MONEY MANAGER APPLICATION USING ANDROID STUDIO

A Project Report

Submitted to the University of Madras in partial fulfillment of the requirement for the award of the Degree of

Bachelor of Computer Applications (BCA)

Submitted by

TARUNMUTHU C S	**(Reg. No. 212103276)**
AVINASH S	**(Reg. No. 212103217)**
BHARATH RAJ S T	**(Reg. No. 212103219)**

Under the Guidance of

Mr. Kirubakaran M.C.A., M.Phil.,

Assistant Professor

ALPHA ARTS AND SCIENCE COLLEGE

CHENNAI – 600116

April- 2024

MONEY MANAGER APPLICATION USING ANDROID STUDIO

A Project Report

Submitted to the University of Madras in partial fulfillment of the requirement for the award of the Degree of

Bachelor of Computer Applications (BCA)

Submitted by

TARUNMUTHU C S	**(Reg. No. 212103276)**
AVINASH S	**(Reg. No. 212103217)**
BHARATH RAJ S T	**(Reg. No. 212103219)**

Under the Guidance of

Mr. Kirubakaran M.C.A., M.Phil.,

Assistant Professor

ALPHA ARTS AND SCIENCE COLLEGE

CHENNAI – 600116

April- 2024

ALPHA ARTS AND SCIENCE COLLEGE
CHENNAI – 600116

DEPARTMENT OF COMPUTER APPLICATIONS

<u>BONAFIDE CERTIFICATE</u>

Certified that this Project report entitled **"MONEY MANAGER USING ANDROID STUDIO"** is a Bonafide record of the project work done by **TARUNMUTHU C S (212103276), AVINASH S (212103217)** and **BHARATH RAJ S T(212103219)** under my supervision and guidance, in partial fulfillment of the requirement for award of the Degree of Computer Applications in Alpha Arts and Science College, Chennai – 600 116.

Signature of the Guide **Head of the Department**

Submitted for the Viva-Voce Examination held on________________.

Internal Examiner **External Examiner**

DECLARATION

We hereby declare that the Project Work entitled **“MONEY MANAGER USING ANDROID STUDIO”** submitted to University of Madras in partial fulfillment of the requirement for the award of the Degree of **“Bachelor of Computer Applications”** is the original work done by us under the supervision of M**r. Kirubakaran, M.C.A., M.Phil.,** Department of Computer Applications, Alpha Arts and Science College, Chennai – 116.

NAME	**REG. NO**
TARUNMUTHU C S	**(212103276)**
AVINASH S	**(212103217)**
BHARATH RAJ S T	**(212103219)**

Place: **SIGNATURE OF THE CANDIDATES**

Date: **1.**

2.

3.

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

CIN: U80902TN2021PTC141464

www.pantechelearning.com

09-04-2024

COMPLETION CERTIFICATE

This is to acknowledge that students from **"ALPHA ARTS AND SCIENCE COLLEGE"** has completed **Project** on the **Title** of **"MONEY MANAGER APP"** at our concern from **DECEMBER 2023** to **MARCH 2024.**

1. **TARUNMUTHU. CS (212103276)**
2. **AVINASH. S (212103217)**
3. **BHARATH RAJ. ST (212103219)**

For Pantech e learning.,

[illegible]

PANTECH E LEARNING CHENNAI-17

Authorized Signatory

ACKNOWLEDGEMENT

Foremost, I express gratitude to the Almighty for His abundant blessings, facilitating the efficient completion of the project entitled **“MONEY MANAGER USING ANDROID STUDIO”**

I am profoundly grateful to **Dr. A. Sivasankar, Principal, Alpha Arts and Science College, Chennai - 600116,** for his unwavering encouragement and support, which played a pivotal role in the successful completion of this Project Work.

I extend my sincere appreciation to **Dr. C. John Paul, Head of the Department**, for his invaluable guidance, constructive criticism, and constant motivation throughout the duration of this Project.

Special thanks are due to my Guide, M**r. Kirubakaran**, for his unwavering support, expert guidance, and invaluable insights, which significantly contributed to the successful execution of this Project.

I am also indebted to the Teaching and Non-Teaching Staff members of Alpha Arts and Science College, Chennai - 116, whose assistance and cooperation were instrumental in overcoming various challenges encountered during the Project.

Lastly, I express my gratitude to all my friends for their unwavering support, encouragement, and assistance throughout the duration of this Project Work. Their camaraderie and collaboration were invaluable assets in accomplishing the objectives of the Project.

[TARUNMUTHU C S]

[AVINASH S]

[BHARATH RAJ S T]

STARS RISE HERE... AJK COLLEGE OF ARTS AND SCIENCE
An Eco-friendly College
(AUTONOMOUS) AN ISO 21001 : 2018 CERTIFIED INSTITU
Affiliated to Bharathiar University, Coimbatore Approved by Govt. of Tamil Nadu
Recognized by UGC, New Delhi Under Section 2(f) and 12(B)
Palakkad Main Road, Navakkarai, Coimbatore, Tamil Nadu - 641 105. Ph:0422 - 3501700
www.ajkcas.com /ajkcollege /ajkinstitutions

ACCREDITED WITH GRADE A+ NAAC

Azteca University

CHAITANYA (Deemed to be University)
(Approved u/s. 3 of UGC act. 1956 by MHRD, Govt. of India)

5th International Multidisciplinary Conference on

INFORMATION SCIENCE, MANAGEMENT RESEARCH & SOCIAL SCIENCES

(IMCISMRSS - 2024)

Certificate

This is to certify that

Mr. / Ms. /Dr. **TARUNMUTHU CS**

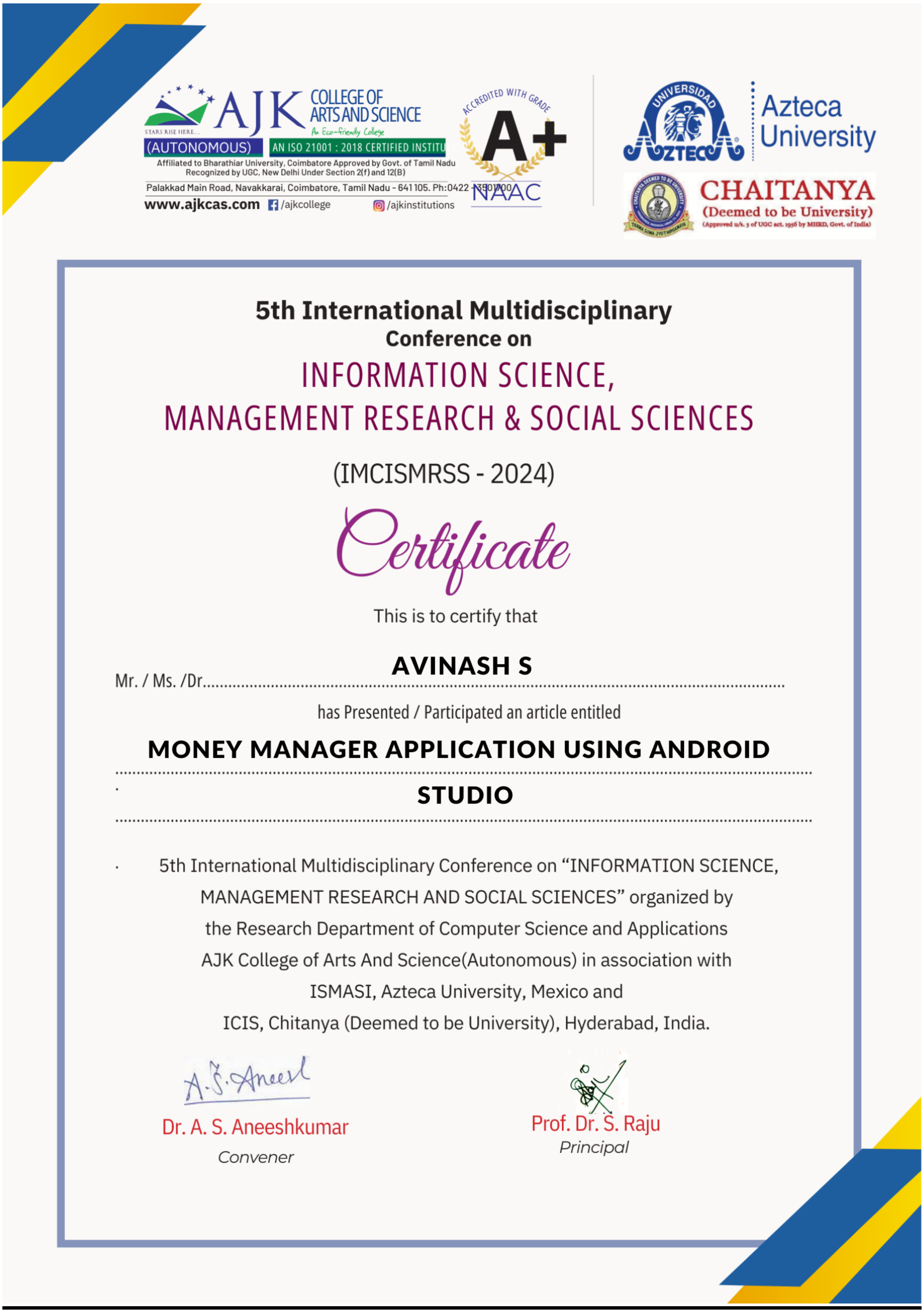
has Presented / Participated an article entitled

MONEY MANAGER APPLICATION USING ANDROID STUDIO

5th International Multidisciplinary Conference on "INFORMATION SCIENCE, MANAGEMENT RESEARCH AND SOCIAL SCIENCES" organized by the Research Department of Computer Science and Applications AJK College of Arts And Science(Autonomous) in association with ISMASI, Azteca University, Mexico and ICIS, Chitanya (Deemed to be University), Hyderabad, India.

[illegible]
Dr. A. S. Aneeshkumar
Convener

[illegible]
Prof. Dr. S. Raju
Principal



AJK COLLEGE OF ARTS AND SCIENCE

STARS RISE HERE...

An Eco-friendly College

(AUTONOMOUS)

AN ISO 21001 : 2018 CERTIFIED INSTITU

Affiliated to Bharathiar University, Coimbatore Approved by Govt. of Tamil Nadu
Recognized by UGC, New Delhi Under Section 2(f) and 12(B)

Palakkad Main Road, Navakkarai, Coimbatore, Tamil Nadu - 641 105. Ph:0422 - 3501700

www.ajkcas.com /ajkcollege /ajkinstitutions

ACCREDITED WITH GRADE A+ NAAC

UNIVERSIDAD AZTECA

Azteca University

CHAITANYA (Deemed to be University) (Approved u/s. 3 of UGC act. 1956 by MHRD, Govt. of India)

5th International Multidisciplinary Conference on

INFORMATION SCIENCE, MANAGEMENT RESEARCH & SOCIAL SCIENCES

(IMCISMRSS - 2024)

Certificate

This is to certify that

Mr. / Ms. /Dr. **AVINASH S**

has Presented / Participated an article entitled

MONEY MANAGER APPLICATION USING ANDROID STUDIO

5th International Multidisciplinary Conference on "INFORMATION SCIENCE, MANAGEMENT RESEARCH AND SOCIAL SCIENCES" organized by the Research Department of Computer Science and Applications AJK College of Arts And Science(Autonomous) in association with ISMASI, Azteca University, Mexico and ICIS, Chitanya (Deemed to be University), Hyderabad, India.

[illegible]

Dr. A. S. Aneeshkumar
Convener

[illegible]

Prof. Dr. S. Raju
Principal

AJK COLLEGE OF ARTS AND SCIENCE
STARS RISE HERE...
An Eco-friendly College
(AUTONOMOUS) AN ISO 21001 : 2018 CERTIFIED INSTITU
Affiliated to Bharathiar University, Coimbatore Approved by Govt. of Tamil Nadu
Recognized by UGC, New Delhi Under Section 2(f) and 12(B)
Palakkad Main Road, Navakkarai, Coimbatore, Tamil Nadu - 641 105. Ph:0422 - 3501700
www.ajkcas.com /ajkcollege /ajkinstitutions

ACCREDITED WITH GRADE A+ NAAC

UNIVERSIDAD AZTECA
Azteca University

CHAITANYA
(Deemed to be University)
(Approved u/s. 3 of UGC act. 1956 by MHRD, Govt. of India)

5th International Multidisciplinary
Conference on

INFORMATION SCIENCE, MANAGEMENT RESEARCH & SOCIAL SCIENCES

(IMCISMRSS - 2024)

Certificate

This is to certify that

Mr. / Ms. /Dr. **BHARATH RAJ ST**

has Presented / Participated an article entitled

MONEY MANAGER APPLICATION USING ANDROID STUDIO

5th International Multidisciplinary Conference on "INFORMATION SCIENCE, MANAGEMENT RESEARCH AND SOCIAL SCIENCES" organized by the Research Department of Computer Science and Applications AJK College of Arts And Science(Autonomous) in association with ISMASI, Azteca University, Mexico and ICIS, Chitanya (Deemed to be University), Hyderabad, India.

[illegible]
Dr. A. S. Aneeshkumar
Convener

[illegible]
Prof. Dr. S. Raju
Principal

ABSTRACT

Money management is an important and unavoidable activity which most people dread. Money management not only involves handling investments but also includes managing multiple accounts and tracking expenses. Each of these activities involves accessing information from different locations and so collecting and consolidating monetary information is not easy.

Currently, there are some stand-alone personal finance applications which address different issues of financial management. There are individual software packages available for portfolio management, budgeting and investment tracking. But each of these applications is limited to only a specific aspect of personal finance. Also, these applications being stand-alone in nature, their usage is limited to the specific system on which they have been installed.

The app places individuals on a firmer financial foundation for both short-term financial management and long-term financial security. Through its intuitive interface, comprehensive expense tracking features, and budget management tools, the app empowers users to take control of their finances, fostering a path towards greater financial stability and success. Hence

PROJECT TITLE	Money Manager Application using Android Studio
LANGUAGES USED	Java
DEVICE SUPPORTED	Android 7.0 +
TOOLS	Android Studio & SQLite
TEAM MANAGER	TARUNMUTHU C S 212103276 AVINASH S 212103217 BHARATH RAJ S T 212103219

TABLE OF CONTENTS

CHAPTER – 1

INTRODUCTION

The “Money Manager Application” has been developed to override the problems prevailing in the practicing of the finance problems. This software is supported to eliminate and in some cases reduce the hardships faced by this existing system. Moreover this system is designed for the particular need of the company to carry out operations in a smooth and effective manner. Thus this application provides the required information in less time and also helps in quicker decision making.

Now a day’s people are concerned about regularity of their daily expenses. This is done mainly for keep a track of the users’ daily expenses to have a control of users’ monthly expenses. We have developed an android application named as “Expense Tracker Application” and this application is used to manage the user‘s daily expenses in a more coherent and manageable way.

Well this application will help us to reduce the manual calculations for their daily expenses and also keep the track of the expenses. With the help of this application, user can calculate his total expenses per day and these results will stored for unique user. As the traditional methods of budgeting, we need to maintain the Excel sheets, Word Documents, notes, and files for the user daily and monthly expenses. There is no as such full-fledged solution to keep a track of our daily expenses easily. Keeping a log in diary is a very monotonous process and also may sometimes lead into problems due to the manual calculations.

Looking on all the above given conditions, we are trying to satisfy the user requirements by building a mobile application which will help them reduces their burdens. “Expense Tracker Application” is an application where one can enter their daily expenses and end of the day, they know their expenses in charts.

1.1. OBJECTIVES:

- The main objective of the Project on Money Manager Application is to manage the details of ones finances, Plans, Payments invoice. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work.
- Money manager application helps create financial stability by tracking expenses and following a plan, a budget makes it easier to pay bills on time, build an emergency fund, and save for major expenses such as a car or home. Overall, a budget puts a person on stronger financial footing for both the day-to-day and the long term
- Design and develop a user friendly system. Easy to use and efficient system.
- yTo develop an accurate and flexible system, it will eliminate data redundancy.Computerization can be helpful as means of saving time & money.
- To provide better graphical user interface.
- Less chances of information leakage.
- The primary objective of the money manager application is to simplify financial management for users. This includes tracking expenses, managing multiple accounts, and monitoring investments in a single, user-friendly platform.
- The app aims to centralize all financial information, providing users with a comprehensive overview of their financial status. This includes bank accounts, credit cards, investments, and other assets.
- The application helps users create and manage budgets effectively. It provides tools for setting financial goals, tracking income and expenses, and analyzing spending patterns to help users stay within their budgetary limits.
- The app allows users to track their investment portfolios, including stocks, mutual funds, and other securities. It provides real-time updates on portfolio performance and helps users make informed investment decisions.
- A key objective of the app is to ensure the privacy and security of users' financial data. It employs robust security measures to protect sensitive information from unauthorized access.
- The app provides users with insights and analytics to help them better

understand their financial habits and make informed financial decisions.This includes spending trends, savings opportunities, and investment performance.

- The app aims to promote financial literacy among users by providing educational resources and tools to help them improve their financial knowledge and skills.
- The app offers flexibility and customization options, allowing users to tailor the app to their specific financial needs and preferences.
- The app encourages users to set and track financial goals, whether it's saving for a vacation, buying a home, or building a retirement fund. It provides tools to help users track their progress towards these goals.
- Ultimately, the objective of the money manager application is to enhance the overall user experience, making financial management simpler, more efficient, and more rewarding for users.

1.2 SCOPE:

- The scope of personal finance is vast. After all, personal finance is all about financial management wherein a person handles his finances by taking into account his budget and savings of the present to stock up resources for the future.
- Personal budgets are usually created to help an individual or a household of people to control their spending and achieve their financial goals. Having a budget can help people feel more in control of their finances and make iteasier for them to not overspend and save money.
- Be easy to understand by the user.
- Be easy to operate.
- Have a good user interface.
- Be expandable.
- Delivered on schedule within the budget.
- Users can track their daily expenses, categorize them, and visualize spending patterns over time.
- Users can track their income sources and manage cash flows.
- The app can provide budgeting tools to help users set and stick to their budgetgoals.

- ➢ Users can link multiple financial accounts (bank accounts, credit cards, investments, etc.) to get a consolidated view of their finances.
- ➢ Users can track the performance of their investment portfolios and get insights into their asset allocation.
- ➢ The app can remind users of upcoming bills and help them avoid late payments.
- ➢ Users can set financial goals (e.g., saving for a vacation, buying a car) and track their progress.
- ➢ The app can provide detailed reports and analytics on users' financial activities.
- ➢ The app should include robust security features to protect users' financial data, such as encryption and multi-factor authentication.
- ➢ A user-friendly interface is essential for ease of use and to ensure that users can navigate the app efficiently.
- ➢ Providing customization options allows users to personalize the app based on their preferences and needs.
- ➢ Integrating with banks and financial institutions can provide users with real-time account information and transaction updates.
- ➢ Including educational resources on personal finance can help users improve their financial literacy.

1.3 TOOLS AND TECHNOLOGIES:

FRONTEND:

XML FILE

XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. It is derived from Standard Generalized Markup Language(SGML). Basically, the XML tags are not predefined in XML. We need to implement and define the tags in XML. XML tags define the data and used to store and organize data. It’s easily scalable and simple to develop. In Android, the XML is used to implement UI-related data, and it’s a lightweight markup language that doesn’t make layout heavy. XML only contains tags, while implementing they need to be just invoked.

BASICS OF USER INTERFACE (UI):

Basically in Android XML is used to implement the UI-related data. So understanding the core part of the UI interface with respect to XML is important. The User Interface for an Android App is built as the hierarchy of main layouts, widgets. The layouts are ViewGroup objects or containers that control how the child view should be positioned on the screen. Widgets here are view objects, such as Buttons and text boxes.

DIFFERENT TYPES OF XML FILES USED IN ANDROID STUDIO:

Different XML files serve different purposes in Android Studio. The list of various XML files in Android Studio with their purposes is discussed below.

1. LAYOUT XML FILES IN ANDROID:

The Layout XML files are responsible for the actual User Interface of the application. It holds all the widgets or views like Buttons, TextViews, EditTexts, etc. which are defined under the ViewGroups.

2. ANDROID MANIFEST.XML FILE:

This file describes the essential information about the application's, like the application's package names which matches code's namespaces, a component of the application like activities, services, broadcast receivers, and content providers. Permission required by the user for the application features also mentioned in this XML file.

3. STRINGS .XML FILE:

This file contains texts for all the TextViews widgets. This enables reusability of code, and also helps in the localization of the application with different languages. The strings defined in these files can be used to replace all hardcoded text in the entire application.

4. THEMES.XML FILE:

This file defines the base theme and customized themes of the application. It also used to define styles and looks for the UI(User Interface) of the application. By defining styles we can customize how the views or widgets look on the User Interface.

5. DRAWABLE .XML FILES:

These are the XML files that provide graphics to elements like custom background for the buttons and its ripple effects, also various gradients can be created. This also holds the vector graphics like icons. Using these files custom layouts can be constructed for EditTexts.

6. COLORS .XML FILE:

The colors.xml file is responsible to hold all the types of colors required for the application. It may be primary brand color and its variants and secondary brand color and its variants. The colors help uphold the brand of the applications. So the colors need to be decided cautiously as they are responsible for the User Experience. The colors need to be defined in the hex code format.

7. DIMENS.XML FILE:

As the file name itself suggests that the file is responsible to hold the entire dimensions for the views. it may be the height of the Button, padding of the views, the margin for the views, etc. The dimensions need to in the format of pixel density(dp) values. Which replaces all the hard-coded dp values for the views. This file needs to be created separately in the values folder.

ADVANTAGES OF XML FILE:

- **DECLARATIVE UI**:

  XML files provide a declarative way to define the UI layout of an Android app, making it easier to visualize and understand the structure of the UI components.

- **SEPARATION OF CONCERNS:**

  By separating the UI layout (XML files) from the code (Java or Kotlin files), XML allows for better organization of the app's components and logic.

- **REUSE AND CONSISTENCY:**

  XML files allow developers to define resources such as strings, colors, and dimensions once and reuse them throughout the app, ensuring consistency in the UI design.

- **EASE OF MAINTENANCE:**

  XML files are text-based and can be easily edited using a text editor or Android Studio's visual editor. This makes it easier to make changes to the UI layout without modifying the code.

- **ACCESSIBILITY:**

  XML files support localization, allowing developers to create different versions of the UI for different languages and regions.

- **PERFORMANCE:**

  XML layouts can be compiled into a more efficient binary format, reducing the overhead of parsing and rendering compared to dynamically creating UI components in code.

- **VERSION CONTROL:**

  XML files can be easily managed using version control systems like Git, allowing developers to track changes and collaborate more effectively.

JAVA FILE

Java is one of the powerful general-purpose programming languages, created in 1995 by Sun Microsystems (now owned by Oracle). Java is Object-Oriented. However, it is not considered as pure object-oriented as it provides support for primitive data types (like int, char, etc). Java syntax is similar to C/C++. But Java does not provide low-level programming functionalities like pointers. Also, Java code is always written in the form of classes and objects. Android heavily relies on the Java programming language all the SDKs required to build for android applications use the standard libraries of Java. If one is coming from a traditional programming background like C, C++, Java is easy to learn.

The Java folder contains the Java source code files. These files are used as a controller for controlled UI (Layout file). It gets the data from the Layout file and after processing that data output will be shown in the UI layout. It works on the backend of an Android application.

ADVANTAGES OF JAVA FILE:

- **PLATFORM INDEPENDENCE:**

  Java is platform-independent, meaning that once you write a Java program, it can run on any device or platform that has a Java Virtual Machine (JVM), including Android.

- **OBJECT - ORIENTED:**

  Java is an object-oriented programming language, which allows for the creation of modular, reusable, and maintainable code. This makes it easier to manage complex applications.

- **RICH API:**

  Java has a rich set of APIs (Application Programming Interfaces) that provide ready-to-use classes and methods for various tasks, such as user interface development, networking, and data storage.

- **COMMUNITY SUPPORT:**

  Java has a large and active community of developers, which means that there are plenty of resources, libraries, and frameworks available to help with Android development.

- **PERFORMANCE:**

  Java is a compiled language, which means that Java code is converted into bytecode, which is then executed by the JVM. This compilation process can lead to better performance compared to interpreted languages.

- **SECURITY:**

  Java has built-in security features, such as automatic memory management (garbage collection) and strong type checking, which help prevent common programming errors and vulnerabilities.

- **COMPATIBILITY:**

  Java is backward compatible, meaning that applications written in older versions of Java can run on newer versions of the Java Virtual Machine without any modifications.

- **SCALABILITY:**

  Java is designed to scale from small applications to large, enterprise-level applications, making it suitable for a wide range of projects.

BACKEND:

SQLITE DATABASE

SQLite is one of the most popular and easy-to-use relational database systems. It possesses many features over other relational databases. Many big MNCs such as Adobe, use SQLite as the application file format for their Photoshop Lightroom product. Airbus, a European multinational aerospace corporation, uses SQLite in the flight software for the A350 XWB family of aircraft.

SQLITE COMMANDS:

In SQLite, **DDL (Data Definition Language)** is used to create and modify database objects such as tables, indices, and views. Some examples of DDL statements in SQLite are:

- **CREATE TABLE:**

  Creates a new table in the database

- **ALTER TABLE:**

  Modifies an existing table in the database

- **DROP TABLE:**

  Deletes a table from the database

- **CREATE INDEX**:

  Creates a new index on a table

- **DROP INDEX:**

  Deletes an index from a table

DML (Data Modification Language) is used to modify the data stored in the database. Some examples of DML statements in SQLite are:

- **INSERT INTO**:

  Inserts a new row into a table

- **UPDATE:**

  Updates the data in one or more rows of a table

- **DELETE FROM:**

  Deletes one or more rows from a table

DQL (Data Query Language) is used to retrieve data from the database. Some examples of DQL statements in SQLite are:

- **SELECT:**

  Retrieves data from one or more tables in the database

- **JOIN:**

  Retrieves data from multiple tables based on a common field

- **GROUP BY:**

  Groups the results of a query by one or more fields

- **HAVING:**

  Filters the results of a query based on a condition

ADVANTAGES OF SQLITE DATABASE:

- **SELF - CONTAINED:**

  SQLite is a self-contained, serverless database engine, meaning it doesn't require a separate server process to function. It operates directly on the database file.

- **ZERO - CONFIGURATION:**

  There is no setup or administration needed to use SQLite. It's easy to install and use, making it ideal for smaller applications or when simplicity is a priority.

- **LIGHT WEIGHT:**

  SQLite is lightweight and has a small footprint, making it suitable for mobile and embedded devices with limited resources.

- **TRANSACTIONAL:**

  SQLite supports ACID (Atomicity, Consistency, Isolation,Durability) transactions, ensuring data integrity and reliability.

- **COMPATIBILITY:**

  SQLite databases are cross-platform and can be easilytransferred between different operating systems and devices.

- **SUPPORTS STANDARD SQL:**

  SQLite supports most of the SQL standard, making iteasy to use for developers familiar with SQL.

- **PERFORMANCE:**

  SQLite is fast and efficient, especially for read operations,making it suitable for applications that require quick access to data.

- **OPEN - SOURCE:**

  SQLite is open-source, which means it's free to use, and itssource code is accessible for customization and optimization.

- **COMMUNITY AND DOCUMENTATION:**

  SQLite has a large and active community ofdevelopers, providing support, documentation, and resource

1.4 FEASIBILITY STUDY:

A feasibility study for a money manager application involves a thorough analysis of various aspects to determine if the project is viable and worth pursuing. This study typically includes assessing the technical, economic, legal, operational, and scheduling aspects of the project. Here's a detailed examination of each aspect:

- **TECHNICAL FEASIBILITY:**

  This aspect assesses whether the technology required for the money manager app is available, feasible, and within budget. It involves evaluating the software and hardware requirements, developmenttools, and expertise needed to build the app. Additionally, it considers the compatibility of the app with various devices and operating systems.

- **ECONOMIC FEASIBILITY**:

  Economic feasibility examines the financial viability of the money manager app project. It involves estimating the costs associated with development, maintenance, marketing, and support. This aspect also considers potential revenue streams, such as app purchases, subscriptions, or in-app purchases. A cost-benefit analysis is conducted to determine if the app will generate enough revenue to justify the investment.

- **LEGAL FEASIBILITY:**

  Legal feasibility assesses whether the money manager app complies with relevant laws, regulations, and industry standards. This aspect examines issues such as data privacy, security, intellectual property rights, andcompliance with financial regulations. It also considers any legal risks or liabilities associated with developing and deploying the app.

- **OPERATIONAL FEASIBILITY:**

Operational feasibility evaluates whether the money manager app can be effectively integrated into the existing operations of the organization or the target market. It considers factors such as user acceptance, ease of use, scalability, and support requirements. This aspect also examines the impact of the app on existing processes and workflows.

- **SCHEDULING FEASIBILITY:**

Scheduling feasibility assesses the timeline for developing and launching the money manager app. It involves creating adetailed project plan that outlines the tasks, milestones, and deadlines for each phase of development. This aspect also considers any potential delays orobstacles that may affect the project timeline.

1.5 FEATURES OF PROJECT:

- **EXPENSE TRACKING:**

  Allows users to track their daily expenses by category (e.g., food, transportation, entertainment) and view spending patterns over time.

- **INCOME TRACKING:**

  Helps users track their income sources and amounts, providing a comprehensive view of their financial situation.

- **BUDGETING:**

  Enables users to set budget limits for different categories and tracks their spending against these limits, providing alerts when they are close to or exceed the budget.

- **BILL REMINDERS:**

  Sends notifications or reminders for upcoming bills or payments, helping users avoid late fees or missed payments.

- **FINANCIAL GOALS:**

  Allows users to set and track financial goals (e.g., saving for a vacation, paying off debt) and provides insights on progress towards these goals.

- **BANK ACCOUNT INTEGRATION:**

  Integrates with users' bank accounts to automatically track transactions and balances, providing a real-time view of their finances.

- **INVESTMENT TRACKING:**

  Allows users to track their investment portfolios, including stocks, mutual funds, and other assets, and provides performance metrics and insights.

- **REPORTS AND ANALYTICS:**

  Provides detailed reports and analytics on income, expenses, and other financial metrics, helping users gain insights into their financial habits and make informed decisions.

- **DATA SECURITY:**

  Ensures that users' financial data is secure and protected, using encryption and other security measures to prevent unauthorized access.

CHAPTER – 2

SYSTEM ANALYSIS

2.1 WHAT IS THE PROBLEM?

- The finance System is working manually. The current system is time consuming for taking notes, distributing money in different categories (Prioritization of money where to spent and how much amount). To manually handle the system was very difficult task. But now-a-days computerization made easy to work.
- The following are the reasons why the current system should be computerized:
  - To increase efficiency with reduced cost.
  - To reduce the burden of paper work.
  - To save time management for recording details of each and every member.
  - To generate required reports easily.

2.2 LIMITATIONS:

- **INACCURATE AND UNREALISTIC**:

A budget is based on assumptions and judgments. If there is any change in the business plan or implementation the whole prediction over the budget plan will get affected. The results of a budget plan,therefore, are always unpredictable and can be inaccurate sometimes.

- **INFLEXIBLE**:

A budget is formed depending on certain policies of an institutionor goals of an individual that leads to decision-making. However, if there is any need to review the financial status considering any change in the marketthere is no way the budget can be altered.

- **FINANCE ORIENTED**:

The budget does not support the interests and requirementsof the people. It is more profit-oriented which is more quantitative while the needs of the people are more qualitative in nature.

- **TIME - CONSUMING:**

The process of planning a budget Or budgeting is a time-consuming affair. It needs to consider all possible aspects of an organizationor an individual before ensuring any expenditure or spending towards a particular goal.

- **CONFLICTS:**

The failure of a budget plan can result in a lot of arising tensions and rifts within the company that ultimately get reflected by the inefficient running of the organization

- **ACCURACY:**

Money manager apps rely on users to manually input their financial transactions. Errors in data entry or incomplete information can lead to inaccurate financial reports and insights.

- **COST:**

While many money manager apps offer free versions, some features maybe locked behind a paywall. Additionally, users may incur data charges for accessing the app or syncing data over the internet.

CHAPTER – 3

SYSTEM REQUIREMENTS STUDY

3.1 USER CHARACTERISTICS:

- **TECH - SAVVINESS:**

  Users' comfort and familiarity with technology will influence the complexity and usability of the app. For less tech-savvy users, the app should be intuitive and easy to navigate, with clear instructions and minimal technical jargon.

- **FINANCIAL LITERACY:**

  Users' knowledge and understanding of financial conceptswill impact the complexity of the app's features and the level of detail provided in financial reports and analysis. The app should offer educational resources or guidance for users with varying levels of financial literacy.

- **BUDGETING EXPERIENCE:**

  Users' experience with budgeting and financial management will influence the level of detail and customization options they expect from the app. Novice users may prefer simple, guided budgeting tools,while more experienced users may require advanced features for detailed financial planning.

- **DATA PRIVACY CONCERNS:**

  Users' attitudes towards data privacy and security will affect their willingness to share sensitive financial information with the app. The app should prioritize data security and provide transparent information about how user data is collected, stored, and used.

- **FINANCIAL GOALS AND PRIORITIES:**

  Users' financial goals and priorities will influence the app's design and features. For example, users saving for retirement may require different features than users managing day-to-day expenses or paying off debt.

- **DEVICE AND PLATFORM PREFERENCES:**

Users' preferences for specific devices (e.g.,smartphones, tablets) and operating systems (e.g., Android, iOS) will impact the app's compatibility and accessibility. The app should be optimized for the most popular devices and platforms among its target audience.

- **ACCESSIBILITY NEEDS:**

Users with accessibility needs, such as visual or hearing impairments, will require the app to be compatible with assistive technologiesand to provide alternative ways of accessing information (e.g., screen readers,voice commands).

3.2 SOFTWARE SPECIFICATIONS:

- Android Operating System.
- Android minimum version Android 5.0 Lollipop.
- For developing application, we use android studio and for database we are use firebase.

3.2.1 ANDROID STUDIO:

Android Studio is the official Integrated Development Environment (IDE) for Android App development. It is a powerful tool that allows developers to build high-quality applications for the Android platform. It has complete tools for the process of Android App development. From writing code to testing and deployment, ANdroid studio has all the functionalities for developers to develop an Android App.

From beginner to advanced developers, it is a very common IDE for developing high-quality Android applications.

3.2.1.1 FEATURES OF ANDROID STUDIO:

- ➢ A flexible Gradle-based build system
- ➢ A fast and feature-rich emulator
- ➢ A unified environment where you can develop for all Android devices
- ➢ Live Edit to update composables in emulators and physical devices in real time
- ➢ Code templates and GitHub integration to help you build common app features and import sample code
- ➢ Extensive testing tools and frameworks
- ➢ Lint tools to catch performance, usability, version compatibility, and other problems
- ➢ C++ and NDK support

- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

Android Studio Icon

3.2.2 SQLITE:

- SQLite does not require a separate server process or system to operate (serverless).
- SQLite comes with zero-configuration, which means no setup or administration needed.
- A complete SQLite database is stored in a single cross-platform disk file.
- SQLite is very small and light weight, less than 400KiB fully configured or less than 250KiB with optional features omitted.
- SQLite is self-contained, which means no external dependencies.
- SQLite transactions are fully ACID-compliant, allowing safe access from multiple processes or threads.

3.3 HARDWARE SPECIFICATIONS:

- Android Based Smart phone
- Minimum 2 GB RAM required.
- App size space required in ROM.

3.4 ASSUMPTIONS AND DEPENDENCIES:

- Project will work for a long time and user will adopt it.
- Project will work with very less maintenance requirement.
- The database update made by the system will always leave the system in consistent state.
- There may be some small problems, which will not affect the system performance, and these will be removed easily.
- This system interface is used to give access to the user for the system, and mean while maintaining the security of the system.

CHAPTER – 4

SOFTWARE SYSTEM ATTRIBUTES

4.1 USABILITY:

➢ **INTUITIVE USER INTERFACE:**

The application should have a clean and intuitive user interface that is easy to navigate. Important features such as expense tracking, budgeting, and reporting should be easily accessible.

➢ **SIMPLIFIED DATA ENTRY:**

Users should be able to quickly and easily enter their financial transactions. The application can use features such as auto-complete, pre-defined categories, and quick-add buttons to streamline the data entry process.

➢ **VISUAL REPRESENTATION OF DATA:**

The application should use charts, graphs, andother visual elements to present financial data in a clear and easy-to- understand manner. This can help users quickly grasp their financial situation and make informed decisions.

➢ **CUSTOMIZATION OPTIONS:**

Users should be able to customize the application to suit their individual needs and preferences. This can include customizing categories, setting budget limits, and choosing which financial metrics to track.

➢ **NOTIFICATION AND REMINDER FEATURES:**

The application should include features that remind users of upcoming bills, payments, or budget limits. Notifications can help users stay on track with their financial goals.

➢ **DATA SECURITY AND PRIVACY:**

Users should feel confident that their financial data is secure and protected. The application should use encryption and other security measures to ensure that sensitive information is not compromised.

4.2 EFFICIENCY:

- **PERFORMANCE:**

  The application should be responsive and fast, especially when loading data, generating reports, or performing calculations. Optimizing database queries and minimizing network requests can help improve performance.

- **RESOURCE MANAGEMENT:**

  The application should manage resources such as memory, CPU, and battery life efficiently. This can involve optimizing code, reducing unnecessary background processes, and using caching mechanisms.

- **DATA SYNCHRONIZATION:**

  If the application supports synchronization with external sources (e.g., bank accounts, cloud storage), it should synchronize data efficiently to minimize delays and ensure data consistency.

- **OFFLINE FUNCTIONALITY:**

  The application should be able to work offline or with limited connectivity. This can involve caching data locally and syncing changes when connectivity is restored.

- **OPTIMIZED UI/UX:**

  The user interface should be designed for efficiency, with intuitive navigation, clear labeling, and minimal steps required to perform common tasks. This can help users complete tasks quickly and easily.

- **AUTOMATION:**

  The application can use automation to streamline repetitivetasks, such as categorizing expenses or generating reports. This can save users time and effort in managing their finances.

- **ERROR HANDLING:**

  The application should handle errors gracefully, providing informative error messages and guiding users on how to resolve issues. This can help prevent frustration and improve the overall user experience.

- **SECURITY AND PRIVACY:**

  While ensuring security and privacy is primarily about protecting user data, efficient implementation of security measures can also contribute to the overall efficiency of the application by minimizing the impact of security checks on performance.

4.3 MAINTAINABILITY:

- **MODULARITY:**

  The application should be organized into modular components that can be easily understood, modified, and tested independently. This can help reduce the impact of changes and make it easier to add new features orfix bugs.

- **CODE READABILITY**:

  The code should be well-structured, well-commented, and adhere to coding standards. This can make it easier for developers to understand and maintain the codebase.

- **TESTING:**

  The application should have a robust testing framework in place, including unit tests, integration tests, and possibly automated UI tests. Thiscan help ensure that changes do not introduce new bugs or regressions.

- **DEPENDENCY MANAGEMENT:**

The application should manage dependencies carefully, ensuring that they are up-to-date and compatible with the rest of the application. This can help prevent compatibility issues and security vulnerabilities.

- **PERFORMANCE MONITORING:**

The application should include monitoring tools that can track performance metrics such as response times, resource usage, anderror rates. This can help identify and address performance issues before they impact users.

- **SECURITY UPDATES:**

The application should be regularly updated to address security vulnerabilities and comply with the latest security standards. This can help protect user data and maintain user trust.

CHAPTER – 5

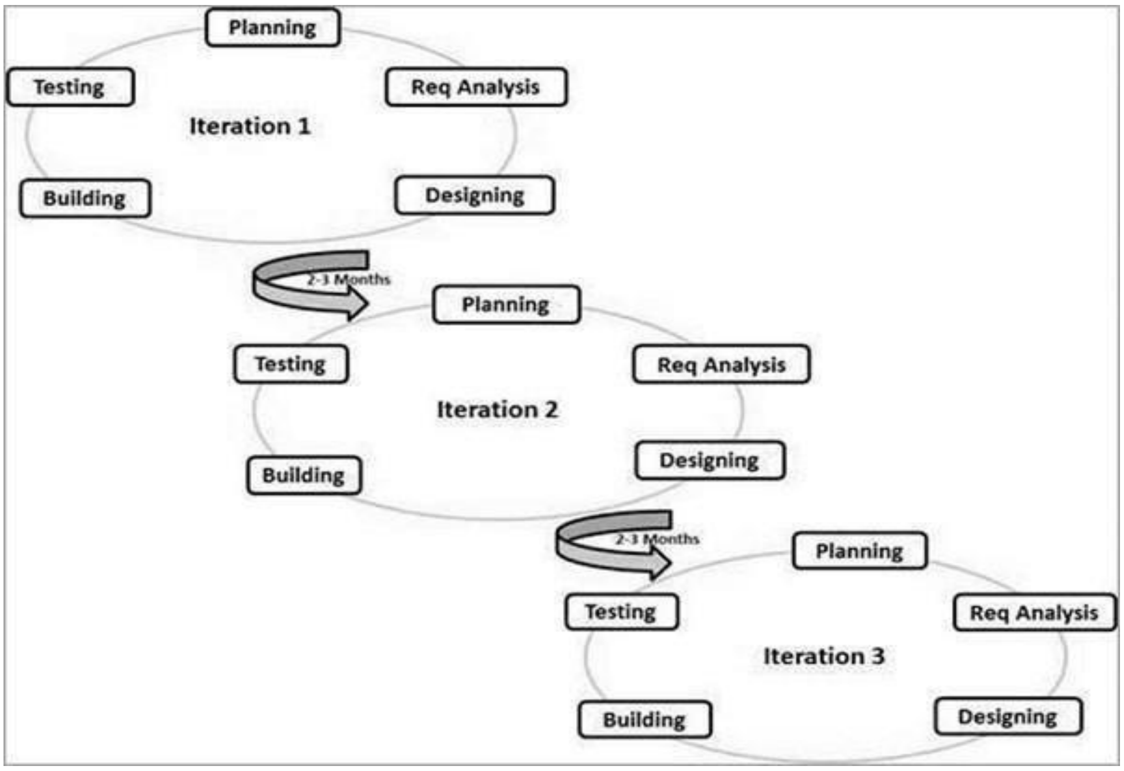
PROJECT MANAGEMENT

5.1 AGILE MODEL:

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

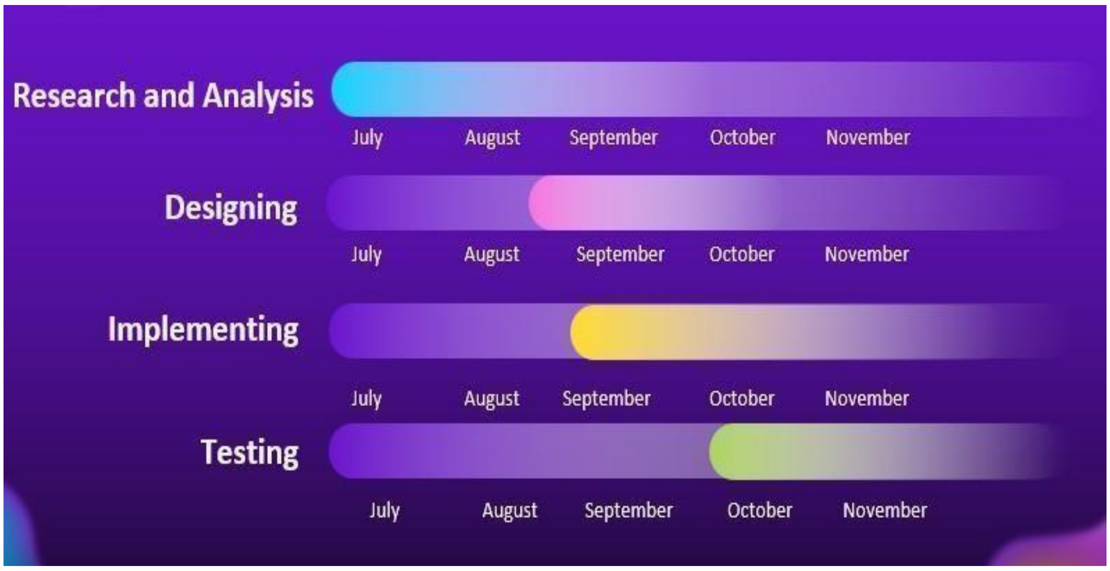
Iterative approach is taken and working software build is delivered after each iteration.Each build is incremental in terms of features; the final build holds all the features required by the customer.

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.



5.1 AGILE MODEL

5.2 GANTT CHART:



5.2 GANTT CHART

CHAPTER – 6

IMPLEMNTATION AND TESTING

6.1 IMPLEMENTATION ENVIRONMENT:

The backbone and the main aspect from the software side of the project was to run theclient and server-side code, perform the Android JAVA and making the mobile application more user friendly by XML (Extensible Markup Language). For better expand our idea we draw diagrams and draw a UI design for our project. So, the implementation was carried out in such a way that co-relation of different files would be done easily. Hence, the use of an IDE was a must. We use android studio to build our android application. Implementation of the code was done all together after we researched and sorted the image data that we obtained. Basic approach that we used was the partial compilation method and went on integrating the small amount of the code to larger ones, module by module.

6.2 TESTING METHODS:

My team used to write/code the module by module for one each, and we had dividedthe test cases accordingly to the if else ladders that we had assigned in the code. Method adapted by us was that we would initially check the code for its functional testing. After it successfully passed that test we would add the detailed functions into it. After the functional testing of every module, unit testing was carried out by the developers of the specific module themselves. After the successful completion of thanwe went with the method of integration testing And successfully cleared the completion of the code and at the end we carried out the quality testing and finally theusability testing.

We have chosen these techniques because of following reasons:

- Easy to find out errors
- Make it efficient for use
- Optimize the code

6.3 TEST CASES:

Throughout the completion of the project, we carried out various testcases of all above mentioned tests. And we even managed to gather successful screenshots of the usability testing that happens at the end of the completion of designing and landed out with successful satisfactory quality results as displayed below:

TABLE 6. 1 TEST SUITE FOR BUDGET PLANS

1	Test Suite for Budget Plans
	Test Case 1: Verify that Budget Plan is not in negative and zero. Test Case 2: Verify that deadline date is not of past dates. Test Case 3: Verify that add plan should not be empty. Test Case 4: Verify that passed out dates should be in red zone(deadline of plan has passed).

TABLE 6. 2 TEST SUITE FOR BUDGET CATEGORY

2	Test Suite for Budget Category
	Test Case 1: Verify that each category should have their name with budget. Test Case 2: Verify if a user is able to go on next page.

TABLE 6. 3 TEST SUITE FOR MONTHS EXPENSES

3	Test Suite for Months Expenses
	Test Case 1: Verify user can select or go ahead with their requirement in application. Test Case 2: Verify that all button are clickable or not. Test Case 3: Verify that user can see their category, Product name, product price and the date of product purchased. Test Case 4: Verify that add product add on in category wise.

	Test Case 5: Verify that total expenses should be display at bottom of page.

TABLE 6. 4 TEST SUITE FOR BUDGET ANALYSIS

4	Test Suite for Budget Analysis
	Test Case 1: Verify that pie chart of budget category should be proper. Test Case 2: Verify that select button are clickable or not. Test Case 3: Verify that pie chart should be visible by monthly expenses. Test Case 4: Verify that each section of pie chart should be properly showing total value, percentage wise of category.

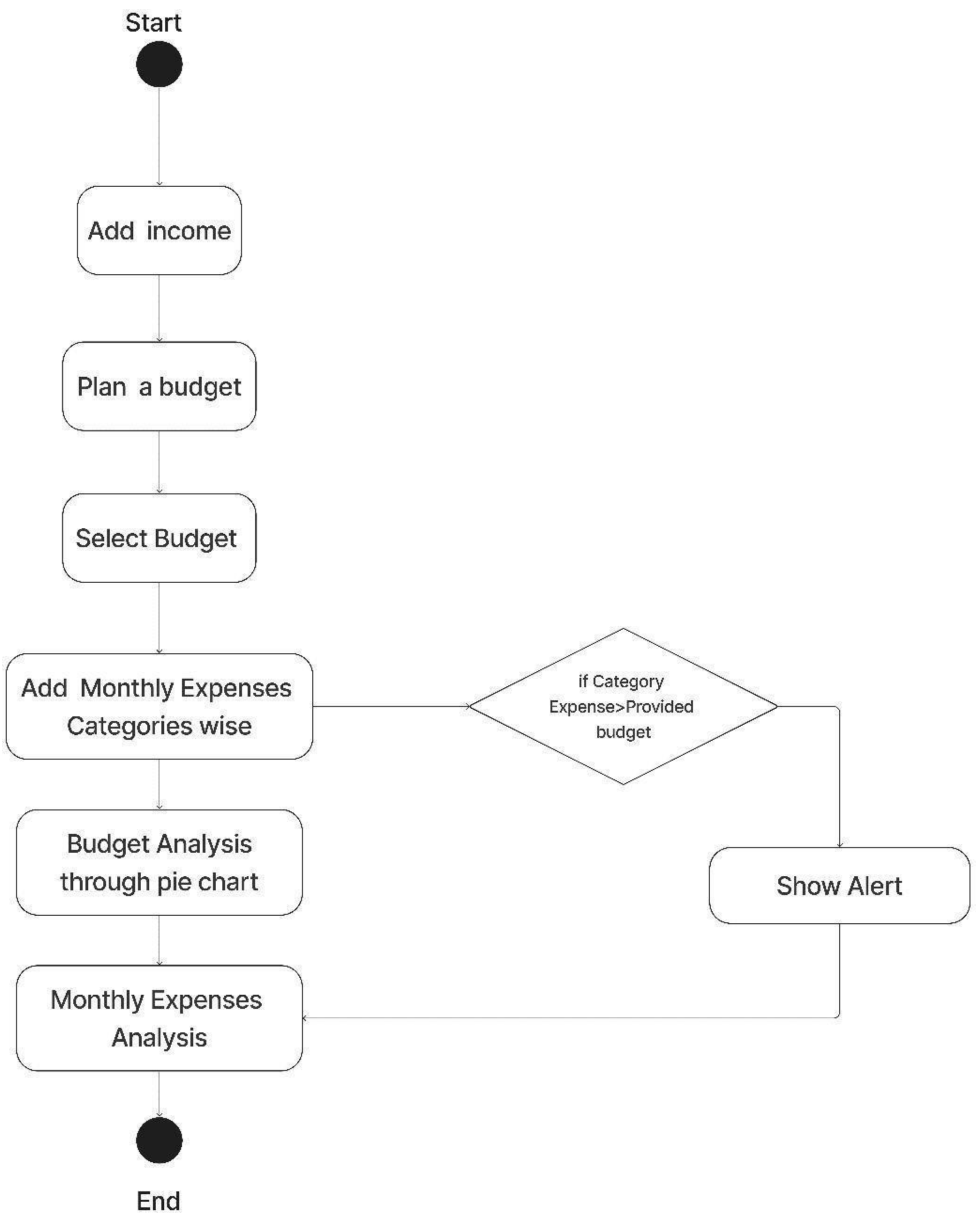
TABLE 6. 5 TEST SUITE FOR MONTHLY EXPENSES

5	Test Suite for Monthly Expenses.
	Test Case 1: A Pay button will appear on your app. Test Case 2: Verify that click the button and make a test transaction to ensure the integration is working as expected. .

CHAPTER – 7

SYSTEM DESIGN

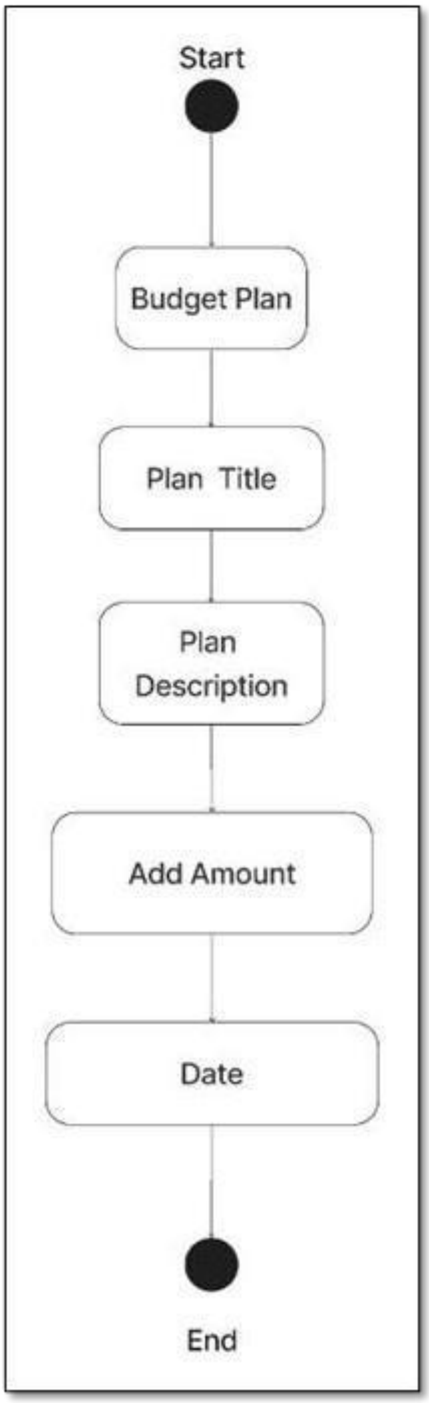
7.1 FLOW DIAGRAM:



Start
Add income
Plan a budget
Select Budget
Add Monthly Expenses Categories wise
if Category Expense>Provided budget
Show Alert
Budget Analysis through pie chart
Monthly Expenses Analysis
End

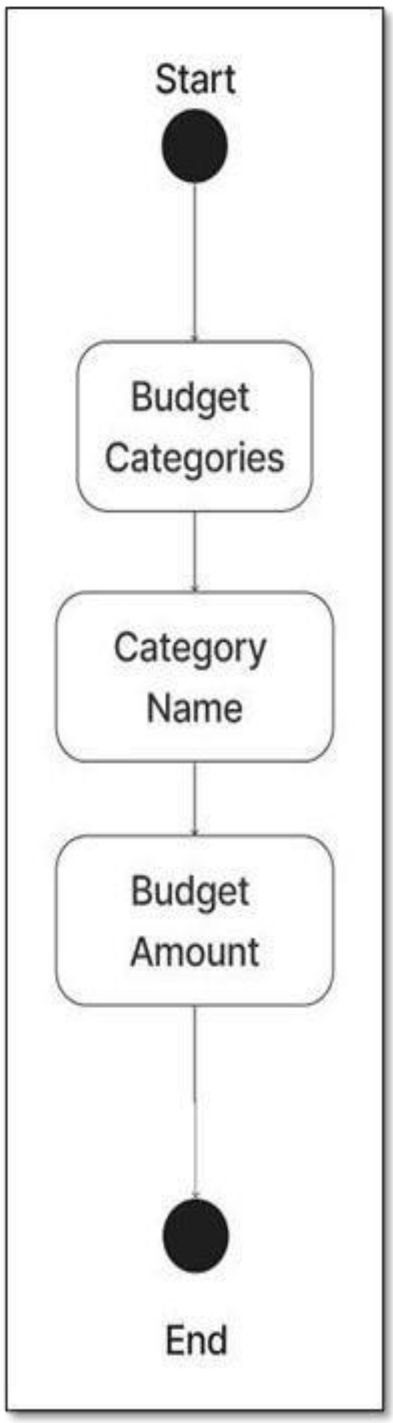
7.1 FLOW DIAGRAM

7.1.1 FLOW CHART OF BUDGET PLAN:



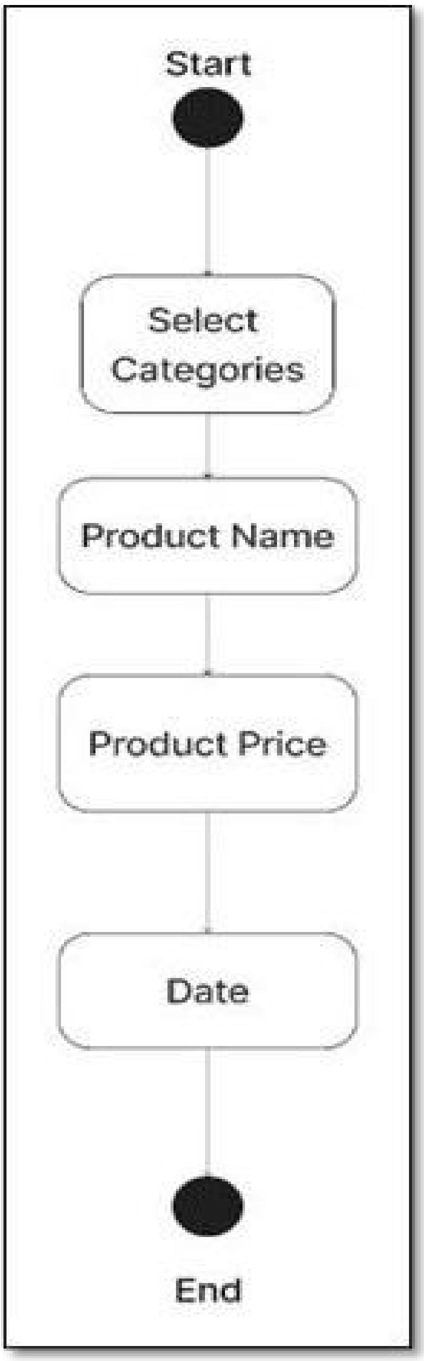
7.1.1 FLOWCHART OF BUDGET PLAN

7.1.2 FLOW CHART OF BUDGET CATEGORY:



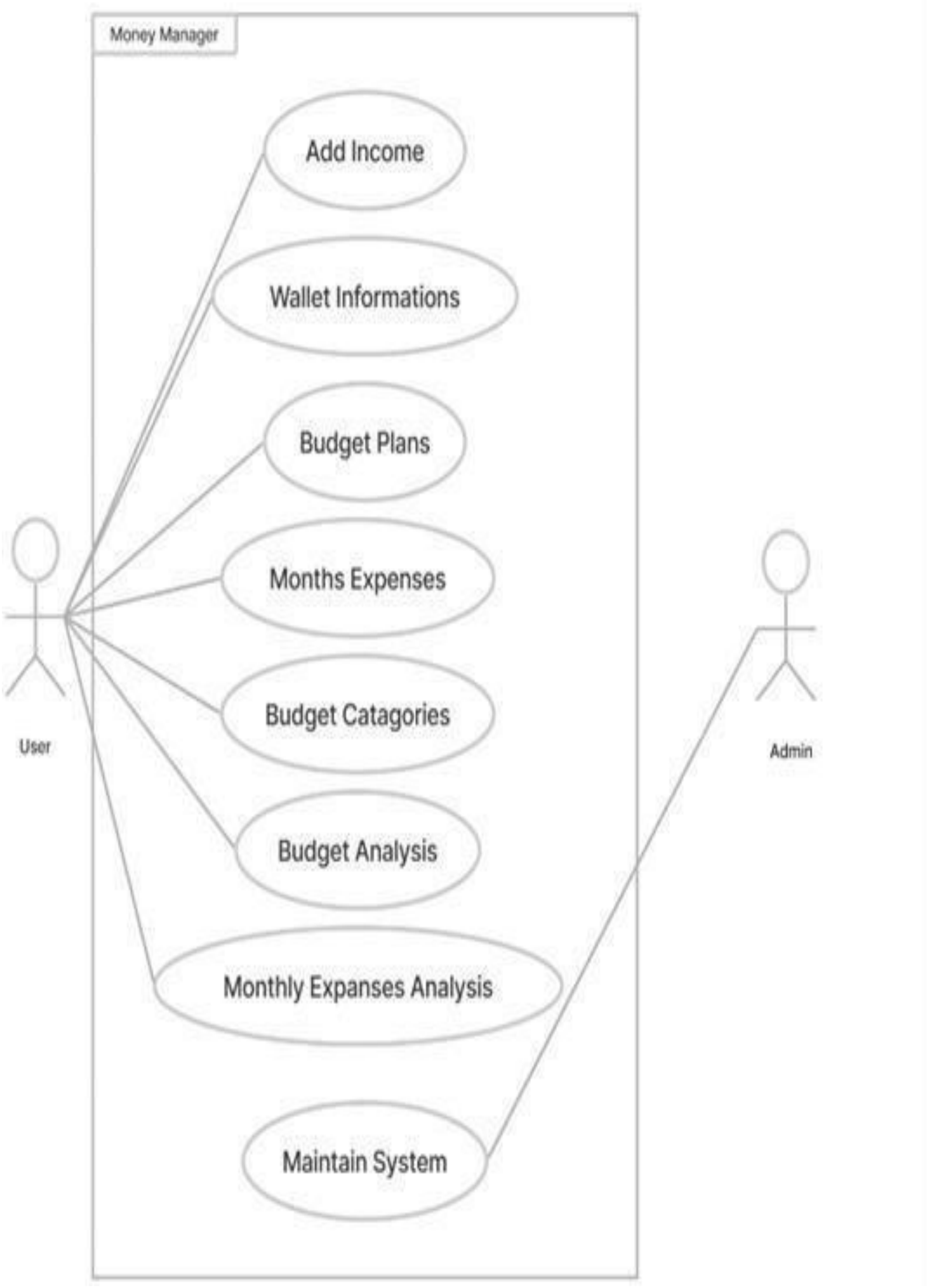
7.1.2 FLOWCHART OF BUDGET CATEGORIES

7.1.3 FLOW CHART OF MONTHLY EXPENSES:



7.1.3 FLOWCHART OF MONTHLY EXPENSES

7.2 USE CASE DIAGHRAM:



7.2 USE CASE DIAGRAM

CHAPTER - 8

SOURCE CODE

8.1 ACTIVITY MAIN.XML:

```
<?xml version="1.0" encoding="utf-8"?>


<androidx.drawerlayout.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true" tools:openDrawer="start">

    <include layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />


 </androidx.drawerlayout.widget.DrawerLayout>
```

8.2 ACTIVITY PLAN INFO.XML:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent" android:layout_height="match_parent"
    tools:context=".budgetplanner.activities.PlanInfoActivity">
 <!--(String title, String description, double moneyNeeded, Date deadline)-->
 <ScrollView

            android:layout_width="match_parent"
            android:layout_height="match_parent">
        <LinearLayout
             android:layout_width="match_parent"
             android:layout_height="match_parent"
             android:orientation="vertical"
             android:padding="5dp">
           <com.google.android.material.textfield.TextInputLayout

              android:layout_width="match_parent"
              android:layout_height="wrap_content"
              android:hint="@string/plan_title"
              app:startIconDrawable="@drawable/icon_shopaholic"
              app:counterEnabled="true"
              app:counterMaxLength="50"
             app:startIconContentDescription="@string/content_description_
end_icon">
```

```
<com.google.android.material.textfield.TextInputEditText
                android:id="@+id/plan_title_tf"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:inputType="textAutoCorrect"/>

</com.google.android.material.textfield.TextInputLayout>
          <androidx.legacy.widget.Space android:layout_width="match_parent"
             android:layout_height="10.0dp"/>
          <com.google.android.material.textfield.TextInputLayout
             android:layout_width="match_parent"
             android:layout_height="wrap_content"
             android:hint="@string/plan_desc"
             app:startIconDrawable="@drawable/icon_shopaholic"
             app:counterEnabled="true" app:counterMaxLength="50"
             app:endIconMode="clear_text"
             app:startIconContentDescription="@string/content_description_
end_icon">

<com.google.android.material.textfield.TextInputEditText
                android:id="@+id/plan_desc_tf"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:inputType="textAutoCorrect"/>
```

```
 </com.google.android.material.textfield.TextInputLayout>
              <androidx.legacy.widget.Space
              android:layout_width="match_parent"
              android:layout_height="10.0dp"/>

<com.google.android.material.textfield.TextInputLayout
                   android:layout_width="match_parent"
                   android:layout_height="wrap_content"
                   android:hint="@string/needed_budget"
                   app:startIconDrawable="@drawable/ic_dollar"
                   app:endIconMode="clear_text" app:prefixText="$"
                   app:startIconContentDescription="@string/content_description_
 end_icon">

 <com.google.android.material.textfield.TextInputEditText
                        android:id="@+id/plan_budget_tf"
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:inputType="numberDecimal"/>

 </com.google.android.material.textfield.TextInputLayout>

<androidx.legacy.widget.Space
android:layout_width="match_parent"
android:layout_height="10.0dp"/
```

```
  <TextView android:layout_marginLeft="10dp"
     android:layout_marginRight="10dp"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:text="@string/plan_deadline_date"
     android:textSize="16.0dp"/>


<DatePicker android:layout_marginTop="10dp"
     android:id="@+id/plan_date_picker"
     android:layout_gravity="center_horizontal"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"/>


<androidx.legacy.widget.Space
android:layout_width="match_parent"
android:layout_height="20.0dp"/>


<Button
          android:id="@+id/submitBu"
          android:onClick="onClickAddPlan"
          android:padding="20dp"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@string/add_plan"
```

```
                    android:textAllCaps="false"
                    android:background="@color/colorPrimary
                    android:textColor="#fff" android:textSize="20.0dp"
                    android:textStyle="bold"
                    android:layout_gravity="center_horizontal"/>

            </LinearLayout>

        </ScrollView>

    </LinearLayout>
```

8.3 ACTIVITY CATEGORY TRANSACTION.XML:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"

   android:layout_height="match_parent"

    android:orientation="vertical"
    tools:context=".transactions.activities.ListingCategoryTransactionsActivity">


    <TextView android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/monthly_expenses_in"
        android:textSize="20.0sp" android:padding="10dp"
        android:layout_margin="5dp"/>


    <TextView android:id="@+id/selected_category_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="CategoryName"
        android:textSize="20.0sp"
        android:textAlignment="center"
        android:padding="10dp"
        android:layout_margin="5dp"
        android:textColor="#000"/>
    <ListView android:background="#d3d2d2"
        android:id="@+id/transaction_lv"
        android:layout_margin="5dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:dividerHeight="2dp"/>


</LinearLayout>
```

8.4 ACTIVITY ADDING BUDGET CATEGORY.XML:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"

    android:layout_height="match_parent"

    android:orientation="vertical"

    tools:context=".budget_categories.activities.AddingBudgetCategoryActivity">


    <HorizontalScrollView
    android:layout_margin="5dp"
    app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toTopOf="@+id/name_layout"
      android:layout_width="match_parent"
      android:layout_height="100dp">


        <LinearLayout android:layout_width="match_parent"
            android:layout_height="match_parent">


            <ListView android:id="@+id/list_view_id"
                android:layout_width="match_parent"
                android:layout_height="100dp"
                android:layout_weight="1"
                android:horizontalSpacing="5dp"
                android:paddingTop="10dp"
                android:verticalSpacing="5dp" />
        </LinearLayout>
    </HorizontalScrollView>
    <com.google.android.material.textfield.TextInputLayout
app:layout_constraintBottom_toBottomOf="@id/name_layout"
app:layout_constraintBottom_toTopOf="@+id/budget_layout"
        android:id="@+id/name_layout" android:layout_margin="5dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/category"
        app:startIconDrawable="@drawable/icon_shopaholic"
        app:counterEnabled="true"
        app:counterMaxLength="50"

app:startIconContentDescription="@string/content_description_ end_icon">

     <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/category_name"
            android:layout_width="match_parent"
               android:layout_height="wrap_content"
               android:inputType="textAutoCorrect"
            />
    </com.google.android.material.textfield.TextInputLayout>
    <com.google.android.material.textfield.TextInputLayout
```

```
                app:layout_constraintTop_toBottomOf="@+id/name_layout"
                android:id="@+id/budget_layout"

               id.material.textfield.TextInputLayout>

    <Button
        android:id="@+id/add_Budget_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:background="@color/colorPrimary"
        android:padding="20dp" android:layout_margin="20dp"
        android:text="@string/submit" android:textAllCaps="false"
        android:textColor="#fff" android:textSize="20.0dp"
        android:textStyle="bold"/>

</LinearLayout>
```

8.5 ACTIVITY ADD TRANSACTION.XML:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"

    android:layout_height="match_parent"
    tools:context=".transactions.activities.AddingTransactionActivity">


    <ScrollView android:layout_width="match_parent"
        android:layout_height="match_parent">


        <LinearLayout android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical" android:padding="20dp">


            <TextView android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/product_category_hint"
            android:textSize="16.0dp"/>


            <Spinner android:id="@+id/product_category_spinner"
            android:layout_width="match_parent"
            android:layout_margin="10dp"
            android:padding="10dp"
```

```
                android:textAlignment="center"
                android:layout_height="wrap_content"/>
            <com.google.android.material.textfield.TextInputLayout
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="@string/product_name"
                    app:startIconDrawable="@drawable/icon_shopaholic"
                    app:counterEnabled="true" app:counterMaxLength="50"
                    app:endIconMode="clear_text"
                    app:startIconContentDescription="@string/content_description_
end_icon">
        <com.google.android.material.textfield.TextInputEditText
                    android:id="@+id/product_name_textField"
                    android:layout_width="match_parent" android:layout_height="wrap_content"
                    android:inputType="textAutoCorrect"/>


        </com.google.android.material.textfield.TextInputLayout>

                <androidx.legacy.widget.Space android:layout_width="match_parent"
                    android:layout_height="20.0dp"/>
        <com.google.android.material.textfield.TextInputLayout

                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="@string/product_price"
                    app:startIconDrawable="@drawable/ic_dollar"
                    app:endIconMode="clear_text"

                    app:prefixText="$"

                    app:startIconContentDescription="@string/content_description_
end_icon">

<com.google.android.material.textfield.TextInputEditText

                        android:id="@+id/product_price_textField"

                        android:layout_width="match_parent"

                        android:layout_height="wrap_content"

                        android:inputType="numberDecimal"/>


</com.google.android.material.textfield.TextInputLayout>

                <androidx.legacy.widget.Space

                android:layout_width="match_parent"

                android:layout_height="20.0dp"/>


                <TextView android:layout_width="match_parent"

                    android:layout_height="wrap_content"

                    android:text="@string/transaction_date_hint"

                    android:textSize="16.0dp"/>


                <DatePicker  android:layout_marginTop="5dp"

                    android:id="@+id/date_picker"

                    android:layout_gravity="center_horizontal"

                    android:layout_width="wrap_content"

                    android:layout_height="wrap_content"/
                <androidx.legacy.widget.Space
                 android:layout_width="match_parent"
                 android:layout_height="30.0dp"/>

              <Button
                 android:padding="20dp" android:layout_width="wrap_content"
                 android:layout_height="wrap_content"
                 android:text="@string/add_transaction"
                 android:textAllCaps="false"
                 android:background="@color/colorPrimary"
                 android:textColor="#fff" android:textSize="20.0dp"
                 android:textStyle="bold"
                 android:onClick="onClickAddTransaction"
                 android:layout_gravity="center_horizontal"/>

          </LinearLayout>
      </ScrollView>

  </LinearLayout>
```

8.6 MAIN ACTIVITY. JAVA:

```
package com.example.expensestracker.mainactivity;


import android.content.Intent; import

android.os.Bundle; import

android.util.Log;

import android.view.MenuItem; import

android.view.View; import

android.view.Menu;

import com.example.expensestracker.budget_categories.ui.BudgetCategoryInfoFragment;

import com.example.expensestracker.R;

import com.example.expensestracker.budget_categories.BudgetCategoryManager;

import com.example.expensestracker.budgetplanner.PlansManager;

import com.example.expensestracker.dbmanagment.BudgetCategoriesTable;

import com.example.expensestracker.dbmanagment.DatabaseController;

import com.example.expensestracker.globalOperations.DateStringFormatter;

import com.example.expensestracker.monthestracker.MonthExpenses;

import  com.example.expensestracker.monthestracker.MonthsTracker;

import com.example.expensestracker.transactions.TransactionManager;

import com.example.expensestracker.transactions.activities.AddingTransactionActivity;

import com.example.expensestracker.usermoney.UserWallet;

import com.example.expensestracker.usermoney.ui.AddingIncomeFragment;

import com.example.expensestracker.usermoney.ui.UserWalletFragment;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import com.google.android.material.navigation.NavigationView;
import androidx.annotation.NonNull;
import androidx.fragment.app.FragmentManager;

import androidx.navigation.NavController;

import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import java.util.Calendar;
public class MainActivity extends AppCompatActivity {


    private AppBarConfiguration mAppBarConfiguration;
    DatabaseController databaseController;
    @Override
    protected void onCreate(Bundle savedInstanceState)
        {super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main); try {
            databaseController = new DatabaseController(this);
            BudgetCategoryManager.LoadCategoryEnvelop();
            MonthsTracker.checkUpdates(this); TransactionManager.loadTransactions();
            PlansManager.loadPlans(); UserWallet.loadUserMoney(this);


            Toolbar toolbar = findViewById(R.id.toolbar); setSupportActionBar(toolbar)
    }

            //Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)

            //.setAction("Action", null).show();

DrawerLayout drawer = findViewById(R.id.drawer_layout);

NavigationView navigationView = findViewById(R.id.nav_view);

            // Passing each menu ID as a set of Ids because each

            // menu should be considered as top level destinations.

            mAppBarConfiguration = new
            AppBarConfiguration.Builder(R.id.nav_home, R.id.nav_budgetManager ,
            R.id.nav_budget_analysis ,R.id.nav_budgetPlanner
            ,R.id.nav_monthly_expenses_analysis)

             .setDrawerLayout(drawer)

             .build();


            NavController navController = Navigation.findNavController(this,
                        R.id.nav_host_fragment);

            NavigationUI.setupActionBarWithNavController(this,navController,
                        mAppBarConfiguration);

        NavigationUI.setupWithNavController(navigationView,navController);

        }

        catch (Exception e)

        {

                e.printStackTrace();

        }

    }
```

```
    @Override

    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it is present.

        getMenuInflater().inflate(R.menu.main, menu); return true;

    }

    @Override

    public boolean onOptionsItemSelected(@NonNull MenuItem item)

    {

        switch (item.getItemId())

        {

            case R.id.add_income:

                addIncome();

                return true;

            case            R.id.user_wallet:

                showUserWallet();  return

                true;

        }

        return super.onOptionsItemSelected(item);

    }
    @Override

    public boolean onSupportNavigateUp()

        {NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);

        return NavigationUI.navigateUp(navController, mAppBarConfiguration)

                || super.onSupportNavigateUp();

    }
    private void addIncome() {

        FragmentManager fragmentManager = getSupportFragmentManager();

        AddingIncomeFragment addingIncomeFragment = new
AddingIncomeFragment();


        addingIncomeFragment.show(fragmentManager,"add    income");

    }

    private void showUserWallet()

        { FragmentManager fragmentManager

        =

getSupportFragmentManager();

        UserWalletFragment userWalletFragment = new UserWalletFragment();

        userWalletFragment.show(fragmentManager,"add income");

    }
    // TODO // Database contain(plans , transactions , budget categories)

}
```

8.7 TRANSACTION TABLE.JAVA:

```
package com.example.expensestracker.dbmanagment;


public class TransactionsTable

{

      public static final String TableName = "transactions";

       /*

      static final String createTransactionTableSQLQuery =

                              "CREATE TABLE IF NOT EXISTS transactions \n"+

                              "( id INT NOT NULL Autoincrement ,\n" + "

                                productName VARCHAR(100) NOT NULL ,\n" +

                             "price DOUBLE NOT NULL ,\n" +

                              "buyDate VARCHAR(100) NOT NULL ,\n" + "

                              budgetID INT(100) NOT NULL ,\n" +

                              "PRIMARY KEY (id)," +

                               "CONSTRAINT FK FOREIGN KEY (budgetID)
                               REFERENCES budget_categories(id)" +");";*/

      static final String DropTransactionsTable = "DROP TABLE IF EXISTS " +
TableName;

      public  static final String colID = "colID";

      public static final String colProductName = "productName";

      public  static final String colPrice = "price";
public static final String colBuyDate = "buyDate"; public  static final

String colBudgetID = "budgetID";

static final String createTransactionTableSQLQuery

=  "CREATE TABLE IF NOT EXISTS " + TableNam+

"(\n " +"colID Integer Primary key Autoincrement ,\n" +

colProductName +" varchar(255),\n" +


          colPrice +" double,\n" + colBuyDate +"

  varchar(255),\n" + colBudgetID  +" Integer,\n" +

  "CONSTRAINT FK FOREIGN KEY (budgetID)

  REFERENCES budget_categories(colID)"+ " ); " ;

  // "CONSTRAINT FK FOREIGN KEY (budgetID)

   REFERENCES budget_categories(id)"
}
```

8.8 TRANSACTION MANAGER. JAVA:

```
package com.example.expensestracker.mainactivity;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.view.Menu;
import com.example.expensestracker.budget_categories.ui.BudgetCategoryInfoFragment;
import com.example.expensestracker.R;
import com.example.expensestracker.budget_categories.BudgetCategoryManager;
import com.example.expensestracker.budgetplanner.PlansManager;
import com.example.expensestracker.dbmanagment.BudgetCategoriesTable;
import com.example.expensestracker.dbmanagment.DatabaseController;
import com.example.expensestracker.globalOperations.DateStringFormatter;
import com.example.expensestracker.monthestracker.MonthExpenses;
import com.example.expensestracker.monthestracker.MonthsTracker;
import  com.example.expensestracker.transactions.TransactionManager;
import com.example.expensestracker.transactions.activities.AddingTransactionActivity;
import com.example.expensestracker.usermoney.UserWallet;
import com.example.expensestracker.usermoney.ui.AddingIncomeFragment;
import com.example.expensestracker.usermoney.ui.UserWalletFragment;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.navigation.NavigationView;
import androidx.annotation.NonNull;

import androidx.fragment.app.FragmentManager;

import androidx.navigation.NavController;

import androidx.navigation.Navigation;

import androidx.navigation.ui.AppBarConfiguration;

import androidx.navigation.ui.NavigationUI;

importandroidx.drawerlayout.widget.DrawerLayout;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import java.util.Calendar;

public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;

    DatabaseController databaseController;

    @Override

    protected void onCreate(Bundle savedInstanceState){

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        try {

            databaseController = new DatabaseController(this);

            BudgetCategoryManager.LoadCategoryEnvelop();

            MonthsTracker.checkUpdates(this);

            TransactionManager.loadTransactions();

            PlansManager.loadPlans();

            UserWallet.loadUserMoney(this);
            }

                Toolbar toolbar = findViewById(R.id.toolbar);

                setSupportActionBar(toolbar);

                //Snackbar.make(view, "Replace with your own action",
        Snackbar.LENGTH_LONG)

                //.setAction("Action", null).show();

                DrawerLayout drawer =
        findViewById(R.id.drawer_layout);

                NavigationView navigationView = findViewById(R.id.nav_view);

                // Passing each menu ID as a set of Ids because each

                // menu should be considered as top level destinations.

                mAppBarConfiguration = new AppBarConfiguration.Builder(

                                R.id.nav_home,R.id.nav_budgetManager                  ,
                                R.id.nav_budget_analysis ,R.id.nav_budgetPlanner,

                               R.id.nav_monthly_expenses_analysis).setDrawerLayout(drawer)

                     .build();
                NavController navController = Navigation.findNavController(this,
        R.id.nav_host_fragment);

                NavigationUI.setupActionBarWithNavController(this,navController,
        mAppBarConfiguration);

             NavigationUI.setupWithNavController(navigationView,navController);
             }

             catch (Exception e)

             {

                e.printStackTrace();

             }

          }
```

```
    @Override

    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it is present.

        getMenuInflater().inflate(R.menu.main, menu); return true;

    }
    @Override

    public boolean onOptionsItemSelected(@NonNull MenuItem item)

    {

        switch (item.getItemId())

        {

            case R.id.add_income:

                addIncome();

                return true;

            case R.id.user_wallet:

                showUserWallet();


            return true;

        }

        return super.onOptionsItemSelected(item);

    }


    @Override

    public boolean onSupportNavigateUp()

        {NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);

        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
```

```
                || super.onSupportNavigateUp();

    }
    private void addIncome()

    {

        FragmentManager fragmentManager=getSupportFragmentManager();


        AddingIncomeFragment addingIncomeFragment = new
AddingIncomeFragment();
        addingIncomeFragment.show(fragmentManager,"add    income");

    }

    private void showUserWallet()

    {

    FragmentManager fragmentManager=getSupportFragmentManager();

    UserWalletFragment userWalletFragment = new UserWalletFragment();
    userWalletFragment.show(fragmentManager,"add income");

    }

   // TODO // Database contain(plans , transactions , budget categories)

}
```

8.9 TRANSACTION. JAVA:

```
package com.example.expensestracker.transactions;


import com.example.expensestracker.budget_categories.BudgetCategory;
import java.util.Date;

public class Transaction

{

    private int budgetCategoryID; private String

    productName; private double price;

    private Date buyDate;

    public Transaction(int budgetCategoryID, String productName, double price,
    Date buyDate)
    {

        this.budgetCategoryID = budgetCategoryID;

        this.productName = productName;

        this.price = price;
        this.buyDate = buyDate;

    }
    public int getBudgetCategoryID()

    {

        return budgetCategoryID;

    }
    public void setBudgetCategoryID(int budgetCategoryID)

    {

        this.budgetCategoryID = budgetCategoryID;

    }
```

```
public String getProductName()

{

        return productName;

}
public void setProductName(String productName)

{

        this.productName = productName;

}
public double getPrice()

{

         return price;

}
public void setPrice(double price)

{

         this.price = price;

}
public Date getBuyDate()

{

         return buyDate;

}
public void setBuyDate(Date buyDate)

{

         this.buyDate = buyDate;

}
```

```
@Override

public String toString()

{

return "Transaction

{

"+"budgetCategory=" + budgetCategoryID + ", productName='" + productName +

 '\'' + ", price=" + price +", buyDate=" + buyDate +
'}';

}

@Override

public boolean equals(Object o)

{

      if (this == o)

            return true;

       if (!(o instanceof Transaction))

             return false;


    Transaction that = (Transaction) o;

    if (getBudgetCategoryID() != that.getBudgetCategoryID())
     return false;

    if (Double.compare(that.getPrice(), getPrice()) != 0)

    return false;

    if (getProductName() !=null ? !getProductName().equals(that.getProductName()) :
     that.getProductName() != null)
        return false;
```

```
            return getBuyDate() != null ? getBuyDate().equals(that.getBuyDate()) :
that.getBuyDate() == null;

      }
      @Override

      public int hashCode()

            {int result;

            long temp;

            result = getBudgetCategoryID();

            result = 31 * result + (getProductName() != null ?
getProductName().hashCode() : 0);

            temp = Double.doubleToLongBits(getPrice()); result = 31 * result +

            (int) (temp ^ (temp >>> 32));

            result = 31 * result + (getBuyDate() != null ? getBuyDate().hashCode() : 0);

            return result;

      }

}
```

8.10 STRING.XML:

```
<resources>

    <string name="app_name">Money Manager</string>


    <string name="navigation_drawer_open">Open navigation drawer</string>

    <string name="navigation_drawer_close">Close navigation drawer</string>

    <string name="nav_header_title">Expenses Tracker</string>

    <string name="nav_header_subtitle">Manage your budget, Track your Expenses
and Control your Plans</string>

    <string name="nav_header_desc">Navigation header</string>

    <string name="action_settings">Settings</string>

    <string name="menu_home">Expenses Tracker</string>

    <string name="menu_gallery">Gallery</string>

    <string name="menu_slideshow">Slideshow</string>

    <string name="total_expenses">Total Expenses</string>

    <string name="product_category">Product Category:</string>

    <string name="content_description_end_icon">write the category name</string>

    <string name="helper_text">"ex.: Food/Gas,...etc "</string>
    <string name="product_name">Product Name</string>

    <string name="product_price">Product Price</string>

    <string name="product_category_hint">Product Category:</string>

    <string name="transaction_date_hint">Transaction Date:</string>

    <string name="monthly_expenses_in">this Month Expenses in :</string>

    <string name="hello_blank_fragment">Hello blank fragment</string>

    <string name="budget_manager">Budget
Categories</string>
```

```
<string name="budget_planner">Budget Plans</string>
<string name="category">category Name</string>
<string name="budget">Budget</string>
<string name="add_budget_category">Add budget category</string>
<string name="add_income">Add Income</string>
<string name="category_price">Budget</string>
<string name="price_helper">write the maximum amount of money you can
spend in this category</string>
<string name="new_budget_category_info">Budget Category Info</string>
<string name="submit">Submit</string>
<string name="errorMsg">You Can not change the category name</string>
<string name="budget_analysis">Budget Analysis</string>
<string name="add_to_your_wallet">Add to your Wallet</string>
<string name="money_amount">Money Amount</string>
<string name="your_wallet">Your Wallet</string>
<string name="close">Close</string>
<string name="my_wallet">Your Wallet</string>
<string name="your_wallet_contains_enough_money_for_this_plan">Y      our
wallet contains enough money for this plan</string>
<string name="the_deadline_of_this_plan_has_passed">the deadline of this plan
has passed</string>
<string name="plan_title">Plan Title</string>
<string name="needed_budget">Plan\'s Budget</string>
<string name="plan_deadline_date">Plan Deadline Date :</string>
<string name="add_plan">Add Plan</string>
```

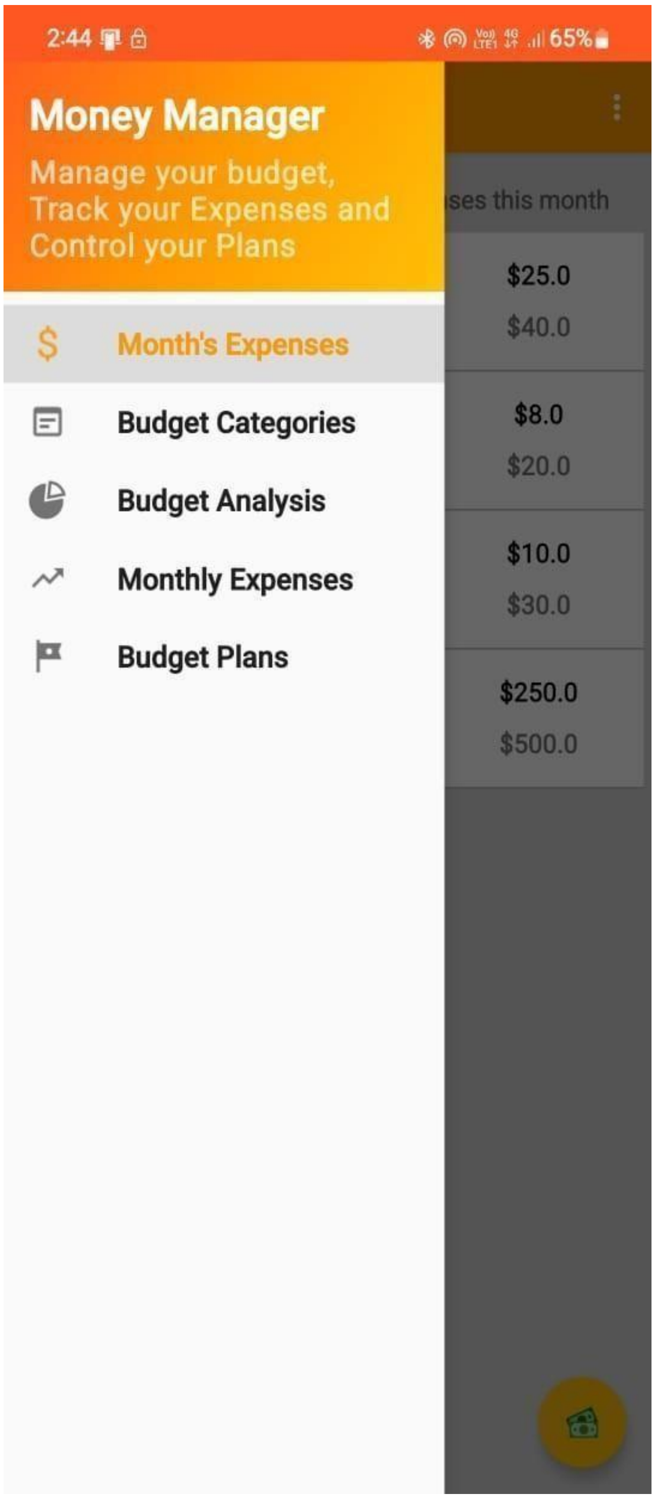
```
    <string name="add_transaction">Add Transaction</string>

    <string name="plan_desc">Plan Description</string>

    <string name="edit_plan">Edit Plan</string>

    <string name="your_expenses_this_month">Your Expenses this month</string>

    <string name="month_s_expenses">Month\'s Expenses</string>

    <string name="new_app_name">Money Manager</string>

    <string name="your_monthly_expenses_analysis">Monthly Expenses
Analysis</string>

    <string name="new_budget_category">New Budget Category</string>

    <string name="plan_info">Plan Info</string>

    <string name="your_expenses">Your Expenses</string>

    <string name="new_transaction">Add New Transaction</string></resources>
```

CHAPTER - 9

OUTPUT SCREENSHOTS

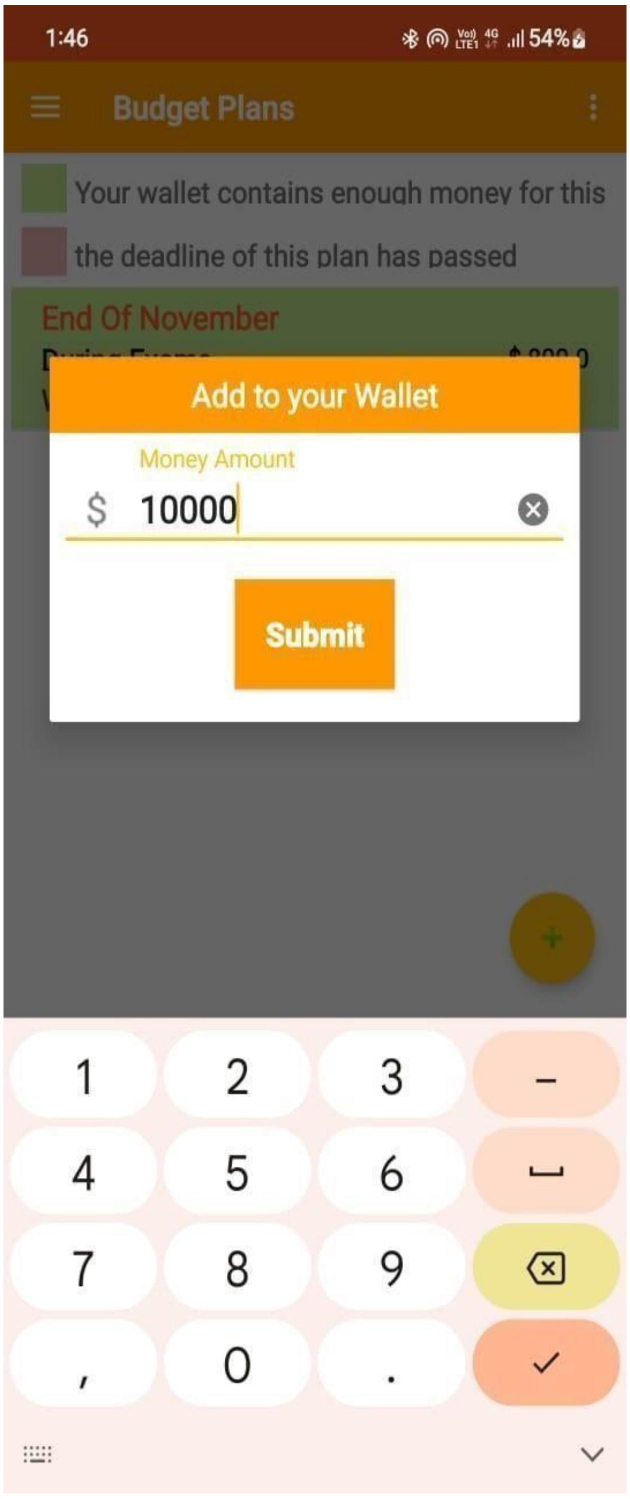
OUTPUT SCREENS:

9.1 MENU BAR:



LOGIN PAGE

9.2 ADD INCOME IN WALLET:



🟩 Your wallet contains enough money for this

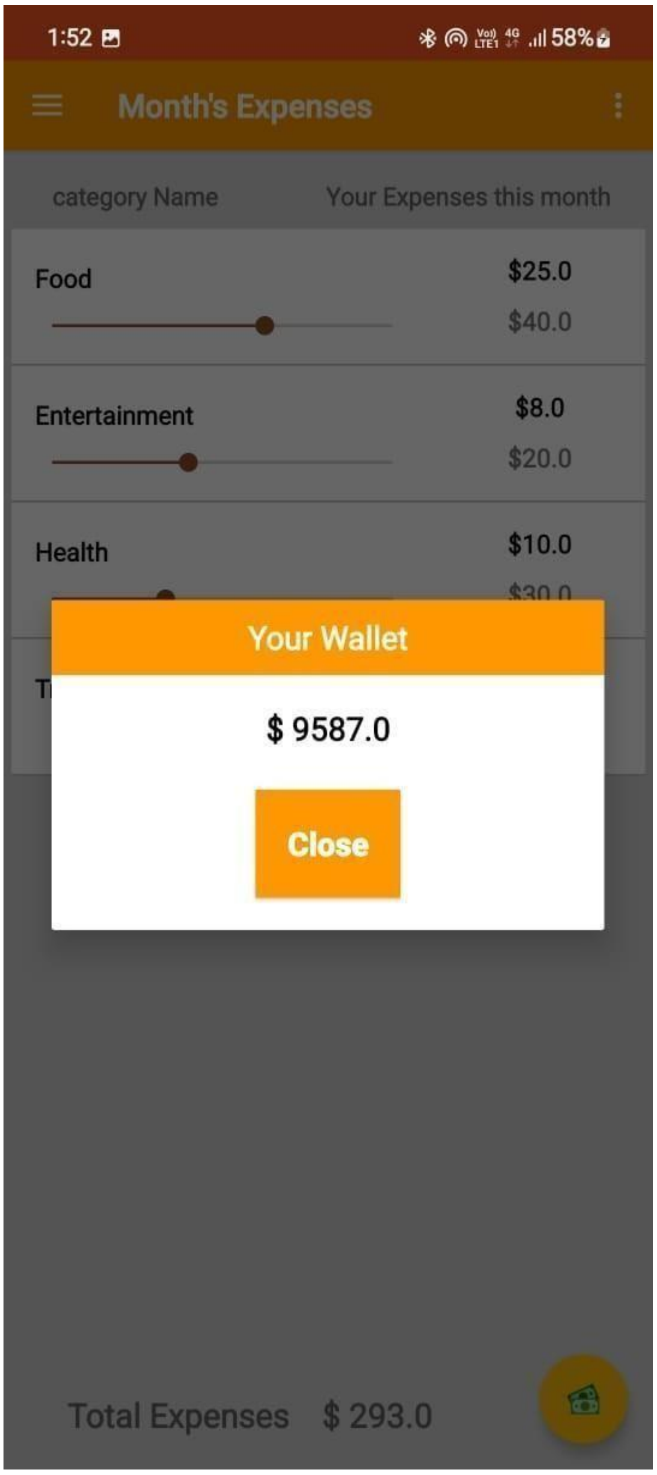
🟥 the deadline of this plan has passed

End Of November

Money Amount: $ 10000

ADD INCOME IN WALLET

9.3 WALLET:



1:52

58%

Month's Expenses

category Name	Your Expenses this month
Food	$25.0 / $40.0
Entertainment	$8.0 / $20.0
Health	$10.0 / $30.0

Your Wallet

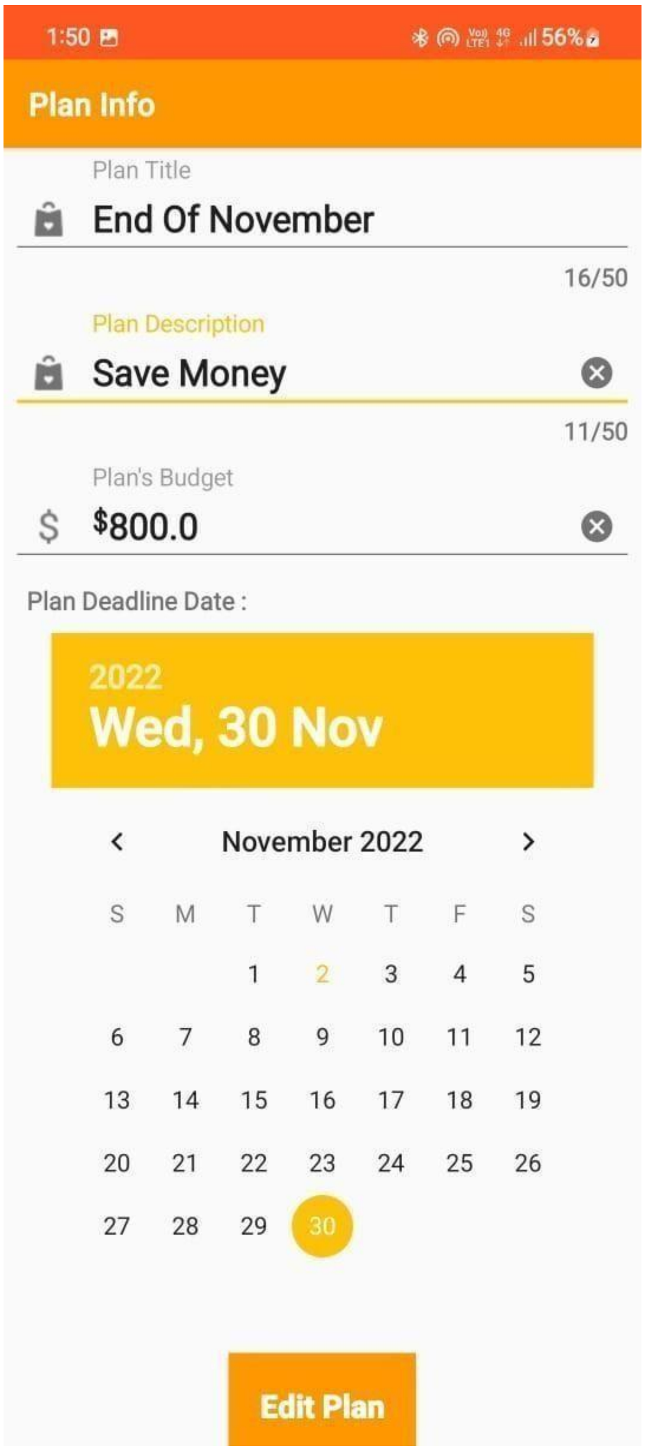
$ 9587.0

Close

Total Expenses $ 293.0

WALLET

9.4 ADD PLAN:


1:50 56%

Plan Info

Plan Title
End Of November
16/50

Plan Description
Save Money
11/50

Plan's Budget
$800.0

Plan Deadline Date :

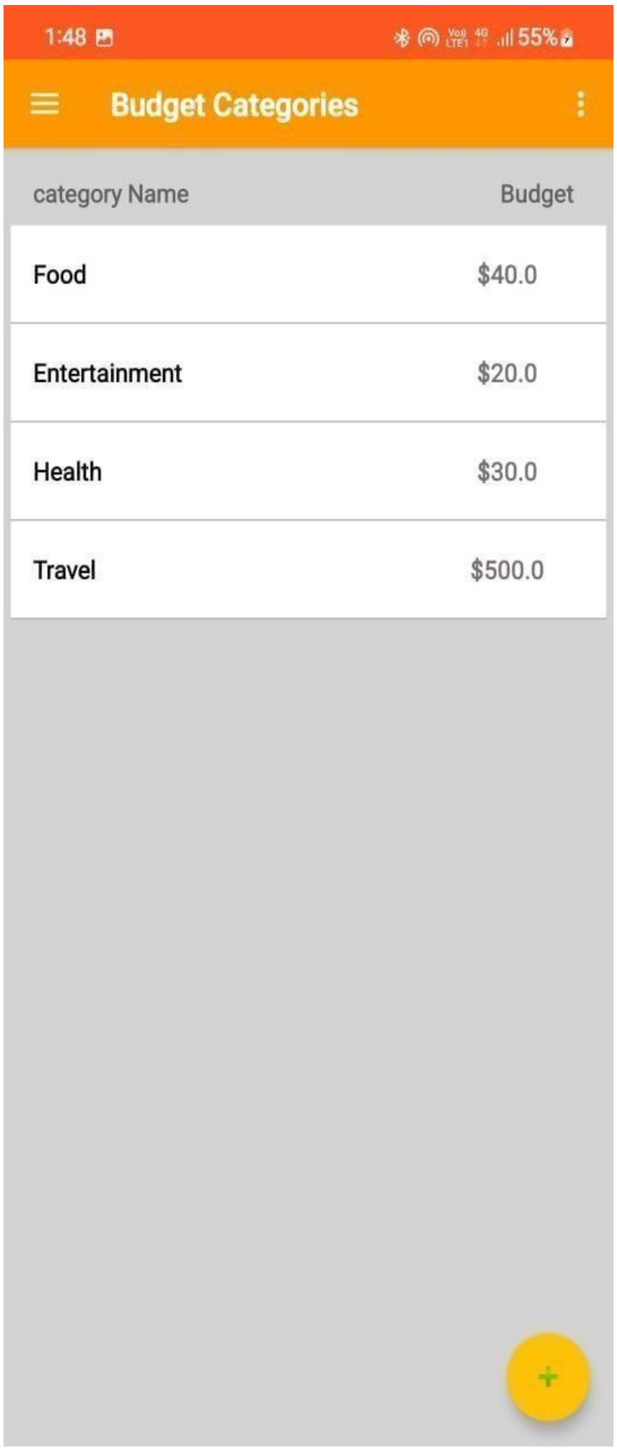
2022
Wed, 30 Nov

November 2022

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

ADD PLAN

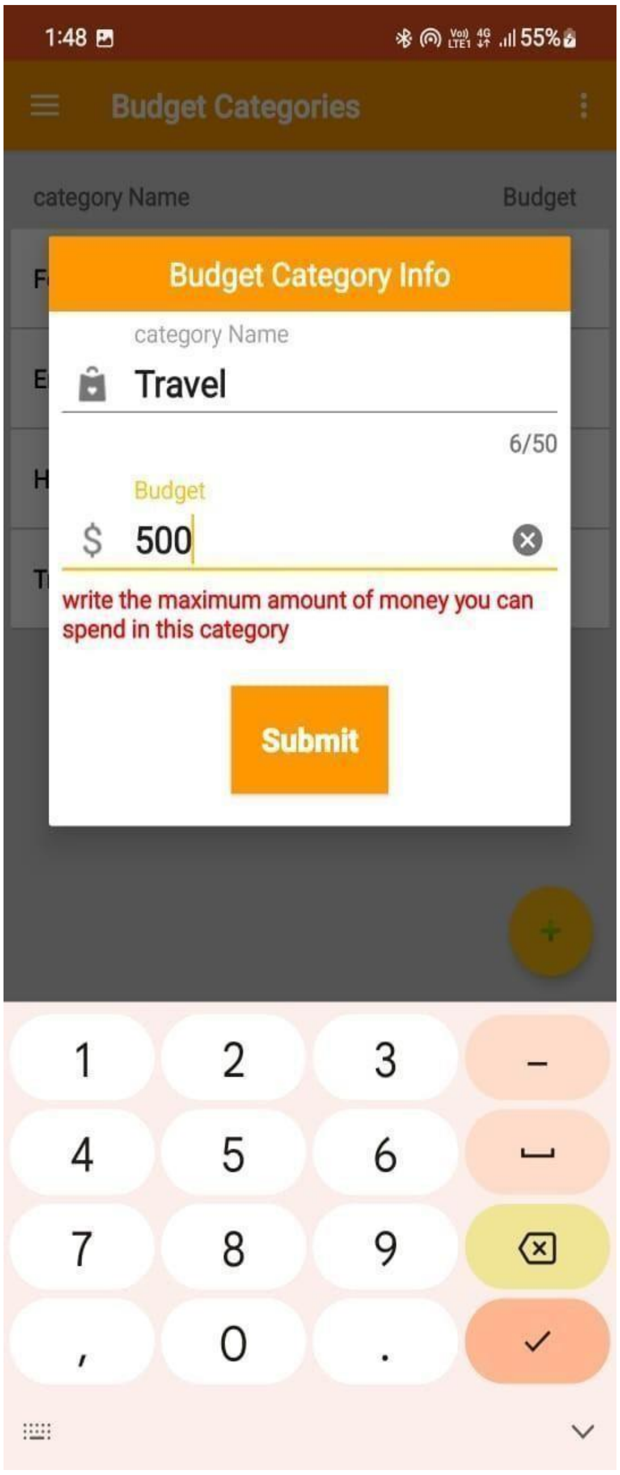
9.5 BUDGET CATEGORIES:



category Name	Budget
Food	$40.0
Entertainment	$20.0
Health	$30.0
Travel	$500.0

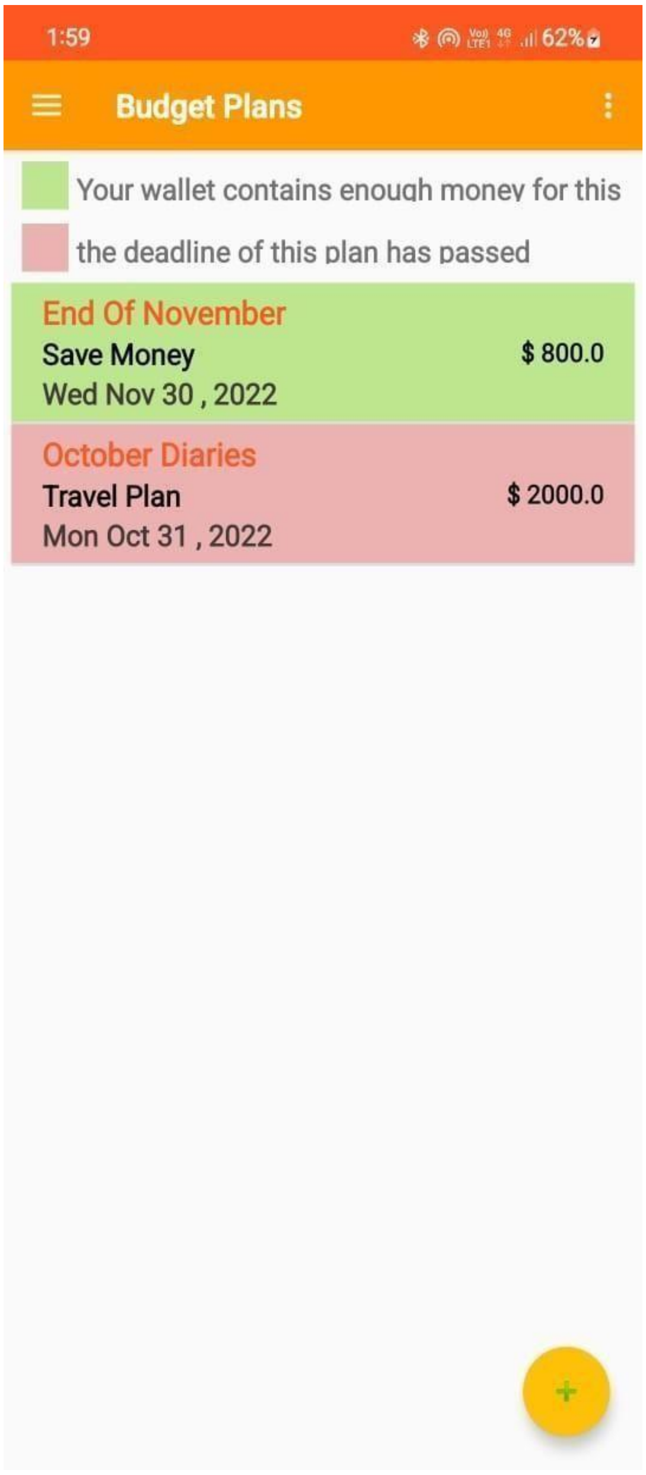
BUDGET CATEGORIES

9.6 ADD/EDIT CATEGORIES:



ADD/EDIT CATEGORIES

9.7 BUDGET PLANS:



1:59 62%

Budget Plans

🟩 Your wallet contains enough money for this

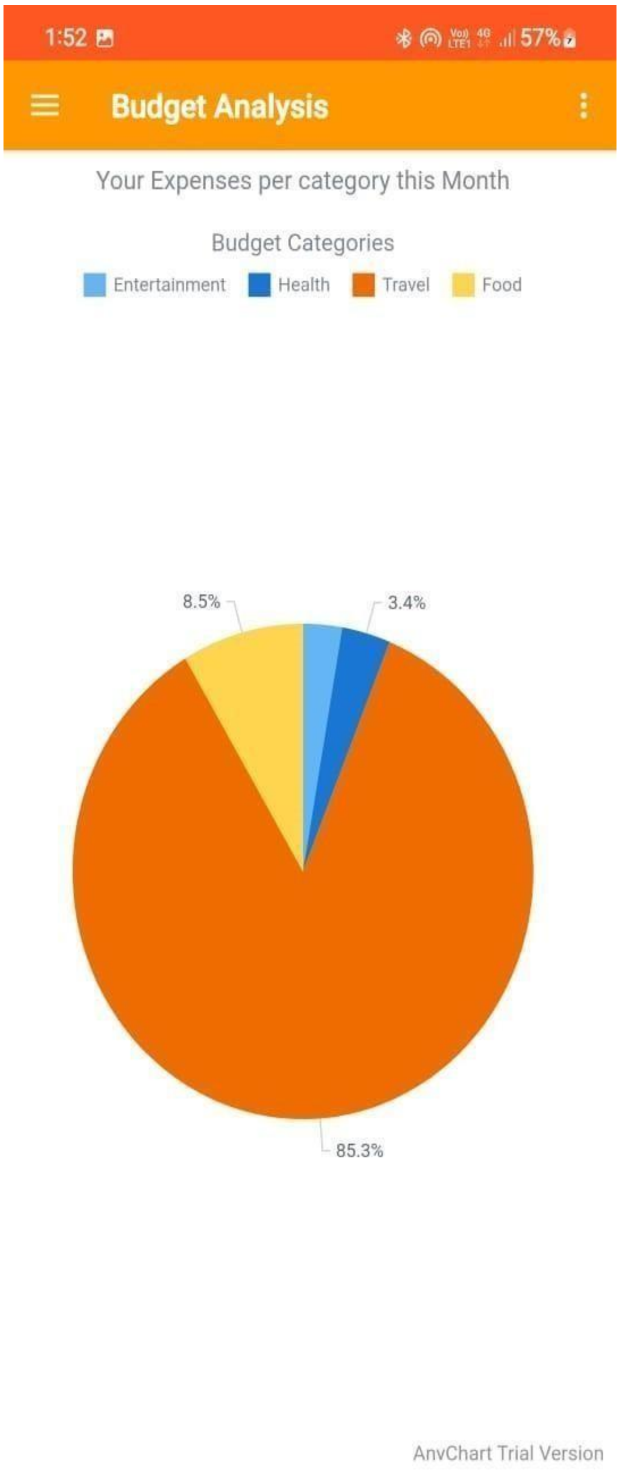
🟥 the deadline of this plan has passed

End Of November
Save Money $ 800.0
Wed Nov 30 , 2022

October Diaries
Travel Plan $ 2000.0
Mon Oct 31 , 2022

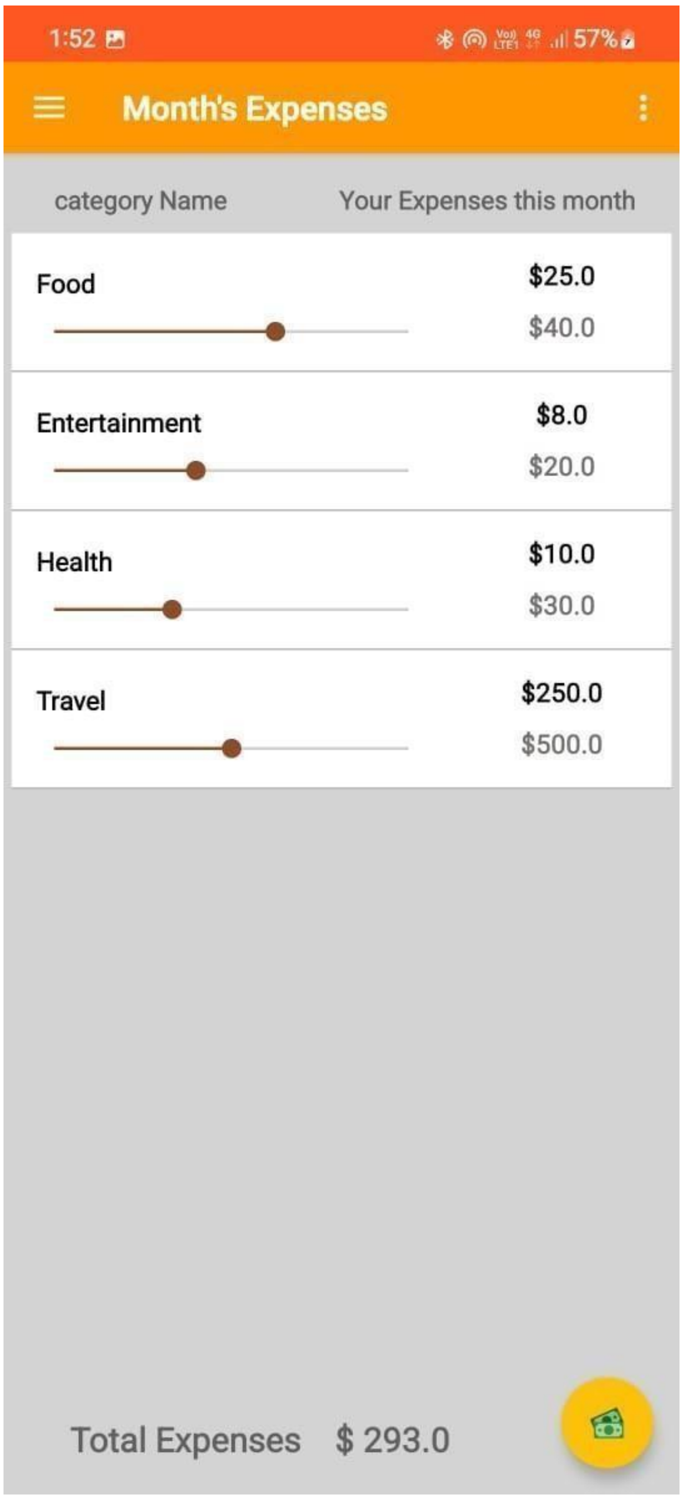
BUDGET PLANS

9.8 BUDGET ANALYSIS:



BUDGET ANALYSIS

9.9 CATEGORY WISE MONTHLY EXPENSE:


1:52

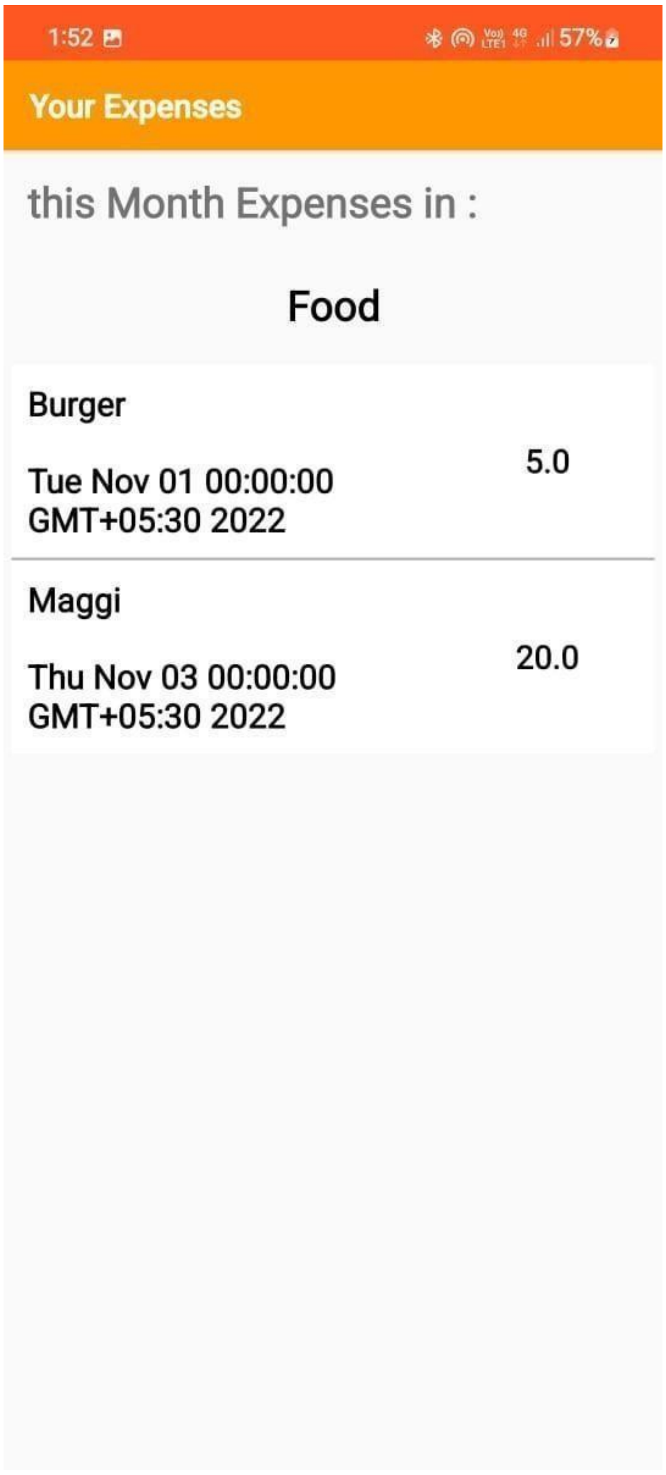
Month's Expenses

category Name	Your Expenses this month
Food	$25.0 / $40.0
Entertainment	$8.0 / $20.0
Health	$10.0 / $30.0
Travel	$250.0 / $500.0

Total Expenses $ 293.0

CATEGORY WISE MONTHLY EXPENSES

9.10 MONTHLY EXPENSE:



1:52

57%

Your Expenses

this Month Expenses in :

Food

Burger

Tue Nov 01 00:00:00 GMT+05:30 2022

5.0

Maggi

Thu Nov 03 00:00:00 GMT+05:30 2022

20.0

MONTHLY EXPENSES

CHAPTER - 10

CONCLUSION

CONCLUSION:

Developing a budget manager and tracking all expenses and spending is a crucial aspect of personal finances. It allows individuals to gain a clear understanding of their financial habits, identify areas where they can save money, and plan for future financial goals. Setting aside a fixed amount in a savings account is often recommended as a safety net for emergencies. Financial experts suggest having at least three months' worth of living expenses saved in an emergency fund to cover unexpected costs such as medical emergencies or job loss.

Moreover, educating children early in life about personal finances is essential. It should be a mandatory class in every school curriculum. Parents also need to be proactive in teaching their children about banking, credit cards, interest rates, and credit scores. By instilling these financial literacy skills at a young age, children can develop healthy money management habits that will benefit them throughout their lives.

The importance of actually seeing one's spending on a budget sheet can be enlightening. It provides a tangible representation of where money is being spent and can help individuals make more informed financial decisions. Learning how to manage money effectively can pave the way for a brighter future. It can lead to financial stability, the ability to achieve long-term financial goals, and a sense of empowerment and control over one's finances.

CHAPTER - 11

REFERENCES

REFERENCE:

- https://www.investopedia.com/terms/m/moneymanager.asp
- https://apps.apple.com/us/app/money-manager-expense-budget
- https://www.youtube.com/watch?v=dZ5aCGr26Q8
- https://www.gadgets360.com/apps/ money-management-finances-budget
- https://www.google.com/search?q=money+management+app
- https://play.google.com/store/apps/details.application.wallet