МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Мобільний додаток контролю персональних витрат»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студентка групи ІТ.мдн-11с
Мілославська Юлія Миколаївна

Кваліфікаційну роботу
Захищено на засіданні ЕК
з оцінкою _____ «___»_____2022 р.

Науковий керівник _____ к.т.н., Бойко О.В.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2022

Сумський державний університет
Центр заочної та дистанційної форм навчання
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
в. о. Зав. кафедри ІТ
_____ С. М. Ващенко
«___» _____ 2022 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Мілославська Юлія Миколаївна
(прізвище, ім'я, по батькові)

1 Тема проекту *Мобільний додаток контролю персональних витрат*

затверджена наказом по університету від «04» 10 2022 р. № 0847-VI
2 Термін здачі студентом закінченого проекту «___» ___грудня__ 2022 р.
3 Вхідні дані до проекту Технічне завдання на розробку мобільного додатка контролю персональних витрат з функцією збереження розрахункових документів

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) Вивчення предметної області, формування вимог до розроблюваного застосунку, вибір методів розробки, проектування додатка, розробка та тестування застосунка.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Порівняльна таблиця застосунків – аналогів, порівняльна таблиця бібліотек роботи з діаграмами, діаграми IDF0, діаграма варіантів використання, діаграма послідовності дій користувача, прототипи додатку, ER-діаграма, табличне представленя переліку класів-адаптерів, табличне представлення переліку класів View, UML діаграма класів, UML діаграма адаптерів, табличне представлення результатів тестування, діаграма Ганта, табличне представлення оцінки ризиків

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Ознайомлення з предметною областю	11.09.2022 – 16.09.2022	
2	Аналіз аналогів та проблем	16.09.2022 – 25.09.2022	
3	Визначення функціональних вимог	26.09.2022 – 30.09.2022	
4	Визначення засобів реалізації	01.10.2022 – 08.10.2022	
5	Розробка макету та планування роботи	09.10.2022 – 25.10.2022	
6	Проектування додатку	26.10.2022 – 20.11.2022	
7	Реалізація та тестування	21.11.2022 – 30.11.2022	
8	Оформлення пояснювальної записки	05.12.2022 – 10.12.2022	

Магістрант _____ Мілославська Ю.М.

Керівник роботи _____ к.т.н. Бойко О.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Мобільний додаток контролю персональних витрат».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 30 найменувань, додатків. Загальний обсяг роботи − 85 сторінок, у тому числі 57 сторінок основного тексту, 4 сторінки списку використаних джерел, 24 сторінки додатків.

Кваліфікаційну роботу магістра присвячено розробці мобільного додатка контролю персональних витрат. В роботі проведено аналіз предметної області і існуючих рішень, проведено огляд засобів реалізації, сформовано вимоги до розроблюваної системи. У роботі виконано планування та моделювання системи і розкриті основні етапи розробки. Результатом проведеної роботи є персональна система контролю видатків з функцією оцифрування даних із розрахункових документів. Розробка проводилась з використанням інтегрованого середовища Android Studio на мові програмування Java. Додаток працює на пристроях з операційною системою Android версії 10 і вище.

Практичне значення роботи полягає у створенні зручної системи, яка дозволить її користувачам вирішити такі задачі:

- Контроль видатків
- Аналіз витрати
- Довгострокове структуроване збереження розрахункових документів
- Зручний пошук по архіву документів

Ключові слова: Android, Java, мобільний додаток, персональні витрати, розрахункові документи.

ЗМІСТ

ВСТУП

Метою роботи є створення мобільного додатка контролю персональних витрат. Тема роботи обрана зважаючи на актуальність питання підвищення фінансової грамотності населення України і відсутність на ринку безкоштовних застосунків для ведення обліку на українській мові.

Мобільний додаток призначений для контролю за персональними витратами і надає інформативну візуалізовану аналітику. Ці функції дозволять користувачам оптимізувати свої видатки, раціонально використовувати фінансові ресурси, спростять процес планування бюджету і дозволять заощаджувати. Саме заощадливість, ефективне управління коштами і грамотне планування бюджету і є основними ознаками фінансової грамотності. Тобто додаток надає користувачеві зручний інструмент для роботи з видатками і опосередковано впливає на підвищення фінансової грамотності.

Окрім контролю персональних витрат, не менш актуальним є питанням довготривалого збереження документів. Щоб скористатися своїми правами на гарантійне обслуговування, заміну або повернення товару, чи підтвердити факт оплати, споживач має пред’явити розрахунковий документ. Така необхідність змушує зберігати розрахункові документи в продовж тривалого часу. Спосіб традиційного фізичного збереження документів є трудомістким і не надійним, адже папір на якому друкуються чеки швидко зношується, а пошук необхідного документу займає багато часу. Зважаючи на розповсюдженість мобільних пристрої, зокрема з ОС Android споживачеві можливо запропонувати обробку і структуроване збереження розрахункових документів у цифровому вигляді [1].

Питання збереження розрахункових документів і контроль витрат, це дві споріднені проблеми. Тому доцільним і правильним було знайти одне спільне рішення для них, яким є додаток для контролю персональних витрат.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проблеми

Зараз багато українських родин часто опиняються в ситуації, коли сімейні кошти закінчуються раніше, ніж буде отримано чергову заробітну плату. Світовий банк провів опитування, що містило низку питань про фінансовий добробут як аспект якості життя. Так, понад 60% українців вважають рівень свого добробуту низьким чи надзвичайно низьким (рис.1.1) [2].

Рисунок 1.1 – Результати опитування «Фіндекс»

Частина тих, кому не вистачало доходу, щоб покрити витрати, вирішувала проблему, позичаючи кошти або відстрочуючи погашення боргів (30% опитаних). Позичали переважно у родичів і друзів, але дехто користувався кредитними картками або брав короткострокові кредити. Інша частина вдавалась виключно до зменшення витрат або отримання додаткового доходу [2].

Опираючись на результати дослідження можна зробити висновок, що для покращення матеріального становища часто не вистачає фінансової

грамотності. Зростання фінансового добробуту у значній мірі залежить від того, наскільки розумно людина використовує ресурси та розпоряджається грошима [3], іншими словами грамотне планування бюджету здатне значної мірою покращити фінансове становище як конкретної людини так і цілої родини. Фінансова грамотність і на основі неї правильно спланований бюджет завжди починається з аналізу витрат. Аналізуючи свої витрати людина знатимете точно, на що йдуть гроші, зможе спланувати великі покупки, зрозуміє, як заощадити та примножити гроші, а також захистить себе та свою родину від фінансових ризиків.

Фінансова грамотність стала темою дискусій останні роки [4]. Поняття фінансової грамотності можна розуміти як знання основних фінансових концепцій, але перш за все це вміння орієнтуватися на фінансовому ринку і ефективно розпоряджатися коштами чи фінансово забезпечити себе та свою родину. Це також уміння заощаджувати і правильно розподіляти господарські витрати [5].

Дослідження даних зі сфери економіки та охорони здоров’я 38 країн показали залежність благополуччя, фінансової мобільності і загального стану здоров’я населення від рівня їх фінансової грамотності. Дослідження показало значний вплив фінансової обізнаності на рівень бідності (рис. 1.2-A). Фінансова грамотність також має важливе значення для економічної мобільності країни (рис. 1.2-B). Вплив фінансової грамотності є явно позитивно лінійним як на загальний стан здоров’я, так і на задоволеність життям (рис. 1.3). Дане дослідження підтверджує актуальність фінансової обізнаності для економічного розвитку країни в цілому і для кожного окремо взятого громадянина зокрема.

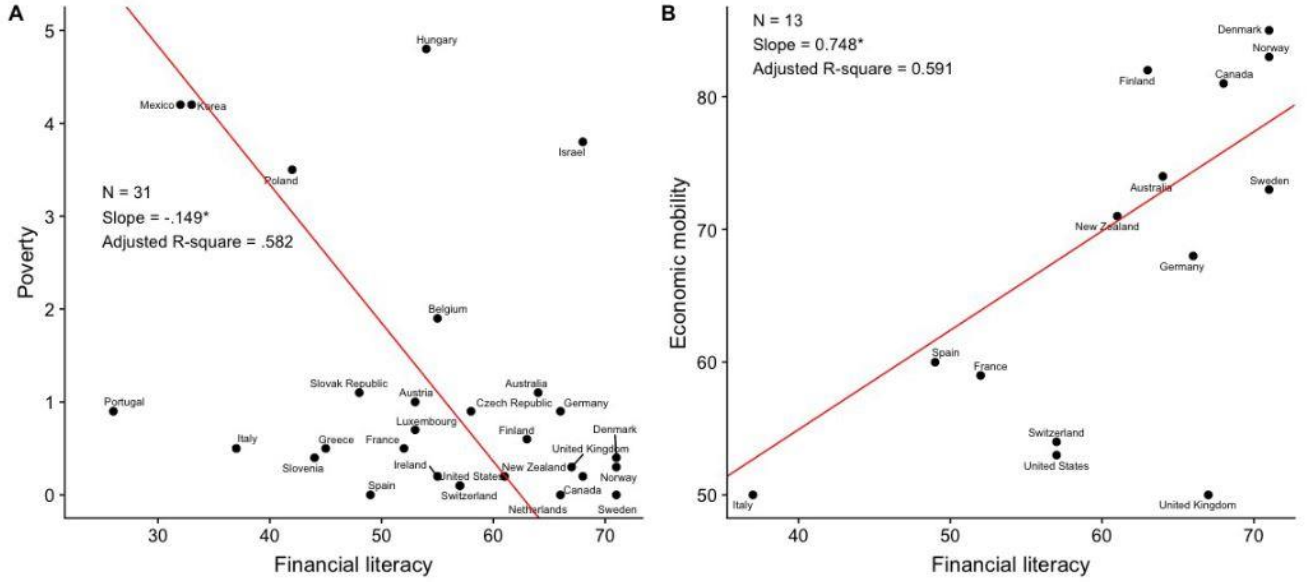


Рисунок 1.2 – Результати дослідження фінансової грамотності

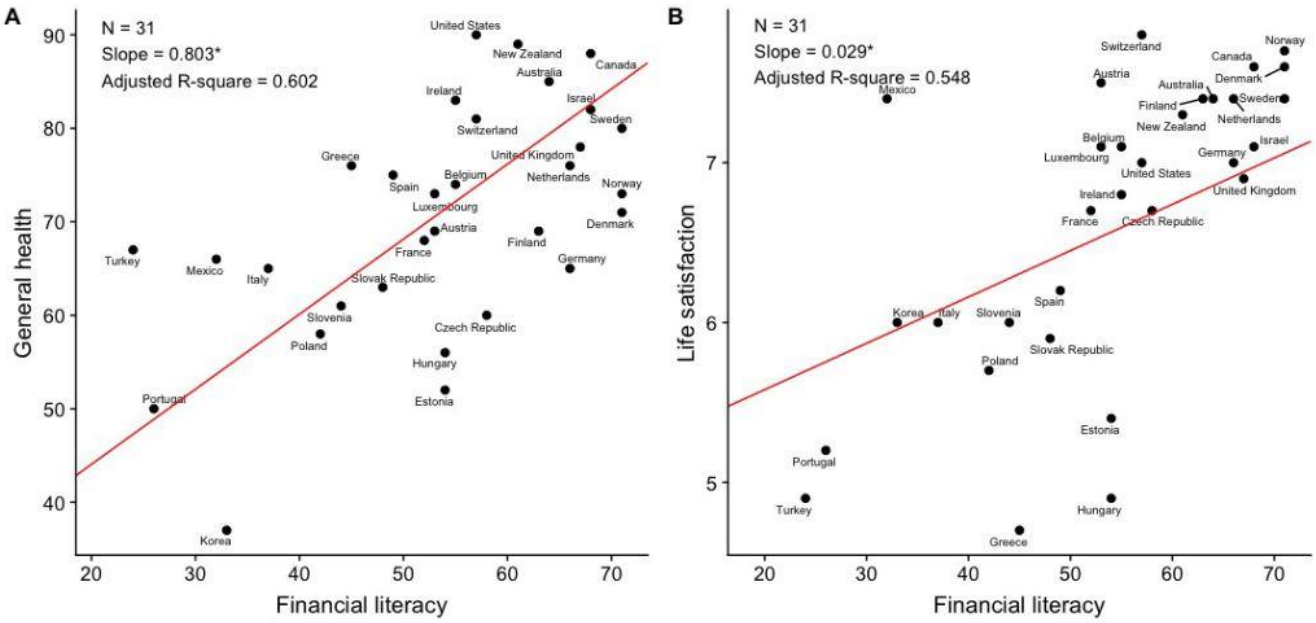


Рисунок 1.3 – Результати дослідження фінансової грамотності

Дане дослідження підтверджує актуальність фінансової грамотності для економічного розвитку країни [6],[7]. Також слід виділити питання саме цифрової фінансової грамотності, адже з розвитком цифрових технологій

люди повинні також ставати все більш обізнаними, щоб ефективно використовувати фінансові продукти і уникати махінацій і шахрайства [8], [9].

Дані про фінансову грамотність в країнах Східної Європи, зібрані Організацією економічного співпраці і розвитку (OECP) і Standard & Poor's Financial literation, показали, що фінансова грамотність населення в цьому регіоні в середньому нижче ніж в Західній Європі, також дані свідчать про більшу залежність обізнаності від статі і віку (рис. 1.4) [10].

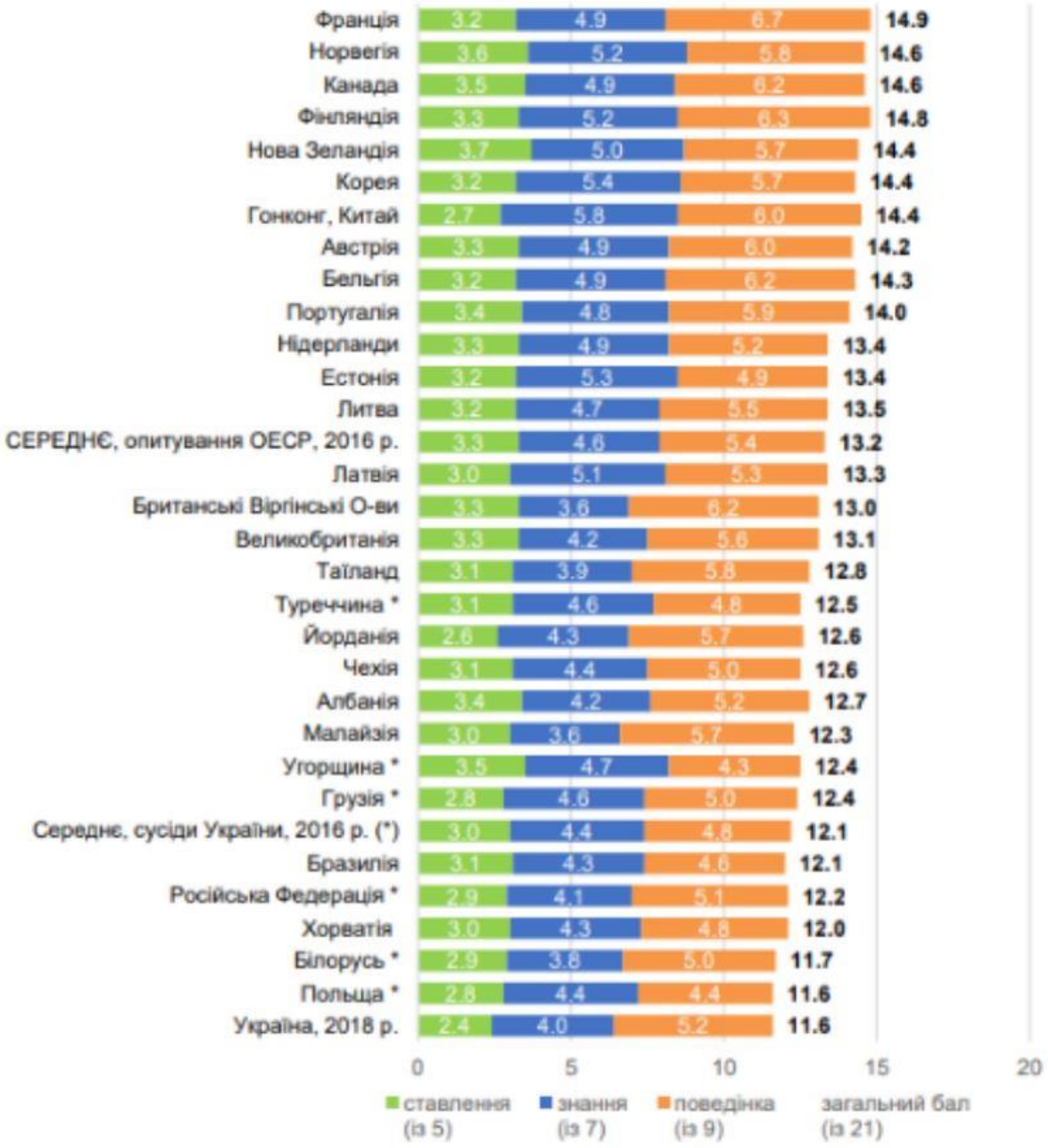


Рисунок 1.4 – Індекс фінансової грамотності

Зважаючи на важливість фінансової грамотності, як для окремо взятого громадянина так і для нації в цілому, і на її низький рівень в нашому регіоні актуальність створення системи направленої на підвищення фінансової освіти є очевидною і обґрунтованою.

Іншим, не менш важливим, завданням роботи є збереження документів, що підтверджують оплату товарів і послуг. Фактом, що підтверджує оплату товарів та послуг є розрахункові документи (чеки, квитанції тощо). З правової точки зору споживач не зобов'язаний зберігати платіжні документи, але для забезпечення права споживача на повернення, заміну чи гарантійне обслуговування товару передбаченого Законом України «Про захист прав споживачів», необхідно пред'явити розрахунковий документ продавцеві. Така необхідність є фактором, що спонукає споживача до довгострокового збереження цих документів [1], [11].

Випадки при яких виникає необхідність підтвердження оплати:

- при необхідності звернутися з претензією до якості товару чи послуги
- при обміні чи повернені товару
- при звернені з питань сервісного обслуговування
- необхідність збереження чеків, що підтверджують внесення коштів через платіжний термінал
- квитанції про оплату комунальних послуг необхідно зберігати протягом 3 років.

Таким чином, для забезпечення можливості захистити свої права, документи, що підтверджують оплату необхідно зберігати до моменту закінчення гарантійного строку товару, строку придатності товару, виконання зобов'язань послугодавця або закінчення строку позовної давності.

Для вирішення даного завдання можливо скористатися способом фізичного зберігання документу. Та такий спосіб має низку недоліків:

- папір на якому друкуються чеки – швидко псується;
- складність пошуку необхідного документу, що зростає зі збільшенням документів;
- необхідність налагодження спеціального місця для зберігання.

Зважаючи на необхідність довготривалого збереження чеків і низьку ефективність традиційного способу збереження, реалізація функції оцифрування розрахункових документів в розроблюваній системі є доцільною та актуальною.

Отже розроблюваний мобільний додаток буде надавати користувачам зручний інструмент для планування бюджету, та опосередковано підвищувати їх фінансову грамотність. А додавання функції оцифування дозволить в зручний спосіб зберігати документи, класифікувати їх, та забезпечувати швидкий доступ до раніше збережених даних.

Для досягнення поставлених задач додаток має відповідати наступним вимогам: система направлена на аналіз персональних витрат, має функцію оцифрування документів, орієнтована на користувачів з України, має відкритий доступ.

1.2 Огляд існуючих рішень

Наступним етапом роботи є дослідження ринку мобільних застосунків. Даний етап роботи направлений на детальний аналіз застосунків-аналогів з метою вивчення їх слабких та сильних сторін. Основними конкурентами розроблюваного застосунку є такі додатки: Expensify, Receipt, Foreceipt і Receipt Lens. Далі розглянемо ці застосунки більш детально:

Expensify

Платформа: iOS, Android

Переваги:

- розпізнавання чека

- формування фінансових звітів в pdf форматі
- можливість вносити витрати в ручному режимі
- можливість додавання раніше збережених документів
- можливість бронювання через додаток квитків, готелю тощо

Недоліки:

- деталізація при розпізнавані чеку (розпізнає тільки кінцеву суму і назву торгівельного закладу)
- мова додатку англійська
- безкоштовний період лише 7 днів

Посилання на додаток в GooglePlay:

https://play.google.com/store/apps/details?id=org.me.mobiexpensifyg

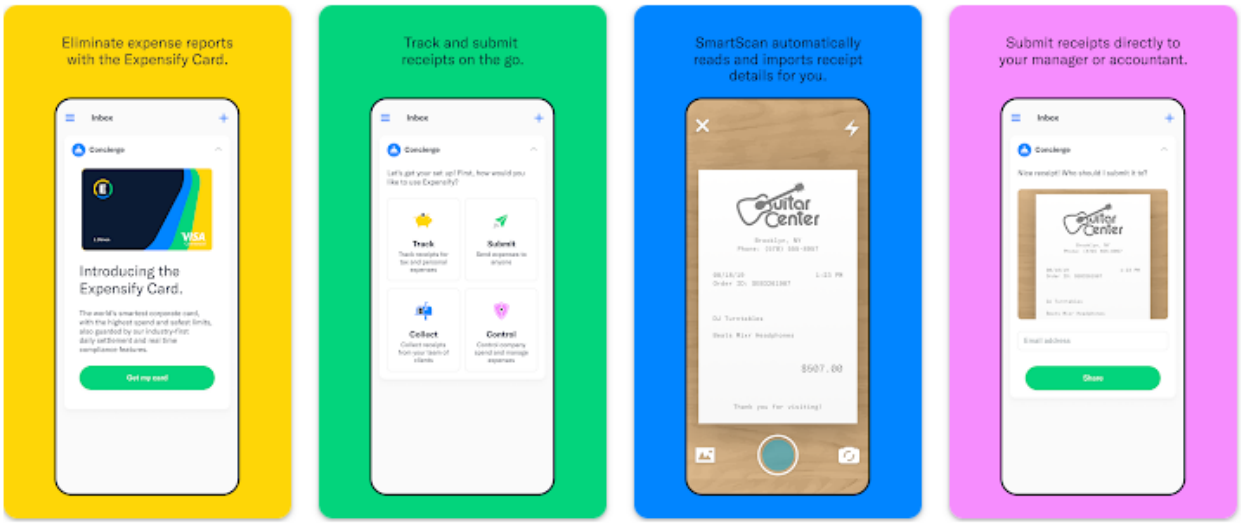


Рисунок 1.5 - Екрани додатку Expensify

Receipt

Платформа: Android

Переваги:

- формування фінансових звітів в pdf форматі з можливістю відправки звітів безпосередньо з додатка
- функція розпізнавання чеків

- можливість вносити витрати в ручному режимі
- можливість додавання раніше збережених документів
- мультивалютність з автоконвертацією валюти при формуванні звіту

Недоліки:

- деталізація при розпізнавані чеку (розпізнає тільки кінцеву суму і назву торгівельного закладу)
- мова додатку англійська
- відсутнє графічне представлення статистики витрат
- для формування звіту необхідно власноруч відмітити витрати, що увійдуть до звітності
- обмежені можливості використання offline
- безкоштовний період лише 3 днів

Посилання на додаток в GooglePlay: https://play.google.com/store/apps/details?id=saldo.receiptscanner.app

Рисунок 1.6 - Екрани додатку Receipt

Foreceipt - Receipt Scanner Ex

Платформа: Android

Переваги:

- додаток самостійно визначить категорію витрат і занесе дані
- функція розпізнавання чеків
- формування звітів в pdf та excel форматі
- зручний інтерфейс

Недоліки:

- деталізація при розпізнавані чеку
- обмежені можливості використання offline
- обмеження функцій в безкоштовній версії
- мова додатку англійська

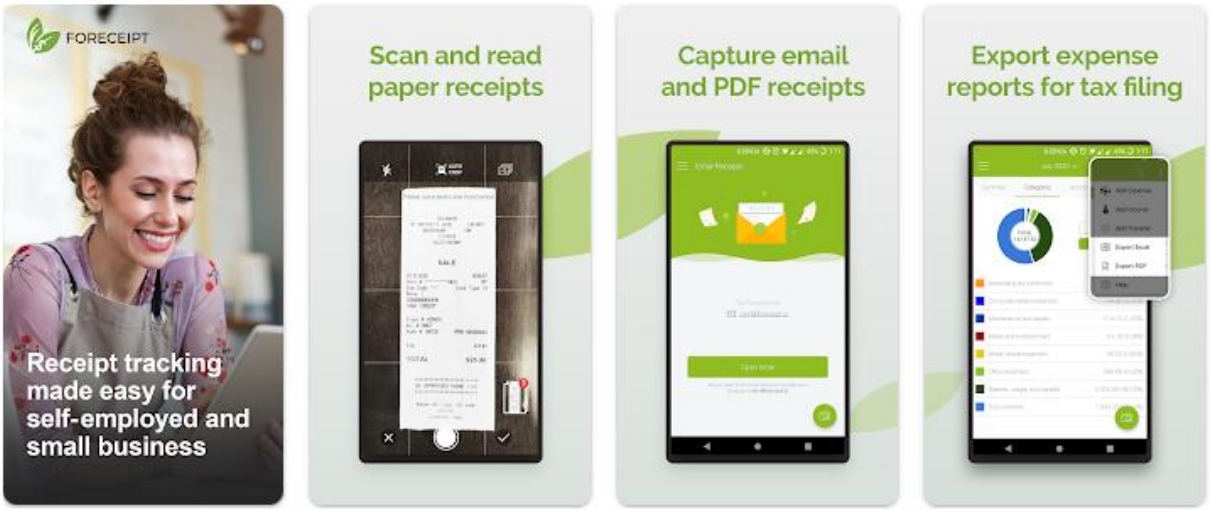


Рисунок 1.7 - Екрани додатку Foreceipt

Посилання на додаток в GooglePlay:
https://play.google.com/store/apps/details?id=com.foreceipt.android.cloud

Receipt Lens

Платформа: Android

Переваги:

- додаток самостійно визначить категорію витрат і занесе дані
- функція розпізнавання чеків

- формування і експорт звітів
- зручний інтерфейс

Недоліки:

- деталізація при розпізнавані чеку
- обмежені можливості використання offline
- обмеження функцій в безкоштовній версії
- мова додатку англійська

Посилання на додаток в GooglePlay:

https://play.google.com/store/apps/details?id=com.glority.receipt

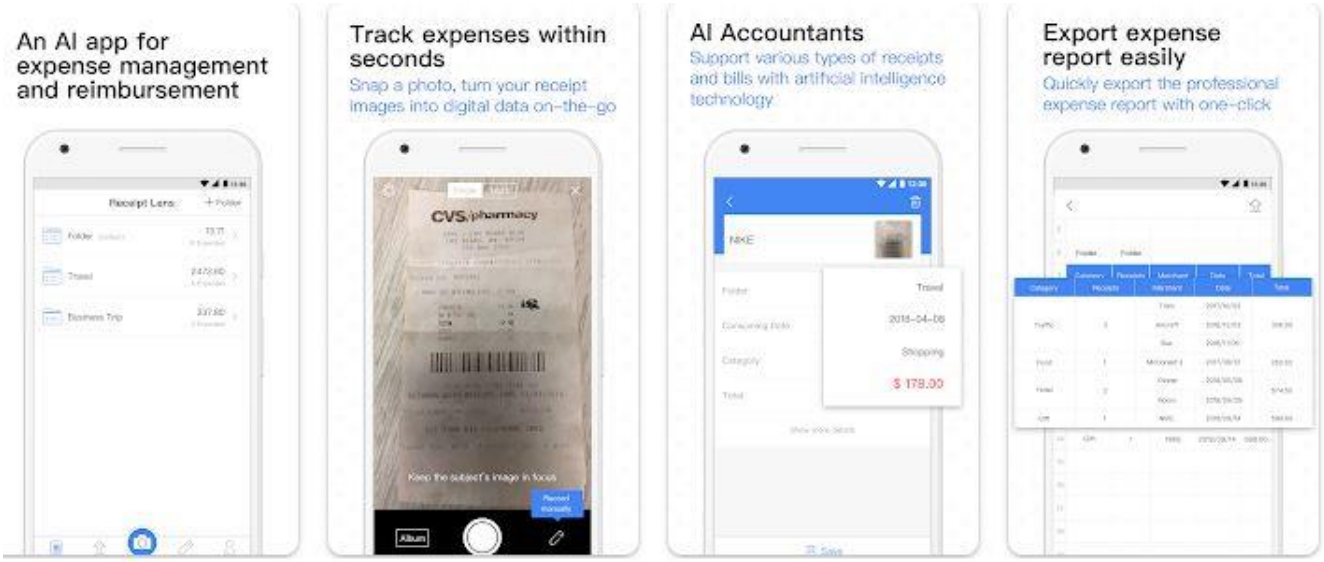


Рисунок 1.8 - Екрани додатку Receipt Lens

Таблиця 1.1 – Порівняння застосунків - аналогів

Характеристики	Expensify	Receipt	Foreceipt	Receipt Lens
Зручний спосіб формування аналітики	-	-	+	-
Візуалізована аналітика витрат	-	-	+	-

Структуроване зберігання документів	+	+	-	+
Наявність української мови	-	-	-	-
Коректна робота в offline режимі	-	-	-	-
Безкоштовний доступ до послуг	-	-	-	-
Зручний інтерфейс	+	-	+	+
Наявність фільтру для пошуку витрат	-	-	-	-

Дослідивши існуючі рішення можна зробити висновок, що досліджувані додатки мають широкий функціонал і зручні в використанні. Але всі вони мають недоліки:

- Платний доступ до окремих функцій чи короткий пробний період після якого додаток доступний тільки по платній підписці;
- Мова цих застосунків – англійська;
- А функція розпізнавання чека коректо працює лише з доступом додатка до інтернет.

Ще одним важливим етапом роботи є дослідження потреб користувачів. Мета такого дослідження направлена на визначення їх вподобань, звичок і очікувань від розроблюваного продукту.

Потужним і доступним механізмом вивчення потреб користувачів є аналіз їх відгуків, які вони лишають до застосунків з подібним функціоналом в Google Play.

Були проаналізовані відгуки до найбільш популярних застосунків подібної тематики з плей маркету. Перелік цих застосунків наведено в таблиці 1.2.

Таблиця 1.2 – Популярні додатки схожої тематики

№	Назва додатку	Кількість завантажень
1.	CamScanner	100 млн. +
2.	Microsoft Office Lens	10 млн. +
3.	Receipt Cat	100 тис. +
4.	Receipt Lens	10 тис. +
5.	Expensify	1 млн. +
6.	DocStorer	50 тис. +
7.	Receipt	10 тис. +
8.	Foreceipt	10 тис. +

В результаті дослідження було встановлено, що користувачі частіше незадоволені роботою функції розпізнавання чеку, платністю застосунків, відсутністю української мови, орієнтованість застосунків на іноземних користувачі, складністю роботи в застосунку, не інформативністю або відсутністю візуалізованої аналітики витрат, тощо.

Проаналізувавши предметну область було встановлено необхідність проведення додаткових досліджень з метою визначення оптимальних засобів, методів і інструментів для вирішення поставлених завдань.

2. ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Метою кваліфікаційної роботи є створення мобільного додатку контролю персональних витрат з функцією збереження розрахункових документів.

Задачі даної роботи:

- Вивчення предметної області;
- Формування вимог до розроблюваного застосунка;
- Вибір методів розробки;
- Проектування додатка;
- Розробка та тестування застосунка.

В попередньому розділі роботи було досліджено предметну область. В результаті досліджень було встановлено, що на ринку мобільних застосунків відсутні україномовні додатки, що поєднують в собі функцію контролю видатків і збереження документів. Також було встановлено, що всі додатки, які мають необхідний функціонал, або платні, або обмежують можливості користувача в безкоштовних версіях. Ще одним важливим недоліком застосунків-аналогів є некоректна робота в режимі offline. На основі отриманих результатів було сформовано функціональні вимоги до мобільного застосунку:

- Безкоштовний доступ до всіх функцій;
- Мова застосунку - українська;
- Можливість створювати/редагувати/видаляти витрати;
- Зберігати історію витрат;
- Можливість оцифрування розрахункових документів;
- Можливість зберігати /редагувати/видаляти розрахункові документи;
- Зберігання документів має бути структурованим;
- Можливість створювати категорії витрат;

- Наявність калькулятору;
- Відображати структуру витрат у вигляді діаграм;
- Можливість обрати період витрат для аналізу;
- Можливість персональних налаштувань (вибір валюти, формату дати).

2.2 Вибір методу рішення

Згідно з даними, отриманими від Statista, кількість людей, які користуються мобільними телефонами з доступом до інтернет, в 2021 році досягла 6,259 мільярда осіб, а до кінця 2022 має налічувати 6,567 мільярдів користувачів, або 83% населення світу, це на 308 мільйонів користувачів більше, ніж у 2021 році (рис. 2.1). Стрімке збільшення користувачів призводить до невпинного зростання попиту на додатки до мобільних телефонів [12].

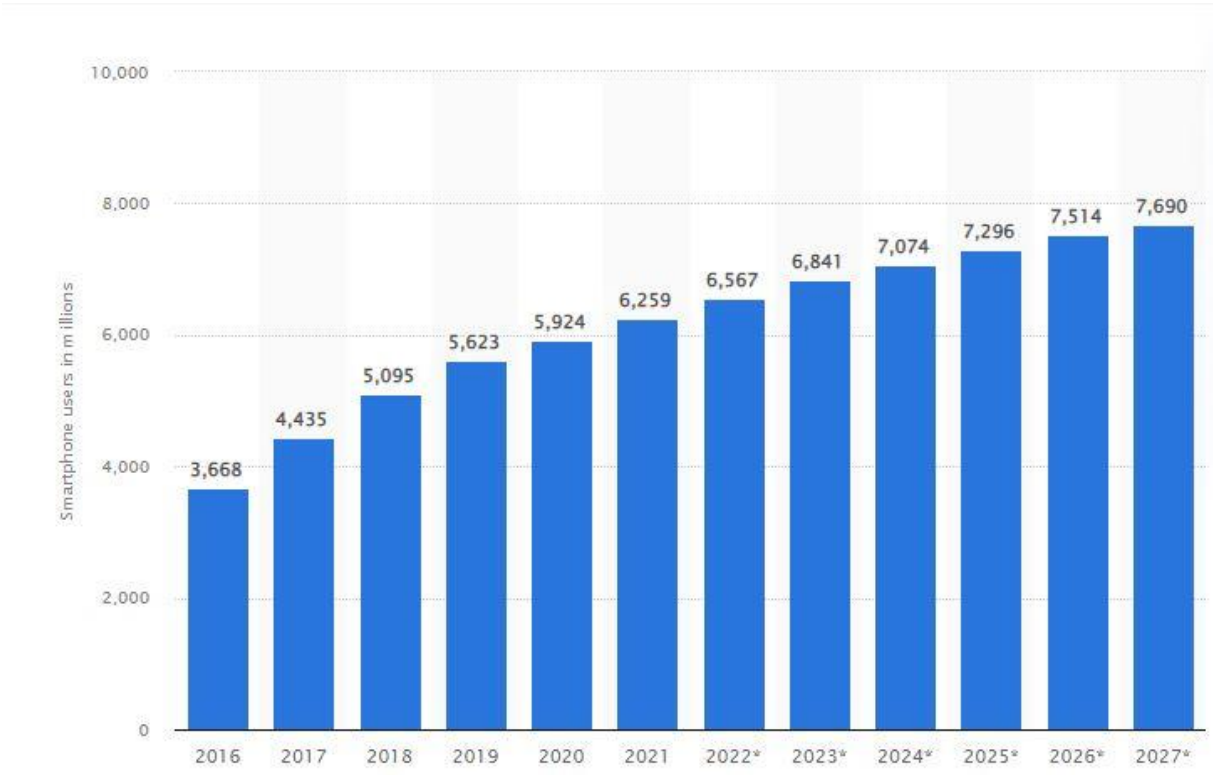


Рисунок 2.1 – Кількість користувачів телефонів з доступом до інтернет

У липні Cisco AppDynamics опублікувала звіт App Attention Index, в якому видно зростання споживання мобільних додатків – у 2021 році він на 30% більше, ніж у 2020. При цьому користувачі стали вибагливішими у виборі додатків. Все частіше вони звертають увагу на продуктивність та зручність, ніж на основний функціонал. За даними аналітиків, споживачі все більше потребують додатків, які об'єднують кілька сервісів [13].

Опираючись на аналітичні дані, які свідчать про те, що попит на додатки невпинно зростає, було прийнято рішення створити саме мобільний додаток для вирішення поставлених в роботі задач. Також враховуючи потреби користувачів в універсальних додатках, вирішення обох поставлених задач і збереження розрахункових документів і аналізу персональних видатків, буде поєднано в одному додатку.

2.2.1 Вибір операційної системи

Наступним кроком потрібно визначитись із операційною системою на якій буде створено додаток. Android від Google та iOS від Apple домінують у галузі операційних систем (ОС) для смартфонів. Згідно з даними Statista сьогодні 7 із 10 телефонів працюють на ОС Android (рис. 2.2)[14].

Хоча за останні 5 років Android і втратив у рейтингу близько 8%, але являється безумовним лідером серед операційних систем для мобільних телефонів. Тому розробку мобільного додатку доцільно здійснювати саме на платформі Android.

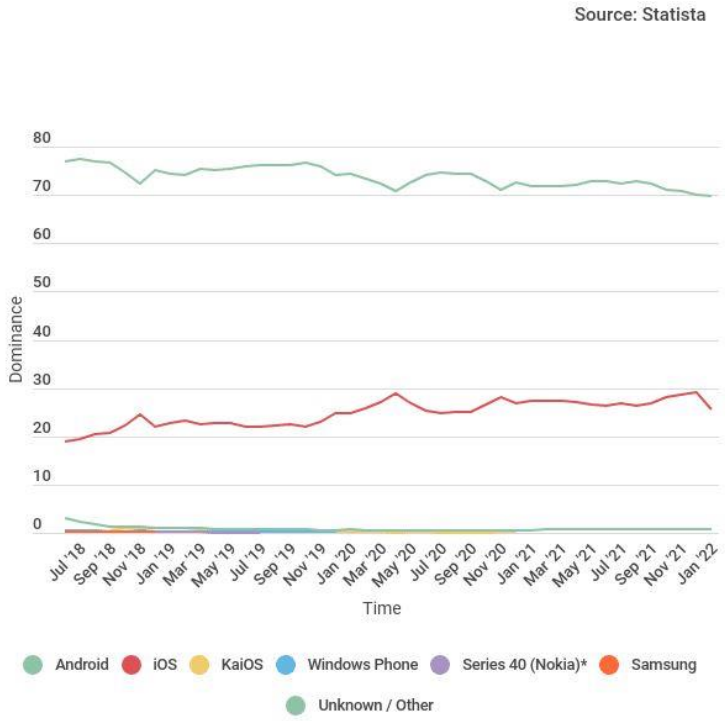


Рисунок 2.2 - «Рейтинг операційних систем для мобільних телефонів»

2.2.2 Вибір середовища розробки

Наступним етапом розробки мобільного додатку є вибір середовища розробки. Найбільш популярними платформами для розробки є офіційне середовище Android Studio і Xamarin.

Особливості та переваги Android Studio:

Android Studio - це офіційне середовище розробки (IDE) від Google для створення додатків на ОС Android (рис. 2.3).

Основні перевагами Android Studio:

- Створене розробниками ОС Android;
- Гнучка система збирання;
- Потужний редактора коду, який максимально спрощує процес розробки;
- Багатофункціональний і швидкий емулятор;
- Єдине середовище для розробки для різних пристроїв на ОС Android;

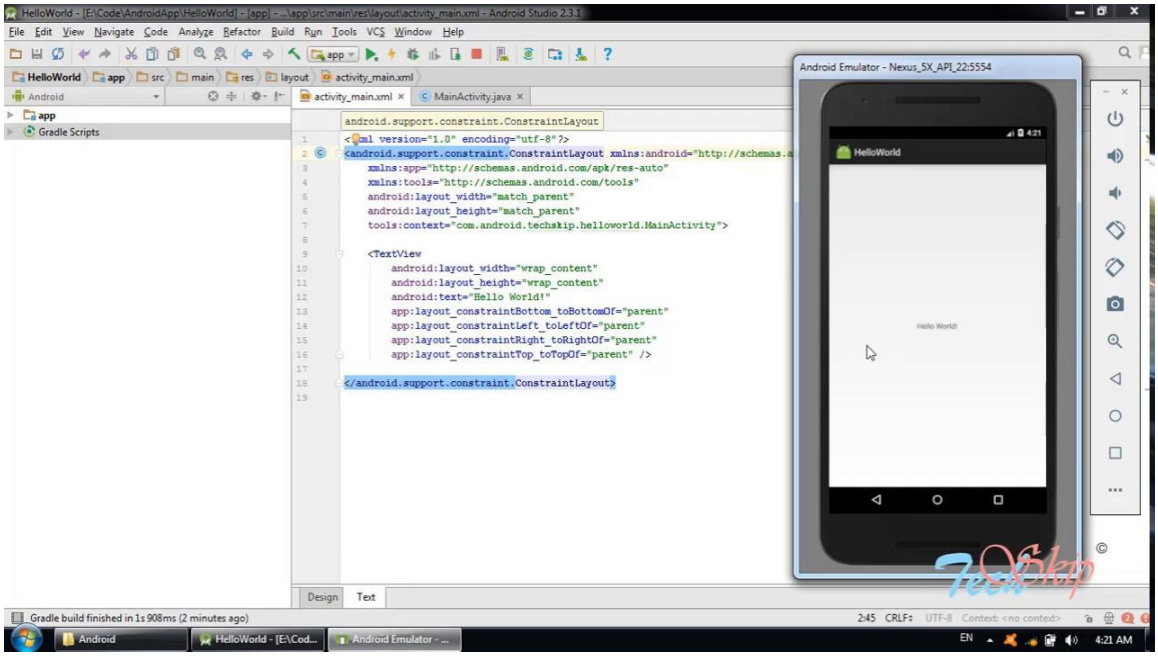


Рисунок 2.3 – Android Studio

- Застосування зміни для push-коду та змін ресурсів у працюючій програмі відбуваються без перезапуску програми;
- Можливість використання шаблонів та інтеграція з GitHub;
- Широкий вибір інструментів та фреймворків для тестування;
- Інструменти Lint для діагностики проблем з сумісністю версій, продуктивністю, зручністю використання, тощо;
- Об’єктно-орієнтовані мови програмування Java, Kotlin і C++;
- Підтримка Google Cloud Platform, що надає можливості зручної інтеграції Google Cloud Messaging та App Engine;
- Можливості роботи з UI компонентами за допомогою Drag-and-Drop;
- Генерація декількох .apk файлів та різноманітні види збірок;
- Утиліта ProGuard для підписки додатків;
- Структурована і вичерпна документація;
- Підтримка розробки додатків для Android TV і Android Wear [15].

Основним конкурентом Android Studio є кросплатформене середовище Xamarin (рис. 2.4). Xamarin – це фреймворк, що надає можливості розробки

мобільних додатків не тільки для Android, а також і для iOS. Мова програмування, що використовується для розробки в цьому середовищі - C#.

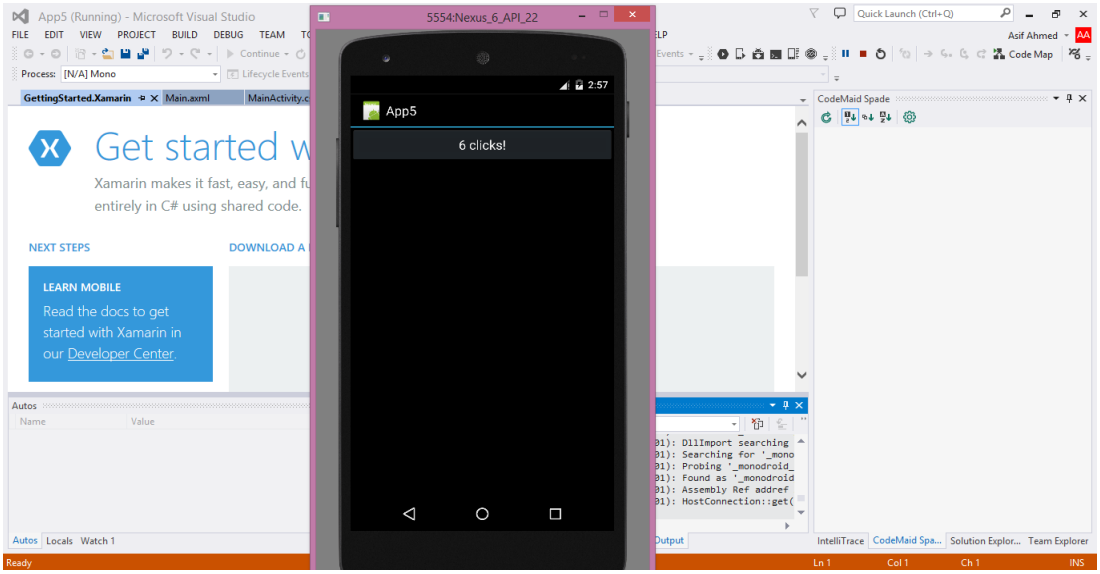


Рисунок 2.4 - Xamarin

Фреймворк складається з декількох основних частин:

- Xamarin.IOS – вбудована бібліотека класів для C# для роботи з iOS SDK;
- Xamarin. Android – вбудована бібліотека класів для C# для роботи з Android SDK;
- iOS і Android компілятори;
- Плагін для Visual Studio.

Основною перевагою Xamarin, в порівнянні з Android Studio – це кросплатформеність. Якщо планується розробка додатка на обох ОС і iOS, і Android, то перевага безумовно надається Xamarin. В такому випадку Xamarin дозволяє зекономити купу часу, адже ви будете створювати лише один варіант додатку, який буде працювати на обох операційних системах [16].

Зважаючи на те, що додаток створюється тільки для пристроїв з ОС Android, більш доцільним є використання Android Studio. Android Studio має такі суттєві переваги над Xamarin, як:

- Більш зручний редактор XML;
- Зручне додавання сторонніх бібліотек;
- Детальна документація по роботі з середовищем;
- Android Studio – це офіційне середовище, яке постійно оновлюється.

Розробка додатку завжди пов'язана з ризиком випадкової зміни структури додатку чи дизайну, або необхідністю повернути попередній варіант коду із-за помилок в новому, тощо. Для того, щоб захистити себе від таких неприємних ситуацій доцільно використовувати репозиторії. Найбільш популярними і зручними є сервіси GitHub і Bitbucket. Ці два сервіси мають схожі можливості і не поступаються в зручності. Та зважаючи на більший досвід роботи з репозиторієм GitHub перевагу було надано саме цьому сервісу.

Для реалізації функціональних вимог додатка були залучені і сторонні бібліотеки, які знаходяться у відкритому доступі на сайті GitHub. Перелік цих бібліотек і їх короткий опис:

- Parceler – це бібліотека, що використовується для генерації коду Android Parcelable. Parcelable – це інтерфейс, який дозволяє передавати складні об'єкти між Activity. Посилання на репозиторій: https://github.com/johncarl81/parceler;
- google-gson() – це бібліотека, що використовується для перетворення об'єктів Java в JSON- представлення. Надає можливості використовувати для перетворення строки JSON в об'єкт Java. Посилання на репозиторій: https://github.com/google/gson;
- Stetho – це бібліотека від Фейсбук, що надає можливості швидкого налагодження додатку. Основними перевагами даної бібліотеки є можливість відстеження мережевої активності і простого та зручного перегляду ієрархії додатку, що реалізується за допомогою інструменту Chrome DevTools. Посилання на бібліотеку: http://facebook.github.io/stetho/;

- Butter Knife – це бібліотека, що використовується для ініціалізації візуальних об’єктів, таких як ImageView, TextView, EditText, тощо. Основними перевагами даної бібліотеки над вбудованою є значне спрощення роботи з елементами, що значною мірою пришвидшує темп розробки. Посилання на бібліотеку: https://github.com/JakeWharton/butterknife;

Робота даних бібліотек не розповсюджується на модуль графічного редактора і використовується лише для Дані бібліотеки були підключені в модулі додатку і не розповсюджуються. Модуль графічного редактора використовує тільки вмонтовані бібліотеки від компанії Google котрі постачаються разом з SDK

Перечисленні бібліотеки були підключені лише в модулі додатку і не впливають на роботу модуля графічного редактора. Для цього модуля використовуються лише вбудовані бібліотеки від компанії Google, що постачаються разом з SDK.

2.2.3 Вибір версії Android SDK і мови програмування

Наступним етапом дослідження методів розробки є вибір версії Android SDK на якій буде проводитись розробка додатка. Android SDK – це пакет розширень Android, який надає додаткові, зручні інструменти розробки. За допомогою цього пакету можна відслідковувати стан операційної системи, читати лог-файли, створювати віртуальні пристрої для тестування роботи додатка на різних версіях ОС і багато іншого [17]. Від версії SDK залежить і версія операційної системи мобільного додатка. Для розробки було обрано версію SDK – 33. Операційна система пристроїв для роботи з додатком має бути версії 10 і вище. За даними API version distribution, що надає Android Studio, кількість пристроїв, які працюють на операційній системі версії 10 і вище – 62,8% (рис. 2.5) [18].

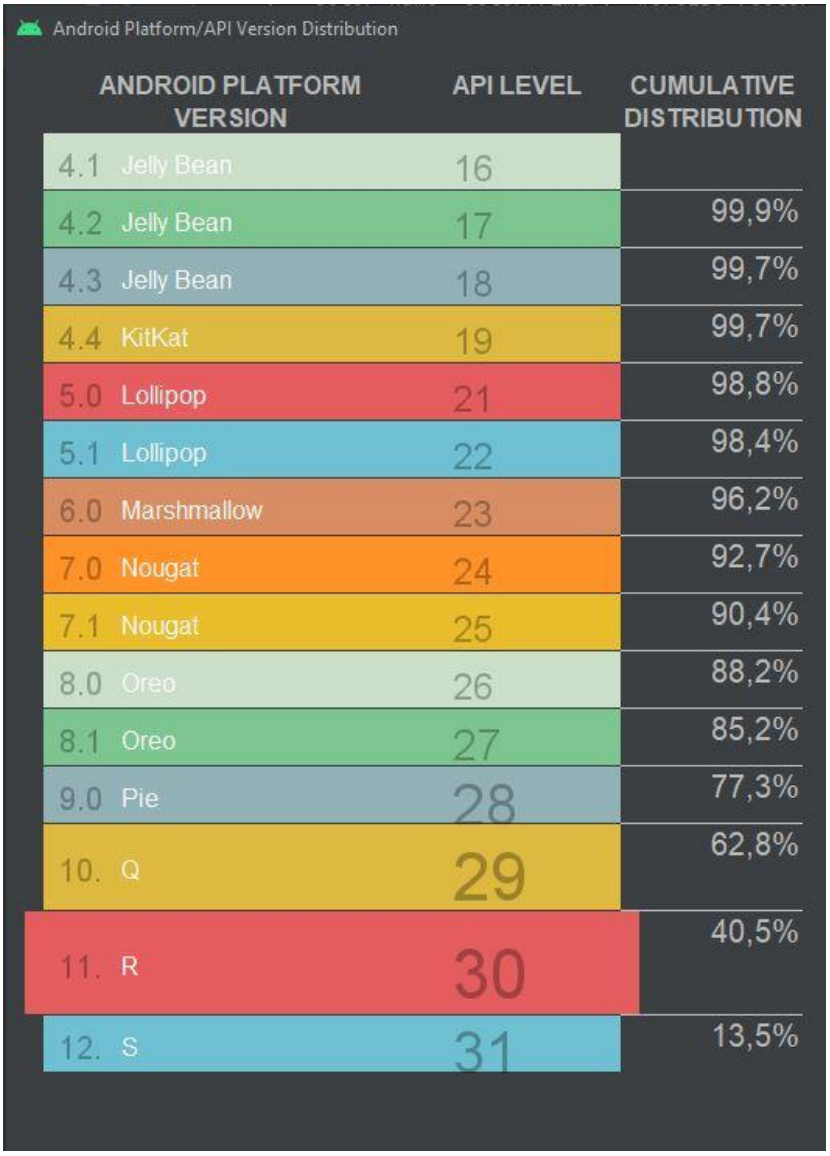


Рисунок 2.4 – Розподіл версій ОС серед користувачів

Зважаючи на те, що нові версії виходять часто, а старі версій так же швидко втрачають актуальність, було надано перевагу не самій популярній у користувачів версії (62,8%), але новій з перспективою, що кількість пристрої буде збільшуватися.

Середовище Android Studio підтримує такі мови програмування:

- Java;
- Kotlin;
- C++.

Всі ці мови програмування є об’єктно-орієнтованими і високопродуктивними і не мають суттєвих переваг в контексті роботи над даним мобільним додатком. Тому було обрано мову Java зважаючи на більший досвід роботи з нею.

2.2.4 Вибір бази даних

Розроблюваний додаток буде інтенсивно працювати з базою даних, тому вибору БД слід приділити особливу увагу.

Планується, що дані будуть зберігатися на пристрої користувача, найбільш підходящими, в даному випадку, базами даних є Realm і SQLite.

SQLite – це вбудована в Android Studio реляційна база даних. Вона є однією з найпопулярніших і простих систем керування даними. Та SQLite має деякі обмеження. Запити на мові SQL можуть бути повільними і зі збільшенням обсягу стає все важче керувати даними. Для спрощення і покращення роботи з SQLite існує високорівневий інтерфейс Room вбудований в Android Studio. Часто в розробці мобільних додатків SQLite і Room використовуються в парі [19].

Realm – це проста у використанні альтернатива SQLite з відкритим кодом. Realm – нереляційна база даних, яка дозволяє встановлювати зв’язки між різними об’єктами. Головною перевагою Realm над SQLite є більш швидка робота з даними. Та Realm має ряд обмежень. Основним з них є складність створення запитів, що вимагають об’єднання більш ніж 4 - 5 таблиць [19].

Нижче наведено графік (рис.2.6), що порівнює швидкість виконання операцій CRUD для SQLite, Room і Realm.

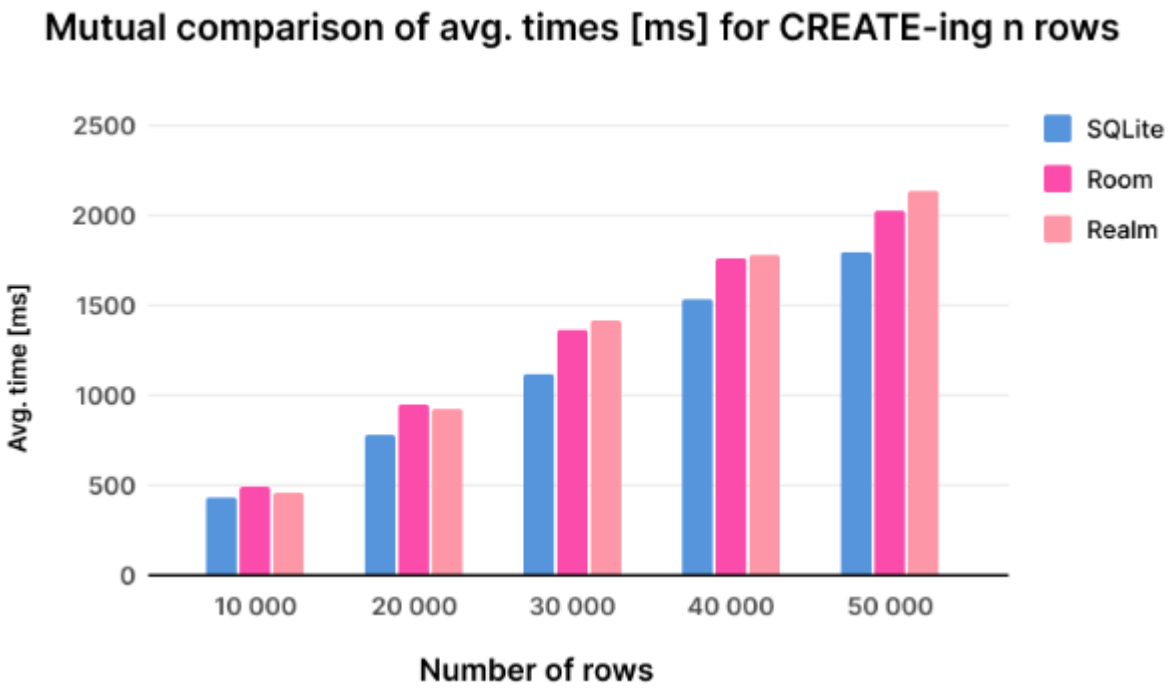


Рисунок 2.6 – Швидкість створення n – записів

Розроблюваний додаток не потребує створення великої кількості таблиць, але буде часто робити запити до даних. В даному випадку доцільно обрати БД Realm враховуючи її перевагу в швидкості роботи над SQLite.

2.2.5 Вибір засобів створення діаграм

Для створення діаграм потрібно використовувати сторонню бібліотеку. Було обрано для порівняння 2 найбільш популярних бібліотеки з відкритим кодом: MPAndroidChart і AnyChart [20], [21]. Порівняння їх можливостей наведено в таблиці 2.1.

Таблиця 2.1 – Порівняння бібліотек роботи з діаграмами

Характеристики	MPAndroidChart	AnyChart

Можливість створення кругових і стовпчастих діаграм	+	+
Можливості створення власних схем кольорів	+	+
Вбудовані шаблони кольорових схем	+	+
Анімація діаграм	+	-
Якісна документація	+	-
Можливості додавання лімітних ліній	+	+
Можливості збереження діаграм як рисунка	+	-

Як видно з таблиці 2.1 бібліотека MPAndroidChart має більше можливостей, тому саме ця бібліотека буде використовуватися для створення діаграм.

Дослідження екосистеми мобільних додатків показує, що в більшості випадків безкоштовні мобільні додатки, які орієнтовані на отримання прибутку від реклами, приносять більший дохід ніж платні аналоги. Опираючись на це дослідження, а також на зацікавленість користувачів у безкоштовному продукті, було обрано стратегію отримання доходу від реклами. Тому розроблюваний мобільний додаток буде безкоштовним, також не буде містити додаткових платних функцій [22].

Так як розроблюваний додаток буде використовувати монетизацію від реклами як основне джерело доходу, то необхідно подбати про оптимізацію показу реклами з метою отримання максимального доходу. Дослідження оптимізації показу реклами в мобільних додатках показало залежність частоти оновлення рекламних банерів від кількості кліків. За результатами дослідження було встановлено, що мінімально ефективний час показу реклами становить 1 хв і зі збільшенням часу показу до 2 хв. 30 сек. кількість кліків збільшується, після 3 хв. зростання кліків стає не суттєвим, тому оптимальний час показу становить ≈ 2 хв. Окрім покращення показників ефективності для розробника мобільного додатку, оптимальний час оновлення оголошень зменшує навантаження від рекламних викликів на пристрої користувача та на серверах постачальників реклами [23].

На основі проведених досліджень було сформовано нефункціональні вимоги до мобільного додатка:

- Операційна система додатка – Android;
- Середовище розробки - Android Studio;
- Версія SDK – 33;
- Версія операційної системи пристроїв - Android 10 і вище;
- БД Realm;
- MPAndroidChart для створення діаграм;
- Стратегія отримання прибутку – реклама.

3. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Концептуальне проектування

На основі сформованих вимог було проведено моделювання роботи додатка з використанням методології IDEF0. Згідно вимог до розроблюваного додатка, основними його функціями є отримання і збереження інформації про персональні витрати від користувача і формулювання аналітики створеної на основі цих даних, а також оцифрування і збереження розрахункових документів. Таким чином, визначимо головний процес контекстної діаграми, як «Контроль персональних витрат». Діаграма містити вхідні і вихідні дані, а також управління і механізми (рис 3.1).

Рисунок 3.1 – Контекстна діаграма процесу «Контроль персональних витрат»

Розглянемо детально ці елементи:

- Вхідні дані – це інформація, яку надає користувач. На основі обробки цієї інформації системою будуть сформовані вихідні дані. До вхідних даних належать: інформація про витрати, розрахункові документи;
- Вихідні дані – це кінцевий результат, який має отримати користувач в результаті взаємодії з додатком. До вихідних даних належать: сформована історія витрат, архів збережених документів і аналітичні дані створені на основі інформації про витрати;
- Управління – елементом управління в роботі додатка є база даних. БД очікує від користувача правильно і в повному обсязі заповнених полів при створенні нової витрати, чи нового документа. Реакцією на помилку користувача буде відмова в збереженні інформації.
- Механізмам – це сам користувач, який взаємодіє з додатком.

Далі проведемо декомпозицію процесу «Контроль персональних витрат», описавши більш детально роботу додатка (рис. 3.2).

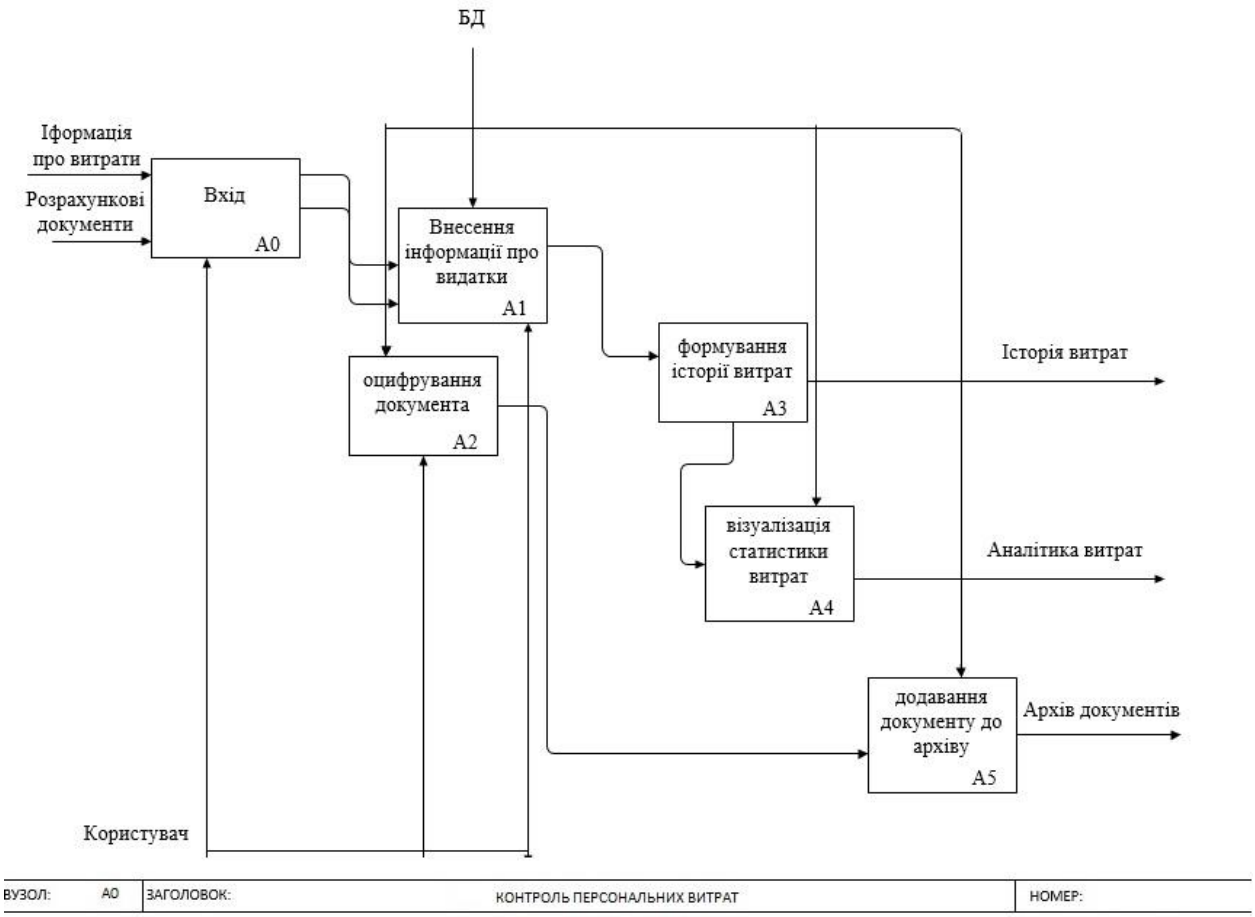


Рисунок 3.2 – Декомпозиція процесу «Контроль персональних витрат»

Розглянемо детально послідовність дій:

- після відкриття додатка користувач може ввести дані про видатки;
- користувач може оцифрувати документ;
- правильність введення даних в цих випадках контролює БД;
- на основі внесених даних формується історія витрат і архід документів;
- на основі історії витрат користувач отримує аналітику даних.

Далі проведемо декомпозицію блока 1 «Внесення інформації про видатки» і блока 2 «Оцифрування документа» (рис. 3.3, 3.4).

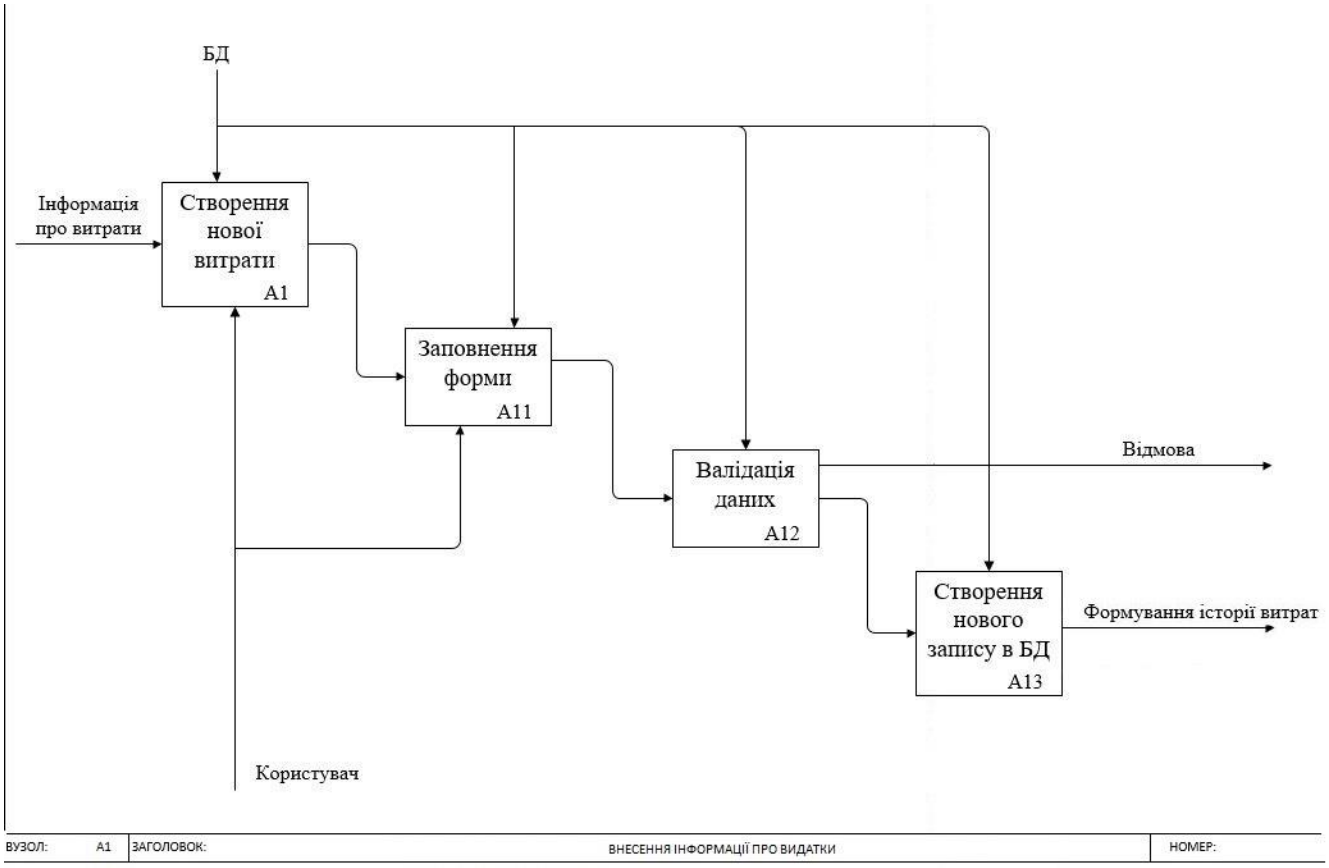


Рисунок 3.3 – Декомпозиція процесу «Внесення інформації про видатки»

Розглянемо послідовність дій при внесенні інформації про витрати:

- Першим, що робить користувач, це ініціює процес створення нової витрати, натиснувши на відповідну кнопку;
- Відкривається форма для заповнення інформації про витрату;
- Проходить перевірка правильності введення даних, на даному етапі користувач може отримати відмову в збережені даних в разі некоректного заповнення полів;
- На основі вірно внесених даних створюється новий запис в БД;
- Вихідними даними даного блоку є формування історії витрат.

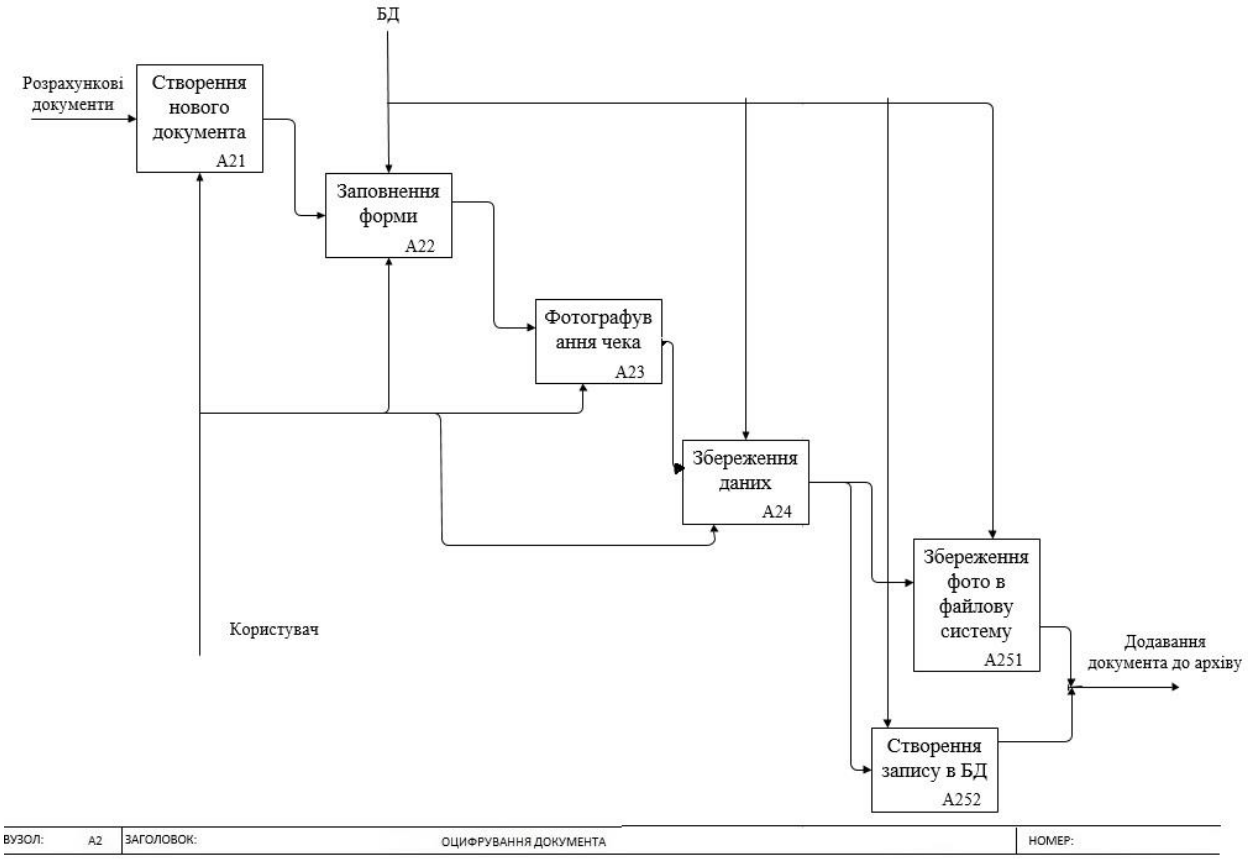


Рисунок 3.4 – Декомпозиція процесу «Оцифрування документа»

Розглянемо послідовність дій при оцифруванні документів:

- Як і при створенні витрат, першим, що робить користувач, це ініціює процес натисненням кнопки;
- Заповнює форму;
- Робить знімок документу;
- Збереження запису:
    - Фото зберігається до файлової системи;
    - Дані з форми зберігаються в БД;
- Процес контролює БД;
- Вихідними даними буде додавання нового документа до архіву.

На даному етапі, коли змодельовані процес додатка, створюємо діаграму варіантів використання. Діаграма варіантів використання складається з

множини акторів і варіантів використання обмежених межею системи, представленою у вигляді прямокутника. Діаграма демонструє асоціації між акторами та варіантами використання, відношення між варіантами використання і узагальнені відношення між акторами. Суть діаграми прецедентів полягає в тому, що проектована система подається у вигляді множини сутностей або акторів, що взаємодіють із системою за допомогою варіантів використання. Варіант використання застосовується для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система під час діалогу з актором [24]. Діаграма використання, розроблюваного мобільного додатка, наведена на рисунку 3.5.

З рисунка 3.5 видно, що при вході в додаток користувач має доступ до таких функцій:

- Створення нового документа;
- Створення нової витрати;
- Перегляд історії витрат;
- Перегляд деталей витрати;
- Головне меню;
- Меню налаштувань;
- Калькулятор.

Також з діаграми видно, як здійснюється перехід від одної одного процесу до іншого і взаємозв’язок та залежність процесів розроблюваного додатка.

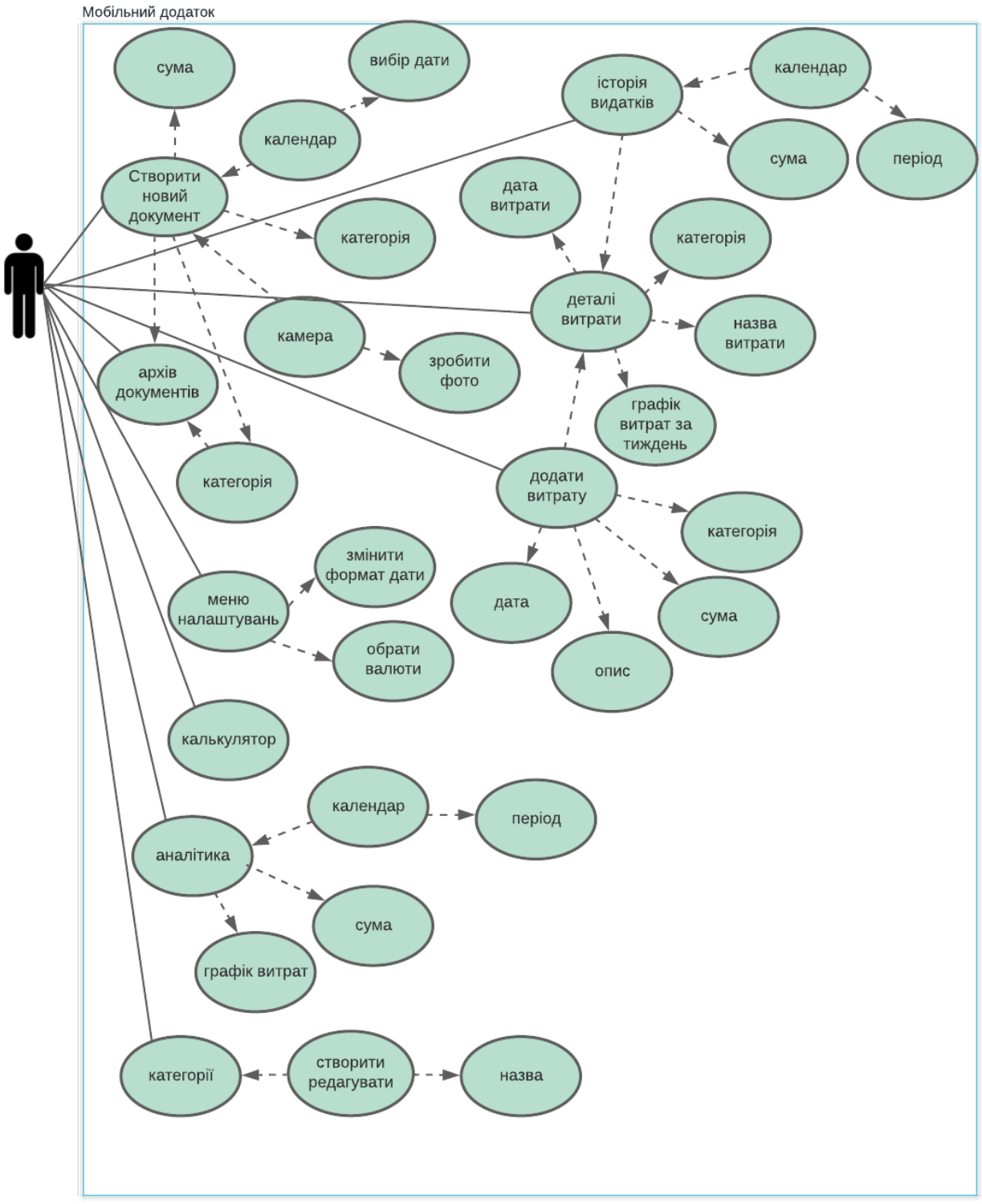


Рисунок 3.5 – Діаграма варіантів використання додатка

3.2 Розробка інтерфейсу

Проектування інтерфейсу є важливою складовою розробки мобільного додатка. Користувач в першу чергу оцінює зовнішній вигляд додатка і те, чи зручно з ним працювати. Ваш додаток може мати унікальний функціонал, з широкими можливостями, але якщо у вас будуть невдало розміщені елементи на екрані чи подразливі кольори оформлення користувач просто видалить його. Тому даному етапу розробки буде приділена особлива увага [25].

Інтерфейс – це інструмент взаємодії користувача з системою. Особливу увагу при розробці інтерфейсу слід приділити простоті і доступності системи для користувача. Функціонал має бути інтуїтивно зрозумілим, оформлення – приємним і витриманим в одному стилі, дизайн – адаптивним, для коректної роботи застосунка на різних пристроях [26].

Можна виділити основні вимоги до розроблюваного інтерфейсу:

- Унікальність і виразність в порівнянні з застосунками-аналогами;
- Важливі елементи мають бути доступними з головного екрану і бути помітними для користувача;
- Має бути врахований користувацький досвід, тобто кнопки мають знаходитись внизу екрана, а елементи «видалити» і «зберегти» мають знаходитись на достатній відстані;
- Прості і зрозумілі візуальні елементи (іконки, кнопки, тощо), що відповідають очікуванням користувача;
- Відгук додатку на дію з боку користувача, наприклад: при натисненні кнопки «зберегти» користувач отримує повідомлення, що його дані збережено;
- Для полегшення введення даних і економії часу користувача має використовуватись випадаючий список в тих місцях де можливо це реалізувати;
- можливість індивідуальних налаштувань;

- Розташування підказок в тих місцях, де у користувача можуть виникнути труднощі;
- Адаптивність дизайну;
- Розмір кнопок має бути достатнім для зручного натиснення.

Основними етапами розробки інтерфейсу є:

- Концепція;
- Дослідження потреб користувачів;
- Визначення структури додатка;
- Проектування дизайну;
- Прототип.

Етапи концептуального проектування і вивчення потреб користувачів були детально розглянуті в попередніх розділах даної роботи. Тому розробку інтерфейсу почнемо з визначення структури розроблюваного додатка.

3.2.1 Структура додатка

На даному етапі розробки інтерфейсу потрібно спроектувати структуру додатка.

Для цього створимо User Flow Diagram (рис.3.6), яка буде відображати послідовність дій користувача під час взаємодії з додатком.

Діаграма послідовності дій користувача дає нам детальне уявлення про структуру додатку і те з яких екранів буде складатися інтерфейс користувача. На даному етапі можемо перейти до створення ескізів застосунка.

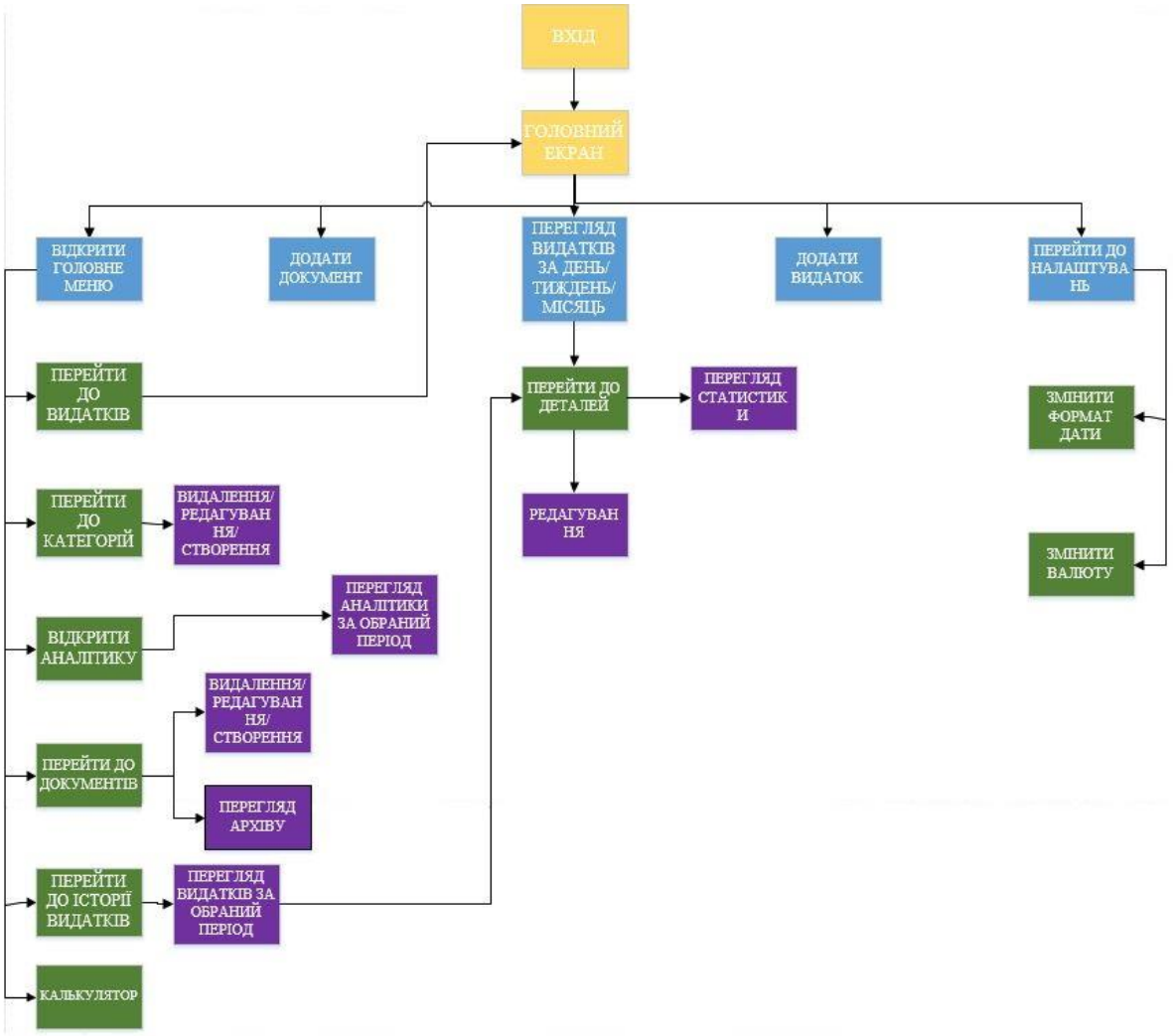


Рисунок 3.6 – Діаграма послідовності дій користувача

3.2.2 Проектування дизайну

Проектування дизайну не менш важливий етап, що дозволяє зробити додаток привабливим для користувачів. Проектування дизайну проводилось на основі засад Material Design. Material Design – це система дизайну, створена та підтримується дизайнерами та розробниками Google. Material.io містить детальні рекомендації щодо UX і впровадження компонентів інтерфейсу користувача для Android [27].

Основними елементами дизайну, що були створені з опорою на Material Design є: головне меню, іконки, палітра кольорів, кнопки.

Для розробки кнопок головного екрану використовувались кнопки - FAB. Ці кнопки слугують для найпоширеніших дій користувача (додавання документа, додавання витрати) і залишаються на екрані під час показу меню чи прокручування вмісту [27].

Головне меню створене з використанням Navigation drawer. Такий тип навігації рекомендовано використовувати у випадках коли список пунктів 5 і більше, для швидкого переходу по не пов’язаним між собою пунктам, тощо. Тому для реалізації головного меню було вибрано саме цей тип навігації [27].

Для роботи з іконками існує спеціальна вбудована бібліотека від Google - Material Design Ikons. Для додавання іконки потрібно просто зайти до папки res і обрати – «створити новий Asset». Далі відкриється вікно Asset Studio де можна обрати тип іконки, саму іконку зі списку, тему, колір, а також змінити назву при необхідності. Використання даної бібліотеки має безліч переваг:

- Зручний спосіб додавання іконки (не потрібно шукати на сторонніх сайтах і завантажувати);
- Широкі можливості налаштування;
- Великий вибір іконок;
- Адаптивність, тобто іконки будуть автоматично змінювати свій розмір в залежності від розміру екрану додатка.

Для стилізації інтерфейсу додатка було обрано палітру кольорів (рис. 3.7).

Рисунок 3.7 – Палітра кольорів додатка

3.2.3 Прототип

На даному етапі ми створюємо прототип екранів розроблюваного додатка. Прототип - це зображення екранів застосунка, що мають максимально наближений вигляд готової версії продукту. Основна мета створення прототипу – це можливість, ще до створення додатка, оцінити його інтерфейс. Розробка прототипу, на відміну від готового додатка, займає набагато менше часу і зусиль, і в разі виявлення дефектів його можна швидко переробити [25]. Процес створення прототипу інтерфейсу дозволяє уникнути помилок і зайвої роботи ще на стадії розробки. Після визначення структури розроблюваного додатка були створені прототипи основних екранів додатка:

- Головний екран – на головному екрані будуть розташовуватись такі елементи: історія видатків (останні записи) і вкладки для зміни періоду показу видатків (день, тиждень, місяць), кнопки для створення нового документа і нової витрати, кнопки відкриття основного меню і меню налаштувань (рис. 3.8);
- Головне меню (3.10) буде містити вкладки: витрати, категорії, аналітика, історія, калькулятор;
- Вкладка «Аналітика» - на даному екрані буде знаходитись фільтр вибору дати і відображатися 2 види діаграм (рис. 3.11);
- Вкладка документи буде містити таці з документами і кнопку для створення нового документа (рис. 3.12);
- Історія витрат – на екрані буде фільтр для вибору дати, загальна сума витрат і список витрат (рис. 3.12);
- Меню налаштувань буде містити пункти: вибір валюти і формат даті (рис. 3.13).

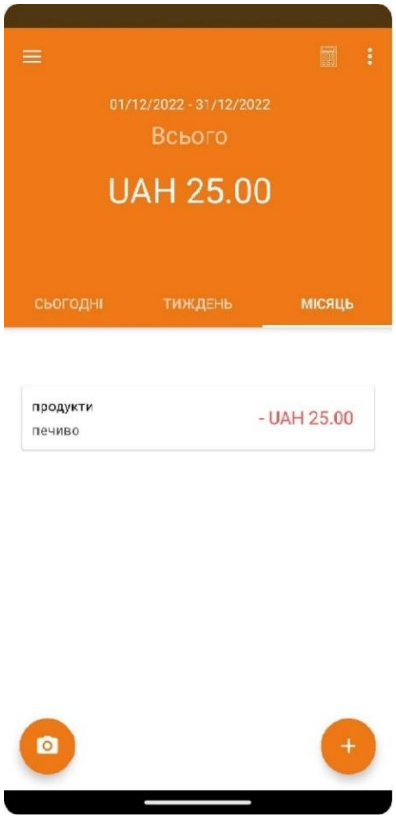


Рисунок 3.8 – Головний екран

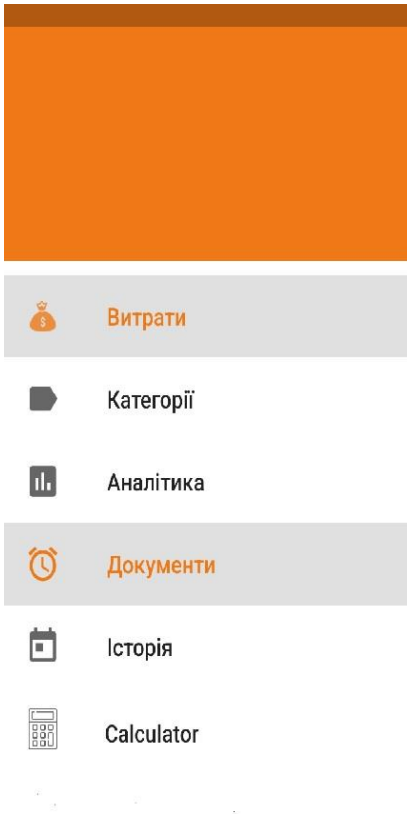


Рисунок 3.9 – Головне меню

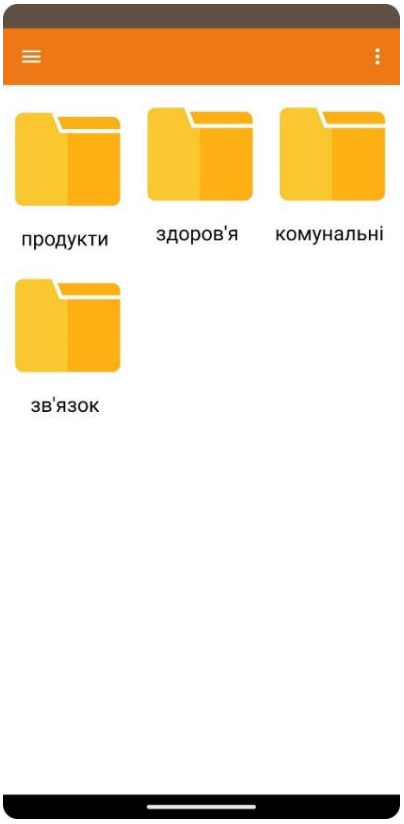
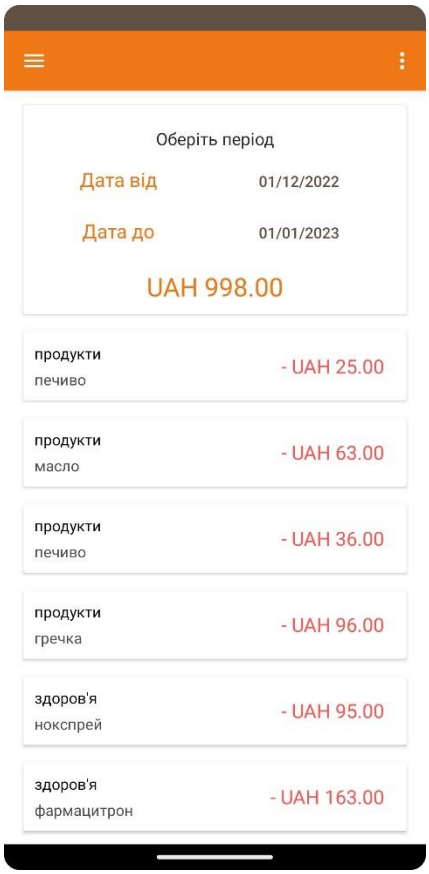


Рисунок 3.10 – Аналітика

Рисунок 3.11 – Документи



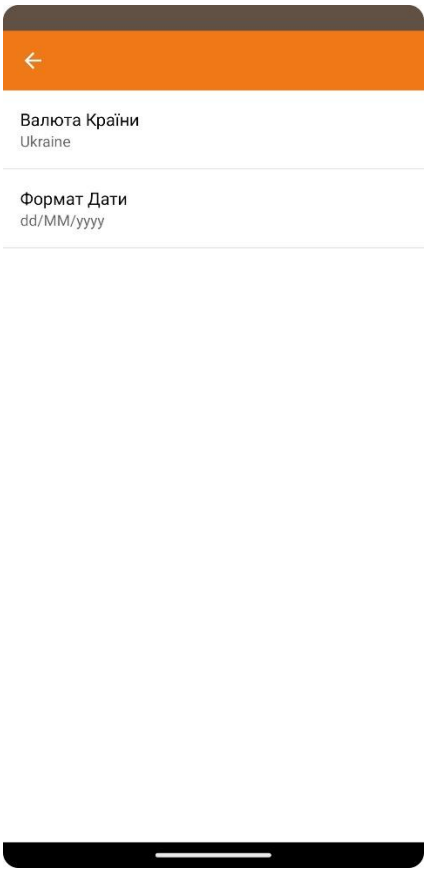


Рисунок 3.12 – Історія витрат

Рисунок 3.13 – Меню налаштувань

4. РЕАЛІЗАЦІЯ

4.1 Архітектура додатку

Мобільний додаток реалізовано із використанням шаблону проектування MVP (Model-View-Presenter) що використовує realm.io у якості двигуна бази даних. Realm.io – це швидка та масштабована альтернатива SQLite [28], має значні переваги у швидкості читання та оновлення даних, гарно підходить для реалізації невеликих баз даних [19]

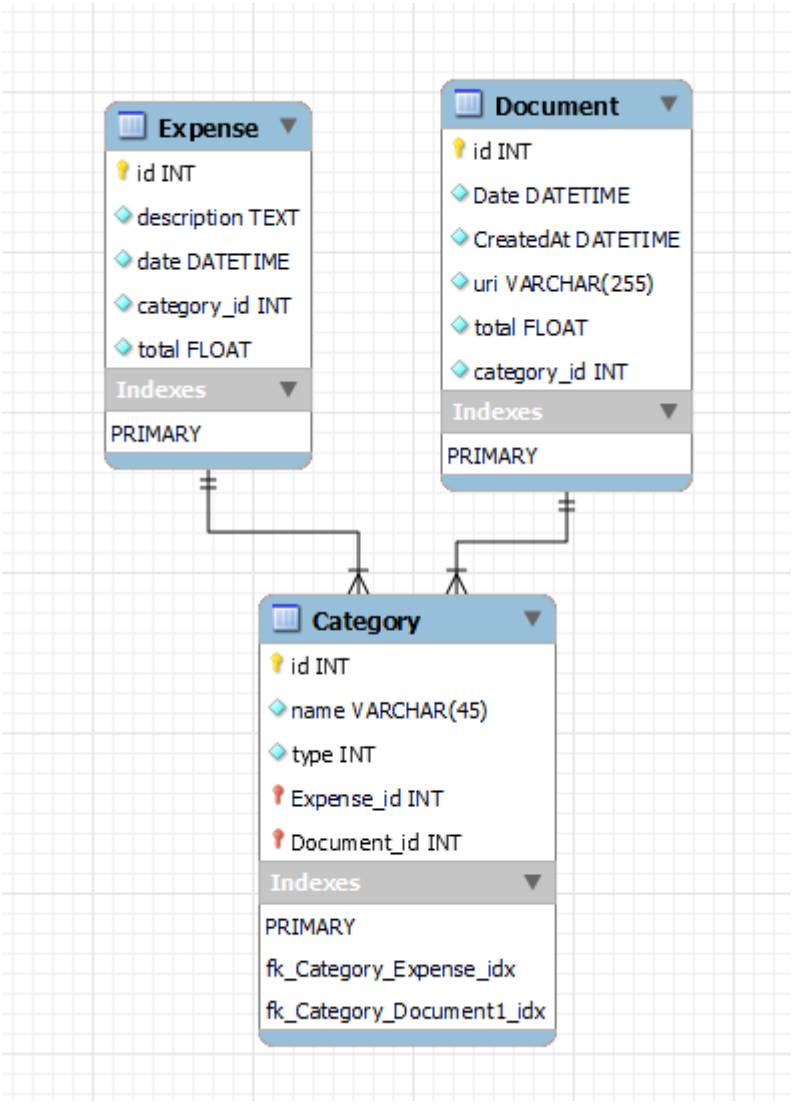


Рисунок 4.1 – ER - діаграма

Як видно з рисунку 4.1 – додаток використовує 3 таблиці.

4.1.1 RecyclerView

Для відображення елементів Категорій, Витрат та Документів у додатку використовується елемент інтерфейсу RecyclerView. Цей елемент створений для швидкого та ефективного відображення великих масивів даних. Коректна робота RecyclerView потребує створення:

- класу відповідального за власне сам елемент;
- класу відповідального за поведінку та зовнішній вигляд окремого елемента списку;
- адаптера – класу, що об’єднує дані із відповідними представленнями;
- layout manager класу, який відповідає за розташування окремих елементів у RecyclerView.

4.1.2 Spinner

Для здійснення вибору одного елементу із набору багатьох однотипних елементів у додатку використовується віджет Spinner. Віджет наповнюється даними за допомогою асоційованого адаптера.

4.2 Програмна реалізація

Presenter класи представлені у додатку у вигляді адаптерів, їх перелік наведено у таблиці 4.1

Таблиця 4.1 – Перелік класів-адаптерів

Назва класу	Опис
BaseExpenseAdapter	Використовується для відображення елементів історії на екрані «історія витрат».
CategoriesAdapter	Відповідальний за побудову списку категорій на екрані «Категорії»

CategoriesAdapterFolder	використовуючи окремий layout manager, відображує категорії у вигляді директорій на екрані «Документи».
CategoriesSpinnerAdapter	Наповнює даними віджети Spinner на екранах створення нової витрати та на екрані створення нового документа.
DocumentsAdapter	Після вибору категорії на екрані Документи даний Адаптер наповнює даними RecyclerView який відповідає за відображення списку документів обраної категорії.
MainExpenseAdapter	Використовується для наповнення даними переліку витрат на екрані «Витрати»

UML діаграму класів адаптерів наведено на рисунку 4.2

Класи View перелічені у таблиці 4.2

Таблиця 4.2 – перелік класів View

Назва	Відображення
Categories: - CategoriesFragment	Екран категорій
Documents: - DocumentFragment - NewDocumentActivity - NewDocumentFragment	Екран документів із екраном додавання/оновлення документа

ExpenseCalculator: - CalculateFactorial - ExCalculatorActivity - ExtendedDoubleEvaluator	Екран калькулятора
Expenses: - ExpenseDetailActivity - ExpenseDetailFragment - ExpensesContainerFragment - ExpensesFragment - ExpensesViewPagerAdapter - NewExpenseFragment	Класи витрати, оновлення/додавання витрати
Folders: - FolderFragment	Категорії у вигляді директорій
Help: - HelpActivity	Екран допомоги
History: - HistoryFragment	Екран історії
Settings: - SettingsActivity	Екран налаштувань
Statistic: - StatisticsFragment	Статистика
MainActivity	Головний екран додатка

UML Діаграма класів View зображена на рисунку 4.3

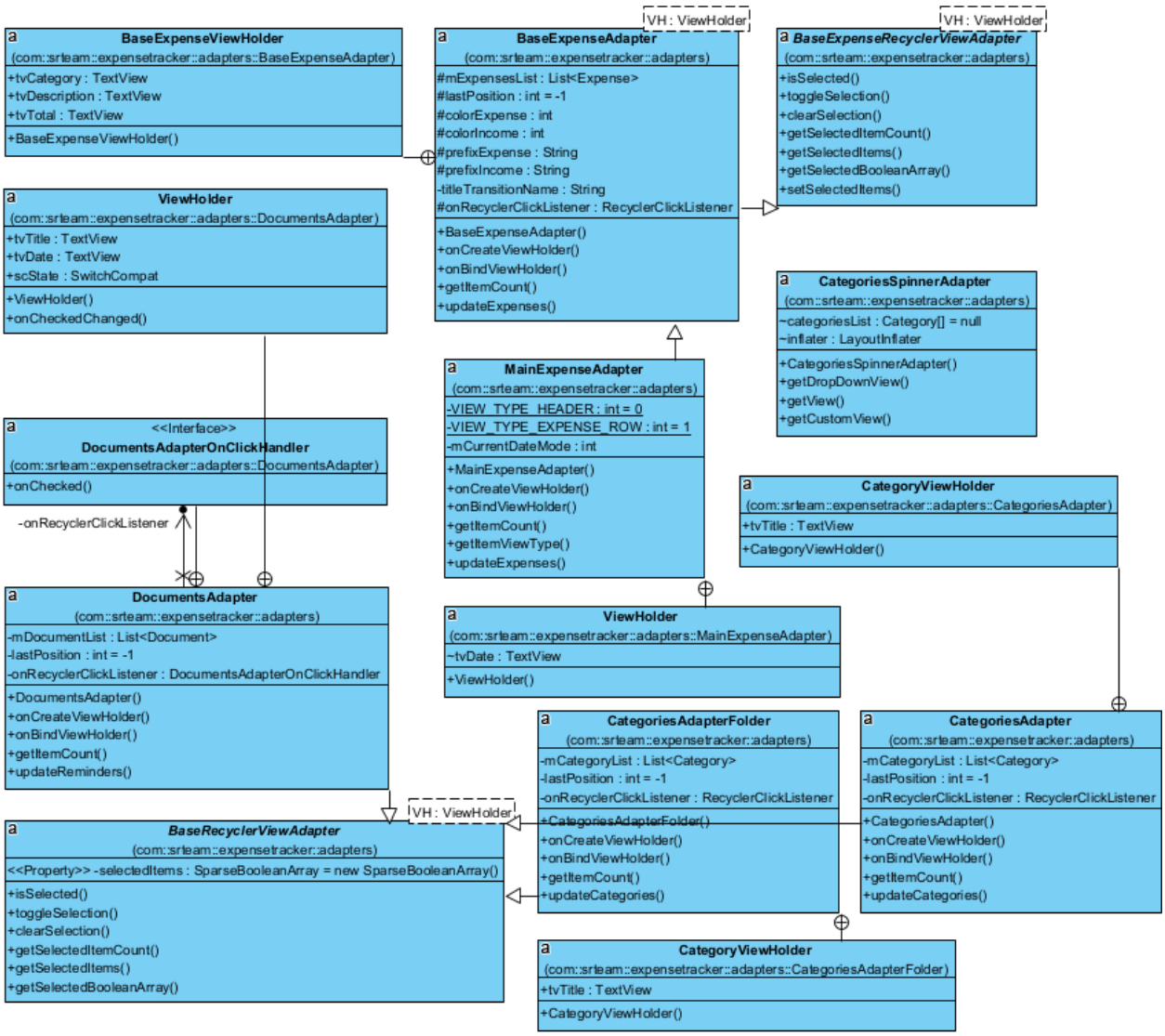


Рисунок 4.2 – UML діаграма адаптерів

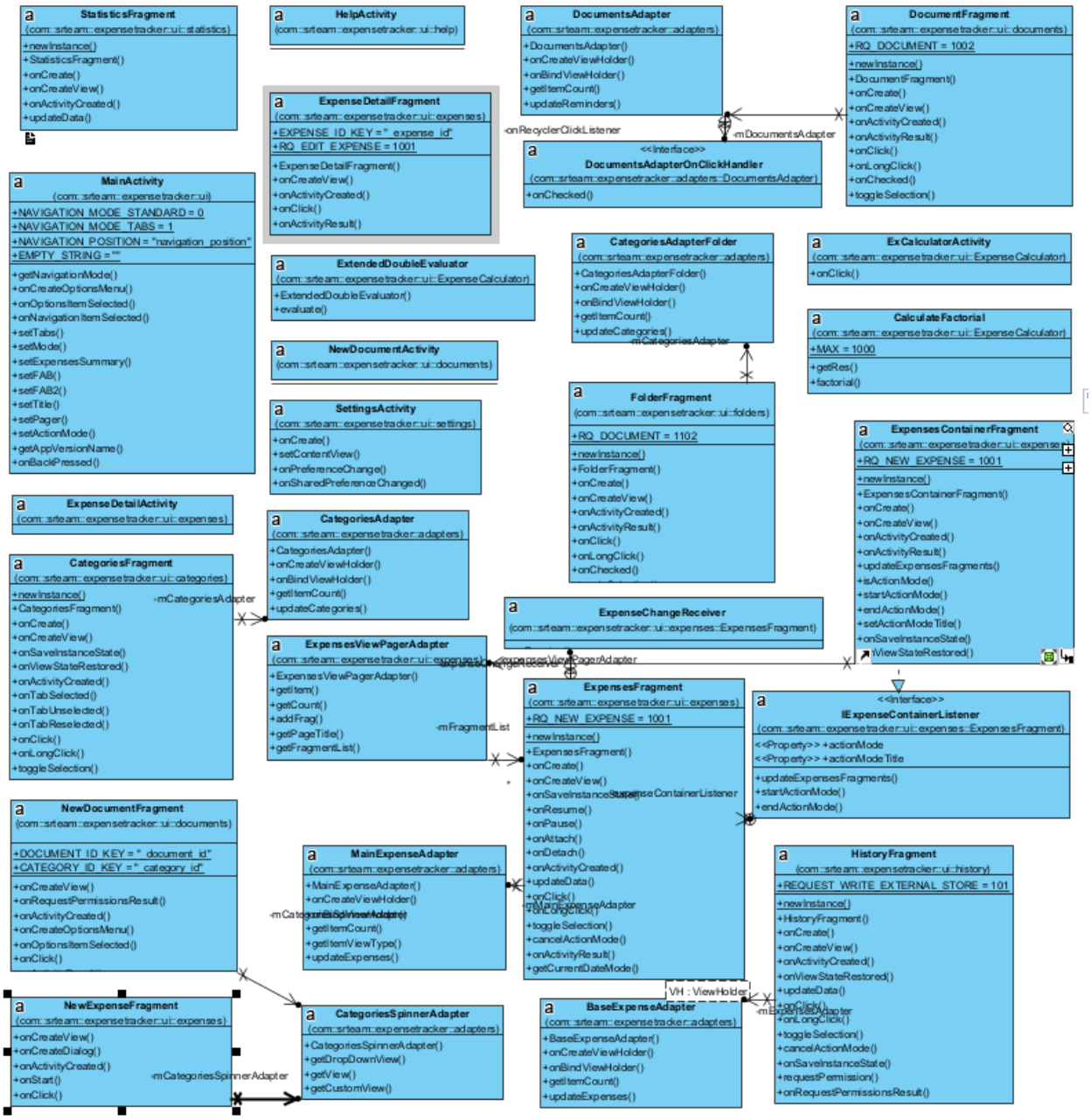


Рисунок 4.3 – Діаграма класів View

Також реалізовано 3 Model класи:

1. Category
2. Document
3. Expense

4.3 Тестування

Останнім, але важливим, етапом роботи є тестування. Тестування мобільно застосунку проводиться з метою перевірка якості і надійності роботи створеного додатка, а також відповідності поставленим вимогам.

Було проведено функціональне і нефункціональне тестування мобільного застосунка.

Функціональне тестування – це тестування, що проводиться з метою перевірки відповідності роботи додатка функціональним вимогам [29].

Функціональне тестування проводиться в умовах наближених до користувача. Включає тестування роботи на різних пристроях, з різними розмірами екрану, з різними версіями ОС і має на меті перевірити:

- Чи виконує додаток свої основні функції;
- Реакцію на помилки користувачів;
- Відмінності між очікуваною поведінко і реальною;
- Наявність прихованих дефектів в роботі;
- Доступність додатку з різних пристроїв;

Для проведення тестування був сформований список основних функцій для тестування:

- Створення нового видатку
- Редагування та видалення видатку
- Перевірка роботи фільтру історії видатків
- Створення нового документа
- Редагування та видалення документа
- Створення нової категорії
- Редагування та видалення категорії
- Перевірка формування аналітики

На основі цього списку було проведено тестування на пристроях з версіями ОС 10, 11 та 12 та розмірами екрану 6,43” 6,71”, 6”52. Результати тестування представлені в таблиці.

Таблиця 4.3 – Результати тестування

Функції	ОС - 10, екран- 6,43”	ОС - 11, екран- 6,71”	ОС - 12, екран- 6,52”
Створення нового видатку	+	+	+
Редагування та видалення видатку	+	+	+
Перевірка роботи фільтру історії видатків	+	+	+
Створення нового документа	+	+	+
Редагування та видалення документа	+	+	+
Створення нової категорії	+	+	+
Редагування та видалення категорії	+	+	+
Перевірка формування аналітики	+	+	+

Тестування проведено успішно, в процесі тестування було виявлено неочікувану поведінку – на деяких додатках екрани відображаються в темному стилі, хоча зміна теми не передбачена в застосунку.

Також було окремо протестовано реакцію додатку на помилки. Для цього були неповністю заповнені форми створення нової витрати і нового документу. В результаті тестування встановлено, що додаток веде себе очікувано – повідомляє користувачеві, що потрібно заповнити відсутню інформацію, збоїв і помилок в роботі не зафіксовано.

Після успішно проведеного функціонального тестування було проведено нефункціональні тести. Нефункціональні тести – це тести, які перевіряють нефункціональні показники додатка такі як: зручність для користувача, стабільність роботи, продуктивність тощо.

В рамках нефункціонального тестування було проведено usability testing. Usability testing – це тест, який має на меті перевірку зручності роботи з додатком для користувачів. Для проведення тестування була залучена група з 5 учасників. Учасникам було запропоновано випробувати основні функції додатка і оцінити за 5 бальною шкалою зручність користування [30]. Результати тестування представлені в таблиці:

Таблиця 4.4 – Результати Usability тестів

Функції	Уч. 1 /оцінка	Уч. 2 /оцінка	Уч. 3 /оцінка	Уч. 4 /оцінка	Уч. 5 /оцінка
Створення нової витрати	4	3	5	4	5
Створення нового документа	5	4	4	5	4
Перегляд історії витрат	5	5	5	5	5

Інформативність аналітики	4	4	4	4	4
Перегляд архіву документів	5	5	4	4	4
Загальне враження	5	4	4	4	4

Учасниками були виділені наступні незручності в роботі:

- Створення нової витрати: відсутнє поле для введення кількості товару;
- Створення нового документа: кнопка «зберегти» має незручне розташування;
- Інформативність аналітики: невдалий вибір кольорів в діаграмах витрат;
- Перегляд архіву документів: відсутність можливості фільтрування документів по даті.

На основі проведених тестувань були визначені можливості для покращення роботи додатка:

- Розширити форму створення нової витрати, додавши поле введення кількості товару;
- Змінити розташування кнопки «зберегти» на екрані створення нового документа;
- Змінити кольори діаграм на більш контрастні;
- Додати можливість пошуку і виведення документів по даті створення;
- Додати можливість змінювати тему додатка (темна, світла).

ВИСНОВКИ

В процесі виконання роботи було розглянуто проблему контролю персональних витрат, проаналізовано актуальність проблеми, розглянуто існуючі рішення. На основі отриманих даних були сформовані мета та задачі дослідження. На підставі об’єктивних даних прийняте рішення про побудову саме мобільного додатку для OS Android із використанням сучасних систем розробки та існуючих бібліотек.

У результаті декомпозиції роботи на складові, використання спроектованого дизайну та прототипізації інтерфейсу задачу було реалізовано із використання мови програмування Java. Було отримано мобільний додаток, що має наступні можливості:

- Контроль персональних витрат;
- Візуалізована аналітика витрат:
- Оцифрування розрахункових документів.

В процесі виконання роботи:

- Проведено аналіз предметної області;
- Сформовано вимоги до розроблюваного застосунку;
- Обрано методи розробки;
- Проведено проектування системи;
- Розроблено і протестовано мобільний додаток.

Тестування показали, що застосунок відповідає функціональним вимогам: надає можливості контролю персональних витрат, оцифрування і структурованого збереження розрахункових документів, формує візуалізовану аналітику витрат, що і було метою даної роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мілославський І. О. Інформаційна технологія проектування веб-сервісу з обліку персональних витрат : робота на здобуття кваліфікаційного ступеня бакалавра: спец. 122 – комп'ютерні науки / наук. кер. Б.О. Кузіков, Суми : Сумський державний університет, 2019. 45 с.

2. Козлова Ю. Чому бідні, бо… Главком | Glavcom. URL: https://glavcom.ua/publications/chomu-bidni-bo-mizhnarodne-doslidzhennya-pokazalo-shcho-ukrajincyam-brakuje-finansovih-znan-599830.html (дата звернення: 02.12.2022).

3. Що таке фінансова грамотність і навіщо вона потрібна. Фонд гарантування вкладів фізичних осіб. URL: https://www.fg.gov.ua/articles/48009-shcho-take-finansova-gramotnist-i-navishcho-vona-potribna.html (дата звернення: 10.11.2022).

4. K. Goyal and S. Kumar, "Financial literacy: A systematic review and bibliometric analysis," *Int J Consum Stud*, vol. 45, no. 1, pp. 80–105, Jan. 2021, doi: 10.1111/ijcs.12605.

5. E. Kicova and P. Gorzelanczyk, "FINANCIAL LITERACY AND SHOPPING BEHAVIOUR," *Ekonomicko-manazerske spektrum*, vol. 16, no. 1, pp. 91–103, 2022, doi: 10.26552/ems.2022.1.91-103.

6. M. Horwood and J. Guo, "The Significance of Financial Literacy on Global Economic, Physical, and Mental Wellbeing – A Key Buffer for Our Most Vulnerable," *European Journal of Health*, 2021, doi: 10.31234/OSF.IO/46PA9.

7. OECD, "OECD/INFE 2020 International Survey of Adult Financial Literacy," 2020. Accessed: Dec. 01, 2022. [Online]. Available: https://www.oecd.org/financial/education/launchoftheoecdinfeglobalfinanciallitera cysurveyreport.htm

8. Y. Shen, W. Hu, and Y. Zhang, “Digital Finance, Household Income and Household Risky Financial Asset Investment,” *Procedia Comput Sci*, vol. 202, pp. 244–251, Jan. 2022, doi: 10.1016/J.PROCS.2022.04.032.

9. Peter Morgan, Bihong Huang, and Long Trinh, “THE FUTURE OF WORK AND EDUCATION FOR THE DIGITAL AGE,” *T 20 Japan 2019*, 2019, Accessed: Dec. 01, 2022. [Online]. Available: https://t20japan.org/policy-brief-need-promote-digital-financial-literacy/

10. A. Cwynar, “Financial literacy and financial education in Eastern Europe,” *The Routledge Handbook of Financial Literacy*, pp. 400–419, Dec. 2021, doi: 10.4324/9781003025221-31.

11. Закон України «Про захист прав споживачів».
Верховна рада України. URL: http://zakon.rada.gov.ua/laws/show/1023-12 - (дата звернення: 9.11.2022).

12. Number of smartphone subscriptions worldwide from 2016 to 2021, with forecasts from 2022 to 2027. Statista. URL: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (дата звернення: 16.11.2022).

13. App Attention Index. Cisco AppDynamics. URL: https://www.appdynamics.com/c/dam/r/appdynamics/Gated-Assets/analyst-reports/AppDynamics-App-Attention-Index-2021.pdf (дата звернення: 14.11.2022).

14. Edith. Android loses 8% of its global OS market share in five years. stockapps.com. URL: https://stockapps.com/blog/android-loses-8-of-its-global-os-market-share-in-five-years/ (дата звернення: 07.11.2022).

15. Meet Android Studio | Android Developers. Android Developers. URL: https://developer.android.com/studio/intro (date of access: 10.11.2022).

16. Документация по Xamarin. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/ru-ru/xamarin/?WT.mc_id=dotnet-35129-website (дата звернення: 12.11.2022).

17. SDK Platform Tools release notes | Android Developers. Android Developers. URL: https://developer.android.com/studio/releases/platform-tools (date of access: 07.12.2022).

18. Android 10 | Android Developers. Android Developers. URL: https://developer.android.com/about/versions/10 (date of access: 07.12.2022).

19. Gomolinski A. SQLite vs Realm: Which Database to Choose in 2023 | Orangesoft. Mobile App Development Company - Custom Application Development Services | Orangesoft. URL: https://orangesoft.co/blog/realm-vs-sqlite (date of access: 18.11.2022).

20. MPAndroidChart - Weeklycoding. Weeklycoding. URL: https://weeklycoding.com/mpandroidchart/ (date of access: 10.11.2022)

21. AnyChart Android Charts | AnyChart. AnyChart. URL: https://www.anychart.com/technical-integrations/samples/android-charts/ (date of access: 10.11.2022).

22. Petsas T., Papadogiannakis A., Polychronakis M., Markatos E.P., Karagiannis T. Measurement, Modeling, and Analysis of the Mobile App Ecosystem: ACM Transactions on Modeling and Performance Evaluation of Computing Systems, vol. 2, no. 2, pp. 1–33, May 2017, doi: 10.1145/2993419.

23. Constantin F., Harris C., Ieong S., Mehta A., Tan X. Optimizing ad refresh in mobile app advertising: *The Web Conference 2018 - Proceedings of the World Wide Web Conference, WWW 2018*, pp. 1399–1408, Apr. 2018, doi: 10.1145/3178876.3186045.

24. Учасники проектів Вікімедіа. Діаграма прецедентів – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Діаграма_прецедентів (дата звернення: 01.12.2022).

25. Мілославська Ю.М. Персональна система оцифрування даних з розрахункових документів: робота на здобуття кваліфікаційного ступеня бакалавра: спец. 122 – комп'ютерні науки / наук. кер. Б.О. Кузіков, Суми : Сумський державний університет, 2019. 47 с.

26. Інтерфейс мобільних програм: що це таке, основні принципи розробки інтерфейсу mobile app. IT-компанія WEZOM - Київ, Україна. URL: https://wezom.com.ua/ua/blog/dizajn-interfejsov-mobilnyh-prilozhenij (дата звернення: 11.12.2022).

27. Material Design. Material Design. URL: https://m3.material.io/get-started (date of access: 19.12.2022).

28. Realm Home. Realm Home | Realm.io. URL: https://realm.io/ (date of access: 11.11.2022).

29. Functional testing and its advantages and disadvantages. *StrongQA*. URL: https://strongqa.com/qa-portal/knowledge-base/testing-types/functional-testing (date of access: 10.12.2022).

30. Usability Testing | Usability.gov. *Home | Usability.gov*. URL: https://www.usability.gov/how-to-and-tools/methods/usability-testing.html (date of access: 10.12.2022).

ДОДАТОК А. ПЛАНУВАННЯ РОБІТ

А.1 ІДЕНТИФІКАЦІЯ МЕТИ ІТ-ПРОЕКТУ

Продуктом дипломного проекту є мобільний додаток персональних витрат. Результати деталізації методом SMART розміщені у табл. А.1.1.

Таблиця А.1.1 − Деталізація мети методом SMART

Specific (конкретна)	Створення мобільного додатку персональних витрат
Measurable (вимірювана)	Реалізація в додатку такий функцій: оцифровування розрахункових документів, внесення даних про витрати і подальший аналіз витрат
Achievable (досяжна)	Зібрано достатню кількість інформації і отримано знання і навики по роботі з Android Studio, що дозволяє оцінити ціль, як досяжну
Relevant (реалістична)	Створення цього додатку вирішить одразу два актуальних питання: збереження фінансових документів і аналіз витрат
Time-framed (обмежена у часі)	Ціль обмежена у часі датою захисту кваліфікаційної роботи згідно навчального плану

А.2 ПЛАНУВАННЯ ЗМІСТУ СТРУКТУРИ РОБІТ ІТ-ПРОЕКТУ

При створенні складних проектів розбиття задач на менші частини значною мірою полегшує роботу. Для цього зручно використовувати ієрархічну структуру робіт WBS (work breakdown structure) (рис. А.2.1). За допомогою WBS структурують і ділять проект на легко керовані компоненти. Вони, в свою чергу, діляться до тих пір, поки не будуть назначені конкретному спеціалісту в команді.

Після декомпозиції проекту доцільно створити організаційну структуру проекту (OBS). Це ієрархічна структура управління проектом, яка показує розподіл завдань проекту між спеціалістами. OBS розбивається до рівня груп, які виконують найнижчий рівень робіт в WBS. Виходячи з того, що данний проект буде виконуватись одноособово, то організаційна структура проекту приводиться більше з цілью ознайомлення з даним методом планування (рис А.2.2).

Рисунок А.2.1 – таблиця WBS (work breakdown structure)

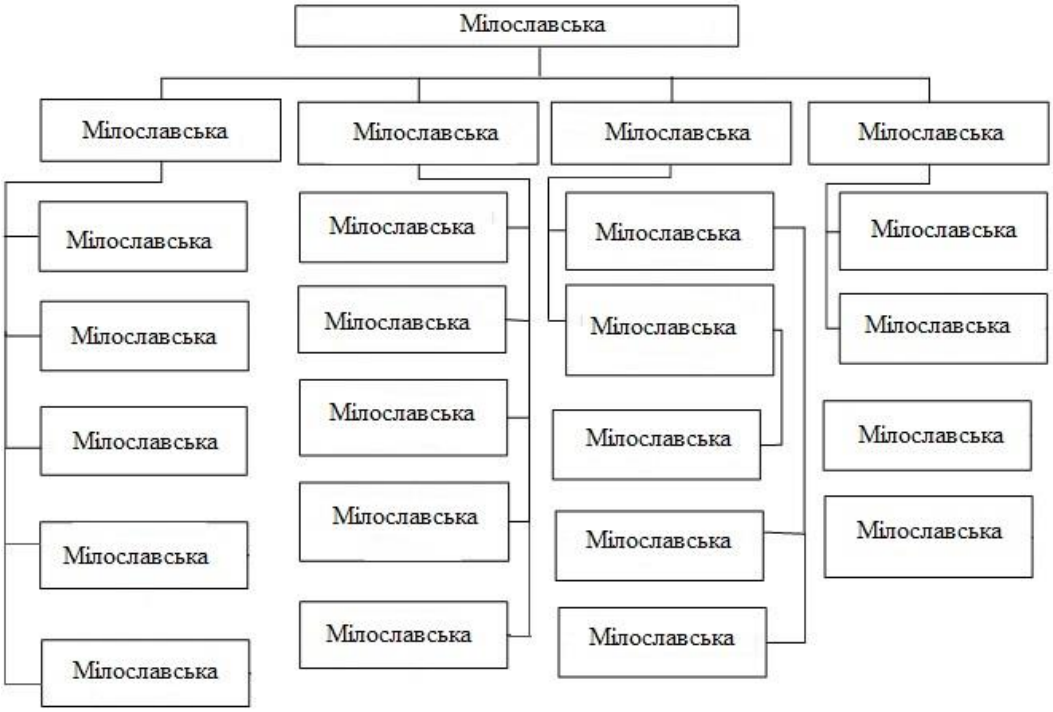


Рисунок А.2.2 – таблиця OBS (Organization Breakdown Structure

А.3 ПОБУДОВА КАЛЕНДАРНОГО ГРАФІКА ВИКОНАННЯ ІТ-ПРОЕКТУ

Для того, щоб мати реальне уявлення про тривалість виконання робіт з урахуванням обмеженості у використанні ресурсів, на підставі часткових мережевих моделей, а також, проекту в цілому з урахуванням вихідних і святкових днів, будують календарний графік робіт.

Він є реальним розподілом робіт по пакету по календарними датами, тобто своєрідним розкладом виконання робіт. Діаграма Ганта є досить зручним для користування. Діаграма Ганта представляє собою відрізки, які розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, як складова плану, розміщуються по вертикалі. Початок, кінець і довжина відрізків на шкалі часу відповідають початку, кінця і тривалості завдання (рис. А.3.1)

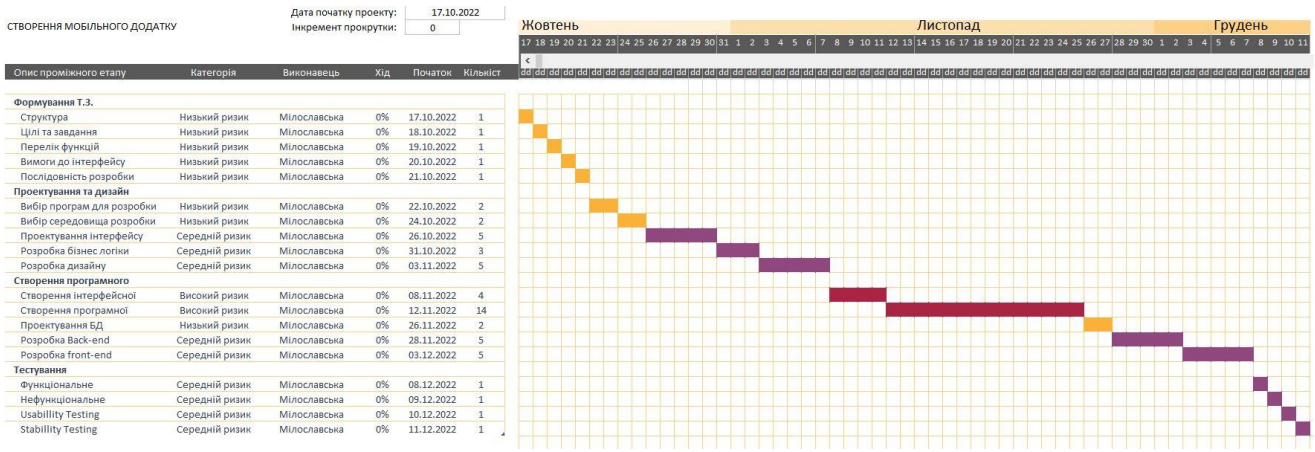


Рисунок А.3.1 –Діаграма Ганта

А.4 ПЛАНУВАННЯ РИЗИКІВ ПРОЕКТУ

Планування та реалізація проектів відбувається в умовах невизначеності, що породжується зміною внутрішнього та зовнішнього середовища. Тому при плануванні роботи над проектом особливу увагу потрібно приділяти ризикам.

Ризик – це невизначена зовнішня чи внутрішня подія, що впливає на досягнення цілей.

До основних ризиків розробки мобільного додатку є:

- відсутність досвіду;
- збільшення навантаження під час реалізації проекту;
- зміна цілей у ході реалізації проекту;
- помилки у плануванні та проектуванні;
- помилкові технологічні рішення;
- недостатня кваліфікованість розробника;
- зміна строків виконання роботи;
- перебої з електропостачанням та/або з інтернет з’єднанням;
- відмова обладнання (компьютера);
- зростання вимог до проекту.

Оцінки ризиків для кожного окремого проекту можуть мати свій набір характеристик. Для оцінювання ризиків даного проекту використаємо такі характеристики:

- величина можливої шкоди;
- ймовірність реалізації ризику (виникнення події);

Таким чином, ризик – це комбінація ймовірності та наслідків.

Таблиця А.4.1 Ймовірність виникнення і величина ризику

№	Ризики	Виникнення	Втрати
1	Відсутність досвіду	3	3
2	Збільшення навантаження під час реалізації проекту	3	2
3	Зміна цілей у ході реалізації проекту	1	4
4	Помилки у плануванні та проектуванні	4	3
5	Помилкові технологічні рішення	4	3
6	Недостатня кваліфікованість розробника	3	4
7	Зміна строків виконання роботи	2	2
8	Перебої з електропостачанням та/або з інтернет з’єднанням	2	3
9	Відмова обладнання	2	4
10	Зростання вимог до проекту	2	3

Класифікація величин можливих втрат:

- Несуттєві (1): не дають реальних негативних наслідків або не становлять суттєвої загрози для проекту;
- Малі(2): мають невеликий потенціал для негативних наслідків, але не вплинуть на загальний успіх;
- Помірні(3): можуть призвести до негативних наслідків, створюючи помірну загрозу для проекту;

- Критичні(4): із суттєвими негативними наслідками, які серйозно впливатимуть на успіх проекту;
- Катастрофічні(5): з вкрай негативними наслідками, які можуть призвести до закриття або довгострокового збою діяльності. Вимагають найбільшої уваги та ресурсів.

Класифікація ймовірності виникнення ризику:

- малоймовірне (1): вкрай рідко, майже без ймовірності виникнення;
- рідко (2): невеликий шанс прояву;
- ймовірно (3): трапляється з ймовірністю 30-50%;
- часто (4): може статися з великою ймовірністю;
- безперечно (5): майже напевно виникне.

На основі таблиці «Ймовірність виникнення і величина ризику» побудуємо матрицю ризику.

Матриця ризику - це інструмент процесу управління загрозами, призначений для підвищення об'єктивності його інтерпретації. Відповідно матриця відображує різні варіанти комбінації ймовірність - втрати.

Таблиця А.4.2 - Матриця ризиків

Втрати	Ймовірність					
		1	2	3	4	5
	1	Ігноровані (1)	Незначні (2)	Незначні (3)	Помірні (4)	Помірні (5)
	2	Незначні (2)	Незначні (4)	Помірні (6)	Помірні (8)	Істотні (10)
	3	Незначні (3)	Помірні (6)	Помірні (9)	Істотні (12)	Істотні (15)
	4	Помірні (4)	Помірні (8)	Істотні (12)	Істотні (16)	Істотні (20)
	5	Помірні (5)	Істотні (10)	Істотні (15)	Істотні (20)	Критичні (25)

Аналізуючи ризики проекту можна розділити їх на:

- Ігноровані
    - Відсутні
- Незначні
    - Зміна цілей у ході реалізації проекту (4)
    - Зміна строків виконання роботи (4)
- Помірні
    - Відсутність досвіду (9)
    - Перебої з електропостачанням або з інтернет з’єднанням (8)
    - Збільшення навантаження під час реалізації проекту (6)
    - Відмова обладнання (6)
    - Зростання вимог до проекту (6)
- Істотні
    - Помилки у плануванні та проектуванні (12)
    - Помилкові технологічні рішення (12)
    - Недостатня кваліфікованість розробника (12)
- Критичні
    - Відсутні

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНОГО КОДУ

MainActivity:

```
package com.srteam.expensetracker.ui;

import android.annotation.SuppressLint;
import android.app.Dialog;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import androidx.annotation.DrawableRes;
import androidx.annotation.IntDef;
import com.google.android.material.appbar.AppBarLayout;
import androidx.coordinatorlayout.widget.CoordinatorLayout;
import
com.google.android.material.floatingactionbutton.FloatingActionB
utton;
import
com.google.android.material.navigation.NavigationView;
import com.google.android.material.tabs.TabLayout;
import androidx.fragment.app.Fragment;
import androidx.core.view.GravityCompat;
import androidx.viewpager.widget.ViewPager;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.appcompat.widget.Toolbar;
import android.view.ActionMode;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
```

```
import android.widget.TextView;

import
com.google.android.gms.ads.interstitial.InterstitialAd;
import com.srteam.expensetracker.R;
import com.srteam.expensetracker.entities.Expense;
import com.srteam.expensetracker.interfaces.IDateMode;
import
com.srteam.expensetracker.interfaces.IMainActivityListener;
import
com.srteam.expensetracker.interfaces.IUserActionsMode;
import
com.srteam.expensetracker.ui.ExpenseCalculator.ExCalculatorActiv
ity;
import
com.srteam.expensetracker.ui.categories.CategoriesFragment;
import
com.srteam.expensetracker.ui.documents.NewDocumentActivity;
import
com.srteam.expensetracker.ui.expenses.ExpensesContainerFragment;
import com.srteam.expensetracker.ui.folders.FolderFragment;
import com.srteam.expensetracker.ui.help.HelpActivity;
import com.srteam.expensetracker.ui.history.HistoryFragment;
import
com.srteam.expensetracker.ui.documents.DocumentFragment;
import
com.srteam.expensetracker.ui.settings.SettingsActivity;
import
com.srteam.expensetracker.ui.statistics.StatisticsFragment;
import com.srteam.expensetracker.utils.AppUtility;
import com.srteam.expensetracker.utils.DateUtils;
import com.srteam.expensetracker.utils.Util;

import java.lang.annotation.Retention;
```

```
import java.lang.annotation.RetentionPolicy;
import java.util.List;

import br.com.joinersa.oooalertdialog.Animation;
import br.com.joinersa.oooalertdialog.OnClickListener;
import br.com.joinersa.oooalertdialog.OoOAlertDialog;

public class MainActivity extends BaseActivity implements
NavigationView.OnNavigationItemSelectedListener,
IMainActivityListener {

        @IntDef({NAVIGATION_MODE_STANDARD,
NAVIGATION_MODE_TABS})
        @Retention(RetentionPolicy.SOURCE)
        public @interface NavigationMode {}

        public static final int NAVIGATION_MODE_STANDARD = 0;
        public static final int NAVIGATION_MODE_TABS = 1;
        public static final String NAVIGATION_POSITION =
"navigation_position";

        private int mCurrentMode = NAVIGATION_MODE_STANDARD;
        private int idSelectedNavigationItem;

        private DrawerLayout mainDrawerLayout;
        private NavigationView mainNavigationView;
        private Toolbar mToolbar;
        private TabLayout mainTabLayout;
        private FloatingActionButton mFloatingActionButton;
        private                              FloatingActionButton
mFloatingActionButtonDocument;

        // Expenses Summary related views
        private LinearLayout llExpensesSummary;
```

```
        private TextView tvDate;
        private TextView tvDescription;
        private TextView tvTotal;
        public static final String EMPTY_STRING = "";
        private InterstitialAd mInterstitialAd;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            initUI();
            setUpDrawer();
            setUpToolbar();
            if ( savedInstanceState != null) {
                int                 menuItemId                =
savedInstanceState.getInt(NAVIGATION_POSITION);
                mainNavigationView.setCheckedItem(menuItemId);

mainNavigationView.getMenu().performIdentifierAction(menuItemId,
0);
            } else {

mainNavigationView.getMenu().performIdentifierAction(R.id.nav_ex
penses, 0);
            }
        }

        @NavigationMode
        public int getNavigationMode() {
            return mCurrentMode;
        }

        private void initUI() {
```

```
            mainDrawerLayout                                    =
(DrawerLayout)findViewById(R.id.drawer_layout);
            mainTabLayout                                       =
(TabLayout)findViewById(R.id.tab_layout);
            mainNavigationView                                  =
(NavigationView)findViewById(R.id.nav_view);
            mFloatingActionButton                               =
(FloatingActionButton)findViewById(R.id.fab_main);
            mFloatingActionButtonDocument                       =
(FloatingActionButton)findViewById(R.id.fab_main2);
            llExpensesSummary                                   =
(LinearLayout)findViewById(R.id.ll_expense_container);
            tvDate = (TextView)findViewById(R.id.tv_date);
            tvDescription                                       =
(TextView)findViewById(R.id.tv_description);
            tvTotal = (TextView)findViewById(R.id.tv_total);


            //showBannerInterstitialAd();
        }

        private void setUpDrawer() {

mainNavigationView.setNavigationItemSelectedListener(this);
        }

        private void setUpToolbar() {
            mToolbar = (Toolbar) findViewById(R.id.toolbar);
            setSupportActionBar(mToolbar);


getSupportActionBar().setDisplayHomeAsUpEnabled(true);
```

```
getSupportActionBar().setHomeAsUpIndicator(R.drawable.ic_menu_wh
ite_24dp);
            mToolbar.setNavigationOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View v) {

mainDrawerLayout.openDrawer(GravityCompat.START);
                }
            });
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action
bar if it is present.
            getMenuInflater().inflate(R.menu.menu_main, menu);
            return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            // Handle action bar item clicks here. The action bar
will
            // automatically handle clicks on the Home/Up button,
so long
            //   as   you   specify   a   parent   activity   in
AndroidManifest.xml.
            int id = item.getItemId();

            //noinspection SimplifiableIfStatement
            if (id == R.id.action_settings) {
```

```
                startActivity(new                    Intent(this,
SettingsActivity.class));
                return true;
            }
            else if (id == R.id.action_cal) {
                startActivity(new                    Intent(this,
ExCalculatorActivity.class));
                return true;
            }
            return super.onOptionsItemSelected(item);
        }

        @Override
        protected void onSaveInstanceState(Bundle outState) {
            outState.putInt(NAVIGATION_POSITION,
idSelectedNavigationItem);
            super.onSaveInstanceState(outState);
        }

        @Override
        public     boolean     onNavigationItemSelected(MenuItem
menuItem) {
            menuItem.setChecked(true);
            mainDrawerLayout.closeDrawers();
            switchFragment(menuItem.getItemId());
            return false;
        }

        @Override
        public   void   setTabs(List<String>   tabList,   final
TabLayout.OnTabSelectedListener onTabSelectedListener) {
            mainTabLayout.removeAllTabs();
            mainTabLayout.setVisibility(View.VISIBLE);
```

```
mainTabLayout.addOnTabSelectedListener(onTabSelectedListener);
            for (String tab : tabList) {

mainTabLayout.addTab(mainTabLayout.newTab().setText(tab).setTag(
tab));
            }
        }

        @SuppressLint("RestrictedApi")
        @Override
        public void setMode(@NavigationMode int mode) {
            mFloatingActionButton.setVisibility(View.GONE);

mFloatingActionButtonDocument.setVisibility(View.GONE);
            llExpensesSummary.setVisibility(View.GONE);
            mCurrentMode = mode;
            switch (mode) {
                case NAVIGATION_MODE_STANDARD:
                    setNavigationModeStandard();
                    break;
                case NAVIGATION_MODE_TABS:
                    setNavigationModeTabs();
                    break;
            }
        }

        @Override
        public void setExpensesSummary(@IDateMode int dateMode)
{
            float                    total                    =
Expense.getTotalExpensesByDateMode(dateMode);
            tvTotal.setText(Util.getFormattedCurrency(total));
            String date;
```

```
            switch (dateMode) {
                case IDateMode.MODE_TODAY:
                    date                                        =
Util.formatDateToString(DateUtils.getToday(),
Util.getCurrentDateFormat());
                    break;
                case IDateMode.MODE_WEEK:
                    date                                        =
Util.formatDateToString(DateUtils.getFirstDateOfCurrentWeek(),
Util.getCurrentDateFormat()).concat("                           -
").concat(Util.formatDateToString(DateUtils.getRealLastDateOfCur
rentWeek(), Util.getCurrentDateFormat()));
                    break;
                case IDateMode.MODE_MONTH:
                    date                                        =
Util.formatDateToString(DateUtils.getFirstDateOfCurrentMonth(),
Util.getCurrentDateFormat()).concat("                           -
").concat(Util.formatDateToString(DateUtils.getRealLastDateOfCur
rentMonth(), Util.getCurrentDateFormat()));
                    break;
                default:
                    date = "";
                    break;
            }
            tvDate.setText(date);
        }

        @Override
        public   void   setFAB(@DrawableRes   int   drawableId,
View.OnClickListener onClickListener) {

mFloatingActionButton.setImageDrawable(getResources().getDrawabl
e(drawableId));
```

```
mFloatingActionButton.setOnClickListener(onClickListener);
            mFloatingActionButton.show();
        }

        @Override
        public void setFAB2(int drawableId, View.OnClickListener
onClickListener) {

mFloatingActionButtonDocument.setOnClickListener(onClickListener
);
            mFloatingActionButtonDocument.show();
        }

        @Override
        public void setTitle(String title) {
            getSupportActionBar().setTitle(title);
        }

        @Override
        public     void     setPager(ViewPager     vp,     final
TabLayout.ViewPagerOnTabSelectedListener
viewPagerOnTabSelectedListener) {
            mainTabLayout.setupWithViewPager(vp);
            mainTabLayout.setOnTabSelectedListener(new
TabLayout.ViewPagerOnTabSelectedListener(vp) {
                @Override
                public void onTabSelected(TabLayout.Tab tab) {
                    @IDateMode int dateMode;
                    switch (tab.getPosition()) {
                        case 1:
                            dateMode = IDateMode.MODE_WEEK;
                            break;
                        case 2:
```

```
                                dateMode = IDateMode.MODE_MONTH;
                                break;
                            default:
                                dateMode = IDateMode.MODE_TODAY;
                        }
                        setExpensesSummary(dateMode);

viewPagerOnTabSelectedListener.onTabSelected(tab);
                    }

                    @Override
                    public void onTabUnselected(TabLayout.Tab tab) {

viewPagerOnTabSelectedListener.onTabUnselected(tab);
                    }
                });
                setExpensesSummary(IDateMode.MODE_TODAY);
            }

            public           ActionMode           setActionMode(final
ActionMode.Callback actionModeCallback) {
                return                    mToolbar.startActionMode(new
ActionMode.Callback() {
                    @Override
                    public    boolean    onCreateActionMode(ActionMode
mode, Menu menu) {

mainDrawerLayout.setDrawerLockMode(DrawerLayout.LOCK_MODE_LOCKED
_CLOSED);
                        return
actionModeCallback.onCreateActionMode(mode,menu);
                    }

                    @Override
```

```
                public boolean onPrepareActionMode(ActionMode
mode, Menu menu) {
                    return
actionModeCallback.onPrepareActionMode(mode, menu);
                }

                @Override
                public boolean onActionItemClicked(ActionMode
mode, MenuItem item) {
                    return
actionModeCallback.onActionItemClicked(mode, item);
                }

                @Override
                public void onDestroyActionMode(ActionMode mode)
{

mainDrawerLayout.setDrawerLockMode(DrawerLayout.LOCK_MODE_UNLOCK
ED);
                    actionModeCallback.onDestroyActionMode(mode);
                }
            });
        }

        private void setNavigationModeTabs() {
            mainTabLayout.setVisibility(View.VISIBLE);
            llExpensesSummary.setVisibility(View.VISIBLE);
        }

        private void setNavigationModeStandard() {
            CoordinatorLayout coordinator = (CoordinatorLayout)
findViewById(R.id.main_coordinator);
            AppBarLayout appbar = (AppBarLayout)
findViewById(R.id.app_bar_layout);
```

```
            CoordinatorLayout.LayoutParams       params       =
(CoordinatorLayout.LayoutParams) appbar.getLayoutParams();
            AppBarLayout.Behavior           behavior           =
(AppBarLayout.Behavior) params.getBehavior();
            if (behavior != null && appbar != null) {
                int[] consumed = new int[2];
                behavior.onNestedPreScroll(coordinator,  appbar,
null, 0, -1000, consumed);
            }
            mainTabLayout.setVisibility(View.GONE);
        }

        @SuppressLint("NonConstantResourceId")
        private void switchFragment(int menuItemId) {
            Fragment              currentFragment              =
getSupportFragmentManager().findFragmentById(R.id.main_content);
            switch (menuItemId) {
                case R.id.nav_expenses:
                    if       (!(currentFragment       instanceof
ExpensesContainerFragment))
replaceFragment(ExpensesContainerFragment.newInstance(), false);
                    break;
                case R.id.nav_categories:
                    if       (!(currentFragment       instanceof
CategoriesFragment))
replaceFragment(CategoriesFragment.newInstance(), false);
                    break;
                case R.id.nav_statistics:
                    if       (!(currentFragment       instanceof
StatisticsFragment))
replaceFragment(StatisticsFragment.newInstance(), false);
                    break;
                case R.id.nav_documents:
```

```
                    if       (!(currentFragment       instanceof
FolderFragment))    replaceFragment(FolderFragment.newInstance(),
false);
                    break;
                case R.id.nav_history:
                    if       (!(currentFragment       instanceof
HistoryFragment))  replaceFragment(HistoryFragment.newInstance(),
false);
                    break;
                case R.id.nav_cal:
                    startActivity(new
Intent(MainActivity.this,ExCalculatorActivity.class));
                    break;
            }
        }

        private void showAboutsDialog() {
            final     Dialog     d     =     new     Dialog(this,
android.R.style.Theme_Translucent);

d.getWindow().setBackgroundDrawableResource(android.R.color.tran
sparent);
            d.requestWindowFeature(Window.FEATURE_NO_TITLE);
            d.setContentView(R.layout.popup_about);
            d.setCancelable(false);

            final         RelativeLayout         finish         =
d.findViewById(R.id.finish);
            final          ImageView          imvBack          =
d.findViewById(R.id.imvBack);
            final          TextView          appVName          =
d.findViewById(R.id.appVersionName);
```

```
            finish.setOnClickListener(new View.OnClickListener()
{
                @Override
                public void onClick(View v) {
                    d.dismiss();
                }
            });
            imvBack.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    d.dismiss();
                }
            });

            String appVersionName = getAppVersionName(this);
            if           (!appVersionName.isEmpty()           ||
appVersionName.equals("")) {
                appVName.setText(getString(R.string.version)  +"
"+ appVersionName);
            } else {

                appVName.setVisibility(View.GONE);
            }

            d.show();
        }

        public String getAppVersionName(Context context) {
            String versionName =EMPTY_STRING;
            try {
                versionName=
context.getPackageManager().getPackageInfo(context.getPackageNam
e(), 0).versionName;
```

```
            } catch (PackageManager.NameNotFoundException e) {

            }
            return versionName;
        }


        @Override
        public void onBackPressed() {
            new OoOAlertDialog.Builder(MainActivity.this)
                    .setTitle(getString(R.string.app_name))
                    .setTitleColor(R.color.primary)
                    .setAnimation(Animation.ZOOM)
                    .setMessage("Are  You  Sure,want  exit  From
App?")
                    .setMessageColor(R.color.primary)
                    .setPositiveButtonColor(R.color.primary)
                    .setPositiveButtonTextColor(R.color.white)
                    .setNegativeButtonTextColor(R.color.white)
                    .setNegativeButtonColor(R.color.primary)
                    .setPositiveButton("Yes",                new
OnClickListener() {
                        @Override
                        public void onClick() {
                            finish();
                        }
                    })
                    .setNegativeButton("No",  null)
                    .build();

        }

    }
```