A

Project Report

On

“Expense Manager”

Submitted in Partial Fulfillment for Degree of

DIPLOMA IN COMPUTER ENGINEERING

Submitted By

Mr. Preet K. Mungara	**(176470307032)**
Mr. Ankit R. Zadafiya	**(176470307063)**

Guided By

Mr. Jaydip D. Ujainiya

2019 - 2020

Department of Computer Engineering

TAPI DIPLOMA ENGINEERING COLLEGE,

SURAT

C E R T I F I C A T E

This is to certify that **Mr. Preet Mungara** and **Mr. Ankit Zadafiya** having Enrollment No. **176470307032** and **176470307063** has completed part-I UDP Project work having title **“Expense Manager”** along with **2** group members.

They have undergone the process of shodh yatra, literature survey and problem definition. They are supposed to carry out the residue UDP Part-II work on same problem during Semester-VI for the final fulfilment of the UDP work which is prerequisite to complete Diploma Engineering.

Institute Guide	Head of Department

Acknowledgement

The success of any task relies on the efforts made by person but it cannot be achieved without cooperation of other persons which are being helpful. So, we would like to thank **TAPI Diploma Engineering College, Surat** for giving us the opportunity of doing this project.

The entire session of our phase I completion was a great experience providing us with the insight & invocation into learning various software engineering concepts & benefits of team work. We would like to take this opportunity to express our sincere thanks to all those people without whose support and co-operation, it would have been difficult to complete this project.

Primarily, we are very much thankful to our internal project guide **Mr. Jaydip D. Ujainiya** for their leading guidance and sincere efforts throughout project work. They took deep interest in simplifying the difficulties. Also, they have been consistent source of inspiration for us.

We also express thanks to **Mr. Jaydip D. Ujainiya**, our registration faculty for his personal involvement, constructive suggestion and thoughtful idea for betterment of the project.

We are grateful to our **H.O.D. Mr. N. M. Sanghani** and our beloved **Principal Prof. Y. S. Choupare** for providing us deep knowledge and all necessary resources.

We are also thankful to our **Friends** and all **staff members** of Computer Engineering Department for their valuable time and help for completion project.

Once again, we are grateful to all those without whom this work would not have been successful.

Team Members

Mr. Preet Mungara

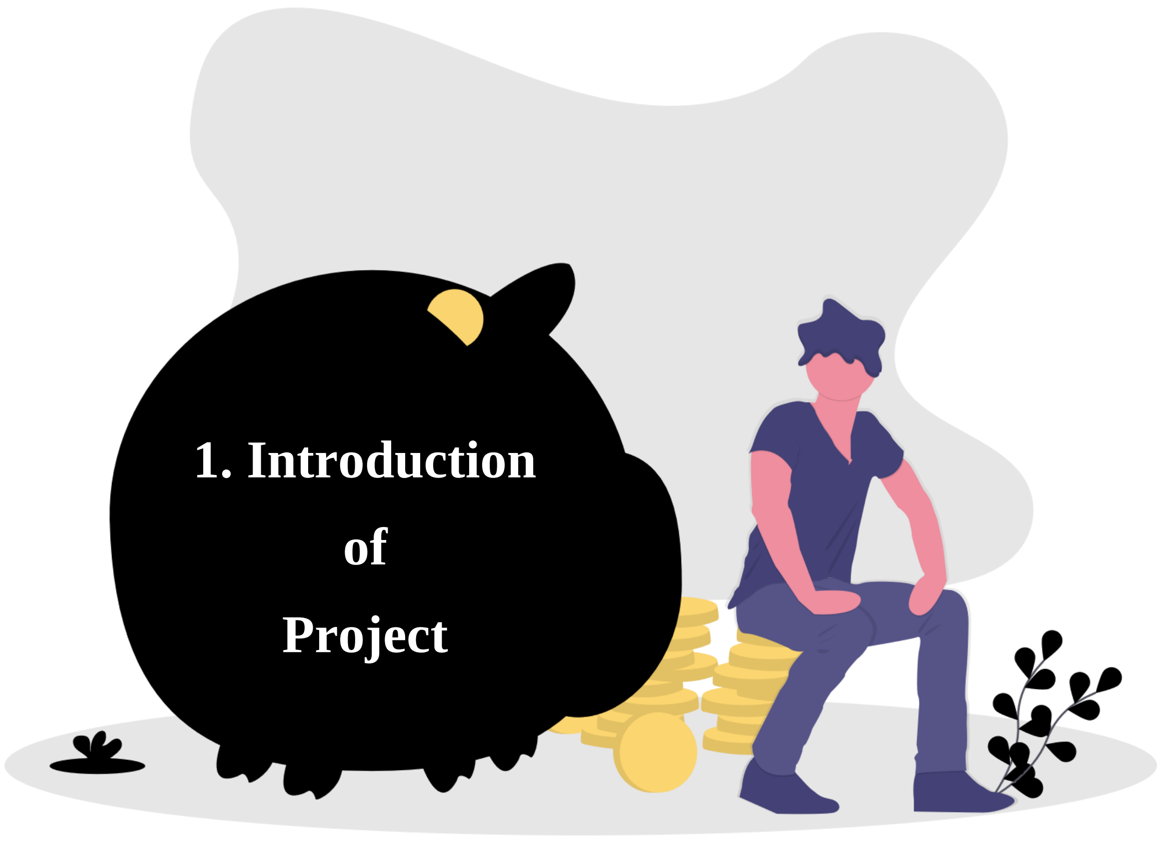
Mr. Ankit Zadafiya

Abstract

Expense Manager is all about managing your money and keep track of your expenses at your own fingertips. It also provides user-friendly categories so user can manage money using categories. It helps user to do depth analysis by chart. It makes personal finances management easy for everyone. It also provides automatic backup of data and exporting data to files.

The purpose of making this application is to provide easy and efficient way to manage your personal finances without internet at anywhere.

Index



1. Introduction of Project

1.1 Introduction of Problem

1.1.1 Problem Statement

- At the end of the month we start to have money crisis.
- Lack of our proper planning of our income.
- Person has to keep log in to diary.
- All the calculations need to be done by the person manually.
- Overload to rely on the daily entry of the expenditure.

1.1.2 Scope of Proposed System

- Our application can be useful as it is usable by anyone who are willing to manage their expenses and aiming to save for the future investments and many more.

1.2 Environment Description

1.2.1 Hardware and Software Requirement

Hardware Specification

At Mobile Side

Processor Speed	600 MHz
Memory	512 MB or greater
Storage	250 MB

At Development Side

Processor	Intel® Pentium IV /AMD Athlon or Higher
Memory Speed	2.20 GHz
Memory	4 GB Minimum
Disk Space	2 GB Minimum

Software Specification

At Mobile Side

Device OS	Android 6.0 (Marshmallow) or Higher

At Development Side

OS	Windows 7 32bit or 64 or Higher
Front End	Flutter v1.9.1, Dart v2.5.0
Back End	Firebase, SQLite

1.2.2 Technology Used

Flutter:

- Flutter is an open-source mobile application development framework created by Google.
- Flutter is Google's UI toolkit for building beautiful, natively compiled applications for mobile, web, and desktop from a single codebase.
- It is used to develop applications for Android and iOS, as well as being the primary method of creating applications for Google Fuchsia.
- Flutter Paint your app to life in milliseconds with Stateful Hot Reload. It uses a rich set of fully-customizable widgets to build native interfaces in minutes.

Flutter Architecture:

The major components of Flutter include:

- Dart Platform
- Flutter Engine
- Foundation Library
- Design-Specific Widgets

Dart Platform:

- Flutter apps are written in the Dart language and make use of many of the language's more advanced features.

- On Android, and on Windows, macOS and Linux via the semi-official Flutter Desktop Embedding project, Flutter runs in the Dart virtual machine which features a just-in-time execution engine. Due to App Store restrictions on dynamic code execution, Flutter apps use ahead-of-time (AOT) compilation on iOS.

- A notable feature of the Dart platform is its support for "hot reload" where modifications to source files can be injected into a running application. Flutter extends this with support for stateful hot reload, where in most cases changes to source code can be reflected immediately in the running app without requiring a restart or any loss of state. This feature as implemented in Flutter has received widespread praise.

Flutter Engine:

- Flutter's engine, written primarily in C++, provides low-level rendering support using Google's Skia graphics library.

- Additionally, it interfaces with platform-specific SDKs such as those provided by Android and iOS.

- It implements Flutter's core libraries, including animation and graphics, file and network I/O, accessibility support, plugin architecture, and a Dart runtime and compile tool chain.

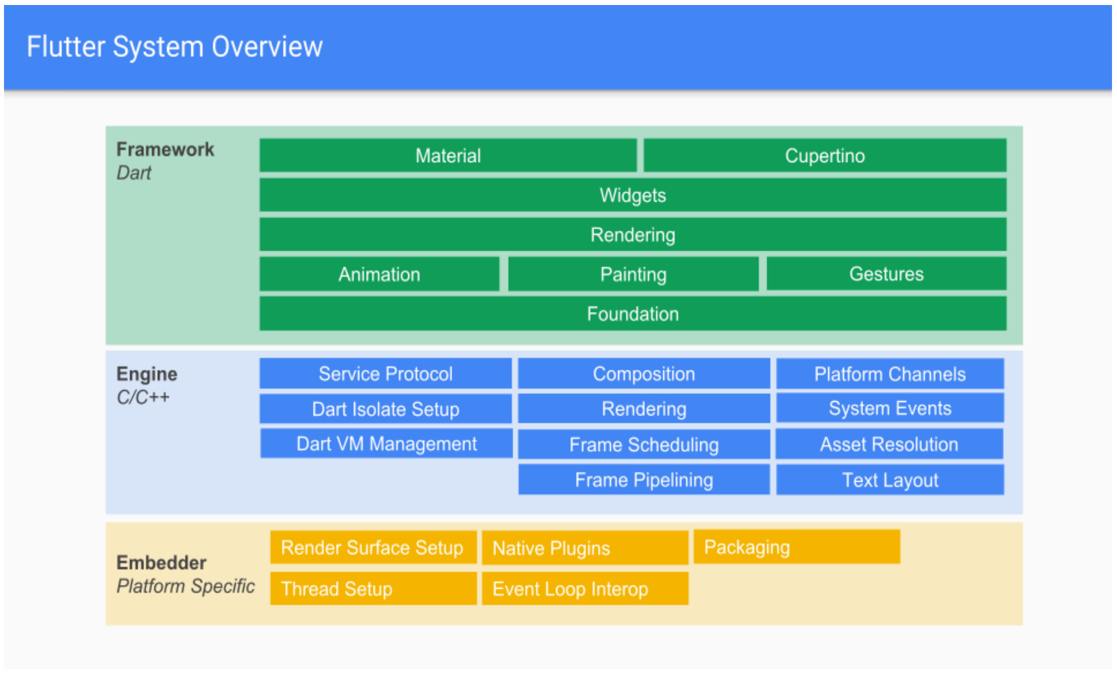
Foundation Library:

- The Foundation library, written in Dart, provides basic classes and functions which are used to construct applications using Flutter, such as APIs to communicate with the engine.

Design-Specific Widgets:

- UI design in Flutter typically involves assembling and/or creating various widgets. A widget in Flutter represents an immutable description of part of the user interface; all graphics, including text, shapes, and animations are created using widgets. More complex widgets can be created by combining many simpler ones.

- The Flutter framework contains two sets of widgets which conform to specific design languages. Material Design widgets implement Google's design language of the same name, and Cupertino widgets imitate Apple's iOS design.



Dart:

- Dart is a client-optimized programming language for fast apps on multiple platforms. It is developed by Google and is used to build mobile, desktop, backend and web applications.

- Dart is an object-oriented, class defined, garbage-collected language using C-style syntax that transcompiles optionally into JavaScript. It supports interfaces, mixins, abstract classes, reified generics, static typing, and a sound type system.

- To run in mainstream web browsers, Dart relies on a source-to-source compiler to JavaScript. According to the project site, Dart was "designed to be easy to write development tools for, well-suited to modern app development, and capable of high-performance implementations. When running Dart code in a web browser the code is precompiled into JavaScript using the dart2js compiler. Compiled as JavaScript, Dart code is compatible with all major browsers with no need for browsers to adopt Dart.

- Through optimizing the compiled JavaScript output to avoid expensive checks and operations, code written in Dart can, in some cases, run faster than equivalent code hand-written using JavaScript idioms.

- The Dart software development kit (SDK) ships with a stand-alone Dart VM, allowing Dart code to run in a command-line interface environment. As the language tools included in the Dart SDK are written mostly in Dart, the stand-alone Dart VM is a critical part of the SDK. These tools include the dart2js compiler and a package manager called pub.

- Dart code can be AOT-compiled into machine code (native instruction sets). Apps built with Flutter, a mobile app SDK built with Dart, are deployed to app stores as AOT-compiled Dart code.

SQLite:

- SQLite is a relational database management system contained in a C programming library.

- In contrast to other database management system, SQLite is not a client server database engine. Rather, it is embedded into the end program.

- SQLite is ACID-compliant and implement most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.

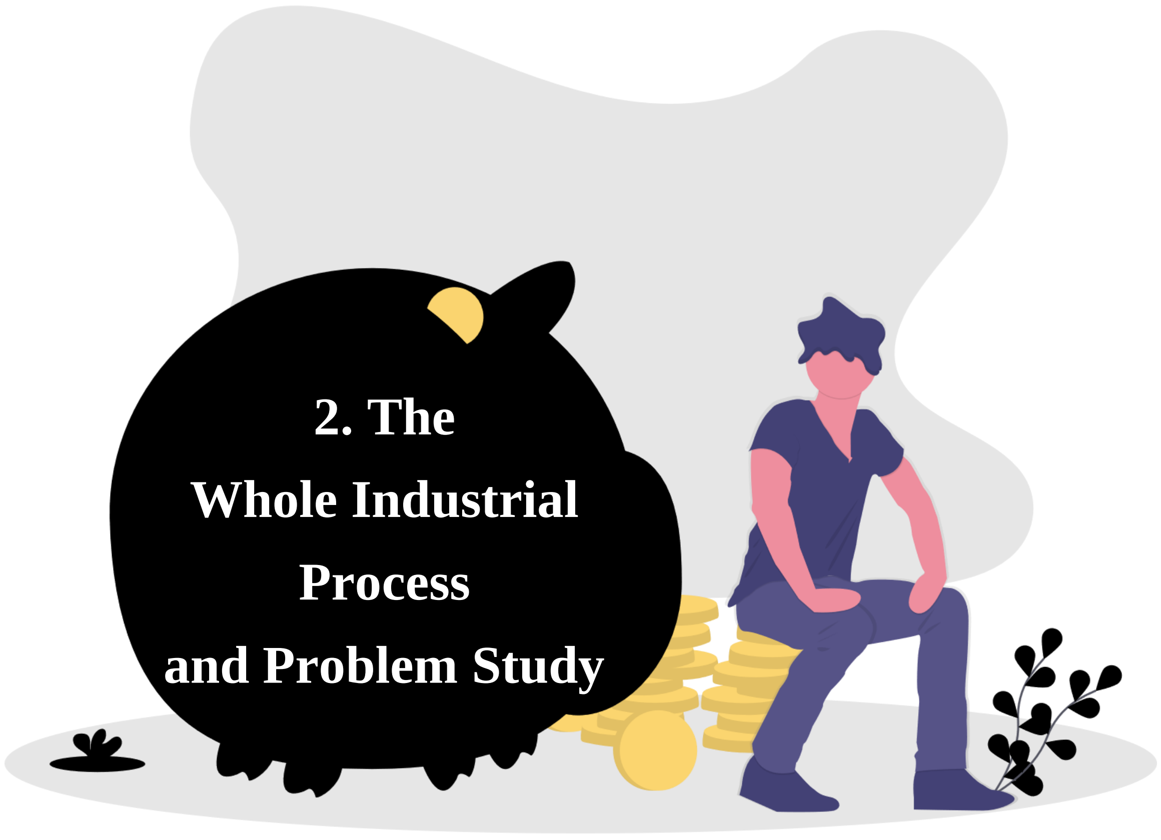
- SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded, among others. SQLite has bindings to many programming languages.

Android Studio:

- Android Studio is the official integrated development environment (IDE) for Google's Android operating system.
- It is built on JetBrains IntelliJ IDEA software and designed specifically for Android development.
- It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse.
- Some Features of Android Studio:
    - Gradle-based build support.
    - Android-specific refactoring and quick fixes.
    - ProGuard integration and app-signing capabilities.
    - A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations.
    - Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Firebase:

- Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014. As of October 2018, the Firebase platform has 18 products, which are used by 1.5 million apps.
- Firebase provides a Realtime database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud.
- Firebase Provide Following Services:
  - Firebase Analytics
  - Firebase Cloud Messaging
  - Firebase Auth
  - Firebase Realtime Database
  - Firebase Cloud Firestore
  - Firebase Storage
  - Firebase Hosting
  - ML Kit
  - Crashlytics



2. The Whole Industrial Process and Problem Study

2.1 System Planning

2.1.1 Requirement Analysis & Data Gathering

- Requirement gathering is the base part of system planning. Most of the basic requirement need to be studied it also includes user's system and hardware requirement.
- Requirement gathering means collecting and understanding by all entire related requirements by all possible sources, by discussing with the user is the possible and easiest source of requirement gathering.
- To understand the natures of software to be build. The software engineer must understand the information domain for the software as the required function, behavior, platform and interface.
- This system basically targets to public so it is good for us to consider people requirement.

Functional Requirement

- Graphical User interface through which the user interacts.
- SQLite and Firebase that stores the information to be displayed to the user.

Non-Functional Requirement

1) Usability

- There would be consistency in all the modules and pages. To ease the navigation there is a back tab to provide access to previous page.
- There would be proper instruction on each page.

2) Supportability

- App is built to support any machine. Maintainability is easy.

3) Performance

- In order to ease the accessibility, types of expenses are categorized along with a name to own. Throughput of system is increased due to lightweight database.

4) Availability

- It is available all the time, no time constraint.

2.1.2 Expected Modules

- **Login module:** this module is responsible for a registered user to login to the application and do the processing.
- **Signup module:** this module is responsible for registering a new user to application and create a new account for him/her.
- **Dashboard module:** this module is responsible for allow user to add, modify, delete expenses/income and show chart and statistic details about it.
- **Categories module:** this module is responsible for allow user to add custom categories and modify or delete in-built categories.
- **Export module:** this module is responsible for allow user to export data to .excel or .csv file.
- **Backup module:** this module is responsible for allow user to backup data.

- **Security module:** this module is responsible for provide local Fingerprint authentication (Client side) to authenticate user.

2.1.3 Feasibility Study

- The entire project is feasible that utilize unlimited resources and infinite time are available.
- The purpose of the feasibility study is to produce a feasibility study document that evaluates the costs and benefits of the proposed system.
- The first thing is necessary to analyze the problem, which is in old system. Based on the definition of the problem during the preliminary analysis we analyze cost and delivery date.
- Thus, feasibility study should follow following area of feasibility:
  - Technical feasibility
  - Schedule feasibility
  - Economically feasibility
  - Operational feasibility

Technical Feasibility

- This application requires some level of technology. It requires database. Interaction this can be easily done.
- System should be expandable configurable and also system would guarantee accuracy and data security.

- The technical feasibility work for this project is done with the present equipment manual procedures, existing software technology and available technology hardware.

- Thus, at the end we found that our project is technically feasible.

Schedule feasibility

- We had given sufficient time to develop a system. So, this is quiet enough to us for developing the system required by organization and according to the system requirements.

- So, our project is also feasible considering time duration we have.

Economically feasibility

- All the developing software are open source so it easily available on internet so it not needs to pay for the license version of anything.

- So, our project is also economically feasible.

Operational feasibility

- This user will not worry about the errors in report. His/his fears about the complex system would be avoided. This will not be involved into the complex report preparation procedure. The system will be configured and as generic as possible.

- It will work any full efficiency and accuracy as used to work any particular computer.

- So, our project is also Operational feasible that means it is able to perform all operation.

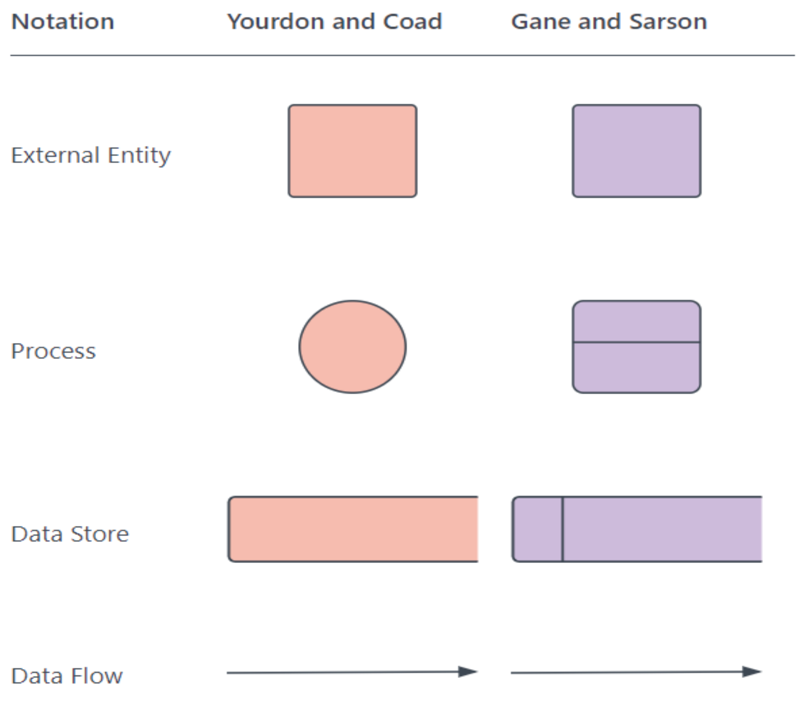
2.1.4 Limitation of project

- User have to entry every record manually. Though we provide an option for recurrence expense which occurs frequently at specific time duration.
- Category divided may be messy or confusing. Though we provide an option to add custom category.
- Person who is using should have requires somewhat technical knowledge.
- Internet is required for signup and login for permanent user. Though user can login as guest account.

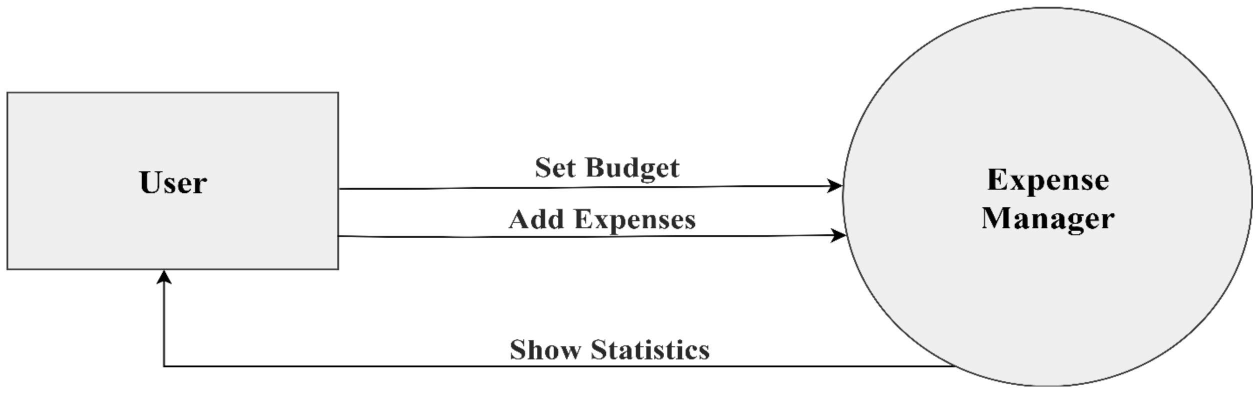
2.2 Data Flow Diagram

- A graphical representation that depicts information flow and the transaction that are applied as data moves from input to output is known as **"data flow diagram"**. The Data Flow Diagram is also known as **"bubble chart"**. The DFD may be represents increasing information levels that represent increasing information Flow and functional details. The DFD Represent the entire s/w element as a **single bubble**. As a level 0 DFD is partitioned additional process are brought out of level more details.
- **Data Flow Diagram Serves two purposes…** It provides a graphical tool. This can be used by the analyst to explain his understanding if the system to the user. It can readily convert into a structured chart which is used only four elements.

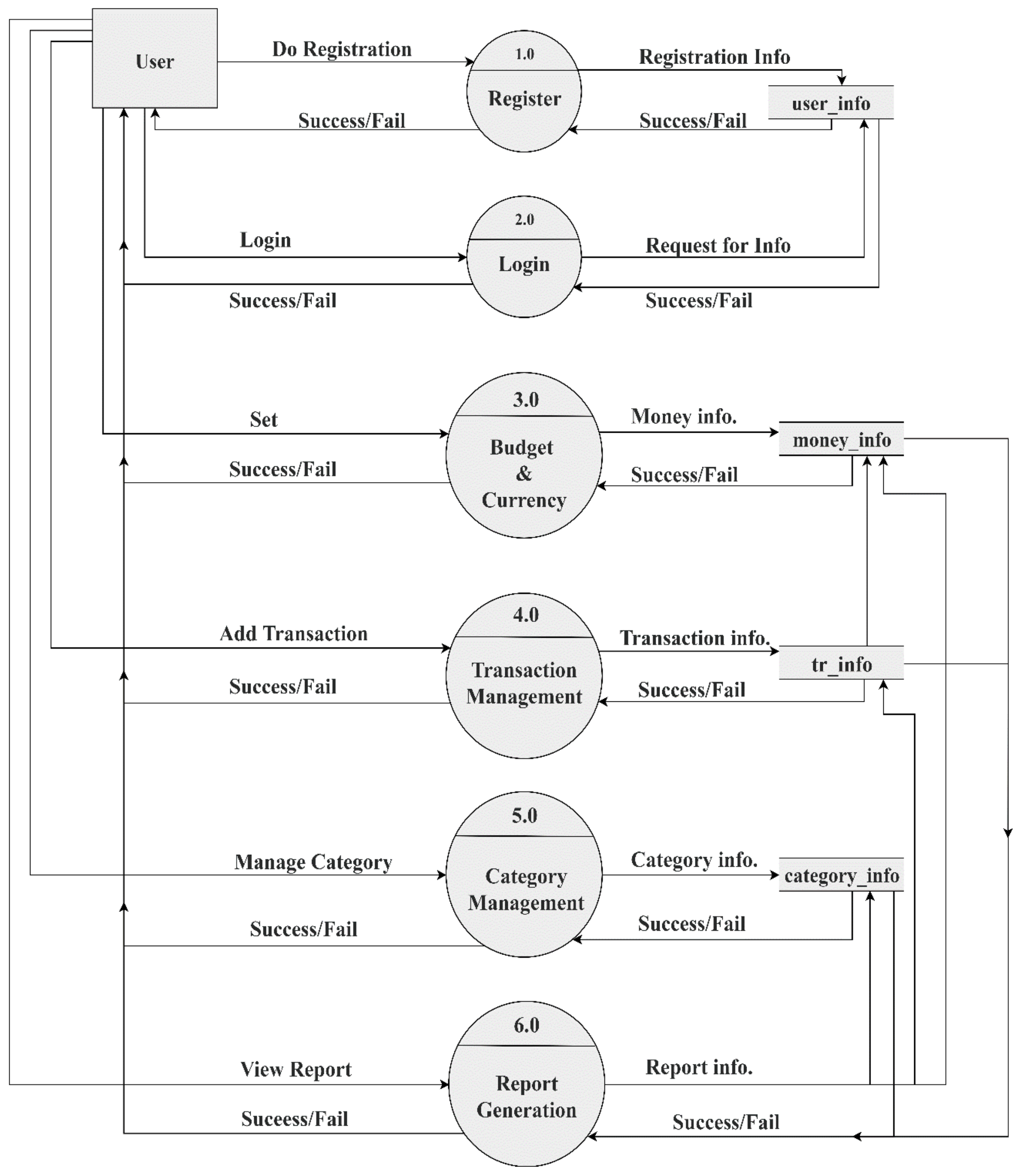
Symbols used in DFD:



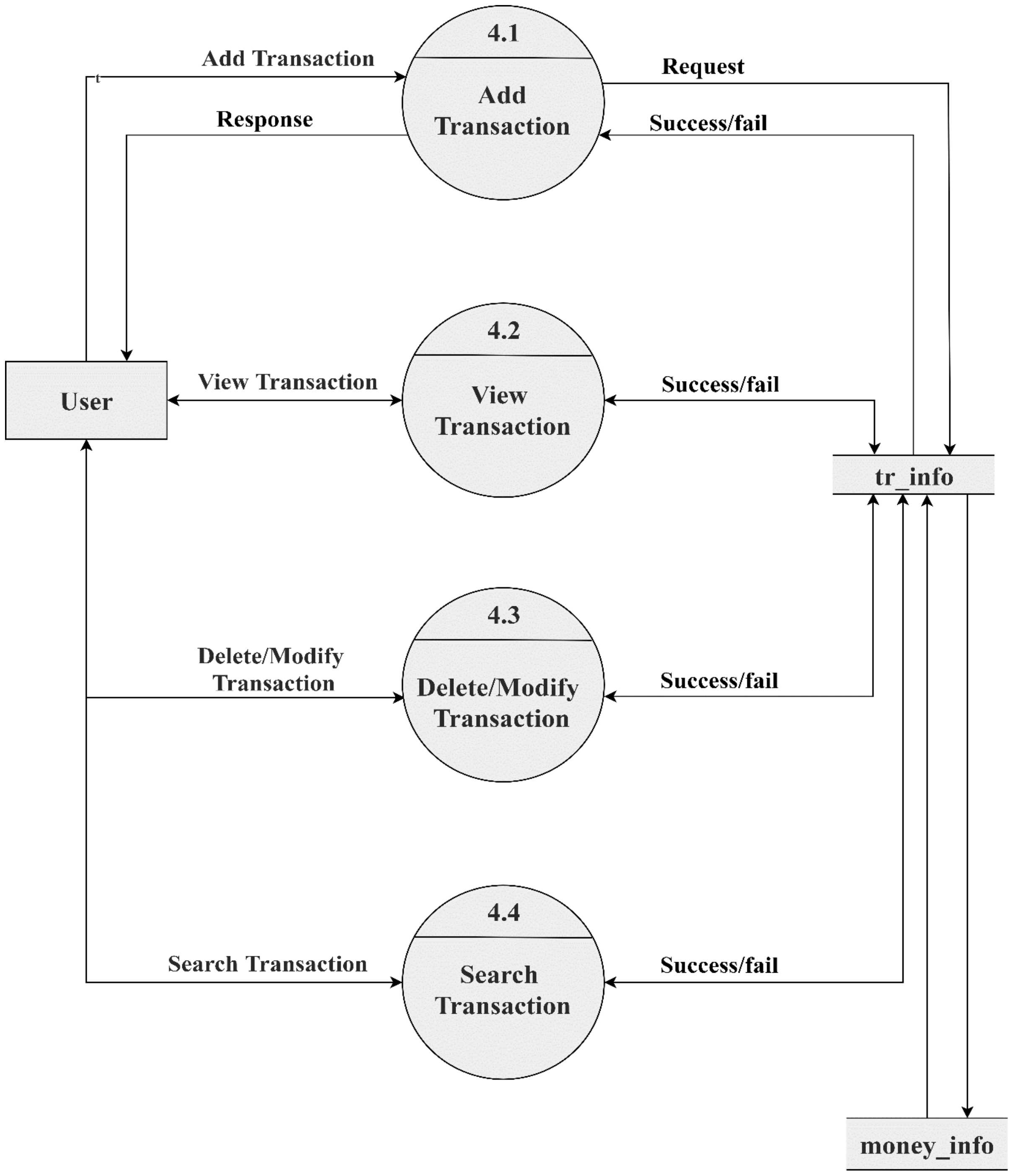
Level 0 DFD for Expense Manager



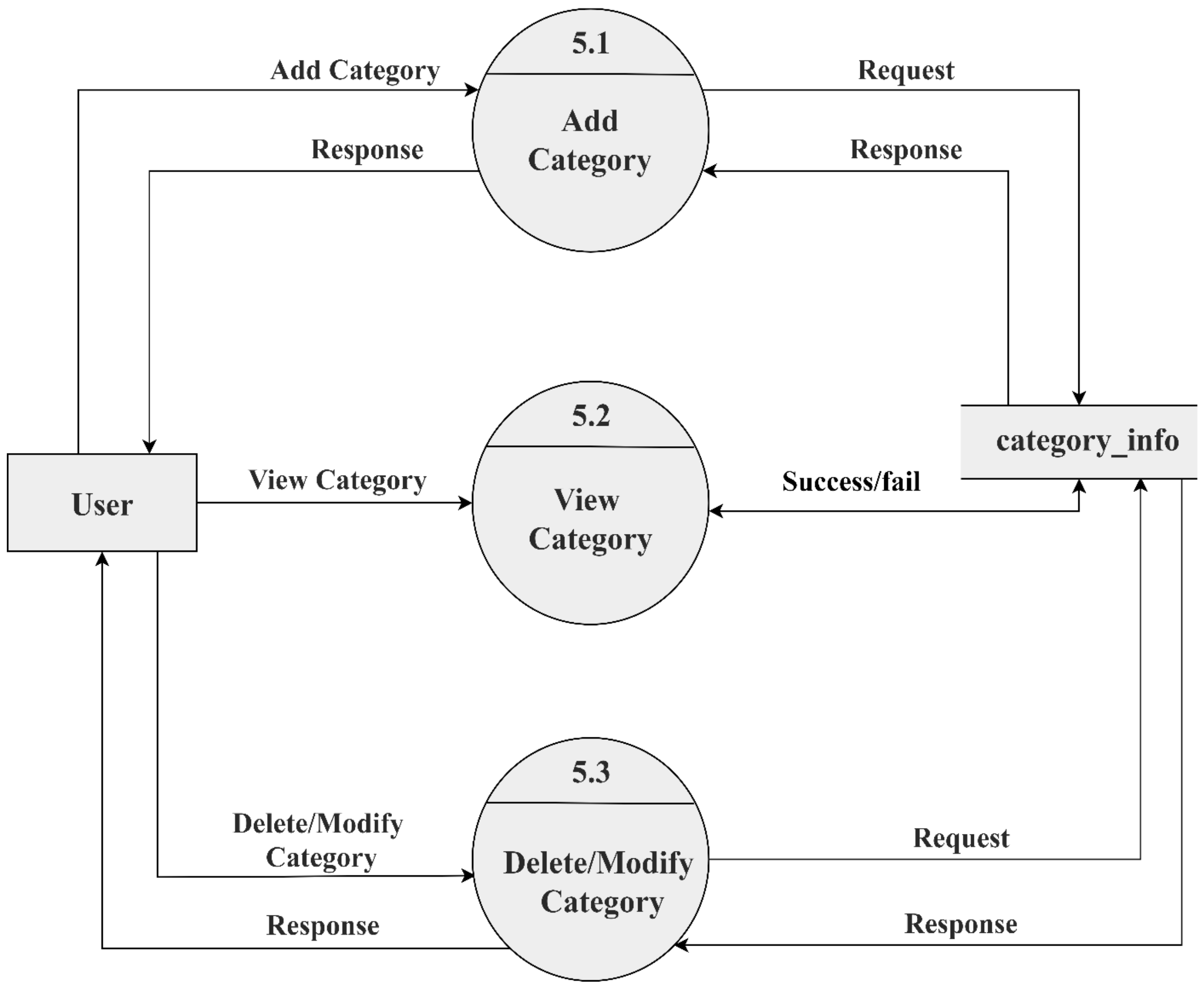
Level 1 DFD for Expense Manager



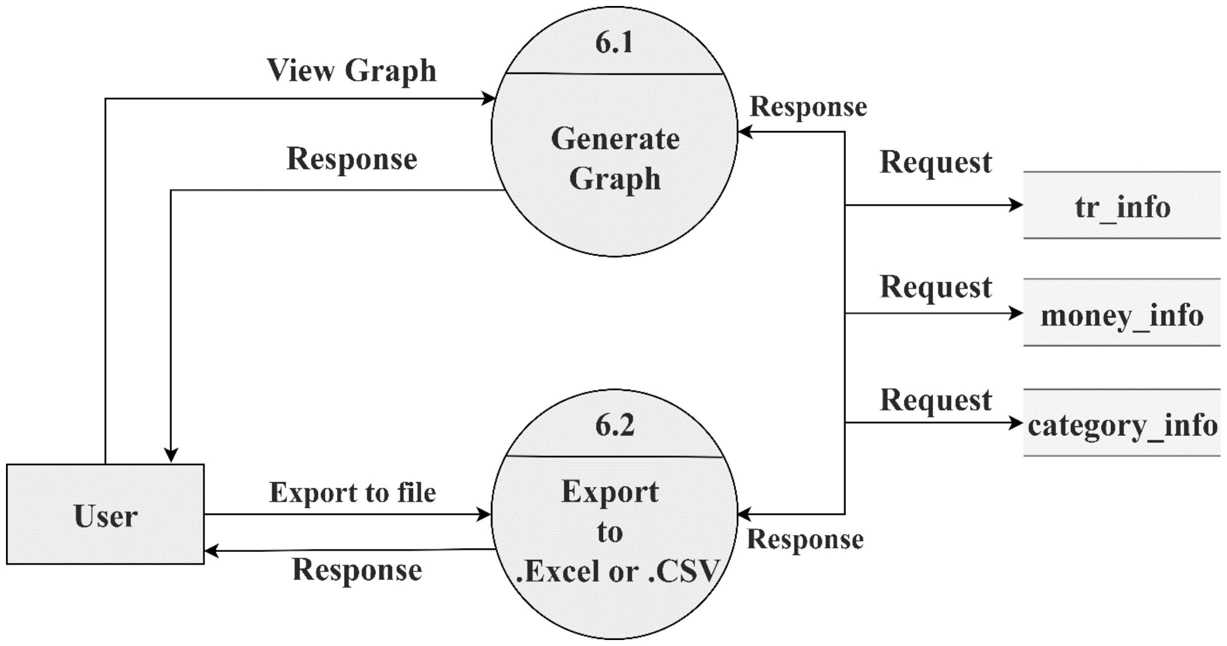
Level 2 DFD for process 4 Transaction Management



Level 2 DFD for process 5 Category Management



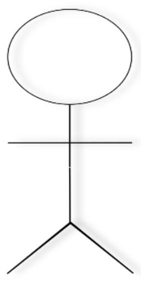
Level 2 DFD for process 6 Graph Management



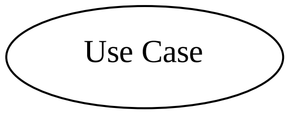
2.3 Use Case Diagram

- A use case diagram at its simplest is a representation of a user's interaction with the system.
- It shows the relationship between the user and the different use cases in which the user is involved.
- A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.
- The use cases are represented by either circles or ellipses.
- The purpose of the use case diagrams is simply to provide the high-level view of the system and convey the requirements in laypeople's terms for the stakeholders.
- Additional diagrams and documentation can be used to provide a complete functional and technical view of the system.

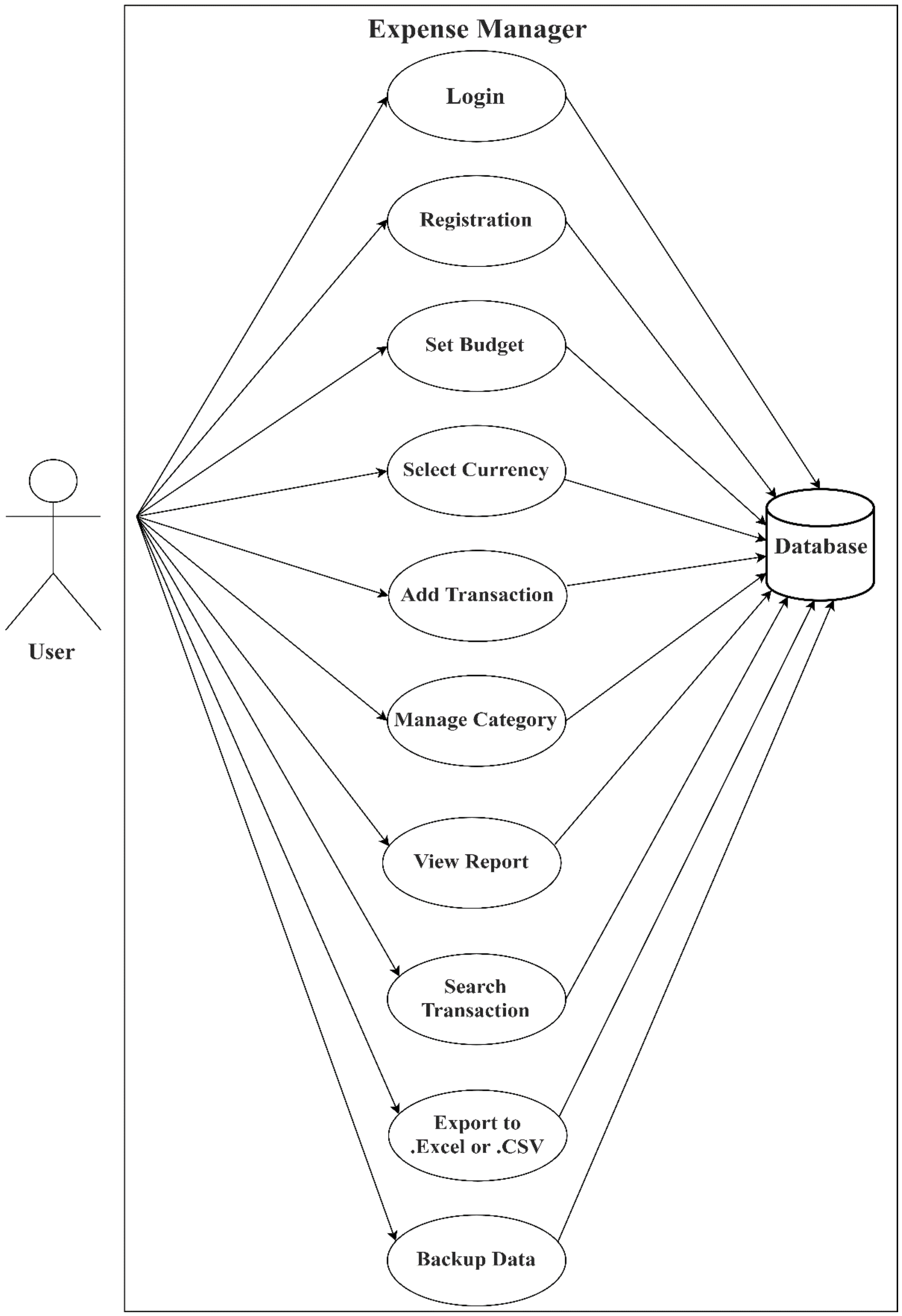
Symbols used in Use Case:



Actor



Use Case Diagram for Expense Manager



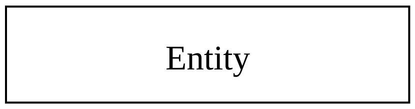
2.4 Entity Relationship Diagram

- In software engineering, an entity–relationship model (ER model) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way.
- Entity–relationship modeling was developed by Peter Chen and published in a 1976 paper.

Symbols used in ERD:

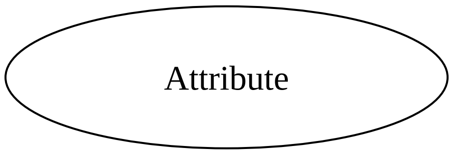
Entity:

- Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.



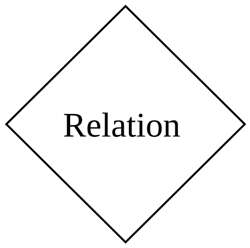
Attributes:

- Attributes are properties of entities. Attributes are represented by means of eclipses. Every eclipse represents one attribute.

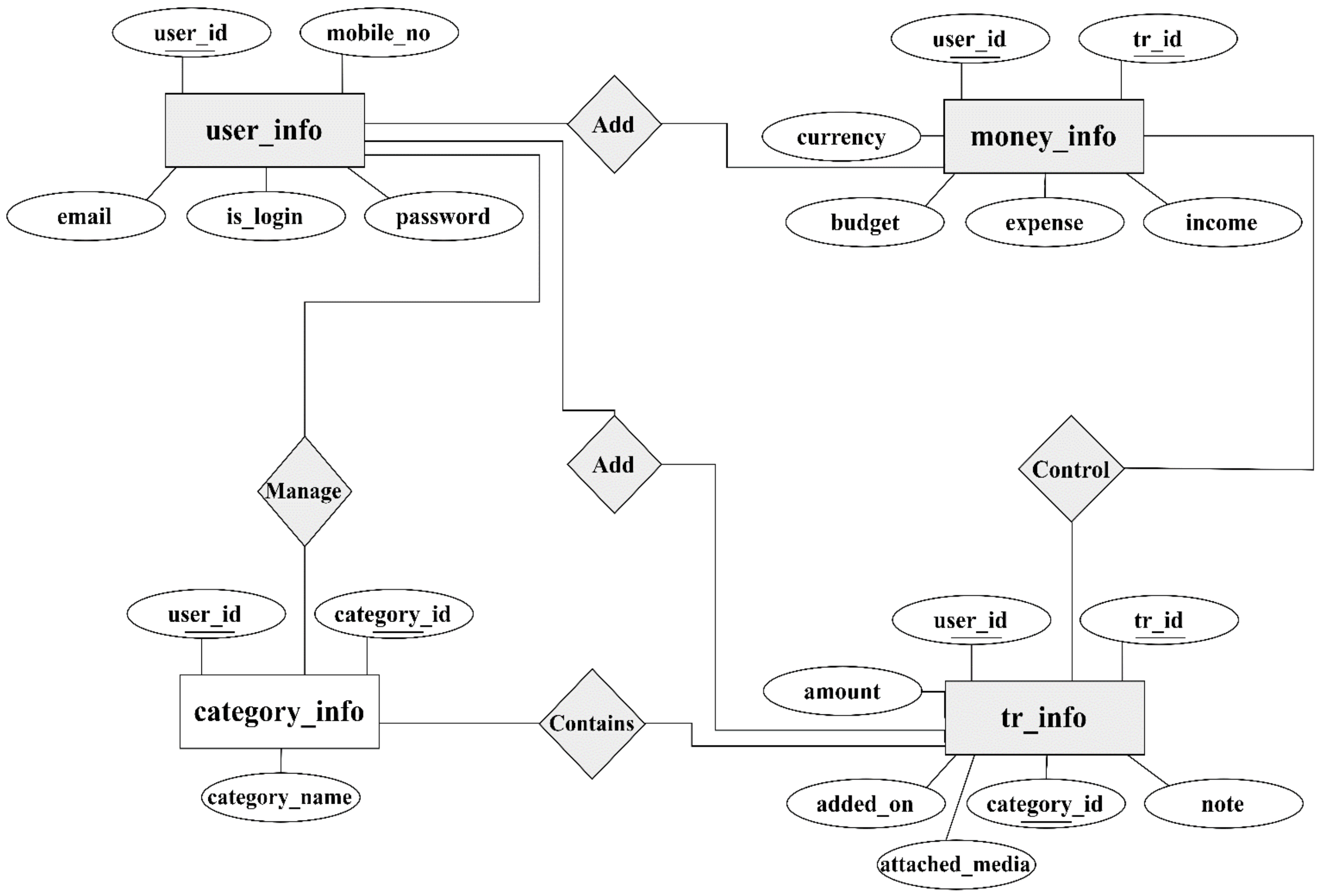


Relationship:

- Relationships are represented by diamond shaped box. Name of the relationship is written in the diamond-box. All entities (rectangles), participating in relationship, are connected to it by a line.



Entity Relationship Diagram for Expense Manager



2.5 Data Dictionary

user_info

- This table store information about username, password and status.

Column	Type	Null	Default	Constraint
user_id	number	No	Auto Generated	Primary Key
password	string	No	-	-
email	string	No	-	Unique Key
mobile-number	number	No	-	-
is_login	number	No	0	-

tr_info

- This table store information about transaction such as transaction id, amount, and other detail about transaction when it is added.

Column	Type	Null	Default	Constraint
tr_id	number	No	Auto Generated	Primary Key
user_id	number	No	Auto Generated	Foreign Key
amount	number	No	-	-
added_on	timestamp	No	-	-
category_id	number	No	-	Foreign Key
note	string	Yes	-	-
attached_media	Base64 string	Yes	-	-

category_info

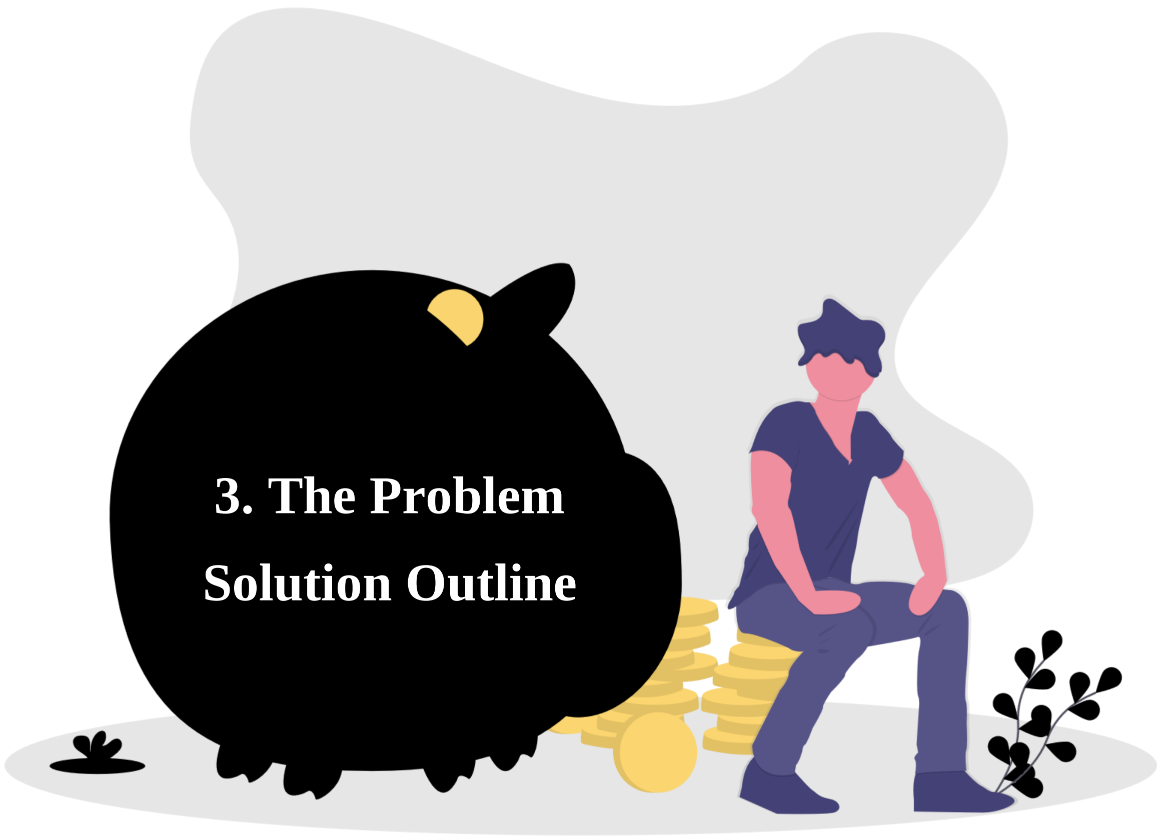
- This table store information about transaction such as category id, category name when it is added.

Column	Type	Null	Default	Constraint
category_id	number	No	Auto Generated	Primary Key
category_name	string	No	-	-
user_id	number	No	Auto Generated	Foreign Key

money_info

- This table store information such as budget and remaining balance when new transaction is added.

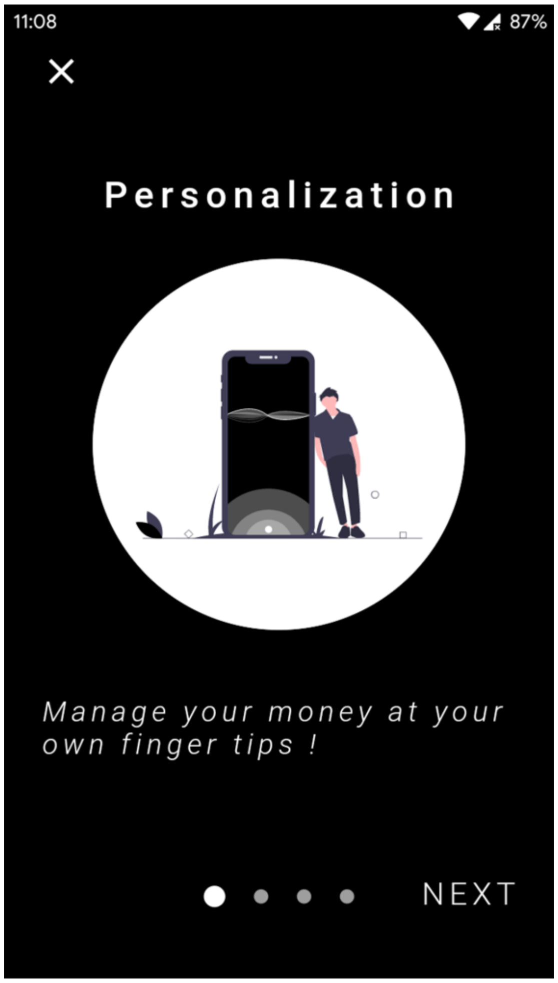
Column	Type	Null	Default	Constraint
tr_id	number	No	Auto Generated	Foreign Key
user_id	number	No	Auto Generated	Foreign Key
budget	number	No	0	-
currency	string	No	-	-
expense	number	No	0	-
income	number	No	0	-



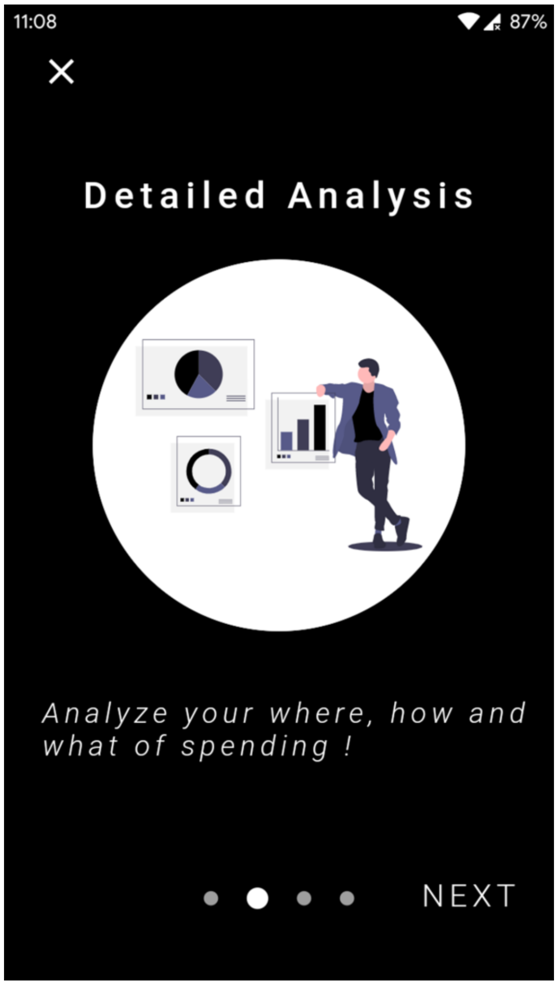
3. The Problem Solution Outline

3.1 Input Design/Output Design

- **Get-Started Pages UI**

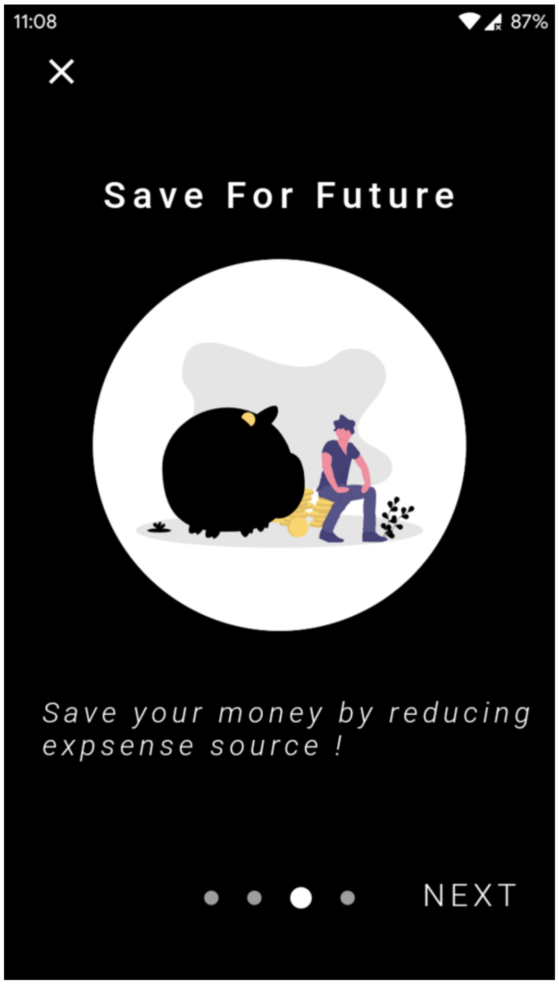


Manage your money at your own finger tips !



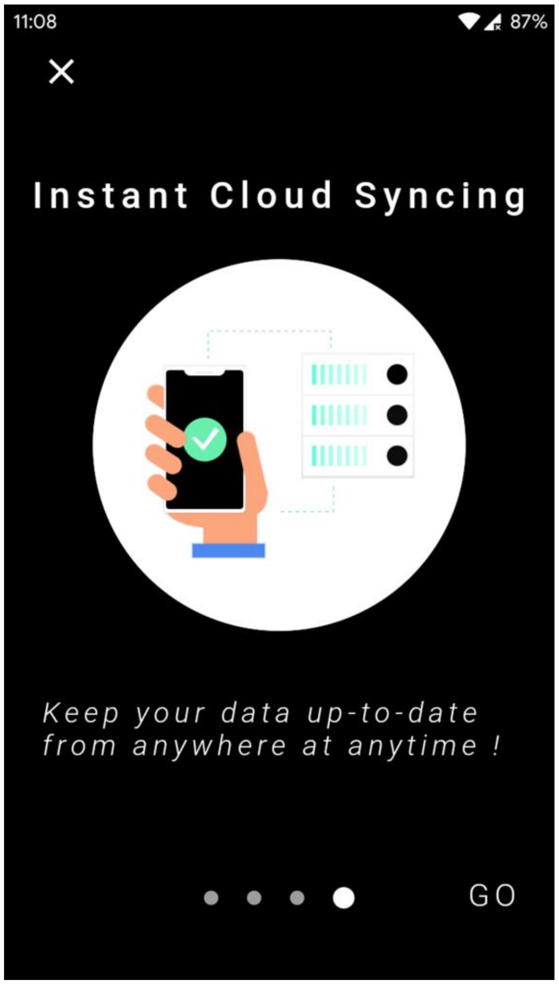
Analyze your where, how and what of spending !

- **Get-Started Pages UI**



Save For Future

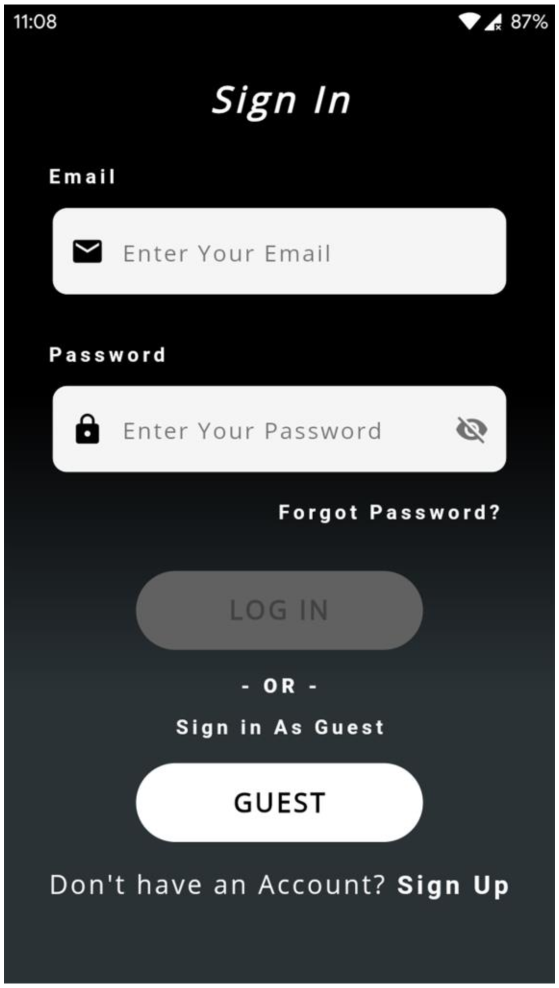
Save your money by reducing expsense source !

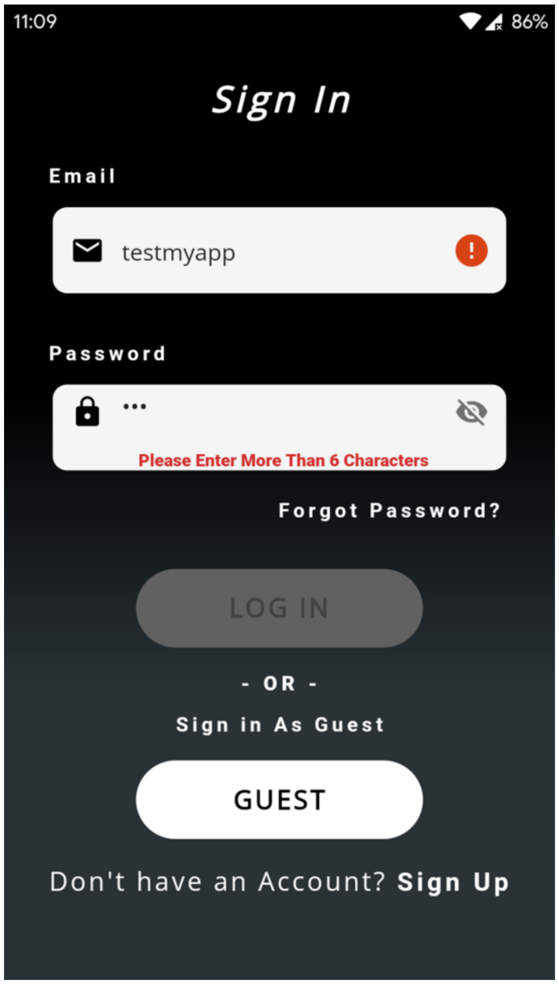


Instant Cloud Syncing

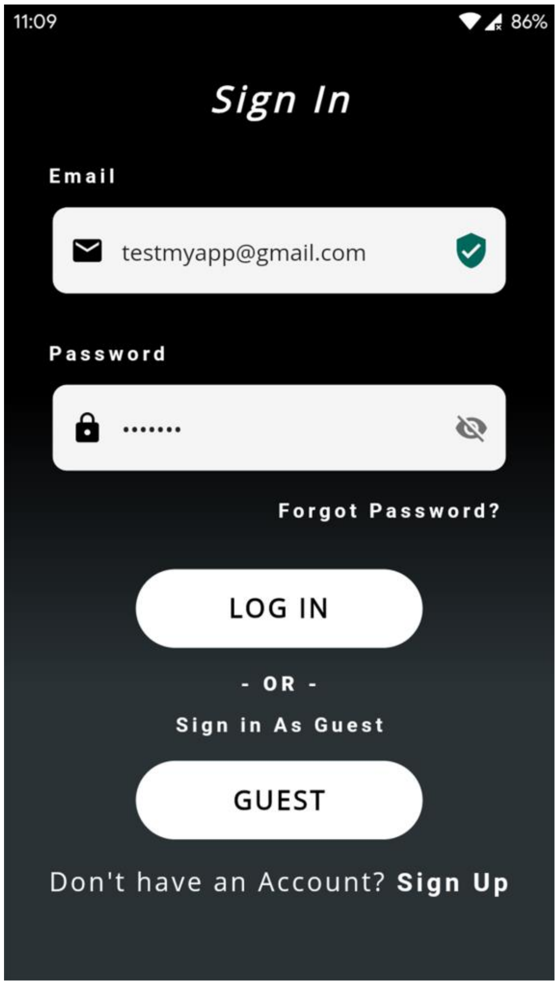
Keep your data up-to-date from anywhere at anytime !

- **Login Screen UI with Validation**

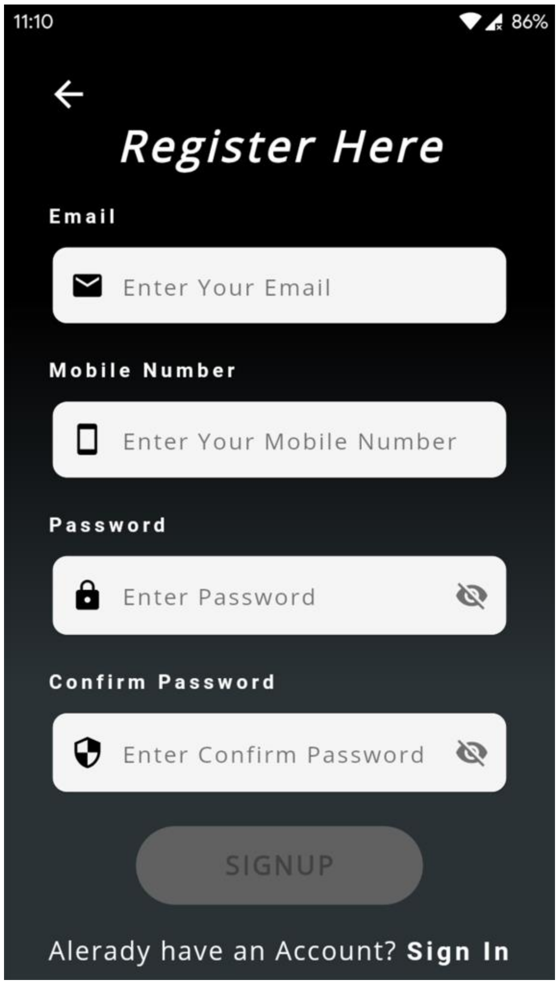




- **Login Screen UI with Validation**



- **Sign Up Screen UI with Validation**



11:10 86%

Register Here

Email

Enter Your Email

Mobile Number

Enter Your Mobile Number

Password

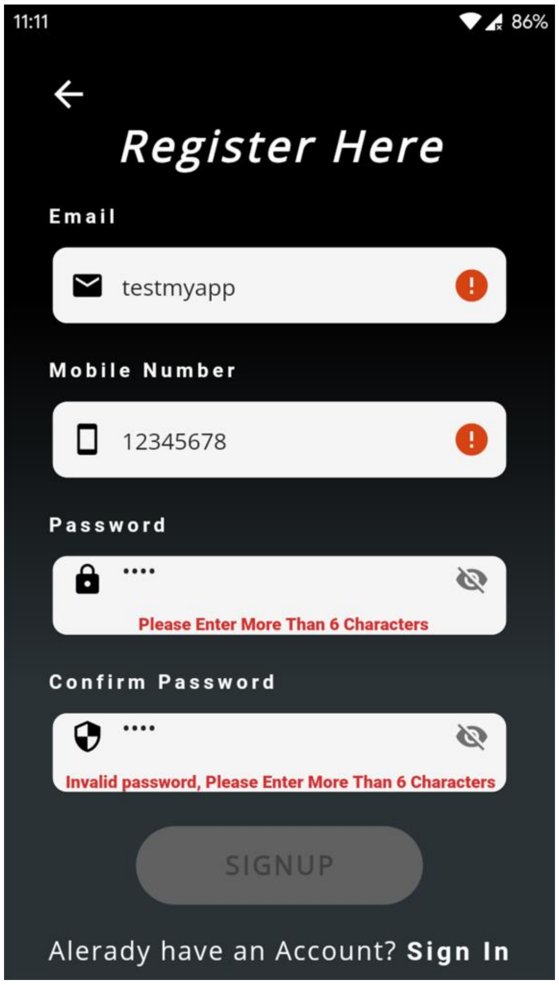
Enter Password

Confirm Password

Enter Confirm Password

SIGNUP

Alerady have an Account? **Sign In**



11:11 86%

Register Here

Email

testmyapp

Mobile Number

12345678

Password

••••

Please Enter More Than 6 Characters

Confirm Password

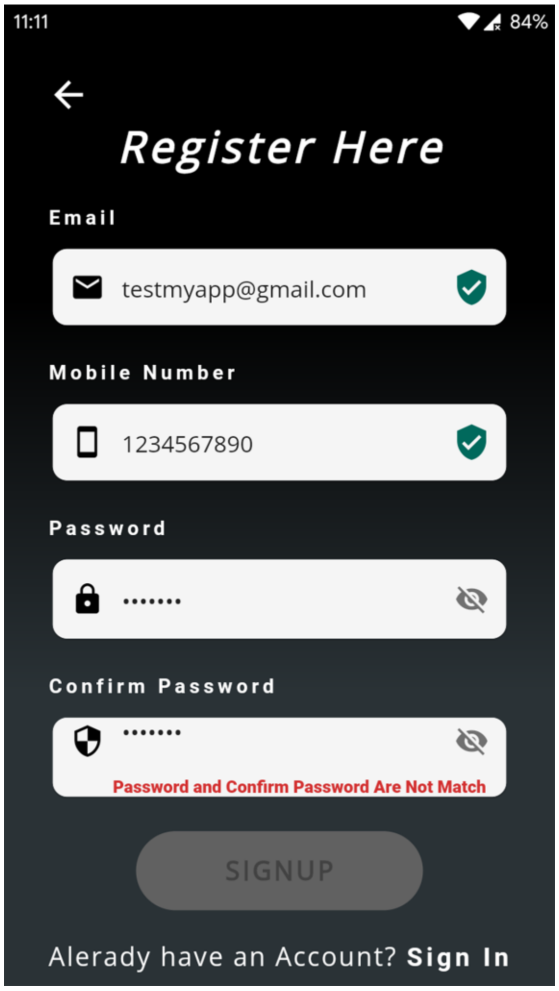
••••

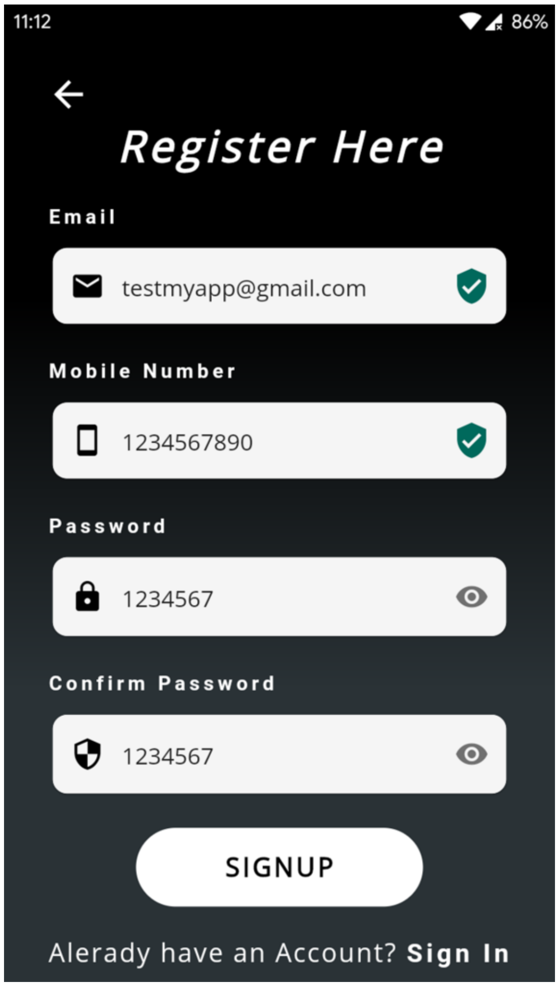
Invalid password, Please Enter More Than 6 Characters

SIGNUP

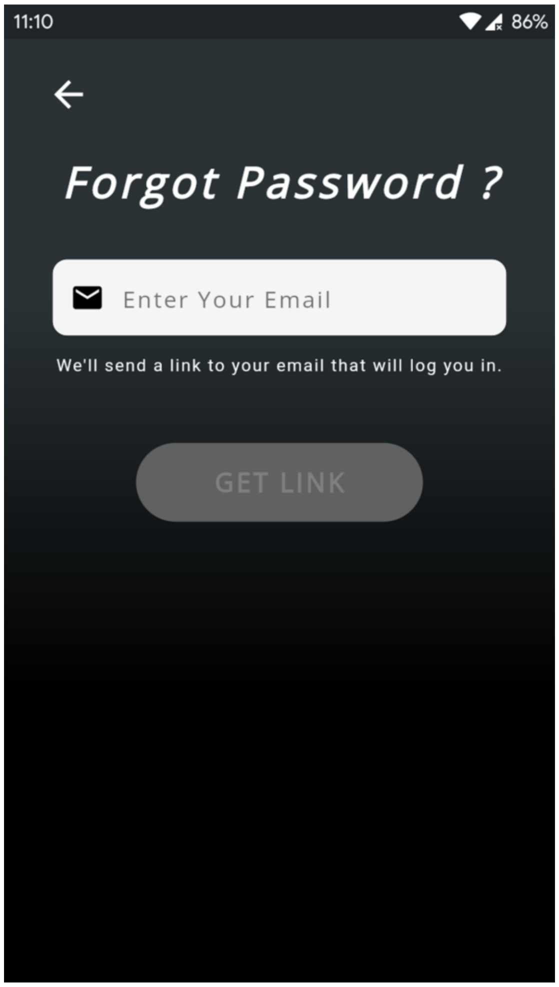
Alerady have an Account? **Sign In**

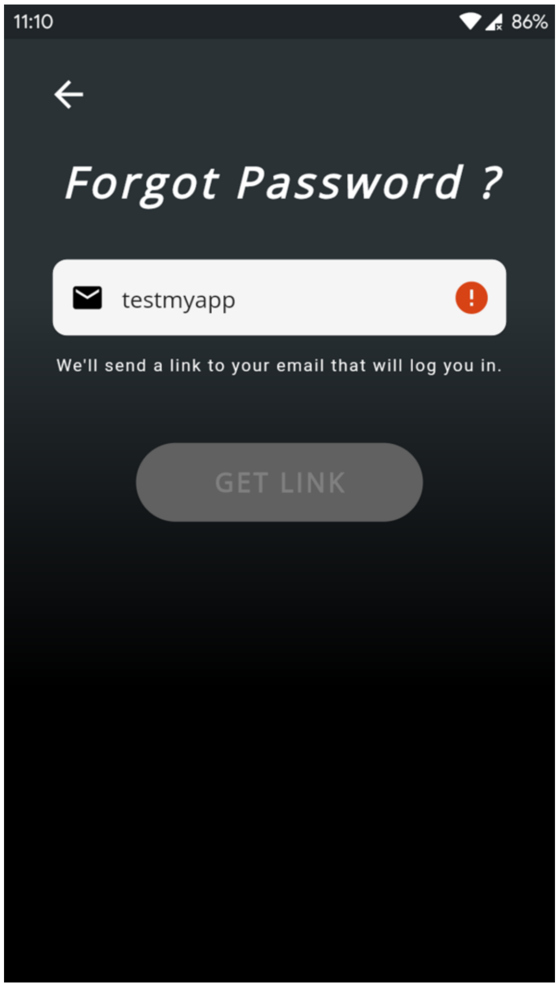
- **Sign Up Screen UI with Validation**



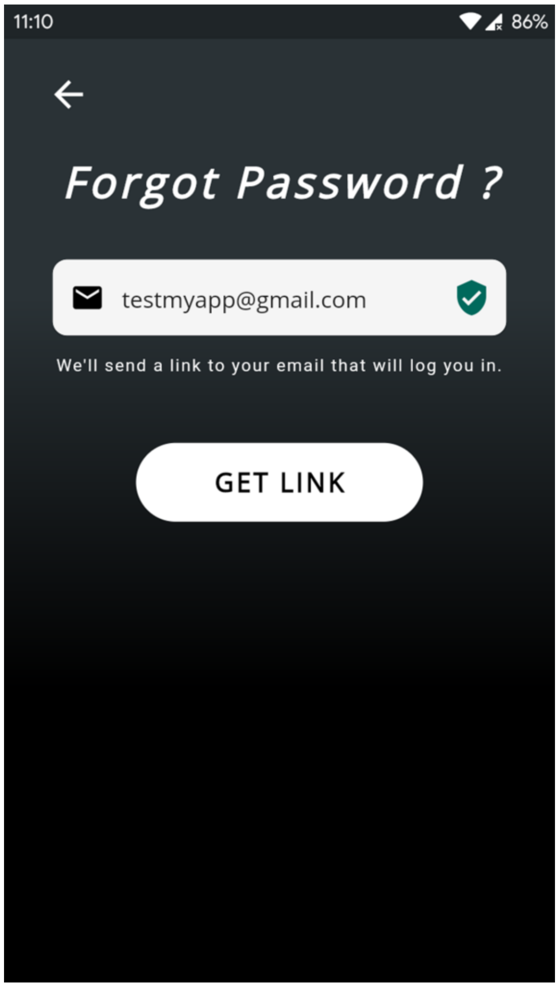


- **Forgot Password Screen Ui with Validation**

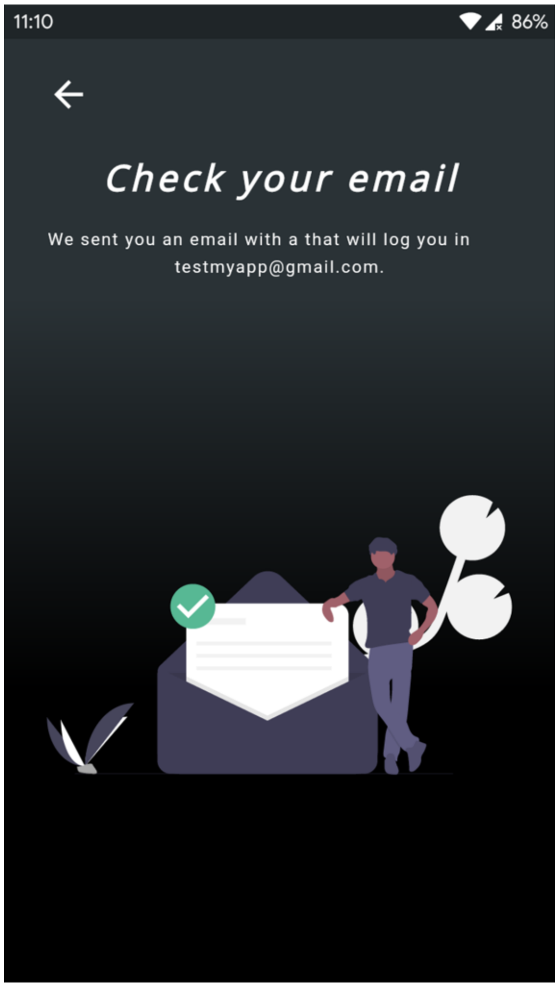




- **Forgot Password Screen Ui with Validation**

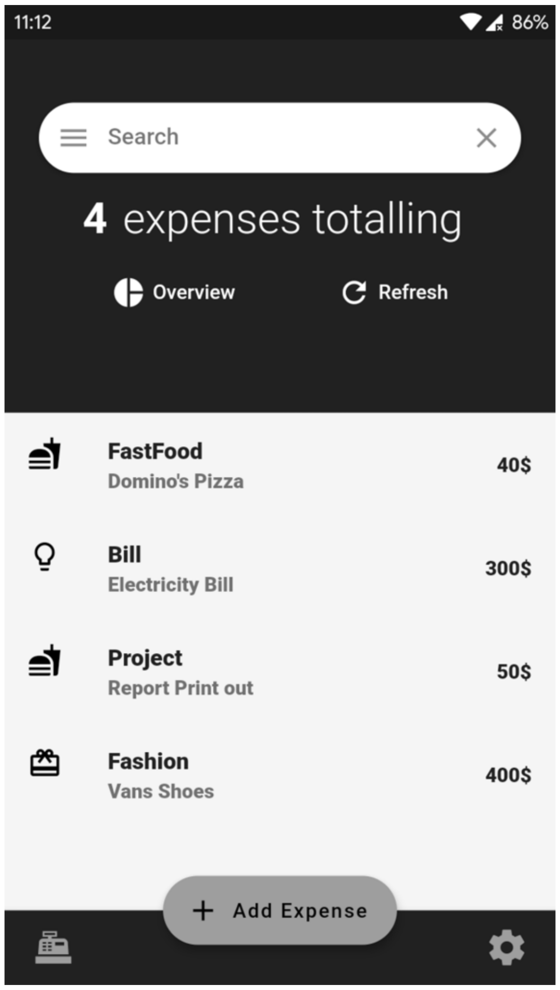


We'll send a link to your email that will log you in.



We sent you an email with a that will log you in testmyapp@gmail.com.

- **Dashboard Screen UI**



FastFood
Domino's Pizza
40$

Bill
Electricity Bill
300$

Project
Report Print out
50$

Fashion
Vans Shoes
400$

4. The Outline of Work to Be Carried Future

4.1 System Enhancement

- We are planning to manage offline database as well as online database.
- We are trying to display user guide so any type of user can use it easily.
- We are planning to make keep everything simple as possible.
- We are trying to keep UI as minimal as simple.
- We are planning to fast data sync across online as well as offline.
- We are trying to implement different types of charts like bar chart, pie chart which will help user to analyze their spending.

Conclusion

Expense Manager is cross platform finance management application which helps users to manage their expense and income. It helps user to know What, Where & How they spend their money. It helps to analyze their annual expenses using different types of charts. Additionally, it allows users to manage their expense at any time at anywhere without requiring internet all the time.

References/Bibliography

For Technologies

- https://flutter.dev/
- https://firebase.google.com/
- https://dart.dev/
- https://www.sqlite.org/index.html

For Feasibility study

- https://www.quora.com/What-does-economic-feasibility-mean

For Data flow diagram

- https://www.lucidchart.com/pages/data-flow-diagram

For Use case diagram

- https://www.techopedia.com/definition/25813/use-case

For ER diagram

- https://beginnersbook.com/2015/04/e-r-model-in-dbms/

For Data Dictionary

- https://www.tutorialspoint.com/What-is-Data-Dictionary

For Queries

- https://medium.com/flutter-community
- https://stackoverflow.com/questions/tagged/flutter
- https://stackoverflow.com/questions/tagged/firebase
- https://github.com/flutter/flutter/issues