МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
01 червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Мобільний додаток для підрахунку фінансових витрат»
здобувача групи ІН-03 Устименка Дмитра Олександровича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____Дмитро УСТИМЕНКО
(підпис)

Керівник,
старший викладач кафедри
комп'ютерних наук,
Кандидат технічних наук Артем КОРОБОВ _____
(підпис)

Суми – 2024

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«Затверджую»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

_____

(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»

здобувача групи ІН-03 Устименка Дмитра Олександровича

1. Тема роботи: «Мобільний додаток для підрахування фінансових витрат»

затверджую наказом по СумДУ від *«» червня 2024 р.*

2. Термін здачі здобувачем кваліфікаційної роботи *до червня 2024 року*

3. Вхідні дані до кваліфікаційної роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1) Аналіз проблеми та актуальності розробки додатку, цільової аудиторії, постановка й формування завдань дослідження. 2) Огляд та вибір програмних засобів. 3) Проектування та розроблення мобільного додатку для підрахування фінансових витрат. 4) Аналіз отриманих результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____

(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз потреб користувачів та актуальності розробки додатку, готових рішень та цільової аудиторії, постановка й формування завдань дослідження.*	06.05.2024 – 08.05.2024	
2	*Огляд та вибір програмних засобів.*	09.05.2024 – 10.05.2024	
3	*Проектування та розроблення мобільного додатку для підрахунку фінансових витрат.*	11.05.2024 – 18.05.2024	
4	*Аналіз отриманих результатів.*	18.05.2024 – 19.05.2024	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи.*	19.05.2024 – 23.05.2024	

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 100 стор., 32 рис., 0 табл., 2 додатки , 10 використаних джерел.

Обґрунтування актуальності теми роботи – У сучасному суспільстві управління особистими фінансами є дуже важливим аспектом життя кожної людини. З розвитком цифрових технологій і більш широким використанням мобільних пристроїв мобільні додатки стали невід'ємною частиною повсякденного життя. Зокрема, мобільні додатки для розрахунку фінансових витрат стають все більш популярними і актуальними.

Об’єкт дослідження – додаток для підрахунку фінансових витрат.

Мета роботи – дослідження, аналіз та розробка мобільного додатку для підрахунку фінансових, який допоможе користувачу більш ретельніше рахувати свої витрати.

Методи дослідження - аналіз існуючих додатків, інструменти проектування та розробки.

Результати - розроблено інформаційну систему, яка ефективно веде підрахунок витрат, забезпечуючи зручний інтерфейс для користувачів та можливість проглядати більшу статистику.

Зміст

Вступ

У сучасному світі технологій, що стрімко розвиваються, та все більш активного способу життя ефективне управління фінансами є невід'ємною частиною успішного життя. Історично склалося так, що люди фіксували свої витрати у різний спосіб, зокрема у книгах та електронних таблицях. Однак з поширенням мобільних технологій мобільні додатки для відстеження витрат стали найбільш зручним і доступним інструментом для багатьох людей.

Метою моєї бакалаврської роботи є дослідження, розробка та аналіз мобільного додатку для відстеження витрат, який відповідає сучасним вимогам та потребам користувачів. Моя мета - розробити ефективні та корисні інструменти, які дозволять користувачам краще розуміти фінанси своїх домогосподарств та приймати обґрунтовані рішення щодо витрат.

Це важливо, оскільки дедалі складніший фінансовий світ вимагає ефективних інструментів для управління та планування витрат. Оскільки ціни на товари та послуги постійно зростають, а спосіб життя змінюється, стає все більш важливим не лише відстежувати витрати, але й аналізувати їх та оптимізувати процес прийняття фінансових рішень. У цьому контексті розробка мобільних додатків для відшкодування витрат є важливою та актуальною темою, яка може сприяти розвитку фінансової грамотності та ефективному управлінню фінансами серед широкої аудиторії користувачів.

Актуальність теми.

Сучасне суспільство стикається з багатьма викликами у сфері фінансів. Ускладнення фінансового світу, швидка зміна економічних умов та збільшення фінансових можливостей створюють нові виклики та вимоги до людей. У цьому контексті потреба в ефективному управлінні особистими фінансами стає дуже важливою.

Мобільні додатки для управління витратами займають особливе місце серед інструментів фінансового менеджменту. Вони забезпечують зручний та швидкий спосіб обліку витрат та аналізу фінансової діяльності. Завдяки мобільним додаткам користувачі можуть більш ефективно управляти своїми фінансами, планувати свої витрати та заощаджувати кошти.

Однак через постійний розвиток технологій та мінливі умови життя мобільні додатки для відшкодування витрат потребують постійного вдосконалення та адаптації. Беручи до уваги сучасні тенденції у фінансовому секторі, вкрай важливо визначити потреби та очікування користувачів і застосувати новітні технології для покращення якості та зручності користування додатком.

Тому розробка та аналіз мобільних додатків для калькуляції витрат є надзвичайно важливим завданням, яке відповідає сучасним вимогам та потребам суспільства. Результати досліджень та розробки нових додатків матимуть значні наслідки з точки зору підвищення фінансової грамотності та покращення фінансового стану користувачів.

1. Аналіз потреб користувачів

В рамках оцінки потреб користувачів буде проведено опитування потенційних користувачів мобільних додатків для обліку витрат на основі широкого обміну інформацією зі споживачами. Основна мета цього етапу - зрозуміти потреби, очікування та пріоритети користувачів щодо використання таких додатків.

Для того, щоб адекватно проаналізувати потреби користувачів, основна увага буде зосереджена на наступних аспектах:

Функціональність: розуміння того, які конкретні можливості та функції є важливими для користувачів у мобільному додатку для калькуляції витрат. До них відносяться функції додавання витрат, створення категорій, аналізу витрат і бюджетування.

Зручність використання: визначення того, які інтерфейсні рішення та функції навігації є найбільш зручними для користувача. Сюди входить розташування елементів у додатку, управління жестами та загальні шаблони взаємодії.

Дизайн та інтерфейс: Визначення дизайну і візуального стилю, який є найбільш привабливим і зрозумілим для цільового користувача.

Безпека та конфіденційність: забезпечити конфіденційність інформації про витрати, враховуючи вимоги користувача щодо захисту приватного життя.

На основі зібраної інформації визначити основні потреби користувачів та відобразити їх у подальшій розробці мобільного додатку.

В рамках аналізу потреб користувачів калькуляційних програм я прагну краще зрозуміти потреби, очікування та пріоритети користувачів. Для досягнення цієї мети я планую провести аналіз уже існуючих програм для підрахунку витрат.

Одним з ключових аспектів мого дослідження є функціональність додатку. Я хочу дізнатися, які функції та можливості є найбільш важливими для користувачів. Це включає можливість додавати, відстежувати та класифікувати витрати, аналізувати звіти про витрати та функції бюджетного планування.

Крім того, я також розглядаю зручність використання програми. Важливо розуміти, які інтерфейсні рішення та методи навігації є найбільш зручними для користувачів. Сюди входить розташування елементів всередині додатку, взаємодія з екраном за допомогою жестів і типові патерни взаємодії.

Далі йде зосередження на дизайні та інтерфейсі додатку. Я хочему визначити, який дизайн і візуальний стиль є найбільш привабливим і зрозумілим для користувача.

Важливу роль відіграє питання безпеки та конфіденційності. Я врахую всі вимоги щодо захисту персональних даних користувачів і прагнутиму забезпечити конфіденційність витрат.

Тому метою аналізу потреб користувачів є отримання повного уявлення про потреби та очікування користувачів, щоб ми могли розробляти мобільні додатки, які відповідають їхнім потребам і забезпечують приємний та зручний користувацький досвід.

При аналізі потреб користувачів увага також приділяється їхнім характеристикам і поведінці. До уваги беруться різні аспекти, такі як вік, стать, освіта, дохід та інші параметри, які можуть вплинути на вимоги до програми.

При зборі даних увага приділяється різним групам користувачів, таким як студенти, молоді сім'ї та бізнесмени. Таким чином можна отримати різноманітний перелік потреб і вимог різних категорій користувачів, які можуть бути враховані при подальшій розробці додатку.

Крім того, враховуються індивідуальні особливості користувачів, такі як їхні особисті вподобання, спосіб життя та фінансові звички. Це забезпечує персоналізований досвід використання додатку, який відповідає конкретним потребам кожного користувача.

Загалом, моя мета - підвищити рівень задоволеності користувачів додатком для прогнозування витрат шляхом ретельного вивчення потреб і вимог користувачів та їхнього врахування в процесі проектування і розробки додатку.

1.1 Опис проблеми

У сучасному суспільстві все більше людей відчувають фінансову нестабільність через погане управління витратами. Загальновідомо, що поганий фінансовий менеджмент може призвести до таких ризиків, як перевитрата бюджету, збільшення боргу або фінансова нестабільність.

Однією з головних причин такої ситуації є відсутність ефективних інструментів для моніторингу та аналізу особистих витрат. Традиційні методи управління бюджетом, такі як паперові щоденники та електронні таблиці, є неефективними і забирають багато часу. Багато людей шукають більш зручні та ефективні способи контролювати свої фінанси в режимі реального часу.

У цьому контексті важливість створення мобільного додатку для підрахунку витрат очевидна. Такий додаток може надати користувачам зручний інструмент для ведення домашнього бюджету, допомогти їм планувати та управляти своїм бюджетом, а також сприяти підвищенню фінансової грамотності.

1.2 Ідентифікація цільової аудиторії

Визначення цільових користувачів є важливим кроком у процесі розробки мобільного додатку для розрахунку витрат. Доцільно провести детальний аналіз різних груп потенційних користувачів, щоб чітко визначити їхні потреби та очікування щодо функціональності додатку.

1. **Студенти:** Студенти можуть бути зацікавлені у використанні додатку для управління своїм навчальним та особистим бюджетом. Вони можуть шукати можливості заощаджувати на їжі, книгах, розвагах та інших необхідних витратах.

2. **Молоді сім'ї:** Молоді сім'ї можуть використовувати додаток як інструмент для ведення сімейного бюджету. Вони можуть шукати інструменти для відстеження сімейних витрат на продукти, дитячі товари, освіту, розваги тощо.

3. **Фрілансери та самозайняті:** Фрілансери та самозайняті люди можуть використовувати додаток як інструмент для ведення сімейного бюджету. Вони можуть шукати можливості відстежувати вартість своїх послуг, оплачувати рахунки та планувати бюджети для майбутніх проектів.

4. **Бізнес-користувачі:** Підприємці та бізнес-користувачі можуть використовувати додаток для управління фінансами у своєму бізнесі. Його можна використовувати як інструмент для відстеження виробничих витрат, реклами, зарплат та інших потреб бізнесу.

5. **Прихильник особистого фінансового планування:** Люди, які приділяють увагу фінансовій дисципліні, можуть використовувати додаток для підтримки своїх фінансових цілей. Вони можуть шукати інструменти для відстеження своїх витрат, планування заощаджень та моніторингу особистих фінансів.

Враховуючи різноманітність нашої цільової аудиторії, я маю можливість розробити додаток, який найкраще буде відповідати потребам та очікуванням кожної з цих груп користувачів.

1.3 Вимоги користувачів до додатку

Вимоги користувачів є ключовим елементом у розробці мобільних додатків для калькуляції витрат. Чітке розуміння потреб та очікувань користувачів допоможе створити продукт, який відповідатиме їхнім потребам. Нижче наведено загальні вимоги, які користувачі можуть мати до таких додатків:

1. **Простота використання:** Користувачі очікують, що додатки будуть простими та інтуїтивно зрозумілими у використанні. Вони хочуть мати можливість швидко та ефективно вносити витрати, не витрачаючи час на непотрібні операції.

2. **Функціональність:** Користувачі очікують можливості додавати витрати, створювати категорії витрат, аналізувати витрати за періодами, встановлювати бюджети та нагадування тощо.

3. **Візуалізація даних:** Користувачів може зацікавити можливість візуального аналізу фінансових даних. Такі функції, як графіки, діаграми та звіти, допоможуть їм краще зрозуміти свої витрати та ефективніше управляти бюджетом.

4. **Синхронізація даних:** Користувачам може знадобитися можливість синхронізувати дані на різних пристроях і мати доступ до них у будь-який час і з будь-якого місця.

2. Аналіз та постановка задачі

Аналіз та визначення завдань - важливий етап розробки мобільного додатку для калькуляції витрат. На цьому етапі визначаються конкретні завдання, які необхідно вирішити для досягнення поставлених цілей і задоволення потреб користувачів.

Основними завданнями є наступні:

1. **Розробка користувацького інтерфейсу:** Необхідно створити зручний та інтуїтивно зрозумілий інтерфейс, щоб користувачі могли легко вводити витрати та користуватися різними функціями додатку.

2. **Створення функцій додатку:** додавання витрат, створення категорій витрат, аналіз витрат, функції управління бюджетом тощо. Кожна функція повинна бути ретельно протестована, щоб забезпечити її ефективність та належне функціонування.

3. **Оптимізація продуктивності:** Додаток повинен працювати швидко і без затримок, навіть при обробці великих обсягів даних. Оптимізація коду та використання ефективних алгоритмів необхідні для забезпечення швидкої роботи додатку.

4. **Тестування та оптимізація:** Після розробки слід провести ретельне тестування додатку, щоб виявити та виправити можливі помилки та

недоліки. Також необхідно прислухатися до відгуків користувачів і вносити відповідні зміни для покращення додатку.

Загальна мета цих завдань - створити якісний мобільний додаток, який відповідає потребам користувачів і забезпечує зручність, безпеку та ефективність фінансового обліку.

2.1 Огляд існуючих рішень на ринку

2.1.1 Money Manager Expense & Budget

Додаток налаштовується і має зручний інтерфейс: ви можете вибрати стиль, темний або світлий режим і колірну схему.(Рис.1.1)

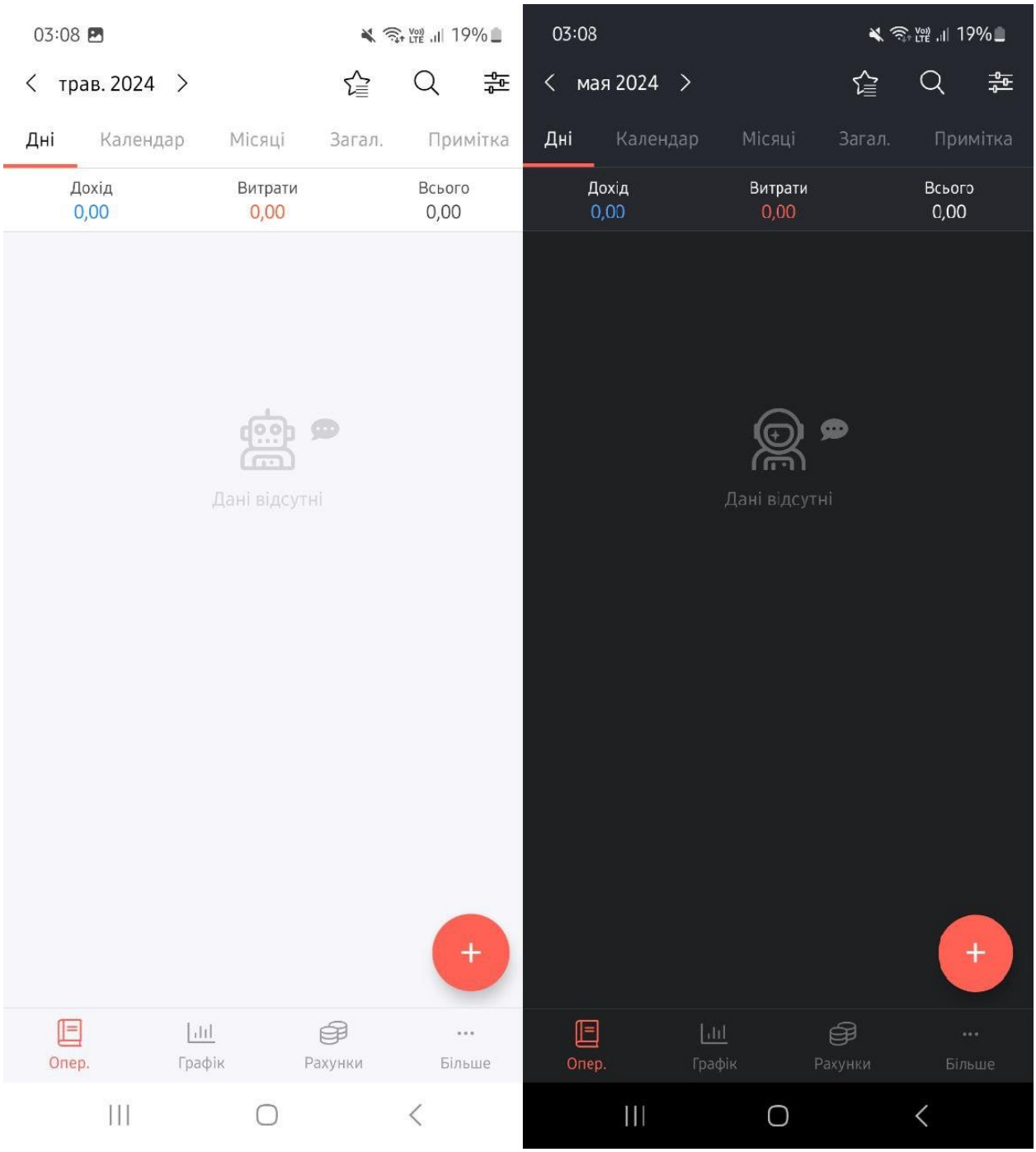


Рисунок 1.1-Зміна теми в додатку Money Manager Expense & Budget

Всі витрати зберігаються у вигляді списку на головній сторінці. Це може бути незручно, якщо у вас багато витрат. Поруч з кожною витратою відображається маленька картинка з категорією. Це дозволяє легко зрозуміти, яка категорія є домінуючою.(Рис.1.2)

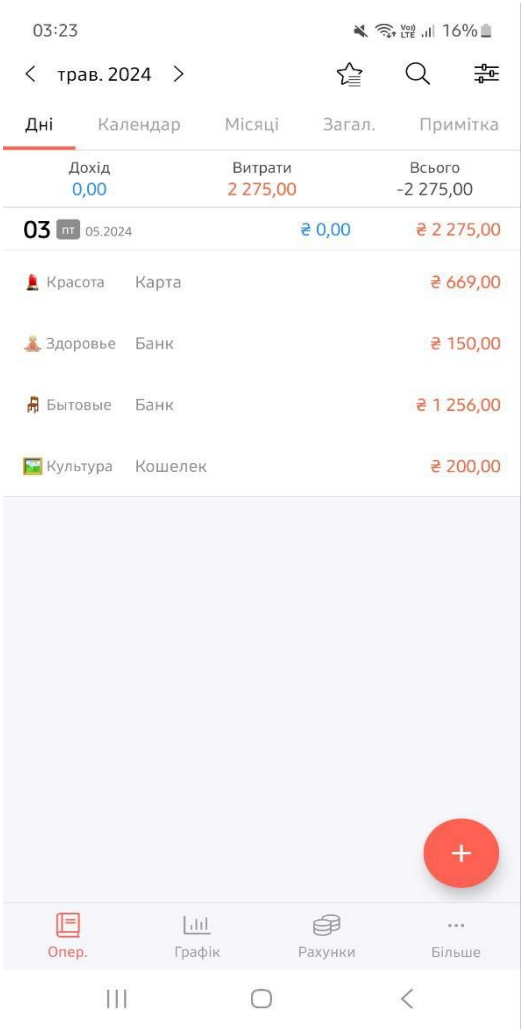


Рисунок 1.2 – Список витрат

У додатку є тижневі, місячні та річні графіки витрат і доходів. Ви також можете відкрити календар витрат і додати витрати пізніше. Ви можете оплатити витрати карткою або готівкою.(Рис.1.3)

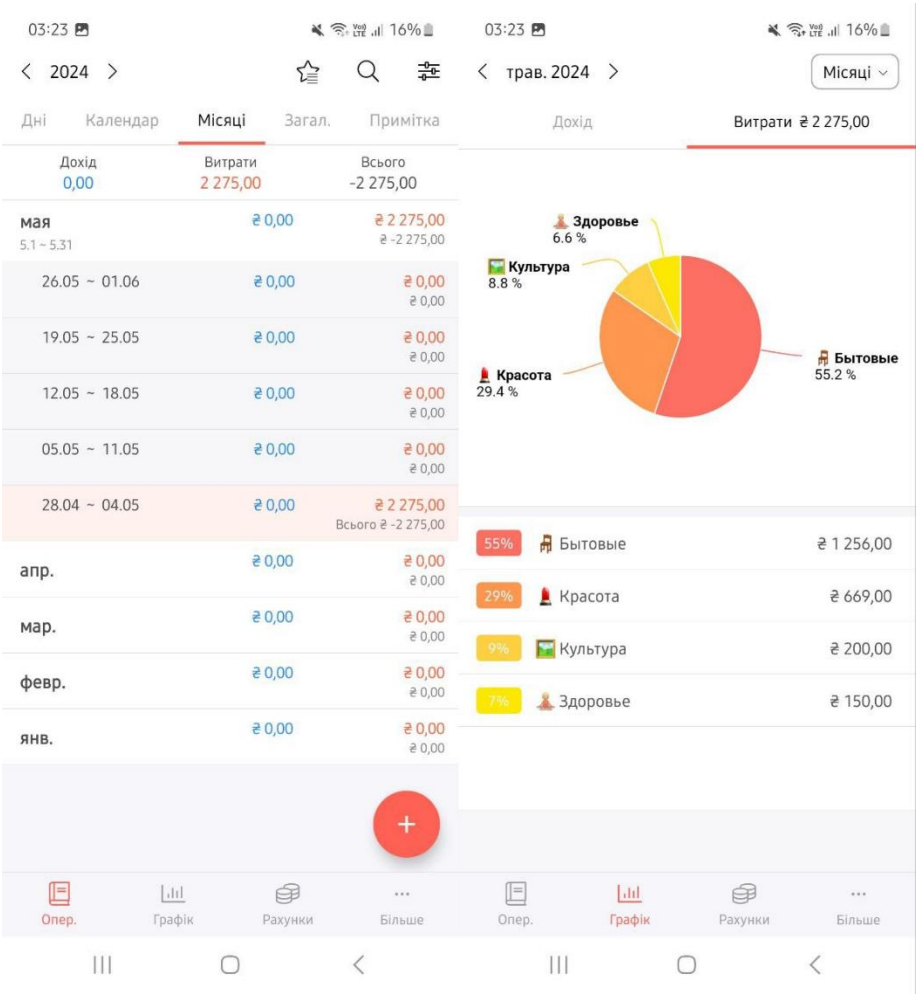


Рисунок 1.3 – Статистика в додаткі

Найбільшою перевагою Money Manager Expense & Budget є те, що ви можете додавати власні категорії та створювати підкатегорії. Вам лише потрібно встановити пароль, щоб захистити свої особисті дані.Недоліком цього додатку є велика кількість реклами. Однак їх розташування не заважає повноцінно користуватися додатком.

Існує платна версія цього додатку: За 6,99 USD (версія для IOS) ви можете здихатися від реклами, отримати необмежену кількість категорій та користуватися версією для персональних комп'ютерів.

2.1.2 Monefy

Додаток має зручний та інтуїтивно зрозумілий інтерфейс. Всі категорії витрат відображаються на головному екрані у вигляді малюнка. Розподіл

витрат також відображається на головному екрані у вигляді діаграми. Це дозволяє швидко побачити, на які категорії витрачається найбільше коштів.(Рис.2.1). За 1 050 грн доступна платна версія застосунку. Тоді з’явиться можливість додавати власні категорії, синхронізуватися з кількома пристроями, додавати різні валюти.

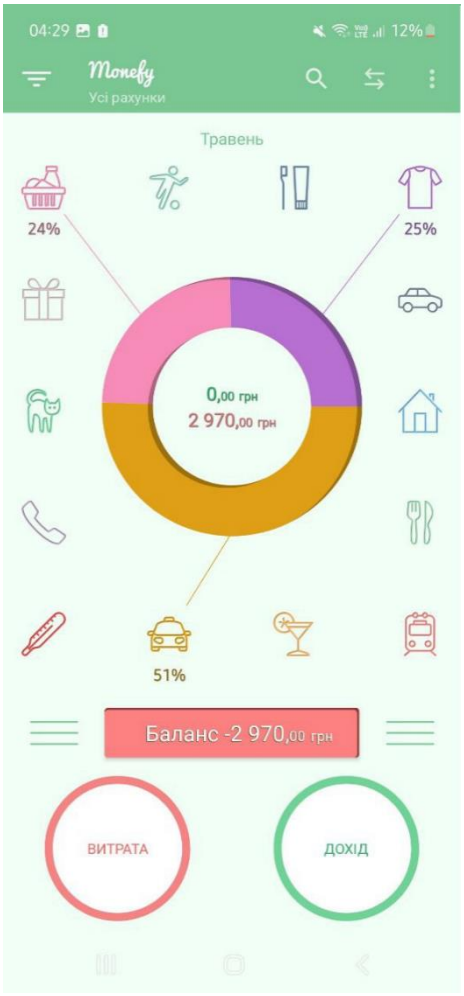


Рисунок 2.1 – Головний екран

Ви можете встановити звітні періоди: один день, один тиждень або один місяць, а також розділити витрати по картках і готівкою.(Рис 2.2)

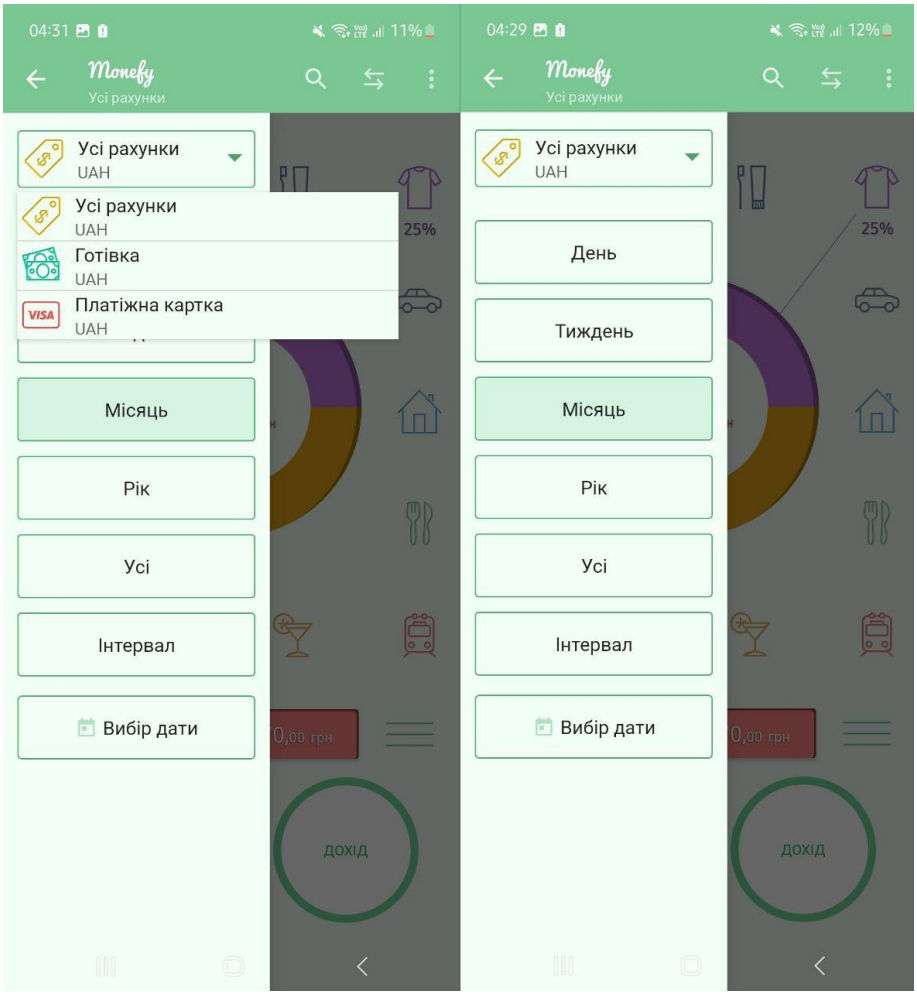


Рисунок 2.2 – Реалізація звітних періодів, а також розподіл витрат.

2.2 Формулювання завдань для розробки додатку

Формулювання завдань для розробки мобільного додатку для підрахунку витрат - це ключовий етап, який дозволяє чітко визначити, що саме потрібно реалізувати під час розробки програмного продукту. Ось кілька завдань, які можна сформулювати для цього процесу:

1. Створення користувацького інтерфейсу: Розробити інтерфейс користувача, який буде зручним та інтуїтивно зрозумілим для користувачів. Це включає в себе розміщення елементів у додатку, розробку графічного дизайну та навігаційні можливості.

2. Реалізація функціональності: Розробити функції додатку, які дозволять користувачам додавати, видаляти та редагувати свої витрати, створювати категорії витрат, переглядати звіти про витрати.

3. Оптимізація продуктивності: Забезпечити оптимальну продуктивність додатку, зокрема швидке завантаження та відкликання даних, ефективне використання ресурсів пристрою та мінімізація споживання енергії.

4. Тестування та валідація: Провести тестування додатку на різних пристроях та платформах, виявити та виправити помилки та недоліки, а також переконатися в його відповідності вимогам та очікуванням користувачів.

3. Технічний опис додатку

3.1 Архітектура додатку

Для свого додатку я обрав архітектуру Model-View-Controller (MVC). У цій архітектурі модель відповідає за управління даними та бізнес-логікою, представлення відповідає за відображення даних та взаємодію з користувачем, а контролер виконує логіку додатку та координує взаємодію між моделлю та представленням. Вибравши таку архітектуру, можна ефективно розподілити обов'язки між різними компонентами додатку, що спрощує розробку та супровід програмного забезпечення.

Що ж таке взагалі Model-View-Controller(MVC) що це за архітектура і що вона собою являє?

Архітектура " Model-View-Controller " (MVC) є однією з найпоширеніших архітектур при розробці програмного забезпечення, в тому числі мобільних додатків. Вона складається з трьох основних компонентів:

Модель(Model): управляє даними та бізнес-логікою додатку. Модель - це об'єкт або набір об'єктів, який зберігає дані і виконує операції над цими даними.

Перегляд(View): Відображає дані користувачеві та збирає вхідні дані від користувача. Перегляди можуть мати різноманітні користувацькі інтерфейси, включаючи екранні форми, кнопки і текст.

Контролер: відповідає за обробку введених користувачем даних, взаємодію з моделлю та оновлення подання. Контролер виступає посередником між моделлю і представленням, обробляючи запити користувача і виконуючи дії, необхідні для оновлення стану додатку.

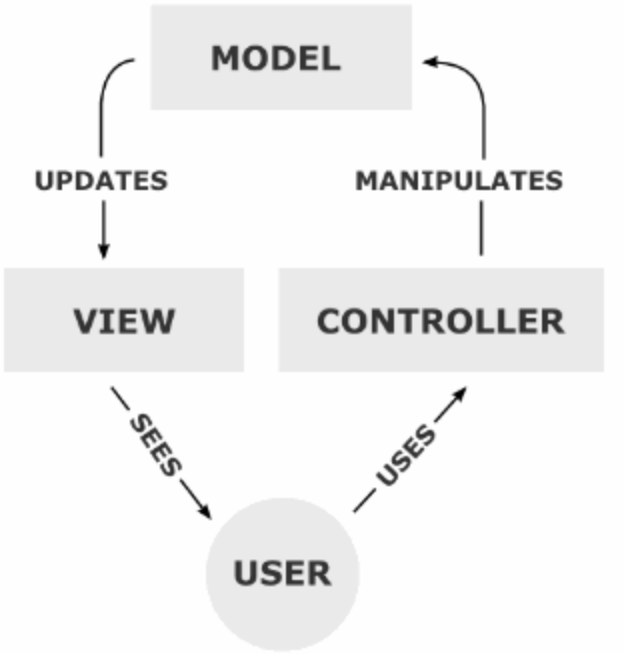


Рисунок 3 – Приклад схема архітектури MVC.

До переваг архітектури MVC можна віднести

1. Поділ обов'язків: MVC спрощує розробку та підтримку додатків, забезпечуючи чітке розділення логіки додатку, представлення даних та управління даними.

2. Підтримка тестування: розділення компонентів дозволяє тестувати моделі, представлення та контролери окремо, що полегшує написання тестів та виявлення помилок.
3. Гнучкість: модель, представлення і контролер можуть бути змінені або замінені без впливу на інші компоненти, що робить архітектуру більш гнучкою.

До недоліків архітектури MVC можна віднести наступні:

1. Складність: складність може виникнути через необхідність керувати великою кількістю контролерів та їх взаємодією з моделями та представленнями.
2. Взаємодія між компонентами: складна взаємодія може викликати проблеми з розподілом обов'язків між компонентами.

Загалом, архітектура MVC є потужним інструментом для розробки додатків, забезпечуючи чітку структуру та підтримуючи гнучкий і тестований код.[1]

3.2 Використані технології та інструменти

У моєму проекті я використав наступні технології та інструменти:

1.Java.

2.Android SDK (Software Development Kit).

3.SQLite.

4.XML (Extensible Markup Language).

5.Android Studio.

6.Gradle.

7.Архітектура Model-View-Controller (MVC)(Прицнип роботи і що собою являє описано вище).

3.2.1 Java

Java - це об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems у 1990-х роках. Вона характеризується

простотою, надійністю та портативністю і широко використовується для розробки широкого спектру додатків, включаючи веб-додатки, мобільні додатки та інші програми.

Принцип роботи Java полягає в тому, що код програми компілюється в байт-код, який виконується на віртуальній машині Java (JVM). Це означає, що розробникам не потрібно модифікувати код для кожної конкретної операційної системи або апаратної платформи, а можна написати код один раз і запускати його на будь-якій платформі, яка має JVM.

Деякі з переваг Java включають:

1. Простота вивчення та використання
2. Надійність і безпека
3. Переносимість на різні платформи
4. Велика екосистема бібліотек та інструментів розробки

Однак, Java також має наступні недоліки

1. Високе споживання пам'яті та ресурсів
2. Низька продуктивність порівняно з іншими мовами
3. Не дуже добре підходить для низькорівневих операцій

У моєму проекті я використовую Java для написання бізнес-логіки, базової логіки додатку та взаємодії з базами даних. Кожен файл з розширенням проекту .java містить код на мові програмування Java. Наприклад, клас DatabaseHelper.java використовує Java для взаємодії з базою даних SQLite. Android SDK (Software Development Kit).

```java
public class DatabaseHelper extends SQLiteOpenHelper {

    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "expenses_db";
    private static final String TABLE_EXPENSES = "expenses";
    private static final String COLUMN_ID = "id";
    private static final String COLUMN_AMOUNT = "amount";
    private static final String COLUMN_COMMENT = "comment";
    private static final String TABLE_FOOD_EXPENSES = "food_expenses";
    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        String CREATE_EXPENSES_TABLE = "CREATE TABLE " + TABLE_EXPENSES + "("
                + COLUMN_ID + " INTEGER PRIMARY KEY,"
                + COLUMN_AMOUNT + " REAL,"
                + COLUMN_COMMENT + " TEXT" + ")";
        String CREATE_FOOD_EXPENSES_TABLE = "CREATE TABLE " + TABLE_FOOD_EXPENSES + "("
                + COLUMN_ID + " INTEGER PRIMARY KEY,"
                + COLUMN_AMOUNT + " REAL,"
                + COLUMN_COMMENT + " TEXT" + ")";
        db.execSQL(CREATE_EXPENSES_TABLE);
        db.execSQL(CREATE_FOOD_EXPENSES_TABLE);

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_EXPENSES);
        onCreate(db);
    }

    public void insertExpense(double amount, String comment) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COLUMN_AMOUNT, amount);
        values.put(COLUMN_COMMENT, comment);
        db.insert(TABLE_EXPENSES, null, values);
        db.close();
```

Рисунок 3.2.1- Приклад використання Java у проєкті

3.2.2 Android SDK (Software Development Kit)

Android SDK (Software Development Kit) - це набір інструментів, бібліотек та ресурсів, що використовуються для розробки мобільних додатків для платформи Android: інструменти SDK, підтримувані бібліотеки та емулятори, документація, приклади коду та всі інші компоненти, необхідні для створення програмного забезпечення для Android.

Android SDK був представлений компанією Google у 2008 році, коли вперше була анонсована операційна система Android; Android SDK постійно оновлюється та розширюється, щоб забезпечити розробників найновішими функціями та можливостями платформи Android.

Принцип роботи Android SDK полягає в наданні розробникам доступу до інтерфейсів API, які дозволяють їм взаємодіяти з різними компонентами системи Android, такими як дії, сервіси, бази даних, мережі та графіка. Розробники можуть писати код додатків за допомогою Java або Kotlin і компілювати його в байт-код Java, який запускається на віртуальній машині Android (Dalvik або ART).

Деякі з переваг Android SDK:

Великий набір інструментів та бібліотек для розробки різноманітних додатків

Активна спільнота розробників і велика кількість документації та прикладів коду

Підтримка різних пристроїв та версій Android

Однак, до недоліків Android SDK можна віднести

Платформозалежність, що може призвести до проблем сумісності та відмінностей між версіями Android.

Складність розробки для різних екранів та пристроїв.

У моєму проєкті Android SDK використовується для розробки мобільного додатку для підрахунку витрат.Я використовуємо SDK для створення різних компонентів додатку, включаючи активності, сервіси, базу даних та взаємодію зі складовими операційної системи Android, такими як дозвіл на доступ до мережі, збереження даних тощо.

3.2.3 SQLite.

SQLite - це вбудована система управління базами даних, яка забезпечує просте, ефективне і надійне управління базами даних без необхідності встановлення додаткових серверів або додатків Вона була розроблена в 2000 році і входить в стандартний комплект розробки програмного забезпечення Android.

Принцип роботи SQLite полягає у зберіганні даних у вигляді бази даних, єдиного файлу, доступного для читання і запису; SQLite підтримує стандартні

операції з базами даних, такі як створення, читання, оновлення і видалення (CRUD), а також SQL-запити для обробки і управління даними.

Переваги використання SQLite включають:

Простота використання та вбудована підтримка в операційній системі Android.

Низьке споживання пам'яті та ресурсів.

Висока продуктивність при роботі з невеликими базами даних до 1-2 ГБ.

Цілісність даних завдяки підтримці механізмів транзакцій та блокування.

Однак SQLite також має деякі обмеження та недоліки:

Він не підтримує високі паралельні робочі навантаження або обробку великих обсягів даних.

Він не підходить для складних операцій з базами даних або великих проектів.

У моєму проекті я використовую SQLite для зберігання даних про витрати користувачів. Я створив базу даних для зберігання записів про витрати та суми для кожної категорії витрат; SQLite дозволить мені ефективно організувати та зберігати ці дані, а також мати швидкий доступ до них з програми.

3.2.4 XML

XML (Extensible Markup Language) - це мова розмітки, яка використовується для представлення та обміну структурованими даними у вигляді текстових файлів; вона була розроблена в 1996 році і широко поширена у веб-розробці та розробці програмного забезпечення в цілому.

Принцип XML полягає у створенні документа, що складається з елементів, які визначають структуру та зміст даних. Кожен елемент має ім'я і може мати атрибути і дочірні елементи; XML дозволяє використовувати власні теги і структури даних, що дозволяє дуже гнучко представляти різні типи інформації.

Деякі з переваг використання XML включають:

Легке і зручне читання та редагування даних у текстовому форматі.

Гнучкість і розширюваність для визначення власних структур і форматів даних.

Підтримка крос-платформного обміну даними між різними системами та додатками.

Однак XML має і недоліки:

Він вимагає більше місця, ніж інші формати даних, такі як JSON.

Його важко читати, розуміти і обробляти машинам, особливо коли йдеться про великі обсяги даних.

У моєму проекті я використовую XML для розмітки інтерфейсу користувача (UI).Я використовую ресурси файлів XML для визначення макета екрану, розташування елементів і властивостей. За допомогою XML я можу легко створити інтерфейс користувача програми і кастомізувати його, роблячи зручним і гнучким для користувача.

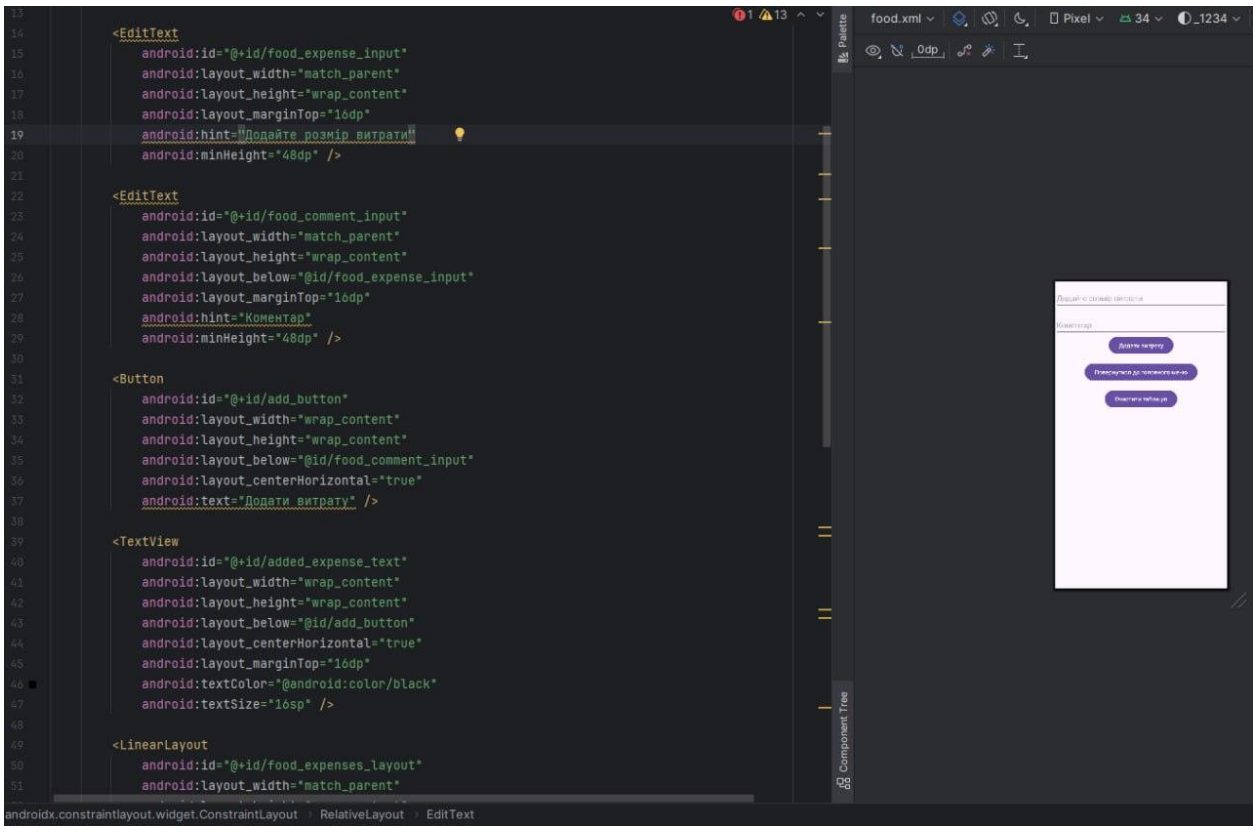


Рисунок 3.2.4 - Приклад коду який вже є у проєкті

3.2.5 Android Studio

Android Studio - це інтегроване середовище розробки (IDE), спеціально призначене для розробки мобільних додатків для платформи Android. IDE було створено компанією Google у 2013 році на основі популярної платформи IntelliJ IDEA і з тих пір стало стандартним інструментом для розробки Android-додатків.

Принцип роботи Android Studio полягає в тому, що вона надає інструменти для розробки, тестування, налагодження та розгортання Android-додатків. Вона включає в себе візуальний редактор макетів, допоміжні інструменти для автоматичного завершення коду, інструменти аналізу проектів, вбудовану підтримку контролю версій і багато іншого.Android Studio заснована на мові програмування Java і використовує систему збірки Gradle для для управління залежностями та збірки проектів.

Android Studio пропонує наступні переваги:

Потужні інструменти розробки, які спрощують процес створення Android-додатків

Інтеграція з Android SDK забезпечує легкий доступ до всіх компонентів і бібліотек, необхідних для розробки додатків.

Велика спільнота користувачів та підтримка від Google для швидкого вирішення проблем та оновлення.

Однак, Android Studio також має наступні недоліки:

Високі вимоги до системних ресурсів, що може вплинути на продуктивність на старих або слабких комп'ютерах.

Потрібен час і ресурси, щоб вивчити всі можливості і функції IDE.

У нашому проекті ми використовуємо Android Studio для розробки та тестування мобільного додатку для калькуляції витрат. Це інтегроване середовище розробки дозволяє нам швидко створювати, тестувати та налагоджувати функціональність нашого додатку, що робить наш проект більш ефективним.

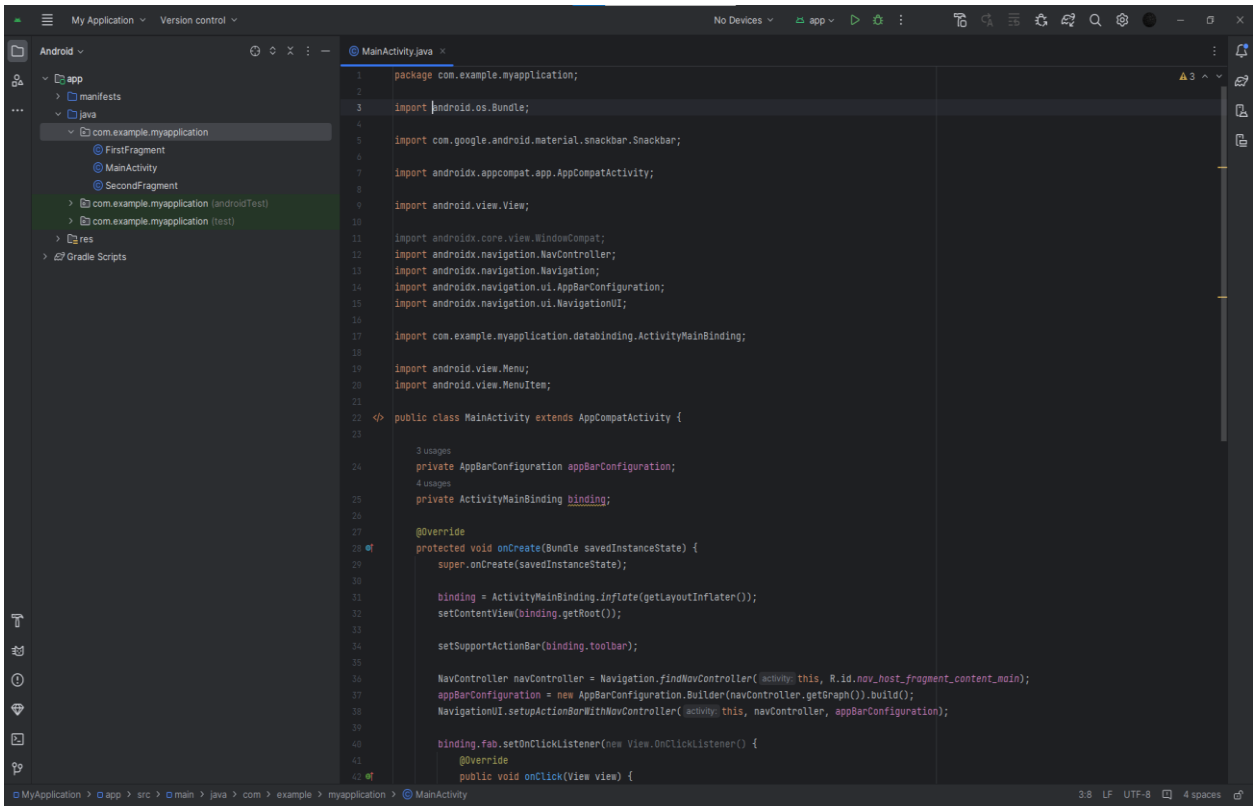


Рисунок 3.2.5 - Вигляд Android Studio

3.2.6 Gradle

Gradle - це система збірки проектів для автоматизації процесу створення, тестування та розгортання програмного забезпечення. Вона була створена для полегшення роботи над проектами різної складності та надання гнучкості розробникам.

Gradle була розроблена компанією Gradle Inc. у 2007 році як альтернатива існуючим системам збірки, таким як Apache Ant та Apache Maven.Gradle базується на мові програмування Groovy, але також підтримуються мови програмування Kotlin та Java. але також підтримуються мови програмування Kotlin та Java.

Основними особливостями Gradle є:

Декларативний підхід до побудови проекту з використанням конфігураційного файлу проекту (зазвичай build.gradle)

Гнучка система залежностей, яка дозволяє легко включати залежності з різних джерел, включаючи локальні файли та веб-репозиторії

Підтримка багатьох мов програмування, включаючи Java, Kotlin та Groovy.

Розширюваність за допомогою плагінів, що дозволяє легко налаштовувати та розширювати функціональність збірки проекту

У нашому проекті Gradle використовується для управління залежностями проекту, а також для побудови та налаштування різних аспектів розробки. Ми використовуємо Gradle для включення необхідних бібліотек і залежностей в наш проект, а також для налаштування збірки і розгортання нашого додатку Gradle дозволяє нам ефективно керувати нашим проектом і забезпечує легку розширюваність і сумісність з іншими інструментами розробки.

3.2.7 Проектування структур системи

Для проектування структури системи я вирішив створити діаграму класів яка дасть нам статичне представлення структури моделі в UML,вона відобразить статичні елементи,такі як класи,типи даних,їх зміст та відношення.На діаграмі класів будуть показані класи,об'єкти і кооперації,а також їх відносини (зв’язки).Класи представлені у вигляді прямокутників які містять:

- Назву класу
- Атрибути – описують властивості класу.
- Операції – показують які дії може виконувати клас.

Для побудови діаграми класів я скористався сайтом draw.io(Повна діаграма буде зазначена в Додатку А)

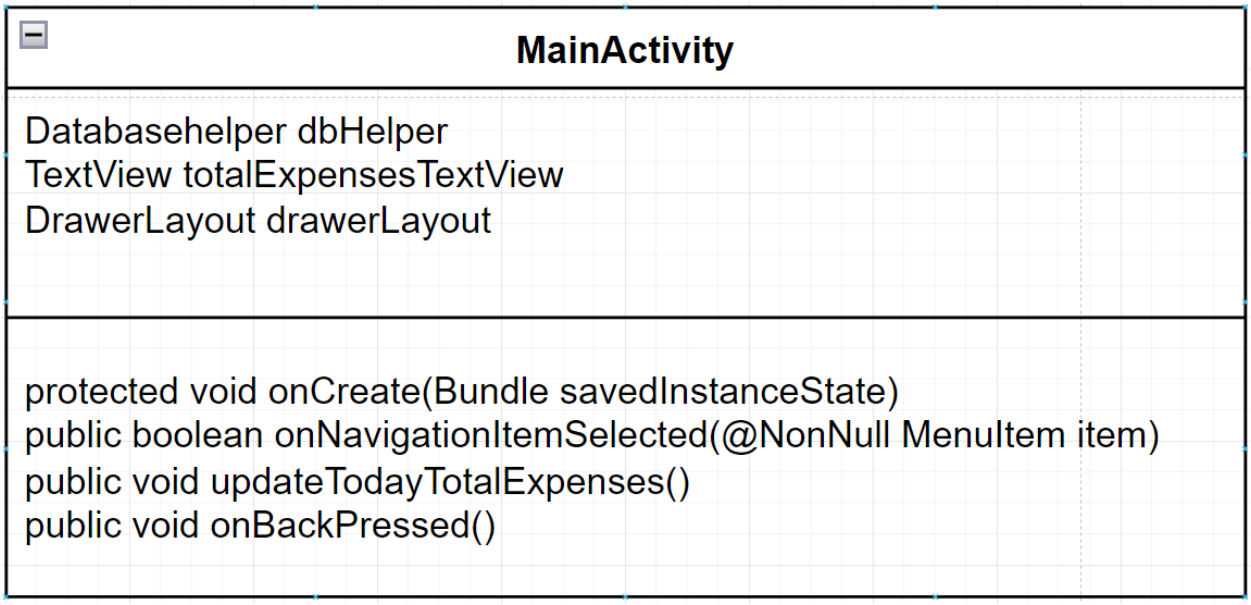


Рисунок 3.2.7.1 – Вікно MainActivity

Клас MainActivity відповідає за виведення сьогоднішніх витрат також за допомогою цього класу можна потрапити до інших вікон витрат.Також цей клас інтегрує роботу базою даних та забезпечує перехід між видами витрат.

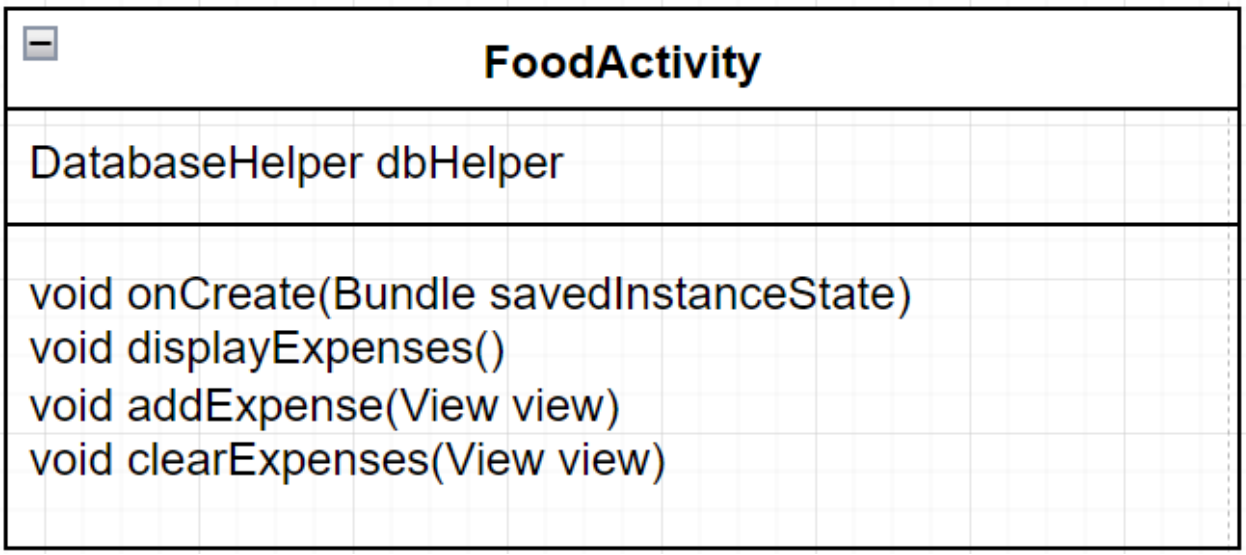


Рисунок 3.2.7.2 – Вікно FoodActivity

Клас FoodActivity відповідає за додавання витрат які пов’язані із їжою,а також цей клас виводить табличку з доданими витрати,також клас відповідає за очищення таблиці введених витрат на їжу.

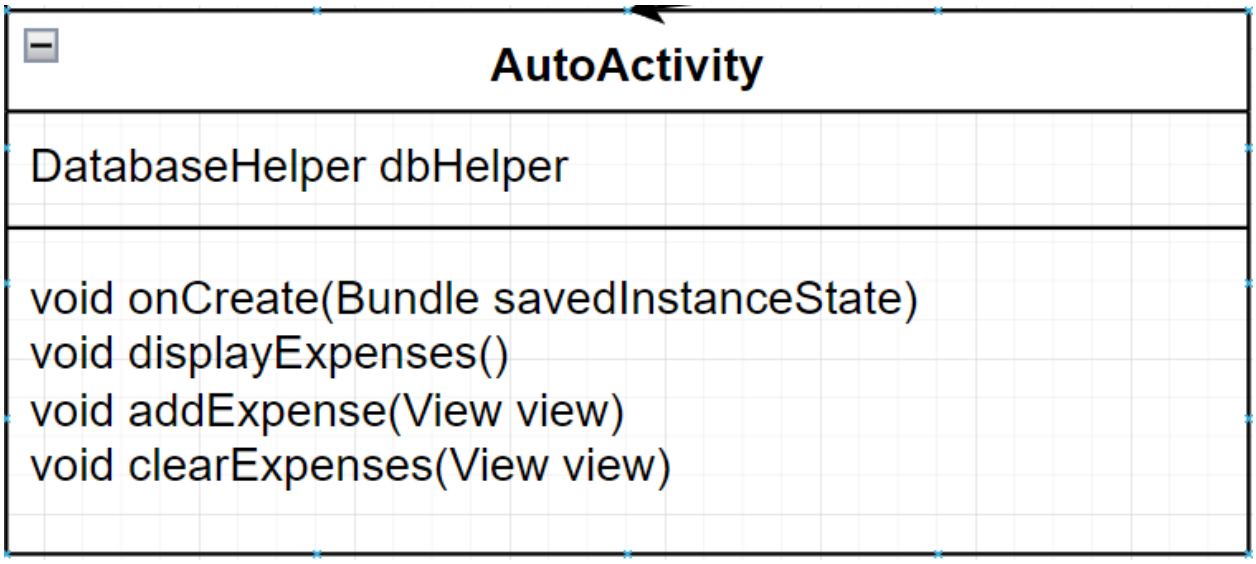


Рисунок 3.2.7.3 – Вікно AutoActivity

Клас AutoActivity відповідає за додавання витрат які пов’язані із автомобілем, а також цей клас виводить табличку з доданими витрати, також клас відповідає за очищення таблиці введених витрат на авто.

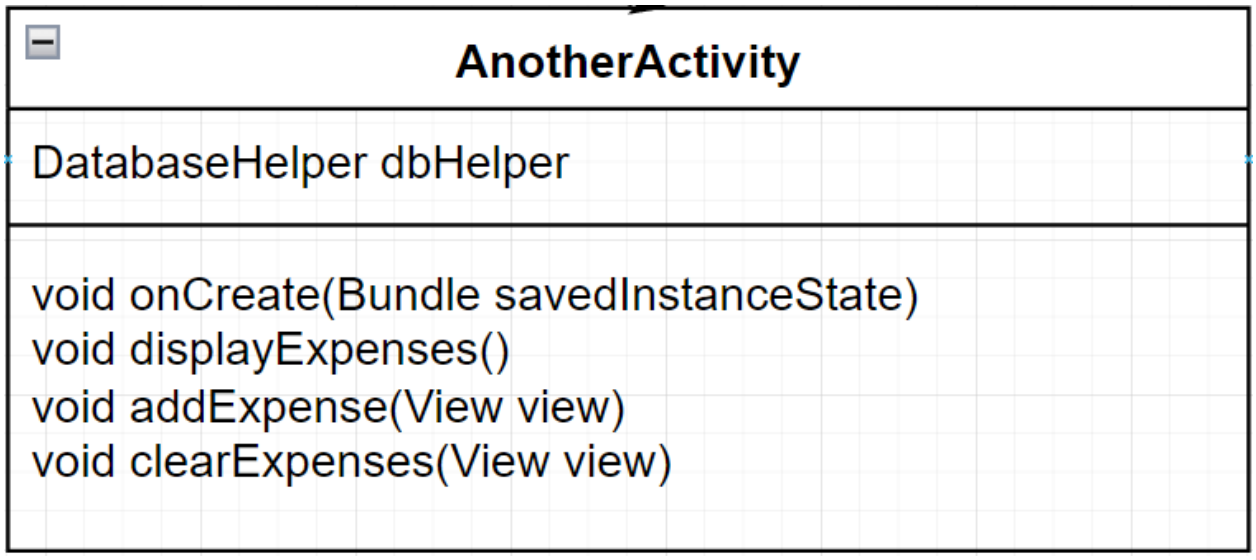


Рисунок 3.2.7.4 – Вікно AnotherActivity

Клас AnotherActivity відповідає за додавання витрат які не передбачені нашим додатком,а також цей клас виводить табличку з доданими витрати,також клас відповідає за очищення таблиці введених витрат.

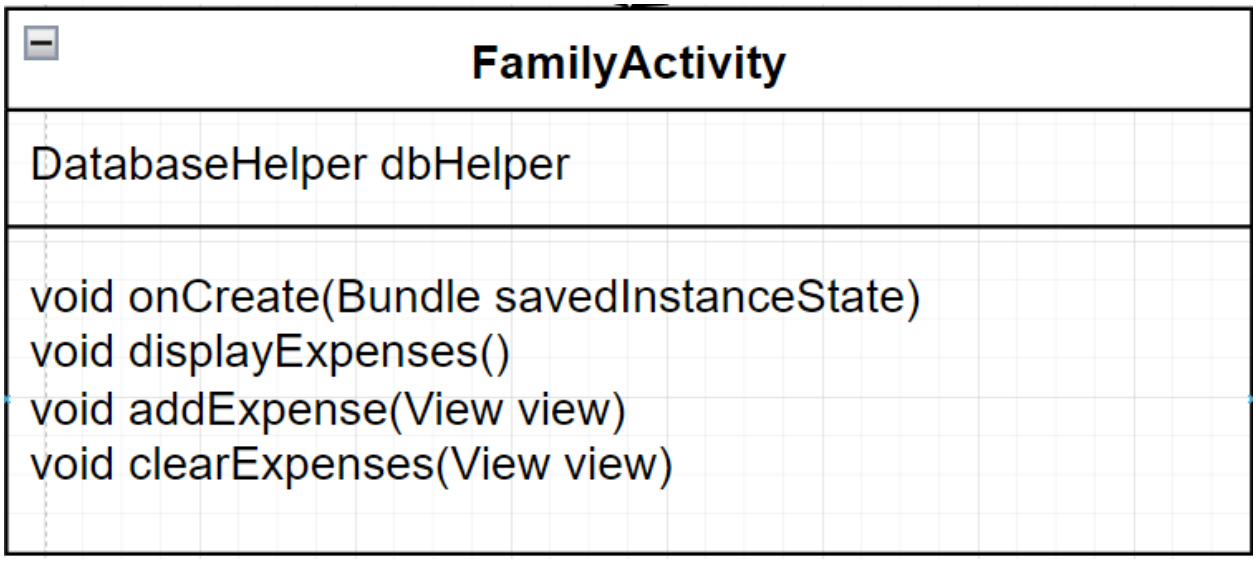


Рисунок 3.2.7.5 – Вікно FamilyActivity

Клас AnotherActivity відповідає за додавання витрат які пов’язані із сім’єю, а також цей клас виводить табличку з доданими витрати, також клас відповідає за очищення таблиці введених витрат.

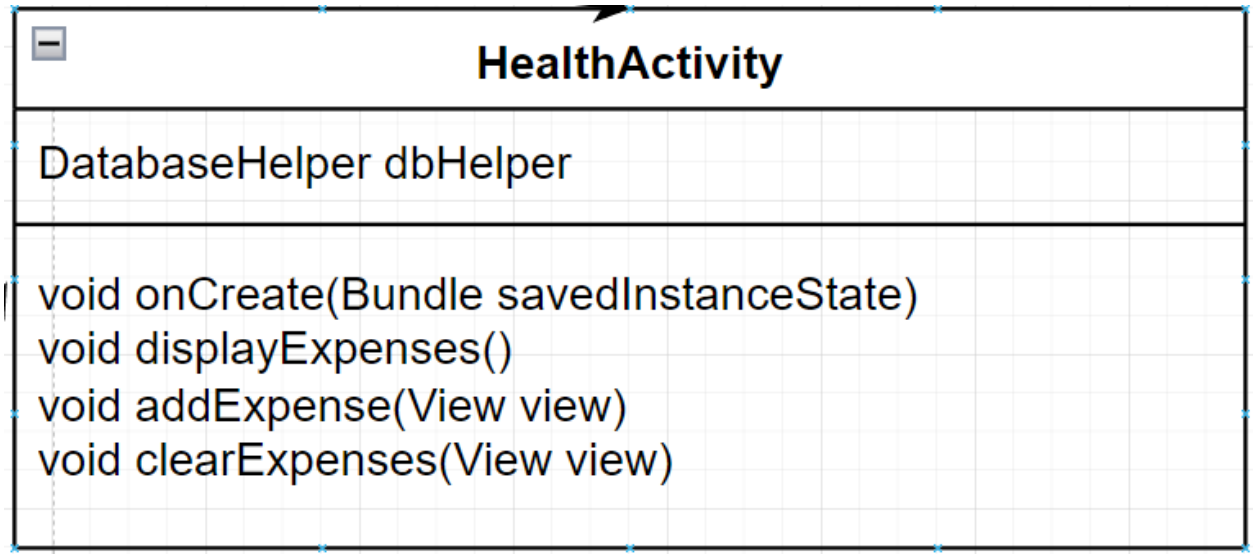


Рисунок 3.2.7.6 – Вікно HealthActivity

Клас HealthActivity відповідає за додавання витрат пов’язані зі здоров’ям, а також цей клас виводить табличку з доданими витрати,також клас відповідає за очищення таблиці введених витрат.

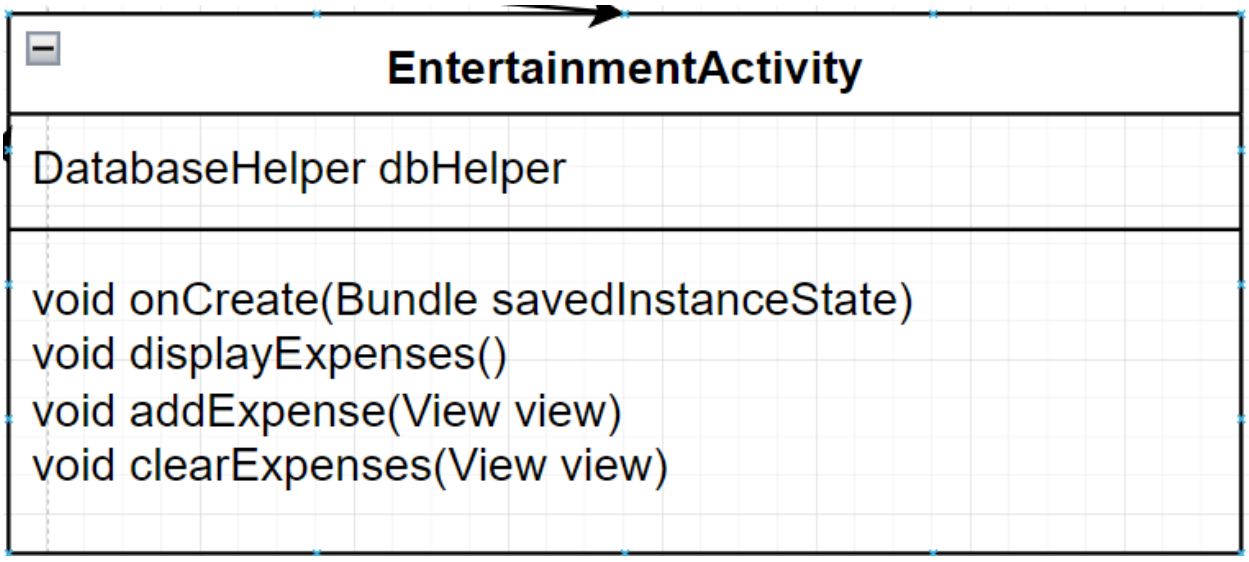


Рисунок 3.2.7.7 – Вікно EntertainmentActivity

Клас EntertainmentActivity відповідає за додавання витрат які пов’язані із розвагами,а також цей клас виводить табличку з доданими витрати,також клас відповідає за очищення таблиці введених витрат.

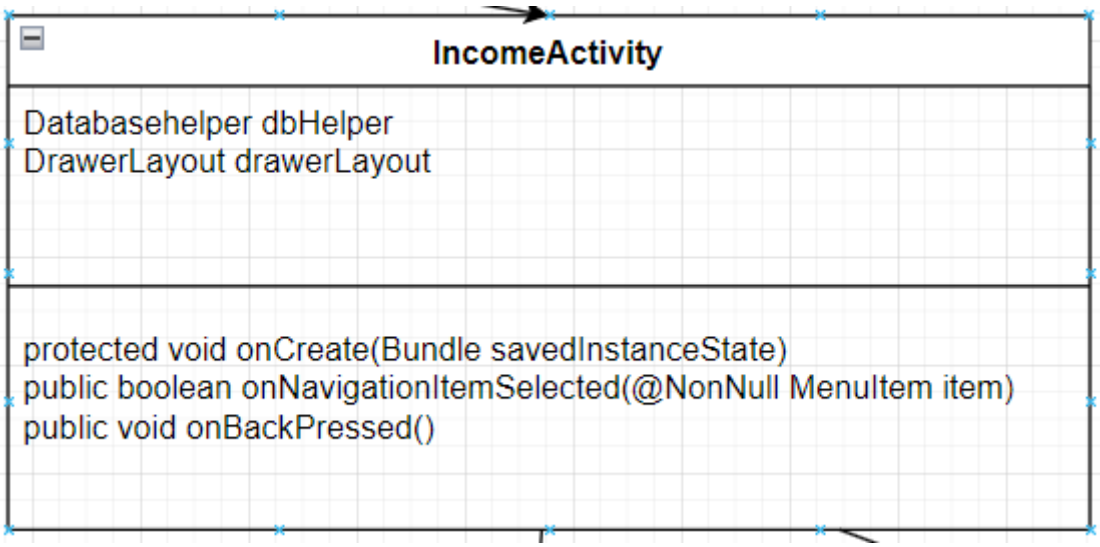


Рисунок 3.2.7.8 – Вікно IncomeActivity

Клас IncomeActivity відповідає за показ балансу користувача також за допомогою нього можна перейти до інших вікон які пов'язані із доходами користувача.

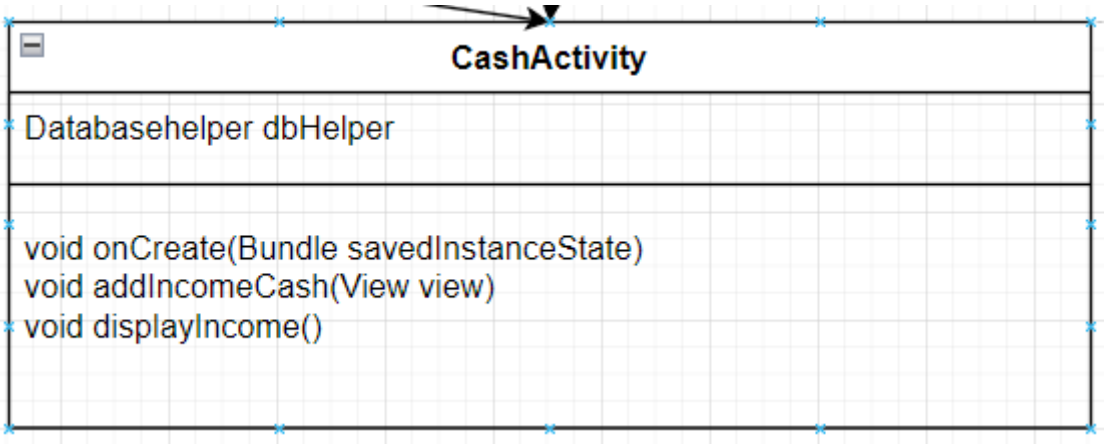


Рисунок 3.2.7.9 – Вікно CashActivity

Вікно CashActivity відповідає за додавання готівки у баланс також дане вікно виводить табличку яка відображає уже додані гроші перед цим(якщо така дія була виконана)

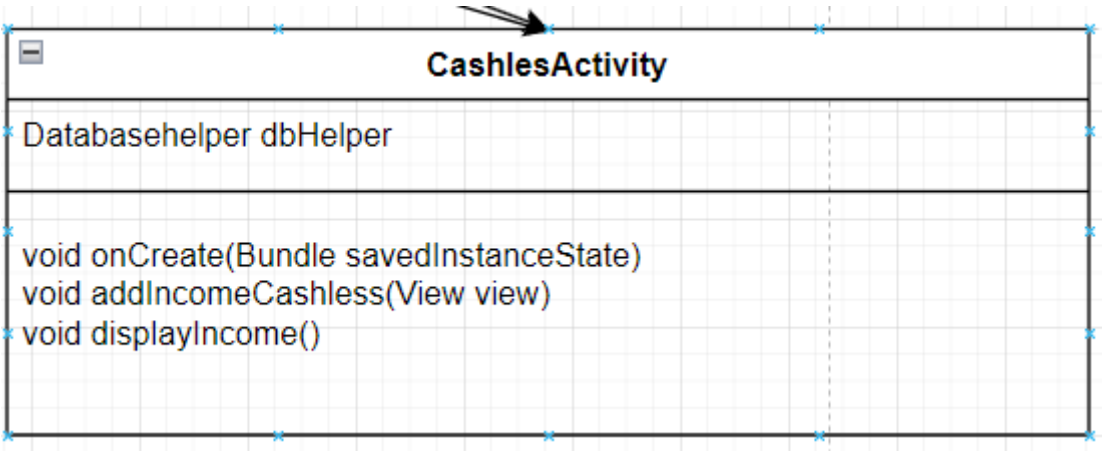


Рисунок 3.2.7.10 – Вікно CashlesActivity

Вікно CashlesActivity відповідає за додавання безготінкових грошей у баланс також дане вікно виводить табличку яка відображає уже додані гроші перед цим(якщо така дія була виконана).

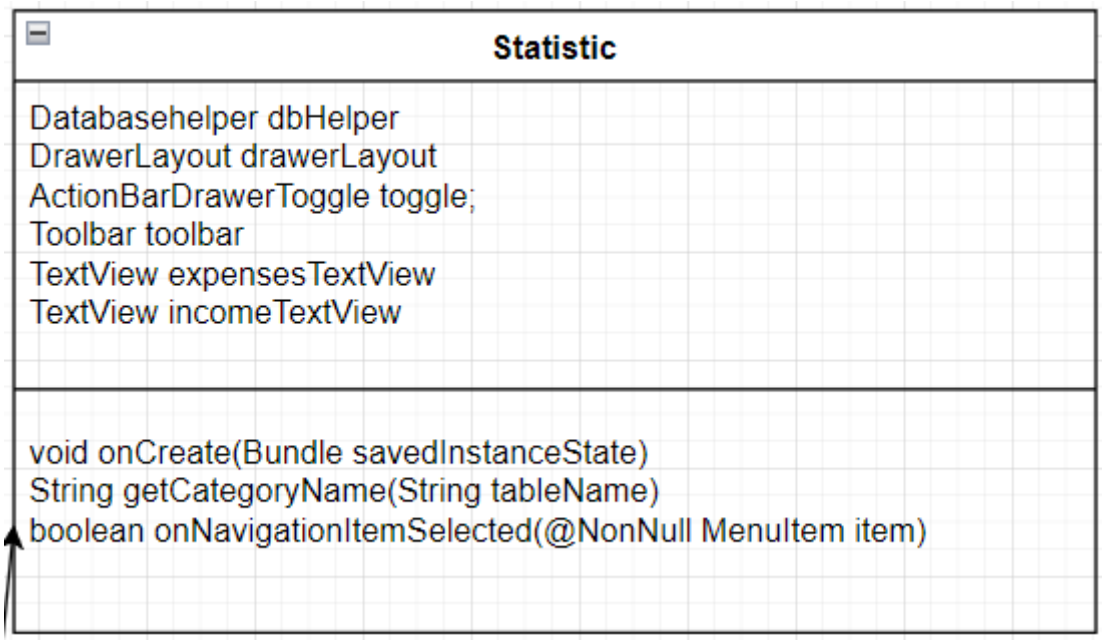


Рисунок 3.2.7.11 – Вікно Statistic

Клас Statistic відповідає за вивод статистики а саме суму витрат за місяць, доходів за місяць і найбільші витрати за місяць.

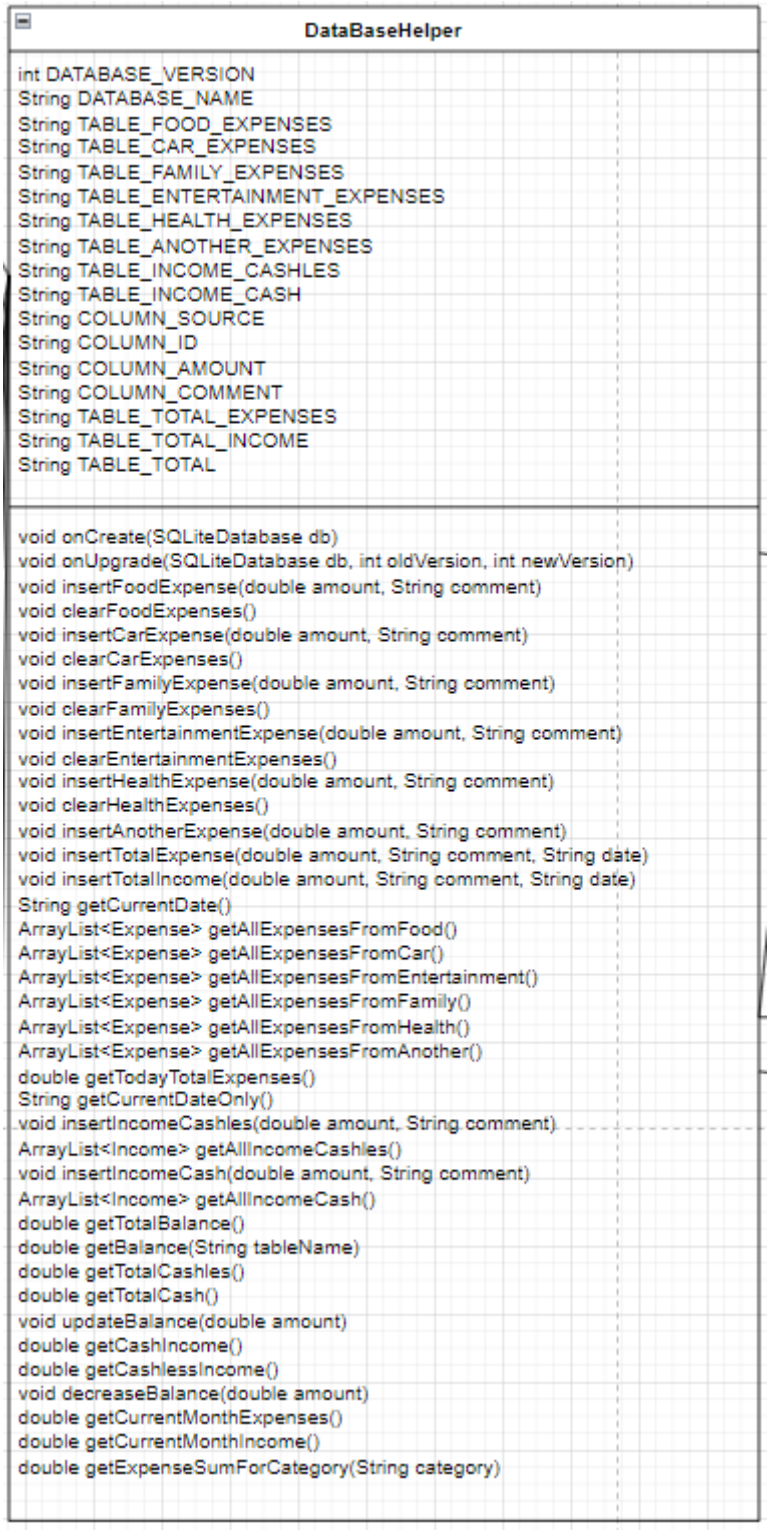


Рисунок 3.2.7.12 – Клас DataBaseHelper

Клас DataBaseHelper віповідає за створення таблиць які будуть зберігати введені витрати або ж дохід користувача,а також клас виконує функцію виконання команд які допоможуть мені взаємодіяти з таблицями або між таблицями.

3.2.8 Структура бази даних

Схема «сутність-зв’язок» (ERD), також відома як ER-діаграма, є важливим інструментом для проектування та моделювання баз даних. Вона дозволяє візуально представити сутності, атрибути та взаємозв'язки між ними.[4].

Основними компонентами ER-діаграми є:

- **Сутність (Entity)** - представляє об'єкт або концепцію з реального світу, про яку зберігається інформація.
- **Атрибут (Attribute)** - це властивість або характеристика сутності.
- **Зв'язок (Relationship)** - Зв'язок показує, як сутності взаємодіють між собою.

Типи зв'язків:

- **Один-до-одного (One-to-One)** - кожна сутність A пов'язана з однією сутністю B, і навпаки.
- **Один-до-багатьох (One-to-Many)** - одна сутність A може бути пов'язана з багатьма сутностями B, але сутність B пов'язана лише з однією сутністю A.
- **Багато-до-багатьох (Many-to-Many)** - кожна сутність A може бути пов'язана з багатьма сутностями B, і кожна сутність B може бути пов'язана з багатьма сутностями A.

Для мого додатку ER-діаграма має даний вигляд:

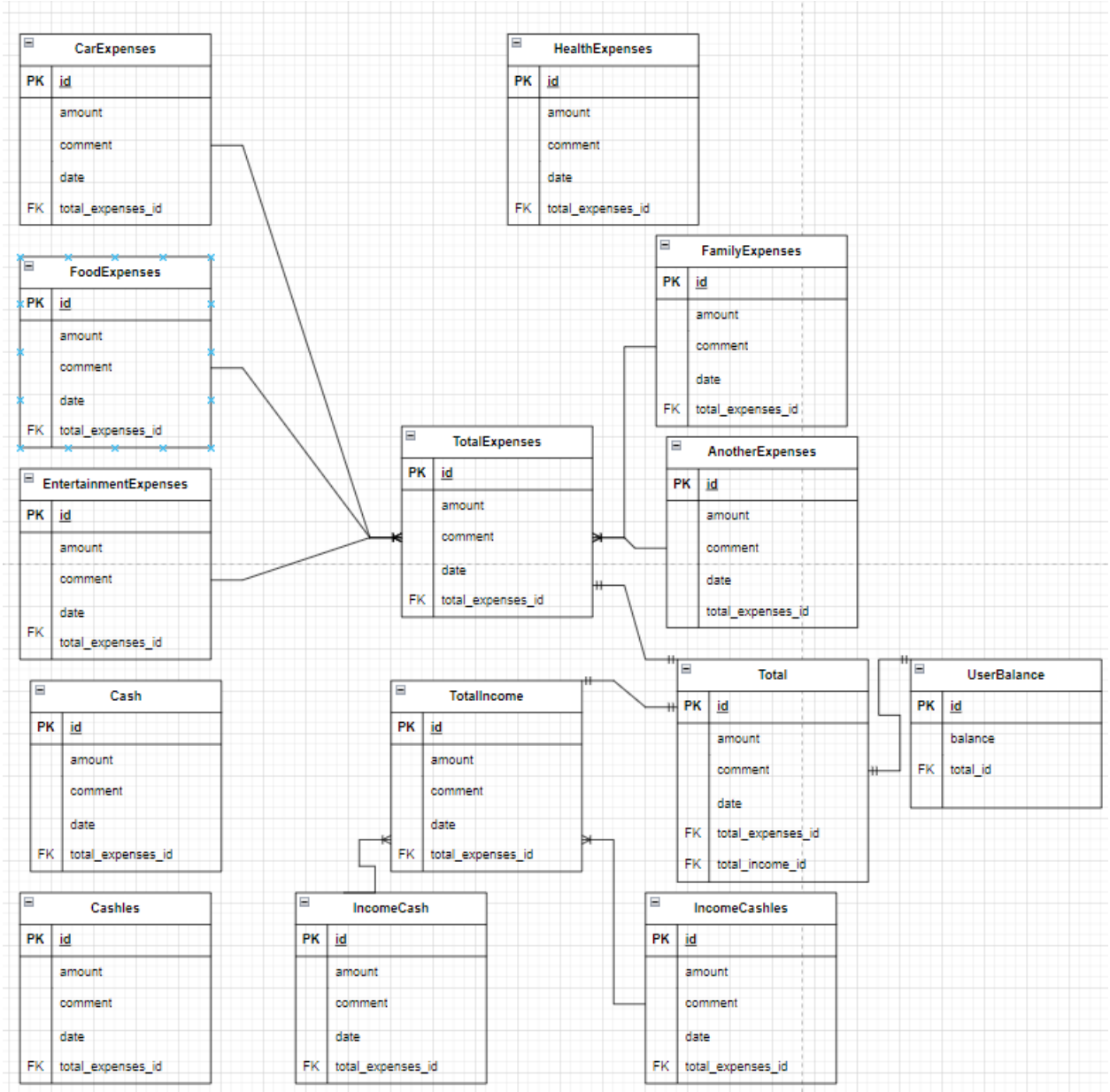


Рисунок 3.2.8.1 – ER-Діаграма

4. Практична реалізація

4.1 Опис програмної реалізації

При старті програми користувач потрапляє на основну активність(MainActivity),в цій активності користувача зустрічає меню в якому він може обрати тип витрати яку він хоче додати. Також на цій активності користувач може побачити чому дорівнюють його витрати на сьогодні. Також користувач в цій активності користувач може побачити ТулБар(ToolBar) який знаходиться зверху додатку який дозволяє користувачеві не заблудитися в додатку описуючи на якій активності він знаходиться в нашому випадку це активність додавання витрат(Рисунок 4.1.1).

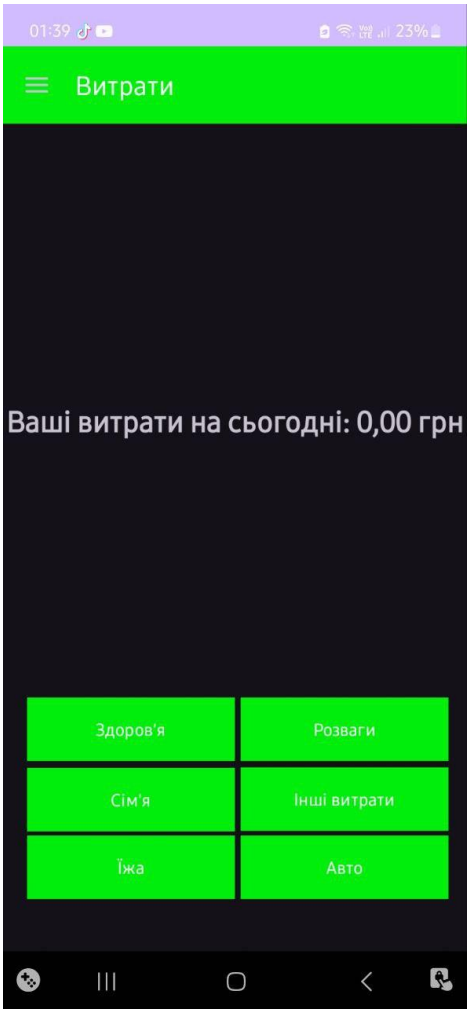


Рисунок 4.1.1 – Вигляд MainActivity.

Для переходу між активностями я вирішив додати NavigationMenu в цьому меню знаходяться три активності це Витрати (MainActivity), Дохід(IncomeActivity) і Статистика(Statistic).(Рисунок 4.1.2)

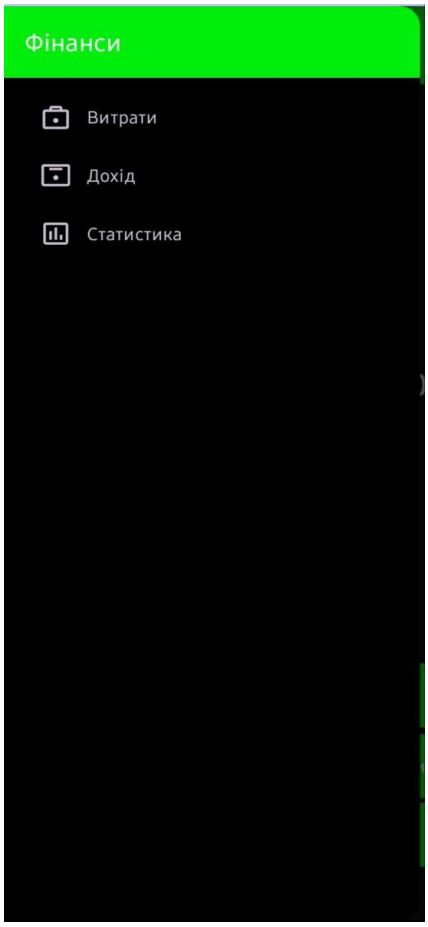


Рисунок 4.1.2 - Вигляд NavigationMenu

Перейшовши через навігацейне меню в вікно дохід(IncomeActivity) ми маємо вивод балансу користувача також вивод грошей у готівкі і на карткі.А також дві кнопки за допомогою яких ми можемо перейти до активності яка буде додавати гроші в баланс.(Рисунок 4.1.3)

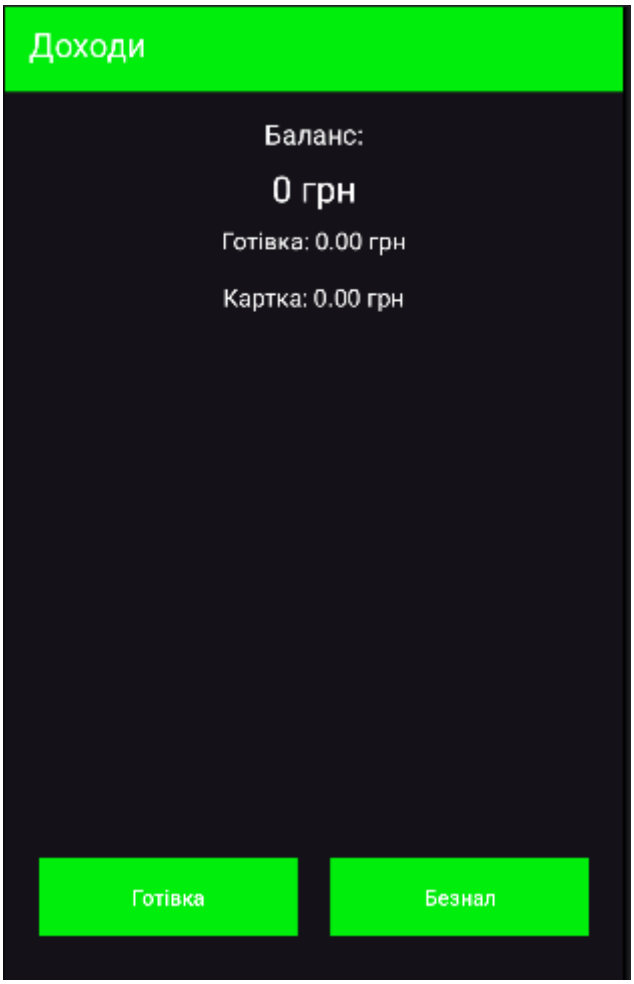


Рисунок 4.1.3 – Вигляд IncomeActivity

Нажавши кнопку статистика ми потрапляємо в вікно Стастистика тут нам буде показано суму всіх витрат за місяць,місячний дохід також строку яка буде вказувати який тип витрат найбільший.

Розглянемо основні компоненти додатку, їх функції та опис:

DataBaseHelper:

setMainActivity(MainActivity activity):

Зберігає посилання на основну активність **MainActivity.setIncomeActivity(IncomeActivity Iactivity):**

Зберігає посилання на активність **IncomeActivity.DatabaseHelper(Context context):**

Конструктор, який ініціалізує базу даних.

getTodayTotalExpensesLiveData():

Повертає LiveData з поточними загальними витратами за сьогодні.

onCreate(SQLiteDatabase db):

Створює таблиці бази даних, якщо вони ще не існують.Перевіряє наявність запису в таблиці user_balance і додає початковий баланс 1000, якщо запис відсутній.

onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion):

Видаляє існуючі таблиці та створює їх знову при оновленні версії бази даних.

insertFoodExpense(double amount, String comment):

Вставляє новий запис у таблицю food_expenses.Додає запис до загальної таблиці витрат.

clearFoodExpenses():Видаляє всі записи з таблиці food_expenses.Оновлює загальні витрати за сьогодні.

insertCarExpense(double amount, String comment):

Вставляє новий запис у таблицю car_expenses.Додає запис до загальної таблиці витрат.

clearCarExpenses():

Видаляє всі записи з таблиці car_expenses.Оновлює загальні витрати за сьогодні.

insertFamilyExpense(double amount, String comment):

Вставляє новий запис у таблицю family_expenses.Додає запис до загальної таблиці витрат.

clearFamilyExpenses():

Видаляє всі записи з таблиці family_expenses.Оновлює загальні витрати за сьогодні.

insertEntertainmentExpense(double amount, String comment):

Вставляє новий запис у таблицю entertainment_expenses.Додає запис до загальної таблиці витрат.

clearEntertainmentExpenses():

Видаляє всі записи з таблиці entertainment_expenses.Оновлює загальні витрати за сьогодні.

insertHealthExpense(double amount, String comment):

Вставляє новий запис у таблицю health_expenses.Додає запис до загальної таблиці витрат.

clearHealthExpenses():

Видаляє всі записи з таблиці health_expenses.Оновлює загальні витрати за сьогодні.

insertAnotherExpense(double amount, String comment):

Вставляє новий запис у таблицю another_expenses.Додає запис до загальної таблиці витрат.

insertTotalExpense(double amount, String comment, String date):

Вставляє новий запис у таблицю total_expenses.

insertTotalIncome(double amount, String comment, String date):

Вставляє новий запис у таблицю total_income.

getCurrentDate():

Повертає поточну дату та час у форматі yyyy-MM-dd HH:mm:ss.

getAllExpensesFromFood():

Повертає список усіх витрат з таблиці food_expenses.

getAllExpensesFromCar():

Повертає список усіх витрат з таблиці car_expenses.

getAllExpensesFromEntertainment():

Повертає список усіх витрат з таблиці entertainment_expenses.

getAllExpensesFromFamily():

Повертає список усіх витрат з таблиці family_expenses.

getAllExpensesFromHealth():

Повертає список усіх витрат з таблиці health_expenses.

getAllExpensesFromAnother():

Повертає список усіх витрат з таблиці **another_expenses.getTodayTotalExpenses():**

Повертає загальну суму витрат за сьогодні.

getCurrentDateOnly():

Повертає поточну дату у форматі yyyy-MM-dd.

insertIncomeCashles(double amount, String comment):

Вставляє новий запис у таблицю cashles.Додає запис до загальної таблиці доходів.Оновлює баланс користувача.

getAllIncomeCashles():

Повертає список усіх доходів з таблиці cashles.

insertIncomeCash(double amount, String comment):

Вставляє новий запис у таблицю cash.Додає запис до загальної таблиці доходів.Оновлює баланс користувача.

getAllIncomeCash():

Повертає список усіх доходів з таблиці cash.

getTotalBalance():

Повертає загальний баланс (сума доходів з таблиць cashles і cash).

getBalance(String tableName):

Повертає баланс для вказаної таблиці доходів.

getTotalCashles():

Повертає загальну суму доходів з таблиці cashles.

getTotalCash():

Повертає загальну суму доходів з таблиці cash.

updateBalance(double amount):

Оновлює баланс користувача, додаючи або віднімаючи вказану суму.

getCashIncome():

Повертає загальну суму доходів з таблиці cash.

getCashlessIncome():

Повертає загальну суму доходів з таблиці cashles.

decreaseBalance(double amount):

Зменшує баланс користувача на вказану суму і додає запис до таблиці total.

getCurrentMonthExpenses():

Повертає загальну суму витрат за поточний місяць.

getCurrentMonthIncome():

Повертає загальну суму доходів за поточний місяць.

getExpenseSumForCategory(String category):

Повертає загальну суму витрат для вказаної категорії за поточний місяць.

MainActivity:

onCreate(Bundle savedInstanceState):

Цей метод викликається, коли активність створюється. В ньому відбувається ініціалізація елементів інтерфейсу користувача, налаштування слухачів подій та інше.

onNavigationItemSelected(MenuItem item):

Цей метод викликається, коли елемент меню вибирається користувачем. Він обробляє події вибору пунктів меню та запускає відповідні активності за допомогою інтентів.

updateTodayTotalExpenses():

Цей метод оновлює відображення загальних витрат за сьогоднішній день, використовуючи дані з бази даних.

onBackPressed():

Цей метод викликається, коли користувач натискає кнопку "Назад" на пристрої. Він перевіряє, чи відкрите бічне меню, і закриває його, якщо так, або викликає стандартну реакцію в іншому випадку.

IncomeActivity:

onCreate(Bundle savedInstanceState):

Цей метод викликається при створенні активності. Він встановлює макет сторінки, налаштовує панель інструментів, налаштовує елементи інтерфейсу користувача та слухачі подій.

onNavigationItemSelected(MenuItem item):

Цей метод обробляє події вибору пунктів меню навігації. Він відкриває відповідні активності при виборі пунктів меню.

updateBalance():

Цей метод оновлює відображення загального балансу, використовуючи дані з бази даних.

Також в коді є кнопки для переходу на активності CashActivity та CashlesActivity при кліку на них, а також виклик методу setSupportActionBar() для встановлення панелі інструментів в якості панелі дій.

AnotherActivity,AutoActivity,FoodActivity,FamilyActivity,HealthActivity,EntertainmentActivity,:

onCreate(Bundle savedInstanceState):

Цей метод викликається при створенні активності. Він встановлює макет сторінки та викликає метод displayExpenses() для відображення списку витрат.

displayExpenses():

Цей метод відображає список витрат у таблиці. Він очищає таблицю, додає заголовок з назвами полів ("Сума", "Коментар", "Дата") та додає рядки з даними про кожну витрату.

addExpense(View view):

Цей метод викликається при натисканні кнопки "Додати витрату". Він отримує значення витрати та коментаря з текстових полів введення, додає новий запис у базу даних та оновлює список витрат.

clearExpenses(View view):

Цей метод викликається при натисканні кнопки "Очистити витрати". Він видаляє всі записи про витрати з бази даних та оновлює список витрат.

CashActivity,CashlesActivity:

onCreate(Bundle savedInstanceState):

Цей метод викликається при створенні активності. Він встановлює макет сторінки та викликає метод displayIncome() для відображення списку доходів.

displayIncome():

Цей метод відображає список доходів у таблиці. Він очищає таблицю, додає заголовок з назвами полів ("Сума", "Коментар", "Дата") та додає рядки з даними про кожний дохід.

addIncomeCash(View view):

Цей метод викликається при натисканні кнопки "Додати дохід". Він отримує значення доходу та коментаря з текстових полів введення, додає новий запис у базу даних та оновлює список доходів.

Statistic:

onCreate(Bundle savedInstanceState):

Цей метод налаштовує користувацький інтерфейс, включаючи навігаційне меню, ініціалізує базу даних та відображає поточні витрати і доходи користувача за поточний місяць, а також визначає категорію з найбільшими витратами і відображає її.

private String getCategoryName(String tableName)

Метод використовується для перетворення назви таблиці з бази даних на зрозумілу користувачеві назву категорії витрат. Це дозволяє відображати користувачеві більш читабельні дані про його витрати у різних категоріях.

public boolean onNavigationItemSelected(@NonNull MenuItem item)

Цей метод обробляє події вибору пунктів меню навігації. Він відкриває відповідні активності при виборі пунктів меню.

4.2 Тестування

Для того щоб провести тестування я вирішив обрати метод мануального тестування він дозволить мені детально перевірити роботу усіх функцій і елементів інтерфейсу користувача. Нижче я навів опис процесу мануального тестування. Тестування проводилося на мобільному телефоні марки Samsung Galaxy S21 FE(Версія Android 14).

Підготовка до тестування:

- **Встановлення додатку:** Я завантажив додаток на свій телефон і успішного його запустив.
- **Налаштування серидовища:** Після встановлення я переконався що мій телефон справно працює і це ніяк не вплине на роботу додатку.

Тестування основних функцій.

1.Додавання витрат у наші таблички.

Мета: Перевірити функціонал додавання витрат у табличку витрат,а також перевірка виводу сьогоднішніх витрат.

Дії: Зайти в додаток, натиснути на кнопку Їжа, перейти в активність де додаються витрати на їжу, після цього подивитися чи з’явилася витрата в табличкі, потім вийти на вікно з витратами і переглянути чи оновився лічильник сьогоднішніх витрат.

Результат: Все вірно запрацювало. Всі витрати були успішно додані і виведені в табличці витрат, також лічільник сьогоднішніх витрат також оновився.(Рисунок 5.1-5.1.2)

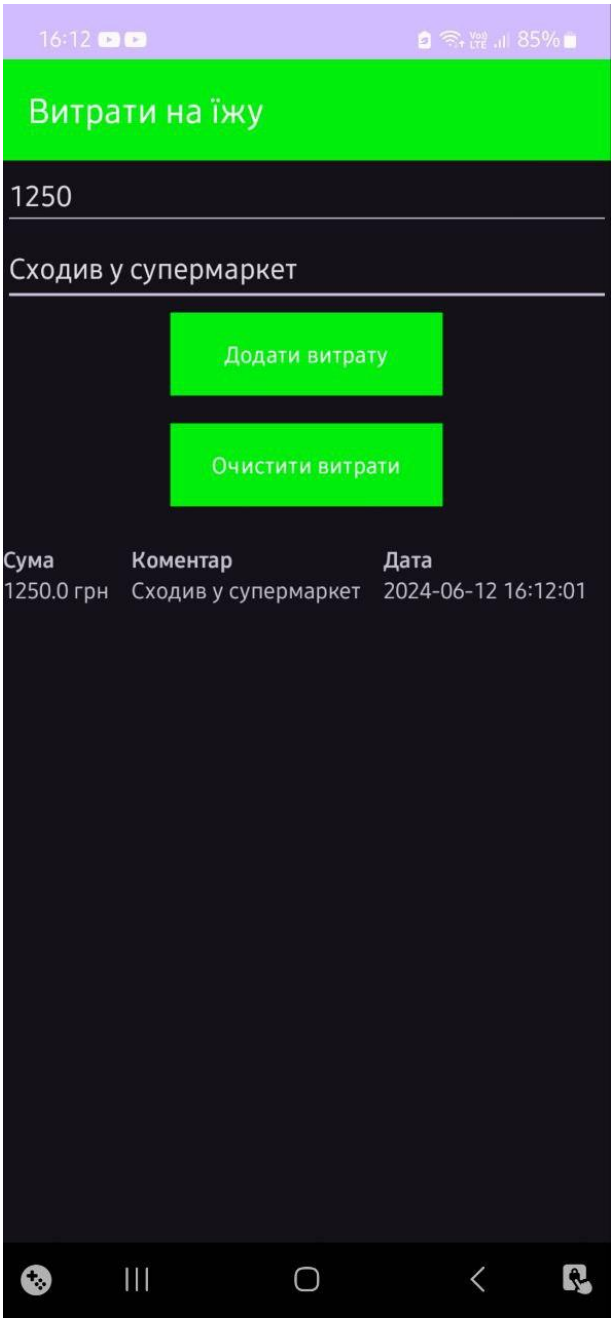


Рисунок 5.1 – Результати тестування додатку №1

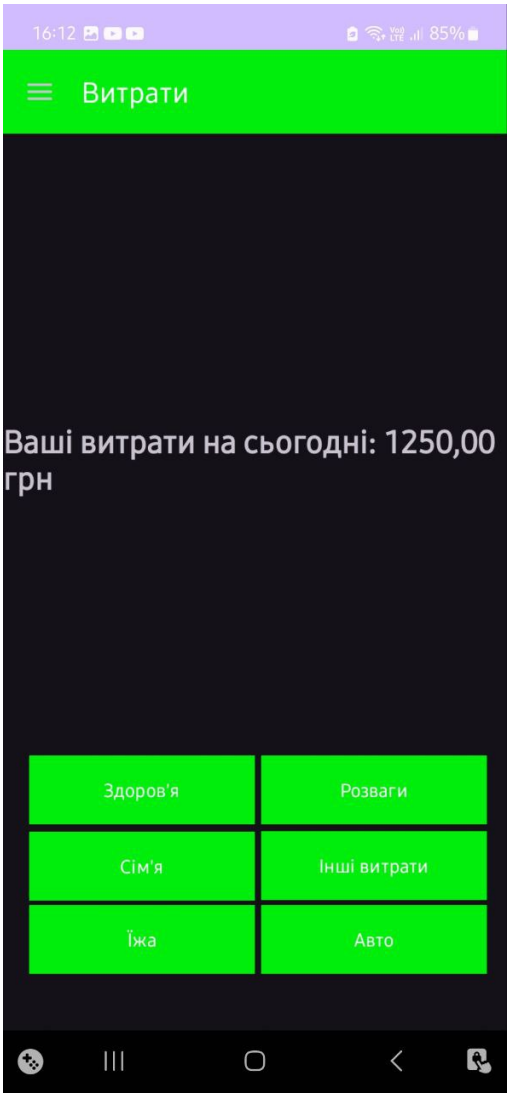


Рисунок 5.1.2 – Результати тестування додатку №1

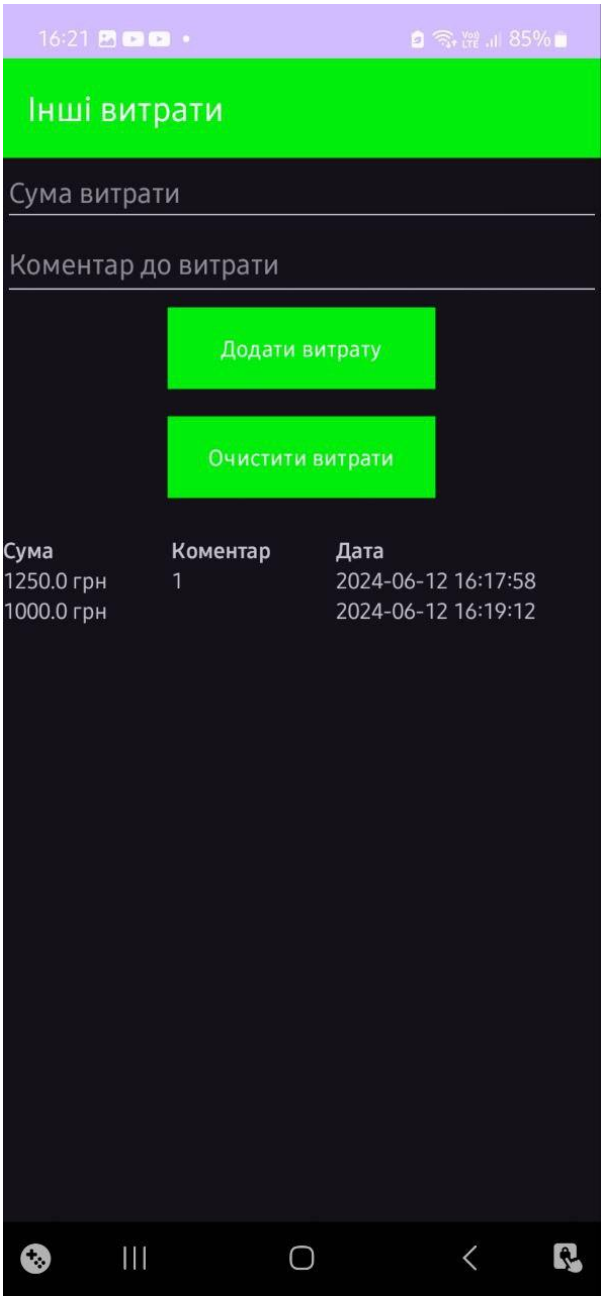


Рисунок 5.1.3 – Результати тестування додатку №1

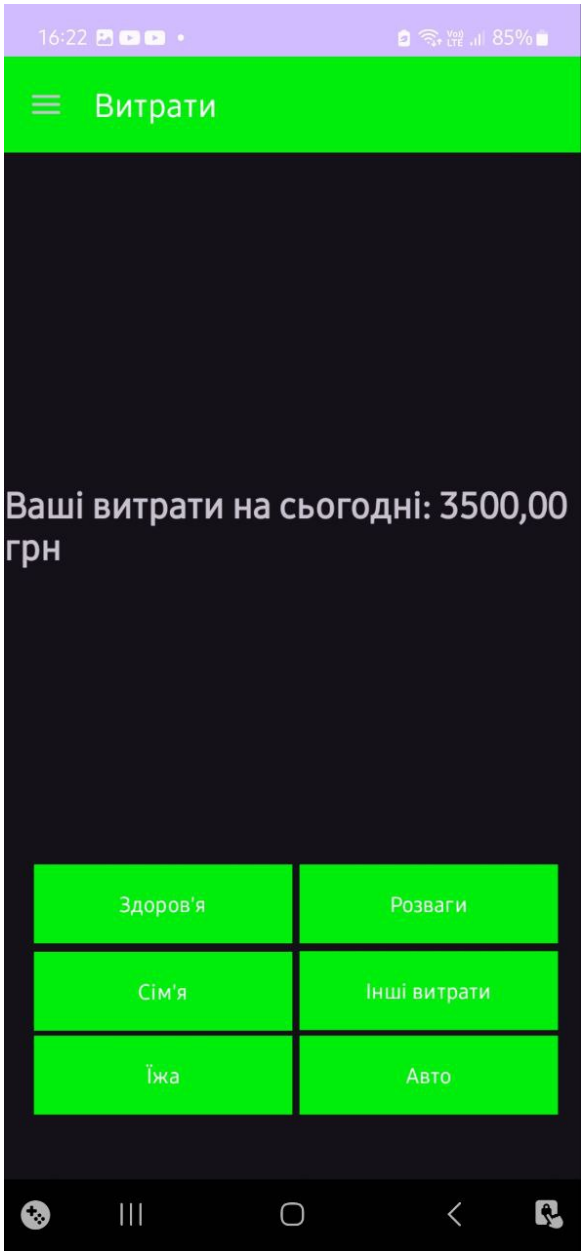


Рисунок 5.1.4 – Результати тестування додатку №1

2.Перевірка роботоспособності NavigationMenu.

Мета: Перевірити чи корректно працює навігаційне меню яке допомагає користувачу переходити між вікнами.

Дії: Спочатку перейти із витрат в статистику,потім із статистики перейти в Дохід.

Результат : Тестування пройшло успішно.(Рисунок 5.2.1-5.2.4)

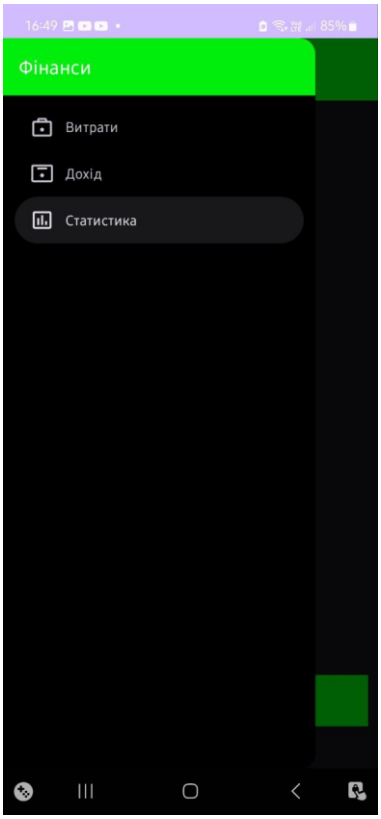


Рисунок 5.2.1 – Результат тестування №2

Рисунок 5.2.2 – Результат тестування №2

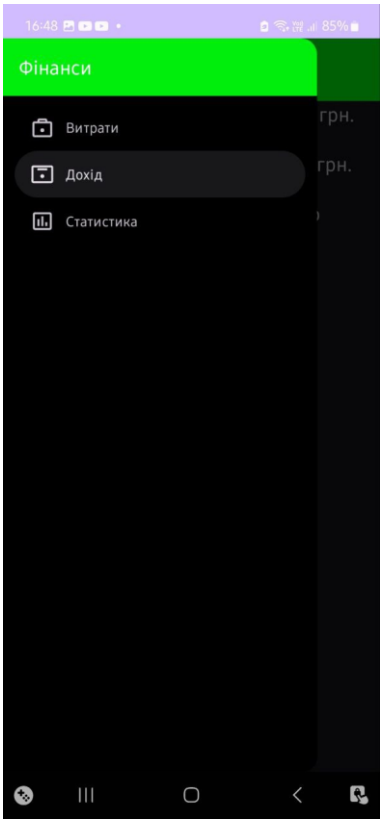


Рисунок 5.2.3 – Результат тестування №2

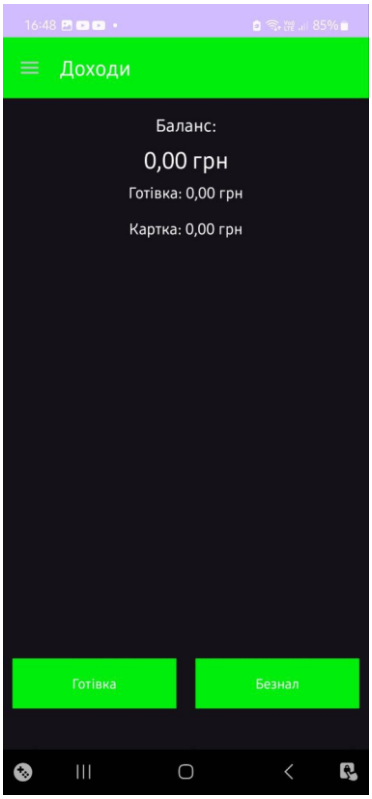


Рисунок 5.2.4 – Результат тестування №2

3.Додавання доходів в табличку і перевірка балансу.

Мета: Додати свої доходи у дві таблички(Готівка і Безнал) і перевірити корректне виведення коштів у табличкі а також у IncomeActivity.

Дії: Зайти в IncomeActivity за допомогою NavigationMenu,спочатку зайти у вікно з додаванням готівки, а потім зайти у вікно з безналом, вийти із останнього вікна і перевірити чи змінився баланс.

Результати: Все вірно запрацювало гроші успішно були додані у таблички і також успішно були виведені у на головному вікні яке відповідає за гроші користувача.(Рисунок 5.3.1-5.3.4).

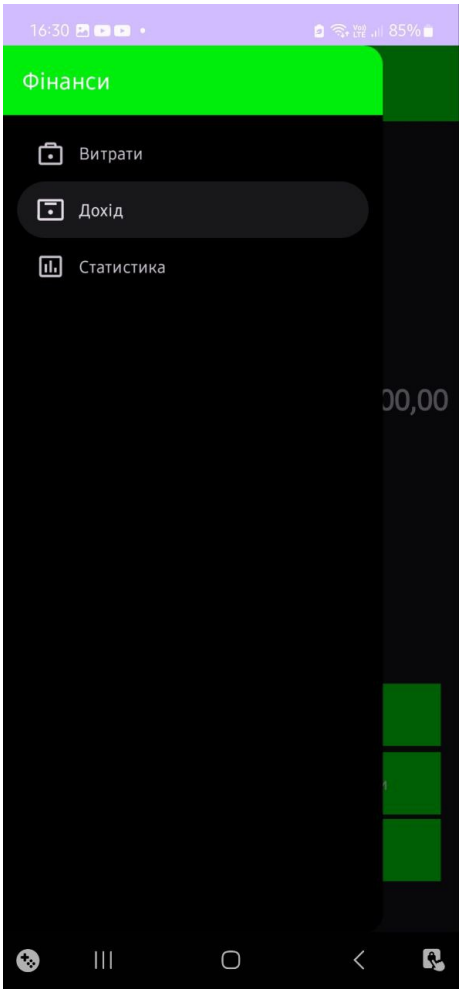


Рисунок 5.3.1 – Результати тестування додатку №3

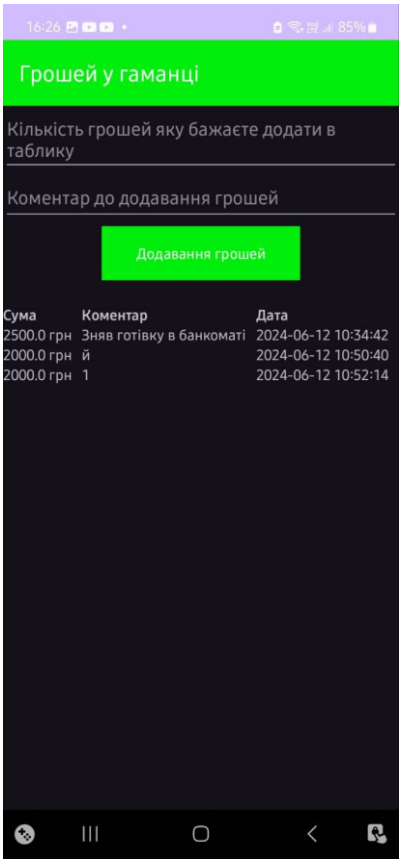


Рисунок 5.3.2 – Результати тестування додатку №3

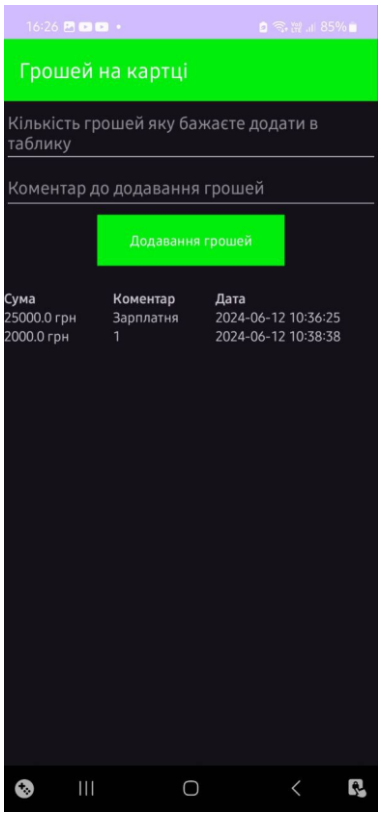


Рисунок 5.3.3 – Результати тестування додатку №3

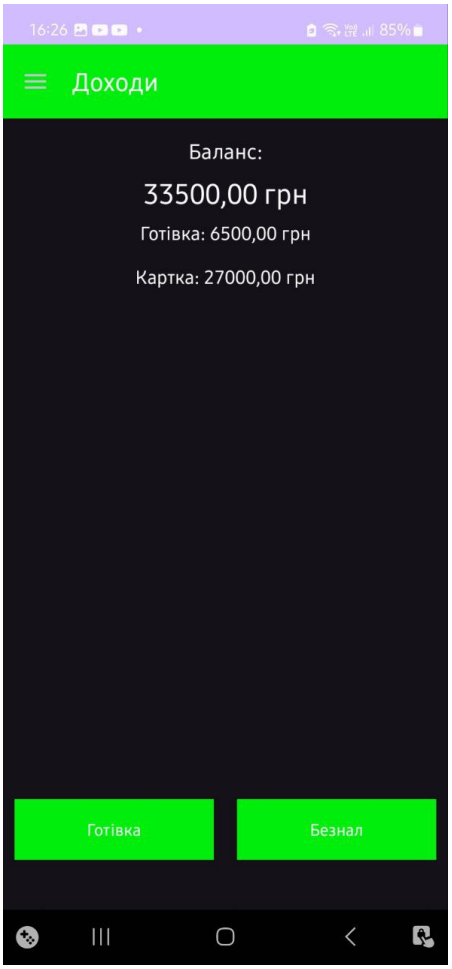


Рисунок 5.3.4 – Результати тестування додатку №3

Висновок

При розробці мобільного додатку для розрахунку витрат були проаналізовані потреби користувачів, визначені цільові користувачі та сформульовані завдання для розробки додатку. Вибравши архітектуру Model-View-Controller (MVC), я зміг створити структурований та ефективний додаток, розділивши логіку, представлення та управління даними.

Додаток реалізує такі функції, як додавання витрат, класифікація витрат і аналіз витрат; за допомогою різних технологій, таких як Java, Android SDK, SQLite і XML, ці функції були ефективно реалізовані для створення зручного і функціонального додатку проект був успішним.

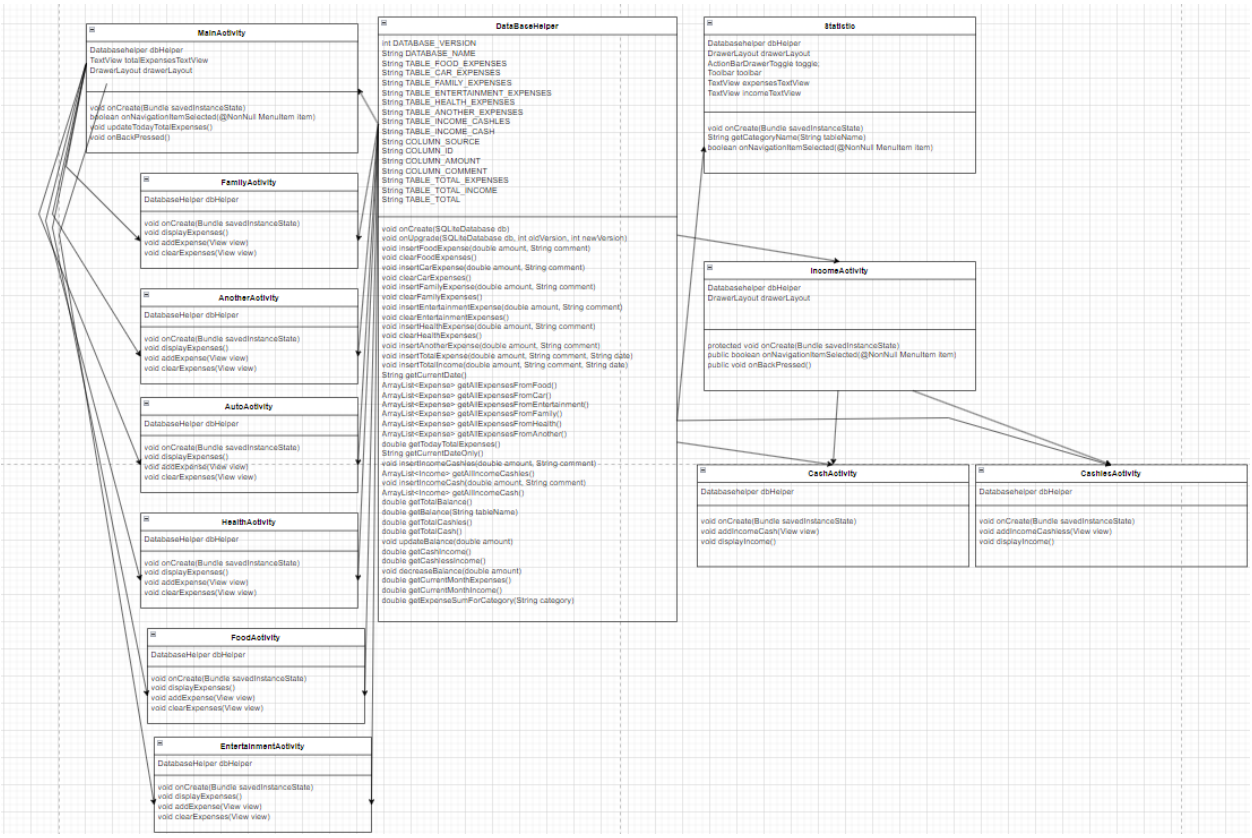
Основне середовище розробки, Android Studio, забезпечило зручний інтерфейс для розробки, налагодження та тестування додатків і дозволило ефективно використовувати всі можливості платформи Android.

Результатом дослідження та розробки став функціональний та корисний мобільний додаток для фінансового менеджменту та калькуляції витрат для широкого кола користувачів.

Список літератури

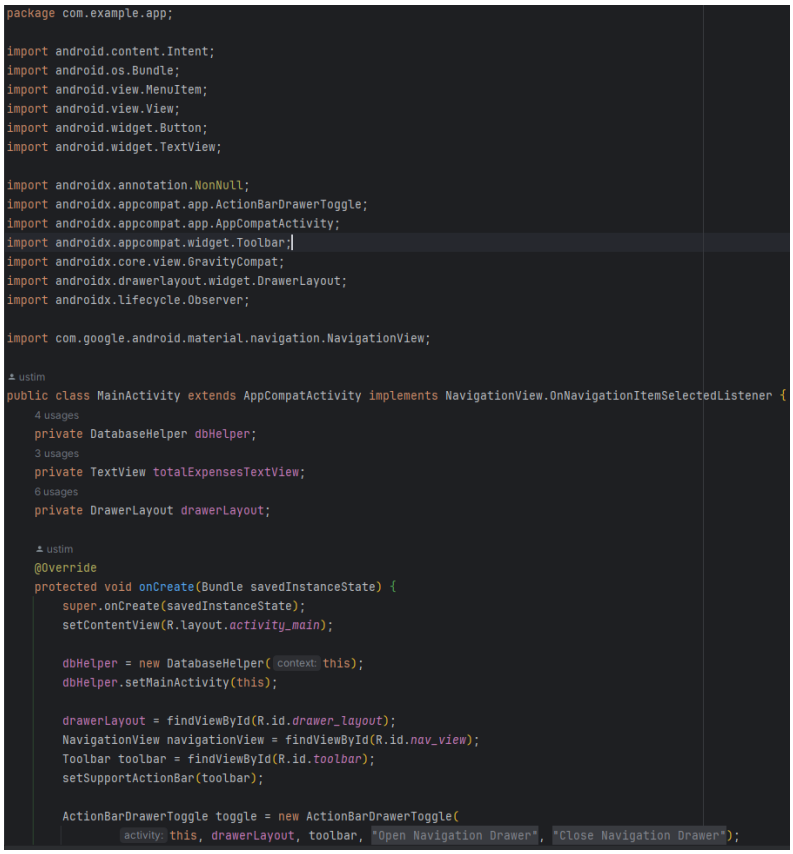
1. Архітектура мобільного застосунку: що це таке і як працює – Wezom. URL: https://wezom.com.ua/ua/blog/arhitektura-mobilnogo-prilozheniya (дата звернення: 27.04.2024).
2. Noodles P. H. Ознайомлення з патерном MVC (Model-View-Controller). *JavaRush*. URL: https://javarush.com/ua/groups/posts/uk.2536.chastina-7-oznayomlennja-z-paternom-mvc-model-view-controller (дата звернення: 30.04.2024).
3. Viacheslav. Коротке знайомство з Gradle. *JavaRush*. URL: https://javarush.com/ua/groups/posts/uk.2126.korotke-znayomstvo-z-gradle (дата звернення: 26.04.2024).
4. Lucidchart. What is an Entity Relationship Diagram (ERD)? URL: https://www.lucidchart.com/pages/er-diagrams (дата звернення: 19.05.2024).
5. Економічна правда. Мобільні застосунки для планування бюджету та контролю витрат: які найбільш популярні. *Економічна правда*. URL: https://www.epravda.com.ua/publications/2023/02/22/697326/ (дата звернення: 08.05.2024).
6. Моралес Дж. Entity Relationship Diagram: What, How, and When to Make One. *MindOnMap | Free Mind Mapping Tool to Draw Ideas Easily Online*. URL: https://www.mindonmap.com/uk/blog/relationship-diagram/ (date of access: 23.05.2024).
7. 10 зручних додатків для контролю особистого бюджету. *Мінфін - все про фінанси: новини, курси валют, банки*. URL: https://minfin.com.ua/ua/2019/07/24/38514719/ (дата звернення: 08.05.2024).
8. Learn Android Tutorial | Android Studio Tutorial - Javatpoint. *www.javatpoint.com*. URL: https://www.javatpoint.com/android-tutorial (date of access: 08.04.2024).
9. Smyth N. Android Studio Electric Eel Essentials - Java Edition: Developing Android Apps Using Android Studio 2022.1.1 and Java. Payload Media, 2023.
10. SQLite Home Page. *SQLite Home Page*. URL: https://sqlite.org (date of access: 15.04.2024).

Додаток А Діаграма класів

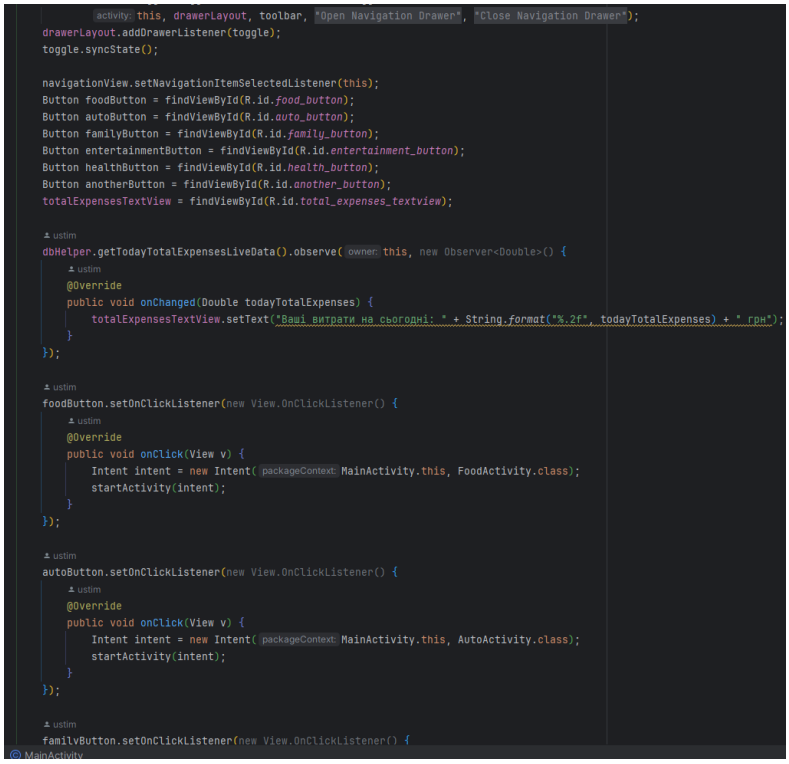


Додаток Б – Код із всих класів

Код класу MainActivity:

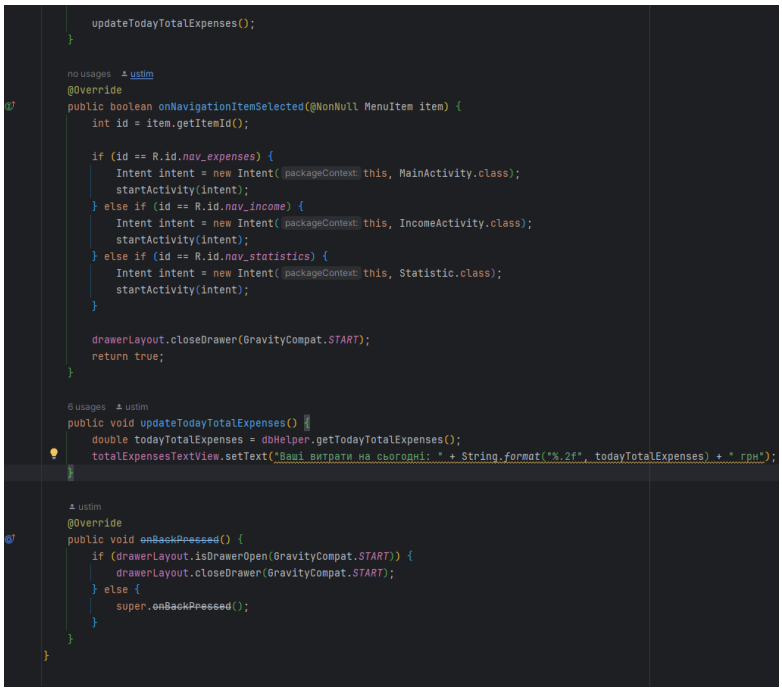


```
package com.example.app;

import android.content.Intent;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.lifecycle.Observer;

import com.google.android.material.navigation.NavigationView;

ustim
public class MainActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener {
    4 usages
    private DatabaseHelper dbHelper;
    3 usages
    private TextView totalExpensesTextView;
    6 usages
    private DrawerLayout drawerLayout;

    ustim
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        dbHelper = new DatabaseHelper( context: this);
        dbHelper.setMainActivity(this);

        drawerLayout = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                activity: this, drawerLayout, toolbar, "Open Navigation Drawer", "Close Navigation Drawer");
```



```
                activity: this, drawerLayout, toolbar, "Open Navigation Drawer", "Close Navigation Drawer");
        drawerLayout.addDrawerListener(toggle);
        toggle.syncState();

        navigationView.setNavigationItemSelectedListener(this);
        Button foodButton = findViewById(R.id.food_button);
        Button autoButton = findViewById(R.id.auto_button);
        Button familyButton = findViewById(R.id.family_button);
        Button entertainmentButton = findViewById(R.id.entertainment_button);
        Button healthButton = findViewById(R.id.health_button);
        Button anotherButton = findViewById(R.id.another_button);
        totalExpensesTextView = findViewById(R.id.total_expenses_textview);

        ustim
        dbHelper.getTodayTotalExpensesLiveData().observe( owner: this, new Observer<Double>() {
            ustim
            @Override
            public void onChanged(Double todayTotalExpenses) {
                totalExpensesTextView.setText("Ваші витрати на сьогодні: " + String.format("%.2f", todayTotalExpenses) + " грн");
            }
        });

        ustim
        foodButton.setOnClickListener(new View.OnClickListener() {
            ustim
            @Override
            public void onClick(View v) {
                Intent intent = new Intent( packageContext: MainActivity.this, FoodActivity.class);
                startActivity(intent);
            }
        });

        ustim
        autoButton.setOnClickListener(new View.OnClickListener() {
            ustim
            @Override
            public void onClick(View v) {
                Intent intent = new Intent( packageContext: MainActivity.this, AutoActivity.class);
                startActivity(intent);
            }
        });

        ustim
        familyButton.setOnClickListener(new View.OnClickListener() {
MainActivity
```

```
        ustim
        familyButton.setOnClickListener(new View.OnClickListener() {
            ustim
            @Override
            public void onClick(View v) {
                Intent intent = new Intent( packageContext: MainActivity.this, FamilyActivity.class);
                startActivity(intent);
            }
        });

        ustim
        entertainmentButton.setOnClickListener(new View.OnClickListener() {
            ustim
            @Override
            public void onClick(View v) {
                Intent intent = new Intent( packageContext: MainActivity.this, EntertainmentActivity.class);
                startActivity(intent);
            }
        });

        ustim
        healthButton.setOnClickListener(new View.OnClickListener() {
            ustim
            @Override
            public void onClick(View v) {
                Intent intent = new Intent( packageContext: MainActivity.this, HealthActivity.class);
                startActivity(intent);
            }
        });

        ustim
        anotherButton.setOnClickListener(new View.OnClickListener() {
            ustim
            @Override
            public void onClick(View v) {
                Intent intent = new Intent( packageContext: MainActivity.this, AnotherActivity.class);
                startActivity(intent);
            }
        });

        updateTodayTotalExpenses();
    }
```



```
        updateTodayTotalExpenses();
    }

    no usages   ustim
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.nav_expenses) {
            Intent intent = new Intent( packageContext: this, MainActivity.class);
            startActivity(intent);
        } else if (id == R.id.nav_income) {
            Intent intent = new Intent( packageContext: this, IncomeActivity.class);
            startActivity(intent);
        } else if (id == R.id.nav_statistics) {
            Intent intent = new Intent( packageContext: this, Statistic.class);
            startActivity(intent);
        }

        drawerLayout.closeDrawer(GravityCompat.START);
        return true;
    }

    6 usages   ustim
    public void updateTodayTotalExpenses() {
        double todayTotalExpenses = dbHelper.getTodayTotalExpenses();
        totalExpensesTextView.setText("Ваші витрати на сьогодні: " + String.format("%.2f", todayTotalExpenses) + " грн");
    }

    ustim
    @Override
    public void onBackPressed() {
        if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
            drawerLayout.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }
}
```

Код класу IncomeActivity:

```java
package com.example.app;

import android.content.Intent;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;

import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.google.android.material.navigation.NavigationView;

import java.util.Locale;

ustim *
public class IncomeActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener {
    6 usages
    private DatabaseHelper dbHelper;
    4 usages
    private DrawerLayout drawerLayout;
    ustim *
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_income);

        dbHelper = new DatabaseHelper( context: this);
        dbHelper.setIncomeActivity(this);
        drawerLayout = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        double cashIncome = dbHelper.getCashIncome();
        double cashlessIncome = dbHelper.getCashlessIncome();
        Button cashButton = findViewById(R.id.cash_button);
        Button cashlessButton = findViewById(R.id.cashless_button);
        setSupportActionBar(toolbar);
```

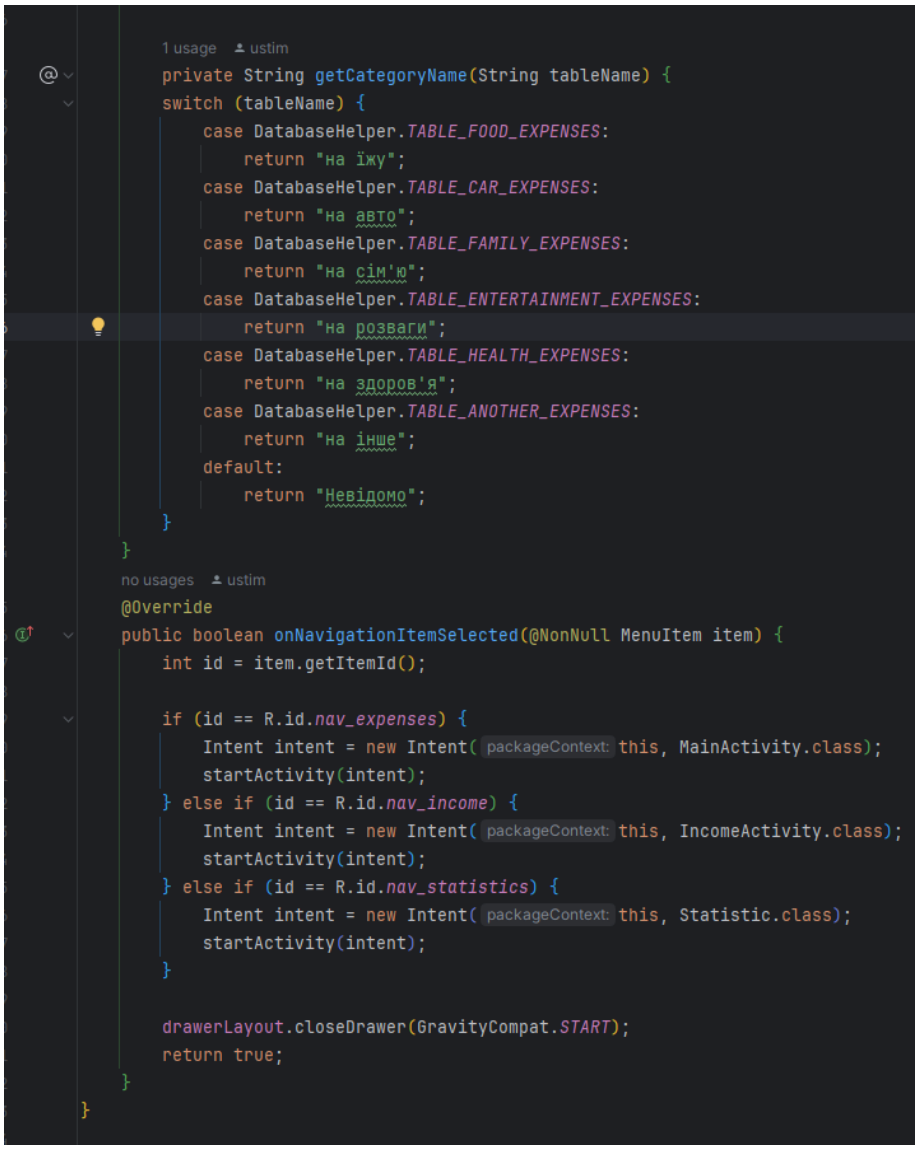
```
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                activity: this, drawerLayout, toolbar, "Open Navigation Drawer", "Close Navigation Drawer");
        drawerLayout.addDrawerListener(toggle);
        toggle.syncState();

        navigationView.setNavigationItemSelectedListener(this);

        TextView cashIncomeTextView = findViewById(R.id.cash_income_text_view);
        TextView cashlessIncomeTextView = findViewById(R.id.cashless_income_text_view);
        cashIncomeTextView.setText(String.format("Готівка: %.2f грн", cashIncome));
        cashlessIncomeTextView.setText(String.format("Картка: %.2f грн", cashlessIncome));

        ustim
        cashButton.setOnClickListener(new View.OnClickListener() {
            ustim
            @Override
            public void onClick(View v) {
                Intent intent = new Intent( packageContext: IncomeActivity.this, CashActivity.class);
                startActivity(intent);
            }
        });

        ustim
        cashlessButton.setOnClickListener(new View.OnClickListener() {
            ustim
            @Override
            public void onClick(View v) {
                Intent intent = new Intent( packageContext: IncomeActivity.this, CashlesActivity.class);
                startActivity(intent);
            }
        });
        updateBalance();
        TextView balanceTextView = findViewById(R.id.balance_amount_text_view);
        double totalBalance = dbHelper.getTotalBalance();
        balanceTextView.setText(String.format(Locale.getDefault(), format: "%.2f грн", totalBalance));
    }
    no usages  ustim
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.nav_expenses) {
            Intent intent = new Intent( packageContext: this, MainActivity.class);
            startActivity(intent);
```

```
    no usages  ustim
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.nav_expenses) {
            Intent intent = new Intent( packageContext: this, MainActivity.class);
            startActivity(intent);
        } else if (id == R.id.nav_income) {
            Intent intent = new Intent( packageContext: this, IncomeActivity.class);
            startActivity(intent);
        } else if (id == R.id.nav_statistics) {
            Intent intent = new Intent( packageContext: this, Statistic.class);
            startActivity(intent);
        }

        drawerLayout.closeDrawer(GravityCompat.START);
        return true;
    }
    1 usage  ustim
    private void updateBalance() {
        TextView balanceTextView = findViewById(R.id.balance_amount_text_view);
        double totalBalance = dbHelper.getBalance( tableName: "total");
        balanceTextView.setText(String.format(Locale.getDefault(), format: "%.2f грн", totalBalance));
    }
}
```

Код класу Statistic:

```java
package com.example.app;

import android.content.Intent;
import android.os.Bundle;
import android.view.MenuItem;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;
import com.google.android.material.navigation.NavigationView;

ustim
public class Statistic extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener {
    4 usages
    private DrawerLayout drawerLayout;
    3 usages
    private ActionBarDrawerToggle toggle;
    3 usages
    private Toolbar toolbar;
    4 usages
    private DatabaseHelper databaseHelper;
    2 usages
    private TextView expensesTextView;
    2 usages
    private TextView incomeTextView;
    ustim
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_statistic);
        toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        drawerLayout = findViewById(R.id.drawer_layout);
        toggle = new ActionBarDrawerToggle( activity: this, drawerLayout, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
        drawerLayout.addDrawerListener(toggle);
        toggle.syncState();
        NavigationView navigationView = findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);
        databaseHelper = new DatabaseHelper( context: this);
```

```
expensesTextView = findViewById(R.id.text_expenses);
incomeTextView = findViewById(R.id.text_income);
double expenses = databaseHelper.getCurrentMonthExpenses();
double income = databaseHelper.getCurrentMonthIncome();

expensesTextView.setText("Витрати за поточний місяць: " + expenses+" грн.");
incomeTextView.setText("Доходи на поточний місяць: " + income+" грн.");
double maxExpense = Double.MIN_VALUE;
String maxCategory = "";
String[] categories = {
        DatabaseHelper.TABLE_FOOD_EXPENSES,
        DatabaseHelper.TABLE_CAR_EXPENSES,
        DatabaseHelper.TABLE_FAMILY_EXPENSES,
        DatabaseHelper.TABLE_ENTERTAINMENT_EXPENSES,
        DatabaseHelper.TABLE_HEALTH_EXPENSES,
        DatabaseHelper.TABLE_ANOTHER_EXPENSES
};

for (String category : categories) {
    double expenseSum = databaseHelper.getExpenseSumForCategory(category);
    if (expenseSum > maxExpense) {
        maxExpense = expenseSum;
        maxCategory = category;
    }
}
String maxCategoryName = getCategoryName(maxCategory);
TextView maxExpenseTextView = findViewById(R.id.text_max_expense);
maxExpenseTextView.setText("Найбільші витрати " + maxCategoryName);


1 usage  ustim
private String getCategoryName(String tableName) {
switch (tableName) {
    case DatabaseHelper.TABLE_FOOD_EXPENSES:
        return "на їжу";
    case DatabaseHelper.TABLE_CAR_EXPENSES:
        return "на авто";
    case DatabaseHelper.TABLE_FAMILY_EXPENSES:
        return "на сім'ю";
    case DatabaseHelper.TABLE_ENTERTAINMENT_EXPENSES:
        return "на розваги";
    case DatabaseHelper.TABLE_HEALTH_EXPENSES:
        return "на здоров'я";
```



```
    1 usage  ustim
    private String getCategoryName(String tableName) {
    switch (tableName) {
        case DatabaseHelper.TABLE_FOOD_EXPENSES:
            return "на їжу";
        case DatabaseHelper.TABLE_CAR_EXPENSES:
            return "на авто";
        case DatabaseHelper.TABLE_FAMILY_EXPENSES:
            return "на сім'ю";
        case DatabaseHelper.TABLE_ENTERTAINMENT_EXPENSES:
            return "на розваги";
        case DatabaseHelper.TABLE_HEALTH_EXPENSES:
            return "на здоров'я";
        case DatabaseHelper.TABLE_ANOTHER_EXPENSES:
            return "на інше";
        default:
            return "Невідомо";
    }
}
    no usages  ustim
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.nav_expenses) {
            Intent intent = new Intent( packageContext: this, MainActivity.class);
            startActivity(intent);
        } else if (id == R.id.nav_income) {
            Intent intent = new Intent( packageContext: this, IncomeActivity.class);
            startActivity(intent);
        } else if (id == R.id.nav_statistics) {
            Intent intent = new Intent( packageContext: this, Statistic.class);
            startActivity(intent);
        }

        drawerLayout.closeDrawer(GravityCompat.START);
        return true;
    }
}
```

Код класу CashActivity:

```
package com.example.app;
import android.graphics.Typeface;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.List;
 ustim *
public class CashActivity extends AppCompatActivity {
    4 usages
    private DatabaseHelper dbHelper;

     ustim
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cash);

        dbHelper = new DatabaseHelper( context: this);
        displayIncome();
    }

    2 usages   ustim
    private void displayIncome() {
        List<Income> allIncomeCash = dbHelper.getAllIncomeCash();
        TableLayout incomeTable = findViewById(R.id.cash_income_table);
        incomeTable.removeAllViews();
        TableRow headerRow = new TableRow( context: this);
        TextView amountHeader = new TextView( context: this);
        amountHeader.setText("Сума");
        amountHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView commentHeader = new TextView( context: this);
        commentHeader.setText("Коментар");
        commentHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView dateHeader = new TextView( context: this);
        dateHeader.setText("Дата");
        dateHeader.setTypeface( tf: null, Typeface.BOLD);

        headerRow.addView(amountHeader);
```

```
        headerRow.addView(commentHeader);
        headerRow.addView(dateHeader);

        incomeTable.addView(headerRow);
        for (Income income : allIncomeCash) {
            TableRow row = new TableRow( context: this);
            TextView amountText = new TextView( context: this);
            amountText.setText(income.getAmount() + " грн");
            TextView commentText = new TextView( context: this);
            commentText.setText(income.getComment());
            TextView dateText = new TextView( context: this);
            dateText.setText(income.getDate());
            row.addView(amountText);
            row.addView(commentText);
            row.addView(dateText);
            incomeTable.addView(row);
        }
    }

    1 usage   ustim *
    public void addIncomeCash(View view) {
        EditText amountEditText = findViewById(R.id.cash_icome_input);
        EditText commentEditText = findViewById(R.id.cash_comment_input);
        double amount = Double.parseDouble(amountEditText.getText().toString());
        String comment = commentEditText.getText().toString();
        dbHelper.insertIncomeCash(amount, comment);
        dbHelper.updateBalance(amount);
        displayIncome();
    }

}
```

Код класу CashlesActivity:

```java
package com.example.app;
import android.graphics.Typeface;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.List;
 ustim *
public class CashlesActivity extends AppCompatActivity {
    4 usages
    private DatabaseHelper dbHelper;

     ustim
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cashles);

        dbHelper = new DatabaseHelper( context: this);
        displayIncome();
    }

    2 usages  ustim
    private void displayIncome() {
        List<Income> allIncomeCashles = dbHelper.getAllIncomeCashles();
        TableLayout incomeTable = findViewById(R.id.cashles_income_table);
        incomeTable.removeAllViews();
        TableRow headerRow = new TableRow( context: this);
        TextView amountHeader = new TextView( context: this);
        amountHeader.setText("Сума");
        amountHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView commentHeader = new TextView( context: this);
        commentHeader.setText("Коментар");
        commentHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView dateHeader = new TextView( context: this);
        dateHeader.setText("Дата");
        dateHeader.setTypeface( tf: null, Typeface.BOLD);

        headerRow.addView(amountHeader);
```

```
        headerRow.addView(amountHeader);
        headerRow.addView(commentHeader);
        headerRow.addView(dateHeader);

        incomeTable.addView(headerRow);
        for (Income income : allIncomeCashles) {
            TableRow row = new TableRow( context: this);
            TextView amountText = new TextView( context: this);
            amountText.setText(income.getAmount() + " грн");
            TextView commentText = new TextView( context: this);
            commentText.setText(income.getComment());
            TextView dateText = new TextView( context: this);
            dateText.setText(income.getDate());
            row.addView(amountText);
            row.addView(commentText);
            row.addView(dateText);
            incomeTable.addView(row);
        }
    }

    1 usage   ustim *
    public void addIncomeCashles(View view) {
        EditText amountEditText = findViewById(R.id.cashles_icome_input);
        EditText commentEditText = findViewById(R.id.cashles_comment_input);
        double amount = Double.parseDouble(amountEditText.getText().toString());
        String comment = commentEditText.getText().toString();
        dbHelper.insertIncomeCash(amount, comment);
        dbHelper.updateBalance(amount);
        displayIncome();
    }

}
```

Код класу Income(Конструктор):

```
public class Income {
    3 usages
    private int id;
    3 usages
    private double amount;
    3 usages
    private String comment;
    3 usages
    private String date;

    no usages  ustim
    public Income(int id, double amount, String comment, String date) {
        this.id = id;
        this.amount = amount;
        this.comment = comment;
        this.date = date;
    }

    2 usages  ustim
    public Income() {

    }

    ustim
    public int getId() { return id; }

    ustim
    public void setId(int id) {
        this.id = id;
    }

    ustim
    public double getAmount() { return amount; }

    ustim
    public void setAmount(double amount) { this.amount = amount; }

    ustim
    public String getComment() { return comment; }

    ustim
    public void setComment(String comment) { this.comment = comment; }
```

```
    2 usages  ustim
    public String getDate() { return date; }

    2 usages  ustim
    public void setDate(String date) { this.date = date; }
}
```

Код класу Expense(Конструктор):

```
public class Expense {
    3 usages
    private int id;
    3 usages
    private double amount;
    3 usages
    private String comment;
    2 usages
    private String date;
    no usages  ustim
    public Expense(int id, double amount, String comment) {
        this.id = id;
        this.amount = amount;
        this.comment = comment;
    }

    6 usages  ustim
    public Expense() {

    }


    ustim
    public int getId() { return id; }

    ustim
    public void setId(int id) { this.id = id; }

    ustim
    public double getAmount() { return amount; }

    ustim
    public void setAmount(double amount) { this.amount = amount; }

    ustim
    public String getComment() { return comment; }

    ustim
    public void setComment(String comment) { this.comment = comment; }
    6 usages  ustim
    public String getDate() { return date; }
    6 usages  ustim
    public void setDate(String date) { this.date = date; }
```

Код класу FoodExpense:

```
package com.example.app;

import android.graphics.Typeface;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.List;

ustim
public class FoodActivity extends AppCompatActivity {

    5 usages
    private DatabaseHelper dbHelper;

    ustim
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_food);

        dbHelper = new DatabaseHelper( context: this);
        displayExpenses();
    }

    3 usages   ustim
    private void displayExpenses() {
        List<Expense> allExpenses = dbHelper.getAllExpensesFromFood();
        TableLayout expensesTable = findViewById(R.id.food_expenses_table);
        expensesTable.removeAllViews();
        TableRow headerRow = new TableRow( context: this);
        TextView amountHeader = new TextView( context: this);
        amountHeader.setText("Сума");
        amountHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView commentHeader = new TextView( context: this);
        commentHeader.setText("Коментар");
        commentHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView dateHeader = new TextView( context: this);
        dateHeader.setText("Дата");
```

```
        dateHeader.setText("Дата");
        dateHeader.setTypeface( tf: null, Typeface.BOLD);
        headerRow.addView(amountHeader);
        headerRow.addView(commentHeader);
        headerRow.addView(dateHeader);
        expensesTable.addView(headerRow);
        for (Expense expense : allExpenses) {
            TableRow row = new TableRow( context: this);
            TextView amountText = new TextView( context: this);
            amountText.setText(expense.getAmount() + " грн");
            TextView commentText = new TextView( context: this);
            commentText.setText(expense.getComment());
            TextView dateText = new TextView( context: this);
            dateText.setText(expense.getDate());
            row.addView(amountText);
            row.addView(commentText);
            row.addView(dateText);
            expensesTable.addView(row);
        }
    }

    ustim
    public void addExpense(View view) {
        EditText amountEditText = findViewById(R.id.food_expense_input);
        EditText commentEditText = findViewById(R.id.food_comment_input);
        double amount = Double.parseDouble(amountEditText.getText().toString());
        String comment = commentEditText.getText().toString();
        dbHelper.insertFoodExpense(amount, comment);
        dbHelper.decreaseBalance(amount);
        displayExpenses();
    }
    ustim
    public void clearExpenses(View view) {
        dbHelper.clearFoodExpenses();
        displayExpenses();
    }
}
```

Код класу FamilyActivity:

```java
package com.example.app;

import android.graphics.Typeface;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.List;

ustim
public class FamilyActivity extends AppCompatActivity {

    5 usages
    private DatabaseHelper dbHelper;

    ustim
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_family);

        dbHelper = new DatabaseHelper( context: this);
        displayExpenses();
    }

    3 usages  ustim
    private void displayExpenses() {
        List<Expense> allExpenses = dbHelper.getAllExpensesFromFamily();
        TableLayout expensesTable = findViewById(R.id.family_expenses_table);
        expensesTable.removeAllViews();
        TableRow headerRow = new TableRow( context: this);
        TextView amountHeader = new TextView( context: this);
        amountHeader.setText("Сума");
        amountHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView commentHeader = new TextView( context: this);
        commentHeader.setText("Коментар");
        commentHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView dateHeader = new TextView( context: this);
        dateHeader.setText("Дата");
```

```
        dateHeader.setText("Дата");
        dateHeader.setTypeface( tf: null, Typeface.BOLD);

        headerRow.addView(amountHeader);
        headerRow.addView(commentHeader);
        headerRow.addView(dateHeader);

        expensesTable.addView(headerRow);
        for (Expense expense : allExpenses) {
            TableRow row = new TableRow( context: this);
            TextView amountText = new TextView( context: this);
            amountText.setText(expense.getAmount() + " грн");
            TextView commentText = new TextView( context: this);
            commentText.setText(expense.getComment());
            TextView dateText = new TextView( context: this);
            dateText.setText(expense.getDate());
            row.addView(amountText);
            row.addView(commentText);
            row.addView(dateText);
            expensesTable.addView(row);
        }
    }

    ustim
    public void addExpense(View view) {
        EditText amountEditText = findViewById(R.id.family_expense_input);
        EditText commentEditText = findViewById(R.id.family_comment_input);
        double amount = Double.parseDouble(amountEditText.getText().toString());
        String comment = commentEditText.getText().toString();
        dbHelper.insertFamilyExpense(amount, comment);
        dbHelper.decreaseBalance(amount);
        displayExpenses();
    }
    ustim
    public void clearExpenses(View view) {
        dbHelper.clearFamilyExpenses();
        displayExpenses();
    }
}
```

Код класу AutoActivity:

```
package com.example.app;

import android.graphics.Typeface;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.List;

ustim *
public class AutoActivity extends AppCompatActivity {

    5 usages
    private DatabaseHelper dbHelper;

    ustim
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_car);

        dbHelper = new DatabaseHelper( context: this);
        displayExpenses();
    }

    3 usages  ustim
    private void displayExpenses() {
        List<Expense> allExpenses = dbHelper.getAllExpensesFromCar();
        TableLayout expensesTable = findViewById(R.id.auto_expenses_table);
        expensesTable.removeAllViews();
        TableRow headerRow = new TableRow( context: this);
        TextView amountHeader = new TextView( context: this);
        amountHeader.setText("Сума");
        amountHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView commentHeader = new TextView( context: this);
        commentHeader.setText("Коментар");
        commentHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView dateHeader = new TextView( context: this);
        dateHeader.setText("Дата");
```

```
        commentHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView dateHeader = new TextView( context: this);
        dateHeader.setText("Дата");
        dateHeader.setTypeface( tf: null, Typeface.BOLD);

        headerRow.addView(amountHeader);
        headerRow.addView(commentHeader);
        headerRow.addView(dateHeader);

        expensesTable.addView(headerRow);
        for (Expense expense : allExpenses) {
            TableRow row = new TableRow( context: this);
            TextView amountText = new TextView( context: this);
            amountText.setText(expense.getAmount() + " грн");
            TextView commentText = new TextView( context: this);
            commentText.setText(expense.getComment());
            TextView dateText = new TextView( context: this);
            dateText.setText(expense.getDate());
            row.addView(amountText);
            row.addView(commentText);
            row.addView(dateText);
            expensesTable.addView(row);
        }
    }

    ustim *
    public void addExpense(View view) {
        EditText amountEditText = findViewById(R.id.auto_expense_input);
        EditText commentEditText = findViewById(R.id.auto_comment_input);
        double amount = Double.parseDouble(amountEditText.getText().toString());
        String comment = commentEditText.getText().toString();
        dbHelper.decreaseBalance(amount);
        dbHelper.insertCarExpense(amount, comment);
        displayExpenses();
    }
    ustim
    public void clearExpenses(View view) {
        dbHelper.clearCarExpenses();
        displayExpenses();
    }
}
```

Код класу HealthActivity:

```java
package com.example.app;

import android.graphics.Typeface;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.List;

ustim *
public class HealthActivity extends AppCompatActivity {

    5 usages
    private DatabaseHelper dbHelper;

    ustim
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_health);

        dbHelper = new DatabaseHelper( context: this);
        displayExpenses();
    }

    3 usages  ustim *
    private void displayExpenses() {
        List<Expense> allExpenses = dbHelper.getAllExpensesFromHealth();
        TableLayout expensesTable = findViewById(R.id.health_expenses_table);
        expensesTable.removeAllViews();
        TableRow headerRow = new TableRow( context: this);
        TextView amountHeader = new TextView( context: this);
        amountHeader.setText("Сума");
        amountHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView commentHeader = new TextView( context: this);
        commentHeader.setText("Коментар");
        commentHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView dateHeader = new TextView( context: this);
        dateHeader.setText("Дата");
```

```
        dateHeader.setTypeface( tf: null, Typeface.BOLD);

        headerRow.addView(amountHeader);
        headerRow.addView(commentHeader);
        headerRow.addView(dateHeader);

        expensesTable.addView(headerRow);
        for (Expense expense : allExpenses) {
            TableRow row = new TableRow( context: this);
            TextView amountText = new TextView( context: this);
            amountText.setText(expense.getAmount() + " грн");
            TextView commentText = new TextView( context: this);
            commentText.setText(expense.getComment());
            TextView dateText = new TextView( context: this);
            dateText.setText(expense.getDate());
            row.addView(amountText);
            row.addView(commentText);
            row.addView(dateText);
            expensesTable.addView(row);
        }
    }

    ustim *
    public void addExpense(View view) {
        EditText amountEditText = findViewById(R.id.health_expense_input);
        EditText commentEditText = findViewById(R.id.health_comment_input);
        double amount = Double.parseDouble(amountEditText.getText().toString());
        String comment = commentEditText.getText().toString();
        dbHelper.insertHealthExpense(amount, comment);
        dbHelper.decreaseBalance(amount);
        displayExpenses();
    }
    ustim *
    public void clearExpenses(View view) {
        dbHelper.clearHealthExpenses();
        displayExpenses();
    }
}
```

Код класу EntertainmentActivity:

```
package com.example.app;

import android.graphics.Typeface;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.List;

ustim
public class EntertainmentActivity extends AppCompatActivity {

    5 usages
    private DatabaseHelper dbHelper;

    ustim
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_entertainment);

        dbHelper = new DatabaseHelper( context: this);
        displayExpenses();
    }

    3 usages  ustim
    private void displayExpenses() {
        List<Expense> allExpenses = dbHelper.getAllExpensesFromEntertainment();
        TableLayout expensesTable = findViewById(R.id.entertainment_expenses_table);
        expensesTable.removeAllViews();
        TableRow headerRow = new TableRow( context: this);
        TextView amountHeader = new TextView( context: this);
        amountHeader.setText("Сума");
        amountHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView commentHeader = new TextView( context: this);
        commentHeader.setText("Коментар");
        commentHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView dateHeader = new TextView( context: this);
        dateHeader.setText("Дата");
```

```
        dateHeader.setTypeface( tf: null, Typeface.BOLD);

        headerRow.addView(amountHeader);
        headerRow.addView(commentHeader);
        headerRow.addView(dateHeader);

        expensesTable.addView(headerRow);
        for (Expense expense : allExpenses) {
            TableRow row = new TableRow( context: this);
            TextView amountText = new TextView( context: this);
            amountText.setText(expense.getAmount() + " грн");
            TextView commentText = new TextView( context: this);
            commentText.setText(expense.getComment());
            TextView dateText = new TextView( context: this);
            dateText.setText(expense.getDate());
            row.addView(amountText);
            row.addView(commentText);
            row.addView(dateText);
            expensesTable.addView(row);
        }
    }

    ustim
    public void addExpense(View view) {
        EditText amountEditText = findViewById(R.id.entertainment_expense_input);
        EditText commentEditText = findViewById(R.id.entertainment_comment_input);
        double amount = Double.parseDouble(amountEditText.getText().toString());
        String comment = commentEditText.getText().toString();
        dbHelper.insertEntertainmentExpense(amount, comment);
        dbHelper.decreaseBalance(amount);
        displayExpenses();
    }
    ustim
    public void clearExpenses(View view) {
        dbHelper.clearEntertainmentExpenses();
        displayExpenses();
    }
}
```

Код класу AnotherActivity:

```java
package com.example.app;

import android.graphics.Typeface;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.List;

ustim *
public class AnotherActivity extends AppCompatActivity {

    5 usages
    private DatabaseHelper dbHelper;

    ustim
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_another);

        dbHelper = new DatabaseHelper( context: this);
        displayExpenses();
    }

    3 usages  ustim
    private void displayExpenses() {
        List<Expense> allExpenses = dbHelper.getAllExpensesFromAnother();
        TableLayout expensesTable = findViewById(R.id.another_expenses_table);
        expensesTable.removeAllViews();
        TableRow headerRow = new TableRow( context: this);
        TextView amountHeader = new TextView( context: this);
        amountHeader.setText("Сума");
        amountHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView commentHeader = new TextView( context: this);
        commentHeader.setText("Коментар");
        commentHeader.setTypeface( tf: null, Typeface.BOLD);
        TextView dateHeader = new TextView( context: this);
        dateHeader.setText("Дата");
```

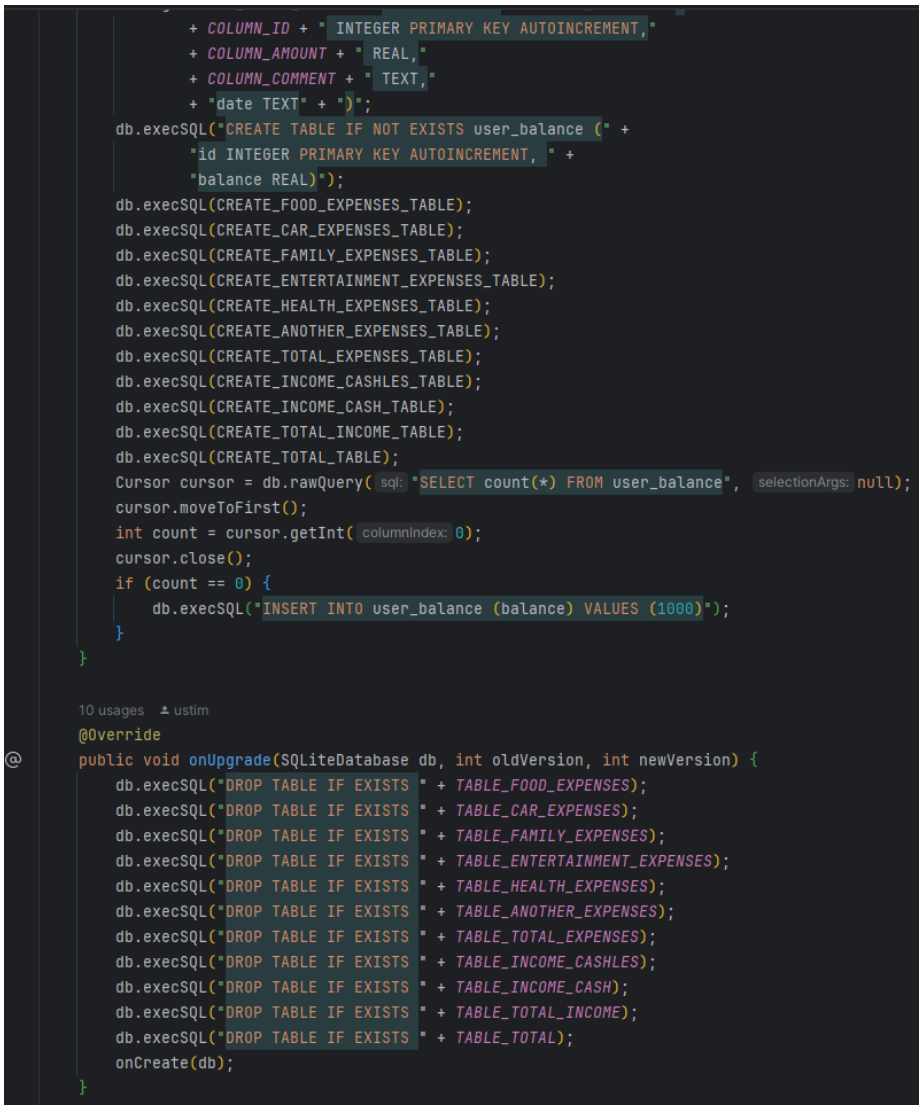
```
        dateHeader.setText("Дата");
        dateHeader.setTypeface( tf: null, Typeface.BOLD);

        headerRow.addView(amountHeader);
        headerRow.addView(commentHeader);
        headerRow.addView(dateHeader);

        expensesTable.addView(headerRow);
        for (Expense expense : allExpenses) {
            TableRow row = new TableRow( context: this);
            TextView amountText = new TextView( context: this);
            amountText.setText(expense.getAmount() + " грн");
            TextView commentText = new TextView( context: this);
            commentText.setText(expense.getComment());
            TextView dateText = new TextView( context: this);
            dateText.setText(expense.getDate());
            row.addView(amountText);
            row.addView(commentText);
            row.addView(dateText);
            expensesTable.addView(row);
        }
    }

    ustim *
    public void addExpense(View view) {
        EditText amountEditText = findViewById(R.id.another_expense_input);
        EditText commentEditText = findViewById(R.id.another_comment_input);
        double amount = Double.parseDouble(amountEditText.getText().toString());
        String comment = commentEditText.getText().toString();
        dbHelper.decreaseBalance(amount);
        dbHelper.insertAnotherExpense(amount, comment);
        displayExpenses();
    }
    ustim
    public void clearExpenses(View view) {
        dbHelper.clearCarExpenses();
        displayExpenses();
    }
}
```

Код класу DataBaseHelper:

```java
package com.example.app;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Locale;


 ustim *
public class DatabaseHelper extends SQLiteOpenHelper {
    6 usages
    private MainActivity mActivity;
    1 usage
    private IncomeActivity iActivity;
    1 usage
    private static final int DATABASE_VERSION = 23;
    1 usage
    private static final String DATABASE_NAME = "expenses_db";
    7 usages
    protected static final String TABLE_FOOD_EXPENSES = "food_expenses";
    7 usages
    protected static final String TABLE_CAR_EXPENSES = "car_expenses";
    7 usages
    protected static final String TABLE_FAMILY_EXPENSES = "family_expenses";
    7 usages
    protected static final String TABLE_ENTERTAINMENT_EXPENSES = "entertainment_expenses";
    7 usages
    protected static final String TABLE_HEALTH_EXPENSES = "health_expenses";
    6 usages
    protected static final String TABLE_ANOTHER_EXPENSES = "another_expenses";
    6 usages
    protected static final String TABLE_INCOME_CASHLES = "cashles";
    6 usages
    protected static final String TABLE_INCOME_CASH = "cash";
```

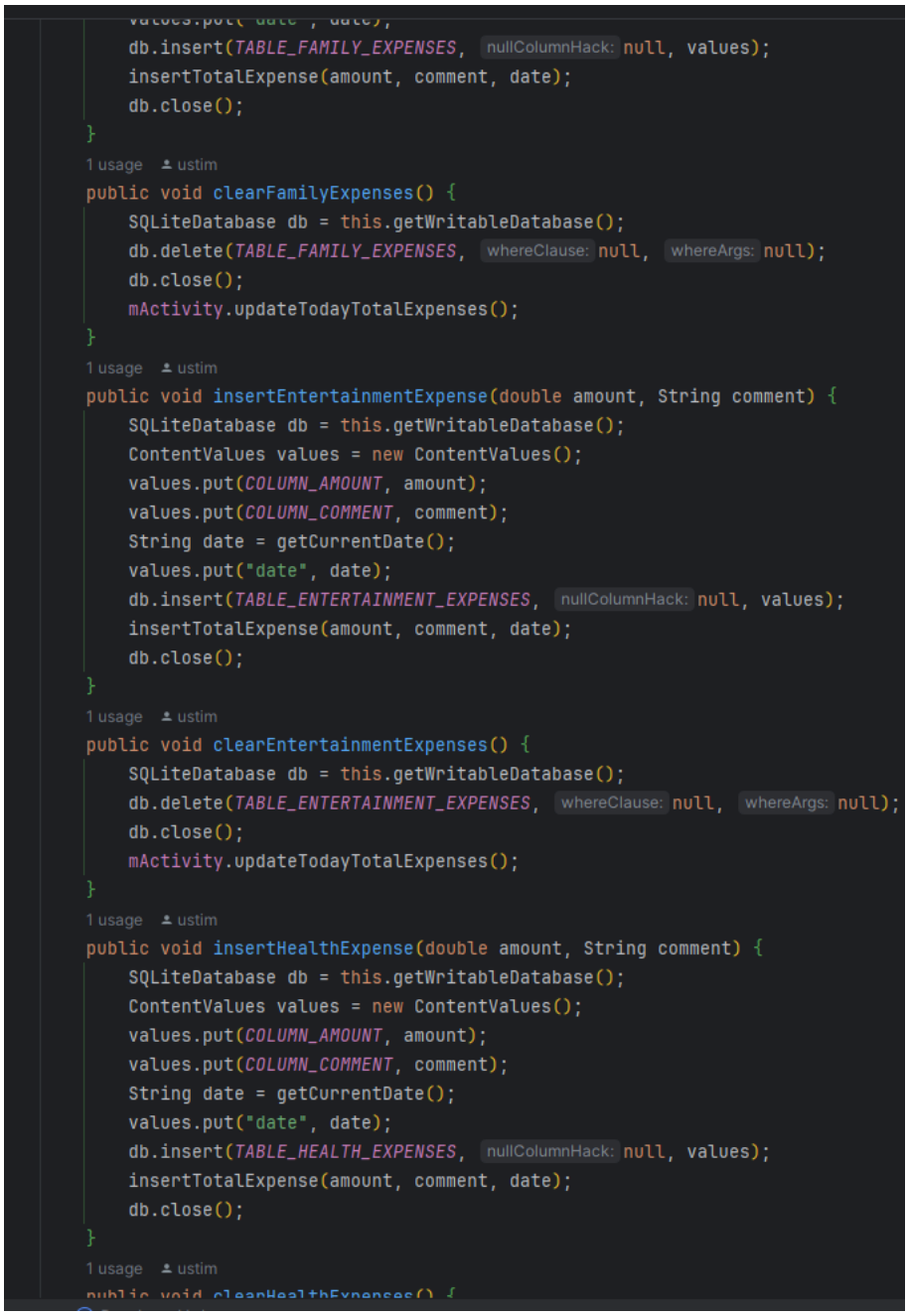
```
6 usages
protected static final String TABLE_INCOME_CASH = "cash";
2 usages
private static final String COLUMN_SOURCE = "source";
11 usages
private static final String COLUMN_ID = "id";
32 usages
protected static final String COLUMN_AMOUNT = "amount";
23 usages
private static final String COLUMN_COMMENT = "comment";
5 usages
private static final String TABLE_TOTAL_EXPENSES = "total_expenses";
4 usages
private static final String TABLE_TOTAL_INCOME = "total_income";
3 usages
private static final String TABLE_TOTAL = "total";
2 usages
private final MutableLiveData<Double> todayTotalExpensesLiveData = new MutableLiveData<>();
1 usage  ustim
public void setMainActivity(MainActivity activity) { mActivity = activity; }
1 usage  ustim
public void setIncomeActivity(IncomeActivity Iactivity) { iActivity = Iactivity; }
 ustim
public DatabaseHelper(Context context) {
    super(context, DATABASE_NAME,  factory: null, DATABASE_VERSION);
}
1 usage  ustim
public LiveData<Double> getTodayTotalExpensesLiveData() { return todayTotalExpensesLiveData; }

 ustim
@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_FOOD_EXPENSES_TABLE = "CREATE TABLE " + TABLE_FOOD_EXPENSES + "("
            + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
            + COLUMN_AMOUNT + " REAL,"
            + COLUMN_COMMENT + " TEXT,"
            + "date TEXT" + ")";
    String CREATE_CAR_EXPENSES_TABLE = "CREATE TABLE " + TABLE_CAR_EXPENSES + "("
            + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
            + COLUMN_AMOUNT + " REAL,"
            + COLUMN_COMMENT + " TEXT,"
            + "date TEXT" + ")";
    String CREATE_FAMILY_EXPENSES_TABLE = "CREATE TABLE " + TABLE_FAMILY_EXPENSES + "("
            + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
```

```
        + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_AMOUNT + " REAL,"
        + COLUMN_COMMENT + " TEXT,"
        + "date TEXT" + ")";
String CREATE_ENTERTAINMENT_EXPENSES_TABLE = "CREATE TABLE " + TABLE_ENTERTAINMENT_EXPENSES + "("
        + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_AMOUNT + " REAL,"
        + COLUMN_COMMENT + " TEXT,"
        + "date TEXT" + ")";
String CREATE_HEALTH_EXPENSES_TABLE = "CREATE TABLE " + TABLE_HEALTH_EXPENSES + "("
        + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_AMOUNT + " REAL,"
        + COLUMN_COMMENT + " TEXT,"
        + "date TEXT" + ")";
String CREATE_ANOTHER_EXPENSES_TABLE = "CREATE TABLE " + TABLE_ANOTHER_EXPENSES + "("
        + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_AMOUNT + " REAL,"
        + COLUMN_COMMENT + " TEXT,"
        + "date TEXT" + ")";
String CREATE_TOTAL_EXPENSES_TABLE = "CREATE TABLE " + TABLE_TOTAL_EXPENSES + "("
        + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_AMOUNT + " REAL,"
        + COLUMN_COMMENT + " TEXT,"
        + "date TEXT" + ")";
String CREATE_INCOME_CASHLES_TABLE = "CREATE TABLE " + TABLE_INCOME_CASHLES + "("
        + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_AMOUNT + " REAL,"
        + COLUMN_COMMENT + " TEXT,"
        + "date TEXT,"
        + COLUMN_SOURCE + " TEXT" + ")";
String CREATE_INCOME_CASH_TABLE = "CREATE TABLE " + TABLE_INCOME_CASH + "("
        + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_AMOUNT + " REAL,"
        + COLUMN_COMMENT + " TEXT,"
        + "date TEXT,"
        + COLUMN_SOURCE + " TEXT" + ")";
String CREATE_TOTAL_INCOME_TABLE = "CREATE TABLE " + TABLE_TOTAL_INCOME + "("
        + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_AMOUNT + " REAL,"
        + COLUMN_COMMENT + " TEXT,"
        + "date TEXT" + ")";
String CREATE_TOTAL_TABLE = "CREATE TABLE " + TABLE_TOTAL + "("
        + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_AMOUNT + " REAL,"
```

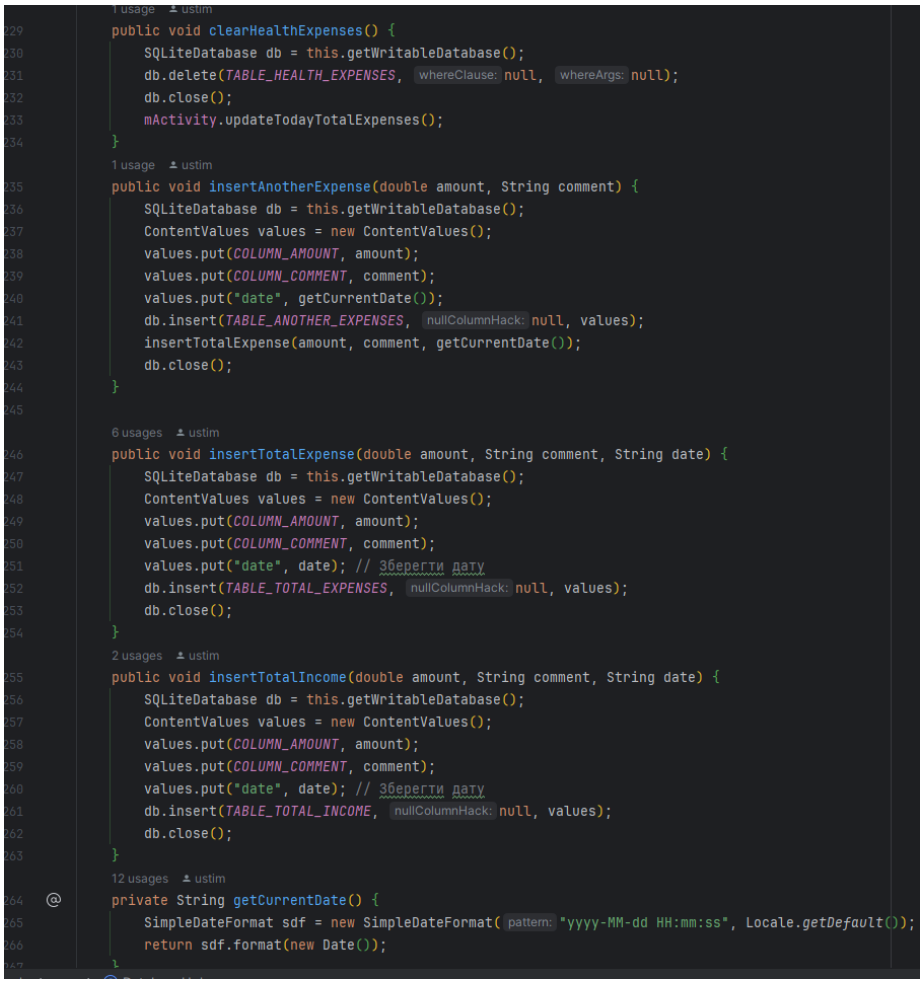


```
            + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
            + COLUMN_AMOUNT + " REAL,"
            + COLUMN_COMMENT + " TEXT,"
            + "date TEXT" + ")";
    db.execSQL("CREATE TABLE IF NOT EXISTS user_balance (" +
            "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "balance REAL)");
    db.execSQL(CREATE_FOOD_EXPENSES_TABLE);
    db.execSQL(CREATE_CAR_EXPENSES_TABLE);
    db.execSQL(CREATE_FAMILY_EXPENSES_TABLE);
    db.execSQL(CREATE_ENTERTAINMENT_EXPENSES_TABLE);
    db.execSQL(CREATE_HEALTH_EXPENSES_TABLE);
    db.execSQL(CREATE_ANOTHER_EXPENSES_TABLE);
    db.execSQL(CREATE_TOTAL_EXPENSES_TABLE);
    db.execSQL(CREATE_INCOME_CASHLES_TABLE);
    db.execSQL(CREATE_INCOME_CASH_TABLE);
    db.execSQL(CREATE_TOTAL_INCOME_TABLE);
    db.execSQL(CREATE_TOTAL_TABLE);
    Cursor cursor = db.rawQuery( sql: "SELECT count(*) FROM user_balance",  selectionArgs: null);
    cursor.moveToFirst();
    int count = cursor.getInt( columnIndex: 0);
    cursor.close();
    if (count == 0) {
        db.execSQL("INSERT INTO user_balance (balance) VALUES (1000)");
    }
}

10 usages  ustim
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_FOOD_EXPENSES);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_CAR_EXPENSES);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_FAMILY_EXPENSES);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_ENTERTAINMENT_EXPENSES);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_HEALTH_EXPENSES);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_ANOTHER_EXPENSES);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_TOTAL_EXPENSES);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_INCOME_CASHLES);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_INCOME_CASH);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_TOTAL_INCOME);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_TOTAL);
    onCreate(db);
}
```

```
public void insertFoodExpense(double amount, String comment) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_AMOUNT, amount);
    values.put(COLUMN_COMMENT, comment);
    String date = getCurrentDate();
    values.put("date", date);
    db.insert(TABLE_FOOD_EXPENSES, nullColumnHack: null, values);
    insertTotalExpense(amount, comment, date);
    db.close();
}
1 usage  ustim
public void clearFoodExpenses() {
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(TABLE_FOOD_EXPENSES, whereClause: null, whereArgs: null);
    db.close();
    mActivity.updateTodayTotalExpenses();
}
1 usage  ustim
public void insertCarExpense(double amount, String comment) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_AMOUNT, amount);
    values.put(COLUMN_COMMENT, comment);
    values.put("date", getCurrentDate());
    db.insert(TABLE_CAR_EXPENSES, nullColumnHack: null, values);
    insertTotalExpense(amount, comment, getCurrentDate());
    db.close();
}
2 usages  ustim
public void clearCarExpenses() {
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(TABLE_CAR_EXPENSES, whereClause: null, whereArgs: null);
    db.close();
    mActivity.updateTodayTotalExpenses();
}
1 usage  ustim
public void insertFamilyExpense(double amount, String comment) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_AMOUNT, amount);
    values.put(COLUMN_COMMENT, comment);
    String date = getCurrentDate();
    values.put("date", date);
```



```
        db.insert(TABLE_FAMILY_EXPENSES, nullColumnHack: null, values);
        insertTotalExpense(amount, comment, date);
        db.close();
    }
    1 usage  ustim
    public void clearFamilyExpenses() {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_FAMILY_EXPENSES, whereClause: null, whereArgs: null);
        db.close();
        mActivity.updateTodayTotalExpenses();
    }
    1 usage  ustim
    public void insertEntertainmentExpense(double amount, String comment) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COLUMN_AMOUNT, amount);
        values.put(COLUMN_COMMENT, comment);
        String date = getCurrentDate();
        values.put("date", date);
        db.insert(TABLE_ENTERTAINMENT_EXPENSES, nullColumnHack: null, values);
        insertTotalExpense(amount, comment, date);
        db.close();
    }
    1 usage  ustim
    public void clearEntertainmentExpenses() {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_ENTERTAINMENT_EXPENSES, whereClause: null, whereArgs: null);
        db.close();
        mActivity.updateTodayTotalExpenses();
    }
    1 usage  ustim
    public void insertHealthExpense(double amount, String comment) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COLUMN_AMOUNT, amount);
        values.put(COLUMN_COMMENT, comment);
        String date = getCurrentDate();
        values.put("date", date);
        db.insert(TABLE_HEALTH_EXPENSES, nullColumnHack: null, values);
        insertTotalExpense(amount, comment, date);
        db.close();
    }
    1 usage  ustim
    public void clearHealthExpenses() {
```



```
1 usage  ustim
public void clearHealthExpenses() {
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(TABLE_HEALTH_EXPENSES,  whereClause: null,  whereArgs: null);
    db.close();
    mActivity.updateTodayTotalExpenses();
}
1 usage  ustim
public void insertAnotherExpense(double amount, String comment) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_AMOUNT, amount);
    values.put(COLUMN_COMMENT, comment);
    values.put("date", getCurrentDate());
    db.insert(TABLE_ANOTHER_EXPENSES,  nullColumnHack: null, values);
    insertTotalExpense(amount, comment, getCurrentDate());
    db.close();
}

6 usages  ustim
public void insertTotalExpense(double amount, String comment, String date) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_AMOUNT, amount);
    values.put(COLUMN_COMMENT, comment);
    values.put("date", date); // Зберегти дату
    db.insert(TABLE_TOTAL_EXPENSES,  nullColumnHack: null, values);
    db.close();
}
2 usages  ustim
public void insertTotalIncome(double amount, String comment, String date) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_AMOUNT, amount);
    values.put(COLUMN_COMMENT, comment);
    values.put("date", date); // Зберегти дату
    db.insert(TABLE_TOTAL_INCOME,  nullColumnHack: null, values);
    db.close();
}
12 usages  ustim
@
private String getCurrentDate() {
    SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss", Locale.getDefault());
    return sdf.format(new Date());
}
```

```
public ArrayList<Expense> getAllExpensesFromFood() {
    ArrayList<Expense> expensesList = new ArrayList<>();
    String selectQuery = "SELECT * FROM " + TABLE_FOOD_EXPENSES;
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery,  selectionArgs: null);

    if (cursor.moveToFirst()) {
        do {
            Expense expense = new Expense();
            expense.setId(cursor.getInt( columnIndex: 0));
            expense.setAmount(cursor.getDouble( columnIndex: 1));
            expense.setComment(cursor.getString( columnIndex: 2));
            expense.setDate(cursor.getString( columnIndex: 3));
            expensesList.add(expense);
        } while (cursor.moveToNext());
    }
    cursor.close();
    return expensesList;
}

1 usage  ustim
public ArrayList<Expense> getAllExpensesFromCar() {
    ArrayList<Expense> expensesList = new ArrayList<>();
    String selectQuery = "SELECT * FROM " + TABLE_CAR_EXPENSES;
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery,  selectionArgs: null);

    if (cursor.moveToFirst()) {
        do {
            Expense expense = new Expense();
            expense.setId(cursor.getInt( columnIndex: 0));
            expense.setAmount(cursor.getDouble( columnIndex: 1));
            expense.setComment(cursor.getString( columnIndex: 2));
            expense.setDate(cursor.getString( columnIndex: 3));
            expensesList.add(expense);
        } while (cursor.moveToNext());
    }
    cursor.close();
    return expensesList;
}
```

```
1 usage  ustim
public ArrayList<Expense> getAllExpensesFromEntertainment() {
    ArrayList<Expense> expensesList = new ArrayList<>();
    String selectQuery = "SELECT * FROM " + TABLE_ENTERTAINMENT_EXPENSES;
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery,  selectionArgs: null);

    if (cursor.moveToFirst()) {
        do {
            Expense expense = new Expense();
            expense.setId(cursor.getInt( columnIndex: 0));
            expense.setAmount(cursor.getDouble( columnIndex: 1));
            expense.setComment(cursor.getString( columnIndex: 2));
            expense.setDate(cursor.getString( columnIndex: 3));
            expensesList.add(expense);
        } while (cursor.moveToNext());
    }
    cursor.close();
    return expensesList;
}

1 usage  ustim
public ArrayList<Expense> getAllExpensesFromFamily() {
    ArrayList<Expense> expensesList = new ArrayList<>();
    String selectQuery = "SELECT * FROM " + TABLE_FAMILY_EXPENSES;
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery,  selectionArgs: null);

    if (cursor.moveToFirst()) {
        do {
            Expense expense = new Expense();
            expense.setId(cursor.getInt( columnIndex: 0));
            expense.setAmount(cursor.getDouble( columnIndex: 1));
            expense.setComment(cursor.getString( columnIndex: 2));
            expense.setDate(cursor.getString( columnIndex: 3));
            expensesList.add(expense);
        } while (cursor.moveToNext());
    }
    cursor.close();
    return expensesList;
}
1 usage  ustim
public ArrayList<Expense> getAllExpensesFromHealth() {
```

```java
public ArrayList<Expense> getAllExpensesFromHealth() {
    ArrayList<Expense> expensesList = new ArrayList<>();
    String selectQuery = "SELECT * FROM " + TABLE_HEALTH_EXPENSES;
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery,  selectionArgs: null);

    if (cursor.moveToFirst()) {
        do {
            Expense expense = new Expense();
            expense.setId(cursor.getInt( columnIndex: 0));
            expense.setAmount(cursor.getDouble( columnIndex: 1));
            expense.setComment(cursor.getString( columnIndex: 2));
            expense.setDate(cursor.getString( columnIndex: 3));
            expensesList.add(expense);
        } while (cursor.moveToNext());
    }
    cursor.close();
    return expensesList;
}
1 usage  ustim
public ArrayList<Expense> getAllExpensesFromAnother() {
    ArrayList<Expense> expensesList = new ArrayList<>();
    String selectQuery = "SELECT * FROM " + TABLE_ANOTHER_EXPENSES;
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery,  selectionArgs: null);

    if (cursor.moveToFirst()) {
        do {
            Expense expense = new Expense();
            expense.setId(cursor.getInt( columnIndex: 0));
            expense.setAmount(cursor.getDouble( columnIndex: 1));
            expense.setComment(cursor.getString( columnIndex: 2));
            expense.setDate(cursor.getString( columnIndex: 3));
            expensesList.add(expense);
        } while (cursor.moveToNext());
    }
    cursor.close();
    return expensesList;
}
1 usage  ustim
public double getTodayTotalExpenses() {
    double total = 0;
    SQLiteDatabase db = this.getReadableDatabase();
```

```
public double getTodayTotalExpenses() {
    double total = 0;
    SQLiteDatabase db = this.getReadableDatabase();
    String todayDate = getCurrentDateOnly();
    String selectQuery = "SELECT SUM(" + COLUMN_AMOUNT + ") as total FROM " + TABLE_TOTAL_EXPENSES + " WHERE date(date) = ?";

    Cursor cursor = db.rawQuery(selectQuery, new String[]{todayDate});

    if (cursor.moveToFirst()) {
        int totalIndex = cursor.getColumnIndex( columnName: "total");
        if (totalIndex != -1) {
            total = cursor.getDouble(totalIndex);
            todayTotalExpensesLiveData.postValue(total);
        } else {
            Log.e( tag: "DatabaseHelper",  msg: "Column 'total' not found in cursor");
        }
    }
    cursor.close();
    return total;
}

1 usage  ustim
private String getCurrentDateOnly() {
    SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyy-MM-dd", Locale.getDefault());
    return sdf.format(new Date());
}
no usages  ustim
public void insertIncomeCashles(double amount, String comment) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_AMOUNT, amount);
    values.put(COLUMN_COMMENT, comment);
    String date = getCurrentDate();
    values.put("date", date);
    db.insert(TABLE_INCOME_CASHLES,  nullColumnHack: null, values);
    insertTotalIncome(amount, comment, date);
    updateBalance(amount);
    db.close();
}
1 usage  ustim
public ArrayList<Income> getAllIncomeCashles() {
    ArrayList<Income> incomeList = new ArrayList<>();
```

```
1 usage  ustim
public ArrayList<Income> getAllIncomeCashles() {
    ArrayList<Income> incomeList = new ArrayList<>();
    String selectQuery = "SELECT * FROM " + TABLE_INCOME_CASHLES;
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery,  selectionArgs: null);

    if (cursor.moveToFirst()) {
        do {
            Income income = new Income();
            income.setId(cursor.getInt( columnIndex: 0));
            income.setAmount(cursor.getDouble( columnIndex: 1));
            income.setComment(cursor.getString( columnIndex: 2));
            income.setDate(cursor.getString( columnIndex: 3));
            incomeList.add(income);
        } while (cursor.moveToNext());
    }
    cursor.close();
    return incomeList;
}
2 usages  ustim
public void insertIncomeCash(double amount, String comment) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_AMOUNT, amount);
    values.put(COLUMN_COMMENT, comment);
    String date = getCurrentDate();
    values.put("date", date);
    db.insert(TABLE_INCOME_CASH,  nullColumnHack: null, values);
    insertTotalIncome(amount, comment, date);
    updateBalance(amount);
    db.close();
}


1 usage  ustim
public ArrayList<Income> getAllIncomeCash() {
    ArrayList<Income> incomeList = new ArrayList<>();
    String selectQuery = "SELECT * FROM " + TABLE_INCOME_CASH;
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery,  selectionArgs: null);
```

```
    if (cursor.moveToFirst()) {
        do {
            Income income = new Income();
            income.setId(cursor.getInt( columnIndex: 0));
            income.setAmount(cursor.getDouble( columnIndex: 1));
            income.setComment(cursor.getString( columnIndex: 2));
            income.setDate(cursor.getString( columnIndex: 3));
            incomeList.add(income);
        } while (cursor.moveToNext());
    }
    cursor.close();
    return incomeList;
}
3 usages  ustim
public double getTotalBalance() {
    SQLiteDatabase db = this.getReadableDatabase();
    double totalCashles = getTotalCashles();
    double totalCash = getTotalCash();
    double total = totalCashles + totalCash;
    ContentValues values = new ContentValues();
    values.put(COLUMN_AMOUNT, total);
    values.put(COLUMN_COMMENT, "Total balance");
    String date = getCurrentDate();
    values.put("date", date);
    return total;
}
1 usage  ustim
public double getBalance(String tableName) {
    SQLiteDatabase db = this.getReadableDatabase();
    double balance = 0;
    Cursor cursor = db.rawQuery( sql: "SELECT SUM(" + COLUMN_AMOUNT + ") FROM " + tableName,  selectionArgs: null);
    if (cursor.moveToFirst()) {
        balance = cursor.getDouble( columnIndex: 0);
    }
    cursor.close();
    return balance;
}

1 usage  ustim
private double getTotalCashles() {
    SQLiteDatabase db = this.getReadableDatabase();
    String selectQuery = "SELECT SUM(" + COLUMN_AMOUNT + ") as total FROM " + TABLE_INCOME_CASHLES;
```

```
private double getTotalCashles() {
    SQLiteDatabase db = this.getReadableDatabase();
    String selectQuery = "SELECT SUM(" + COLUMN_AMOUNT + ") as total FROM " + TABLE_INCOME_CASHLES;
    Cursor cursor = db.rawQuery(selectQuery,  selectionArgs: null);
    double total = 0;
    if (cursor.moveToFirst()) {
        int totalIndex = cursor.getColumnIndex( columnName: "total");
        if (totalIndex != -1) {
            total = cursor.getDouble(totalIndex);
        }
    }
    cursor.close();
    return total;
}

1 usage   ustim
private double getTotalCash() {
    SQLiteDatabase db = this.getReadableDatabase();
    String selectQuery = "SELECT SUM(" + COLUMN_AMOUNT + ") as total FROM " + TABLE_INCOME_CASH;
    Cursor cursor = db.rawQuery(selectQuery,  selectionArgs: null);
    double total = 0;
    if (cursor.moveToFirst()) {
        int totalIndex = cursor.getColumnIndex( columnName: "total");
        if (totalIndex != -1) {
            total = cursor.getDouble(totalIndex);
        }
    }
    cursor.close();
    return total;
}

4 usages   ustim *
protected void updateBalance(double amount) {
    SQLiteDatabase db = this.getWritableDatabase();
    double currentBalance = getTotalBalance();
    double newBalance = currentBalance + amount;
    ContentValues values = new ContentValues();
    values.put("balance", newBalance);
    db.update( table: "user_balance", values,  whereClause: null,  whereArgs: null);
    db.close();
    Log.d( tag: "DatabaseHelper",  msg: "Updated balance: " + newBalance);
}
```

```java
public double getCashIncome() {
    SQLiteDatabase db = this.getReadableDatabase();
    String query = "SELECT SUM(" + COLUMN_AMOUNT + ") FROM " + TABLE_INCOME_CASH;
    Cursor cursor = db.rawQuery(query, selectionArgs: null);
    double cashIncome = 0.0;
    if (cursor.moveToFirst()) {
        cashIncome = cursor.getDouble( columnIndex: 0);
    }
    cursor.close();
    return cashIncome;
}

1 usage  ustim
public double getCashlessIncome() {
    SQLiteDatabase db = this.getReadableDatabase();
    String query = "SELECT SUM(" + COLUMN_AMOUNT + ") FROM " + TABLE_INCOME_CASHLES;
    Cursor cursor = db.rawQuery(query, selectionArgs: null);
    double cashlessIncome = 0.0;
    if (cursor.moveToFirst()) {
        cashlessIncome = cursor.getDouble( columnIndex: 0);
    }
    cursor.close();
    return cashlessIncome;
}
6 usages  ustim
public void decreaseBalance(double amount) {
    SQLiteDatabase db = this.getWritableDatabase();
    double currentBalance = getTotalBalance();
    double newBalance = currentBalance - amount;
    ContentValues values = new ContentValues();
    values.put(COLUMN_AMOUNT, newBalance);
    values.put(COLUMN_COMMENT, "Updated balance");
    String date = getCurrentDate();
    values.put("date", date);
    db.insert(TABLE_TOTAL, nullColumnHack: null, values);
    db.close();
}
1 usage  ustim
public double getCurrentMonthExpenses() {
    SQLiteDatabase db = this.getReadableDatabase();
    String query = "SELECT SUM(" + COLUMN_AMOUNT + ") FROM " + TABLE_TOTAL_EXPENSES + " WHERE strftime('%m', date) = strftime('%m', 'now')";
    Cursor cursor = db.rawQuery(query, selectionArgs: null);
```

```
    double expenses = 0.0;
    if (cursor.moveToFirst()) {
        expenses = cursor.getDouble( columnIndex: 0);
    }
    cursor.close();
    return expenses;
}

1 usage  ustim
public double getCurrentMonthIncome() {
    SQLiteDatabase db = this.getReadableDatabase();
    String query = "SELECT SUM(" + COLUMN_AMOUNT + ") FROM " + TABLE_TOTAL_INCOME + " WHERE strftime('%m', date) = strftime('%m', 'now')";
    Cursor cursor = db.rawQuery(query,  selectionArgs: null);
    double income = 0.0;
    if (cursor.moveToFirst()) {
        income = cursor.getDouble( columnIndex: 0);
    }
    cursor.close();
    return income;
}
1 usage  ustim
public double getExpenseSumForCategory(String category) {
    SQLiteDatabase db = this.getReadableDatabase();
    String selectQuery = "SELECT SUM(" + COLUMN_AMOUNT + ") as total FROM " + category +
            " WHERE strftime('%Y-%m', date) = strftime('%Y-%m', 'now')";
    Cursor cursor = db.rawQuery(selectQuery,  selectionArgs: null);
    double total = 0;
    if (cursor.moveToFirst()) {
        int totalIndex = cursor.getColumnIndex( columnName: "total");
        if (totalIndex != -1) {
            total = cursor.getDouble(totalIndex);
        }
    }
    cursor.close();
    return total;
}
```