

**SOFTWARE REQUIREMENTS SPECIFICATION (SRS)
ON Bucks-Bunny**



SUBMITTED BY
Second Year Information Technology
Ameya Kshirsagar(16070124025)
Akshada Mali(16070124029)
Prakruti Chandak(16070124041)
Tapan Bolia(16070124058)



**SYMBIOSIS INSTITUTE OF TECHNOLOGY
(A CONSTITUENT OF SYMBIOSIS INTERNATIONAL
UNIVERSITY)
Pune 412115 2017-18**

1.	<u>INTRODUCTION</u>	
1.1	<u>Purpose.....</u>	<u>4</u>
1.2	<u>Document Conventions</u>	<u>4</u>
1.3	<u>Intended Audience and Reading Suggestions</u>	<u>4</u>
1.4	<u>Project Scope.....</u>	<u>4</u>
2.	<u>OVERALL DESCRIPTION</u>	
2.1	<u>Product Perspective.....</u>	<u>4</u>
2.2	<u>Product Features.....</u>	<u>5</u>
2.3	<u>User Classes and Characteristics.....</u>	<u>6</u>
2.4	<u>Design and Implementation Constraints.....</u>	<u>7</u>
2.5	<u>Operating Environment.....</u>	<u>7</u>
2.6	<u>Assumptions and Dependencies.....</u>	<u>8</u>
3.	<u>SYSTEM FEATURES</u>	
3.1	<u>User Registration and Welcome.....</u>	<u>8</u>
3.2	<u>Group Creation and Management.....</u>	<u>9</u>
3.3	<u>Member-to-Member Transactions.....</u>	<u>10</u>
3.4	<u>Group History.....</u>	<u>11</u>
3.5	<u>Show All Debts.....</u>	<u>11</u>
3.6	<u>Help Menu.....</u>	<u>12</u>
4.	<u>OTHER NON-FUNCTIONAL REQUIREMENTS</u>	
4.1	<u>Performance Requirements.....</u>	<u>12</u>
4.2	<u>Safety Requirements.....</u>	<u>12</u>
4.3	<u>Security Requirements.....</u>	<u>13</u>
4.4	<u>Software Quality Attributes.....</u>	<u>13</u>
5.	<u>ANALYSIS MODELS</u>	
5.1	<u>Use-Case Diagram.....</u>	<u>14</u>
5.2	<u>Class Diagram.....</u>	<u>15</u>

5.3	Data Flow Diagram.....	16
6.	PROJECT SCHEDULING AND ESTIMATION	
6.1	Functional decomposition (WBS).....	18
6.2	Roles and Responsibilities.....	18
6.3	Task Duration.....	19
6.4	Size and Budget estimation.....	19
6.5	RMMM.....	21
7.	IMPLEMENTATION.....	22
8.	TEST CASES.....	23
9.	CONCLUSION AND FUTURE SCOPE.....	26

1.INTRODUCTION

1.1 PURPOSE

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the functionalities and purpose of the Bucks-Bunny application (system). This srs will cover each of the system's intended features, and offer a preliminary site of the software application's User Interface (UI).

1.2. INTENDED AUDIENCE

This document is useful for all individuals participating in and supervising the Bucks-Bunny project.

1.3. PROJECT SCOPE

The Bucks-Bunny system is composed of two main components: a client-side server-side. The system is designed to for the process of tracking and settling shared expenses for easy budgeting. Scenarios where the application can be used include paying rent, splitting a bill at a party, sharing trips expenses, etc.

1.4. READING SUGGESTIONS

Readers interested in a brief overview of the product should focus on the rest of Introduction, as well as Overall Description, which provide a brief overview of each aspect of the project. The readers may also be interested in Key Milestones which lays out a summarised timeline of the project. Readers who wish to explore the features of Bucks-Bunny in more detail may go through System Features, which expands upon the information as given in the main overview.

Readers who have not found the information they are looking for, should go through check Other Requirements, which includes any additional information which does not fit logically into the other sections.

2.OVERALL DESCRIPTION

2.1. PRODUCT PERSPECTIVE

The Bucks-Bunny project is a new, self-contained product intended for use on the Android platform for day to accounting and budgeting ease.

2.2. PRODUCT FEATURES

The following list offers a brief outline and description of the main features and functionalities of the Bucks-Bunny system.

1. USER REGISTRATION & WELCOME

- Only appears once (the first time the application is run)
- Allows the user to register with the Bucks-Bunny server
- Enables the user to customize his/her account settings and preferences

2. GROUP CREATION & MANAGEMENT

- Streamlines the process of creating and organizing groups
- Provides support for multiple groups
- Allows the user to add group members manually or from contacts list

3. MEMBER-TO-MEMBER TRANSACTIONS

- Enables group members to simulate transfers of debt, payments made, etc.
- Adjusts member balances accordingly
- Records relevant information (amount paid, members involved, etc.)

4. FINAL DEBT RESOLUTION

- Calculates the most efficient method of sorting out debts
- Notifies group members of unresolved debts, credits, etc.
- Offers the option to disband a group once all payments are made

5. GROUP HISTORY

- Automatically records all transactions and bills posted to each group
- Provides users with access to a detailed history of transactions
- Supports sorting transactions by date, amount, payer, etc.

6. SHOW ALL DEBTS

- Enumerates all of a user's unresolved debts across each group he/she is a part of.
- Provides easy access to relevant information (past transactions, group info, etc.)
- Offers the option to resolve a debt (or debts) immediately

7. HELP MENU

- Displays a list of topics covering the different components of Bucks-Bunny
- Offers detailed information on each feature, menu, etc.
- Can be accessed at any time via the Settings menu

2.3. USER CLASSES AND CHARACTERISTICS

The Bucks-Bunny project is meant to offer a shared expenses solution that is faster, easier, and more convenient than manually calculating and handling debts. Consequently, the application will have little or no learning curve, and the user interface will be as intuitive as possible. Thus, technical expertise and Android experience should not be an issue. Instead, anticipated users can be defined by how they will use the product in a particular situation. The following list categorizes the scenarios in which Bucks-Bunny is expected to be utilized:

Bucks-Bunny: POTENTIAL SCENARIOS

1. Long-term recurring expenses (e.g. rent, groceries, utilities)

- Key functions:
 - Keep track of expenses
 - Notify users when debts are incurred
 - Record who has paid and who still owes
- Requirements:
 - Method for inputting expenses into the application
 - Support for automated notifications
 - Database containing current debts, past transactions, etc.

2. Short-term recurring expenses (e.g. travel costs – gas, food, hotel)

- Key functions:
 - Add new expenses (quickly and easily)
 - Record who is paying and what he/she is paying for
 - Update member balances on the fly
- Requirements:
 - Simple and intuitive user interface for adding expenses
 - Expense-tracking system
 - Automated, background algorithm for calculating debts

3. Single expense (e.g., splitting a bill at dinner)

- Key functions:

- Create a group (quickly and easily)

- Add non-registered individuals to the group

- Quickly calculate each member's balance

- Requirements:

- Simple and intuitive user interface for adding groups

- Support for group members without the Bucks-Bunny application

- Option to calculate all balances on the spot

These groups are not meant to separate or categorize users, just the different situations in which Bucks-Bunny is likely to be used. In fact, a user may utilize the application for all of these scenarios simultaneously. This is another defining feature of the Bucks-Bunny system: support for multiple groups. This functionality allows a user to track expenses pertaining to several unrelated groups at the same time.

It is crucial that each of these situations be fully supported in the final product so as to

maximize the overall value of the product. It is also important that the application be as user-friendly as possible, otherwise it will not be a viable alternative to handling shared expenses manually. Most importantly, the application must be reliable. Regardless of the situation, the application must accurately distribute costs. There is zero tolerance for error when dealing with financial transactions.

2.4. DESIGN AND IMPLEMENTATION CONSTRAINT

The primary design constraint is the mobile platform. Since the application is designated for mobile handsets, limited screen size and resolution will be a major design consideration. Creating a user interface which is both effective and easily navigable will pose a difficult challenge. Other constraints such as limited memory and processing power are also worth considering. Bucks-Bunny is meant to be quick and responsive, even when dealing with large groups and transactions, so each feature must be designed and implemented with efficiency in mind.

2.5. OPERATING ENVIRONMENT

The main component of the Bucks-Bunny project is the software application, which will be limited to the Android operating system. The application is not

resource- or graphics-intensive, so there are no practical hardware constraints. Beyond that, the application is a self-contained unit and will not rely on any other Android-related software components.

2.6. ASSUMPTION AND DEPENDENCIES

TIME DEPENDENCIES

Core features are crucial to the basic functionality of the Bucks-Bunny application. These features must all be implemented in order for the application to be useful. Thus, the implementation of these features is entirely dependent upon the time spent designing and implementing the core features. The final decision on whether or not to implement these features will be made during the later stages of the design phase.

3. SYSTEM FEATURES

3.1. USER REGISTRATION AND WELCOME

When the application is installed and run for the very first time, the user is presented with an initial registration/welcome screen. This screen prompts the user to create an account on the Bucks-Bunny server using the email address. The user also enters a "Display Name" Which will be the name that is shown as their handle within the groups. Completing this process will create and store an account for the user on the Bucks-Bunny server, enabling all of the application's synchronization capabilities.

3.1.1 STIMULUS/RESPONSE SEQUENCE

Input: Bucks-Bunny app launched from android home screen

Output: The user is prompted to enter an email address & display name

Input: This information is sent to server and stored in database

Output: Registration is completed & user is taken to main screen

3.1.2 USER REQUIREMENTS

VALID EMAIL ADDRESS

The user cannot proceed until a valid email address is entered. The application will verify that the user's input is consistent with the format of an email address.

3.1.3 SYSTEM REQUIREMENTS

SECURE DATABASE SYSTEM

The application must insure that the user's information is encrypted and safely stored.

3.2 GROUP CREATION AND MANAGEMENT _____

The "Groups" screen will be the main screen of the application. From this screen, users will be able to view and manage existing groups. Groups may be created by adding members from the user's contacts or by manually entering an email address and name

3.2.1 STIMULUS/RESPONSE SEQUENCES: GROUP CREATION

Input: The user selects "Create a New group" from the main Group screen

Output: The user must choose a unique name to be used as the group's identifier. The user can add members to the group from contacts stored on the phone or via manual entry (email and display name are required).

- Users who are not registered with the Bucks-Bunny server are designated on the server as "offline users".
- These users cannot participate in transactions directly, so the leader is responsible for reconciling their balances.

At any time, these offline users may register a Bucks-Bunny account (using the same information stored on the server) and take control of their transactions

Input: The user finishes selecting members and confirms that group is complete.

Output: The user is then designated as the leader of the group

3.2.2 STIMULUS/RESPONSE SEQUENCE: GROUP MANAGEMENT

Input: From the management Screen, the leader can choose from following options

- Remove member(s)
- Add member(s)
- Disband group (leave all debts hanging)

- Disband and Remove Group
 - Final Balance sheet is posted to emails and notifications
 - Group is removed from “Groups
- STOP Group and Reconcile Group
 - The Group still persists in the main screen but new charges cannot be added.
 - Transactions to resolve the final balances may still be processed within the group
- Manage Transactions (Confirm/Deny member to member payment)
- Notify All (sends current balance notification to all members in group)

3.2.3 USER REQUIREMENTS

VALID EMAIL ADDRESS(ES)

The group leader must enter valid email add. from group members in order to obtain the full functionality of the application.

3.3. MEMBER-TO-MEMBER TRANSACTIONS

This feature represents a real world transaction between two or more members of a group. A common scenario would involve the user resolving his/her debt by making payments to other group members. The resulting changes in balances are calculated automatically and displayed for the users involved.

3.3.1 STIMULUS/RESPONSE SEQUENCES

Input: The user selects Member to member Transaction and inputs the following data:

- Transaction Description(Optional)
- Member(s) to be paid
- Amount to be paid(to each member if multiple)

The user confirms the transaction

Output: Each involved member has his/her balance adjusted automatically. Only an active internet connection is required.

3.3.3 SYSTEM REQUIREMENTS

REAL-TIME BALANCING

The application must be able to update each user's balance on the fly.

3.4. GROUP HISTORY

This screen provides a view of all transactions and bills that occurred within a group. This list will be presented in chronological order by default, but can also be sorted by paper, amount , etc. It will show the names of members involved in each transaction and amounts paid/received, and the user has the option of viewing each item in more detail by selecting it. The detailed view display all members involved in the bill/transaction, any comments about it, and any additional information that was included when it was created.

3.5.1 STIMULUS/RESPONSE SEQUENCES

The user is presented with a list of all transactions/bills posted to the current group.

- The user may sort these items by date, amount, payer, etc.

Input: The user select any one of the transactions/bills for detailed viewing.

Output: Upon Selection, a dialog is presented with the details of the transaction/bill.

3.5. SHOW ALL DEBTS

This is a global feature that will enumerate an individual's debt across all groups of which he/she is a member. There will also be an option to reconcile all debts from this screen. Selecting this option will show the user a list of transactions which are required before his/her debts can be resolved. The user will be able to initiate these transactions within the app from this screen.

3.6.1 STIMULUS/RESPONSE SEQUENCES

Input: The user enters this screen from the popup context menu from any other screen in the app.

The user can see his/her status in each of the groups he/she is a member of **Output:** Selecting a specific group from this list displays that group's History page.

The user has the option to resolve his/her debts with a transaction for each group.

3.6. HELP MENU

The Help menu is meant to answer any questions the user may have while using Bucks-Bunny. The menu displays a list of topics related to features, menus, and the app in general. The user can select any of these topics to access further information and explanations.

3.7.1 STIMULUS/RESPONSE SEQUENCES

Input: From any screen, the user may press the Settings button built into the phone.

Output: This displays the Settings menu, where he/she can then select the Help menu.

The user is presented with a list of help topics which he/she can scroll through.

Input: Once the user selects a topic, more information on that topic is displayed.

Output: The user can navigate back to the Help screen or exit the Help menu altogether.

4. OTHER NONFUNCTIONAL REQUIREMENTS

4.1 PERFORMANCE REQUIREMENTS

The cost-division algorithms used by in application will be highly efficient, taking only a fraction of second to compute.

4.2 SAFETY REQUIREMENTS

The only potential safety concern associated with this application applies to virtually all handset apps: Bucks-Bunny should not be used while operating a vehicle or in any other situation where the user's attention must be focused elsewhere.

4.3 SECURITY REQUIREMENTS

This application assumes that only the user or whoever he/she allows will have access to his/her Android handset.

4.4 SOFTWARE QUALITY ATTRIBUTES

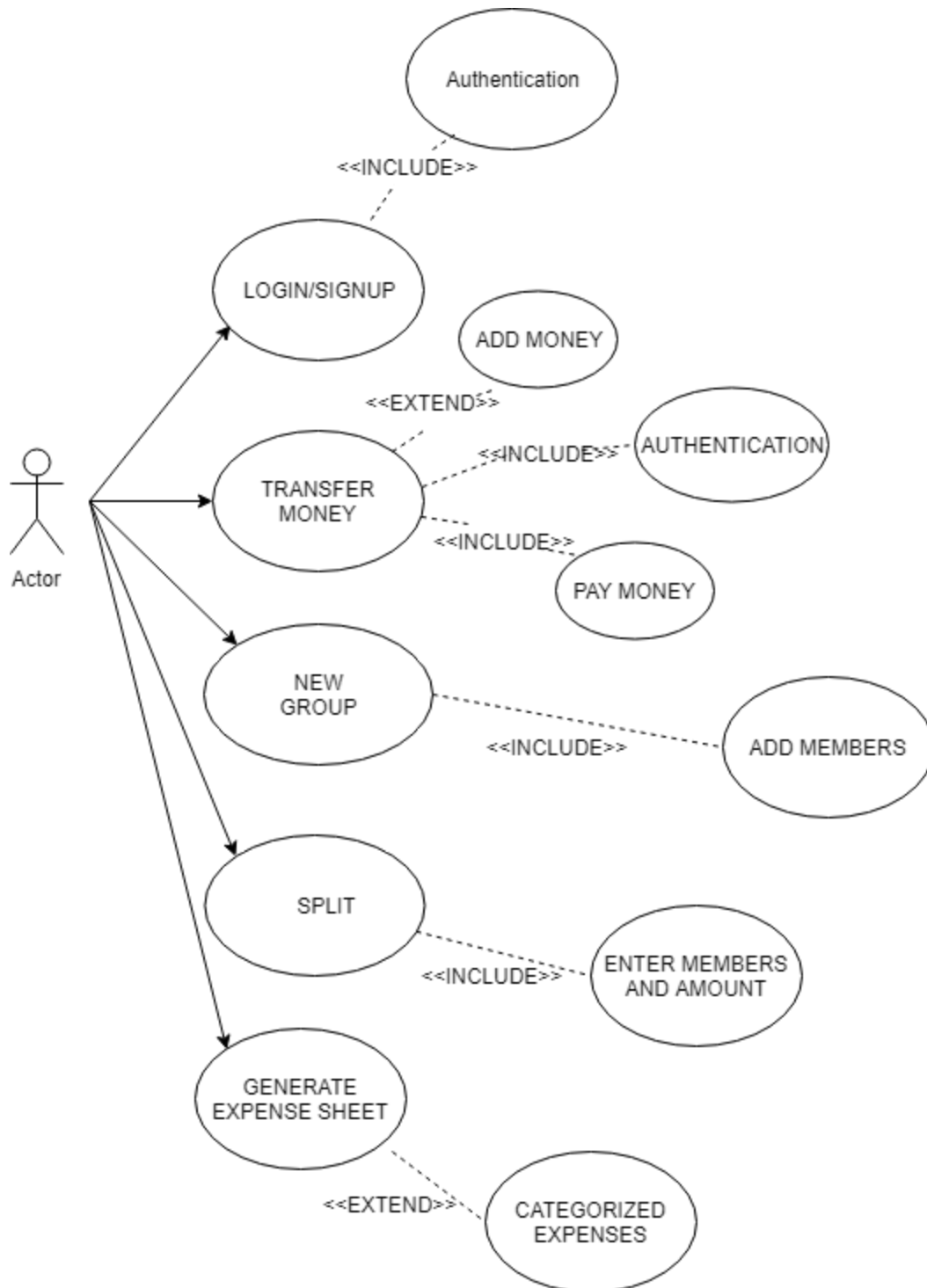
The app will be presented and organized in a manner that is both visually appealing and easy for the user to navigate. There will be feedbacks and visual cues such as notifications to inform users of updates and pop-ups to provide users with instructions.

To ensure reliability and correctness, there will be zero tolerance for errors in the algorithm that computes and splits expenses between group members. T

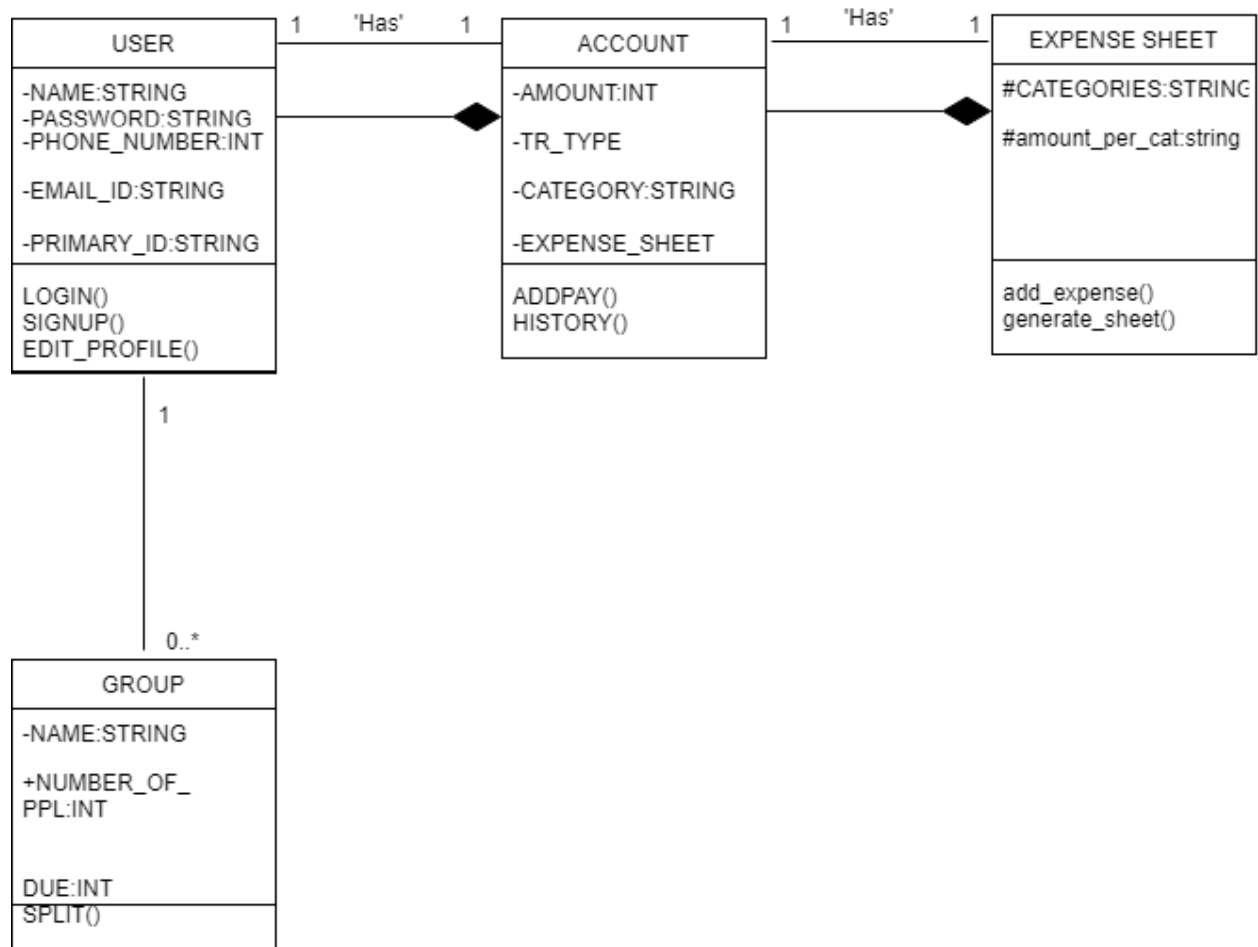
Overall, the app balances both the ease of use and the ease of learning. The layout and UI of the app will be simple enough that users will take no time to learn its features and navigate through it with little difficulty.

5. ANALYSIS MODELLING

5.1 USE-CASE DIAGRAM



5.2 CLASS DIAGRAM

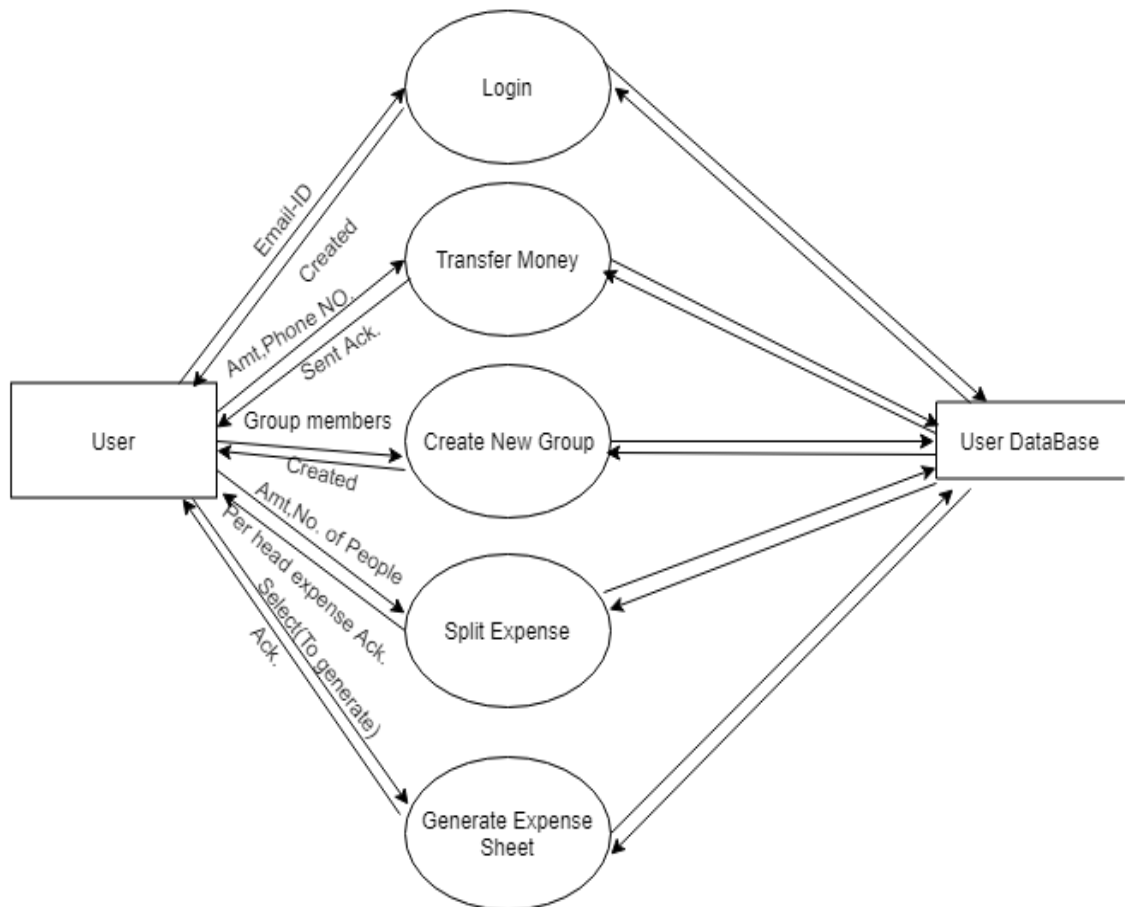


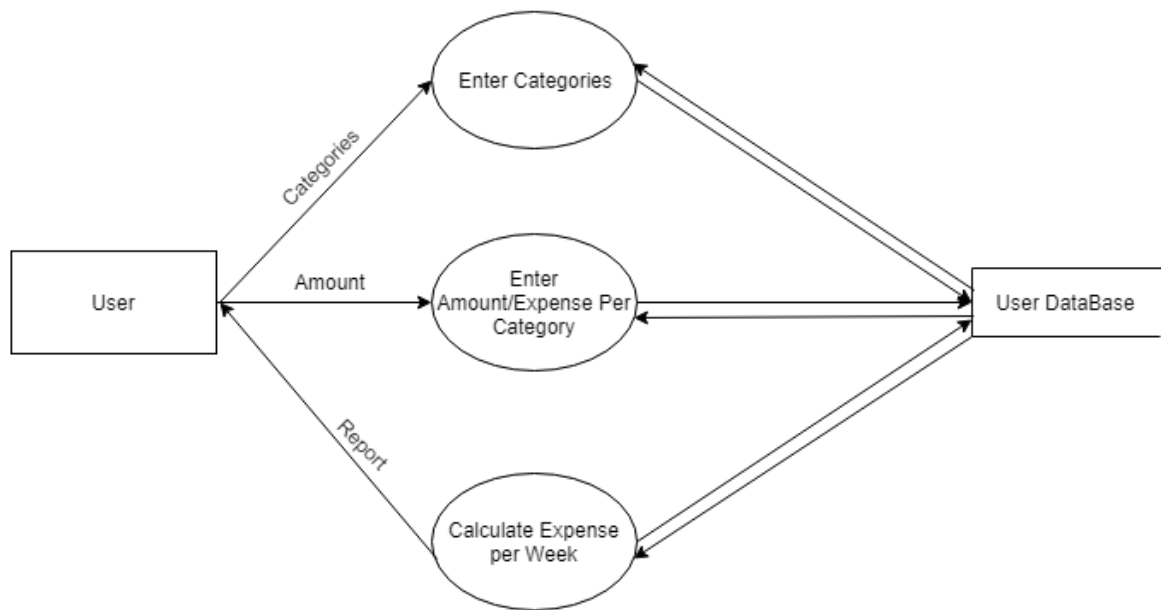
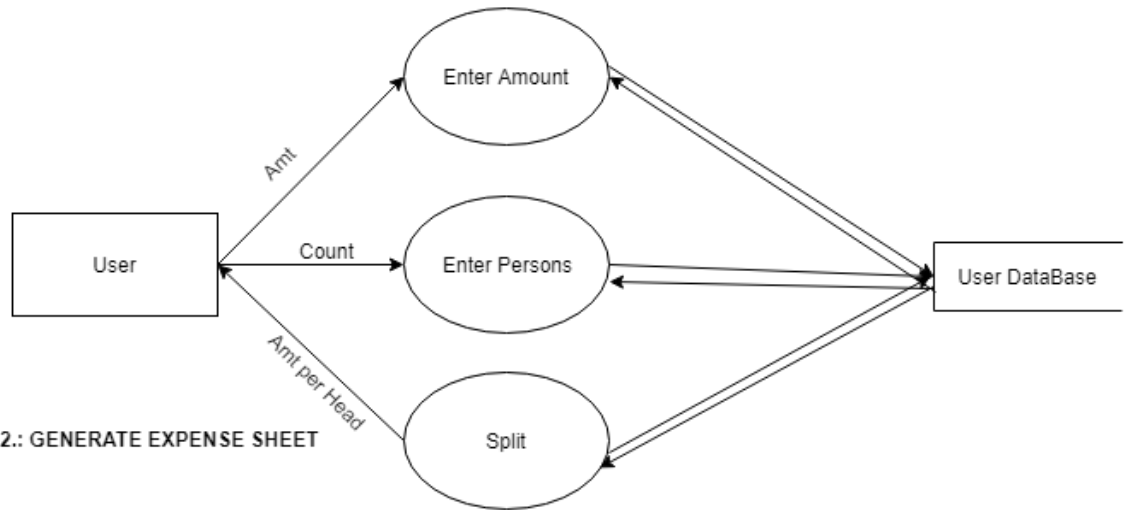
5.3 DFD DIAGRAM

LEVEL 0



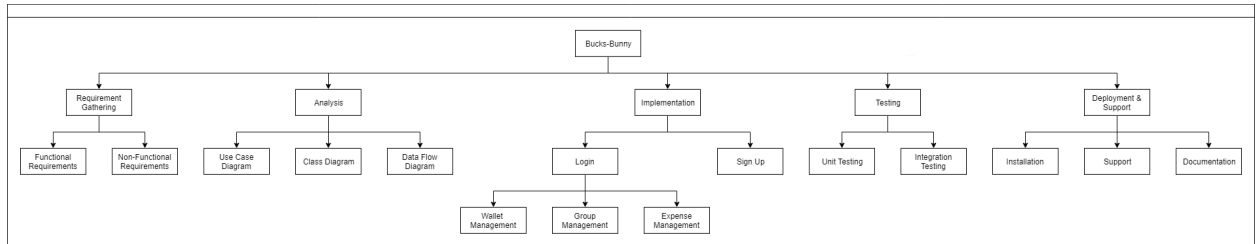
LEVEL 1





6. SCHEDULING AND ESTIMATION

6.1 Functional Decomposition(WBS)



[View WBS diagram](#)

6.2 Roles and Responsibilities

A: Ameya

B: Tapan

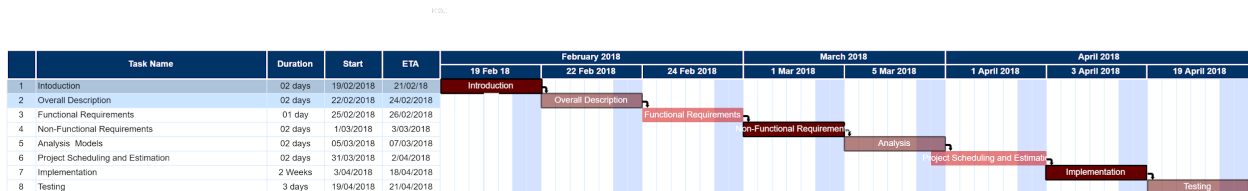
C: Akshada

D: Pravruti

Tasks	SubTasks	Responsibilities
Requirement Gathering		A,B,C,D
Analysis	Use Case Diagram	B,C
	Class Diagram	A,B
	Data Flow Diagram	C,D
Implementation	Sign Up	B,C
	Login	A,D
Wallet Management		A,C
Group Management		B,D
Expense Management		B,C

Testing	Unit Testing	A,B,C,D
	Integration Testing	A,B,C,D
Deployment & Support		A,B,C,D

6.3 Task Duration (GANTT Chart)



[View Gantt chart](#)

6.4 Size and Budget Estimation

The project size,duration and cost is calculated using LOC and basic COCOMO technique.

LOC:

The project has major five functions.The following table illustrates the functionalities and their size estimates:

Sr No.	Function	Optimistic	Most Likely	Pessimistic
1.	Signup/ Login	80	90	100
2.	Pay Money	350	400	450

3.	Expense Sheet	450	500	550
4.	Group	90	100	110

$Ev = (\text{optimistic} + 4 * \text{Most Likely} + \text{pessimistic}) / 6$

$Ev(1) = 0.09$

$Ev(2) = 0.4$

$Ev(3) = 0.5$

$Ev(4) = 0.1$

$\text{Total} = (Ev1 + Ev2 + Ev3 + Ev4) = 1.09 \text{KLOC}$

$\text{COST} = \text{Cost/LOC} * \text{LOC}$

Considering $\text{Cost/LOC} = 80/\text{LOC}$

$\text{COST} = \text{Rs } 87,200$

COCOMO Technique(Time estimation)

Our project fits under the organic category since we are a small team with experienced members. Using basic COCOMO model for effort and time estimation:

$\text{EFFORT} = a1x (\text{KLOC})^{a2} \text{ PM}$

$\text{TDEV} = b1x (\text{EFFORT})^{b2} \text{ MONTHS}$

For organic project:

$a1 = 2.4 \quad b1 = 2.5$

$a2 = 1.08 \quad b2 = 0.32$

Modules are:

1.Login-90

2.Pay Money-400

3.Expense sheet-500

4.Make group-100

Total LOC=1090LOC

Effort = $2.4 \times (1.09)^{1.08} = 3$ PM


Tdev= $2.5 \times (3)^{0.32} = 3.55$ M

6.5 Risk Mitigation, Monitoring and Management (RMMM)


Risk	Impact	Probability	Risk mitigation,monitoring, management
Loss of data	catastrophic	0.8	<ul style="list-style-type: none">• Keep second copy of the data• Track and retrieve from second copy of data• Caused by data overflow thus increasing the storage size
Unsatisfactory user response	marginal	0.5	<ul style="list-style-type: none">• Check already existing apps• Add updates to get better reviews• Provide refer and win opportunity
Error prone,unmanageable code	catastrophic	0.8	<ul style="list-style-type: none">• Divide project into appropriate modules• Identify data flow between functions• Fix errors by analysing bug report
No updates available	negligible	0.6	<ul style="list-style-type: none">• Keep add-ons and update for each function• Take reviews from the current users• Implement updates from time to time

7. IMPLEMENTATION:

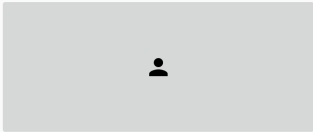
Wallet



Group


 Create New Group


My Account





I am someone

Contact Us

 Group

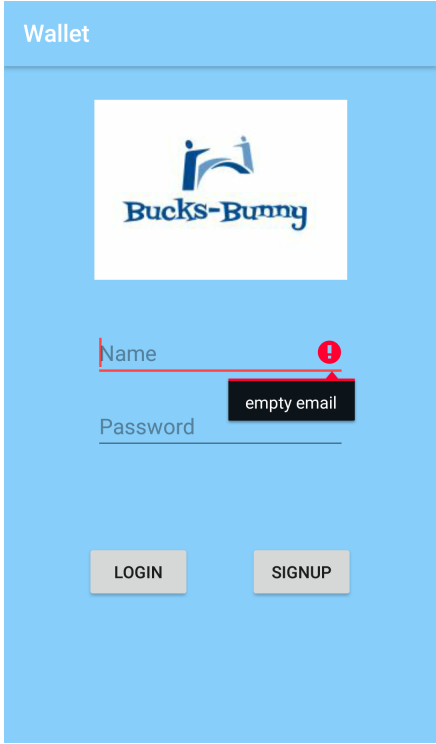
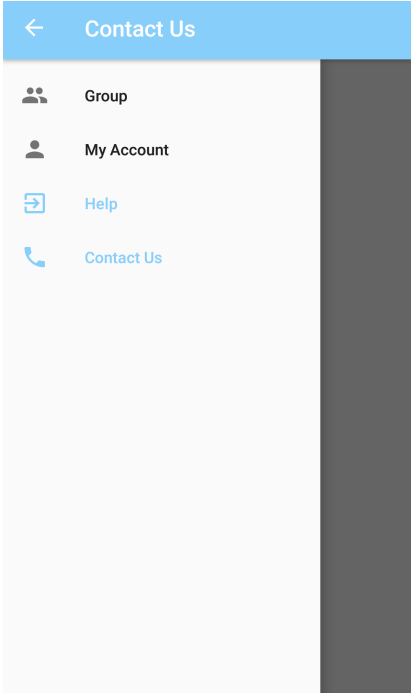
 My Account

 Help





 Contact Us

8. TEST CASES:

a)Login

Login fail	Login successful
 <p>The screenshot shows the 'Wallet' login page for 'Bucks-Bunny'. It features a logo at the top, followed by input fields for 'Name' and 'Password'. A red error message 'empty email' is displayed next to the 'Name' field, indicated by a red line and an exclamation mark icon. Below the input fields are 'LOGIN' and 'SIGNUP' buttons.</p>	 <p>The screenshot shows the 'Contact Us' screen after a successful login. It features a blue header with a back arrow and the text 'Contact Us'. Below the header is a list of menu items: 'Group', 'My Account', 'Help', and 'Contact Us', each with a corresponding icon. A dark grey vertical bar is visible on the right side of the screen.</p>

b)Sign Up

Signup fail	Signup successful
<div><div>Wallet</div><div>Name</div><div><div>!</div><div>name not given</div></div><div>Phone Number</div><div>E-Mail</div><div>Password</div><div>Confirm Password</div><div>SIGN UP</div></div>	<div><div>← Contact Us</div><div><div> Group</div><div> My Account</div><div> Help</div><div> Contact Us</div></div></div>

C)Add expense to group

<u>Fail</u>	<u>Successful</u>
<div><div>Wallet</div><div><div>Name</div><div>Not Given Yet</div><div>Amount</div><div>Not Given Yet</div><div>ADD</div><div>name or amount not given</div></div></div>	<div><div>Wallet</div><div><div>Name</div><div>Prakruti</div><div>Amount</div><div>150</div><div>ADD</div><div>added</div></div></div>

D)Add expense to self account

<u>Fail</u>	<u>Successful</u>
<div><div>Wallet</div><div>Amount</div><div><u>Not Given Yet</u></div><div>Category</div><div><div><input type="radio"/> Food</div><div><input type="radio"/> Health</div><div><input type="radio"/> Clothing</div><div><input type="radio"/> Travel</div><div><input checked="" type="radio"/> Others</div></div><div>Type</div><div><div><input type="radio"/> Credit</div><div><input checked="" type="radio"/> Debit</div></div><div>amount not given!</div><div>DONE</div></div>	<div><div>Wallet</div><div>Amount</div><div>150</div><div>Category</div><div><div><input type="radio"/> Food</div><div><input type="radio"/> Health</div><div><input checked="" type="radio"/> Clothing</div><div><input type="radio"/> Travel</div><div><input type="radio"/> Others</div></div><div>Type</div><div><div><input type="radio"/> Credit</div><div><input checked="" type="radio"/> Debit</div></div><div>Added</div><div>DONE</div></div>

9. CONCLUSION AND FUTURE SCOPE:

With the continued evolution in technology, portal is becoming more and more essential nowadays, with very less reliability on use of handmade records, the application tries to eliminate the regular diary keeping for the accounts, keeping for the trips you have been, it will keep a track of all your expenditure, notifies when you spend out of the budget limit which is set by the user himself and many more features as mentioned already. The app is user friendly and interactive, the features will be improved according to the feedback of our users. The project is valid and necessary for the future requirements of the analysis of your expenditure, most of the requirements are fulfilled upto the mark and the additional requirements will be completed within a short period of extension. Currently the system works for a limited number of administrators. In near future it will be extended for many users.