженою. Ця мова є потужною та простою у використанні навіть якщо розробник має невеликі навички програмування. Також Python є популярним для роботи зі штучним інтелектом та машинним навчанням.

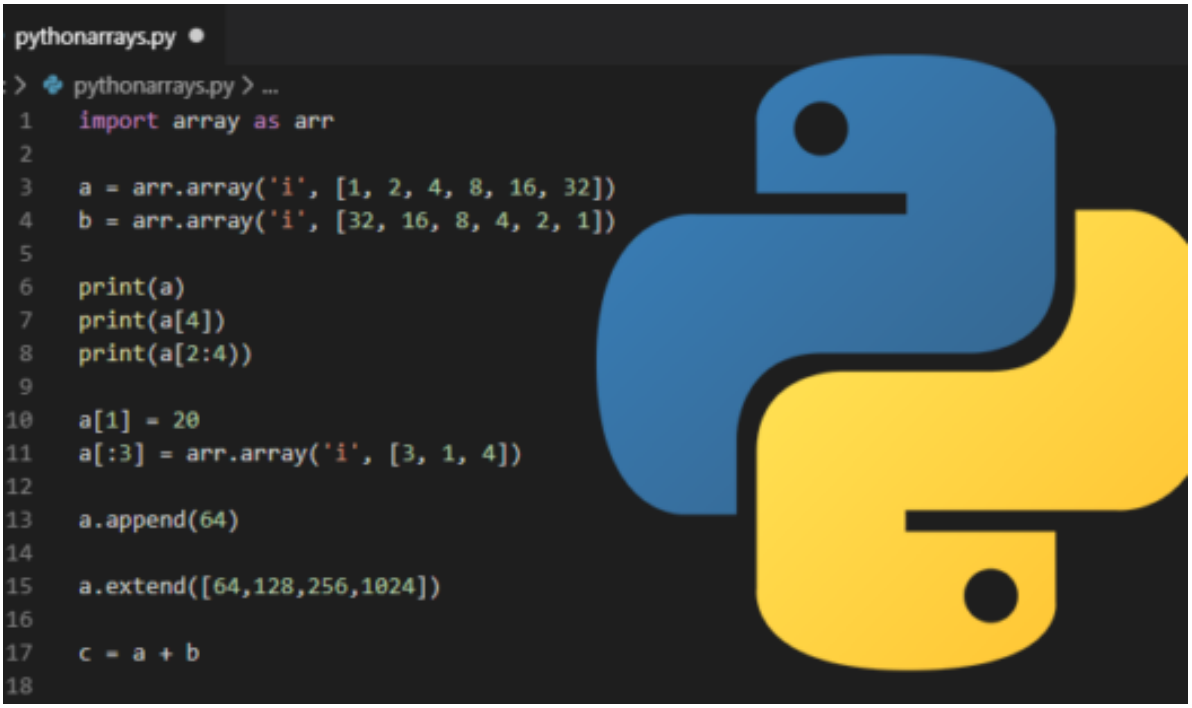


Рисунок 2.1 – Приклад коду та логотип мови Python

2.1.2 Середовище розробки PyCharm

Для того, щоб забезпечити ефективну та комфортну розробку додатків мовою Python, важливо вибрати оптимальне середовище розробки.

PyCharm – це інтегроване середовище розробки (IDE) для мови програмування Python, створене компанією JetBrains. Воно забезпечує розширені можливості для розробки програмного забезпечення на Python, включаючи підтримку керування версіями, автоматичну перевірку коду, автодоповнення та інші корисні функції.

Для програмування мовою Python найбільш раціональним вибором є використання PyCharm, оскільки ця IDE розроблена тими ж людьми, що і мова програмування Python, тому вона має максимально можливі функції та оптимальний інтерфейс для роботи з цією мовою.

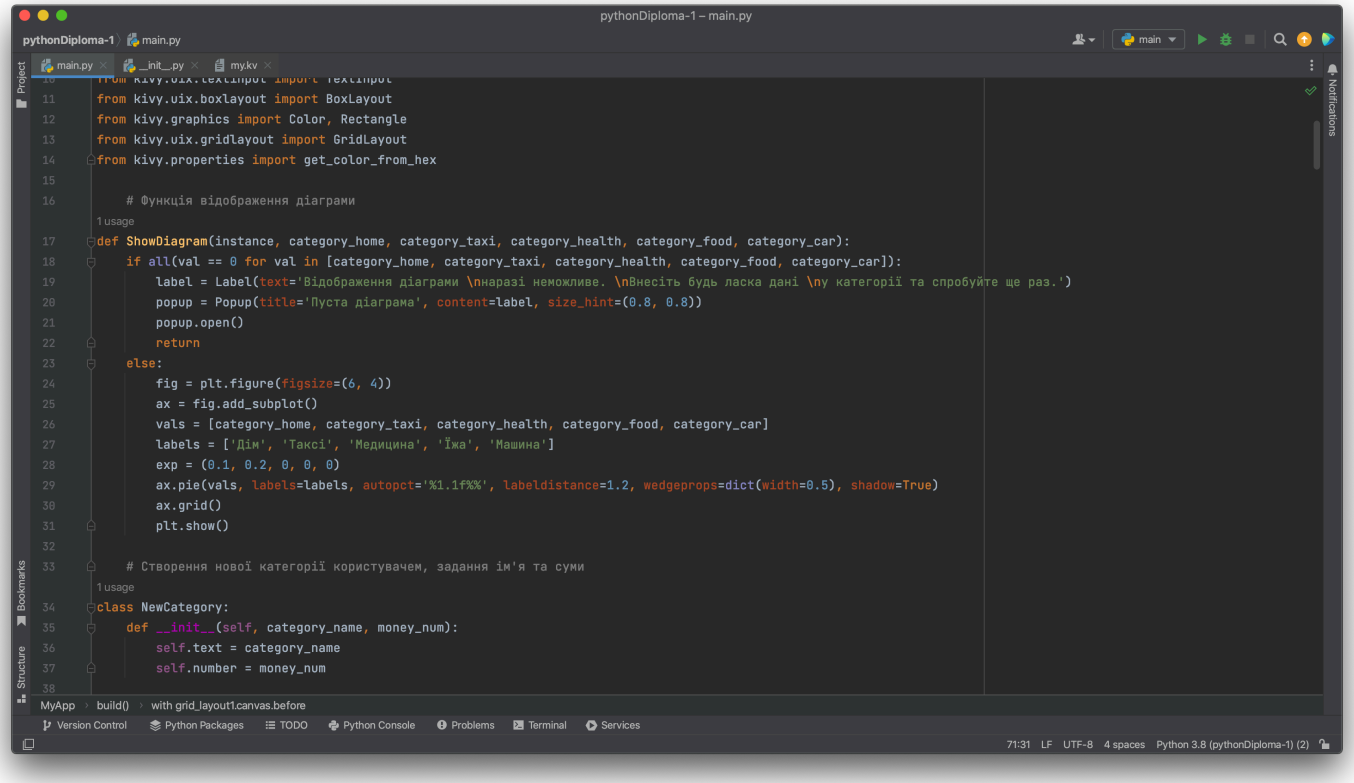


Рисунок 2.2 – Робоче середовище в PyCharm

2.1.3 Огляд бібліотек та інструментів Python для мобільної розробки

Для досягнення максимальної ефективності у розробці додатків мовою програмування Python, корисно використовувати додаткові бібліотеки та інструменти. Вони допомагають збільшити можливості та потенціал розробки.

Важливим інструментом для розробки на Python є модуль functools, який надає різноманітні функції для роботи з функціями. Наприклад, функції partial і reduce, які допомагають зменшити кількість коду та полегшити роботу з функціями. Функція partial дозволяє частково задати аргументи для функції, тоді як reduce здійснює послідовне застосування функції до елементів списку. Такі інструменти дозволяють збільшити швидкість розробки та зменшити кількість коду, що потрібно писати.

Для візуалізації даних у Python часто використовують бібліотеку matplotlib. Вона надає можливість будувати різні графіки та діаграми, що дозволяє аналізувати дані в зручному візуальному форматі, також є дуже гнучкою та налаштовуваною, що дозволяє розширювати її можливості за допомогою різноманітних налаштувань та параметрів. Matplotlib має велику кількість прикладів та документацію, що допомагає при розробці, бо є можливість швидко знайти потрібну підказку та інформацію.

Рисунок 2.3 – Логотипи бібліотек Matplotlib та Functools відповідно

2.1.4 Огляд фреймворків для мобільної розробки

Фреймворки для мобільної розробки – це комплекси програм, які включають в себе інструменти для розробки графічного інтерфейсу, роботи з базами даних, мережевої взаємодії та багато іншого. З їх допомогою розробники можуть швидко створювати мобільні додатки для різних платформ з використанням спільного коду.

Kivy – це кросплатформений фреймворк для розробки мобільних та настільних додатків з відкритим кодом, який використовує Python. Він надає інструменти для розробки графічного інтерфейсу користувача, роботи з мультимедіа, створення анімації та інші функціональні можливості. Тому, Kivy можна розглядати як фреймворк для розробки мобільних додатків з використанням Python.

Інший популярний фреймворк – це Flutter, що базується на мові програмування Dart. Він дозволяє створювати мобільні додатки для Android, iOS та інших платформ. Особливість Flutter полягає у тому, що він використовує власну рендерингову машину, що дозволяє побудувати швидкі та високоякісні інтерфейси користувача.

Рисунок 2.3 – Логотипи фреймворків Kivy та Flutter відповідно

3. ПРОЕКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Планування розробки ПЗ

Під час розробки будь–якого програмного продукту необхідно відстежувати список задач, які потрібно виконати. Для кращої організації роботи кожному учаснику проекту слід мати доступ до dashboard, де є відображення завдань, включаючи активні, завершені, плановані тощо. Популярною платформою для цього є KanbanFlow.

KanbanFlow – це онлайн–інструмент для управління проектами, який базується на методології Канбан. Він надає зручну дошку з колонками, що дозволяє візуалізувати та керувати задачами у реальному часі. Деякі переваги KanbanFlow включають гнучкість налаштування проектів, можливість спільної роботи команди та зручний спосіб відстежування прогресу завдань.

Перевагами цього інструменту є:

- спільний доступ та робота над тікетами
- гнучкість налаштувань проекту
- простий та зрозумілий інтерфейс

Також можна створювати типи задача, або використовувати створені за замовчуванням:

- заплановані задачі, тобто to–do
- задачі, які можуть бути в майбутньому, тобто future
- задачі в процесі, тобто in progress
- завершені задачі, тобто done

Як ми бачимо статусів задач за замовчуванням вже достатньо для примітивної організації роботи. Але було прийняте рішення додати ще статус Future, тобто те, що можливо буде додано у майбутньому при оновленнях додатку. Цей пункт було додано, оскільки на основний функціонал був виділений час, але для реалізації усіх функцій його бракувало.

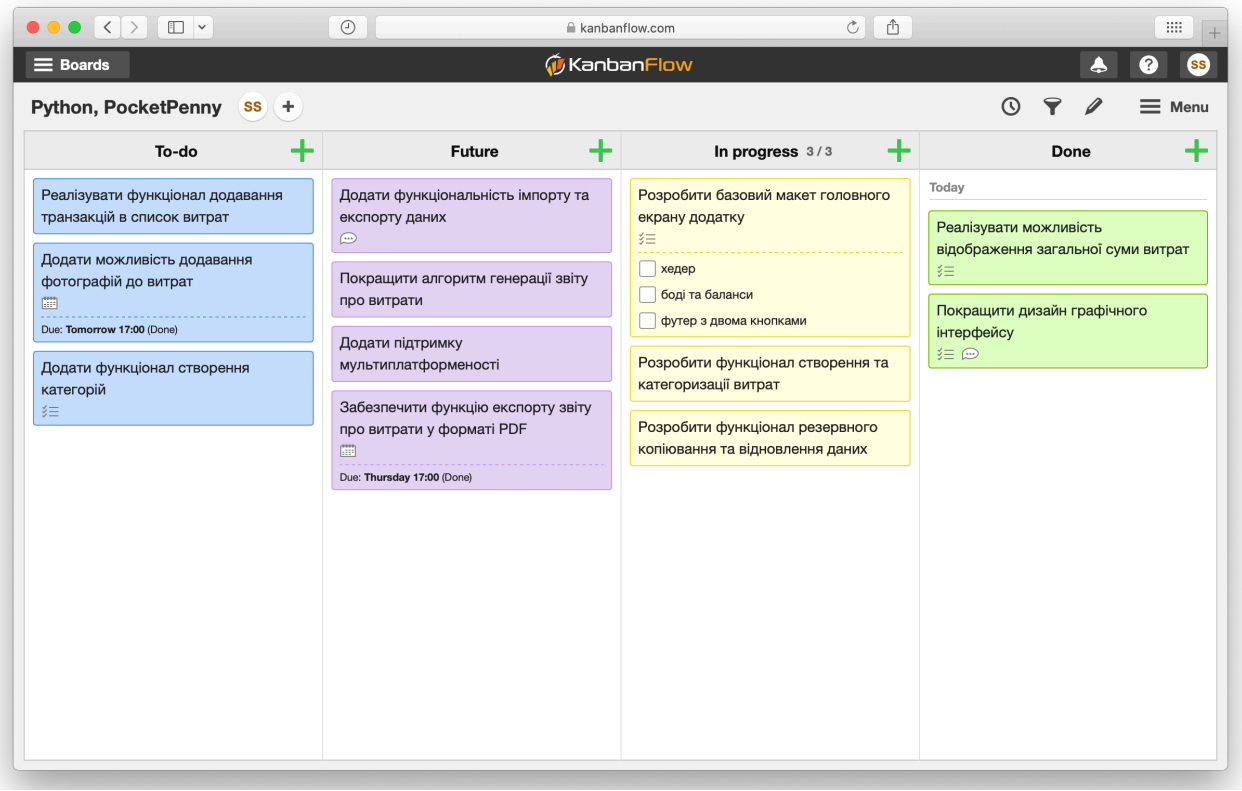


Рисунок 3.1 – Дашбоард на сайті KanbanFlow для створення проекту

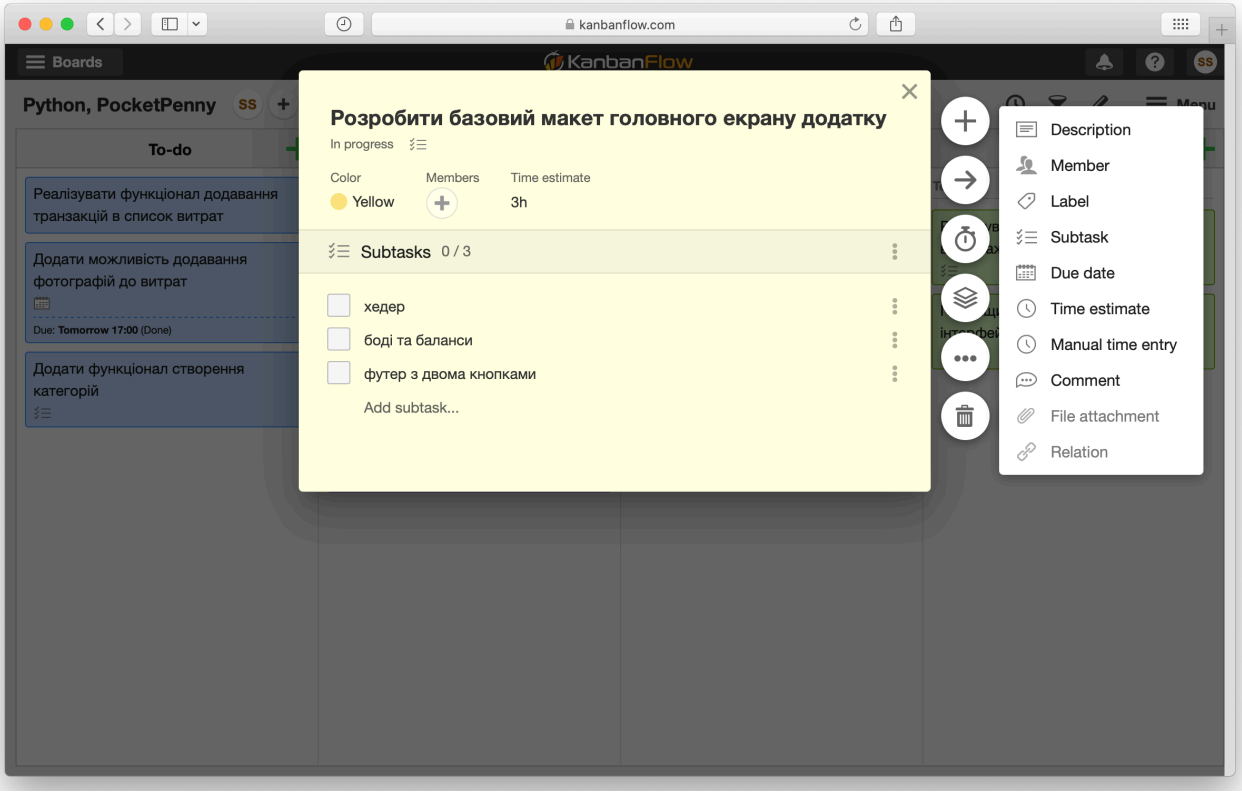


Рисунок 3.2 – Тікет створеної задачі зі статусом In Progress на сайті KanbanFlow

3.2 Діаграма використання додатку

Діаграма використання – діаграма яка відображає взаємодію між актором (користувачем), та нашим додатком. Ця діаграма вказує, як користувачі взаємодіють з різними функціями та можливостями додатку, а також якими даними вони обмінюються з ним.

Актор є особою або сутністю, що взаємодіє з системою, зазвичай представленим користувачем. Однак, іноді актором може бути інша система, яка знаходиться поза контекстом системи. Для нашої діаграми використання ми обрали одного актора – користувача, який буде взаємодіяти з нашим додатком. Враховуючи специфіку програми, використання інших акторів не передбачено, оскільки програма не передбачає багатокористувацького середовища. Важливою особливістю є те, що всі дані зберігаються локально на пристрої користувача, що забезпечує безпеку зберігання даних.

Під час створення діаграми використання важливо мати інформацію не тільки про акторів, але й про варіанти використання системи. Таким чином, можемо описати використання додатку:

Використання PocketPenny на головному екрані:

- перехід до категорій
- перехід до рахунків
- перехід до внесення прибутку
- перехід до внесення витрати
- перехід до перегляду діаграми (якщо є дані для відображення)
- перегляд поточного балансу
- перегляд загальної суми витрат
- перегляд загальної суми прибутків

Використання PocketPenny в категоріях:

- перехід до створення нової категорії
- перегляд існуючих категорій
- повернення назад (тап за межі вікна категорій)

Використання PocketPenny при від’ємній (баланс мінус) операції:

- перехід до внесення суми витрати
- перехід до вибору категорії, на яку прийшлася витрата
- повернення назад (тап за межі вікна категорій)

Використання PocketPenny при позитивній (баланс плюс) операції:

- перехід до внесення суми прибутку
- перехід до вибору категорії, з якої надійшов прибуток
- повернення назад (тап за межі вікна категорій)

Використання PocketPenny при перегляді діаграми:

- перегляд кругової діаграми
- перегляд витрат кожної категорії у % співвідношенні
- повернення назад (закриття вікна перегляду діаграми)

За допомогою зібраних пунктів та інформації, ми можемо побудувати діаграму використання, яка відображатиме способи взаємодії користувача (актора) з додатком.

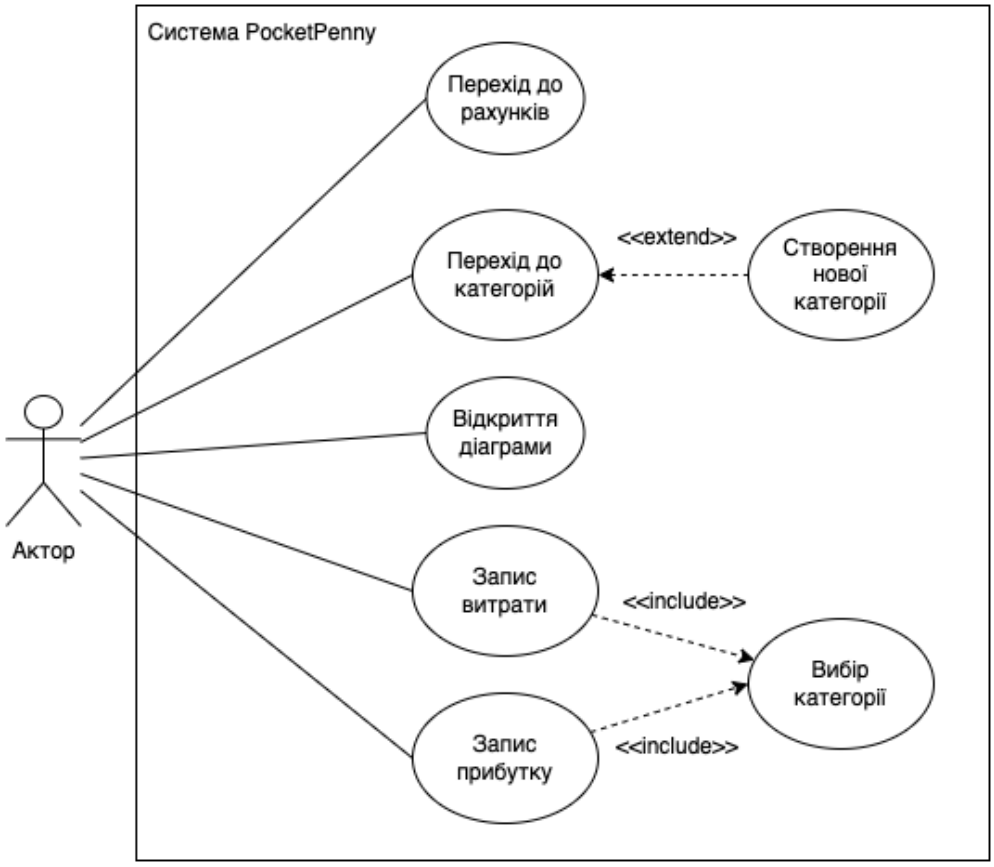


Рисунок 3.3 – Use case діаграма

3.3 Діаграма класів

Діаграма класів дозволяє відобразити всі класи, що використовуються у процесі створення додатку. Під час розробки було створено два класи, кожен з яких виконує свою роль у функціонуванні програмного засобу. На малюнку 3.4 представлені зв’язані між собою класи.

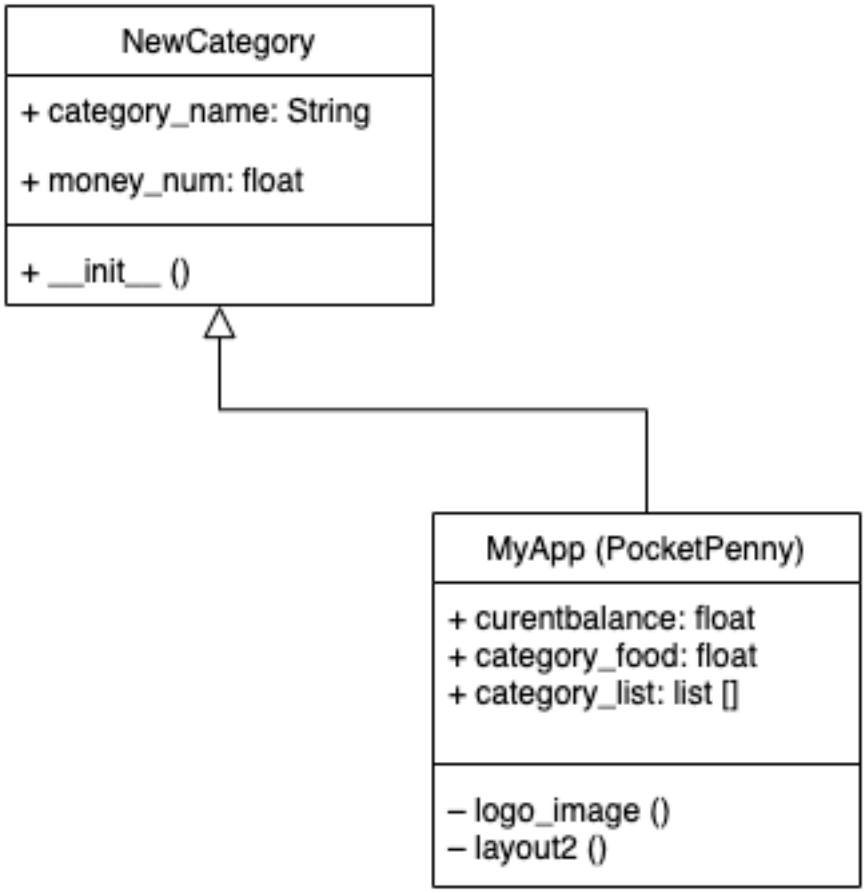


Рисунок 3.4 – Діаграма класів

Клас **MyApp (PocketPenny)** – основний клас додатку, в якому знаходяться функціональні можливості та властивості програми.

Клас **NewCategory** – нащадок класу "MyApp (PocketPenny)", який відповідає за створення нової категорії у списку категорій.

3.4 Схема функціонування програми

Схема функціонування ілюструє основні елементи та взаємодію екранів у програмі PocketPenny. Розберемо які елементи включає в себе головний екран:

- кнопка "Категорії"
- кнопка "Рахунки"
- кнопка "Діаграма"
- кнопка "Плюс"
- кнопка "Мінус"

Після натискання на кнопку "Категорії" ми також розглянемо елементи, які знаходяться на екрані "Категорії":

- кнопка "Нова категорія"

Також важливо зазначити контекстні кнопки, які з'являються при взаємодії з кнопками "Плюс" та "Мінус":

- кнопка "Далі"
- кнопки для обрання "Категорії"

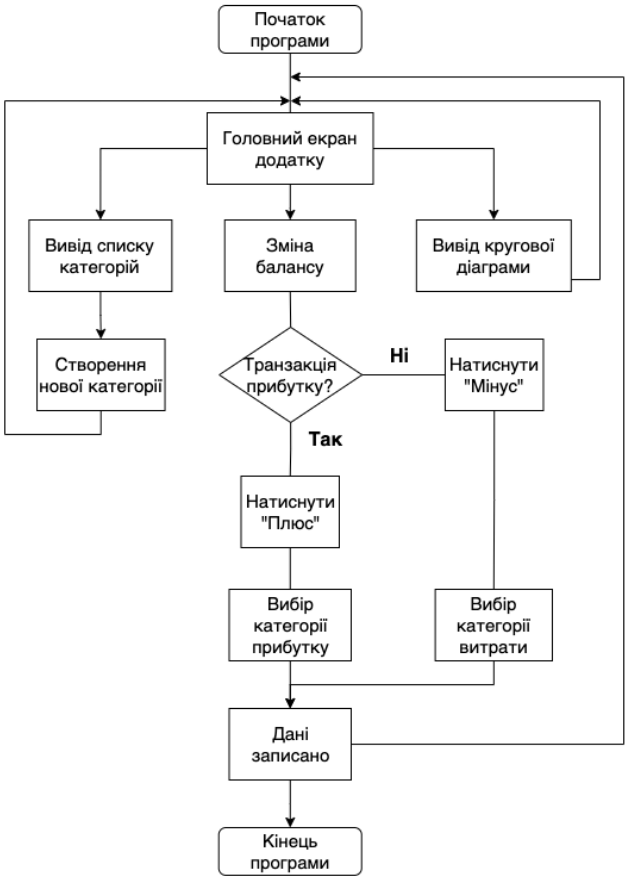


Рисунок 3.5 – Схема функціонування програми

3.5 Проектування графічного інтерфейсу програми

На цьому етапі розробки головною метою є створення зручного та привабливого інтерфейсу, що сприятиме інтуїтивній навігації та взаємодії користувача з додатком. Перш ніж розпочати проектування графічного інтерфейсу, було визначено основні компоненти, які повинні бути доступні на головному екрані. Ці компоненти включають:

- У верхній частині екрану має розташовується логотип та назва додатку PocketPenny, щоб забезпечити легку ідентифікацію програми. Також у хедері мають міститися дві кнопки у правій верхній частині екрану для зручного доступу до них.
- Показники фінансового стану мають бути розміщені на головному екрані, такі як поточний баланс, баланс витрачених коштів та баланс прибутків. Ці показники відображаються у вигляді числових значень.
- Кнопка "Мінус" призначена для введення витрат, а кнопка "Плюс" – для введення доходів. Ці кнопки мають дозволяти користувачу швидко і зручно внести фінансові операції у додаток.
- Кнопка "Категорії" має відкривати розділ з категоріями витрат, де користувач може переглянути та керувати категоріями.
- Кругова діаграма має відображати розподіл витрат по категоріях у відсотковому значені. Вона має дозволяти користувачеві швидко оцінити, які категорії займають більшу частку витрат. Кольорове кодування різних сегментів діаграми допомагає зрозуміти цей розподіл на перший погляд.

При проектуванні графічного інтерфейсу було враховано принципи зручності та інтуїтивності. Елементи інтерфейсу розташовані у логічному порядку, що дозволяє користувачу швидко знайти потрібну функцію та легко взаємодіяти з нею. Кольорова схема та типографіка були підібрані таким чином, щоб створити гармонійний та привабливий зовнішній вигляд.

Проектування графічного інтерфейсу проводилося з урахуванням масштабування. Було забезпечено адаптивність інтерфейсу для різних розмірів вікна, щоб забезпечити комфортне використання додатку. Нижче наведено декілька скріншотів з програми PocketPenny:

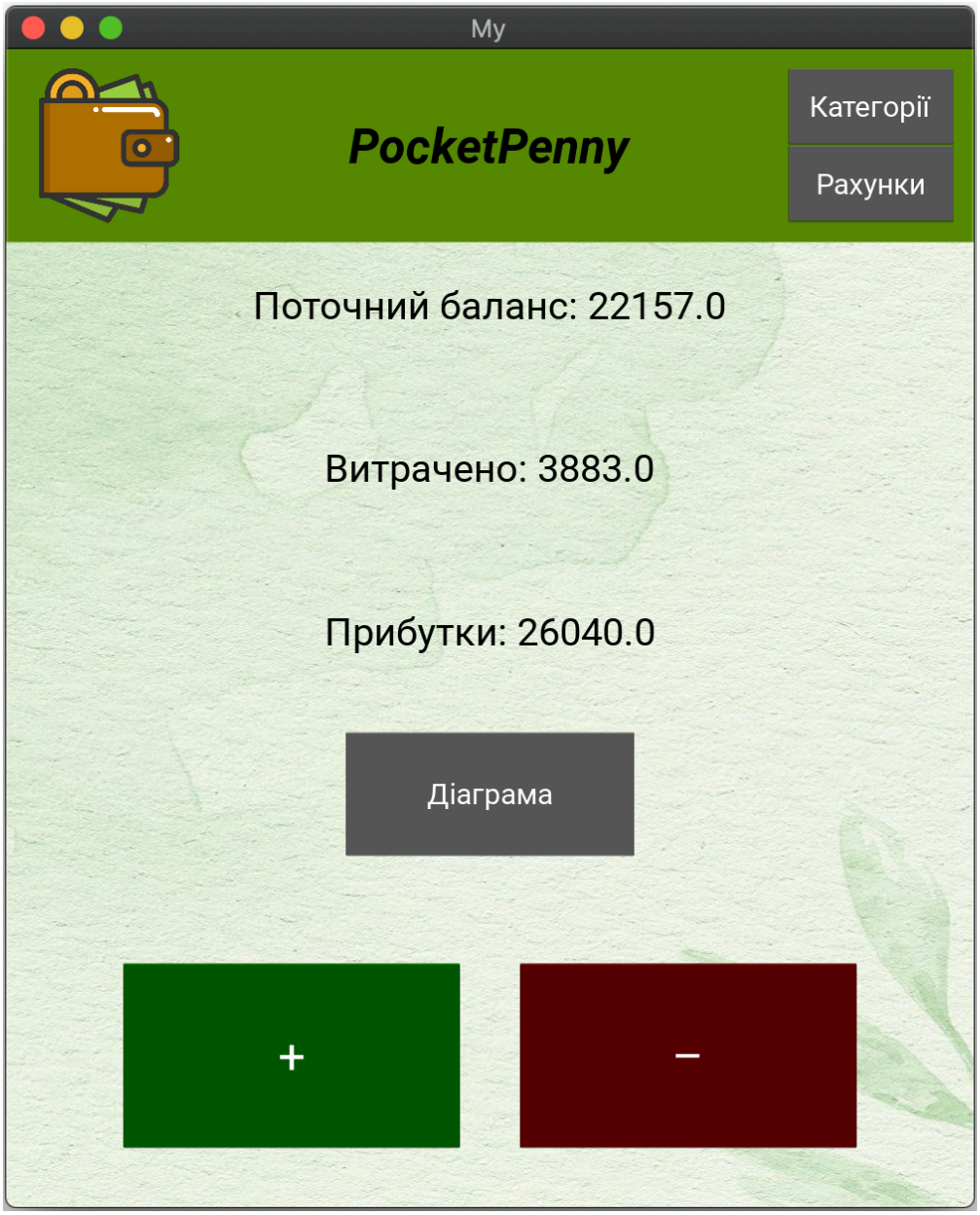


Рисунок 3.6 – Скріншот головного екрану додатку

На головному екрані користувач бачить найголовніші елементи інтерфейсу, які були описані вище. Для кращого прикладу в програму вже були введені деякі записи, щоб краще відобразити поточний баланс, витрати та прибутки.

У програмі також передбачено відображення від'ємного балансу, якщо значення поточного балансу стає меншим за нуль. У такому випадку поточний баланс буде відображатися у червоному кольорі, що одразу привертає увагу користувача.

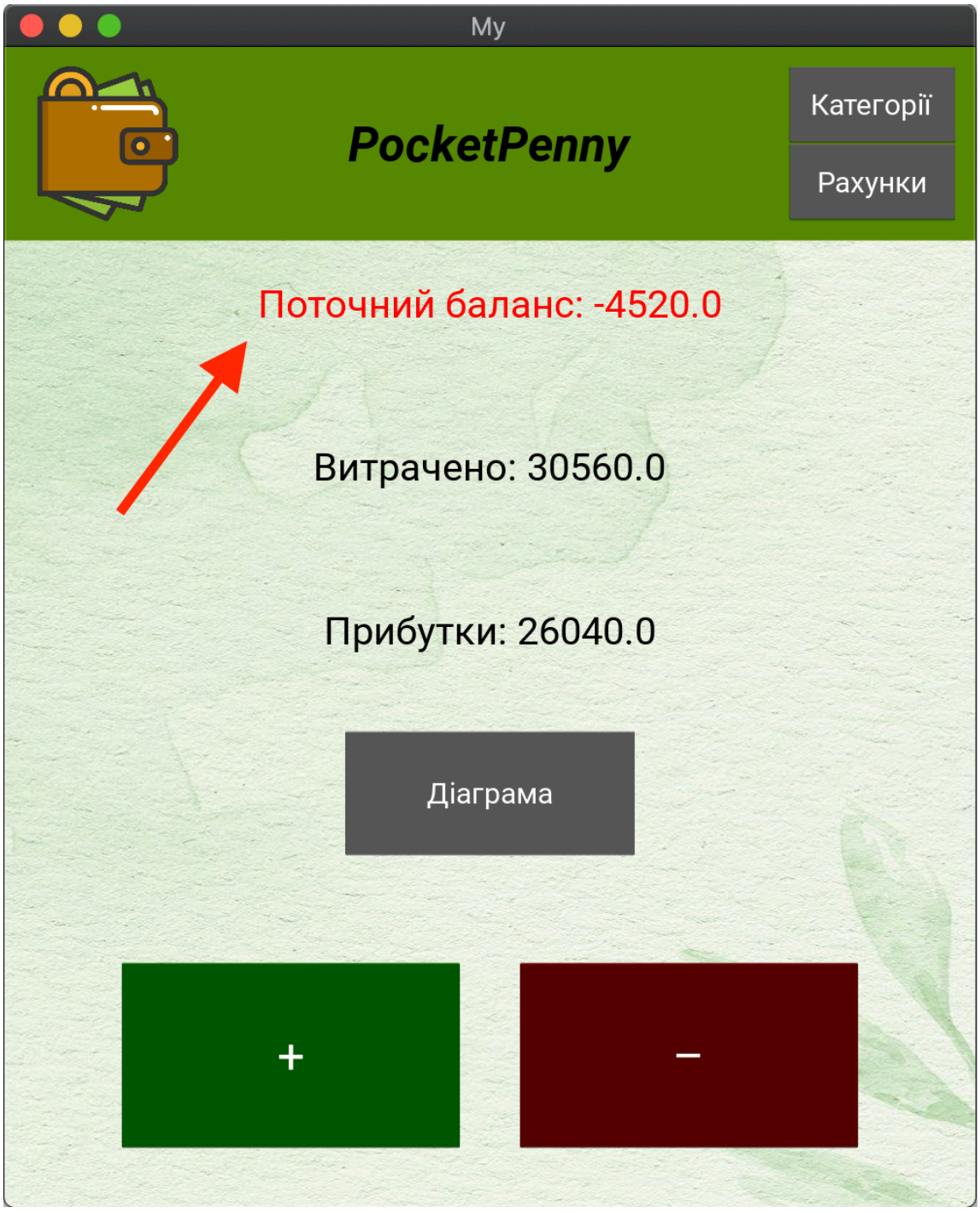


Рисунок 3.7 – Поточний баланс червоного кольору при значенні менше нуля

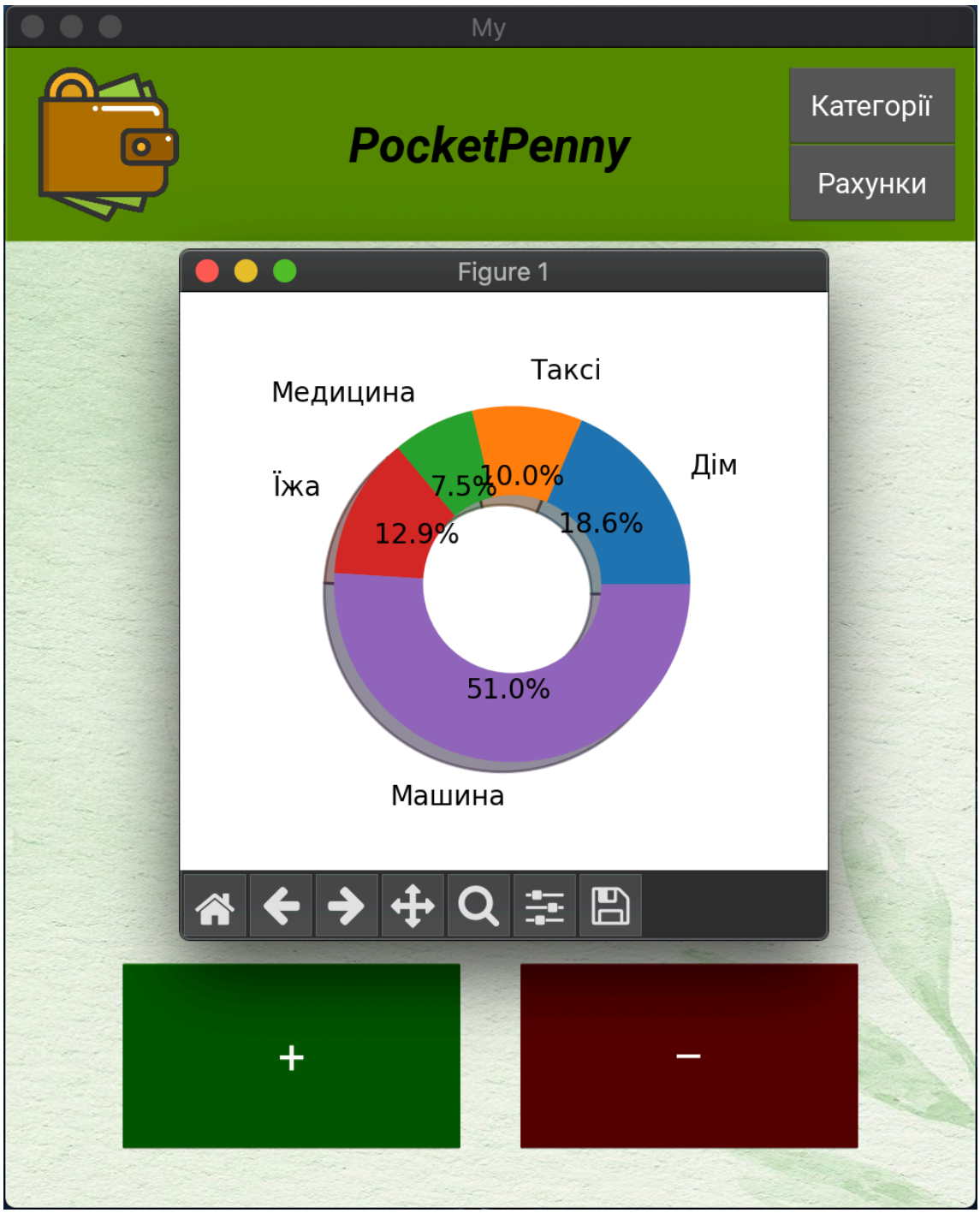


Рисунок 3.8 – Відображення кругової діаграми з категоріями у % відношенні

Для зручного перегляду даних секторів категорій, діаграма виводиться у відокремлене вікно. Крім того, в майбутньому це вікно можна розширити, додавши інші способи візуалізації та відображення інформації.

Рисунок 3.9 – Список категорій для запису витрат та кнопка для створення нової категорії

При натискані на кнопку "Категорії" перед користувачем відкривається поп–ап вікно, в якому він може бачити усі категорії для запису витрат. Для кращої візуалізації присутні іконки відповідно до категорії. Внизу списку є кнопка "Нова категорія", за допомогою якої користувач може створити нову категорію для свої потреб. Їй буде задана іконка за замовчуванням.

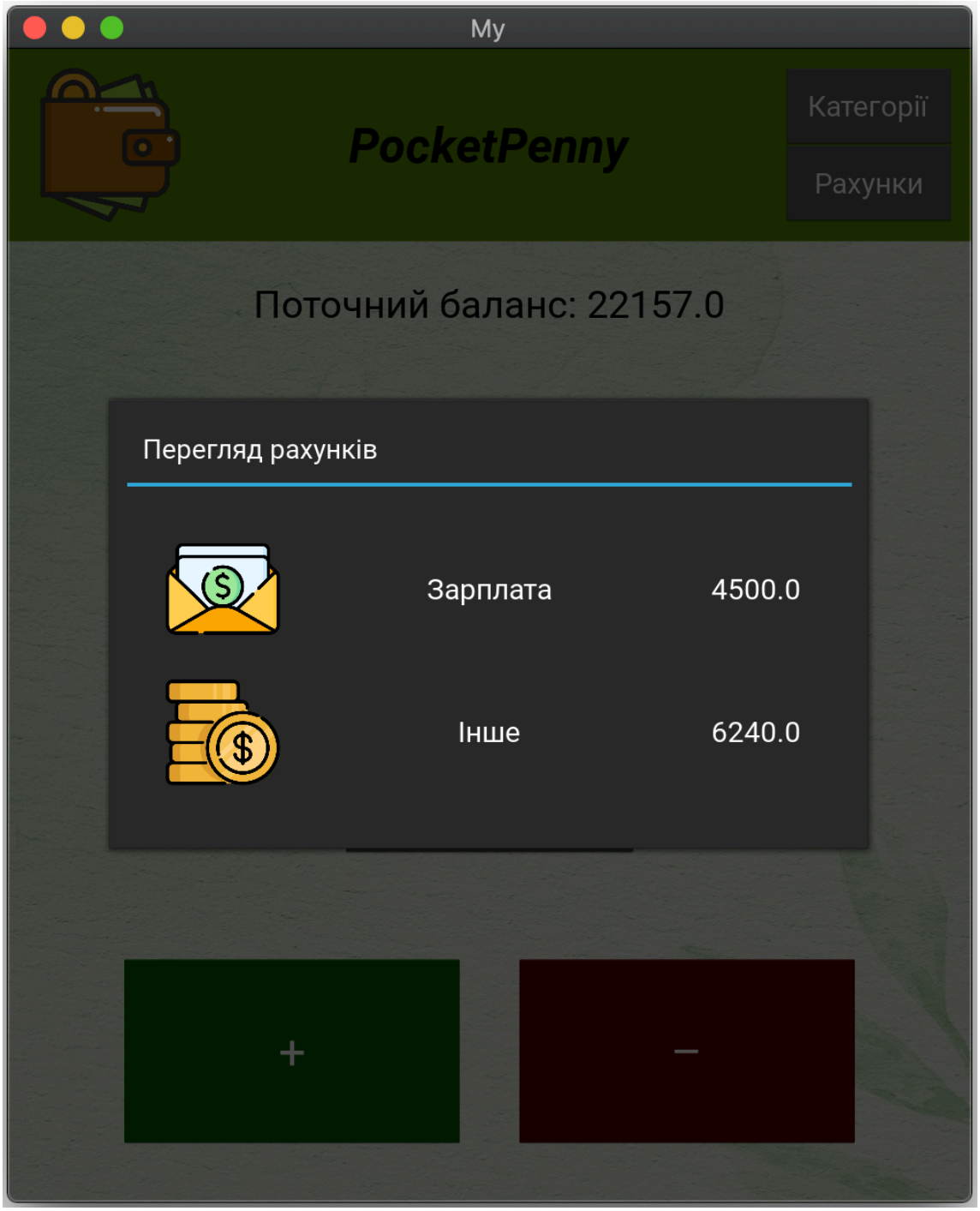


Рисунок 3.10 – Список категорій для запису прибутків

У цьому поп–ап вікні перегляду рахунків користувач може бачити категорії "Зарплата" та "Інше". Для кращого сприйняття біля категорій відображаються іконки. У цих категоріях відсутня можливість додавання додаткових категорій прибутку.

3.6 Реалізація функціональної частини

3.6.1 Створення кнопок "Категорії" та "Рахунки"

Для розробки інтерфейсу програми були використані бібліотеки фреймворку Kivy, які надали необхідні засоби та інструменти для створення графічного інтерфейсу. Спочатку, було створено ряд з трьох комірок у хедері вікна програми, який слугує частиною для розміщення елементів. У третій комірці цього ряду були розміщені кнопки "Категорії" та "Рахунки". Вибір цього положення був обґрунтований зручністю та логічністю розташування кнопок у контексті взаємодії з користувачем. Таким чином це положення дозволяє забезпечити легкий доступ до функціональності "Категорії", а також "Рахунки" для зручної навігації користувача.

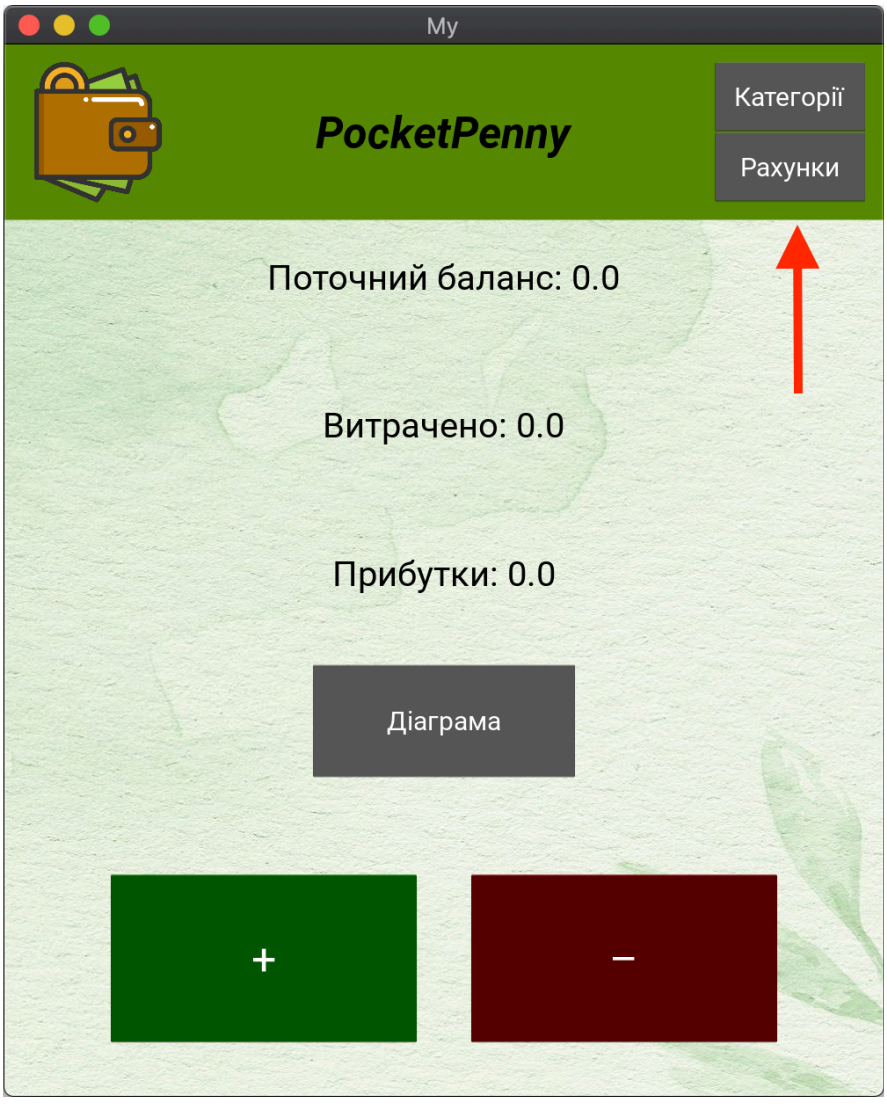


Рисунок 3.11 – Кнопки "Категорії" та "Рахунки"

3.6.2 Створення показників рахунків та балансу

Під час планування додатку PocketPenny, було чітко визначено, які елементи є найголовнішими та повинні бути видимими на головному екрані для користувача. Основними елементами є показники фінансового стану користувача, а саме:

- Поточний баланс відображає суму на рахунку користувача, враховуючи всі витрати та прибутки, які він вносить у програму
- Баланс витрачених коштів відображає загальну суму, яку користувач витратив на різні категорії
- Баланс прибутків відображає суму всіх доходів користувача

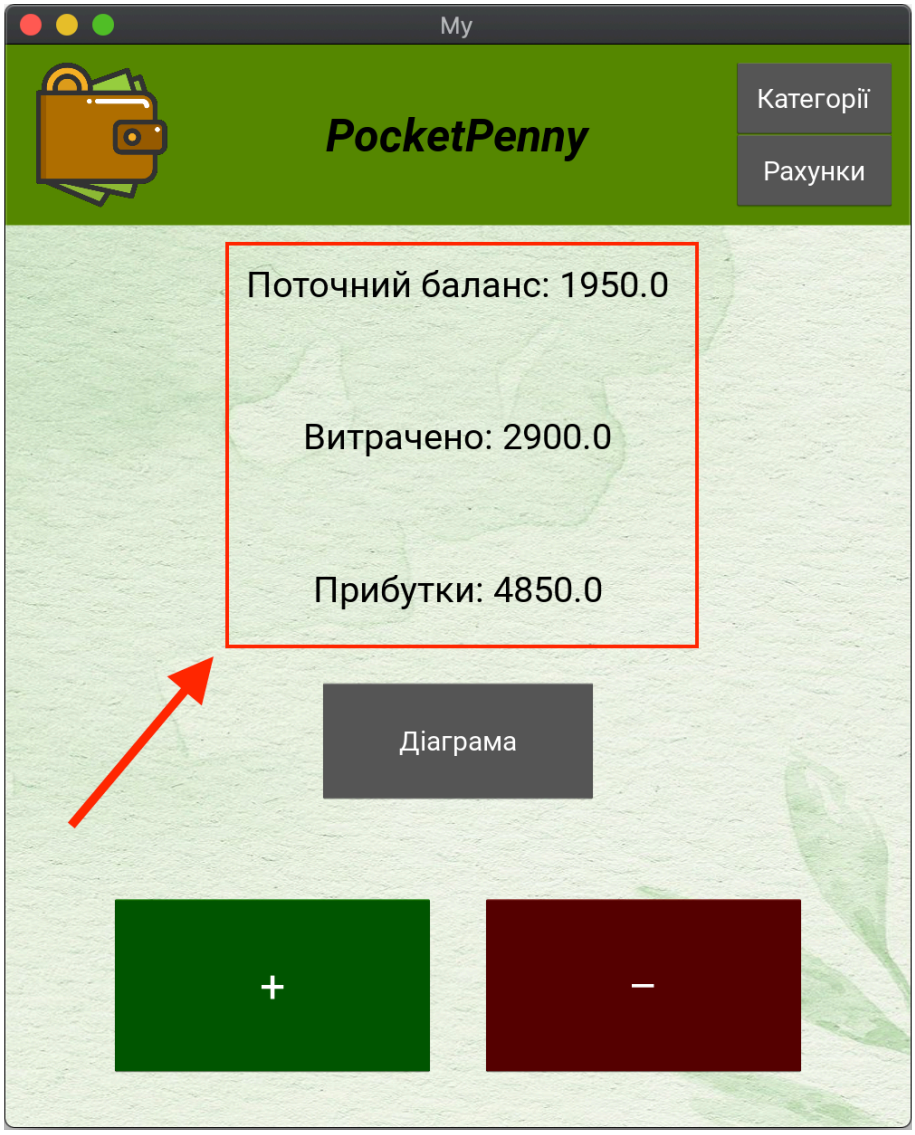


Рисунок 3.12 – Відображення балансів на головному екрані

3.6.3 Створення кнопки "Діаграма"

Ще одним ключовим елементом в додатку PocketPenny, що дозволяє користувачеві відстежувати та контролювати особисті витрати, є використання діаграм для представлення інформації. Це полегшує сприйняття та аналіз даних, які вводяться в програму. Після аналізу різних видів діаграм та форм–фактору додатку було обрано кругову діаграму з порожнім центром для стильного та сучасного вигляду.

Під час реалізації цього функціоналу зустрілися деякі труднощі, пов'язані з функціональними обмеженнями використовуваних бібліотек. Не вдалося відобразити діаграму безпосередньо на головному екрані через складнощі з оновленням балансів та категорій користувача в реальному часі, особливо з урахуванням обмежень використовуваних бібліотек.

Було прийняте рішення виділити відображення кругової діаграми в окреме вікно. Це дало змогу рухатися далі та продовжувати розробку. Крім того, винесення діаграми до окремого вікна сприяє поліпшенню сприймання інформації, оскільки користувач може фокусуватися на деталях діаграми без зайвих відволікань на головному екрані. Також таке рішення проблеми відкриває можливість для подальшого розширення функціональності, дозволяючи додавати більше графіків та діаграм, перемикатися між ними, тим самим забезпечуючи користувачеві більше інструментів для перегляду та аналізу своїх фінансів.

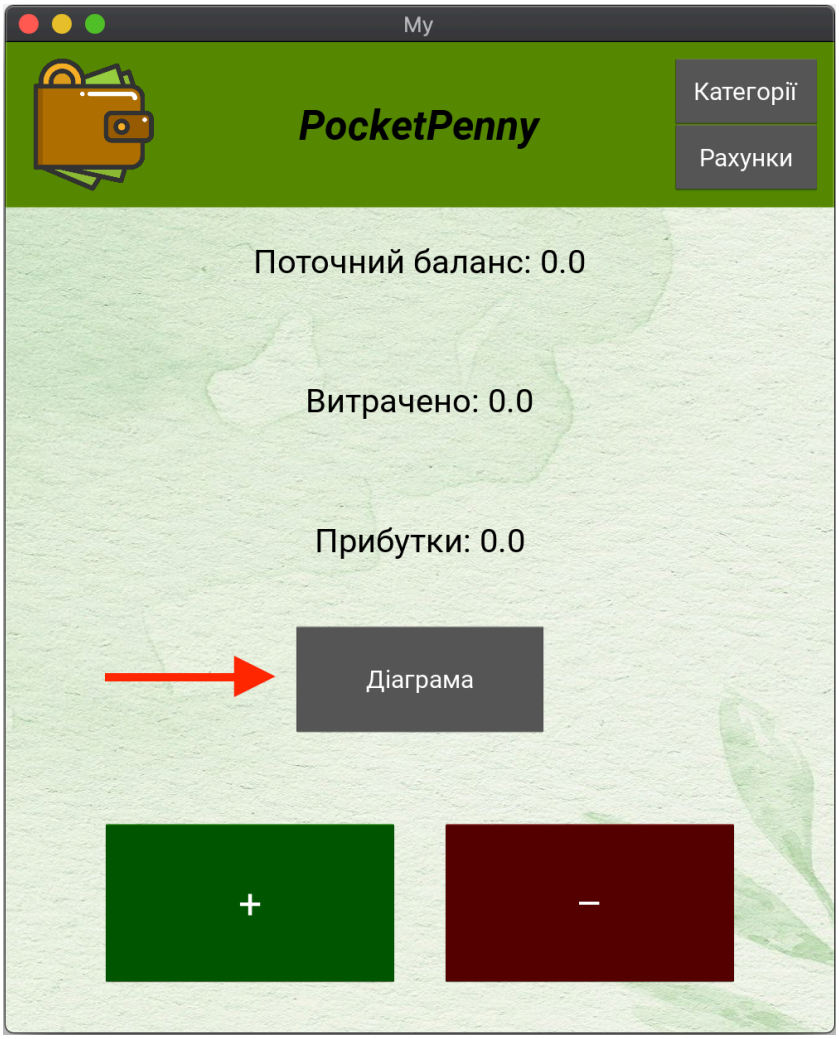


Рисунок 3.13 – Розташування кнопки для побудови та відображення діаграми на головному екрані

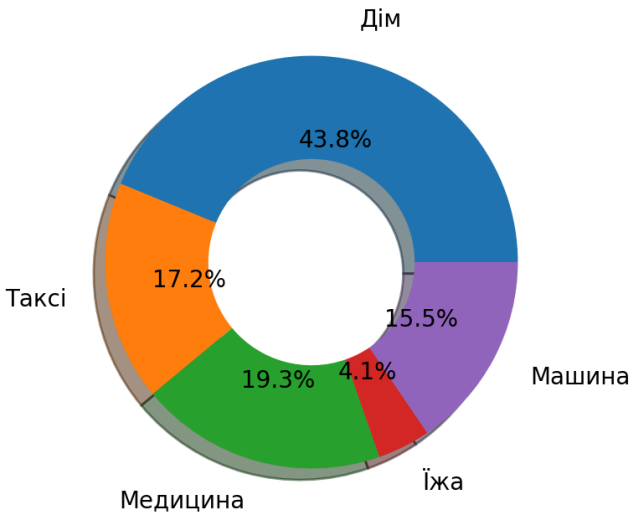


Рисунок 3.14 – Кругова діаграма з відсотковим відображенням категорій

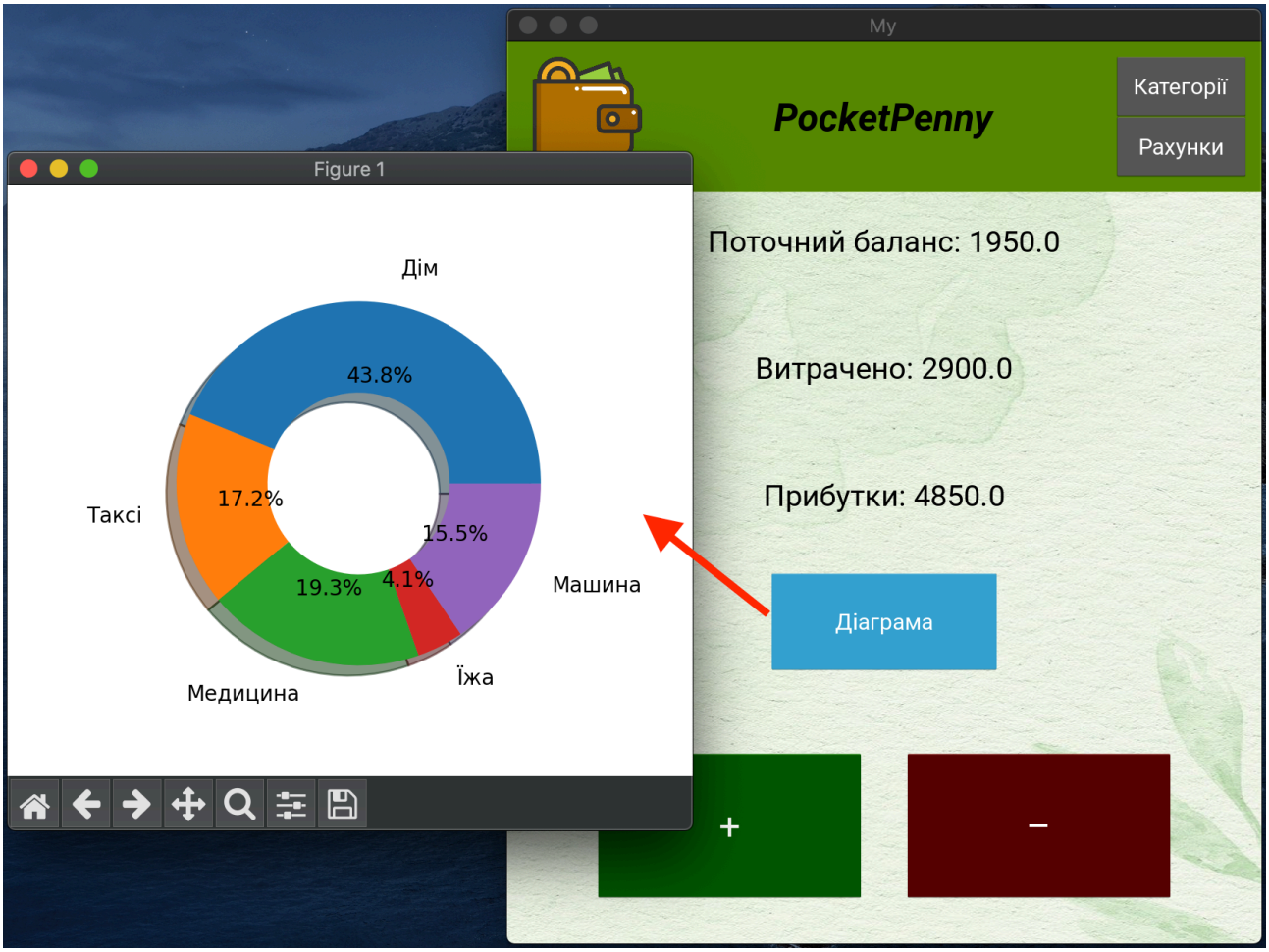


Рисунок 3.15 – Генерація кругової діаграми в окремому вікні після натискання на кнопку "Діаграма" та її відображення відносно основного вікна програми

Закриття вікна діаграми та повернення до головного вікна PocketPenny відбувається за допомогою червоної кнопки в лівому верхньому куті вікна діаграми. Після цього користувач знову бачитиме лише головне вікно програми і може продовжити взаємодію з нею. Оскільки проблема з оновленням діаграми у реальному часі вже була згадана, головне вікно програми не може бути використане, поки діаграма відкрита – елементи просто не клікабельні.

3.6.4 Створення кнопок "Плюс" та "Мінус"

Ще одним основним компонентом у програмі з керування фінансами є функціонал кнопок, які забезпечують додавання або віднімання суми до/з балансу користувача. Ці кнопки називаються "Дохід" і "Витрата", або відповідно "Плюс" і "Мінус".

Під час проектування та розміщення кнопок було обране оптимальне та логічне розташування на екрані, щоб забезпечити найлегший доступ до них для користувача. Таким чином, кнопка "Мінус" була розташована у правому нижньому куті екрану. Це обумовлено тим, що витрати переважають над прибутками в нашому повсякденному житті. Тому задля зручності було доцільно розмістити її ближче до великого пальця правої руки користувача.

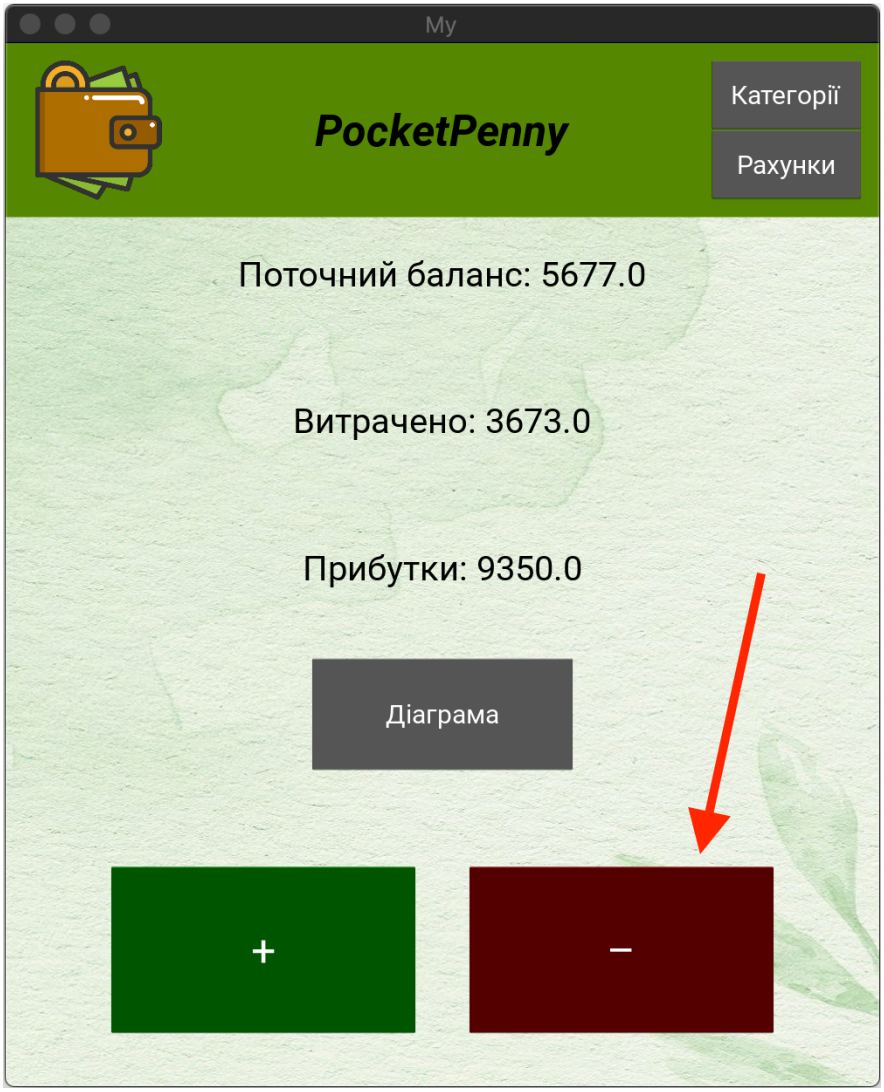


Рисунок 3.16 – Розташування кнопки "Мінус" на головному екрані

При натисканні на кнопку "Мінус" користувачу буде відкрито вікно, де він зможе ввести суму витрати, яку він бажає записати. Після цього користувач повинен натиснути кнопку "Далі", щоб перейти до наступного вікна з вибором категорії витрати. Таким чином користувач впише суму та обере категорію витрати, такі як витрати на медицину, їжу, або розваги.

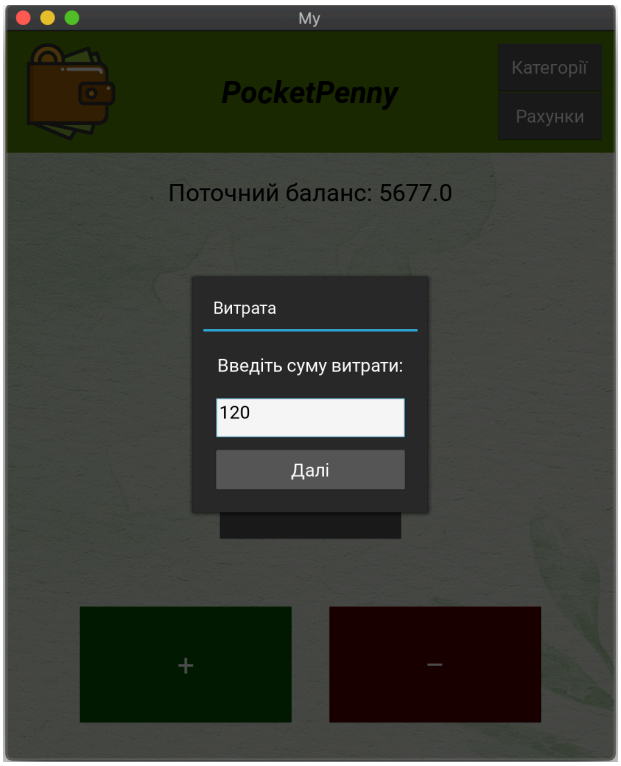


Рис. 3.17 – Введення суми витрати

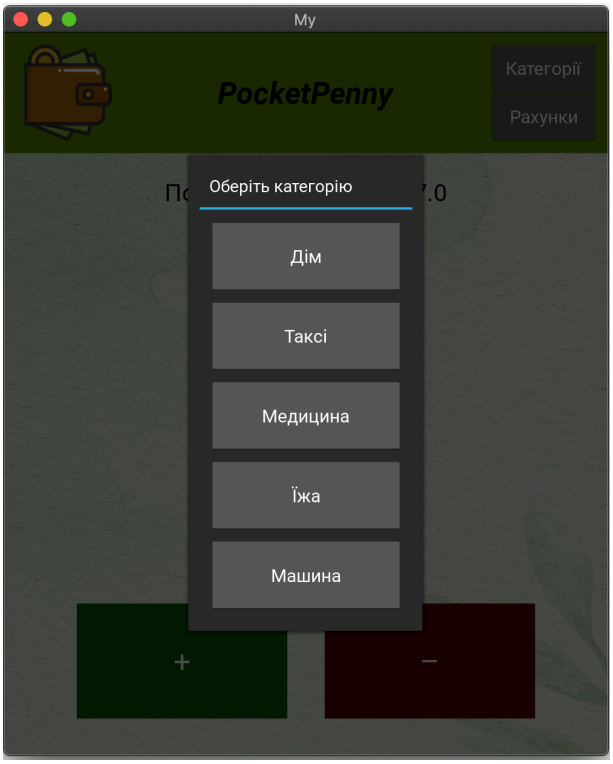


Рисунок. 3.18 – Вибір категорії

Після введення суми витрати та вибору категорії, дані будуть збережені в програмі і стануть доступними для відображення на круговій діаграмі. Крім того, ці дані також будуть відображатися у розділі "Категорії", до якого можна перейти, натиснувши кнопку "Категорії" на головному екрані.

При натисканні на кнопку "Плюс" користувачу відкриється віконце, де він зможе ввести суму доходу, яку він бажає записати. Після цього користувач повинен натиснути кнопку "Далі", щоб перейти до наступного вікна з вибором категорії доходу. У цьому вікні користувач може обрати одну з двох категорій: "Зарплата" або "Інше". Функція додавання інших категорій прибутку відсутня.

Таким чином, користувач вводить суму доходу та обирає відповідну категорію, що може включати доходи від заробітку, тобто зарплату, чи інші додаткові доходи, наприклад акції або монетизація.

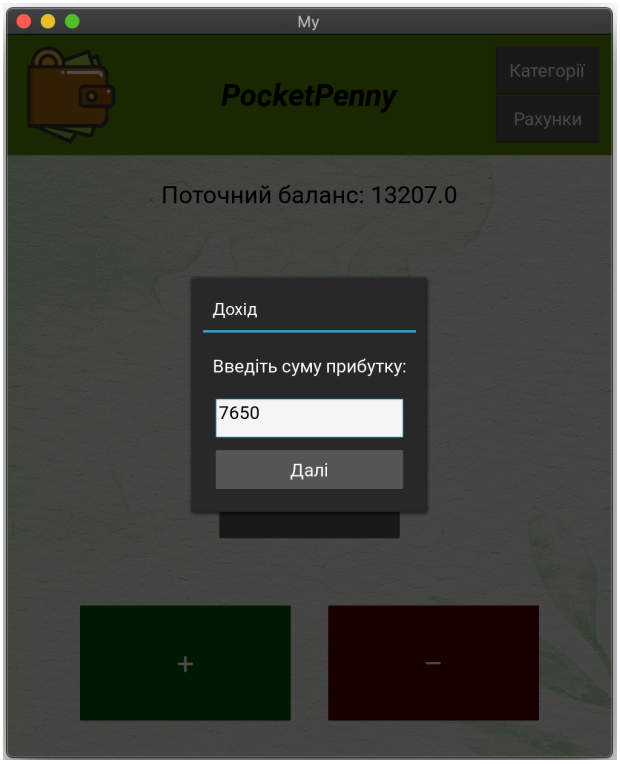


Рис. 3.19 – Введення суми прибутку

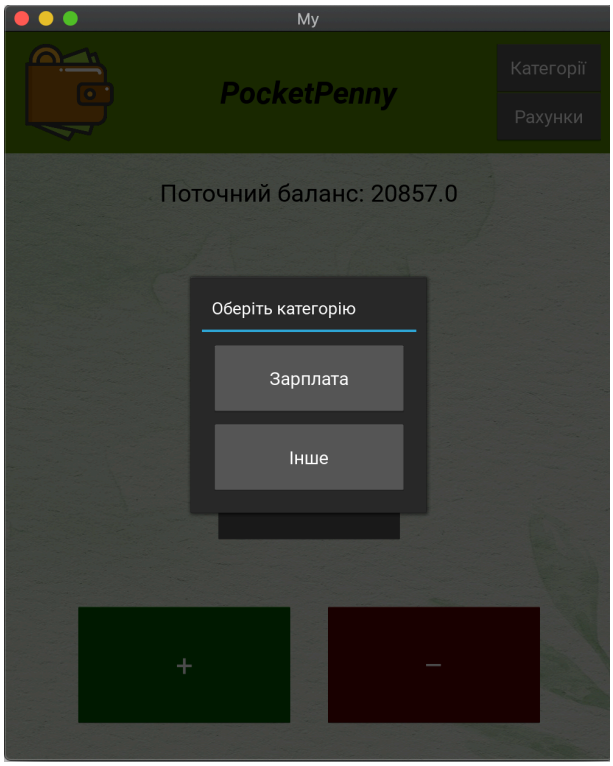


Рисунок. 3.20 – Вибір категорії

Після введення суми прибутку, користувач може перейти до наступного вікна для вибору категорії доходу, такої як "Зарплата" або "Інше". Таким чином, користувач може легко внести нові доходи до програми і відстежувати їх у розділі "Категорії".

Як вже було розглянуто та обговорено розташування кнопки "Мінус", то логічним кроком було розмістити кнопку "Плюс" зліва від неї. Беручи до уваги те, що операції прибутку записуються рідше, ніж витрати, використання кнопки "Плюс" не таке часте, як кнопки "Мінус".

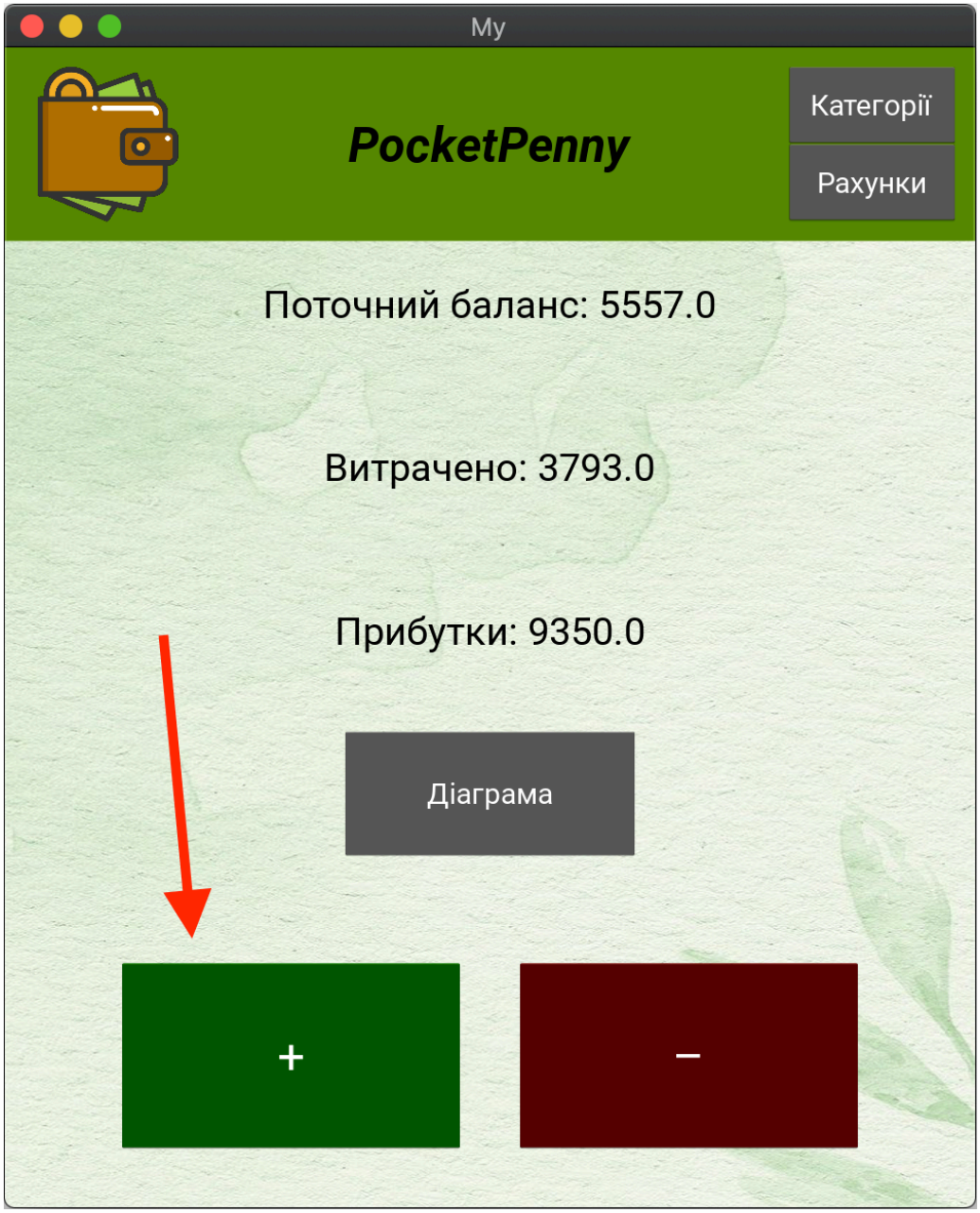


Рисунок 3.21 – Розташування кнопки "Плюс" на головному екрані

3.7 Проведення тестування

Тестування програмного забезпечення є необхідною складовою процесу розробки будь-якого програмного продукту, оскільки цей етап дозволяє виявити та виправити помилки і недоліки, що можуть вплинути на його функціональність та якість.

Для забезпечення якості тестування в умовах обмеженого часу під час розробки додатку PocketPenny було проведено два етапи тестування:

- тестування під час розробки для виявлення критичних та блокуючих помилок для подальшого їх виправлення
- тестування повністю готового продукту, виявлення багів та мінорні виправлення

Під час розробки додатку помилки, що виникали, відображалися у консолі IDE середовища PyCharm, яке використовувалося під час розробки. Зважаючи на розмір проекту та обсяг коду, під час останнього етапу тестування спочатку проводився дебаг коду. У разі виявлення помилок, на них зверталася увага для їх усунення. Після проведення код–рев'ю та виправлення помилок, дебаг проводився ще раз, і при відсутності помилок було здійснено запуск програми та подальшу перевірку.

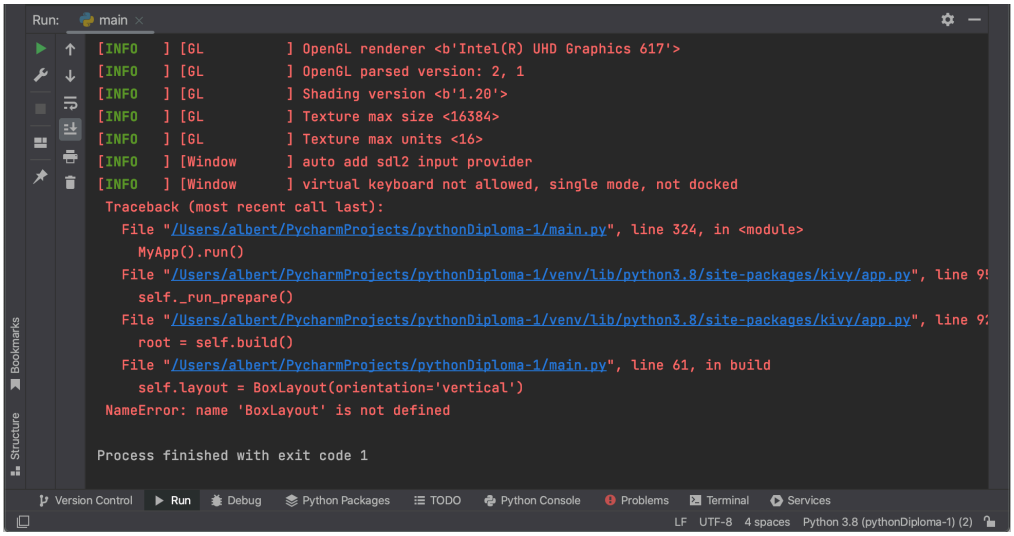


Рисунок 3.22 – Помилки у проекті під час компіляції

ВИСНОВКИ

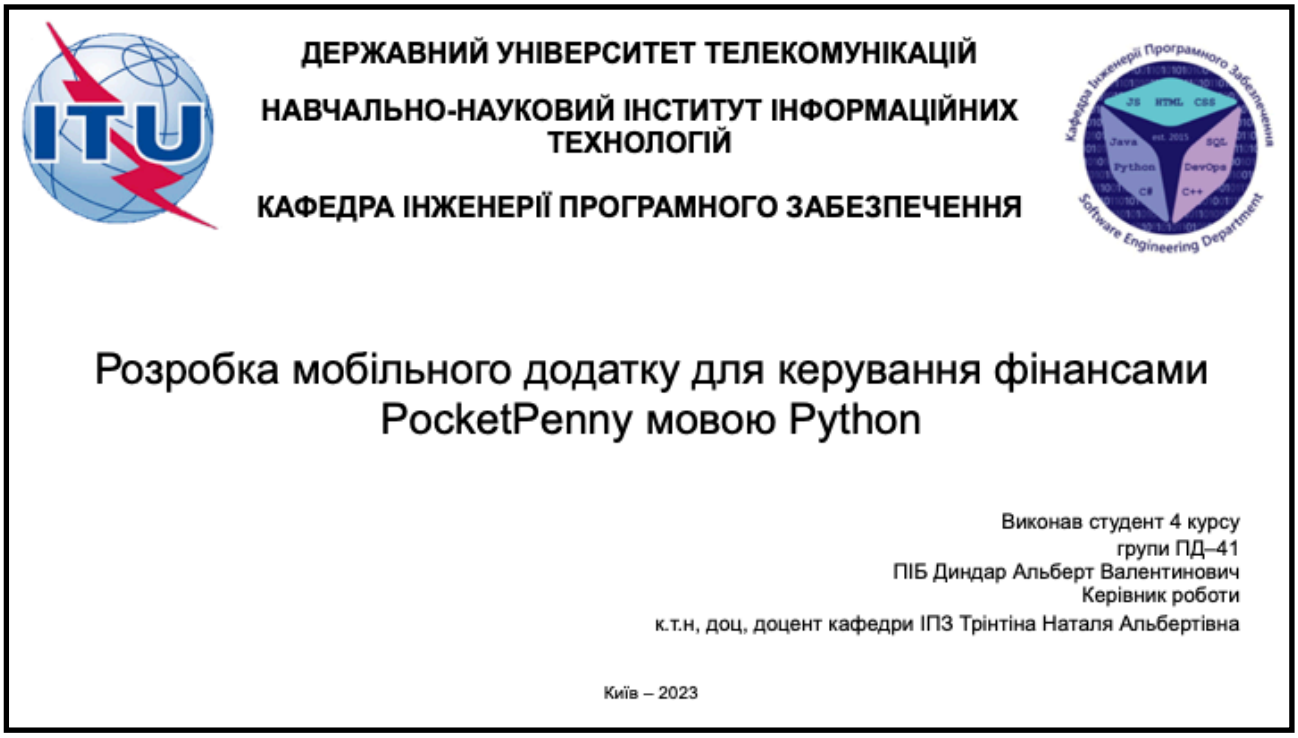
У дипломній роботі розроблено додаток для керування особистими фінансами PocketPenny. Додаток орієнтований на платформу Андроід.

1. Було проаналізовано існуючі фінансові додатки, оцінено їх функціонал, користувацький інтерфейс та загальну ефективність. В результаті аналізу були виявлені переваги, такі як зручний спосіб керування фінансами та детальна статистика, але також виявлені й недоліки, такі як обмежений функціонал та неінтуїтивний інтерфейс. Дані висновки були враховані при розробці власного фінансового додатку для забезпечення більшої задоволеності користувачів та покращення їх фінансового управління.
2. Проведено аналіз технічних засобів, які були доступні для розробки фінансового додатку. У результаті аналізу були враховані критерії ефективності, доступності та сумісності з розроблюваною платформою. Після ретельного вибору були обрані необхідні засоби, що найкраще відповідали потребам проекту і забезпечували оптимальну реалізацію функціоналу додатку..
3. Було проведено аналіз використання аналогічних фінансових додатків користувачами. На його основі було спроектовано та розроблено додаток PocketPenny, що відповідає потребам та вимогам користувачів. Під час розробки було звернуто особливу увагу на функціональність, зручність використання та естетичний дизайн додатку.
4. Після завершення розробки додатку було проведено тестування, під час якого були виявлені певні помилки. Задля забезпечення якості роботи додатку, виявлені помилки були уважно проаналізовані та виправлені.

ПЕРЕЛІК ПОСИЛАНЬ

1. Сайт PDFdrive [Електронний ресурс] – Режим доступу до ресурсу: https://www.pdfdrive.com/rich-dad-poor-dad-e136494023.html.
2. Habr [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com.
3. Python Software Foundation. Python Documentation. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/.
4. Kivy офіційна документація [Електронний ресурс] – Режим доступу до ресурсу: https://kivy.org/doc/stable/.
5. Kivy: Interactive Applications and Games in Python by Roberto Ulloa, 2018.
6. Python для дітей і підлітків by Jason R. Briggs, 2016.
7. Python Crash Course by Eric Matthes, 2019
8. Android Programming: The Big Nerd Ranch Guide by Bill Phillips and Brian Hardy, 2020
9. The Pragmatic Programmer: Your Journey to Mastery by Andrew Hunt and David Thomas, 2019
10. PY4E [Електронний ресурс] – Режим доступу до ресурсу: https://www.py4e.com/book.
11. Software Testing: Concepts and Practices by Nageshwar Rao Pusuluri, 2020
12. Effective Python: 90 Specific Ways to Write Better Python by Brett Slatkin, 2019
13. The Art of Software Testing by Glenford J. Myers, Corey Sandler, and Tom Badgett, 2011
14. Сайт PDFdrive [Електронний ресурс] – Режим доступу до ресурсу: https://www.pdfdrive.com/your-money-or-your-life-9-steps-to-transforming-your-relationship-with-money-and-achieving-financial-independence-revised-and-updated-for-the-21st-century-e159459889.html.
15. Research Methods for Business Students by Mark N.K. Saunders, Philip Lewis, and Adrian Thornhill, 2019

ДОДАТОК А



МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – спростити процес керування особистими фінансами за рахунок використання мобільного додатку PocketPenny.
- **Об'єкт дослідження** – процес керування особистими фінансами.
- **Предмет дослідження** – мобільний додаток для керування особистими фінансами.

X

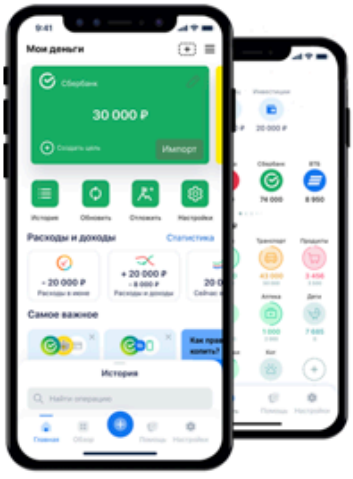
ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати переваги та недоліки існуючих аналогів.
2. Проаналізувати технічні засоби, що будуть використовуватися для розробки.
3. Спроектувати та розробити мобільний додаток на основі потреб користувачів.
4. Провести тестування додатку.

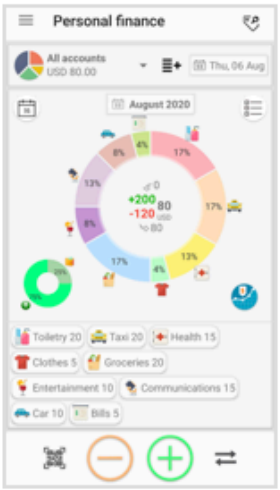
X

АНАЛІЗ АНАЛОГІВ

Monefy



CoinKeeper



Money manager, expense tracker

X

АНАЛІЗ АНАЛОГІВ

Назва	Переваги	Недоліки
Monefy	• Інтуїтивний інтерфейс • Резервна копія на Google Disk • Немає нав'язливої реклами • Можливість придбати Про версію	• Застарілий дизайн • Немає автозміни теми денна/нічна • Найкращі функції тільки у Про версії
CoinKeeper	• Приємні звуки при роботі з програмою • Гарні анімації іконок • Таблиця та дашбоард • Пароль на додаток та автозамикання	• Не юзерфрендлі інтерфейс, новачок може заплутатись • Майже весь функціонал платний • Нав'язлива реклама Про версії
Money manager, expense tracker	• Інтуїтивний інтерфейс • Пароль на додаток • Автозміна денної/нічної теми додатку • Великий вибір мов програми • Весь функціонал безкоштовний	• Не юзерфрендлі інтерфейс, новачок може заплутатись • Присутні фрізи в додатку
PocketPenny	• Інтуїтивний інтерфейс • Немає реклами • Діаграма та статистика	• Немає паролю на додаток • Відсутня зміна теми додатку

X

ВИМОГИ ДО ДОДАТКУ

1. Керування фінансами: додаток повинен забезпечувати можливість внесення та відстеження фінансових операцій, таких як доходи, витрати, перекази.
2. Статистика: додаток повинен надавати зручну статистику та звіти про фінансові показники, такі як загальний баланс, графіки доходів та витрат, категорії витрат.
3. Керування категоріями: додаток повинен дозволяти користувачу створювати, редагувати та видаляти категорії для класифікації фінансових операцій.
4. Інтерфейс користувача: додаток повинен мати зрозумілий та привабливий інтерфейс, що дозволяє зручно та ефективно взаємодіяти з функціями додатку.
5. Продуктивність: додаток повинен працювати швидко та ефективно, надаючи миттєву відповідь на запити користувача.
6. Масштабованість: додаток повинен бути гнучким та масштабованим, здатним розширюватися для підтримки більшої кількості користувачів та функцій.
7. Доступність: додаток повинен бути доступним для різних категорій користувачів, включаючи людей з обмеженими можливостями, та забезпечувати легку навігацію та використання.

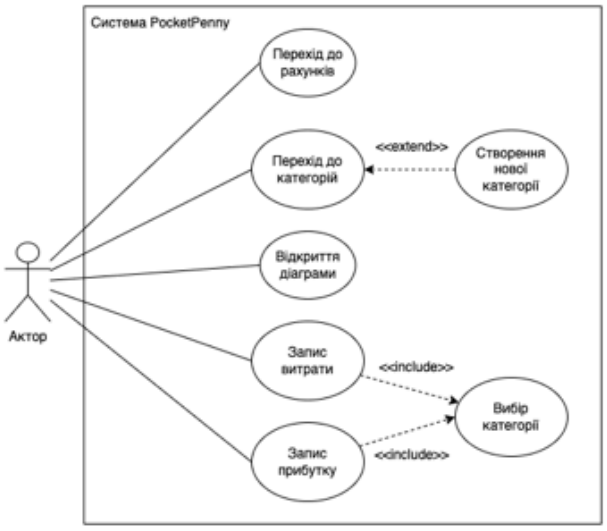
X

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

python™ kivy matplotlib

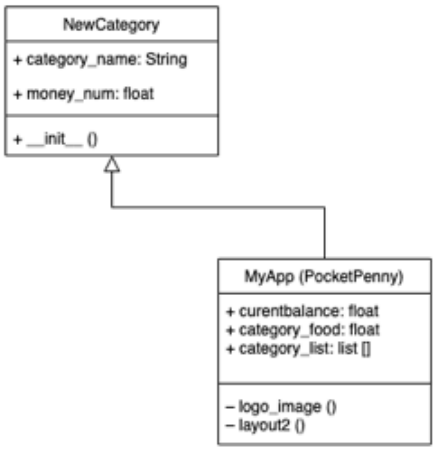
X

Діаграма варіантів використання



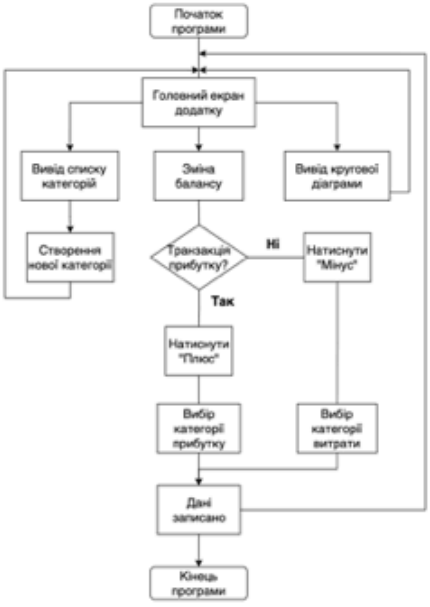
X

Діаграма класів



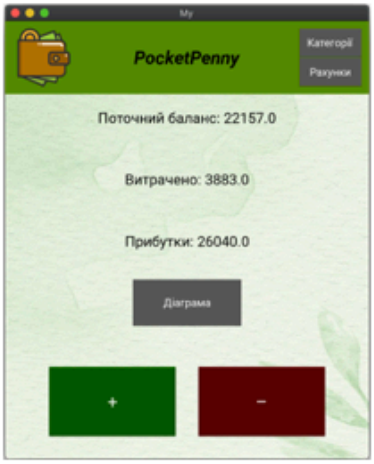
X

Блок-схема функціонування додатку



X

ЕКРАННІ ФОРМИ

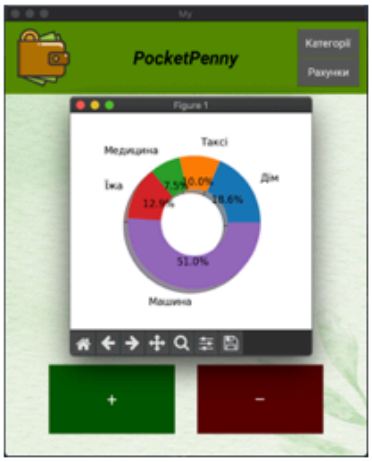


Головний екран додатку

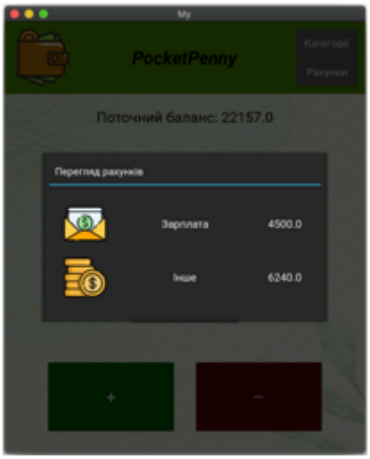
Список категорій

X

ЕКРАННІ ФОРМИ



Кругова діаграма у додатку



Список рахунків користувача

X

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Диндар А.В. Розробка мобільного додатку для керування фінансами PocketPenny мовою Python / Диндар А.В., Трінтіна Н.А. // Застосування програмного забезпечення в інфокомунікаційних технологіях: Матеріали міжнародної науково–технічної конференції. Збірник тез. 20.04.2023, ДУТ, м. Київ – К.: ДУТ, 2023. – С. 5

2. Диндар А.В. PocketPenny: як мобільний додаток допомогає людям керувати своїми фінансами / Диндар А.В., Трінтіна Н.А. // Застосування програмного забезпечення в інфокомунікаційних технологіях: Матеріали міжнародної науково–технічної конференції. Збірник тез. 20.04.2023, ДУТ, м. Київ – К.: ДУТ, 2023. – С. 89

X

ВИСНОВКИ

1. Було проаналізовано існуючі фінансові додатки та проведено аналіз їх переваг та недоліків.
2. Проведено аналіз технічних засобів доступних для розробки, і обрано необхідні засоби для реалізації додатку.
3. Було спроектовано та розроблено додаток на основі аналізу використання додатків аналогів користувачами.
4. Проведено тестування розробленого додатку, виявлено помилки та виправлено їх.

X

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б

```
import matplotlib.pyplot as plt
from kivy.app import App
from kivy.metrics import dp
from functools import partial
from kivy.uix.image import Image
from kivy.uix.label import Label
from kivy.uix.popup import Popup
from kivy.uix.button import Button
from kivy.core.window import Window
from kivy.uix.textinput import TextInput
from kivy.uix.boxlayout import BoxLayout
from kivy.graphics import Color, Rectangle
from kivy.uix.gridlayout import GridLayout
from kivy.properties import get_color_from_hex
    # Функція відображення діаграми
def ShowDiagram(instance, category_home, category_taxi, category_health,
category_food, category_car):
    if all(val == 0 for val in [category_home, category_taxi,
category_health, category_food, category_car]):
        # warning_image = Image(source='mainlogo.png')
        label = Label(text='Відображення діаграми наразі неможливе.
\n\nБудь ласка, внесіть дані у категорії \nта спробуйте ще раз.')
        popup = Popup(title='Порожня діаграма', content=label,
size_hint=(0.8, 0.8))
        popup.open()
        return
    else:
        fig = plt.figure(figsize=(6, 4))
        ax = fig.add_subplot()
        vals = [category_home, category_taxi, category_health,
category_food, category_car]
```

```
        labels = ['Дім', 'Таксі', 'Медицина', 'Їжа', 'Машина']
        exp = (0.1, 0.2, 0, 0, 0)
        ax.pie(vals, labels=labels, autopct='%1.1f%%', labeldistance=1.2,
wedgeprops=dict(width=0.5), shadow=True)
        ax.grid()
        plt.show()
    # Створення нової категорії користувачем, задання ім'я та суми
class NewCategory:
    def __init__(self, category_name, money_num):
        self.text = category_name
        self.number = money_num
    # Головний класс та функція build
class MyApp(App):
    def build(self):
    # Задаємо білий колір фону програми. Пізніше все одно поверх нього
накладемо фонове зображення
        Window.clearcolor = (1, 1, 1, 1)
    # Створюємо змінні Поточний баланс, Додати баланс, Відняти баланс
        self.curentbalance = 0.0
        self.plusbalance = 0.0
        self.minusbalance = 0.0
    # Створюємо змінні усіх категорій що нам потрібні
        self.category_home = 0.0
        self.category_taxi = 0.0
        self.category_health = 0.0
        self.category_food = 0.0
        self.category_car = 0.0
    # Створюємо змінні Зарплатня та Інше
        self.salary = 0.0
        self.other = 0.0
    # Створюємо лист категорій, що будуть створені користувачем крім вже
існуючих категорій
```

```
        self.category_list = []
    # Задаємо бекграунд програми зображенням
        self.layout = BoxLayout(orientation='vertical')
        with self.layout.canvas.before:
            self.bg = Image(source='main_bg-3.png').texture
            # Створюємо прямокутник
            self.rect = Rectangle(size=self.layout.size,
pos=self.layout.pos, texture=self.bg)
        self.layout.bind(pos=self.update_background,
size=self.update_background)
    # Створюємо grid layout з трьома колонками для хедера
        grid_layout1 = GridLayout(cols=3, size_hint=(1, 0.2),
spacing=dp(10), padding=(dp(10)))
        # Задаємо колір для хедера
        with grid_layout1.canvas.before:
    # Задаємо зелений колір
            Color(85/255, 139/255, 3/255, 1)
    # Створюємо прямокутник
            self.rect2 = Rectangle(size=grid_layout1.size,
pos=grid_layout1.pos)
    # Додаємо зображення лого як дочірній елемент
            self.logo_image = Image(source='mainlogo.png', size_hint=(0.3,
0.3))
            grid_layout1.add_widget(self.logo_image)
        grid_layout1.bind(size=self._update_rect2, pos=self._update_rect2)
        self.layout.add_widget(grid_layout1)
    # Додаємо текст у другу колонку
        text_label = Label(text='PocketPenny', color='black', size_hint=(1,
1), font_size='25sp', italic=True, bold=True)
        grid_layout1.add_widget(text_label)
    # Додаємо кнопки у третю колонку
        layoutaccount = BoxLayout(orientation='vertical', size_hint=(0.3,
0.3))
```

```
        category_button = Button(text='Категорії', size_hint=(1, 1))
        category_button.bind(on_release=self.category_button_pressed)
        layoutaccount.add_widget(category_button)
        category_button = Button(text='Рахунки', size_hint=(1, 1))
        category_button.bind(on_release=self.account_button_pressed)
        layoutaccount.add_widget(category_button)
        grid_layout1.add_widget(layoutaccount)
    # Створюємо gridlayout для відображення балансів
        grid_layout2 = GridLayout(cols=1, size_hint=(1, 1), spacing=dp(20))
        self.layout.add_widget(grid_layout2)
    # Додаємо пусту комірку у перший рядок
        self.balance_label = Label(text=f'Поточний баланс:
{self.curentbalance}', font_size='20sp', color='black', size_hint=(1, 0.4))
        grid_layout2.add_widget(self.balance_label)
    # Додаємо текст у другий рядок
        self.expense_label = Label(text=f'Витрачено: {self.minusbalance}',
font_size='20sp', color='black', size_hint=(1, 0.4))
        grid_layout2.add_widget(self.expense_label)
    # Додаємо текст у третій рядок
        self.earn_label = Label(text=f'Прибутки: {self.plusbalance}',
font_size='20sp', color='black', size_hint=(1, 0.4))
        grid_layout2.add_widget(self.earn_label)
    # Додаємо кнопку відображення діаграми нижче
        diagram_layout = BoxLayout(orientation='vertical', size_hint=(1,
0.4))
        diagram_button = Button(text='Діаграма', size_hint=(0.3, 0.4),
pos_hint={'center_x':0.5, 'center_y': 0.5})
        diagram_button.bind(on_release=lambda instance:
ShowDiagram(instance, self.category_home, self.category_taxi,
self.category_health, self.category_food, self.category_car))
        diagram_layout.add_widget(diagram_button)
        grid_layout2.add_widget(diagram_layout)
```

```
    # Створюємо нижче кнопки плюс та мінус, задаємо їх параметри відорбраження, відступи, та розміри

        layout2 = BoxLayout(orientation='horizontal', spacing=dp(30), padding=(dp(60), dp(35), dp(60), dp(30)))

        green_button = Button(text='+', font_size='25sp', background_color=(0, 1, 0, 1), size_hint=(1, 1), size=(50,50))

      green_button.bind(on_release=self.green_button_pressed)

        red_button = Button(text='—', font_size='25sp', background_color=(1, 0, 0, 1), size_hint=(1, 1), size=(50,50))

        red_button.bind(on_release=self.red_button_pressed)

    # Розміщуємо кнопки в один layout і потім у grid для їх коректного відображення

        layout2.add_widget(green_button)

        layout2.add_widget(red_button)

        grid_layout2.add_widget(layout2)

        return self.layout

    # Функція для оновлення бекграунду додатку

    def update_background(self, instance, value):

        self.rect.pos = instance.pos

        self.rect.size = instance.size

    # Функція для оновлення зображення лого додатку

    def _update_rect2(self, instance, value):

        self.rect2.pos = instance.pos

        self.rect2.size = instance.size

    # Функція при натискані на кнопку категорій

    def category_button_pressed(self, instance=None):

    # Створюємо layout для списку кнопок

        buttons_layout = BoxLayout(orientation='vertical',spacing=dp(10), padding=(dp(10)))

        grid_layout3 = GridLayout(cols=3, size_hint=(1, 1), spacing=dp(10), padding=(dp(10)))

        buttons_layout.add_widget(grid_layout3)

    # Створюємо категорію дім, задаємо їй зображення, назву та відображення витрат
```

```
        image = Image(source='home-2.png', size_hint=(0.3, 0.3))
        grid_layout3.add_widget(image)
        category_label = Label(text=f'Дім', size_hint=(1, 1))
        grid_layout3.add_widget(category_label)
        category_label = Label(text=f'{str(self.category_home)}',
size_hint=(0.3, 0.3))
        grid_layout3.add_widget(category_label)
    # Створюємо категорію  таксі, задаємо їй зображення, назву та
відображення витрат
        image = Image(source='taxi-2.png', size_hint=(0.3, 0.3))
        grid_layout3.add_widget(image)
        category_label = Label(text=f'Таксі', size_hint=(1, 1))
        grid_layout3.add_widget(category_label)
        category_label = Label(text=f'{str(self.category_taxi)}',
size_hint=(0.3, 0.3))
        grid_layout3.add_widget(category_label)
    # Створюємо категорію медицина, задаємо їй зображення, назву та
відображення витрат
        image = Image(source='med-2.png', size_hint=(0.3, 0.3))
        grid_layout3.add_widget(image)
        category_label = Label(text=f'Медицина', size_hint=(1, 1))
        grid_layout3.add_widget(category_label)
        category_label = Label(text=f'{str(self.category_health)}',
size_hint=(0.3, 0.3))
        grid_layout3.add_widget(category_label)
    # Створюємо категорію їжа, задаємо їй зображення, назву та
відображення витрат
        image = Image(source='food-2.png', size_hint=(0.3, 0.3))
        grid_layout3.add_widget(image)
        category_label = Label(text=f'Їжа', size_hint=(1, 1))
        grid_layout3.add_widget(category_label)
        category_label = Label(text=f'{str(self.category_food)}',
size_hint=(0.3, 0.3))
```

```
        grid_layout3.add_widget(category_label)

    # Створюємо категорію машина, задаємо їй зображення, назву та
відображення витрат

        image = Image(source='car-2.png', size_hint=(0.3, 0.3))

        grid_layout3.add_widget(image)

        category_label = Label(text=f'Машина', size_hint=(1, 1))

        grid_layout3.add_widget(category_label)

        category_label = Label(text=f'{str(self.category_car)}',
size_hint=(0.3, 0.3))

        grid_layout3.add_widget(category_label)

    # Цикл для створення нової категорії, який задає зображення (за
замовчуванням), назву та відображення витрат

        for category in self.category_list:

            image = Image(source='new_category_icon.png', size_hint=(0.3,
0.3))

            grid_layout3.add_widget(image)

            category_label = Label(text=str(category.text), size_hint=(1,
1))

            grid_layout3.add_widget(category_label)

            category_label = Label(text=str(category.number),
size_hint=(0.3, 0.3))

            grid_layout3.add_widget(category_label)

    # Створюємо кнопку для створення нової категорії

        create_layout = BoxLayout(orientation='vertical', size_hint=(1,
0.1))

        create_category_button = Button(text='Нова категорія',
size_hint=(1, 1))
create_category_button.bind(on_release=self.create_category_button)

        create_layout.add_widget(create_category_button)

        buttons_layout.add_widget(create_layout)

    # Створюємо popup, який містить в собі список кнопок

        self.popup = Popup(title='Перегляд категорій',
content=buttons_layout, size_hint=(0.8, 0.8))

        self.popup.open()
```

```
    # Створення нової категорії користувачем
    def create_category_button(self, instance):
        create_layout = BoxLayout(orientation='vertical', size_hint=(0.3,
0.3))
        popup = Popup(title='Створення нової категорії',
content=create_layout, size_hint=(0.8, 0.8))
        self.input = TextInput()
        btn_ok = Button(text='OK')
btn_ok.bind(on_release=partial(self.add_to_category_list, popup))
        btn_back = Button(text='Назад')
btn_back.bind(on_release=self.category_button_pressed)
        create_layout.add_widget(Label(text='Назва нової категорії:'))
        create_layout.add_widget(self.input)
        create_layout.add_widget(btn_ok)
        create_layout.add_widget(btn_back)
        popup.open()
    # Додавання нової створеної категорії до загального списку категорій
    def add_to_category_list(self, popup, instance=None):
        TextInput = self.input.text
        self.category_list.append(NewCategory(TextInput, 0.0))
        popup.dismiss()
        self.popup.dismiss()
        self.category_button_pressed()
    # Функція акаунту та балансу
    def account_button_pressed(self, instance):
    # Створюємо layout для списку кнопок
        buttons_layout = BoxLayout(orientation='vertical', spacing=dp(10),
padding=(dp(10)))
        grid = GridLayout(cols=3, size_hint=(1, 1), spacing=dp(10),
padding=(dp(10)))
        buttons_layout.add_widget(grid)
    # Створюємо категорію зарплата, задаємо їй зображення, назву та
відображення суми
```

```
        image = Image(source='salary-2.png', size_hint=(0.3, 0.3))
        grid.add_widget(image)
        category_label = Label(text=f'Зарплата', size_hint=(1, 1))
        grid.add_widget(category_label)
        category_label = Label(text=f'{str(self.salary)}', size_hint=(0.3,
0.3))
        grid.add_widget(category_label)
    # Створюємо категорію інше, задаємо їй зображення, назву та
відображення суми
        image = Image(source='other-2.png', size_hint=(0.3, 0.3))
        grid.add_widget(image)
        category_label = Label(text=f'Інше', size_hint=(1, 1))
        grid.add_widget(category_label)
        category_label = Label(text=f'{str(self.other)}', size_hint=(0.3,
0.3))
        grid.add_widget(category_label)
    # Створюємо popup, який містить в собі список кнопок
        popup = Popup(title='Перегляд рахунків', content=buttons_layout,
size_hint=(0.8, 0.4))
        popup.open()
    # Функція кнопки плюс, прибуток
    def green_button_pressed(self, instance):
        layout = BoxLayout(orientation='vertical', spacing=dp(10),
padding=(dp(10)))
        input = TextInput()
        btn = Button(text='Далі')
        layout.add_widget(Label(text='Введіть суму прибутку:'))
        layout.add_widget(input)
        layout.add_widget(btn)
        popup = Popup(title='Дохід', content=layout, size_hint=(None,
None), size=(400, 400))
    # Функція при натискані кнопки плюс
        def on_button_press(instance):
```

```
    # Повідомлення в консоль
            print(f'Ви ввели {input.text}')
            self.curentbalance += float(input.text)
            self.plusbalance += float(input.text)
            self.balance_label.text = f'Поточний баланс:
{str(self.curentbalance)}'
            self.earn_label.text = f'Прибутки: {str(self.plusbalance)}'
            print(f'Ваш баланс {self.curentbalance}')
            popup.dismiss()
    # Вікно для вибору способу прибутку та категорії
            layout = BoxLayout(orientation='vertical', spacing=dp(10),
padding=(dp(10)))
            btn = Button(text='Зарплата')
            btn.on_press = lambda: (setattr(self, 'salary', self.salary +
float(input.text)), popup_category.dismiss())
            layout.add_widget(btn)
            btn = Button(text='Інше')
            btn.on_press = lambda: (setattr(self, 'other', self.other +
float(input.text)), popup_category.dismiss())
            layout.add_widget(btn)
            popup_category = Popup(title='Оберіть категорію',
content=layout, size_hint=(None, None), size=(400, 400))
            popup_category.open()
        btn.bind(on_release=on_button_press)
        popup.open()
    # Функція кнопка мінус, витрата
    def red_button_pressed(self, instance):
        layout = BoxLayout(orientation='vertical', spacing=dp(10),
padding=(dp(10)))
        input = TextInput()
        btn = Button(text='Далі')
        layout.add_widget(Label(text='Введіть суму витрати:'))
        layout.add_widget(input)
```

```
        layout.add_widget(btn)

        popup = Popup(title='Витрата', content=layout, size_hint=(None,
None), size=(400, 400))

    # Функція при натискані кнопки мінус

        def on_button_press(instance):

    # Повідомлення в консоль

            print(f'Ви ввели {input.text}')

            self.curentbalance -= float(input.text)

    # Перевірка на баланс менше нуля, і встановлення червоного кольору у
разі мінусового балансу

            if self.curentbalance < 0:

                self.balance_label.color = get_color_from_hex("#FF0000")  #
Red color

            else:

                self.balance_label.color = get_color_from_hex("#000000")

            self.minusbalance += float(input.text)

            self.balance_label.text = f'Поточний баланс:
{str(self.curentbalance)}'  # Вынести в отдельную фн-кц

            self.expense_label.text = f'Витрачено:
{str(self.minusbalance)}'

            print(f'Ваш баланс {self.curentbalance}')

            popup.dismiss()

    # Вікно для вибору категорії витрати

            layout = BoxLayout(orientation='vertical', spacing=dp(10),
padding=(dp(10)))

            btn = Button(text='Дім')

            btn.on_press = lambda: (setattr(self, 'category_home',
self.category_home + float(input.text)), popup_category.dismiss())

            layout.add_widget(btn)

            btn = Button(text='Таксі')

            btn.on_press = lambda: (setattr(self, 'category_taxi',
self.category_taxi + float(input.text)), popup_category.dismiss())

            layout.add_widget(btn)
```

```
            btn = Button(text='Медицина')
            btn.on_press = lambda: (setattr(self, 'category_health',
self.category_health + float(input.text)), popup_category.dismiss())
            layout.add_widget(btn)
            btn = Button(text='Їжа')
            btn.on_press = lambda: (setattr(self, 'category_food',
self.category_food + float(input.text)), popup_category.dismiss())
            layout.add_widget(btn)
            btn = Button(text='Машина')
            btn.on_press = lambda: (setattr(self, 'category_car',
self.category_car + float(input.text)), popup_category.dismiss())
            layout.add_widget(btn)
    # Цикл для запису витрати у відповідну категорію
            for category in self.category_list:
                btn = Button(text=category.text)
                btn.on_press = lambda: (setattr(category, 'number',
str(float(category.number) + float(input.text))), popup_category.dismiss())
                layout.add_widget(btn)
            popup_category = Popup(title='Оберіть категорію',
content=layout, size_hint=(None, None), size=(400, 800))
            popup_category.open()
        btn.bind(on_release=on_button_press)
        popup.open()
if __name__ == '__main__':
    MyApp().run()
```