Галицький фаховий коледж імені В’ячеслава Чорновола

відділення комп’ютерних та видавничих технологій

циклова комісія інформатики та комп’ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач відділенням
комп’ютерних та видавничих
технологій

Чубей О.О. /_____/

підпис

«___»_____2022 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту

освітньо-кваліфікаційного рівня «молодший спеціаліст»

зі спеціальності 122 «Компютерні науки»

на тему: «Мобільний додаток автоматизації обліку сімейного бюджету»

Студент групи КН-41 Холод Ю.В _____
(підпис)

Керівник проекту Івасьєв С.В. _____
(підпис)

Консультанти:

з техніко-економічного
обґрунтування Меленчук Л.І. _____
(підпис)

нормо контролер Сиротюк Н.С _____
(підпис)

Тернопіль – 2022

Галицький фаховий коледж імені В’ячеслава Чорновола

відділення комп’ютерних та видавничих технологій

циклова комісія інформатики та комп`ютерних дисциплін

ЗАТВЕРДЖУЮ
Завідувач відділенням
комп’ютерних та видавничихтехнологій

Чубей О.О. /_____/
підпис
«___»_____2021 р.

ЗАВДАННЯ

на дипломне проектування

на здобуття освітньо-кваліфікаційного рівня «молодший спеціаліст»

студенту _____
(прізвище, ім’я та по-батькові студента)

1. Тема проєкту Мобільний додаток автоматизації обліку введення сімейного бюджету
затверджена наказом по коледжу від 01.10.2021 р., № 178-н
2. Термін здачі студентом завершеного проекту “_____” _____2022 р
3. Вихідні дані до проекту_____
_____
_____
_____
4. Перелік питань, які повинні бути розроблені в проекті: _____
а) основна частина _____
_____
_____
_____
б) техніко-економічного обґрунтування _____
_____
_____
_____
_____
5. Перелік графічного матеріалу_____
_____

6. Консультанти проекту: _____

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
з техніко-економічного обґрунтування	_____ (вчена ступень, звання П.І.Б. _____ консультанта)		

КАЛЕНДАРНИЙ ПЛАН
дипломного проектування

№ п/п	Найменування етапу	Терміни	
		початку	завершення
1.	Вибір теми, ознайомлення з вимогами до дипломного проектування.	20.09.21 р.	01.10.21 р.
2.	Огляд типових рішень та написання відповідного розділу ПЗ	06.12.21 р.	26.01.22 р.
3.	Дослідження технологій реалізації та написання відповідного розділу ПЗ	26.01.22 р.	14.02.22 р.
4.	Розробка функціональних вимог до проекту та робота над структурою програмного продукту. Написаннявідповідного розділу ПЗ	14.02.22 р.	02.03.22 р.
5.	Встановлення на налаштування середовища реалізації та написання відповідного розділу ПЗ	02.03.22 р.	16.03.22 р.
6.	Проектування програмного засобу (функціоналу, інтерфейсу, бази даних продукту) та написання відповідного розділу ПЗ	16.03.22 р.	17.04.22 р.
7.	Реалізація та налаштування програмного засобу та написання відповідного розділу ПЗ	17.04.22 р.	03.05.22 р.
8.	Доопрацювання модулів	05.05.22 р.	18.05.22 р.
9.	Тестування на налагодження програмного продукту та написання відповідного розділу ПЗ	18.05.22 р.	01.06.22 р.
10.	Опрацювання економічного розділу дипломного проекту та оформлення спеціального розділу	20.05.22 р.	05.06.22 р.
11.	Робота над оформленням пояснювальної записки	05.06.22 р.	12.06.22 р.
12.	Попередній захист дипломного проекту, доопрацювання	15.06.22 р.	15.06.22 р.
13.	Підготовка до захисту дипломного проекту	15.06.22 р.	22.06.22 р.
14.	Захист дипломного проекту	25.06.22 р.	26.06.22 р.

7. Дата видачі завдання ”____” _____ 2021 р.

Керівник _____ / _____

Завдання прийняв до виконання_____ / _____

Реферат

Дипломний проєкт. Тема: Мобільний додаток автоматизації обліку сімейного бюджету. 88 сторінок, 14 рисунків, 6 таблиці, 13 джерел.

Об’єктом дослідження є розробка додатків для мобільних платформ.

Методологічною основою роботи послужили загальнонаукові та спеціальні методи дослідження, зокрема, історичний, аналітичний, аналізу та синтезу, порівняння й інші. Історичний метод було застосовано для вивчення сутності захисту інформації та розвитку засобів захисту від несанкціонованого доступу. Теоретико-методологічною основою роботи є наукові концепції в сфері розробки програмного забезпечення. Аналітичні методи використано для аналізу існуючих мов програмування та засобів створення мобільних додатків. Метод порівняння застосовано при визначенні платформи для створення додатку, а також його основних модулів.

Завданням проєкту є розробка мобільного додатку для автоматизації обілку сімейного бюджету

Результат – мобільний додаток, що повністю готовий до експлутації.

Зручний та досить легкий додаток для швидкого та якісного обліку і ведення сімейного бюджету

Для розробки мобільного додатку було обрано мобільну платформу Android та середовище розробки Eclipse з орієнтацією на створення програмних додатків для пристроїв на базі операційної системи Android. Мовою на якій буде реалізовано програмний продукт буде Java.

ECLIPSE, JAVA, VISUAL STUDIO CODE, PROGREMM, MOBILE APPLICATION.

Abstract

Diploma project. Topic: Mobile application for automation of family budget accounting. 88 pages, 14 figures, 6 tables, 13 sources.

The object of research is the development of applications for mobile platforms.

The methodological basis of the work were general and special research methods, in particular, historical, analytical, analysis and synthesis, comparison and others. The historical method was used to study the essence of information security and the development of means of protection against unauthorized access. Theoretical and methodological basis of the work are scientific concepts in the field of software development. Analytical methods were used to analyze existing programming languages and tools for creating mobile applications. The comparison method was used to determine the platform for creating the application, as well as its main modules.

The task of the project is to develop a mobile application for automation of the family budget

The result is a mobile application that is completely ready for use.

Convenient and fairly easy application for fast and high-quality accounting and family budgeting

For the development of the mobile application, the Android mobile platform and the Eclipse development environment with a focus on creating software applications for devices based on the Android operating system were chosen. The language in which the software will be implemented will be Java.

ECLIPSE, JAVA, VISUAL STUDIO CODE,PROGRAMMING, MOBILE APPLICATION.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЗМІ – засоби масової інформації
ПЗ – програмне забезпечення
БД – база даних
UML – діаграма в мові моделювання
КТЗ – комплекс технічних засобів
ЗП – заробітна плата

ВСТУП

Актуальність теми**.** В наш час виникає необхідність у економному витрачанні коштів на основні потреби людини та родини, тому часто в родинах проводиться облік доходів та витрат, коштів на основні виплати, можливості заощадження чи придбання нової техніки, раціональна закупівля продуктів та побутової хімії тощо. Кожна родина сама обирає метод обліку – запис у зошиті, табличка в персональному комп'ютері чи більш сучасні варіанти. Оскільки майже в кожній родині у працездатних членів, що відповідають за бюджет, є смартфони, то актуальною є розробка програмного забезпечення по обліку веденню сімейного бюджету саме на мобільній платформі.

Метою роботи є обґрунтування та розробка мобільного додатку для обліку сімейного бюджету

Виходячи з мети, перед нами постають наступні завдання:

– здійснити аналіз предметної області та постановку завдань по розробці мобільного додатку автоматизації обліку сімейного бюджету;

– провести проектування системи для мобільного додатку автоматизації обліку сімейного бюджету;

– розробити реалізацію та провести тестування системи для мобільного додатку автоматизації обліку сімейного бюджету;

– надати техніко-економічне обґрунтування та дослідити економічну ефективність розробки.

Предметом дослідження є створення мобільних додатків для системи Android.

Об'єктом дослідження є розробка додатків для мобільних платформ.

Методологічною основою роботи послужили загальнонаукові та спеціальні методи дослідження, зокрема, історичний, аналітичний, аналізу та синтезу, порівняння й інші. Історичний метод було застосовано для вивчення

сутності захисту інформації та розвитку засобів захисту від несанкціонованого доступу. Теоретико-методологічною основою роботи є наукові концепції в сфері розробки програмного забезпечення. Аналітичні методи використано для аналізу існуючих мов програмування та засобів створення мобільних додатків. Метод порівняння застосовано при визначенні платформи для створення додатку, а також його основних модулів.

Структура роботи**.** Кваліфікаційна робота має в своєму складі вступ, чотири основні розділи з підрозділами, висновки та список використаних джерел, який містить 13 елементів. Загальний обсяг роботи складає 88 сторінок

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ ПО РОЗРОБЦІ МОБІЛЬНОГО ДОДАТКУ АВТОМАТИЗАЦІЇ ОБЛІКУ СІМЕЙНОГО БЮДЖЕТУ

1.1 Сутність мобільних додатків, їх призначення

Одним з найбільш перспективних та таких, що активно розвиваються на сьогоднішній день є ринок мобільних технологій. Саме вони у поєднанні з бездротовими мережами, можуть мати значний потенціал розширення часу, місця і ефективності роботи. Мобільні технології відкривають нові канали зв'язку між народами і урядами, нові можливості для обліку та введеня бізнесу та інше, потенційно пропонуючи більш широкий доступ до суспільної інформації і основних послуг. Ніяка інша технологія не була в руках такої значної кількості людей у багатьох країнах в такий короткий період часу. Останні оцінки показують, що інформаційно-комунікаційні технології можуть бути доступні кожному у 2022 році [9].

В даний час існує велика безліч різних практик і методологій управління проектами. Дані технології необхідні організаціям для успішної реалізації поставлених завдань, а також для виживання організації в принципі. На сьогоднішній день ринок мобільних додатків бурхливо розвивається. І в зв'язку з цим з'являється безліч проектів з розробки додатків на мобільних платформах. Це або компанії, що створюють ігри, або ЗМІ чи новинні сайти, пошукові ресурси, або незалежні команди, що реалізують один проект, великі організації, що створюють корпоративні додатки.

В результаті такого стрімкого розвитку, мобільні технології починають чинити значний вплив на людський розвиток в цілому та різні сфери діяльності, такі як охорона здоров'я, освіта, сільське господарство, зайнятість, запобігання криз, навколишнього середовища та інші. Саме з цих причин щорічно з'являється велика кількість різних мобільних додатків,

які пропонуються не лише споживчому ринку, але й на ринку масової інформації.

Світ мобільної розробки почав бурхливо розвиватися в 2008 році. Цьому послужили такі події як поява IPhone з IOS, а також App Store. Дані події підштовхнули розробників до переосмислення призначення мобільних додатків. Відкрилися нові перспективи і можливості. Компанії почали боротися, щоб зайняти певну нішу. Пізніше з'явилися нові мобільні операційні системи, що надало ще більше можливостей, з'явилися нові майданчики для реалізації проектів. Доступність і простота даних майданчиків дозволила будь-яким розробникам реалізувати свої ідеї. І тепер в основному необхідні тільки команда і невеликі кошти, щоб придбати середовища для розробки.

Оскільки мобільні додатки є відносно новими засобами використання мобільних телефонів та можуть застосовуватись тільки на досить нових пристроях під управлінням відповідних мобільних операційних систем, то ступінь дослідженості проблематики використання мобільних додатків є невисокою, і більшість досліджень спрямовані на проблеми розробки, а не використання мобільних додатків. З користувацької точки зору функціональність та зручність, а також контент окремих мобільних додатків є майже не дослідженим, зокрема, існують тільки дослідження використання мобільних додатків у бізнес-середовищі, для мобайл-банкінгу тощо. Також проводились дослідження з використання мобільних додатків для освітніх потреб. Проте досліджень застосування мобільних додатків для ЗМІ нами не було віднайдено.

На сьогодні мобільні технології здатні значно підвищити ефективність роботи підприємства в різних сферах його господарювання. Однак не дивлячись на це, досі існує певний опір з боку споживачів-підприємств до використання мобільних технологій в своїй діяльності. Це пов'язано з рядом причин, основними з яких є відсутність чіткої інформації про напрями

використання мобільних додатків та їх ефективність та небажання персоналу перенавчатись. Однак нівелювати ці проблеми можна значним чином за допомогою чіткої комплексної програми просування мобільних технологій на ринку, яка міститиме як елементи поінформованості так і стимулювання збуту [9].

Що стосується українських досліджень мобільних додатків, то тут доцільно відмітити дослідження:

– Лубко Д.В., який досліджував методологію проектування та інструментарій для створення мобільних додатків в залежності від операційної системи, для якої додаток розроблено;

– Богатирьової О.М., яка аналізувала можливості використання мобільних додатків для ритейлерів;

– Грабар О.І., яка вивчала особливості створення програмного забезпечення для web-додатків для мобільних телефонів в Україні, що є цікавим з точки зору ретрансляції web-версій ЗМІ у мобільний контент;

– Вахріної В.А., яка детально досліджувала проблематику проектування та розробка мобільного додатку для людей з вадами зору, що може бути також цікавим для створення модифікацій мобільних додатків ЗМІ для людей з обмеженими можливостями.

Розробка мобільних додатків відіграє все більш важливу роль для організацій, яким необхідно спілкування зі співробітниками або клієнтами за допомогою вбудованих додатків. На сьогоднішній день існує великий вибір мов програмування для розробки мобільних додатків. Це пов’язано з тим, що для різних мобільних пристроїв доводиться використовувати різні мови програмування, що обумовлене тим, що мобільні пристрої мають різні операційні системи (ОС). Цільова платформа (або платформи) – iOS, Android, Windows Phone, BlackBerry – буде мати значний вплив на мову розробки, яка буде використовуватися.

Наприклад, можна розробляти рідні додатки для кожної платформи або використовувати сторонній інструмент для оптимізації своїх додатків на різних платформах. Другий підхід може заощадити час і зусилля, хоча це може вплинути на зручність використання. Сучасні мобільні пристрої пропонують широкий спектр варіантів розробки [12; с. 117].

Мобільний додаток (або програми) являє собою програмне забезпечення, призначене для роботи на смартфонах, планшетних комп'ютерах та інших мобільних пристроях. Вони, як правило, доступні через сервіси розповсюдження, унікальні для кожного виробника операційної системи, такі як Apple AppStore, Google Play, Windows Phone Store і BlackBerry App Store. Деякі програми можна встановити безкоштовно, а інші потрібно придбати. Як правило, вони завантажуються з платформи відразу на пристрій, такі як iPhone, BlackBerry, Android телефон або Windows Phone, але іноді вони можуть бути завантажені на ноутбуки або настільні комп'ютери, і вже потім встановлені на мобільний пристрій. Прибуток від купленої програми розподіляється так: 20-30% йде поширювачу послуг (таких як iTunes, GooglePlay тощо), а решта йде виробнику програми.

Мобільні додатки спочатку виконували функції для загальної продуктивності і пошуку інформації, включаючи електронну пошту, календар, контакти та інформацію про погоду. Тим не менше, попит населення і наявність інструментів розробника спричинили швидке розширення в інших категоріях, таких як мобільні ігри, автоматизація виробництва, банківська справа, відстеження замовлень, покупки квитків і огляд останніх новин. Вибух в кількості і різноманітності додатків зробили відкриття, яке у свою чергу призвело до створення широкого спектру оглядів, рекомендацій, а також керування джерел, включаючи блоги, журнали, присвячені онлайн-додаткам і відкриттям послуг.

Популярність мобільних додатків продовжує рости, оскільки їх використання стає все більш поширеними серед всіх користувачів мобільних телефонів.

Мобільний додаток – програмний продукт, призначений для роботи на мобільних пристроях (смартфонах, планшетних комп'ютерах). Економіка додатків (App economy) – сукупність суспільно-виробничих відносин, пов'язана з розробкою та використанням мобільних додатків.

Сьогодні кожна компанія чи фірма, котра «йде в ногу з часом», розуміє, що створити хороший веб-сайт чи надрукувати якісну поліграфію – недостатньо. В час, коли більшість людей мають під рукою свій портативний мікросвіт у смартфоні, єдине рішення для тих, хто хоче бути серед лідерів – створити якісні та зручні мобільні додатки.

На сьогодні компанії-розробнику мобільних додатків необхідно шукати способи створення продуктів для максимальної кількості мобільних платформ за мінімальною ціною та з мінімальною витратою часу.

Ринок диктує жорсткі умови – мобільні додатки повинні запускатися на максимальній кількості платформ, коштувати мінімально та бути готовими до впровадження за мінімальний термін [12, с. 118].

Процес розробки програмного забезпечення для мобільних пристроїв вимагає підтримки певних обмежень що до специфіки роботи додатків у мобільних операційних системах. Особливості роботи мобільних додатків стосуються перш за все:

– витрат живлення акумуляторної батареї мобільного пристрою;

– обмеження на кількість даних, що передаються через мережу Інтернет;

– зменшеної швидкість передачі пакетів в мобільному Інтернет з'єднанні

– можливості втрати пакетів при передачі по мобільному Інтернет з'єднанню;

– кількості обмінів даними із зовнішніми пристроями на Bluetooth;

– безпеки даних користувача;

– значної фрагментації версій операційних систем та фреймворків;

– великої різноманітності розмірів екранів мобільних пристроїв;

– обмеження запитів до системи визначення місцезнаходження абонента;

– обмеження на розмір файлу додатку;

– порівняно невеликого ліміту оперативної пам’яті мобільного пристрою;

– порівняно обмеженої швидкості передачі даних в мобільному Інтернет з’єднанні [10; с. 72].

Якщо говорити про мобільні додатки, що використовуються для обліку бізнесу, то сфера їх використання відрізняється від сфери звичайних споживачів. Також, можливості від їх впровадження представлено на рисунку 1.1 [9].

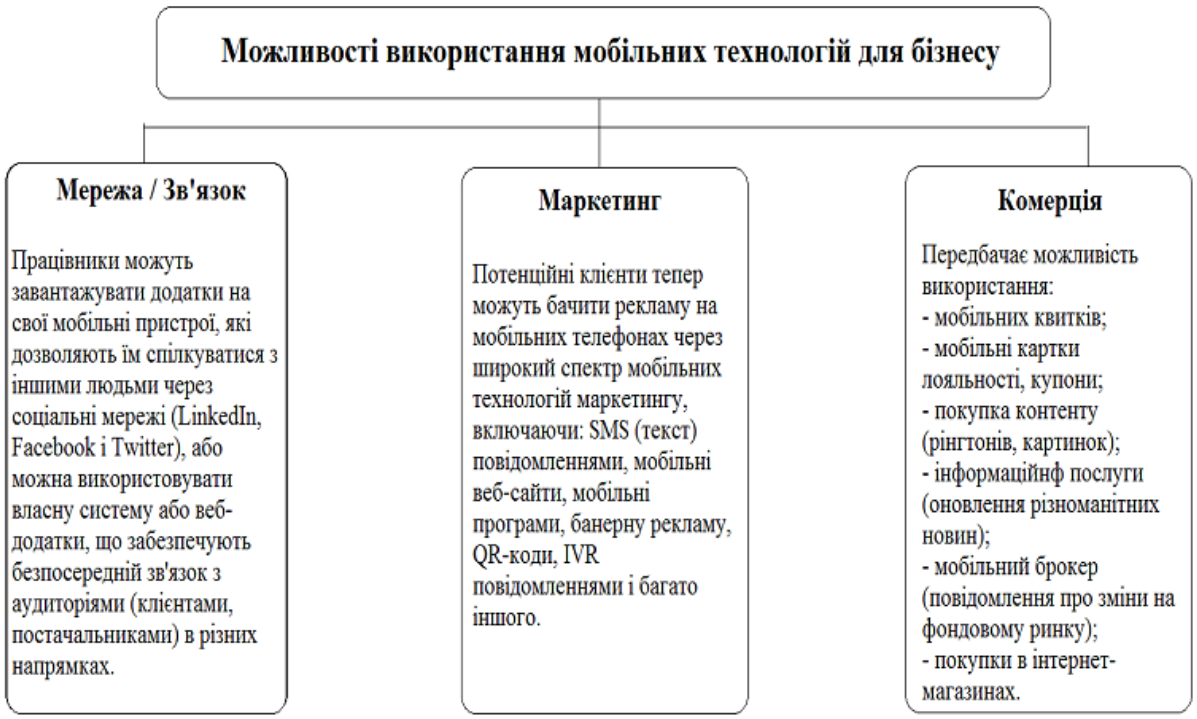

Рисунок 1.1 – Можливості використання мобільних додатків для бізнесу

Нові мобільні додатки для Android було розроблено, щоб зробити читання газети за допомогою електронних пристроїв дуже простим. Ця програма є ексклюзивною для мобільних користувачів Android. Вона поставляється в новому розмірі таким чином, що додаток обробляє вміст із друкованих статей, використовуючи смартфон.

Читачі можуть завантажити мобільний додаток для читання і миттєво проглядати відео, слайд-шоу та інші цікаві контенти з газети. Мобільний додаток для читання газети використовує камеру телефону, щоб розпізнати цифрові водяні знаки, які були вбудовані в друковані зображення. Він забезпечує і надає інтерактивний, інформативний багатовимірний огляд із статичних сторінок газети. Ця програма дає можливість масовій інформації перейти в нову еру інтеграції друкованих ЗМІ в електронні засоби. Додаток значною мірою підвищить читацький досвід користувачів за допомогою відео в режимі реального часу та інформації, що дає освіту і задоволення. Рекламодавці також виграють, оскільки вони мають новий засіб для реклами.

Розробка мобільних додатків – складний та багатоетапний процес. Для того, щоб розпочати даний процес потрібно розмежувати потреби, які мають користувачі і клієнти, функції, які цей додаток повинен виконувати, а також визначитись з типом та мобільною платформою додатку.

Розробка мобільних додатків складається шести основних етапів, кожен із яких надзвичайно важливий, адже це надзвичайно вартісна річ, тому вкладення коштів потребує чіткого формулювання потреб. Отож, перше що потрібно зробити для розробки мобільного додатку – це обрати тип додатку та операційну систему для його реалізації.

Залежно від того, які саме функції будуть покладатися на мобільний додаток, залежить тривалість його розробки, рівень складності і, звичайно, ціна. Загалом виділяють чотири основних типи мобільних додатків:

1. Корпоративні. Вони створені для спрощення роботи компанії, швидкої передачі даних між працівниками та отримання корпоративної

інформації. Цільовою аудиторією таких мобільних додатків є, перш за все, працівники компанії, а також реальні та потенційні клієнти та партнери. Розробка мобільних додатків такого типу простіша, для їх дизайну використовуються кольори та елементи корпоративного стилю компанії.

2. Контентні. Це мобільні додатки, які спеціалізуються на наданні різних типів інформації. Вона може бути текстова або ж у відео чи аудіоформаті. В основному, розробка таких мобільних додатків здійснюється для засобів масової інформації, радіо, телеканалів чи порталів.

3. Сервісні. Їх метою є надання певних сервісних послуг, тобто виконання задач в реальному часі, які ставить користувач. Розробка мобільних додатків такого типу – складна, адже вони повинні працювати, починаючи від калькулятора чи будильника і закінчуючи програмами для роботи з великими об’ємами тексту або графіки.

4. Ігрові. Основною метою даних мобільних додатків є розваги, але ще одну важливу функцію покладають на них – вдале розміщення в них реклами, і це є обов’язковою умовою.

У травні 2021 року дослідження провідних ЗМІ показало, що в попередньому кварталі, більше користувачів мобільних пристроїв віддають перевагу саме додаткам, ніж переглядам веб-сторінок на своїх гаджетах: 51,1% проти 49,8% відповідно. Дослідники виявили, що використання мобільних додатків сильно корелює з призначеним для користувача контекстом і залежить від місця розташування користувача і часу доби.

В Україні до кінця 2021 року поширення смартфонів серед абонентів мобільного зв’язку у віці від 18 до 50 років досягло близько 70%. Про це свідчать результати дослідження Київського міжнародного інституту соціології (КМІС) [13].

На сьогоднішній день сенсорними смартфонами користуються 41% українців у віці 18-50 років. При цьому серед молодих людей у віці 18-30 років ця цифра становить 59%.

Більшість покупців смартфонів (57%) вважають за краще операційну систему Android. Поширення інших систем не перевищує 15% – по стільки мають iOS і Symbian.

Серед усіх власників смартфонів 72,5% скачують через Інтернет мобільні додатки. Найбільш популярними є соціальні мережі (82% скачали додатків), ігри (52%), програми навігації (41%).

Для того, щоб розробка мобільних додатків була найбільш ефективною слід заздалегідь обрати операційну систему.

Отож вибір має початися із дослідження цільової аудиторії (визначення які платформі користувачі віддають перевагу), інформацію для цього можна отримати із даних, які надає Google Analytics, що підключений до веб-сайту, і показує, за допомогою яких пристроїв та браузерів клієнти найчастіше заходять на певний сайт.

Також можна скористатися загальновідомими статистичними даними. Наприклад, IOS – операційна система для Apple iPhone, iPad, а також iPod Touch. App Store пропонує понад півмільйона мобільних додатків, загальна кількість скачувань яких вже перевищила 25 мільярдів.

Android – гнучка операційна система для смартфонів та планшетів. Техніка на базі Android зараз надзвичайно популярна, тож розробка мобільних додатків, розміщених на Google Play, користується особливою популярністю.

Таким чином, мобільні додатки мають велику популярність, все більша кількість послуг та процесів переведена в мобільний формат – банківські додатки для контролю фінансів, додатки «Хелсі» тощо для отримання медичних послуг, додаток «Дія» для роботи з персональними даними, додатки служб таксі, замовлення ліків, програм лояльності супермаркетів тощо. Всі ці додатки дозволяють спланувати свій час та витрати, тому на основі основних позицій, які надають існуючі додатки, можна сформувати вимоги до додатку по плануванню сімейного бюджету з врахуванням даних

від банків про наявні кошти та час їх надходження, від додатків супермаркетів про вартість продуктів та товарів першої необхідності тощо. Тому маємо спроектувати мобільний додаток для автоматизації обліку сімейного бюджету, який допоможе раціонально розподілити сімейні кошти, не забути вчасно сплатити комунальні платежі, поповнити мобільні телефони та розрахуватись за послуги Інтернет-провайдера. В розширеній версії додаток може містити нагадування про потребу замовити ліки, сплатити за автостоянку чи поставити інші важливі нагадування.

1.2 Постановка завдань по розробці мобільного додатку

Призначення задачі: автоматизувати обліку сімейного бюджету.

У рамках даної роботи необхідно обґрунтувати мету розробки задачі «Автоматизація обліку сімейного бюджету», обґрунтувати вибрані елементи інформаційних технологій для використання під час розробки задачі, обґрунтувати обране технічне забезпечення, спроектувати та розробити рішення для програмного, математичного та інформаційного забезпечень задачі, створити методичні рекомендації використання задачі, обрати та обґрунтувати вибір рішень захисту інформації.

Для розробки рішень елементів програмного забезпечення (ПЗ) необхідно обрати та обґрунтувати вибір середовища для розробки, технологій розробки ПЗ, архітектури ПЗ, мови програмування для створення ПЗ, способів вводу та виведення інформації через ПЗ.

Для розробки рішень елементів інформаційного забезпечення необхідно обрати та обґрунтувати вибір технології для створення та управління базами даних, а також розробити логічну та фізичну схеми даних для задачі, що розробляється.

Для розробки рішень елементів математичного забезпечення задачі «Автоматизація обліку сімейного бюджету» необхідно обрати та обґрунтувати вибір інструментів для розробки схеми роботи задачі та

розробити схему задачі «Автоматизація обліку сімейного бюджету» за допомогою обраної технології.

Для вибору елементів технічного забезпечення необхідно проаналізувати фізичні пристрої (смартфони) та їх технічні характеристики, що необхідні для нормальної роботи додатку «Автоматизація обліку сімейного бюджету» при якому враховуються та виконуються вимоги для роботи програмного, математичного та інформаційного забезпечень задачі «Автоматизація обліку сімейного бюджету», а також обрати та обґрунтувати вибір особливостей пристроїв для задачі.

Необхідно розробити методичні рекомендації для задачі. Це передбачає розробку рекомендацій, що необхідні для впровадження та використання задачі співробітниками організації.

Необхідно обрати рішення для інформаційного захисту задачі «Автоматизація обліку сімейного бюджету». Це передбачає вибір та обґрунтування вибору комплексу заходів та технологій, що необхідні для захисту від несанкціонованого доступу, спотворення, викрадення або втрати інформації, та контролю доступу до задачі.

Модульний принцип побудови припускає як ізольоване використання окремих програмних модулів, так і довільні комбінації їх, залежно від користувацької необхідності.

Модуль «Доходи» повністю автоматизує введення даних по заробітній платі і дозволяє зчитувати повідомлення з банку про надходження, а також дозволяє ручне введення даних за потреби (надходження готівкових коштів).

Модуль «Витрати» дозволяє отримувати дані з банківських виписок за MCC-кодом про тип витрат (за потреби може бути коригованим вручну, оскільки в ряді супермаркетів родина купляє різні види товарів, які відповідно відносяться до різних статей).

При розробці додатку мають бути реалізовані два основних принципи:

– універсальність – можливість використання в будь-яких родинах,

починаючи від однієї особи і до 10 чоловік, з будь-яким режимом надходжень та витрат – регулярні, разові, обмежені цільовим призначенням тощо;

– адаптованість – забезпечення можливості користувачеві самостійно провести настроювання з урахуванням специфіки конкретного домогосподарства і змін у статтях.

Додаток повинен бути простим і зрозумілим, не варто відразу розробляти масу функцій і пропонувати все – аудиторія сама підкаже, що треба доробити й дописати. Після запуску слід не просто рахувати кількість завантажень, але й підключати аналітику – аналізувати, які функції і як часто використовують з мобільного.

Зараз багато мобільних додатків дає можливість надавати інформацію іншим додаткам, що полегшить обмін інформацією. Серед інших можливостей: функція гарячої новини (за бажанням користувач може отримувати сповіщення при перевищенні планових витрат над надходженнями, зміни статей витрат іншим користувачем при сімейному доступі до додатку тощо), можливість самостійно формувати перелік витрат (тобто обирати категорію витрат і формувати відсоткову частку чи фіксовану суму на кожний тип витрат).

2 ПРОЕКТУВАННЯ СИСТЕМИ ДЛЯ МОБІЛЬНОГО ДОДАТКУ АВТОМАТИЗАЦІЇ ОБЛІКУ СІМЕЙНОГО БЮДЖЕТУ

2.1 Опис архітектури системи

Система матиме наступні базові функції:

– Перегляд рахунків та історії операцій.

– Запис поточних витрат, доходів та майбутніх надходжень і витрат.

– Робота без Internet (оффлайн).

– Синхронізація з хмарним сховищем при наявності Internet підключення.

Проектуємо архітектуру баз даних (БД). Всі архітектури представляються нотацією UML і при потребі, доповнюються текстовими описами.

Під час проектування системи аналізу фінансового стану підприємства доцільно звернутись до такого інструменту, як UML-діаграми. Для моделювання поведінки окремих показників діяльності підприємства на логічному рівні в мові UML можна використовувати декілька канонічних діаграм станів, діяльності, послідовностей та співробітництва одночасно, кожна з яких зосереджується на певному аспекті системи. На відміну від інших діаграм, діаграма станів описує процес зміни стану лише одного класу, а точніше одного екземпляра даного класу, тобто моделює всі можливі зміни стану конкретного об’єкта. Таким чином, зміна стану об'єкта може бути викликана зовнішніми впливами з боку інших об'єктів або ззовні. Це для опису реакції об’єкта на такі зовнішні впливи та використання діаграм станів.

Основна мета діаграми станів — описати можливі послідовності станів і переходів, які разом характеризують поведінку змодельованої системи протягом її життєвого циклу. Діаграма стану описує поведінку елементів на

основі їх реакцій на сприйняття конкретних подій, наприклад, зміну показника рентабельності в залежності від запланованого рівня витрат на рекламу чи фонду оплати праці. Системи, які реагують на користувачів або зовнішні дії іншої системи, називаються реактивними. Якщо такі дії ініціюються в будь-який випадковий момент часу, говорять про асинхронну поведінку моделі.

Хоча діаграми станів найчастіше використовуються для опису поведінки окремих екземплярів класів, їх також можна використовувати для визначення функціональності інших компонентів моделі, таких як варіанти використання, дійові особи, підсистеми, операції та методи. Діаграма стану, по суті, є особливим видом графіка, який представляє автомат. Поняття автоматів у контексті UML має дуже специфічну семантику, засновану на теорії автоматів. Вершинами цього графа є стани та деякі інші типи елементів автомата (псевдостани), які зображуються відповідними графічними символами. Дуги графіка використовуються для позначення переходів із стану в стан. Діаграми станів можуть бути вкладені для створення вкладених діаграм більш детального представлення окремих елементів моделі [**Error! Reference source not found.**].

Через своє призначення діаграма стану не є обов’язковою для представлення моделі.

Вибираючи стани та переходи, слід пам’ятати, що час роботи окремих переходів має бути значно коротшим, ніж знаходження елементів в окремих станах – інакше, звичайно, під час переходу елемент потрапляє в якийсь неурегульований стан. Кожна із держав повинна мати певну стабільність у часі.

При розробці діаграми станів необхідно постійно стежити за тим, щоб об’єкт в будь-який момент міг перебувати тільки в одному стані. Якщо це не так, то ця обставина може бути як наслідком помилки, так і неявною ознакою наявності паралельної поведінки об’єкта, що моделюється.

Необхідно провести обов’язкову перевірку, щоб не працювали два переходи з одного стану одночасно. Іншими словами, необхідно виконати вимогу неконфліктності для всіх переходів, що виходять з одного стану.

Використання історичних станів виправдано лише в тому випадку, якщо необхідно організувати обробку виняткових ситуацій (переривання) без втрати даних або виконаної роботи. Водночас слід бути обережним з історичними станами, особливо глибокими. Варто звернути увагу, що кожен з кінцевих підавтоматів може мати лише одну історичну умову. В іншому випадку можливі помилки, особливо коли підавтомати показані на окремих схемах стану [2; с. 73].

Для візуалізації діаграм станів можна використовувати мережеві операційні системи, які можуть надати клієнту доступ до сервера збережених даних, на якому побудована діаграма.

Діаграма станів додатку наведена на рисунку 2.1.

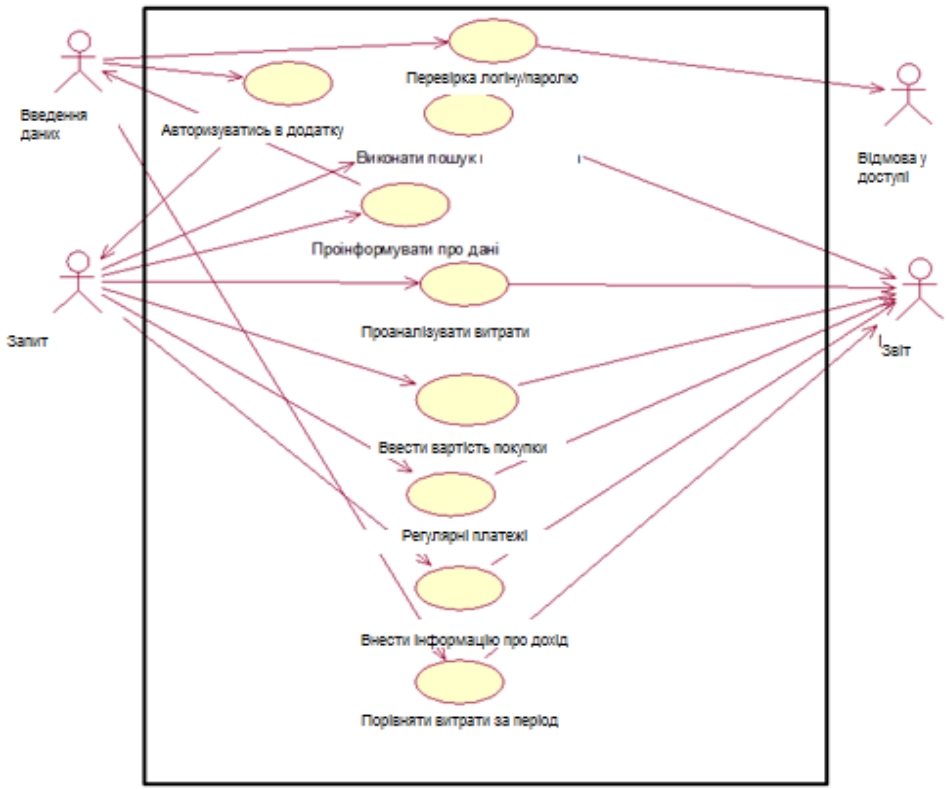

Рисунок 2.1 – Діаграма станів системи

Діаграма розгортання призначена для візуалізації елементів і компонентів програми, які існують лише на етапі її виконання. При цьому подаються тільки компоненти-екземпляри програми, які є виконавчими файлами або динамічними бібліотеками.

Діаграма розгортання містить графічні зображення процесорів, пристроїв, процесів і зв'язків між ними. На відміну від діаграм логічного відображення, діаграма розгортання є єдиною для системи в цілому, оскільки повинна повністю відображати особливості її реалізації. Ця діаграма, по суті, завершує процес проектування для конкретної програмної системи і її розробка, як правило, є останнім етапом специфікації моделі. Основні цілі розробки діаграми розгортання:

– визначити розподіл компонентів системи по її фізичних вузлах;

– показати фізичні зв'язки між всіма вузлами реалізації системи на етапі її виконання;

– виявити вузькі місця системи і реконфігурувати її топологію для досягнення необхідної продуктивності.

Дана діаграма зображена на рисунку.2.2.

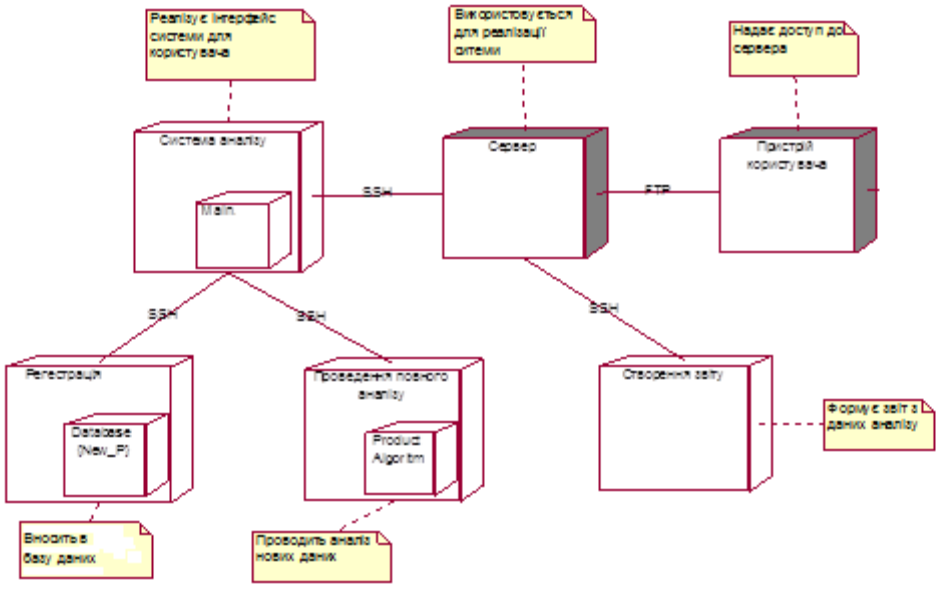

Рисунок 2.2 – Діаграма розгортання

В ході проектування архітектури БД, було обрано для її відтворення реляційну модель даних та виділено наступні відношення, що відображені на рисунку 2.3-2.7.

Структура бази даних користувачів, що містить інформацію про зареєстровані логіни та відповідні їм паролі, наведена на рисунку 2.3.

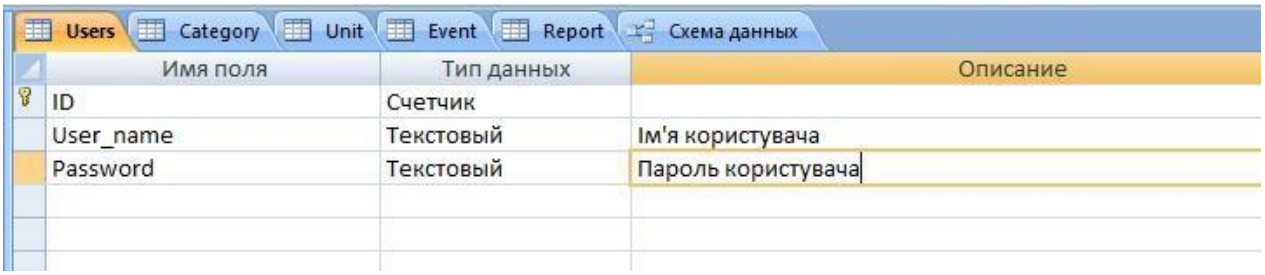

Рисунок 2.3 – Структура бази даних «Users»

Структура бази даних категорій, які вводяться користувачем з пропонованого списку чи самостійно, наведена на рисунку 2.4. Дана база містить перелік категорій у довільному текстовому форматі.

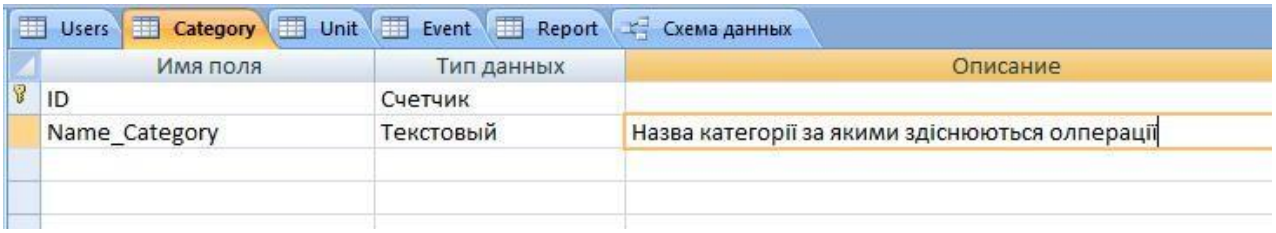

Рисунок 2.4 – Структура бази даних «Category»

База даних подій, на основі якої може форФУмуватись той чи інший звіт, містить дату, тип події, суму та категорію (рисунку. 2.5.)

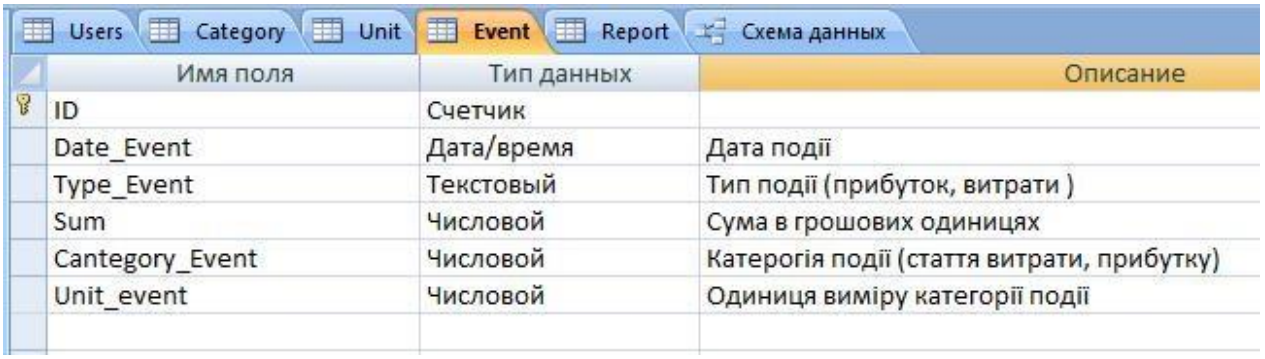

Рисунок 2.5 – Структура бази даних «Event»

Звіти можуть бути сформовані за будь-якою категорією чи декількома категоріями, за всіма видами операцій чи окремо за витратами, доходами тощо. Структура бази даних наведена на рисунку 2.6.

Users | Category | Unit | Event | Report | Схема данных

Имя поля	Тип данных	Описание
ID	Счетчик	
Date_Start	Дата/время	Дата початку формування звіту
Date_end	Дата/время	Дата кінця формування звіту
Category	Числовой	Категорія за якою формується звіт
Income	Денежный	Прибутки за категорією
Costs	Денежный	Витрати за категорією
Balance	Денежный	Баланс за категорією
User	Числовой	Користувач за яким формується звіт

Рисунок 2.6 – Структура звітів бази даних «Report»

Реляційна модель даних містить зв’язки типів «один до одного» та «один до багатьох). Діаграма реляційної моделі зображена на рисунку 2.7.

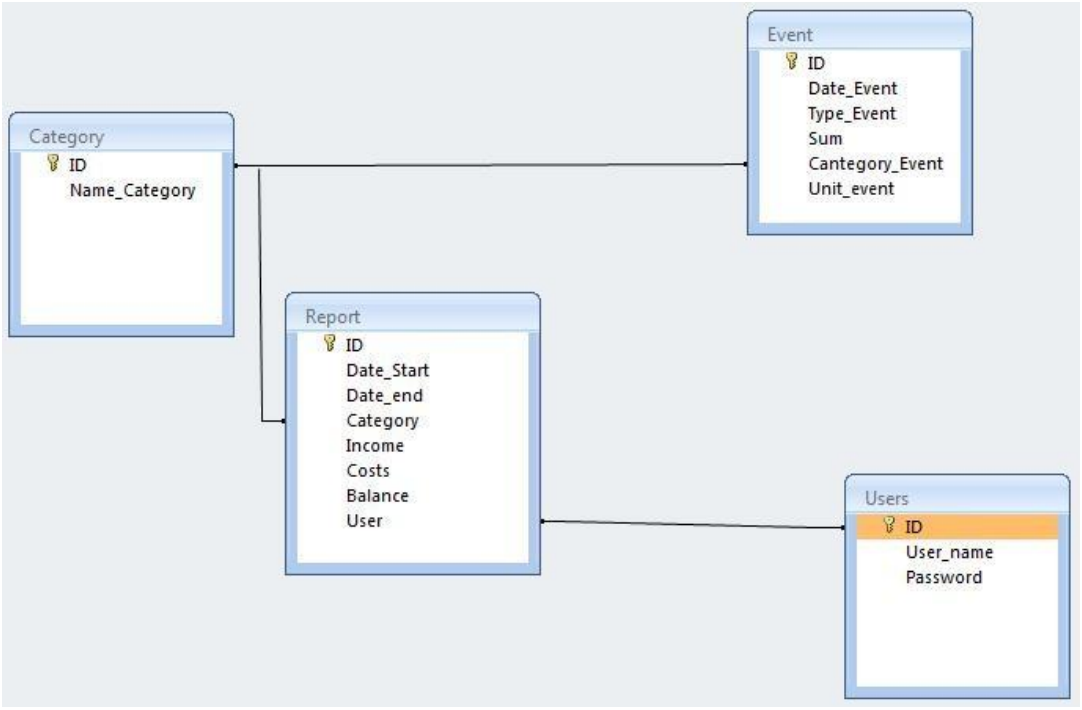

Рисунок 2.7 – Діаграма бази даних мобільного додатку автоматизації сімейного бюджету

Отже, ми розробили основні структурні елементи баз даних, на яких буде побудовано роботу проектованого додатку.

2.2 Опис комплексу технічних засобів

Для розробки самого програмного забезпечення може бути використано персональний комп’ютер чи інший пристрій, який підтримує

можливість створення віртуальної андроїд-системи для тестування розроблюваного додатка.

Вимоги до апаратної частини обчислювальної системи:

– шестиядерний процесор мобільного телефону з тактовою частотою не менше ніж 1.4 ГГц;

– оперативна пам’ять мобільного телефону об’ємом не менше ніж 4 Гб;

– жорсткий диск мобільного телефону об’ємом не менше ніж 16 Гб.

Комплексом технічних засобів (КТЗ) є сукупність технічних засобів, що пов’язі між собою та реалізують або підтримують виконання розробленої задачі. Технічним забезпеченням задачі є КТЗ, документація та технологічні процеси. Елементами технічного забезпечення є обчислювальна техніка, мережеве обладнання, засоби зберігання інформації, засоби збору інформації, та ін.

Необхідно розробити вимоги для реалізації КТЗ. До вимог КТЗ належать відношення ефективності до ціни, надійність роботи, мінімізація витрат на реалізацію КТЗ, безпека інформації та використання.

КТЗ задачі розробки мобільного додатку автоматизації сімейного бюджету має включати наступні пристрої:

– клавіатура;

– миша;

– принтер;

– користувацький комп’ютер;

– монітор.

У таблиці 2.1 наведений склад КТЗ задачі розробки мобільного додатку автоматизації сімейного бюджету.

Таблиця 2.1 – Опис компонентів КТЗ

Найменування компонента КТЗ (пристрою)	Характеристики (вимоги), які є ключовими для компонентів КТЗ (пристрою)	Кількість пристроїв
Системний блок	Процесор: Intel Core i7-8700 Відеокарта: Intel HD Graphics Частота процесора: 3.2 ГГц Материнська плата: Intel H310 Обсяг HDD: 256 ГБ Обсяг ОЗП: 8 ГБ	1
Монітор	Назва: 20" LGE 21B34A-B Діагональ: 20 Тип матриці екрану: TN + Film Дозвіл екрану: 1600x900 Співвідношення сторін екрану: 16: 9 Споживання енергії (в роботі) екраном: 20 Вт Споживання енергії (очікування) екраном: 0,3 Вт	1
Клавіатура	Назва клавіатури: Logitech K120 OEA Ukr USB Загальна кількість клавіш клавіатури:105 штук.	1
Миша	Назва миші: Logitech B100 Optical USB Black Тип пристрою миші: оптичний Кількість кнопок миші: 2 + 1 штуки.	1
Принтер	Назва принтеру: A4 Canon LBP-6020 (8458B001) Технологія друку принтеру: лазерна Кольоровість друку принтеру: монохромна Формат друку: A4 Обсяг пам’яті принтеру: 32 Мб	1
Маршрутизатор	Назва маршрутизатора: TP-Link TL-WR861N 802.12n Швидкість маршрутизатора: WAN – 100 Мбіт/с Вихідна потужність передавача маршрутизатора: 20 дБм;	1

Отже, ми обрали комплект технічних засобів для розробки мобільного програмного забезпечення.

2.3 Політика безпеки та захист інформації

Крадіжка даних, що зберігаються в програмах, завжди вважається однією з найважливіших загроз безпеці Android. Дослідження показують, що це може статися навіть із програмами, які по суті є бездоганними, але це може бути в самій операційній системі, наприклад, при обміні мережевими даними. Під час атаки шкідлива програма може працювати у фоновому режимі та збирати дані з цільової програми.

Багато програм використовують інформацію, що зберігається в базах даних. Крім того, програма може взаємодіяти з базами даних іншої програми, яка надає таку функціональність. Уразливий додаток може дозволити шкідливій програмі порушити цілісність і конфіденційність даних, що зберігаються в базах даних. Така легкість отримання несанкціонованого доступу до вмісту файлів привела до появи великої кількості спеціальних засобів, покликаних усунути проблему доступу до коду програми.

Додаток містить конфіденційні дані, оскільки він має функціонал додавання банківських карт чи виписок з них. Дані в додаток вводяться вручну чи в автоматизованому режимі. Загалом, для програми на платформі Android можна визначити такі ризики безпеки:

- несанкціонований доступ до даних у програмі;
- захоплення даних у каналах передачі;
- несанкціонований доступ до даних у базах даних;
- аналіз коду програми;
- несанкціоноване використання програми.

Існуючі інструменти безпеки додатків реалізують окремі механізми (наприклад, обфускація коду або аутентифікація користувача під час доступу до даних), не враховуючи весь спектр загроз, які можуть знизити безпеку, а не очікуване покращення. А багаторівневий підхід з урахуванням різних функцій програми в операційній системі Android дозволить реалізувати накладання цих загроз і тим самим підвищити безпеку програми.

Тому для захисту інформації в додатку буде створено окремий модуль, який відповідатиме за авторизацію та аутентифікацію користувача в системі.

Для реалізації заходів захисту персональних даних охорони використовуються універсальні механізми захисту, які базуються на ідентифікації особи, яка отримує доступ до інформаційної системи.

До таких механізмів належать:

– ідентифікація (найменування та розпізнавання), аутентифікація (підтвердження дійсності) та авторизація (передавання прав) суб'єктів;

– контроль (обмеження) доступу до мобільного додатку;

– реєстрація та аналіз подій, що відбуваються в додатку;

– контроль цілісності системи безпеки.

Механізми ідентифікації, аутентифікації та авторизації необхідні для підтвердження дійсності суб’єкта, забезпечення його функціонування в системі та визначення законності прав суб’єкта на об’єкт або на визначену ним діяльність. Ідентифікація — це процес розпізнавання компонента системи безпеки охоронюваного об’єкта, зазвичай за допомогою попередньо визначеного ідентифікатора або іншої унікальної інформації; кожна сутність або об’єкт у системі має бути однозначно ідентифікованим.

Аутентифікація — це перевірка автентичності користувача, процесу, пристрою чи іншого елемента системи (зазвичай перед наданням авторизації); а також перевірка цілісності та авторства даних під час зберігання або передачі даних для запобігання несанкціонованої модифікації.

Авторизація - це надання суб'єкту прав доступу до об'єкта.

Ідентифікація базується на базі окремого секретного елемента, доступного як для суб’єкта, так і для системи безпеки об’єкта мобільного середовища. Даний елемент і називають ідентифікатором. Роль ідентифікатора може виконувати певний ключ, пароль чи інший засіб підтвердження унікальності користувача.

Звичайно, система безпеки об’єкта має у своєму розпорядженні не сам секретний елемент, а деяку інформацію про нього, на підставі якої приймається рішення про адекватність суб’єкта ідентифікатору. Наприклад, перед інтерактивним сеансом більшість додатків запитують у користувача ім’я користувача та пароль. Введене ім’я – це ідентифікатор користувача, а пароль — аутентифікація. Операційна система технічного забезпечення безпеки охоронюваного об’єкта зазвичай зберігає не тільки пароль, а й його хеш-суму, що ускладнює відновлення пароля.

В сучасних безпекових технологіях використовуються такі методи ідентифікації/аутентифікації:

– одностороння аутентифікація, коли клієнт системи доступу до інформації доводить її достовірність;

– двостороння аутентифікація, коли, крім клієнта, системі (наприклад, банківській) також необхідно підтвердити свою автентичність;

– Тристороння аутентифікація, коли з метою підтвердження автентичності кожного з партнерів використовується так звана послуга нотаріального засвідчення.

Методи аутентифікації також можна розділити на однофакторні та двофакторні. Однофакторні методи поділяються на:

– логічні (паролі, ключові фрази, введені з клавіатури комп'ютера або клавіатури спеціалізованого пристрою);

– ідентифікація (носієм ключової інформації є фізичні об'єкти: дискета, магнітна картка, смарт-карта, карта зі штрих-кодом тощо. Недоліки: для зчитування інформації з фізичного об'єкта (носія) потрібен спеціальний зчитувач, носій може бути випадково втрачений, пошкоджений , викрадені або зроблені копії);

– біометричні (на основі аналізу унікальних характеристик людини, таких як: відбитки пальців, райдужна оболонка, голос, обличчя. Недоліки:

біометричні методи дорогі та складні в обслуговуванні; чутливі до змін параметрів медіа; мають низьку надійність;

– призначені лише для ідентифікації людини, а не програми чи пристрою).

Двофакторна аутентифікація використовується з метою підвищення рівня безпеки. Вона зазвичай використовує додаткове підтвердження, наприклад, окрім введення паролю, потрібно також надати надісланий у СМС секретний код, або окрім пін-коду потрібно ввести відповідь на секретне питання тощо.

2.4 Проектування інтерфейсу системи

Враховуючи те, що кожен мобільний пристрій компактний і самодостатній, прагнення користувачів працювати з інтерфейсом, що не залежить від характеру завдань, які вирішує пристрій, виглядає цілком природним. Кожен мобільний пристрій має свій функціональний колір. Типовий, добре продуманий додаток сприймається не як окремий інструментарій, а як гармонійне розширення самого пристрою.

З цієї причини при розробці мобільних додатків дуже важливо дотримуватися одного стилю. Для кожного типу пристроїв необхідно ретельно продумати способи запуску та зупинки програм, навігації по інтерфейсу та організації діалогів між користувачем і програмами. Користувачі мобільних пристроїв підсвідомо підлаштовуються під метафори інтерфейсу користувача, і будь-які відхилення від звичних шаблонів завдають їм значних незручностей. Набагато краще мати чотири різні версії програми, кожна з яких відповідає специфічній метафорі інтерфейсу користувача для конкретного пристрою, а не одну загальну програму, яку неможливо належним чином інтегрувати в будь-який цільовий пристрій.

Наведемо приклади реалізації інтерфейсу окремих функцій додатку. На рисунок 2.8 показано діалогове вікно для внесення нової категорії витрат сімейного бюджету.

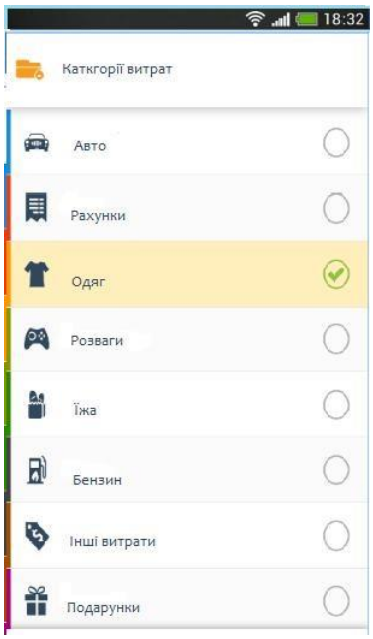

Рисунок 2.8 – Діалогове вікно створення категорії витрат сімейного бюджету

Створення звіту відбувається у відповідному вікні з поділом на категорії витрат. На рисунку 2.9 показано діалогове вікно створення звіту за витратами.

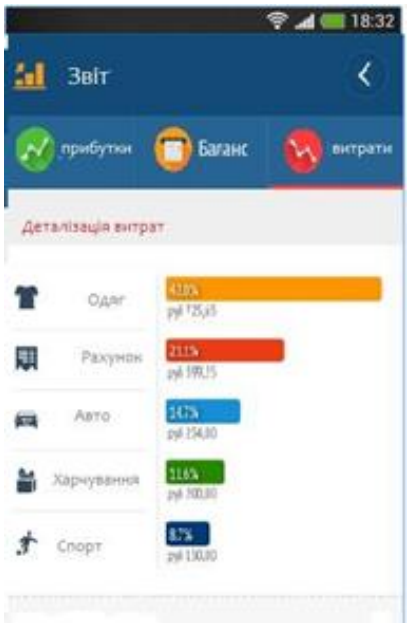

Рисунок 2.9 – Діалогове вікно створення звіту по витратах

Аналогічним чином буде спроектовано й інші представлення інтерфейсу окремих функцій. Створення звичного середовища робочого столу означає багато для настільних програм, але через різноманітність

функцій, які пропонують такі програми, вони можуть досягти тієї ж мети кількома способами (наприклад, за допомогою комбінацій клавіш, клацань мишею, меню та панелей інструментів). У випадку з мобільними пристроями часто існує лише один спосіб вирішення цієї проблеми, і користувач мимоволі звикає до одного конкретного.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ ДЛЯ МОБІЛЬНОГО ДОДАТКУ АВТОМАТИЗАЦІЇ ОБЛІКУ СІМЕЙНОГО БЮДЖЕТУ

3.1 Налаштування необхідного технічного та програмного забезпечення

Android використовує спеціальні механізми дії на основі моделі намірів. Якщо потрібно виконати дію (дзвінок, відправка електронної пошти, SMS), вона називається Intent. Провайдери контенту використовуються для обміну даними між додатками. Android SDK, Java та IDE зазвичай використовуються для створення проекту.

Кожна мова програмування унікальна і копіює методи та класи. У парадигмі програмування кожна мова пов’язана одна з одною поняттями, принципами та абстракціями, які визначають основний стиль програмування. Рідними мовами для Android є мови розмітки XML, C++ і C#. Ефективні методи програмування під Android включають в себе наступні позиції:

– основа для розробка Android-додатку – це Java, додатки якої транслюються в байт-код, що виконується віртуальною машиною Java JVM;

– використовуються тег RelativeLayout і властивість fill_parent;

– використовуються пусті елементи textField нульової висоти та ширини з параметром «centerInParent», рівним «true», щоб вирівняти елементи по центру екрана;

– використання зручних способів доступу до даних, таких як значення [SensorManager.DATA_X];

– використовуються методи onPause () / onResume (), щоб зберегти або закрити все, що вимагає цього;

– використання кольорових маркерів для предметів. Можна встановити колір фону для деяких об’єктів. Це дозволяє виявляти помилки та швидше їх знаходити;

– використання IDE відповідно до особистих уподобань і вимог розробника. Eclipse – клас IDE «все-в-одному», для якого існує багато доповнень. Eclipse має розширений редактор візуального інтерфейсу, ви можете редагувати файли XML в графічному інтерфейсі на основі конструктора, просто збираючи макети із запропонованих «деталей»; Плагін Eclipse MAT допоможе знайти витоки в пам'яті проекту; IntelliJ IDEA — одна з найшвидших і найзручніших IDE;

– використання кількох моніторів завдань для програмування. Додавання вікна програми на кілька екранів (вікно IDE, емулятор, попередній перегляд і документація);

– використання команди Source - Format, щоб відформатувати файли XML для читання;

– використання плагінів XML Tools для Блокнота ++ для швидкого редагування файлів XML і розуміння структури коду;

– використання функцій Intents за допомогою окремого методу;

– використання пулів потоків, вбудованих класів AsyncTask;

– зберігання даних SQLite, якщо воно займає багато місця;

– використання шаблонів (заздалегідь підготовлені фрагменти коду), які можна швидко використати для вставки в проект, натиснувши Ctrl + пробіл.

Існують принципи розробки продуктивних додатків під Android. Основні з них такі: необхідно економити апаратні ресурси, ефективно працювати з виділенням пам'яті, протоколювати і аналізувати хід виконання додатку, уникати зайвих об'єктів і створювати методи статичними; для констант, класів необхідно використовувати модифікатор static final і не використовувати enum там, де модифікатор не вписується.

При розробці програми для Android основним класом є клас Activity. Він представляє візуальну активність програми та визначає дії, які може виконувати користувач. У процесі запуску класу Activity спочатку

створюється об’єкт Activity, потім він запускається, тестується та знищується, і користувач смартфона переходить до нового об’єкта.

Протягом життєвого циклу додаток може перебувати в одному з трьох станів:

– Активний і запущений – цей інтерфейс користувача знаходиться на передньому плані (у верхній частині стеку активності).

– Призупинено – якщо цей інтерфейс користувача втратив фокус, але все ще видимий. У цьому стані код не виконується.

– Готово - якщо інтерфейс користувача невидимий. У цьому стані код не виконується.

Діяльність класу Activity показано на рисунку 3.1.

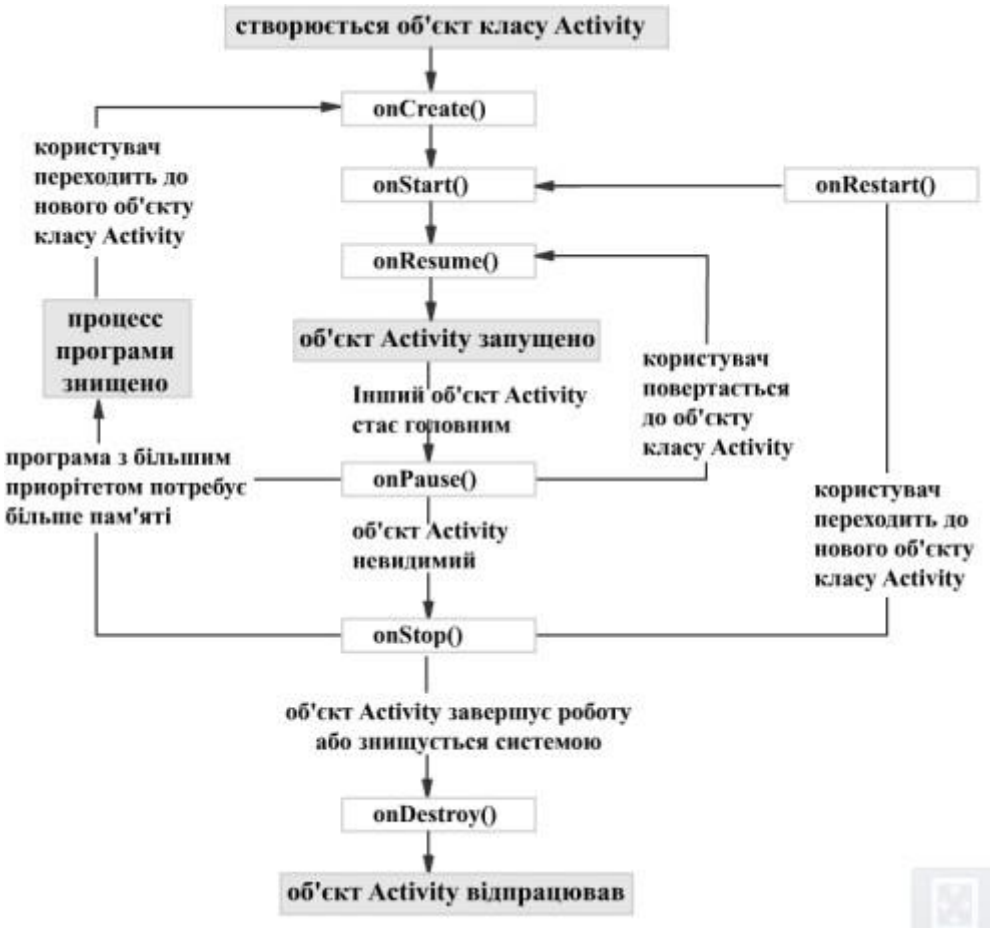

Рисунок 3.1- Розробка додатку на базі класу Activity

Під час створення нової активності, наприклад під час запуску програми, Android викликає метод OnCreate. Цей метод ініціалізує дію. Зокрема, створюються об’єкти візуального інтерфейсу. Цей метод отримує

Bundle, який містить попередній активний стан, якщо він збережений. Якщо дію відтворено, об'єкт є нульовим. Якщо дію було створено раніше, але в повільному стані, пакет містить інформацію, пов'язану з цією діяльністю.

Код розмітки класу LoginActivity наведений нижче:

```
    <RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:background="@drawable/login"
tools:context="com.example.familybudget.LoginActivity" >
```

Після встановлення розмітки формуємо блок введення паролю користувача, який повинен мати текстовий формат. За замовчуванням стоїть параметр «показувати введений пароль», щоб користувач не помилився при виборі розкладки тощо.

Блок введення паролю:

```
    <EditText
    android:id="@+id/editText2" android:layout_width="206dp"
android:layout_height="wrap_content"
android:layout_marginLeft="40dp"
android:layout_marginTop="181dp" android:ems="10"
android:hint="Password" android:inputType="textPassword"
android:textSize="15dp" android:visibility="visible" />
```

Блок введення логіну аналогічний за структурою для блоку введення паролю. Проте формат логіну не обмежується текстовим, тут дозволені цифри та інші символи.

Блок введення логіну:

```
<EditText
android:id="@+id/editText1" android:layout_width="206dp"
android:layout_height="wrap_content"
android:layout_marginLeft="40dp"
android:layout_marginTop="148dp" android:ems="10"
android:hint="Login"
android:textSize="15dp"
android:visibility="visible" />
```

Для реєстрації нового користувача і внесення даних про нього в базу використовується кнопка, яка підтверджує бажання зареєструватись. Вона відправляє користувача до реєстраційної форми. Аналогічним чином функціонує кнопка, яка дозволяє увійти зареєстрованому користувачеві.

Блок кнопки реєстрації/входу:

```
<ImageButton
android:id="@+id/imageButton2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignRight="@+id/editText2"
android:layout_alignTop="@+id/imageButton1"
android:layout_marginRight="32dp"
android:onClick="goToRegistration"
android:background="@drawable/button2" />
<ImageButton
android:id="@+id/imageButton1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
android:layout_below="@+id/editText2"
android:layout_marginRight="26dp"
android:layout_marginTop="18dp"
android:layout_toLeftOf="@+id/imageButton2"
android:background="@drawable/button1"
android:onClick="Logining" />
    </RelativeLayout>
```

Розмітка для програм Android створюється за допомогою ієрархії об'єктів View і ViewGroup. Об'єкти перегляду — це віджети інтерфейсу користувача, такі як кнопки або текстові поля, а ViewGroup — це невидимий тип контейнера, який визначає розташування дочірніх переглядів, таких як сітка або вертикальний список. Існує кілька стандартних типів тегів:

1. FrameLayout - це найпростіший тип тегів. Зазвичай це порожнє місце на екрані, яке можна заповнити лише дочірніми елементами View або ViewGroup. Усі дочірні елементи FrameLayout прикріплені до верхнього лівого кута екрана. Розташування дочірнього об'єкта не можна вказати в іншому місці тегів FrameLayout. Подальші дочірні об'єкти View будуть просто намальовані поверх попередніх компонентів, частково або повністю затінюючи їх, якщо об'єкт зверху непрозорий, тому єдиний дочірній для FrameLayout зазвичай розтягується до розміру батьківського контейнера.

2. LinearLayout – вирівнює всі дочірні об'єкти в одному напрямку – по вертикалі чи горизонталі. Напрямок визначається атрибутом android: orientation orientation. Усі дочірні елементи укладаються один за одним, так що список вертикального перегляду міститиме лише одну дочірню частину в рядку, незалежно від ширини. Горизонтальне положення списку розмістить елементи в одному рядку з висотою, що дорівнює висоті найвищого дочірнього елемента в списку.

3. TableLayout – розміщує дітей у рядках і стовпцях так само, як це робили веб-майстри в таблиці таблиці. TableLayout не відображає межі для своїх рядків, стовпців або клітинок. TableLayout може мати рядки з різною

кількістю комірок. При створенні макета таблиці деякі клітинки можна залишити порожніми, якщо це необхідно. При створенні розмітки для рядків використовуються об’єкти TableRow, які є дочірніми класами TableLayout (кожен TableRow визначає один рядок у таблиці). Рядок може не містити комірок, або він може містити одну або кілька комірок з іншими об’єктами. У комірці дозволяється інший TableLayout або LinearLayout.

4. RelativeLayout - дозволяє дочірнім компонентам визначати їх положення щодо батьківського компонента або відносно сусідніх дочірніх (ідентифікатор елемента). У RelativeLayout дочірні елементи розташовані таким чином, що якщо перший елемент розташовується по центру екрана, інші елементи, вирівняні за першим елементом, будуть вирівняні по центру екрана. У цьому макеті під час оголошення тегів у файлі XML елемент, на який посилаються інші об’єкти перегляду, має бути оголошений раніше, ніж інші елементи, які звертаються до нього за допомогою його ідентифікатора.

5. GridLayout – на перший погляд це може виглядати як TableLayout. Але насправді це набагато зручніше і функціональніше. Тег належить до класу android.widget.GridLayout і має стовпці, рядки та клітинки, як у TableLayout, але елементи можна налаштувати гнучко. У GridLayout ви можете вказати рядок і стовпець для будь-якого компонента, який буде розміщено в цьому місці таблиці. Вам не потрібно вказувати елемент для кожної клітинки, вам потрібно лише вказати елемент для тих осередків, де компоненти дійсно повинні бути. Компоненти можна розподілити на кілька елементів таблиці. Причому в одну комірку можна помістити кілька компонентів.

Усі описані теги є підкласами ViewGroup і успадковують властивості, визначені в класі View.

3.2. Процес розробки програмного продукту

Запустивши розроблений додаток, першою користувач бачить LoginActivity – вікно авторизації користувача, де йому необхідно ввести свій

логін і пароль для подальшої роботи з додатком. Частина коду реалізації LoginActivity наведена нижче:

```
    public void onClick(View view) {
    String login = inputLogin.getText().toString().trim();
String password = inputPassword.getText().toString().trim();
```

Як було вказано в діаграмі станів, при неправильно введених даних користувачеві має бути відмовлено в доступі. Для цього передбачено блок, який порівнює введені дані з наявними в базі, чи є такі дані і чи відповідають вони одному і тому ж користувачеві.

Блок перевірки логіну та пароля, якщо вони не введені чи введені неправильно приведений в лістингу.

```
    if (!login.isEmpty() && !password.isEmpty()) {
checkLogin(login, password);
    } else {
    Toast.makeText(getApplicationContext(),
    "Введіть дані для входу в додаток", Toast.LENGTH_LONG)
    .show();
    }
    }
```

Для перевірки логіну на правильність введення створено наступний лістинг. Функція перевіряє можливість створення json об’єкта.

```
    private void checkLogin(final String login, final String
password) {String tag_string_req = "req_login";
    StringRequest strReq = new StringRequest(Method.POST,
AppConfig.URL_LOGIN, new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
    Log.d(TAG, "Login Response: " + response.toString());
hideDialog();
```

```
try {
JSONObject jObj = new JSONObject(response);
boolean error = jObj.getBoolean("error");
if (!error) {
session.setLogin(true);
String uid = jObj.getString("uid");
```

Для створення підключення до БД та створення обєктів створено наступний код.

```
JSONObject user = jObj.getJSONObject("user");String name =
user.getString("name");
String email = user.getString("login");String created_at =
user
.getString("created_at"); db.addUser(name, login, uid,
created_at);
Intent intent = new Intent(LoginActivity.this,
BudgetActivity.class);
startActivity(intent);finish();
}
```

Спочатку відбувається перевірка на те, чи не пусті поля для вводу даних за допомогою конструкції if-else. Якщо поля заповнені, то викликається функція для вводу та перевірки логіна і пароля користувача. Використовуючи екземпляр класу StringRequest, посилається запит в базу на перевірку правильності введених даних для входу. Якщо дані вірні, то відкривається робоче вікно додатку

Одним з компонентів мобільного додатку є база даних. Частина коду створення таблиць бази даних наведено нижче:

```
private static final String DATABASE_NAME = "FBdatabase.db";
private static final int DATABASE_VERSION = 1;
private static final String DATABASE_TABLE_USERS = "Users";
```

```
    private static final String DATABASE_TABLE_BPERIOD =
"Budget_Period";private static final String DATABASE_TABLE_TTYPE
= "TargetType"; private static final String
DATABASE_TABLE_BUDGET = "Budget";
    static final String LOGIN_COLUMN = "login"; static final
String PASSWORD_COLUMN = "password";static final String
EMAIL_COLUMN = "email";
    static final String USER_NAME_COLUMN = "user_name"; static
final String NAME_PERIOD_COLUMN = "name_period";static final
String NAME_TYPE_COLUMN = "name_type"; static final String
TITLE_COLUMN = "title";
```

Як було зазначено про розробці структури баз даних, ми створюємо базу даних, де будуть зберігатись логіни та паролі зареєстрованих користувачів, щоб далі прив'язувати до певного логіну певні транзакції.

Для створенння бази даних користувачів додатку створено код приведений нижче:

```
    private static final String DATABASE_CREATE_SCRIPT_USERS =
"create table "
    + DATABASE_TABLE_USERS + " (" + BaseColumns._ID
    + " integer primary key autoincrement, " + LOGIN_COLUMN
    + " text not null, " + PASSWORD_COLUMN +" text not null, " +
    EMAIL_COLUMN
    + " text not null," + USER_NAME_COLUMN + "text not null);";
```

Для кожної категорії створюються окремі таблиці, які містять дані про дату «BPERIOD» та тип даних «TTYPE»

Створення таблиць:

```
    private static final String DATABASE_CREATE_SCRIPT_BPERIOD =
"create table "
    + DATABASE_TABLE_BPERIOD + " (" + BaseColumns._ID
```

```
    + " integer primary key autoincrement, " +
NAME_PERIOD_COLUMN
    + " text not null);";
    private static final String DATABASE_CREATE_SCRIPT_TTYPE =
"create table "
    + DATABASE_TABLE_TTYPE + " (" + BaseColumns._ID
    + " integer primary key autoincrement, " + NAME_TYPE_COLUMN
    + " text not null);";
```

Після створення окремих таблиць ми створюємо основну базу бюджету, яка містить розрахункові поля, що дозволяють отримати підсумки по кожній категорії витрат, а також порівняти їх з доходами.

Створення бази даних бюджету :

```
    private static final String DATABASE_CREATE_SCRIPT_BUDGET =
"create table "
    + DATABASE_TABLE_BUDGET + " (" + BaseColumns._ID
    + " integer primary key autoincrement, " + TITLE_COLUMN
    + " text not null);";
    @Override
    public void onCreate(SQLiteDatabase db) {
```

Окремо створюються скрипти для кожної бази даних, які дозволятимуть обробку даних з відповідної таблиці.

Створення нових даних:

```
    db.execSQL(DATABASE_CREATE_SCRIPT_USERS);
db.execSQL(DATABASE_CREATE_SCRIPT_BPERIOD);
db.execSQL(DATABASE_CREATE_SCRIPT_TTYPE);
db.execSQL(DATABASE_CREATE_SCRIPT_BUDGET);
    }
    @Override
```

```
    public void onUpgrade(SQLiteDatabase db, int oldVersion,
int newVersion) { db.execSQL("DROP TABLE IF IT EXISTS" +
DATABASE_TABLE_USERS); db.execSQL("DROP TABLE IF IT EXISTS" +
DATABASE_TABLE_BPERIOD); db.execSQL("DROP TABLE IF IT EXISTS" +
DATABASE_TABLE_TTYPE); db.execSQL("DROP TABLE IF IT EXISTS" +
DATABASE_TABLE_BUDGET);
    onCreate(db);
    }
```

Таким чином, ми створили основні бази даних для нашого додатку. Вони містять відповідні поля, які були запроектовані на рис. 2.3.-2.7., а також мають можливість для використання даних згідно побудованих користувачем запитів в розрізі періодів, категорій, типів операцій.

3.3 Реалізація інтерфейсу користувача

Інтерфейс — це зовнішня оболонка програми разом із програмами контролю доступу та іншими прихованими елементами керування, що дозволяє працювати з документами, даними та іншою інформацією, що зберігається на комп’ютері чи поза ним. Основна мета кожного додатка - забезпечити максимальну зручність і ефективність роботи з інформацією: документами, базами даних, графікою, зображеннями.

При створенні мобільного додатка були дотримані всі вищезазначені правила та розроблено оптимальний на вигляд та інтуїтивно зрозумілий інтерфейс.

При розробці інтерфейсу користувача слід визначити структуру діалогу, який буде відбуватися між користувачем і мобільним пристроєм. Чотири варіанти структури діалогу, розглянуті нижче, є варіаціями структури запитання та відповіді, але кожен має свої особливості та найбільш зручний для конкретного класу завдань.

Діалог запитань і відповідей. Структура діалогу запитань і відповідей заснована на аналогії зі звичайним інтерв'ю. Система виконує роль

інтерв'юера і отримує інформацію від користувача у вигляді відповідей на запитання. Це найвідоміша структура для діалогу; усі діалогові вікна, керовані пристроєм, певною мірою складаються із запитань, на які відповідає користувач. Однак у структурі запитання та відповіді цей процес яскраво виражений. У кожній точці діалогового вікна система відображає одне запитання як підказку, на яке користувач дає одну відповідь. Залежно від отриманої відповіді система може вирішити, яке запитання поставити далі. Структура запитань і відповідей не гарантує мінімальний обсяг введення, оцінений за кількістю натискань клавіш, але відповідний вибір ярликів може зменшити надмірність. Проте структура запитань і відповідей має один істотний недолік. Навіть якщо вступ досить швидкий, людині, яка вже знає, які питання ставить система і які на них відповідає, відповідати на цілу серію запитань досить нудно.

Діалоги, керовані меню. Меню є найпопулярнішим способом організації запитів на введення під час комп’ютерної розмови. Існує кілька основних форматів екранного меню::

– список об'єктів, обираних прямою вказівкою, або вказівкою номера (або мнемонічного коду);

– меню у вигляді блоку даних;

– меню у вигляді рядка даних;

– меню у вигляді піктограм.

Меню рядка даних може з’являтися у верхній або нижній частині екрана і часто залишається там протягом діалогового вікна. Як наслідок, за допомогою меню зручно відображати можливі варіанти даних, які необхідно ввести, доступними в будь-який момент роботи з системою. Додаткові меню у вигляді блоків даних «спливають» на екрані в позиції, зазначеній поточною позицією вказівника, або «спливають» безпосередньо з рядка меню верхнього рівня. Ці меню зникають, коли ви вибираєте опцію. Меню значків — це набір об’єктів, які можна вибрати, розкиданих по екрану; часто вибрані

об’єкти містять графічне представлення опцій. Якщо діалогове вікно складається лише з меню, можна реалізувати послідовний інтерфейс, де користувач використовує лише вказівні пристрої; проте така послідовність рідко досягається на практиці.

Меню є найбільш зручною діалоговою структурою для непідготовлених користувачів; сувора послідовність відкриття та ієрархічна вкладеність меню можуть дратувати професіонала, сповільнювати його. Традиційна структура меню недостатньо гнучка і не повністю узгоджується з методами адаптації діалогу, такими як розширений введення, що може прискорити темп роботи навченого користувача.

Діалог на основі екранних форм. Як структура типу «питання – відповідь», так і структура типу меню припускають обробку єдиної відповіді накожному кроці діалогу. Діалог на основі екранних форм допускає обробку на одному кроці діалогу декількох відповідей. На практиці форми використовуються в основному там, де облік якоїсь діяльності вимагає введення достатньо стандартного набору даних. Людина, що заповнює форму, може вибирати послідовність відповідей, тимчасово пропускати деяке питання, повертатися назад для корекції попередньої відповіді і навіть «порвати бланк» і почати заповнювати новий. Вона працює з формою до тих пір, поки не заповнить її повністю і не передасть системі. Програмна система може перевіряти кожну відповідь безпосередньо після введення або почекати і вивести список помилок лише після заповнення форми повністю.

Структура діалогу форми на екрані забезпечує високий рівень підтримки користувача: повідомлення про помилки та довідкова інформація можуть бути надані для кожного питання форми. Ви також можете допомогти користувачеві, включивши деякі елементи формату відповіді в поле запитання чи відповіді. Ця структура дозволяє підвищити швидкість введення даних у порівнянні зі структурою питання-відповідь та

маніпулювати більш широким діапазоном вхідних даних, ніж меню; крім того, він може співпрацювати з користувачами будь-якої кваліфікації.

Діалог на основі командної мови. Структура діалогу на основі командної мови є такою ж поширеною, як і структура типу меню. Він дуже часто використовується в операційних системах і знаходиться на іншому кінці спектру діалогових структур щодо структури типу меню. Історично це перша з реалізованих структур діалогу. При такій організації діалогу програмна система не відображає нічого, крім безперервної підказки (запрошення ввести команду), що означає, що система готова до роботи. Кожна команда вводиться з нового рядка і зазвичай закінчується натисканням клавіші «Enter». Відповідальність за правильність інструкцій несе користувач. Система повідомляє про неможливість виконати неправильну команду, зазвичай без пояснення причин.

Як і меню, діалогове вікно на основі команд зручно для введення керуючих повідомлень, але надає більше можливостей у будь-якому місці діалогового вікна та не вимагає ієрархічної організації допоміжних програм. Програмна система може обробляти велику кількість команд, але на практиці їх кількість має бути обмежена, щоб не обтяжувати пам'ять користувача. Структура командної мови не має належної підтримки користувачів і в основному підходить для кваліфікованих спеціалістів.

Структура команди є найшвидшою та найгнучкішою з усіх діалогових структур. Більшість користувацьких інтерфейсів природною мовою реалізовані за допомогою командних мов з дуже великим набором ключових слів.

Ручне тестування призначеного для користувача інтерфейсу проводиться тестувальником-оператором, який керується в своїй роботі описом тестових прикладів у вигляді набору сценаріїв. Кожен сценарій включає перерахування послідовності дій, які повинні виконати оператор, і

опис важливих для аналізу результатів тестування у відповідь реакцій системи, відбиваних в призначеному для користувача інтерфейсі.

Форма запису сценарію для проведення ручного тестування - таблиця, в якій в одній колонці описані дії (кроки сценарію), в іншій - очікувана реакція системи, а третя призначена для запису того, чи співпала очікувана реакція системи з реальною і перерахування неспівпадань.

Таблиця 3.1 - Результати проведення ручного тестування

№	Дія	Реакція системи	Результат
1	Запустіть програму	Появляється стартове програми для введення логіну та паролю. На формі дорступна реєстрація новго користувача системи.	
2	Ввести логін та пароль та натисніть «Вхід»	Перехід до діалогового вікна додавання інформації, де доступні поля: сума, опис, дата, теги, час повторення.	

Продовження таблиці 3.1

3	Вибрати необхідні параметри пошуку Вибрати команду «шукати»	Появитьсь інформація про записи з необхідною інформацією	Вірно
4	У вікні внесення прибутків вибрати рахунок, категорію, дату, натиснути кнопку «+»	Відповідна інформація буде записана в базу даних	Вірно
5	У вікні внесення витрат вибрати рахунок,категорію, дату, Натиснути кнопку «Записати»	Відповідна інформація буде записана в базу даних	Вірно
6	У ві кні внесення нової категорії внести назву категорії та натиснути кнопку «Зберегти»	Відповідна інформація буде записана в базу даних	Вірно
7	У вікні внесення нової категорії витрам внестиназву одиниці виміру та натиснути кнопку «Зберегти»	Відповідна інформація буде записана в базу даних	Вірно
8	У вікні створення звіту вибрати аналіз витрат, період, вибрати категорії та рахунки	Поява звіту у вибраній формі: Назва призначення, ліміт, тип призначення, період, витрати, залишок.	

Таким чином, додаток працює коректно, дозволяє зареєструватись чи увійти з раніше зареєстрованими даними, внести дані про доходи та витрати та отримати підрахунок бюджету за період (в нашому прикладі – за місяць).

В майбутньому додаток можна доопрацювати, розширивши його функції.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

4.1 Розрахунок витрат на розробку та впровадження проектного рішення

Для розрахунку економічної ефективності проектного рішення системи для мобільного додатку автоматизації обліку сімейного бюджету необхідно мати дані про можливі витрати, пов‘язані з розробкою і експлуатацією програми.

Даний програмний продукт проектується з розрахунку на існуюче апаратне забезпечення. Розробка не вимагає придбання спеціального обладнання, тому розрахунок витрат на придбання спеціального обладнання для проведення експериментальних робіт проводитись не буде.

Витрати на розробку і впровадження програмного засобу (К) визначаються як:

$$K_{заг} = K_1 + K_2, \tag{4.1}$$

де K_1 - витрати на розробку програмного засобу, грн.;

K_2 - витрати на налагодження і дослідну експлуатацію програмного засобу на комп’ютерній техніці, грн.

Витрати на розробку програмного засобу включають в себе:

- витрати на оплату праці розробників ($B_{оп}$);
- відрахування у спеціальні державні фонди ($B_{ф}$);
- вартість додаткових виробів, що закуповуються ($B_{д}$);
- транспортно-заготівельні витрати ($B_{тр}$);
- витрати на придбання спецобладнання ($B_{со}$);
- накладні витрати ($B_{н}$);
- інші витрати ($B_{ін}$).

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування. Розмір ЗП обчислюється на основі трудоємності відповідних робіт у людино-днях та середньої ЗП відповідних категорій працівників.

Для розробки програмного продукту необхідно два спеціаліста-розробника, а саме:

Керівник проекту (К);

Інженер-програміст (Іп);

Витрати на оплату праці спеціалістів становить :

1. Місячний оклад керівника проекту складає 12000 грн./міс.;
2. Місячний оклад інженера-програміста складає 8400 грн./міс.;

Отримаємо наступні значення денної ЗП розробників (з розрахунку по 24 робочі дні в місяць):

$ЗП_{Д1}$ = 12000/24 = 500 грн.

$ЗП_{Д2}$ = 8400/24 = 350 грн.

Розрахунок витрат на оплату праці усіх розробників проекту обчислюємо за формулою:

$$B_{ОП} = \sum_{i=1}^{N} n_i \cdot t_i \cdot ЗП_{Дi} \qquad (4.2)$$

де n_i - чисельність розробників проекту *i*-ої спеціальності, ос.;

t_i - час, витрачений на розробку проекту працівником *i*-ої спеціальності, днів;

$ЗП_{Д1}$ - денна ЗП розробника *i*-ої спеціальності, грн.

Таким чином, витрати на оплату праці розробників складають:

Bon1 = 1*15*500 = 7500 грн.

Bon2 = 1*60*350 = 21000 грн.

Сумарні витрати на оплату праці:

$В_{СОП}$ = 7500+21000 = 28500 грн.

Розрахунок витрат оплати праці розробників проекту зведений в табл. 4.1.

Таблиця 4.1- Розрахунок витрат на оплату праці

Спеціальність розробника	Кількість розробників.чол	Час роботи, днів	Денна ЗП розробника, грн.	Витрати на оплату праці, грн.
Керівник проекту	1	15	500	7500
Інженер програміст	1	60	350	21000
Всього				28500

Величину зборів у спеціальні державні фонди (*ВФ*) визначають у відсотковому співвідношенні від суми фонду оплати праці. Єдиний соціальний внесок складає 22% від ФОП.

Сумарні витрати на збори у спеціальні державні фонди складають:

$$ВФ=28500*0,22=6270 \text{ грн}$$

Витрати на додаткові вироби, що закуповуються $B_д$ (папір, картридж) визначаються за їхніми фактичними цінами з врахуванням найменування, номенклатури та необхідної їх кількості в проекті. Транспортно-заготівельні витрати складають 12% від вартості виробу.

Вихідні дані та результати розрахунків оформлені у вигляді табл. 4.2.

Витрати на придбання спеціального обладнання $(B_{со})$ для проведення експериментальних робіт розраховується в тому випадку, коли для розробки та впровадження проектного рішення необхідне придбання додаткових технічних засобів. У нашому випадку витрат нема.

Таблиця 4.2 - Розрахунок витрат на куповані вироби

Найменування купованих виробів	Марка, тип	Кількість на розробку, шт.	Ціна за одиницю, грн.	Сума витрат, грн.	Сума витрат з врахуванням трансп.-загот. витрат, грн..
Папір	Maestro	1	85,00	85,00	95,2
Картридж до принтера	Canon MP240	1	420,00	420,00	470,4
Всього					565,6

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими процентами (25%) до витрат на оплату праці і складають:

$$B_H = B_{ОП} \cdot 0{,}25 = 28500*0{,}25 = 7125 \text{ грн.}$$

Інші витрати відображають видатки, які не враховані в попередніх статтях витрат. Вони розраховуються за встановленими процентами (10%) до витрат на оплату праці і складають:

$$B_{ін} = B_{ОП} \cdot 0{,}1 = 28500* 0{,}1 = 2850 \text{ грн.}$$

Витрати на розробку проектного рішення обчислюємо за формулою:

$$K_1 = B_{ОП} + B_Ф + B_д + B_{СО} + B_Н + B_{ін} \quad (4.3)$$

Підставивши результати розрахунків отримаємо:

$$K_1 = 28500 + 6270 + 565{,}6 + 0 + 7125 + 2850 = 45310{,}6 \text{ грн.}$$

Витрати на відлагодження і дослідну експлуатацію програмного пакету визначаємо з формули:

$$K_2 = S_{м.г.} \cdot t_{від}, \quad (4.4)$$

де $S_{м.г}$ – вартість однієї машинної години роботи ПК, (3 грн./год.);

$t_{від}$ – кількість машинних годин (за експериментальними даними на відлагодження програми потрібно 16 годин машинного часу).

Отримаємо:

$$К_2 = 3 \cdot 16 = 48 \text{ грн.}$$

Витрати на розробку і впровадження програмного засобу:

$$К_{заг} = К_1 + К_2,$$

$$К_{заг} = 45310,6 + 48 = 45358,6 \text{ грн.}$$

Результати розрахунків зведені в табл. 4.3:

Таблиця 4.3 - Кошторис витрат на розробку проектного рішення

№ п/п	Найменування елементів витрат	Сума витрат, грн.
1	Витрати на оплату праці	28500,0
2	Відрахування у спеціальні державні фонди	6270,0
3	Витрати на експлуатацію ПК	48,00
4	Витрати на куповані вироби	565,6
5	Накладні витрати	7125,0
6	Інші витрати	2850,0
Всього		45358,6

4.2. Визначення комплексного показника якості

Комплексний показник якості ($П_я$) визначається шляхом порівняння показників якості проектованої системи і вибраного аналогу.

Вибір показників якості здійснюється експертним методом. Номенклатура основних груп показників якості наступна:

- показники призначення;
- показники надійності;
- показники безпеки;
- ергономічні показники;

Комплексний показник якості проектованої підсистеми визначаємо методом арифметичного середньозваженого з формули:

$$П_я = \sum_{i=1}^{m} C_i * q_i \qquad (4.5)$$

де m – кількість одиничних показників (параметрів), прийнятих для оцінювання якості проектованої системи;

q_i – коефіцієнт вагомості кожного з параметрів щодо їхнього впливу на технічний рівень та якість проектованої системи (встановлюється експертним шляхом), причому:

$$\sum_{i=1}^{m} q_i = 1{,}0 \qquad (4.6)$$

C_i – часткові показники якості, визначені порівнянням числових значень одиничних показників проектованої системи і аналога за формулами:

$$C_i = \frac{П_{npi}}{П_{ai}} \text{ або } C_i = \frac{П_{ai}}{П_{npi}}, \qquad (4.7)$$

де $П_{npi}, П_{ai}$ – кількісні значення і-го одиничного показника якості відповідно проектованої системи і аналога.

Результати розрахунку зведені в табл. 4.4:

Таблиця 4.4 - Визначення комплексного показника якості програм

Показники	Числове значення показників		Відносний показник якості C_i	Коефіцієнт вагомості q_i, %	$C_i * q_i$ %
	Аналог $П_{2i}$	Проектована система $П_{1i}$			
1	2	3	4	5	6
1. Показники призначення					
1.1. Швидкодія, бали	7	8	1,14	0,1	0,114
1.2. Ступінь новизни	7	8	1,14	0,05	0,057
1.3. Налаштування, %	60	75	1,25	0,05	0,0625
2. Показники надійності					
2.1.Стійкість програми до некоректних дій користувача, %	60	75	1,25	0,15	0,1875
2.2. Ймовірність помилки (бали)	6	5	0,83	0,2	0,1667
3. Показники безпеки					

Продовження таблиці 4.4

3.1. Можливий негативний вплив на середовище виконання (%)	60	60	1	0,15	0,15
4.Ергономічні показники					
4.1. Зручність інтерфейсу (бали)	5	7	1,4	0,2	0,28
4.3. Товарний вигляд (бали)	6	6	1	0,1	0,1
Всього	1	1,118			

Отже, $П_я = 1{,}118$

4.3. Визначення показників економічної ефективності

Показник конкурентоздатності розраховується відносно вартості аналогів програмних засобів:

$$K_{КЗ} = \frac{П_я * Ц_{С(а)}}{Ц_{С(п)}} \quad (4.8)$$

$K_{КЗ}$ = 1018*1,118/1096 = 1,4

Економічний ефект в сфері експлуатації:

$$E_{ЕКС} = B_{(Е)а} - B_{(Е)П} \quad (4.9)$$

$E_{ЕКС}$ = 53881 – 6095 = 47786 грн.

Термін окупності витрат на проектування рішення:

$$T_{ОК} = \frac{K}{E_{ЕКС}} \quad (4.10)$$

$T_{ОК}$ = 45358,6/ 47786 = 0,95 року.

Результати розрахунків зведені в табл. 4.8:

У цьому розділі проведена економічна оцінка проектного рішення розроблення системи для мобільного додатку автоматизації обліку сімейного бюджету.

Таблиця 4.8 - Показники економічної ефективності проектного рішення

Назва показників	Значення показників	
	Аналог	Проектне рішення
Капітальне вкладення, грн.	-	45358,6
Ціна придбання одиниці програмного продукту, грн.	1096	1018
Обсяг продаж в перший рік, грн.	6095	53881
Економічний ефект в сфері експлуатації, грн.	-	47786
Термін окупності витрат на проектування рішення, роки	-	0,95
Коефіцієнт конкурентоздатності	-	1,4

Зроблені економічні обчислення показали, що проектне рішення має високий коефіцієнт конкурентоспроможності (1,4), що робить його кращим за порівнюваний аналог. Економічний ефект в сфері експлуатації складає 47786 грн.. Все це дає підстави вважати розробку проекту доцільною, незважаючи на те, що капітальні вкладення складають 45358,6 гривень.

Програмна система має малий термін окупності, що складає менше 1 року. Це є нормальним терміном для програмної системи такого класу.

ВИСНОВКИ

Робота присвячена теоретичним та практичним питанням розробки мобільних додатків на прикладі додатку для обліку обліку сімейного бюджету.

В першому розділі розглянуто сутність та призначення мобільних додатків, а також проведено постановку завдань для розробки. При розробці додатку мають бути реалізовані два основних принципи: універсальність – можливість використання в будь-яких родинах, починаючи від однієї особи і до 10 чоловік, з будь-яким режимом надходжень та витрат – регулярні, разові, обмежені цільовим призначенням тощо; адаптованість – забезпечення можливості користувачеві самостійно провести настроювання з урахуванням специфіки конкретного домогосподарства і змін у статтях.

В другому розділі здійснено проектування архітектури системи та вимог до її основних модулів.

Система матиме наступні базові функції:

– перегляд рахунків та історії операцій;

– запис поточних витрат, доходів та майбутніх надходжень і витрат; робота без Internet (оффлайн);

– синхронізація з хмарним сховищем при наявності Internet підключення.

Також розроблено вимоги до системи безпеки додатку для уникнення викрадення платіжних даних та персональної інформації.

В третьому розділі проведено тестування системи та розробка інтерфейсу користувача. ак. Структура діалогу форми на екрані забезпечує високий рівень підтримки користувача: повідомлення про помилки та довідкова інформація можуть бути надані для кожного питання форми.

В четвертому розділі проведена економічна оцінка проектного рішення розроблення системи для мобільного додатку автоматизації обліку сімейного

бюджету. Зроблені економічні обчислення показали, що проектне рішення має високий коефіцієнт конкурентоспроможності (1,4), що робить його кращим за порівнюваний аналог. Економічний ефект в сфері експлуатації складає 47786 грн.. Все це дає підстави вважати розробку проекту доцільною, незважаючи на те, що капітальні вкладення складають 45358,6 гривень.

Програмна система має малий термін окупності, що складає менше 1 року. Це є нормальним терміном для програмної системи такого класу

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Журавель Н. О. Організаційна регламентація бізнес-процесів як умова забезпечення їх ринкової безпеки / Н. О. Журавель // Управління розвитком. - 2014. - № 2. - С. 121-124. – [Електронний ресурс]. Режим доступу: http://nbuv.gov.ua/UJRN/Uproz_2014_2_50

2. Дубовой В.М. Моделювання процесів і систем керування / В.М.Дубовой, С.М.Москвіна, О.Д.Никитенко. – Вінниця, ВНТУ, 2015. –103 с.

3. Бевз О.М. Проектування програмних засобів систем управління / О.М. Бевз, В.М. Папінов, Ю.А. Скидан. – [Електронний ресурс]. Режим доступу: http://posibnyky.vntu.edu.ua/bevz/

4. Моделювання бізнес-процесів [Текст]/ Державний вищий навчальний заклад “Українська академія банківської справи Національного банку України” ; [уклад. О. І. Подоляка, К. М. Жулінська]. – Суми : ДВНЗ “УАБС НБУ”, 2014. – 20 с.

5. Козенков Д.Е. Проектування бізнес-процесів як основа створення архітектури підприємства / Д.Е. Козенков // Вісник СумДУ. Серія Економіка, №3'2013. – с. 126-136.

6. Лаврінський Г.В. Моделювання системних характеристик в економіці /Г.В. Лаврінський, О.С. Пшенишнюк, С.В. Устинко, О.Д. Шарапod. – К.: ЕКМО, 2014. – 169с.

7. Шкіль Р.А. Аналітичний огляд методів моделювання бізнес-процесів в електронній комерції / Р.А. Шкіль //Вісник Дніпропетровського національного університету залізничного транспорту, № 6, 2015. – с. 194-198.

8. Иванов Д. Ю., Новиков Ф. А. Основы моделирования на UML: Учеб. пособие. – СПб.: Изд-во Политехн. ун-та, 2013. – 249с.

9. Ілляшенко Н.С., Савченко О.С. Розроблення програми просування мобільних технологій на B2B ринку. URL: http://essuir.sumdu.edu.ua/dspace/1/Illiashenko_Savchenko_stat_EE_2014.pdf (дата звернення 12.01. 2022)

10. Харченко К.В. Методи та засоби розробки програмних додатків для операційної системи Андроїд. Вісник НТУ “ХПІ». 2014. №17(1060). с.68-72.

11. Ілляшенко С.М. Сучасні тенденції застосування інтернет-технологій. URL: http://www.nbuv.gov.ua/portal/Soc_Gum/Mimi/2011_4_2/2_1.pdf (дата звернення 12.01. 2022)

12. Лубко Д.В. Методологія проектування та інструментарій для створення мобільних додатків. Вісник НТУ “ХПІ». 2014. №56(1029). с. 117-122.

13. Офіційний сайт Київського міжнародного інституту соціології. URL: http://www.kiis.com.ua/ (дата звернення 12.01. 2022)

ДОДАТКИ

Додаток А

Код програмного засобу

FamilyBudget.sql

```
IF EXISTS(select * from sys.databases where name='FamilyBudget') DROP DATABASE FamilyBudget;
CREATE DATABASE FamilyBudget; USE FamilyBudget;
CREATE TABLE Users (
ID_User INT IDENTITY(1,1) PRIMARY KEY, [Login] VARCHAR(32) NULL,
[Password] VARCHAR(32) NUll, Email VARCHAR(32) NOT NULL, UserName NVARCHAR(32) NOT NULL
);
CREATE TABLE Budget_Period (
ID_Period INT IDENTITY(1,1) PRIMARY KEY, NamePeriod NVARCHAR(32) NOT NULL
);
CREATE TABLE TargetType (
ID_TargetType int IDENTITY(1,1) PRIMARY KEY, Name_Type NVARCHAR(32) NOT NULL
);
CREATE TABLE BudgetTarget (
ID_Target int IDENTITY(1,1) PRIMARY KEY, Name_Target NVARCHAR(32) NOT NULL,
Target_amount MONEY, Startingdate DATE,
ID_TargetType INT NOT NULL, CONSTRAINT TargetType_ID_TargetType_fk FOREIGN KEY (ID_TargetType) REFERENCES TargetType (ID_TargetType),
ID_Period INT NOT NULL, CONSTRAINT Budget_Period_ID_Period_fk FOREIGN KEY (ID_Period) REFERENCES Budget_Period (ID_Period)
```

```
);
CREATE TABLE Budget (
ID_Budget INT IDENTITY(1,1) PRIMARY KEY, Title
NVARCHAR(32),
);
CREATE TABLE ExpensesTags (
ID_Tag INT IDENTITY(1,1) PRIMARY KEY, TagName NVARCHAR(32)
NOT NULL
);
CREATE TABLE IncomeTags (
ID_Tag INT IDENTITY(1,1) PRIMARY KEY, TagName NVARCHAR(32)
NOT NULL
);
CREATE TABLE Repeat_Type (
ID_Repeat INT IDENTITY(1,1) PRIMARY KEY, Name NVARCHAR(32)
NOT NULL
);
CREATE TABLE BudgetTarget_ExpensesTags (
ID INT IDENTITY(1,1) PRIMARY KEY,
ID_Target INT NULL, CONSTRAINT
BudgetTarget_ExpensesTags_ID_Target_fk FOREIGN KEY (ID_Target)
REFERENCES BudgetTarget (ID_Target),
ID_Tag INT NULL, CONSTRAINT
BudgetTarget_ExpensesTags_ID_Tag_fk FOREIGN KEY (ID_Tag)
REFERENCES ExpensesTags (ID_Tag),
ID_Budget INT NULL, CONSTRAINT
BudgetTarget_ExpensesTags_ID_Budget_fk FOREIGN KEY (ID_Budget)
REFERENCES Budget (ID_Budget)
);
CREATE TABLE User_Budget (
ID INT IDENTITY(1,1) PRIMARY KEY,
ID_User INT NOT NULL, CONSTRAINT User_Budget_ID_User_fk
FOREIGN KEY (ID_User) REFERENCES Users (ID_User),
ID_Budget INT NOT NULL, CONSTRAINT User_Budget_ID_Budget_fk
FOREIGN KEY (ID_Budget) REFERENCES Budget (ID_Budget)
```

```
);
CREATE TABLE Repeat_Income (
ID_RepeatIncome INT IDENTITY(1,1) PRIMARY KEY,
ID_User INT NOT NULL, CONSTRAINT Repeat_Income_ID_User_fk
FOREIGN KEY (ID_User) REFERENCES Users (ID_User),
Amount MONEY NOT NULL,
[Description] NVARCHAR(255) NULL,
ID_Repeat INT NOT NULL, CONSTRAINT
Repeat_Income_ID_Repeat_fk FOREIGN KEY (ID_Repeat) REFERENCES
Repeat_Type (ID_Repeat),
RepeatStartDate DATE, EndRepeatDate DATE, IncomeRepeats INT
NULL
);
CREATE TABLE Repeat_Expenses (
ID_RepeatExpenses INT IDENTITY(1,1) PRIMARY KEY,
ID_User INT NOT NULL, CONSTRAINT Repeat_Expenses_ID_User_fk
FOREIGN KEY (ID_User) REFERENCES Users (ID_User),
Amount MONEY NOT NULL,
[Description] NVARCHAR(255) NULL,
ID_Repeat INT NOT NULL, CONSTRAINT
Repeat_Expenses_ID_Repeat_fk FOREIGN KEY (ID_Repeat) REFERENCES
Repeat_Type (ID_Repeat),
RepeatStartDate DATE, EndRepeatDate DATE,
ExpensesRepeats INT NULL
);
CREATE TABLE Income (
ID_Income INT IDENTITY(1,1) PRIMARY KEY,
ID_Budget INT NOT NULL, CONSTRAINT Income_ID_Budget_fk
FOREIGN KEY (ID_Budget) REFERENCES Budget (ID_Budget),
ID_RepeatIncome INT NULL, CONSTRAINT
Income_ID_RepeatIncome_fk FOREIGN KEY (ID_RepeatIncome)
REFERENCES Repeat_Income (ID_RepeatIncome),
ID_User INT NOT NULL, CONSTRAINT Income_ID_User_fk
FOREIGN KEY (ID_User) REFERENCES Users (ID_User),
Amount MONEY NOT NULL, [Date] DATE,
```

```
    [Description] NVARCHAR(255) NULL,
    );
    CREATE TABLE Expenses (
    ID_Expenses INT IDENTITY(1,1) PRIMARY KEY,
    ID_Budget INT NOT NULL, CONSTRAINT Expenses_ID_Budget_fk
FOREIGN KEY (ID_Budget) REFERENCES Budget (ID_Budget),
    ID_RepeatExpenses INT NULL, CONSTRAINT
Expenses_ID_RepeatIncome_fk FOREIGN KEY (ID_RepeatExpenses)
    REFERENCES Repeat_Expenses
    (ID_RepeatExpenses),
    ID_User INT NOT NULL, CONSTRAINT Expenses_ID_User_fk
FOREIGN KEY (ID_User) REFERENCES Users (ID_User),
    Amount MONEY NOT NULL, [Date] DATE,
    [Description] NVARCHAR(255) NULL, Private_expenses BIT
    );
    CREATE TABLE Repeat_Income_IncomeTag (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    ID_Tag INT NOT NULL, CONSTRAINT
Repeat_Income_IncomeTag_ID_Tag_fk FOREIGN KEY (ID_Tag)
REFERENCES IncomeTags (ID_Tag),
    ID_RepeatIncome INT NOT NULL, CONSTRAINT
Repeat_Income_IncomeTag_ID_RepeatIncome_fk FOREIGN KEY
(ID_RepeatIncome) REFERENCES Repeat_Income (ID_RepeatIncome),
    ID_Budget INT NOT NULL, CONSTRAINT
Repeat_Income_IncomeTag_ID_Budget_fk FOREIGN KEY (ID_Budget)
REFERENCES Budget (ID_Budget)
    );
    CREATE TABLE Repeat_Expenses_ExpensesTag (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    ID_Tag INT NOT NULL, CONSTRAINT
Repeat_Expenses_ExpensesTag_ID_Tag_fk FOREIGN KEY (ID_Tag)
REFERENCES ExpensesTags (ID_Tag),
    ID_RepeatExpenses INT NOT NULL, CONSTRAINT
Repeat_Expenses_ExpensesTag_ID_RepeatExpenses_fk FOREIGN KEY
(ID_RepeatExpenses) REFERENCES Repeat_Expenses
```

```
(ID_RepeatExpenses),
     ID_Budget INT NOT NULL, CONSTRAINT
     Repeat_Expenses_ExpensesTag_ID_Budget_fk     FOREIGN     KEY
(ID_Budget) REFERENCES Budget (ID_Budget)
     );
     CREATE TABLE Expenses_ExpensesTag (
     ID INT IDENTITY(1,1) PRIMARY KEY,
     ID_Tag        INT        NOT        NULL,        CONSTRAINT
Expenses_ExpensesTag_ID_Tag_fk FOREIGN KEY (ID_Tag)   REFERENCES
ExpensesTags (ID_Tag),
     ID_Expenses       INT       NOT       NULL,       CONSTRAINT
Expenses_ExpensesTag_ID_Expenses_fk  FOREIGN  KEY  (ID_Expenses)
REFERENCES Expenses (ID_Expenses),
     ID_Budget       INT       NOT       NULL,       CONSTRAINT
Expenses_ExpensesTag_ID_Budget_fk   FOREIGN   KEY   (ID_Budget)
REFERENCES Budget (ID_Budget)
     );
     CREATE TABLE Income_IncomeTag (
     ID INT IDENTITY(1,1) PRIMARY KEY,
     ID_Tag INT NOT NULL, CONSTRAINT Income_IncomeTag_ID_Tag_fk
FOREIGN KEY (ID_Tag) REFERENCES IncomeTags (ID_Tag),
     ID_Income       INT       NOT       NULL,       CONSTRAINT
Income_IncomeTag_ID_Income_fk FOREIGN KEY (ID_Income) REFERENCES
Income (ID_Income),
     ID_Budget       INT       NOT       NULL,       CONSTRAINT
Income_IncomeTag_ID_Budget_fk FOREIGN KEY (ID_Budget) REFERENCES
Budget (ID_Budget)
     );
     Procedures.sql
     --USE FamilyBudget;
     --GO
     CREATE  PROCEDURE  ADDUser  (@Login  varchar(32),  @Password
varchar(32), @Mail varchar(32), @UserName nvarchar(32) )
     AS
     DECLARE @CountRows int
```

```
    SELECT @CountRows =COUNT(*) FROM Users WHERE [Login] =
@Login AND Email = @Mail IF @CountRows = 0
    INSERT INTO Users ([Login], [Password], Email, UserName)
VALUES
    (@Login, @Password, @Mail, @UserName);
        Budget target
    GO
    CREATE PROCEDURE ADDBudgetTarget (@Name_Target
nvarchar(32),@Amount money,@StartingDate date, @ID_TargetType
int)
    AS
    INSERT INTO BudgetTarget(Name_Target, Target_amount,
Startingdate,ID_TargetType)
    VALUES(@Name_Target,@Amount,@StartingDate,@ID_TargetType);
    GO
    CREATE PROCEDURE ADDBudgetTarget_ExpensesTags(@ID_Target
int, @ID_Tag int, @ID_Budget int)
    AS
    DECLARE @countRows int
    SELECT @countRows =COUNT(*) FROM BudgetTarget_ExpensesTags
WHERE ID_Target=@ID_Target AND ID_Tag = @ID_Tag AND ID_Budget =
ID_Budget IF @countRows=0
    INSERT INTO BudgetTarget_ExpensesTags(ID_Target, ID_Tag,
ID_Budget) VALUES(@ID_Target, @ID_Tag, @ID_Budget);
    --add expense and expense tags GO
    CREATE PROCEDURE ADDExpenses(@ID_Budget int,
@ID_RepeatExpenses int, @ID_User int, @Amount money, @Date date,
@Description nvarchar(255), @Private_expenses bit = 0)
    AS
    INSERT INTO Expenses(ID_Budget, ID_RepeatExpenses, ID_User,
Amount, [Date], [Description], Private_expenses)
    VALUES (@ID_Budget, @ID_RepeatExpenses, @ID_User, @Amount,
@Date, @Description, @Private_expenses);
    GO
    CREATE PROCEDURE ADDExpensesTags_Expenses(@TagName
```

```
nvarchar(32),@ID_Budget int)
     AS
     DECLARE @IdExpenses int
     SELECT @IdExpenses = MAX(ID_Expenses) FROM Expenses DECLARE
@count int
     DECLARE @IdTag int DECLARE @countRows int
     SELECT   @count=COUNT(TagName)   FROM   ExpensesTags   WHERE
TagName = @TagName IF @count=0
     BEGIN
     END ELSE
     INSERT INTO ExpensesTags(TagName) VALUES(@TagName);
     SET @IdTag = @@IDENTITY
     SELECT  @IdTag=ID_Tag  FROM  ExpensesTags  WHERE  TagName  =
@TagName SELECT @countRows = COUNT(*) FROM Expenses_ExpensesTag
WHERE
     ID_Expenses   =   @IdExpenses   AND   ID_Tag   =   @IdTag   IF
@countRows=0
     INSERT INTO Expenses_ExpensesTag(ID_Tag, ID_Expenses,
     ID_Budget)
     VALUES (@IdTag, @IdExpenses,@ID_Budget);
     - GO
     -- Add income and income tags GO
     CREATE PROCEDURE ADDIncome(@ID_Budget int, @ID_RepeatIncome
int,  @ID_User  int,  @Amount  money,  @Date  date,  @Description
nvarchar(255))
     AS
     INSERT  INTO  Income(ID_Budget,  ID_RepeatIncome,  ID_User,
Amount, [Date],[Description])
     VALUES  (@ID_Budget,  @ID_RepeatIncome,  @ID_User,  @Amount,
@Date, @Description);
     GO
     CREATE        PROCEDURE        ADDIncomeTag_Income(@TagName
nvarchar(32),@ID_Budget int) AS
     DECLARE @IdIncome int
     SELECT  @IdIncome  =  MAX(ID_Income)  FROM  Income  DECLARE
```

```
@count int
     DECLARE @IdTag int DECLARE @countRows int
     SELECT @count=COUNT(TagName) FROM IncomeTags WHERE TagName
= @TagName IF @count=0
     BEGIN
     END ELSE
     INSERT INTO IncomeTags(TagName) VALUES(@TagName);
     SET @IdTag = @@IDENTITY
     SELECT @IdTag=ID_Tag FROM IncomeTags WHERE TagName =
@TagName SELECT @countRows=COUNT(*) FROM Income_IncomeTag WHERE
ID_Income =
     @IdIncome AND ID_Tag = @IdTag
     IF @countRows=0
     INSERT INTO Income_IncomeTag(ID_Tag, ID_Income,ID_Budget)
VALUES (@IdTag, @IdIncome,@ID_Budget);
     GO
     CREATE PROCEDURE ADDRepeat_Type(@Name nvarchar(32)) AS
     INSERT INTO Repeat_Type(Name)
     VALUES(@Name);
     GO
     CREATE PROCEDURE ADDBudget(@Title nvarchar(32)) AS
     INSERT INTO Budget(Title)
     VALUES(@Title);
     GO
     CREATE PROCEDURE ADDUser_Budget(@ID_User int, @ID_Budget
int) AS
     DECLARE @CountRows int
     SELECT @CountRows=COUNT(*) FROM User_Budget WHERE ID_User =
@ID_User AND ID_Budget = @ID_Budget
     IF @CountRows = 0
     INSERT INTO User_Budget(ID_User, ID_Budget)
VALUES(@ID_User, @ID_Budget);
     --add repeat expense tags GO
     CREATE PROCEDURE ADDRepeat_ExpensesTag(@TagName
nvarchar(32) , @IdUser int,@ID_Budget int)
```

```
     AS
     DECLARE @IdRepeat_Expenses int
     SELECT  @IdRepeat_Expenses  =  MAX(ID_RepeatExpenses)  FROM
Repeat_Expenses WHERE ID_User = @IdUser
     DECLARE @IdExpenses int
     SELECT @IdExpenses = MAX(ID_Expenses) FROM Expenses WHERE
ID_User = @IdUser DECLARE @count int
     DECLARE @IdTag int DECLARE @countRows int
     SELECT   @count=COUNT(TagName)   FROM   ExpensesTags   WHERE
TagName = @TagName IF @count=0
     BEGIN
     END ELSE
     INSERT INTO ExpensesTags(TagName) VALUES(@TagName);
     SET @IdTag = @@IDENTITY
     SELECT  @IdTag=ID_Tag  FROM  ExpensesTags  WHERE  TagName  =
@TagName          SELECT          @countRows=COUNT(*)          FROM
Repeat_Expenses_ExpensesTag WHERE
     ID_Tag = @IdTag AND ID_RepeatExpenses = @IdRepeat_Expenses
IF @countRows = 0
     INSERT       INTO       Repeat_Expenses_ExpensesTag(ID_Tag,
ID_RepeatExpenses,ID_Budget)
     VALUES (@IdTag, @IdRepeat_Expenses,@ID_Budget)
     -- REPEAT- Income tag
     GO
     CREATE  PROCEDURE  ADDRepeat_IncomeTag(@TagName  nvarchar(32)
, @IdUser int,@ID_Budget int)
     AS
     DECLARE @IdRepeat_Income int DECLARE @IdIncome int
     SELECT   @IdRepeat_Income   =   MAX(ID_RepeatIncome)   FROM
Repeat_Income WHERE ID_User
     = @IdUser
     SELECT @IdIncome = MAX(ID_Income) FROM Income WHERE ID_User
= @IdUser DECLARE @count int
     DECLARE @countRows int DECLARE @IdTag int
     SELECT  @count=COUNT(TagName)  FROM  IncomeTags  WHERE  TagName
```

```
= @TagName IF @count=0
     BEGIN
     END ELSE
     INSERT INTO IncomeTags(TagName) VALUES(@TagName);
     SET @IdTag = @@IDENTITY
     SELECT  @IdTag=ID_Tag  FROM  IncomeTags  WHERE  TagName  =
@TagName SELECT @countRows=COUNT(*) FROM Repeat_Income_IncomeTag
WHERE

     ID_Tag  =  @IdTag  AND  ID_RepeatIncome  =  @IdRepeat_Income  IF
@countRows = 0
     INSERT          INTO          Repeat_Income_IncomeTag(ID_Tag,
ID_RepeatIncome,ID_Budget)              VALUES              (@IdTag,
@IdRepeat_Income,@ID_Budget);
     GO
     -- Delete User
     CREATE PROCEDURE DeleteUser(@UserID int) AS
     DECLARE  @UserBudgets  table  (RowId  int  IDENTITY(1,1),
BudgetID int) DECLARE @count int
     DECLARE @RowIterator int
     DECLARE @countUserBudget int -- count user`s budgets what
he used DECLARE @idBudgetforDelete int
     --for income and expenses records DECLARE @countRecords int
DECLARE @Iterator int
     DECLARE @idforDelete int
     DECLARE  @BudgetIncomes  table  (IDRow  int  IDENTITY(1,1),
IncomeID  int)  DECLARE  @BudgetExpenses  table  (IDRow  int
IDENTITY(1,1), ExpensesID int) DECLARE @BudgetRepeatIncome table
(RowId int IDENTITY(1,1), IncomeID int)
     DECLARE    @BudgetRepeatExpenses    table    (RowId    int
IDENTITY(1,1),  ExpensesID  int)  DECLARE  @BudgetTarget  table
(RowId int IDENTITY(1,1), TargetID int)
     INSERT  INTO  @UserBudgets(BudgetID)  SELECT  ID_Budget  FROM
User_Budget WHERE ID_User = @UserID
     DELETE FROM User_Budget WHERE ID_User = @UserID
```

```
     SELECT @count=COUNT(*) FROM @UserBudgets SET @RowIterator=1
     SET @countUserBudget=0 WHILE @RowIterator<= @count BEGIN
     SELECT @idBudgetforDelete=BudgetID FROM @UserBudgets WHERE
RowId = @RowIterator SELECT @countUserBudget=COUNT(ID_Budget)
FROM User_Budget WHERE ID_Budget = @idBudgetforDelete
     IF @countUserBudget=0 BEGIN
     INSERT INTO @BudgetExpenses(ExpensesID) SELECT ID_Expenses
FROM Expenses WHERE ID_Budget=@idBudgetforDelete
     SELECT @countRecords = COUNT(ExpensesID) FROM
     @BudgetExpenses
     SET @Iterator =1
     WHILE @Iterator<=@countRecords BEGIN
     SELECT @idforDelete= ExpensesID FROM
     @BudgetExpenses WHERE IDRow = @Iterator
     DELETE FROM BudgetTarget_ExpensesTags WHERE
     ID_Budget=@idBudgetforDelete
     = @idforDelete @idforDelete
     DELETE FROM Expenses_ExpensesTag WHERE ID_Expenses DELETE
FROM Expenses WHERE ID_Expenses =
     SET @Iterator = @Iterator+1 END

     INSERT INTO @BudgetTarget(TargetID) SELECT ID_Target FROM
BudgetTarget_ExpensesTags WHERE ID_Budget=@idBudgetforDelete
     SELECT @countRecords=COUNT(*) FROM @BudgetTarget SET
@Iterator=1
     WHERE RowId=@Iterator @idforDelete
     WHILE @Iterator<=@countRecords BEGIN
     SELECT @idforDelete= TargetID FROM @BudgetTarget DELETE
FROM BudgetTarget WHERE ID_Target =
     SET @Iterator = @Iterator+1 END
     INSERT INTO @BudgetIncomes(IncomeID) SELECT ID_Income FROM
Income WHERE ID_Budget=@idBudgetforDelete
     SELECT @countRecords = COUNT(IncomeID) FROM
     @BudgetIncomes
     WHERE IDRow = @Iterator @idforDelete
```

```
    SET @Iterator =1
    WHILE @Iterator<=@countRecords BEGIN
    Select @idforDelete= IncomeID FROM @BudgetIncomes DELETE
From Income_IncomeTag WHERE ID_Income =
    DELETE From Income WHERE ID_Income = @idforDelete SET
@Iterator= @Iterator+1
    END
    INSERT INTO @BudgetRepeatExpenses(ExpensesID) SELECT
ID_RepeatExpenses FROM Repeat_Expenses WHERE ID_User=@UserID
    SELECT @countRecords = COUNT(ExpensesID) FROM
@BudgetRepeatExpenses SET @Iterator =1
    WHILE @Iterator<=@countRecords BEGIN
    SELECT @idforDelete= ExpensesID FROM @BudgetRepeatExpenses
WHERE RowId = @Iterator
    DELETE FROM Repeat_Expenses_ExpensesTag WHERE
ID_RepeatExpenses = @idforDelete DELETE FROM Repeat_Expenses
WHERE ID_RepeatExpenses = @idforDelete
    SET @Iterator = @Iterator+1 END
    INSERT INTO @BudgetRepeatIncome(IncomeID) SELECT
ID_RepeatIncome FROM Repeat_Income WHERE ID_User=@UserID
    SELECT @countRecords = COUNT(IncomeID) FROM
@BudgetRepeatIncome SET @Iterator =1
    WHILE @Iterator<=@countRecords BEGIN
    SELECT @idforDelete= IncomeID FROM @BudgetRepeatIncome
WHERE RowId = @Iterator DELETE FROM Repeat_Income_IncomeTag
WHERE ID_RepeatIncome = @idforDelete DELETE FROM Repeat_Income
WHERE ID_RepeatIncome = @idforDelete
    SET @Iterator = @Iterator+1 END
    DELETE FROM Budget WHERE ID_Budget = @idBudgetforDelete
    END
    SET @RowIterator = @RowIterator+1 END
    DECLARE @CountUserRefIncome int
    DECLARE @CountUserRefExpenses int
    SELECT @CountUserRefIncome=count(*) FROM Income WHERE
ID_User=@UserID SELECT @CountUserRefExpenses=count(*) FROM
```

```
Expenses WHERE ID_User=@UserID IF @CountUserRefExpenses=0 AND
@CountUserRefIncome=0
     BEGIN
     DELETE FROM Users WHERE ID_User=@UserID END
     ELSE
     UPDATE Users SET [Login]=NULL, [Password] = NULL WHERE
ID_User=@UserID
     --Delete Expense and income GO
     CREATE PROCEDURE DeleteExpenses (@ExpensesID int) AS
     DELETE FROM Expenses_ExpensesTag WHERE ID_Expenses =
@ExpensesID DELETE FROM Expenses WHERE ID_Expenses = @ExpensesID
     GO
     CREATE PROCEDURE DeleteIncome (@IncomeID int) AS
     DELETE FROM Income_IncomeTag WHERE ID_Income = @IncomeID
DELETE FROM Income WHERE ID_Income = @IncomeID
     --Delete Repeat expenses and incomes GO
     CREATE PROCEDURE DeleteRepeatExpenses (@RepeatId int) AS
     DECLARE @RepaetExpenses table (IDRow int IDENTITY(1,1),
ExpensesId int) DECLARE @countExpenses int
     DECLARE @ExpensesIterator int DECLARE @ExpenseForDelete int
     INSERT INTO @RepaetExpenses(ExpensesId) SELECT ID_Expenses
FROM Expenses WHERE ID_RepeatExpenses = @RepeatId
     SELECT @countExpenses=COUNT(*) FROM @RepaetExpenses SET
@ExpensesIterator = 1
     WHILE @ExpensesIterator<=@countExpenses BEGIN
     SELECT @ExpenseForDelete=ExpensesID FROM @RepaetExpenses
WHERE IDRow= @ExpensesIterator
     DELETE FROM Expenses_ExpensesTag WHERE ID_Expenses =
@ExpenseForDelete DELETE FROM Expenses WHERE ID_Expenses =
@ExpenseForDelete
     SET @ExpensesIterator = @ExpensesIterator+1 END
     DELETE FROM Repeat_Expenses_ExpensesTag WHERE
ID_RepeatExpenses = @RepeatId DELETE FROM Repeat_Expenses WHERE
ID_RepeatExpenses = @RepeatId
     -- GO
```

```
CREATE PROCEDURE DeleteRepeatIncome (@RepeatId int) AS
DECLARE @RepaetIncome table (IDRow int IDENTITY(1,1),
IncomeId int) DECLARE @countIncome int
DECLARE @IncomeIterator int DECLARE @IncomeForDelete int
INSERT INTO @RepaetIncome(IncomeId) SELECT ID_Income FROM
Income WHERE ID_RepeatIncome = @RepeatId
SELECT @countIncome=COUNT(*) FROM @RepaetIncome
SET @IncomeIterator = 1
WHILE @IncomeIterator<=@countIncome BEGIN
SELECT @IncomeForDelete=IncomeID FROM @RepaetIncome WHERE
IDRow= @IncomeIterator
DELETE FROM Income_IncomeTag WHERE ID_Income =
@IncomeForDelete DELETE FROM Income WHERE ID_Income =
@IncomeForDelete
SET @IncomeIterator = @IncomeIterator+1 END
DELETE FROM Repeat_Income_IncomeTag WHERE ID_RepeatIncome =
@RepeatId DELETE FROM Repeat_Income WHERE ID_RepeatIncome =
@RepeatId
-- DELETE Budget GO
CREATE PROCEDURE DeleteBudget (@IDBudget int) AS
DECLARE @countRecords int DECLARE @Iterator int DECLARE
@idforDelete int
DECLARE @BudgetIncomes table (IDRow int IDENTITY(1,1),
IncomeID int) DECLARE @BudgetExpenses table (IDRow int
IDENTITY(1,1), ExpensesID int) DECLARE @BudgetRepeatIncome table
(RowId int IDENTITY(1,1), IncomeID int)
DECLARE @BudgetRepeatExpenses table (RowId int
IDENTITY(1,1), ExpensesID int) DECLARE @BudgetTarget table
(RowId int IDENTITY(1,1), TargetID int)
INSERT INTO @BudgetIncomes(IncomeID) SELECT ID_Income FROM
Income WHERE ID_Budget=@IDBudget
SELECT @countRecords=COUNT(*) FROM @BudgetIncomes SET
@Iterator=1
WHILE @Iterator<=@countRecords BEGIN
SELECT @idforDelete=IncomeID FROM @BudgetIncomes WHERE
```

```
IDRow=@Iterator
     INSERT     INTO     @BudgetRepeatIncome(IncomeID)     SELECT
ID_RepeatIncome FROM Income WHERE ID_Income = @idforDelete
     DELETE FROM Income_IncomeTag WHERE ID_Income = @idforDelete
SET @Iterator= @Iterator+1
     END
     INSERT INTO @BudgetExpenses(ExpensesID) SELECT ID_Expenses
FROM Expenses WHERE ID_Budget=@IDBudget
     SELECT   @countRecords=COUNT(*)   FROM   @BudgetExpenses   SET
@Iterator=1
     WHILE @Iterator<=@countRecords BEGIN
     SELECT  @idforDelete=ExpensesID  FROM  @BudgetExpenses  WHERE
IDRow=@Iterator
     INSERT    INTO    @BudgetRepeatExpenses(ExpensesID)    SELECT
ID_RepeatExpenses FROM Expenses WHERE ID_Expenses = @idforDelete
     DELETE   FROM   Expenses_ExpensesTag   WHERE   ID_Expenses   =
@idforDelete SET @Iterator = @Iterator+1
     END
     SELECT  @countRecords=COUNT(*)  FROM  @BudgetRepeatIncome  SET
@Iterator=1
     WHILE @Iterator<=@countRecords BEGIN
     SELECT   @idforDelete=   IncomeID   FROM   @BudgetRepeatIncome
WHERE RowId=@Iterator
     DELETE FROM Repeat_Income_IncomeTag WHERE ID_RepeatIncome =
@idforDelete
     SET @Iterator = @Iterator+1 END
     SET @Iterator=1
     WHILE @Iterator<=@countRecords BEGIN
     SELECT   @idforDelete=   IncomeID   FROM   @BudgetRepeatIncome
WHERE RowId=@Iterator
     DELETE   FROM   Repeat_Income   WHERE   ID_RepeatIncome   =
@idforDelete SET @Iterator = @Iterator+1
     END
     SELECT   @countRecords=COUNT(*)   FROM   @BudgetRepeatExpenses
SET @Iterator=1
```

```
    WHILE @Iterator<=@countRecords BEGIN
    SELECT @idforDelete= ExpensesID FROM @BudgetRepeatExpenses
WHERE RowId=@Iterator
    DELETE FROM Repeat_Expenses_ExpensesTag WHERE
ID_RepeatExpenses = @idforDelete
    SET @Iterator = @Iterator+1 END
    SET @Iterator=1
    WHILE @Iterator<=@countRecords BEGIN
    SELECT @idforDelete=ExpensesID FROM @BudgetRepeatExpenses
WHERE RowId=@Iterator
    DELETE FROM Repeat_Expenses WHERE ID_RepeatExpenses =
@idforDelete SET @Iterator = @Iterator+1
    END
    INSERT INTO @BudgetTarget(TargetID) SELECT ID_Target FROM
BudgetTarget_ExpensesTags WHERE ID_Budget=@IDBudget
    SELECT @countRecords=COUNT(*) FROM @BudgetTarget SET
@Iterator=1
    WHILE @Iterator<=@countRecords BEGIN
    SELECT @idforDelete= TargetID FROM @BudgetTarget WHERE
    RowId=@Iterator
    DELETE FROM BudgetTarget WHERE ID_Target = @idforDelete SET
@Iterator = @Iterator+1
    END

    DELETE FROM Income WHERE ID_Budget = @IDBudget DELETE FROM
Expenses WHERE ID_Budget = @IDBudget
    DELETE FROM BudgetTarget_ExpensesTags WHERE
ID_Budget=@IDBudget DELETE FROM User_Budget WHERE ID_Budget =
@IDBudget
    DELETE FROM Budget WHERE ID_Budget= @IDBudget
    LoginActivity.java
    public class LoginActivity extends Activity {
    private static final String TAG =
RegisterActivity.class.getSimpleName();
    private Button btnLogin;
```

```
    private Button btnLinkToRegister; private EditText
inputEmail; private EditText inputPassword; private
ProgressDialog pDialog; private SessionManager session; private
SQLiteHandler db;
    @Override
    public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_login);
    inputEmail = (EditText) findViewById(R.id.email);
inputPassword = (EditText) findViewById(R.id.password); btnLogin
= (Button) findViewById(R.id.btnLogin); btnLinkToRegister =
(Button)
    findViewById(R.id.btnLinkToRegisterScreen);
    // Progress dialog
    pDialog = new ProgressDialog(this);
pDialog.setCancelable(false);
    // SQLite database handler
    db = new SQLiteHandler(getApplicationContext());
    // Session manager
    session = new SessionManager(getApplicationContext());
    // Check if user is already logged in or not
    if (session.isLoggedIn()) {
    // User is already logged in. Take him to main activity
    Intent intent = new Intent(LoginActivity.this,
MainActivity.class); startActivity(intent);
    finish();
    }
    // Login button Click Event btnLogin.setOnClickListener(new
View.OnClickListener() {
    public void onClick(View view) {
    String email = inputEmail.getText().toString().trim();
String password = inputPassword.getText().toString().trim();
    // Check for empty data in the form
    if (!email.isEmpty() && !password.isEmpty()) {
    // login user checkLogin(email, password);
```

```
} else {
// Prompt user to enter credentials
Toast.makeText(getApplicationContext(),
"Please enter the credentials!", Toast.LENGTH_LONG)
.show();
}
}
});
// Link to Register Screen
btnLinkToRegister.setOnClickListener(new View.OnClickListener()
{

public void onClick(View view) {
Intent i = new Intent(getApplicationContext(),
RegisterActivity.class);
startActivity(i); finish();
}
});
}
/**
* function to verify login details in mysql db
* */
private void checkLogin(final String email, final String
password) {
// Tag used to cancel the request String tag_string_req =
"req_login";
pDialog.setMessage("Logging in ..."); showDialog();
StringRequest strReq = new StringRequest(Method.POST,
AppConfig.URL_LOGIN, new Response.Listener<String>() {
@Override
public void onResponse(String response) {
Log.d(TAG, "Login Response: " + response.toString());
hideDialog();
try {
JSONObject jObj = new JSONObject(response);
```

```
boolean error = jObj.getBoolean("error");
// Check for error node in json
if (!error) {
// user successfully logged in
// Create login session session.setLogin(true);
// Now store the user in SQLite String uid =
jObj.getString("uid");
JSONObject user = jObj.getJSONObject("user"); String name =
user.getString("name");
String email = user.getString("email"); String created_at =
user
.getString("created_at");
// Inserting row in users table db.addUser(name, email,
uid, created_at);
// Launch main activity
Intent intent = new Intent(LoginActivity.this,
MainActivity.class); startActivity(intent); finish();
} else {
// Error in login. Get the error message String errorMsg =
jObj.getString("error_msg");
Toast.makeText(getApplicationContext(),
errorMsg, Toast.LENGTH_LONG).show();
}
} catch (JSONException e) {
// JSON error e.printStackTrace();
Toast.makeText(getApplicationContext(), "Json error: " +
e.getMessage(), Toast.LENGTH_LONG).show();
}
}
}, new Response.ErrorListener() {
@Override
public void onErrorResponse(VolleyError error) { Log.e(TAG,
"Login Error: " + error.getMessage());
Toast.makeText(getApplicationContext(),
error.getMessage(), Toast.LENGTH_LONG).show();
```

```
hideDialog();
     }
     }) {
     @Override
     protected Map<String, String> getParams() {
     // Posting parameters to login url
     Map<String, String> params = new HashMap<String, String>();
params.put("email", email);
     params.put("password", password);
     return params;
     }
     };
     AppController.getInstance().addToRequestQueue(strReq,
tag_string_req);
     }
     private void showDialog() {
     if (!pDialog.isShowing()) pDialog.show();
     }
     private void hideDialog() {
     if (pDialog.isShowing()) pDialog.dismiss();
     }
     }
     RegisterActivity.java
     public class RegisterActivity extends Activity {
     private      static      final      String      TAG      =
RegisterActivity.class.getSimpleName();
     private Button btnRegister; private Button btnLinkToLogin;
private EditText inputFullName; private EditText inputEmail;
private EditText inputPassword; private ProgressDialog pDialog;
private SessionManager session; private SQLiteHandler db;
     @Override
     public    void    onCreate(Bundle    savedInstanceState)    {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_register);
     inputFullName    =    (EditText)    findViewById(R.id.name);
```

```
inputEmail = (EditText) findViewById(R.id.email); inputPassword
= (EditText) findViewById(R.id.password); btnRegister = (Button)
findViewById(R.id.btnRegister);
     btnLinkToLogin                    =                    (Button)
findViewById(R.id.btnLinkToLoginScreen);
     // Progress dialog
     pDialog         =         new         ProgressDialog(this);
pDialog.setCancelable(false);
     // Session manager
     session = new SessionManager(getApplicationContext());
     // SQLite database handler
     db = new SQLiteHandler(getApplicationContext());
     // Check if user is already logged in or not
     if (session.isLoggedIn()) {
     // User is already logged in. Take him to main activity
Intent intent = new Intent(RegisterActivity.this,
     MainActivity.class); startActivity(intent); finish();
     }
     //        Register        Button        Click        event
btnRegister.setOnClickListener(new View.OnClickListener() {
     public void onClick(View view) {
     String  name  =  inputFullName.getText().toString().trim();
String  email  =  inputEmail.getText().toString().trim();  String
password = inputPassword.getText().toString().trim();
     if     (!name.isEmpty()     &&     !email.isEmpty()     &&
!password.isEmpty())
     {
     registerUser(name, email, password);
     } else {
     Toast.makeText(getApplicationContext(),
     "Please enter your details!", Toast.LENGTH_LONG)
     .show();
     }
     }
     });
```

```
// Link to Login Screen
btnLinkToLogin.setOnClickListener(new
View.OnClickListener() {
public void onClick(View view) {
Intent i = new Intent(getApplicationContext(),
LoginActivity.class);
startActivity(i); finish();
}
});
}
/**
* Function to store user in MySQL database will post
params(tag, name,
* email, password) to register url
* */
private void registerUser(final String name, final String
email,
final String password) {
// Tag used to cancel the request String tag_string_req =
"req_register";
pDialog.setMessage("Registering ..."); showDialog();
StringRequest strReq = new StringRequest(Method.POST,
AppConfig.URL_REGISTER, new Response.Listener<String>() {
@Override
public void onResponse(String response) {
Log.d(TAG, "Register Response: " + response.toString());
hideDialog();
try {
JSONObject jObj = new JSONObject(response); boolean error =
jObj.getBoolean("error"); if (!error) {
// User successfully stored in MySQL
// Now store the user in sqlite String uid =
jObj.getString("uid");
JSONObject user = jObj.getJSONObject("user"); String name =
user.getString("name");
```

```
String email = user.getString("email"); String created_at =
user
.getString("created_at");
// Inserting row in users table db.addUser(name, email,
uid, created_at);
Toast.makeText(getApplicationContext(), "User successfully
registered. Try login now!", Toast.LENGTH_LONG).show();
// Launch login activity Intent intent = new Intent(
RegisterActivity.this, LoginActivity.class);
startActivity(intent); finish();
} else {
// Error occurred in registration. Get the error
// message
String errorMsg = jObj.getString("error_msg");
Toast.makeText(getApplicationContext(),
errorMsg, Toast.LENGTH_LONG).show();
}
} catch (JSONException e) { e.printStackTrace();
}
}
}, new Response.ErrorListener() {
@Override
public void onErrorResponse(VolleyError error) {
Log.e(TAG, "Registration Error: " + error.getMessage());
Toast.makeText(getApplicationContext(),
error.getMessage(), Toast.LENGTH_LONG).show();
hideDialog();
}
}) {
@Override
protected Map<String, String> getParams() {
// Posting params to register url
Map<String, String> params = new HashMap<String, String>();
params.put("name", name);
params.put("email", email); params.put("password",
```

```
password);
      return params;
      }
      };
      //      Adding      request      to      request      queue
AppController.getInstance().addToRequestQueue(strReq,
tag_string_req);
      }
      private void showDialog() {
      if (!pDialog.isShowing()) pDialog.show();
      }
      private void hideDialog() {
      if (pDialog.isShowing()) pDialog.dismiss();
      }
      }
```