Client Requirements of Project NITC SPLIT BILL System

Prepared by

Group Number: 9

Amanpreet Kaur	M200680CA	(**Team Leader**)
Ankit Hammad	M200683CA	
Chetan Patidar	M200688CA	

Project Owner: Mr. KONU PRUTHVIRAJ

Course: CS4096D Software Engineering Laboratory Date: 12/01/2022

1. Goals

Hangoutwithfriendsandtrynewcuisineswithoutworryingaboutaccuratebillingor money transfers.

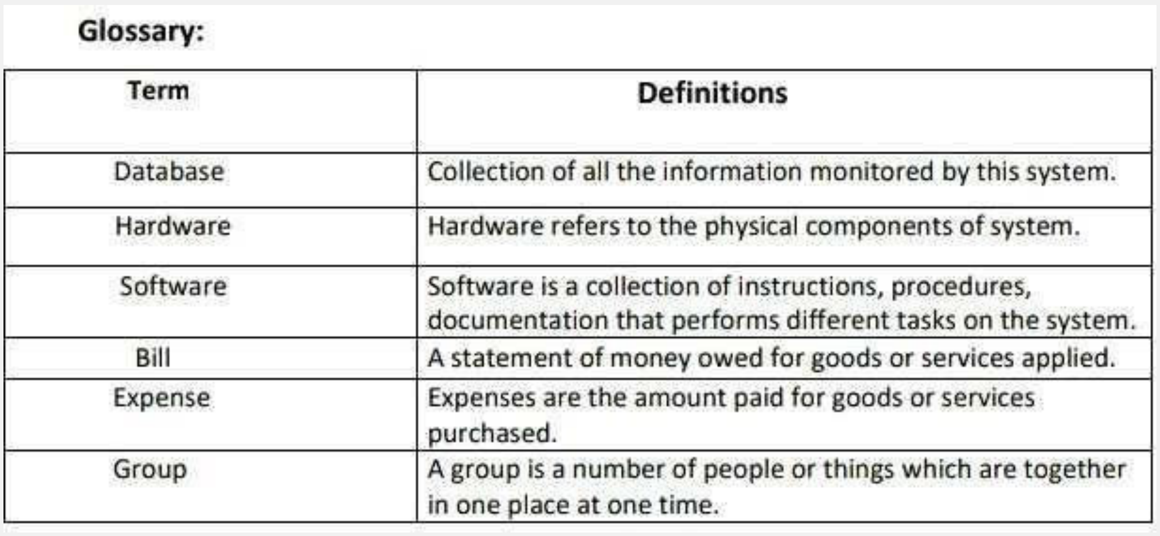
2. What's Happening!

it can be stressful and even embarrassing when you hang out with a group of people ,order something, then when the receipt comes everyone is perplexed because they are uncertain .

3. Domain Analysis Document

Purpose: It belongs to the “Bill Management” Domain. Actually, we love to try new cuisines every week However, it can be stressful and even embarrassing when you hang out with a group of people ,order something,thenwhenthereceiptcomeseveryoneisperplexedbecausetheyare uncertain how much every person should be paying. So the main motivation to develop this application is to divide the expenses individually.

Customers and users: This application is totally develop for helping NITC peoples such as Students, Faculties and other members to Split Bills whenever they travel or going somewhere in the group and want to split expenses among them fairly.



Glossary:

Term	Definitions
Database	Collection of all the information monitored by this system.
Hardware	Hardware refers to the physical components of system.
Software	Software is a collection of instructions, procedures, documentation that performs different tasks on the system.
Bill	A statement of money owed for goods or services applied.
Expense	Expenses are the amount paid for goods or services purchased.
Group	A group is a number of people or things which are together in one place at one time.

Competing Software: Split wise App., Split ,Trimount.

4. Problem Statement:

1. It will be an android app where NITC members or groups can divide their expenses among each other.

2. The application remembers the total expenses and there will be no miscalculations in expenses since it will keep track of all transactions.

3. This application helps us to add our expenses and split it among different people. The app keeps balances between people as in who owes how much to whom.

5. Motivations:

➢ Living in big cities like Indore and Delhi with tons of recreational visits.

➢ Food events.

➢ Easy payment options (phone wallet apps etc).

6. Frustrations

➢ Embarrassments and awkwardness that come with trying to split bills fairly.

➢ Payments options are plenty but non provides it all.

7. Stakeholder:

1. NITC members and Groups.

8. Design and Implementation Constraints

This product is an android based application, so we are using android studio here. Java programming language will be used here. For data management, we are using Firebase fire store DB. Software Development Life Cycle will be followed.

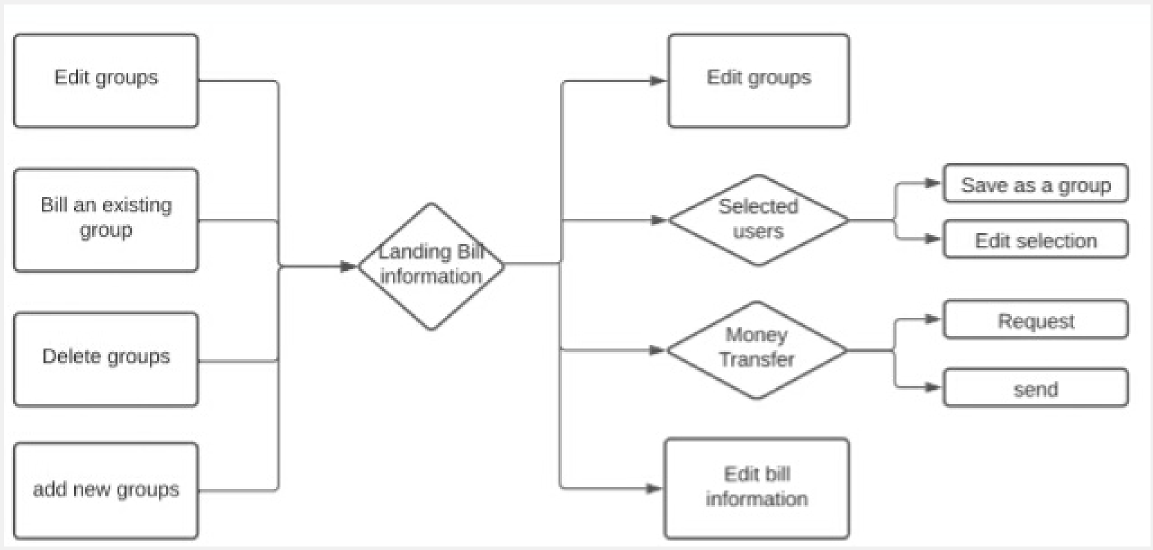
Assumptions and Dependencies Assumptions

- The Application is available 24x7.
- All users are part of NITC and have a NITC mail ID.
- All users are using a device with OS android 7.0 or above.

Dependencies

- For Using this application User must join Group.

9. Workflow Diagram



10. Functional Requirements:

F1: The system shall allow new users to register to the application using their NITC mail ID.

F2: The system shall allow user to Add new groups.

F3: the system shall allow user to edit groups.

F4: The System shall allow user to delete groups.

F5: The System shall allow user to Bill an existing group.

F6: The System shall allow user to Select friends to split.

F7: The System shall allow user to Request Money Transfer.

F8: The System shall allow user to Request Money Send.

F9: The System shall allow user to Edit bill Information.

F10: The System shall allow user to save Bill into temporary Card.

F11: The System should have acceptability to show expenses for a single user as well as balances for everyone.

F12: When asked to show balances, the System should show balances of a user with all the users where there is a non-zero balance.

11. Non-Functional Requirements:

Performance Requirements

- The System must be able to run concurrently on multiple android devices.
- Queries on the DB shall be performed faster (in less than 2 seconds).
- UIshouldbelightweightanddoesnottaketoomuchtimewhileRequestingandsendingMoney.
- The System must be capable of storing a large amount of data, including user's

details, amount and events.

- The System should give updates timely.

Safety and Security Requirements

- Every user must have to log in with NITC mail ID before Making any Group.
- itwillnothandleanycashtransactionsnoritdoesitlinktopersonalbankingdetails making it safer than other applications.

Software Quality Attributes:

Availability

The system shall be available to all the users 24X7.

. Correctness

All Calculations Should be done Correctly.

Response Time

The system shall be capable of executing the functions within a few seconds.

Recoverability

There is no recovery for this application. The machine can handle zero failure and optimum

output on its own.

Reliability

The system should be stable because failure will lead to loss all data or History. It should be ensured that the machine has not crashed more than 1-time in a week and the backup would not have to be completed.

Reusability

The application will be built using Android Studio, in which we will use OOPs using

JAVA, so that the same code can be reused to get a submission.

Platform

The application should be able to run seamlessly on version 7.0 and beyond android

platforms.

12. Optional Requirements:

- Away to add an expense name while adding the expense. Can also add notes, images, etc.

13. Hardware Interfaces:

No additional hardware interface is needed. A mobile device capable of running an android

application is sufficient.

14. Software Interfaces:

As it is a standalone app, no additional software interface is needed.

15. Similar Software:

1. Split wise App.

Difference:-

1. Adding Participants without Having to enter their Mobile number or email

2. The Activities list doesn't Show Dates.

Software Requirements Specification for

Team No.8: NITC Split bill System

Version <1.0>

Prepared by

Team Number:7

Isha Gupta	**M200694CA**
Aman Kumar Sankhwar	**M200679CA**
Ayushi Kumari	**M200687CA**
Amarjeet Budhsaini	**M200681CA**

Project Owner:	***Konu Pruthviraj***
Course:	**CS4096D Software Engineering Laboratory**
Date:	**<24-01-2022>**
	This template is based on the one available from the GMU site by Dr. Rob Pettit. Modifications specific to NITC are made and will be used for academic purposes only.

1 Introduction

Document Purpose

- The purpose of this Software Requirement Specification (SRS) document is to provide a detailed description of the functionalities of the NITC Split Bill System.
- This document will cover each of the system's intended features, as well as offer a preliminary glimpse of the software application's User Interface (UI). This document will also cover hardware, software and various other technical dependencies.

Project scope

We love to hang out with friends and try new cuisines. Sometimes we face some awkward moments like how much every person should be paying.
So, the problems we aim to alleviate with the help of this app are-:

- Difficulty of calculating divided costs.
- Unfair or imbalanced payments among individuals.
- Lack of accountability for money owed.
- Difficulty in remembering payments already made, pending debts etc.

The purpose of the application we will design will solve these problems and several more.

Functionalities of the app are –

- Automatic Calculation of money owed by each member of the group.
- Customizable formula for splitting costs.

Goal we want to achieve with this app—

The number of people evenly divides the entire bill, and each person pays an equal share, regardless of his order. This is usually done just to calculate how much money each person owes to settle the payment.

Benefits of using this app—

- Easy with money reimbursement.
- No headache of calculation.
- Features of a digital wallet.
- Transparent while splitting.

Intended Audience and Document Overview

This document is intended for all individuals participating in and/or supervising the NITC split bill system project. Readers interested in a brief overview of the application should focus on the rest of Part 1 (Introduction), as well as Part 2 of the document (Overall Description), which provide a brief overview of each aspect of the application as a whole.

Readers who wish to explore the features of the NITC split bill system in more details should read on to Part 3 (Specific Requirements), which expands upon the information laid out in the main overview. It offers further technical details, including information on the user interface as well as the hardware and software platforms on which the application will run.

Readers interested in the non-technical aspects of the project should read Part 4, which covers performance, safety, security and various other attributes that will be important to users.

Definitions, acronyms and abbreviations

Robustness -It is the ability of a computer system to cope with errors during execution and cope with erroneous input. It can encompass many areas of computer science, such as robust programming, robust machine learning, and Robust Security Network.

Correctness - Correctness from software engineering perspective can be defined as the adherence to the specifications that determine how users can interact with the software and how the software should behave when it is used correctly.

Availability - It is the probability that the system is operating properly when it is requested for use. In other words, availability is the probability that a system is not failed or undergoing a repair action when it needs to be used.

Usability- It is the degree to which software can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use.

Portability - Software portability is the possibility to use the same software in different environments. It applies to the software that is available for two or more different platforms or can be recompiled for them.

Abbreviations-

API – Application Programming Interface
UI – User Interface
HTTP – Hyper Text Transfer Protocol
PHP – Hypertext Preprocessor

Document Convention

Text size– 14
Heading size – 20
Sub heading size – 16
Font – Times New Roman

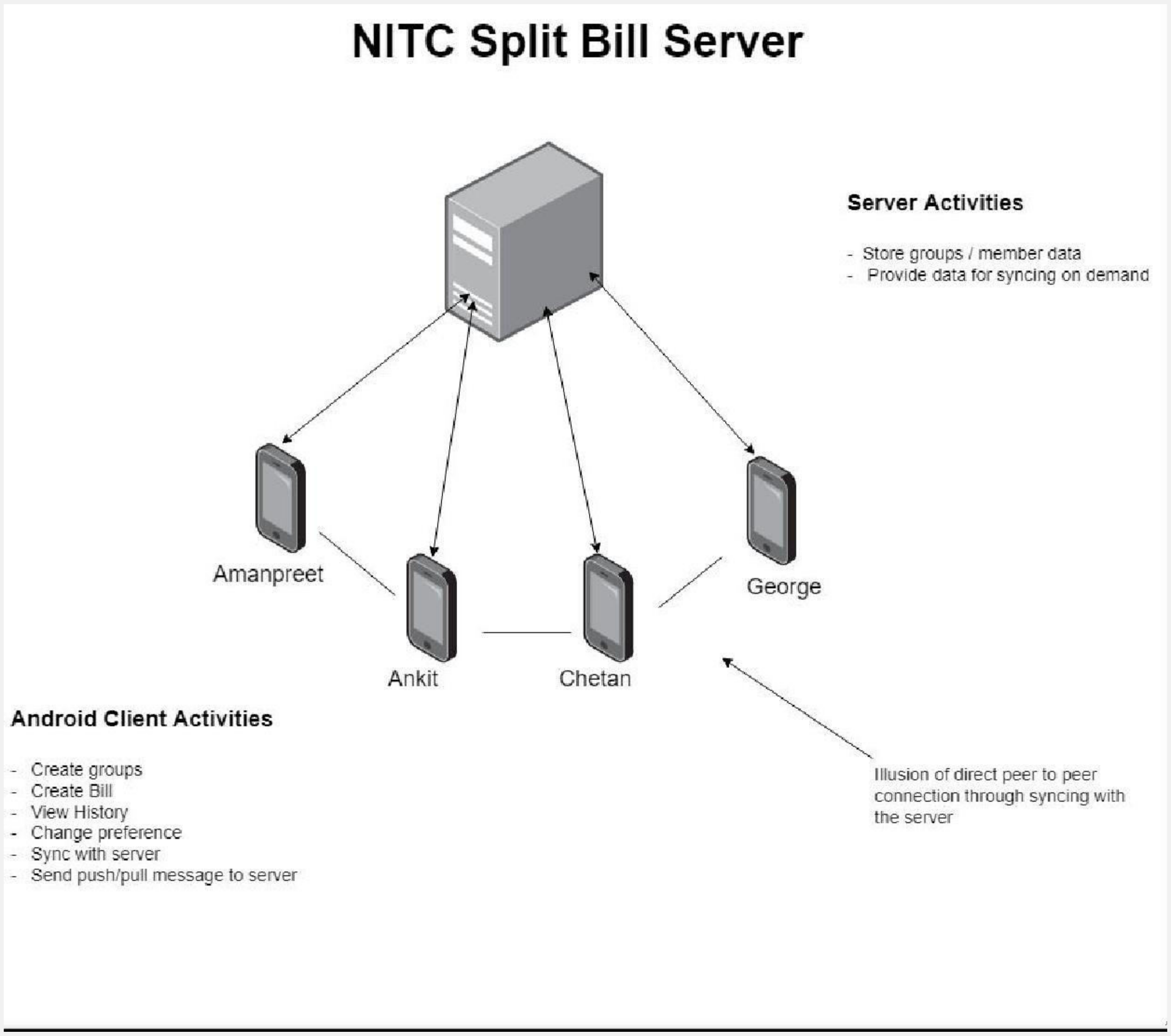
References and Acknowledgement

www.studocu.com

2. Overall Description

Product Overview

The Project NITC Split Bill System is a new, self-contained product intended for use on the Android platform. While the NITC Bill mobile application is the main focus of the project there is also a server-side component which will be responsible for database and synchronization services. The scope of the project encompasses both server and client-side functionalities, so both aspects are cover in detail within this document. Below is a diagram of the NITC Split Bill system which illustrate the interactions between the server and client applications.


NITC Split Bill Server
Server Activities
- Store groups / member data
- Provide data for syncing on demand
Amanpreet
Ankit
Chetan
George
Android Client Activities
- Create groups
- Create Bill
- View History
- Change preference
- Sync with server
- Send push/pull message to server
Illusion of direct peer to peer connection through syncing with the server

Product Functionality

The following list offers a brief outline and description of the main features and functionalities of the NITC Split Bill System. The features are split into two major categories:
core feature and additional feature.
core features are essential to the application’s operation, whereas additional features simply add new functionalities. The latter features will only be implemented as time permits.

Core Features

1. User Registration & Welcome

Only appears once (the first time the application run)
Allows the user to register with the NITC Split Bill System 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 contact list

3. Posting a Bill

Stores and monitors the bill amount, the individuals involved, etc.
Includes support for multiple simultaneous bills
Efficiently distributes debt amongst the individuals responsible for the bill

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 .

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 (group info etc.)
   Offers the option to resolve a debt (or debts) immediately

7. Setting Menu

   Allows the user to customize his/her preferences
   Enables the user to modify certain features and functionalities
   Can be accessed at any time using the built-in Settings button on Android phones

8. Help Menu

   Displays a list of topics covering the different component of NITC Split Bill
   Offers detailed information on each feature, menu, etc.
   Can be accessed at any time via the settings menu

9. Push Notifications

   Appear after any significant event occurs in a group
   Alert group members of newly incurred expenses
   Remind users of unresolved debts

Additional Features

10. Member Debt Visualization

    Presents a visual representation of current member balances
    Allows users to navigate through financial information in a more intuitive fashion

11. E-mail/SMS Notifications

    Extends the standard notifications service built into NITC Split Bill System
    Automatically delivers notifications via e-mail and/or text message
    Enables individuals without NITC Split Bill System to receive group notifications

12. NITC Split Bill Tutorial

    Provides an abridged version of the Help menu for first-time users
    Offers a step-by-step run through of each feature, menu, etc.
    Enables any user to quickly and easily take advantage of all of NITC Split Bill System's functionalities.

A major functionality present in several of these features is automatic synchronization. Using Android's internet capabilities, the application periodically communicates with the NITC Split Bill server. This allows bills, transactions, groups, and group histories to be uploaded to a central server where the data can be shared with all other Android users in the group. This process of exchanging data between the server and the phone(s) is referred to as syncing.

For more detailed information, see Part 3 of the document (System Feature).

User Classes and Characteristics

The NITC Split Bill project is meant to offer a shared expenses solution that is faster, easier, and more convenient that 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 NITC Split Bill is expected to be utilized:

NITC Split Bill: Potential Scenarios

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

   Key functions:
   - Keep track of expenses
   - Notify users when debts are incurred

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

   Key functions:
   - Add new expenses (quickly and easily)
   - Record for what he/she is paying for
   - Update member balances on the fly

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

These groups are not meant to separate or categories users, just the different situations in which NITC Split Bill 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 NITC Split Bill system: support for multiple groups. This functionality allows 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.

Design and Implementation Constraints

The primary design constraints are 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. NITC Split Bill is meant to be quick and responsive, even when dealing with large groups, so each feature must be designed and implemented with efficiency in mind.

Assumptions and Dependencies

Time Dependencies

As mentioned previously, the features of NITC Split Bill are divided into two groups: core features and additional features. Core features are crucial to the basic functionality of the NITC Split Bill application. These features must all be implemented in order for the application to be useful.

Optional features, however, are not critical to the function of the application. They are usability improvements and convenience, enhancements that may be added after the application has been developed. 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.

Hardware Dependencies

Some of the additional features rely on hardware components present in android handsets.

External Dependencies

Several of the features presented in this document rely on the existence and maintained operation of several APIs. A non-exhaustive list follows.

Email Notifications

The Android platform is not suited for sending mass emails. Thus, the central server will be responsible for this feature of the application. The Smartphone client will notify the server when message need to be sent using a custom API that is to be created. This API will use standard HTTP messaging to facilitate client-server communications. The API will be implemented using PHP.

SMS Notifications

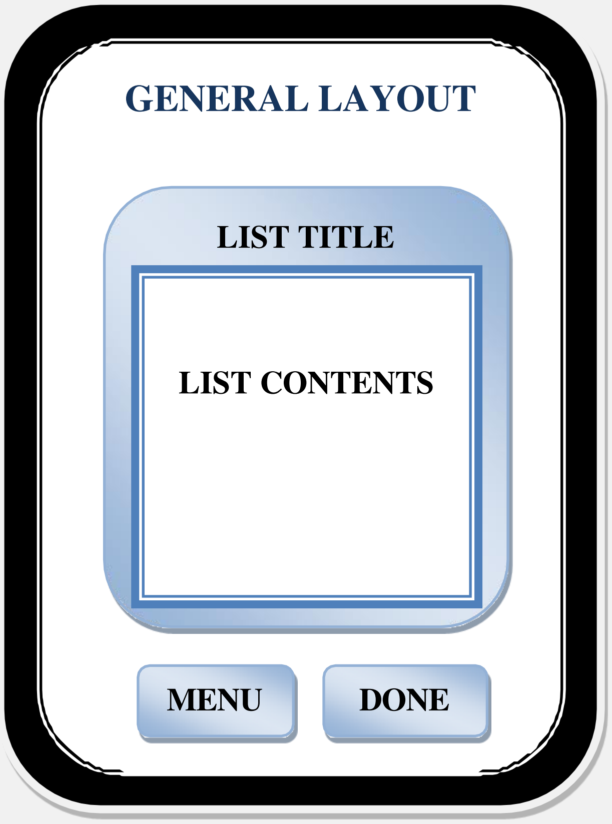
This feasibility of this feature is yet to be determined. If implemented, this feature would allow offline users without an Android Smartphone to receive notifications of outstanding debt and other information via text message. A suitable and free text messaging API that can be called from the server has yet to be found. The possibility of sending text messages from the Android Smartphone client itself is also being reviewed.

3 SPECIFIC REQUIREMENTS

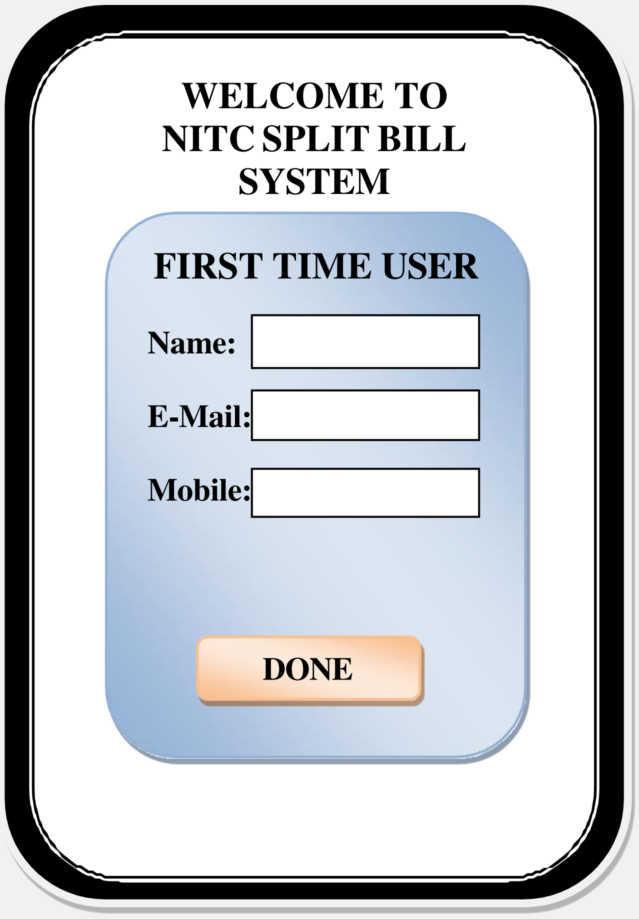
External Interface Requirements

User Interfaces

GENERAL LAYOUT



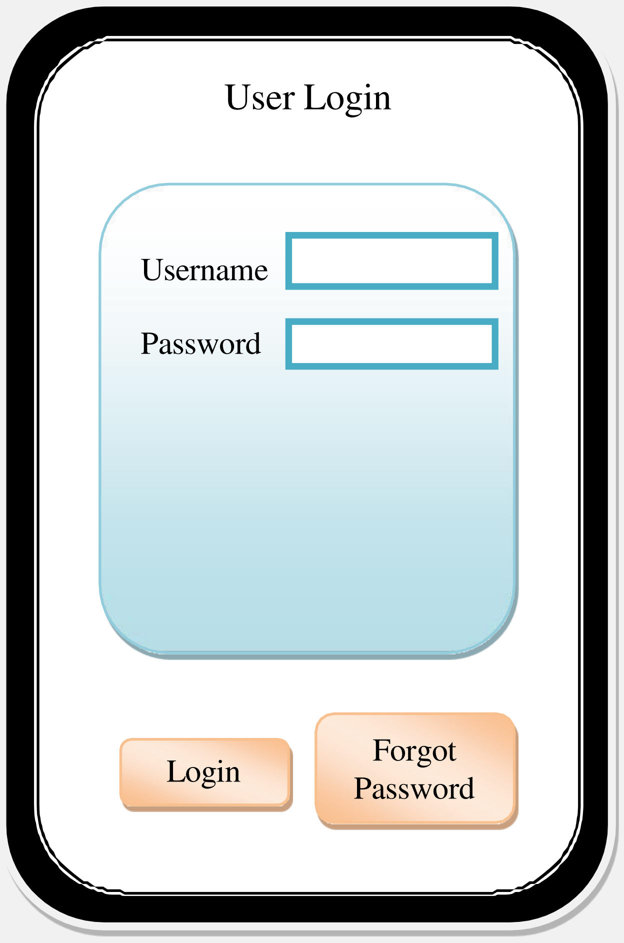
WELCOME SCREEN



For First Time User Only

- Here, User will enter account information (e.g. Name, E-mail, mobile number) to be stored in the server.
- Notifies the users if their information is invalid.

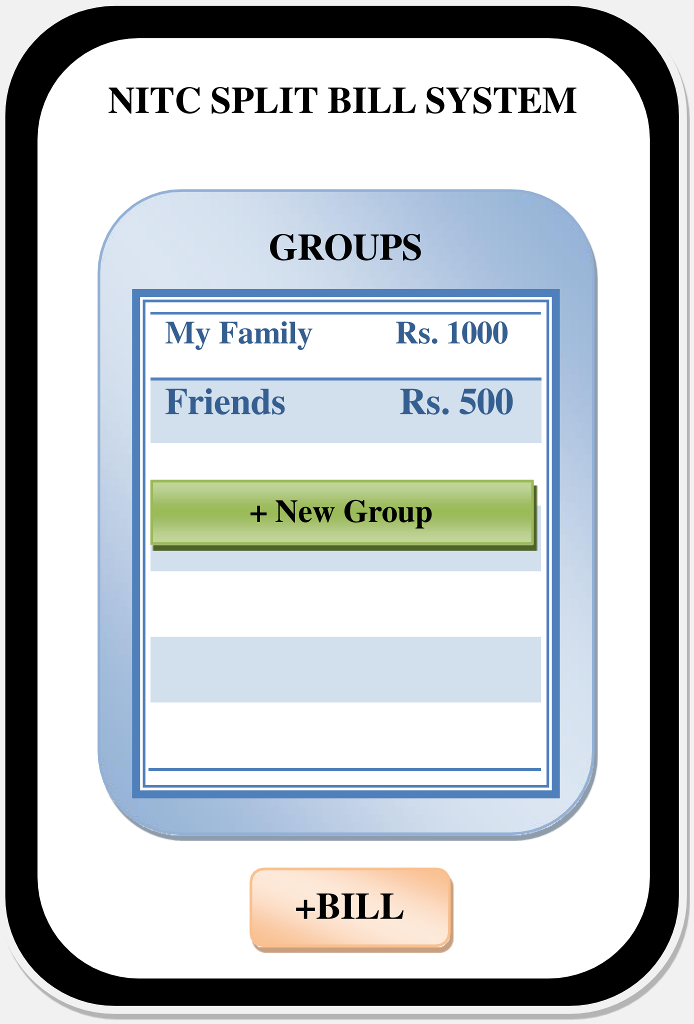
Login Page



For the registered user

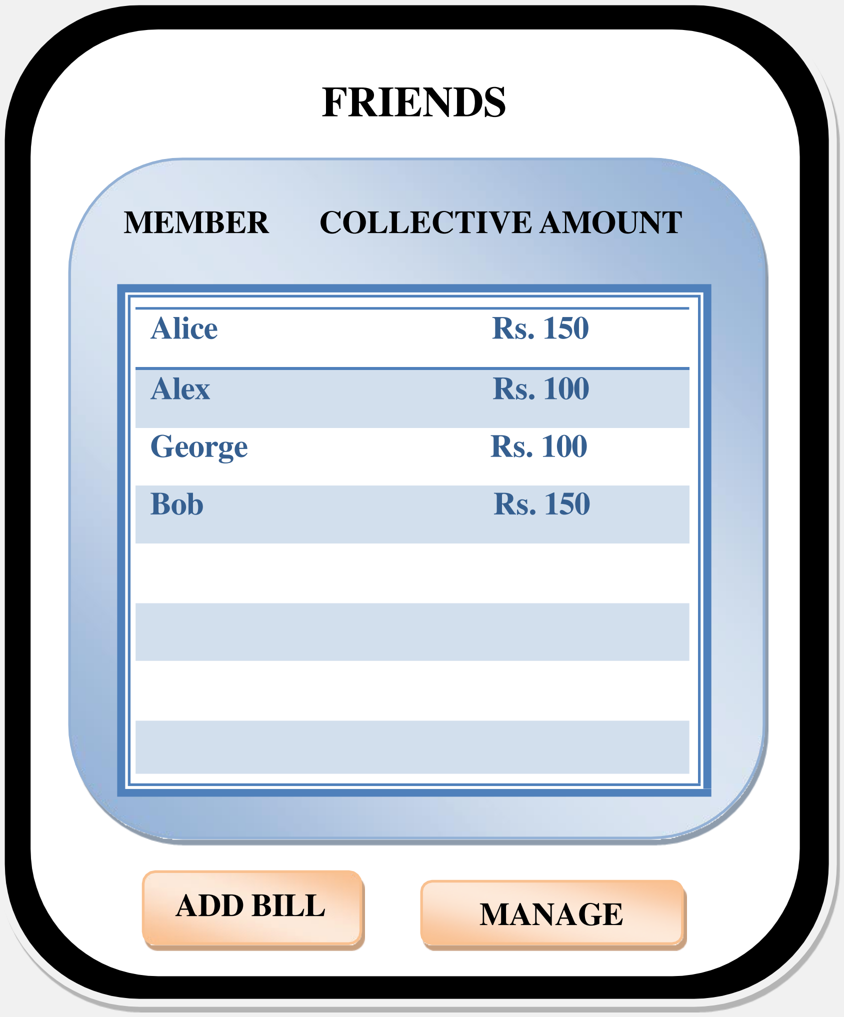
- After the registration completed, now the user is allowed to login to the app. By entering mail ID and password, the user can login to the app.
- If the user forgot his/her password then he/she must be directed to this use case in which he/she can reset his password and then he will be able to login to the app.

MAIN MENU



- Shows all groups, and how much the user owes the group collectively.
- User can select a group to open a view Group (in a separate menu).
- Create a Bill Option (Separate Menu).
- Create Group Option (Separate Menu).

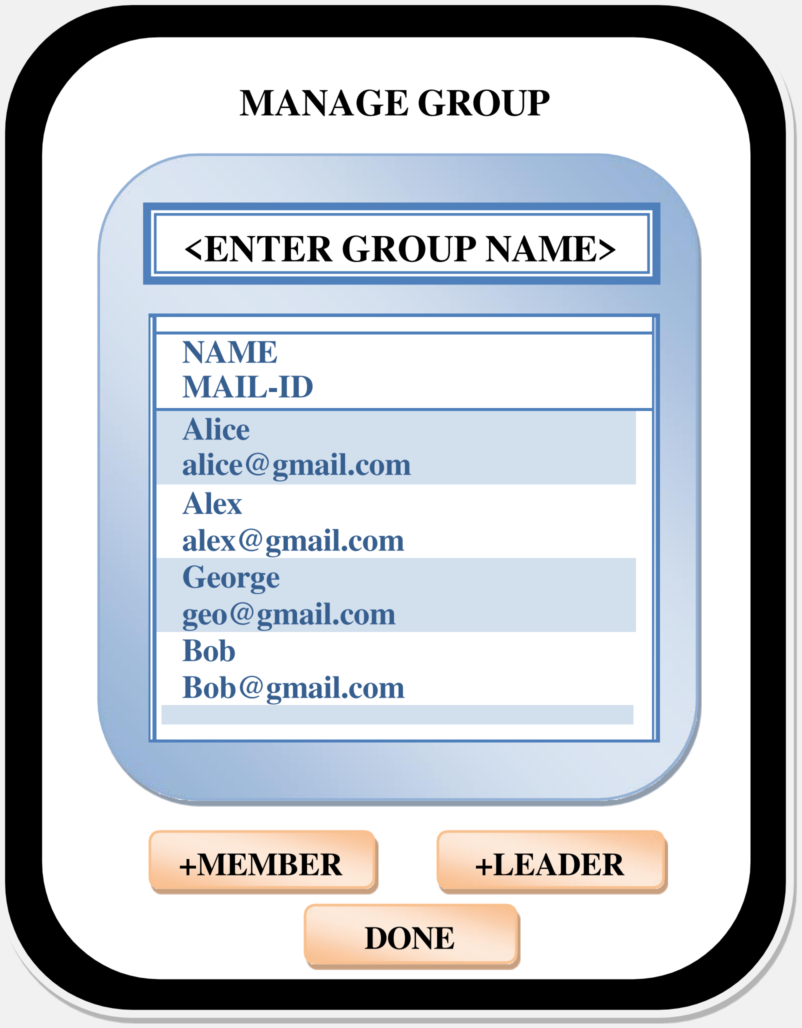
VIEW GROUP



MEMBER	COLLECTIVE AMOUNT
Alice	Rs. 150
Alex	Rs. 100
George	Rs. 100
Bob	Rs. 150

- Shows how much the users owe or is owed by each member of the group.
- Add Bill option (separate menu).
- User can make a "Member to Member" transaction by selecting a member and entering a payment amount (in a separate section).
- Transaction History of the group.
- Group Leader have additional options
  - Add / Remove members.
  - Designate a new leader.
  - Disband group.

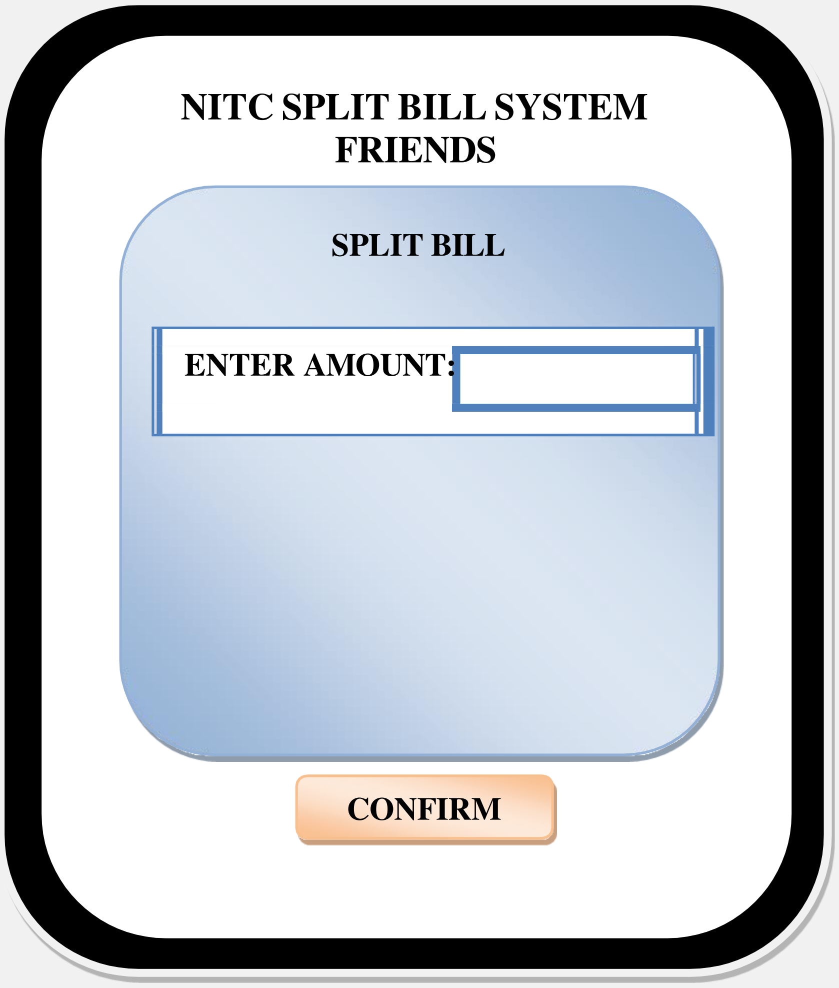
CREATE/ MANAGE GROUP



NAME MAIL-ID
Alice alice@gmail.com
Alex alex@gmail.com
George geo@gmail.com
Bob Bob@gmail.com

- User can add a group name.
- Allow the users to select members from the contacts or enter their information.
- The group creator will initially assume the role of “Group Leader”, however they will have the option to pick someone else in the menu.

CREATE A BILL



- User can enter the amount to pay the selected individual.

– Hardware Interfaces:

No additional hardware interface is needed. A mobile device capable of running an android application is sufficient.

– Software Interface:

As it is a standalone app, no additional software interface is needed.

FUNCTIONAL REQUIREMENTS

Functionalities of the app are

- Automatic Calculation of money owed by each member of the group.
- Customizable formula for splitting costs.
- Transaction history (including all payments made and costs incurred by each individual).

Some of the additional Features are

- Support for multiple transactions which can take place over extended period of time.
- Automatic notifications, alerting uses of pending debts and/or otherrelevant info.
- Transaction synchronization, keeping each group member up-to-date at all times.
- Receipt storage, which allows the user(s) to save photos of receipts associated with particular purchase/expenses.

Use Case Model

Use Case #1

User Registration
Author: Isha Gupta
Purpose: The Purpose of this use case is to allow new users to register into the application.
Priority: High
Pre conditions: The user should be student of NITC and have a valid college e-mail ID.
Post conditions: A pop-up message will be showed to the user that "You are Successfully Registered".
Actors: The student of NITC.
Flow of event: After downloading the application, the user has to register into the app by entering his/her name, contact number and college mail ID. After this, the user can create a password to login to the app. As the user enter details, all the details saved into the database.

Use Case #2.

User Login

Author: Ayushi Kumari
Purpose: This allow the user to login to the application, so that, he/she can be able to access the facilities of the app.
Priority: High
Pre conditions: The user must be a registered user.
Post conditions: The user interface page can be shown where he/she can make groups.
Actors: Users who have successfully registered.
Flow of event: After the registration completed, now the user is allowed to login to the app. By entering mail ID and password, the user can login to the app.

Use Case #3.

If User login fails

Author: Isha Gupta
Purpose: If user forgets his/her password then this facility will be used**.**
Priority: Medium
Pre conditions: The user must be a registered user.
Post conditions: The user must reset his/her password then the user will be able to login.
Actors: Users who have successfully registered.
Extend: This is an extended use case of Use Case #2
Flow of event: If the user forgot his/her password then he/she must be directed to this use case in which he/she can reset his password and then he will be able to login to the app.

Use Case #4.

<u>Make Groups</u>

Author: Aman Kumar Sankhwar
Purpose: This facility allows the user to make groups and choose the suitable event for which he wants to make groups.
Priority: Medium
Pre conditions: The user must be login to the application.
Post conditions: The user will select the members which he wants to add in the particular group.
Actors: Authenticated User.
Flow of event: After login to the app, now the user can make groups.

Use Case #5

<u>Add Members</u>

Author: Amarjeet Budhsaini
Purpose: This allows the user to add members in the group.
Priority: Medium
Pre conditions: The user must have created the group.
Post conditions: The user must be directed to the split bill page**.**
Actors: Leader of the group.

Flow of event: The user can add members in the group by entering the contact number of the members or can select the member from his contact list.

Use Case #6

<u>Edit Groups/Members</u>

Author: Isha Gupta
Purpose: This allows user to edit groups/ members. This means the user can add any member, remove any member, make anyone leader, remove group etc.
Priority: Medium
Pre conditions: The user first shall make a group and have members in the group.
Actors: Authenticated user.
Extend: This is an extended use case of Use Case #5.
Flow of event: The user should edit groups, members then he will be directed to the split bill interface.

Use Case #7

<u>Add and split Bill</u>

Author: Ayushi Kumari
Purpose: This allows the user to enter the bill amount which he/she wants to split.
Priority: High
Pre conditions: The user should make a group to split among the members of the groups.
Post conditions: The user will be directed to the interface in which individual amount will be shown**.**
Actors: Authenticated user
Flow of event: The user should enter the bill amount and as a result the amount for each individual person will be shown.

Use Case #8

History

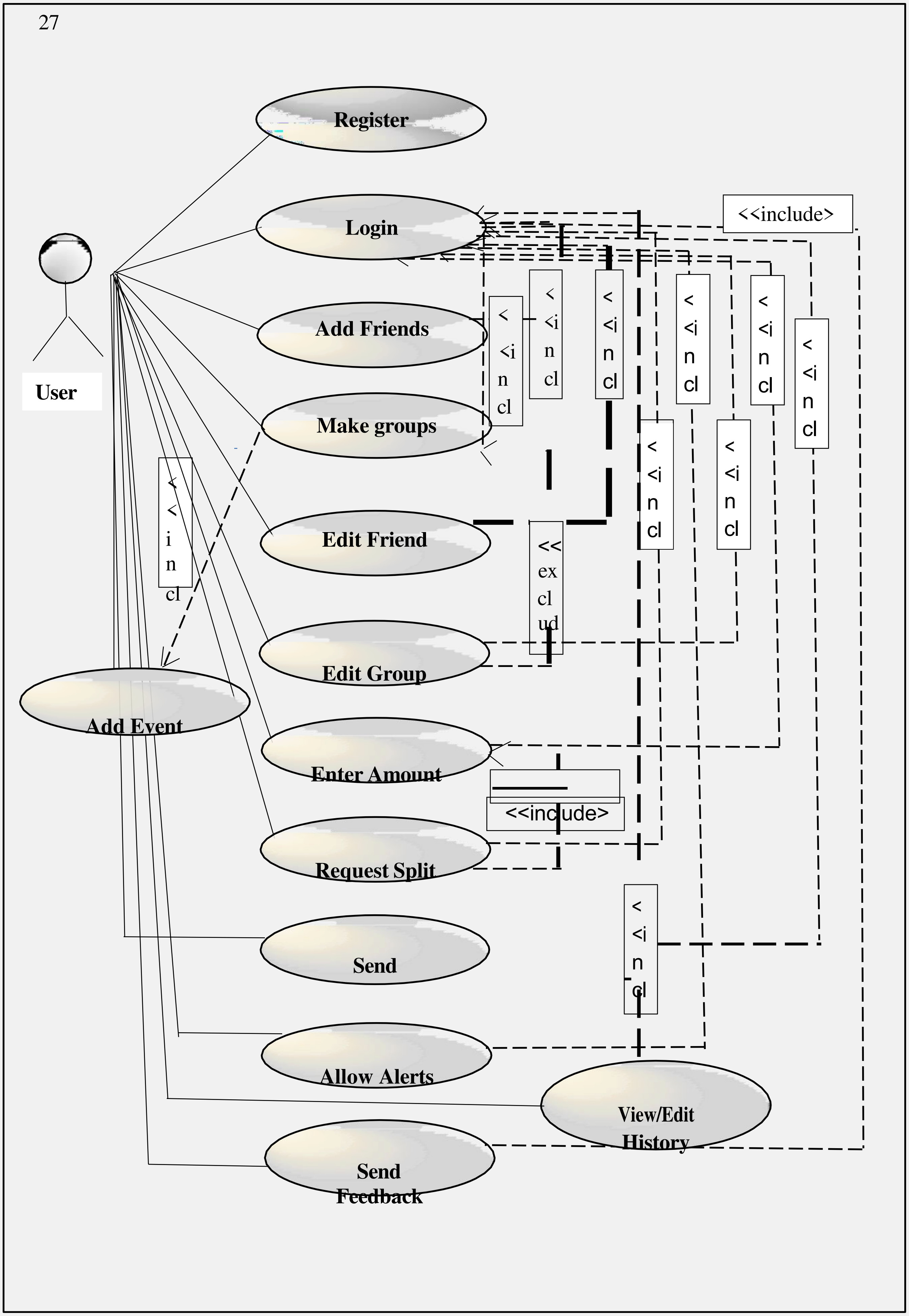
Author: Amarjeet Budhsaini
Purpose: This allows the user to check and download the history of the split money later.
Priority: Low
Pre conditions: The user should have the previous history of splitting the bill.
Post conditions: The data should be fetched from the database and will be shown to the user.
Actors: Authenticated user
Flow of event: The user should enter the date of which the user wants to see/download the history.

Use Case #9

Feedback

Author: Ayushi Kumari
Purpose: This allows the use to give feedback about the application.
Priority: Medium.
Pre conditions: If the user wants to convey his views about the performance of the app.
Post conditions: The feedback must be saved into the database.
Actors: Authenticated user
Flow of event: The user should send his feedback through the feedback option which must be saved into the database of the app.

Use Case Diagram of NITC Split Bill



4 Other Non-functional Requirements

Performance Requirements

- Performance should not be an issue because all of our server queries involve small pieces of data.
- Changing screens will require very little computation and thus will occur very quickly. Server updates should only take a few seconds as long as the phone can maintain a steady signal.

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

Safety and Security Requirements

- As the app is basically for the students of NITC, so every student who have a valid NITC email id is eligible to register himself within the application.
- Application neither demand any details of the user related his/her banking credentials nor does it affect any data stored outside of its server.
- The app only demands access to your contacts so that friends and groups can be added easily and thus making it safe as compare to other available options.

Software Quality Attributes

- **Robustness-*:***To assure robustness in a system we use exception handler in computing algorithm i.e., if any user by mistake enters any character other than digits then the alert will be generated with the message -entered value is invalid.

- **Correctness-*:***This is achieved by using appropriate algorithms and programming logic that splits the bill into the entered number of members.

- **Availability-*:***The app is always be able to operate properly when it is requested, it also take account the situations when user losses the internet connection the users were still be able to use the application.

- **Usability-*:*** The UI of the app is simple enough so the users should not face any difficulty in operating it and should be able the get the accurate results in seconds.

- **Portability-*:***The app is being ported solely for android platform.

5 Functional Point Estimation

FP Counting Process involves the following steps −

- **Step 1** − Determine the type of count.
- **Step 2** − Determine the boundary of the count.
- **Step 3** − Identify each Elementary Process (EP) required by the user.
- **Step 4** − Determine the unique EPs.
- **Step 5** − Measure data functions.
- **Step 6** − Measure transactional functions.
- **Step 7** − Calculate functional size (unadjusted function point count).
- **Step 8** − Determine Value Adjustment Factor (VAF).
- **Step 9** − Calculate adjusted function point count.

Note − General System Characteristics (GSCs) are made optional in CPM 4.3.1 and moved to Appendix. Hence, Step 8 and Step 9 can be skipped.

Step 1: Determine the Type of Count

There are three types of function point counts −

- Development Function Point Count
- Application Function Point Count
- Enhancement Function Point Count

Development Function Point Count

Function points can be counted at all phases of a development project from requirement to implementation stage. This type of count is associated with new development work and may include the prototypes, which may have been required as temporary solution, which supports the conversion effort. This type of count is called a baseline function point count.

Application Function Point Count

Application counts are calculated as the function points delivered, and exclude any conversion effort (prototypes or temporary solutions) and existing functionality that may have existed.

Enhancement Function Point Count

When changes are made to software after production, they are considered as enhancements. To size such enhancement projects, the Function Point Count gets Added, Changed or Deleted in the Application.

Step 2: Determine the Boundary of the Count

The boundary indicates the border between the application being measured and the external applications or the user domain. (Refer Figure 1)

To determine the boundary, understand −

- The purpose of the function point count
- Scope of the application being measured
- How and which applications maintain what data
- The business areas that support the applications

Step 3: Identify Each Elementary Process Required by the User

Compose and/or decompose the functional user requirements into the smallest unit of activity, which satisfies all of the following criteria −

- Is meaningful to the user.
- Constitutes a complete transaction.
- Is self-contained.
- Leaves the business of the application being counted in a consistent state.

For example, the Functional User Requirement − "Maintain Employee information" can be decomposed into smaller activities such as add employee, change employee, delete employee, and inquire about employee.

Each unit of activity thus identified is an Elementary Process (EP).

Step 4: Determine the Unique Elementary Processes

Comparing two EPs already identified, count them as one EP (same EP) if they −

- Require the same set of DETs.
- Require the same set of FTRs.
- Require the same set of processing logic to complete the EP.

Do not split an EP with multiple forms of processing logic into multiple Eps.

For e.g., if you have identified 'Add Employee' as an EP, it should not be divided into two EPs to account for the fact that an employee may or may not have dependents. The EP is still 'Add Employee', and there is variation in the processing logic and DETs to account for dependents.

Step 5: Measure Data Functions

Classify each data function as either an ILF or an EIF.

A data function shall be classified as an −

- Internal Logical File (ILF), if it is maintained by the application being measured.
- External Interface File (EIF) if it is referenced, but not maintained by the application being measured.

ILFs and EIFs can contain business data, control data and rules based data. For example, telephone switching is made of all three types - business data, rule data and control data. Business data is the actual call. Rule data is how the call should be routed through the network, and control data is how the switches communicate with each other.

Consider the following documentation for counting ILFs and EIFs −

- Objectives and constraints for the proposed system.
- Documentation regarding the current system, if such a system exists.
- Documentation of the users' perceived objectives, problems and needs.
- Data models.

Step 5.1: Count the DETs for Each Data Function

Apply the following rules to count DETs for ILF/EIF −

- Count a DET for each unique user identifiable, non-repeated field maintained in or retrieved from the ILF or EIF through the execution of an EP.
- Count only those DETs being used by the application that is measured when two or more applications maintain and/or reference the same data function.
- Count a DET for each attribute required by the user to establish a relationship with another ILF or EIF.
- Review related attributes to determine if they are grouped and counted as a single DET or whether they are counted as multiple DETs. Grouping will depend on how the EPs use the attributes within the application.

Step 5.2: Count the RETs for Each Data Function

Apply the following rules to count RETs for ILF/EIF −

- Count one RET for each data function.
- Count one additional RET for each of the following additional logical sub-groups of DETs.
  - Associative entity with non-key attributes.
  - Sub-type (other than the first sub-type).
  - Attributive entity, in a relationship other than mandatory 1:1.

Step 5.3: Determine the Functional Complexity for Each Data Function

RETS	**Data Element Types (DETs)**		
	1-19	**20-50**	**>50**
1	L	L	A
2 to 5	L	A	H
>5	A	H	H

Functional Complexity: **L** = Low; **A** = Average; **H** = High

Step 5.4: Measure the Functional Size for Each Data Function

Functional Complexity	FP Count for ILF	FP Count for EIF
Low	7	5
Average	10	7
High	15	10

Step 6: Measure Transactional Functions

To measure transactional functions following are the necessary steps −

Step 6.1: Classify each Transactional Function

Transactional functions should be classified as an External Input, External Output or an External Inquiry.

External Input

External Input (EI) is an Elementary Process that processes data or control information that comes from outside the boundary. The primary intent of an EI is to maintain one or more ILFs and/or to alter the behavior of the system.

All of the following rules must be applied −

- The data or control information is received from outside the application boundary.
- At least one ILF is maintained if the data entering the boundary is not control information that alters the behavior of the system.
- For the identified EP, one of the three statements must apply −
    - Processing logic is unique from processing logic performed by other EIs for the application.
    - The set of data elements identified is different from the sets identified for other EIs in the application.
    - ILFs or EIFs referenced are different from the files referenced by the other EIs in the application.

External Output

External Output (EO) is an Elementary Process that sends data or control information outside the application’s boundary. EO includes additional processing beyond that of an external inquiry.

The primary intent of an EO is to present information to a user through processing logic other than or in addition to the retrieval of data or control information.

The processing logic must −

- Contain at least one mathematical formula or calculation.
- Create derived data.
- Maintain one or more ILFs.
- Alter the behavior of the system.

All of the following rules must be applied −

- Sends data or control information external to the application’s boundary.
- For the identified EP, one of the three statements must apply −
    - Processing logic is unique from the processing logic performed by other EOs for the application.
    - The set of data elements identified are different from other EOs in the application.
    - ILFs or EIFs referenced are different from files referenced by other EOs in the application.

Additionally, one of the following rules must apply −

- The processing logic contains at least one mathematical formula or calculation.
- The processing logic maintains at least one ILF.
- The processing logic alters the behavior of the system.

External Inquiry

External Inquiry (EQ) is an Elementary Process that sends data or control information outside the boundary. The primary intent of an EQ is to present information to the user through the retrieval of data or control information.

The processing logic contains no mathematical formula or calculations, and creates no derived data. No ILF is maintained during the processing, nor is the behavior of the system altered.

All of the following rules must be applied −

- Sends data or control information external to the application’s boundary.
- For the identified EP, one of the three statements must apply −

  - Processing logic is unique from the processing logic performed by other EQs for the application.
  - The set of data elements identified are different from other EQs in the application.
  - The ILFs or EIFs referenced are different from the files referenced by other EQs in the application.

Additionally, all of the following rules must apply −

- The processing logic retrieves data or control information from an ILF or EIF.
- The processing logic does not contain mathematical formula or calculation.
- The processing logic does not alter the behavior of the system.
- The processing logic does not maintain an ILF.

Step 6.2: Count the DETs for Each Transactional Function

Apply the following Rules to count DETs for EIs −

- Review everything that crosses (enters and/or exits) the boundary.
- Count one DET for each unique user identifiable, non-repeated attribute that crosses (enters and/or exits) the boundary during the processing of the transactional function.
- Count only one DET per transactional function for the ability to send an application response message, even if there are multiple messages.
- Count only one DET per transactional function for the ability to initiate action(s) even if there are multiple means to do so.
- Do not count the following items as DETs −
  - Attributes generated within the boundary by a transactional function and saved to an ILF without exiting the boundary.
  - Literals such as report titles, screen or panel identifiers, column headings and attribute titles.
  - Application generated stamps such as date and time attributes.
  - Paging variables, page numbers and positioning information, for e.g., 'Rows 37 to 54 of 211'.
  - Navigation aids such as the ability to navigate within a list using "previous", "next", "first", "last" and their graphical equivalents.

Apply the following rules to count DETs for EOs/EQs −

- Review everything that crosses (enters and/or exits) the boundary.
- Count one DET for each unique user identifiable, non-repeated attribute that crosses (enters and/or exits) the boundary during the processing of the transactional function.

- Count only one DET per transactional function for the ability to send an application response message, even if there are multiple messages.
- Count only one DET per transactional function for the ability to initiate action(s) even if there are multiple means to do so.
- Do not count the following items as DETs −
    - Attributes generated within the boundary without crossing the boundary.
    - Literals such as report titles, screen or panel identifiers, column headings and attribute titles.
    - Application generated stamps such as date and time attributes.
    - Paging variables, page numbers and positioning information, for e.g., ‘Rows 37 to 54 of 211’.
    - Navigation aids such as the ability to navigate within a list using “previous”, “next”, “first”, “last” and their graphical equivalents.

Step 6.3: Count the FTRs for Each Transactional Function

Apply the following rules to count FTRs for EIs −

- Count a FTR for each ILF maintained.
- Count a FTR for each ILF or EIF read during the processing of the EI.
- Count only one FTR for each ILF that is both maintained and read.

Apply the following rule to count FTRs for EO/EQs −

- Count a FTR for each ILF or EIF read during the processing of EP.

Additionally, apply the following rules to count FTRs for EOs −

- Count a FTR for each ILF maintained during the processing of EP.
- Count only one FTR for each ILF that is both maintained and read by EP.

Step 6.4: Determine the Functional Complexity for Each Transactional Function

FTRs	**Data Element Types (DETs)**		
	1-4	**5-15**	**>=16**
0-1	L	L	A

2	L	A	H
>=3	A	H	H

Functional Complexity: **L** = Low; **A** = Average; **H** = High

Determine the functional complexity for each EO/EQ, with an exception that EQ must have a minimum of 1 FTR −

EQ must have a minimum of 1 FTR	**Data Element Types (DETs)**		
FTRs	1-4	5-15	>=16
0-1	L	L	A
2	L	A	H
>=3	A	H	H

Functional Complexity: **L** = Low; **A** = Average; **H** = High

Step 6.5: Measure the Functional Size for Each Transactional Function

Measure the functional size for each EI from its functional complexity.

Complexity	**FP Count**
Low	3
Average	4
High	6

Measure the functional size for each EO/EQ from its functional complexity.

Complexity	**FP Count for EO**	**FP Count for EQ**

Low	4	3
Average	5	4
High	6	6

Step 7: Calculate Functional Size (Unadjusted Function Point Count)

To calculate the functional size, one should follow the steps given below −

Step 7.1

Recollect what you have found in Step 1. Determine the type of count.

Step 7.2

Calculate the functional size or function point count based on the type.

- For development function point count, go to Step 7.3.
- For application function point count, go to Step 7.4.
- For enhancement function point count, go to Step 7.5.

Step 7.3

Development Function Point Count consists of two components of functionality −

- Application functionality included in the user requirements for the project.
- Conversion functionality included in the user requirements for the project. Conversion functionality consists of functions provided only at installation to convert data and/or provide other user-specified conversion requirements, such as special conversion reports. For e.g. an existing application may be replaced with a new system.

DFP = ADD + CFP

Where,

DFP = Development Function Point Count

ADD = Size of functions delivered to the user by the development project

CFP = Size of the conversion functionality

ADD = FP Count (ILFs) + FP Count (EIFs) + FP Count (EIs) + FP Count (EOs) + FP Count (EQs)

CFP = FP Count (ILFs) + FP Count (EIFs) + FP Count (EIs) + FP Count (EOs) + FP Count (EQs)

Step 7.4

Calculate the Application Function Point Count

AFP = ADD

Where,

AFP = Application Function Point Count

ADD = Size of functions delivered to the user by the development project (excluding the size of any conversion functionality), or the functionality that exists whenever the application is counted.

ADD = FP Count (ILFs) + FP Count (EIFs) + FP Count (EIs) + FP Count (EOs) + FP Count (EQs)

Step 7.5

Enhancement Function Point Count considers the following four components of functionality −

- Functionality that is added to the application.
- Functionality that is modified in the Application.
- Conversion functionality.
- Functionality that is deleted from the application.

EFP = ADD + CHGA + CFP + DEL

Where,

EFP = Enhancement Function Point Count

ADD = Size of functions being added by the enhancement project

CHGA = Size of functions being changed by the enhancement project

CFP = Size of the conversion functionality

DEL = Size of functions being deleted by the enhancement project

ADD = FP Count (ILFs) + FP Count (EIFs) + FP Count (EIs) + FP Count (EOs) + FP Count (EQs)

CHGA = FP Count (ILFs) + FP Count (EIFs) + FP Count (EIs) + FP Count (EOs) + FP Count (EQs)

CFP = FP Count (ILFs) + FP Count (EIFs) + FP Count (EIs) + FP Count (EOs) + FP Count (EQs)

DEL = FP Count (ILFs) + FP Count (EIFs) + FP COUNT (EIs) + FP Count (EOs) + FP Count (EQs)

Step 8: Determine the Value Adjustment Factor

GSCs are made optional in CPM 4.3.1 and moved to Appendix. Hence, Step 8 and Step 9 can be skipped.

The Value Adjustment Factor (VAF) is based on 14 GSCs that rate the general functionality of the application being counted. GSCs are user business constraints independent of technology. Each characteristic has associated descriptions to determine the degree of influence.

General System Characteristic	**Brief Description**
Data Communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
Distributed Data Processing	How are distributed data and processing functions handled?
Performance	Did the user require response time or throughput?
Heavily Used Configuration	How heavily used is the current hardware platform where the application will be executed?
Transaction Rate	How frequently are transactions executed daily, weekly, monthly, etc.?
On-Line Data Entry	What percentage of the information is entered online?
End-user Efficiency	Was the application designed for end-user efficiency?

Online Update	How many ILFs are updated by online transaction?
Complex Processing	Does the application have extensive logical or mathematical processing?
Reusability	Was the application developed to meet one or many user's needs?
Installation Ease	How difficult is conversion and installation?
Operational Ease	How effective and/or automated are start-up, back-up, and recovery procedures?
Multiple Sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
Facilitate Change	Was the application specifically designed, developed, and supported to facilitate change?

The degree of influence range is on a scale of zero to five, from no influence to strong influence.

Rating	**Degree of Influence**
0	Not present, or no influence
1	Incidental influence
2	Moderate influence

3	Average influence
4	Significant influence
5	Strong influence throughout

Determine the degree of influence for each of the 14 GSCs.

The sum of the values of the 14 GSCs thus obtained is termed as Total Degree of Influence (TDI).

$$\textbf{TDI} = \sum^{14} \textbf{Degrees of Influence}$$

Next, calculate Value Adjustment Factor (VAF) as

$$\textbf{VAF} = (\textbf{TDI} \times 0.01) + 0.65$$

Each GSC can vary from 0 to 5, TDI can vary from (0×14) to (5×14), i.e. 0 (when all GSCs are low) to 70 (when all GSCs are high) i.e. $0 \leq \text{TDI} \leq 70$. Hence, VAF can vary in the range from 0.65 (when all GSCs are low) to 1.35 (when all GSCs are high), i.e., $0.65 \leq \text{VAF} \leq 1.35$.

Step 9: Calculate Adjusted Function Point Count

As per the FPA approach that uses the VAF (CPM versions before V4.3.1), this is determined by,

Adjusted FP Count = Unadjusted FP Count × VAF

Where, unadjusted FP count is the functional size that you have calculated in Step 7.

As the VAF can vary from 0.65 to 1.35, the VAF exerts an influence of ±35% on the final adjusted FP count.

Benefits of Function Points

Function points are useful −

- In measuring the size of the solution instead of the size of the problem.
- As requirements are the only thing needed for function points count.
- As it is independent of technology.
- As it is independent of programming languages.
- In estimating testing projects.
- In estimating overall project costs, schedule and effort.

- In contract negotiations as it provides a method of easier communication with business groups.
- As it quantifies and assigns a value to the actual uses, interfaces, and purposes of the functions in the software.
- In creating ratios with other metrics such as hours, cost, headcount, duration, and other application metrics.

FP Repositories

International Software Benchmarking Standards Group (ISBSG) grows and maintains two repositories for IT data.

- Development and Enhancement Projects
- Maintenance and Support Applications

There are more than 6,000 projects in the Development and Enhancement Projects repository.

Data is delivered in Microsoft Excel format, making it easier for further analysis that you wish to do with it, or you can even use the data for some other purpose.

6 Appendix A – Activity Log

Our team has met three times to complete this document for approximate 45 minutes to show his /her work progress in the assigned task.

Introduction part was done by – Ayushi Kumari
Overall Description was done by – Aman Kumar Sankhwar
External Interface Requirements was done by – Amarjeet Budhsaini
Non-Specific requirements was done by – Isha Gupta
Use Case Analysis was done by – All Members

Design Document

for

NITC Split Bill System

Version 1.1

Prepared by Team 7:
(Based on SRS Version 1 prepared by Team 7)

Isha Gupta	**M200694CA**	**isha_m200694ca@nitc.ac.in**
Ayushi Kumari	**M200687CA**	**ayushi_m200687ca@nitc.ac.in**
Aman Kumar Sankhwar	**M200679CA**	**aman_m200679ca@nitc.ac.in**
Amarjeet Budhsaini	**M200681CA**	**amarjeet_m200681ca@nitc.ac.in**

Project Owner:	**Konu Pruthviraj**
Course:	**CS4096 Software Engineering Laboratory**
Date:	**21-02-2022**

Glossary

API	Application Programming Language
UI	User Interface
HTTP	Hyper Text Transfer Protocol
PHP	Hypertext Protocol
ER diagram	Entity Relationship diagram
OOP's	Object Oriented Programming

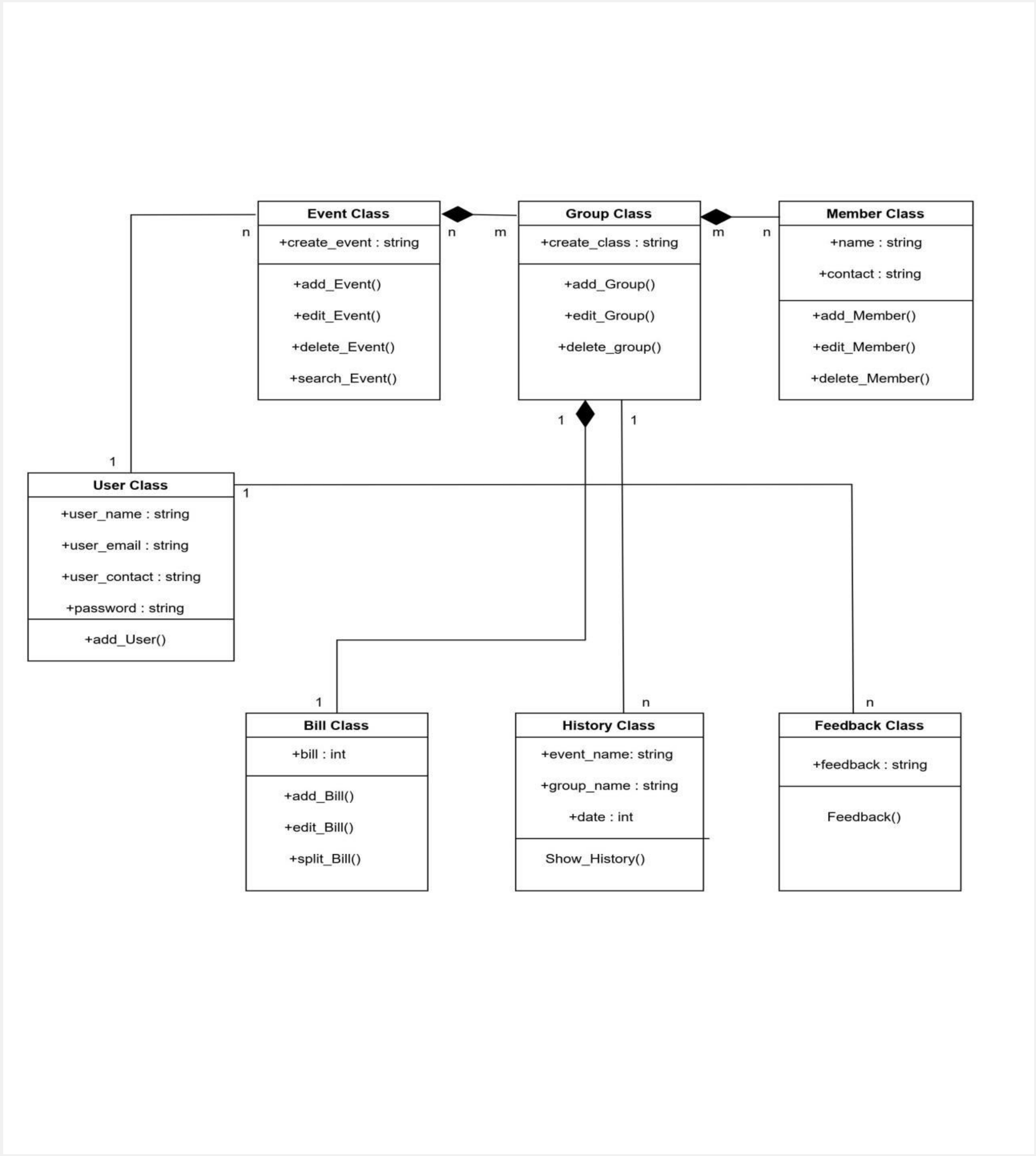
Table of contents

1. Detailed Design through UML diagrams

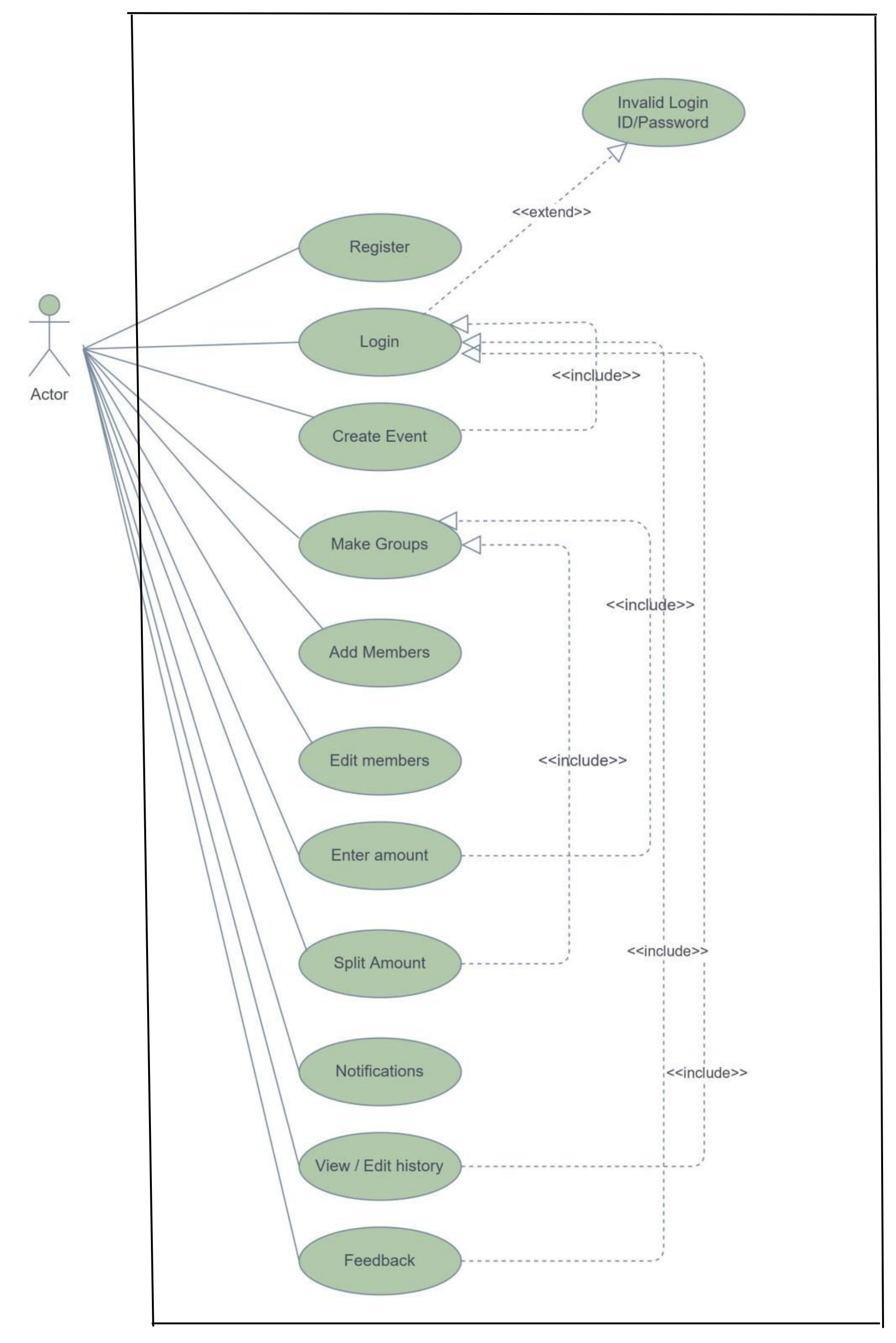
System model using Class Diagram

Class Diagram in the Unified Modelling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods) and the relationships among classes.

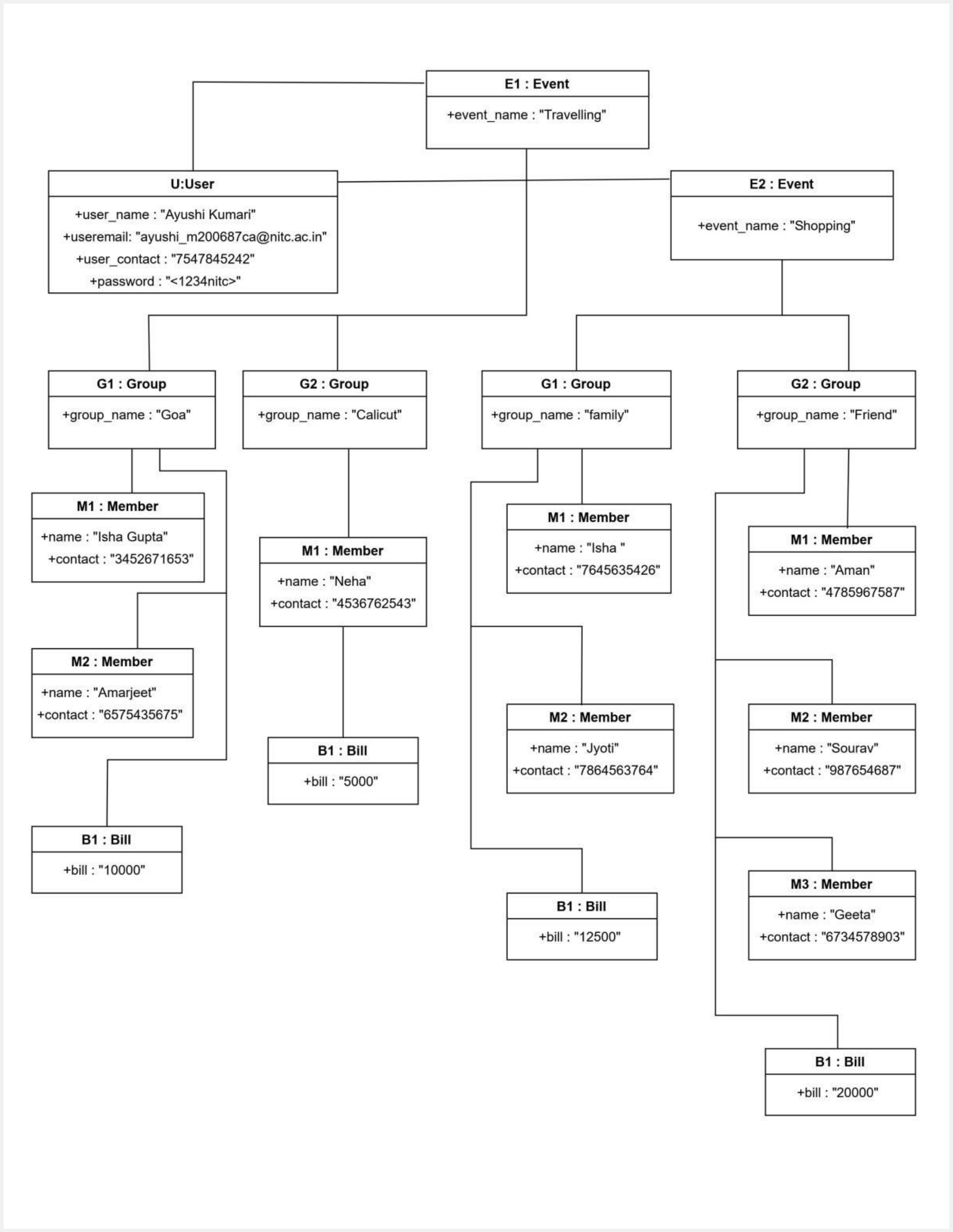
Class Diagram



Responsibilities – Use case Diagram



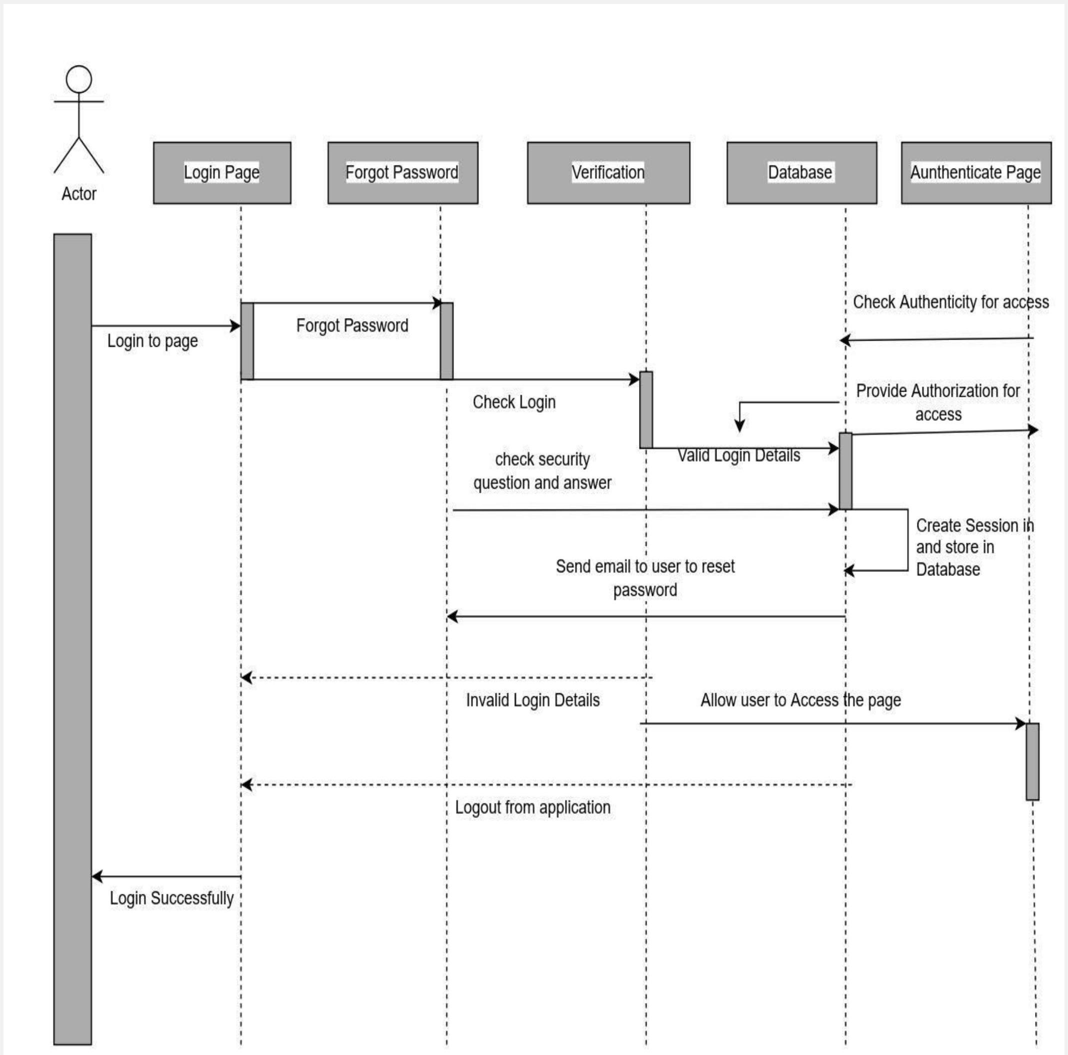
Static snapshot of the system - Object Diagram



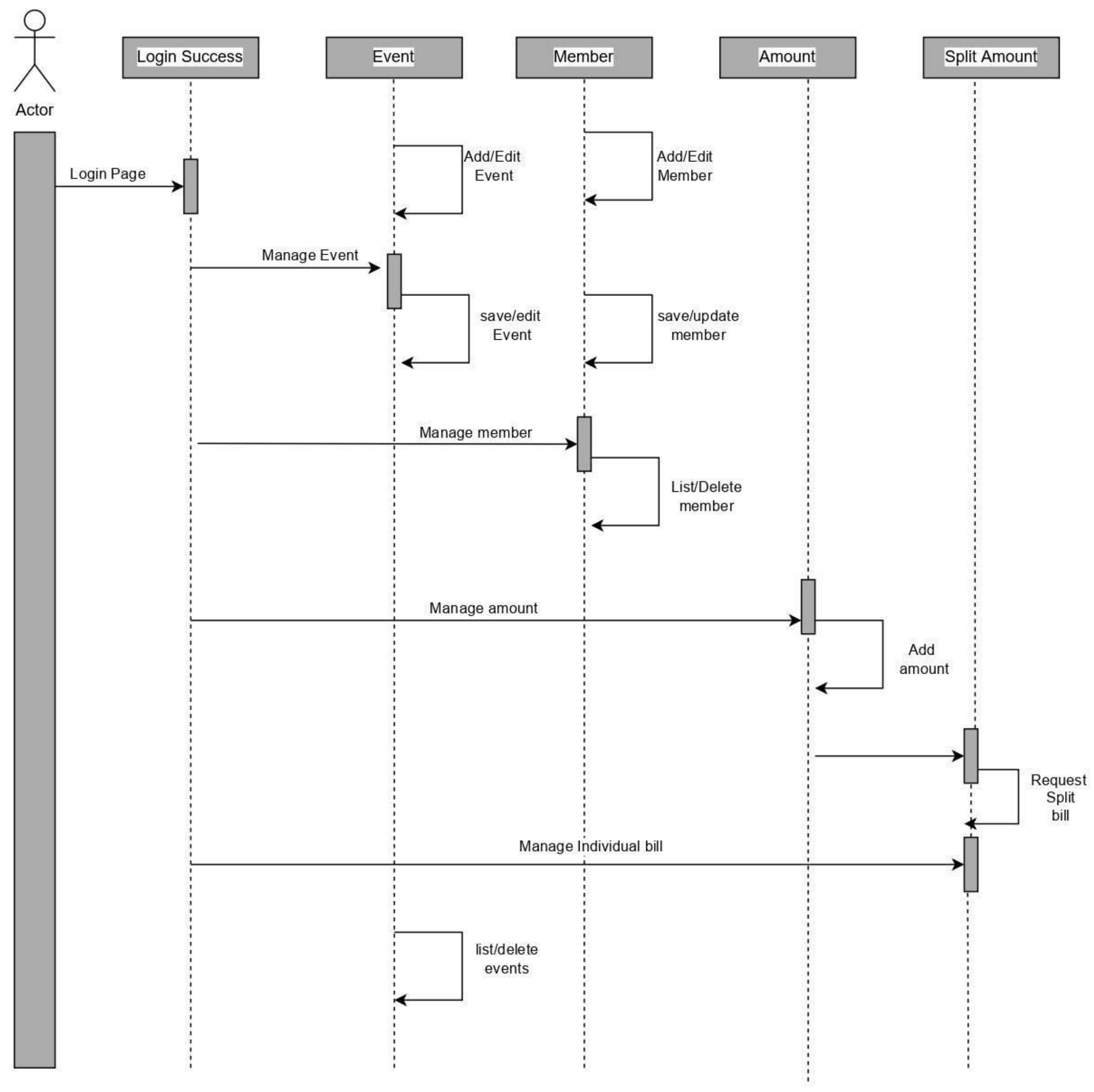
System Interactions through Sequence Diagrams

Sequence diagrams are interaction diagrams that show the sequence of messages exchanged by the set of objects performing a certain task. A sequence diagram shows, as parallel vertical lines (lifeline), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur.

Registration & Login Sequence Diagram

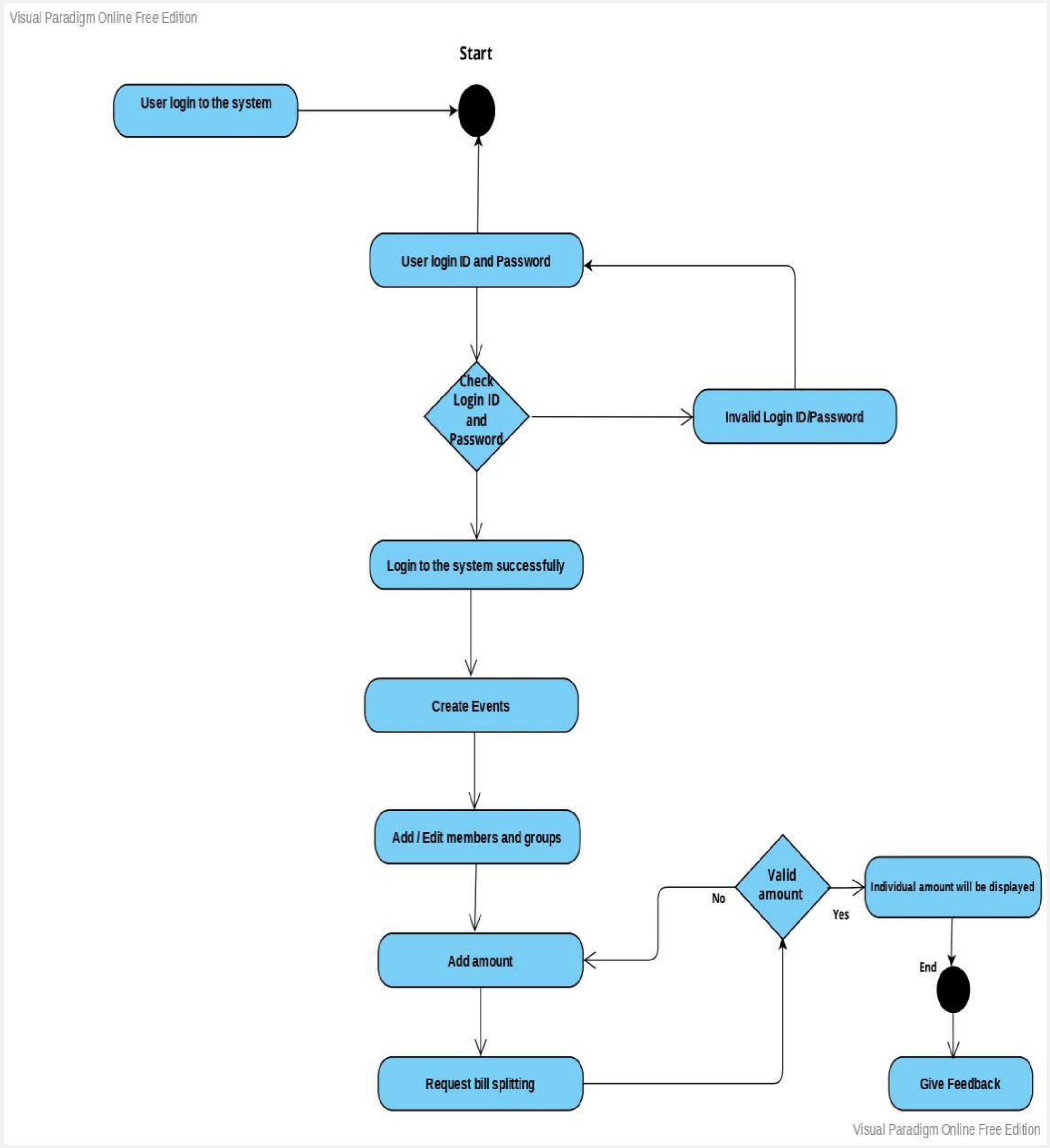

Actor
Login Page
Forgot Password
Verification
Database
Aunthenticate Page
Check Authenticity for access
Login to page
Forgot Password
Check Login
Provide Authorization for access
Valid Login Details
check security question and answer
Create Session in and store in Database
Send email to user to reset password
Invalid Login Details
Allow user to Access the page
Logout from application
Login Successfully

User Interaction Sequence Diagram

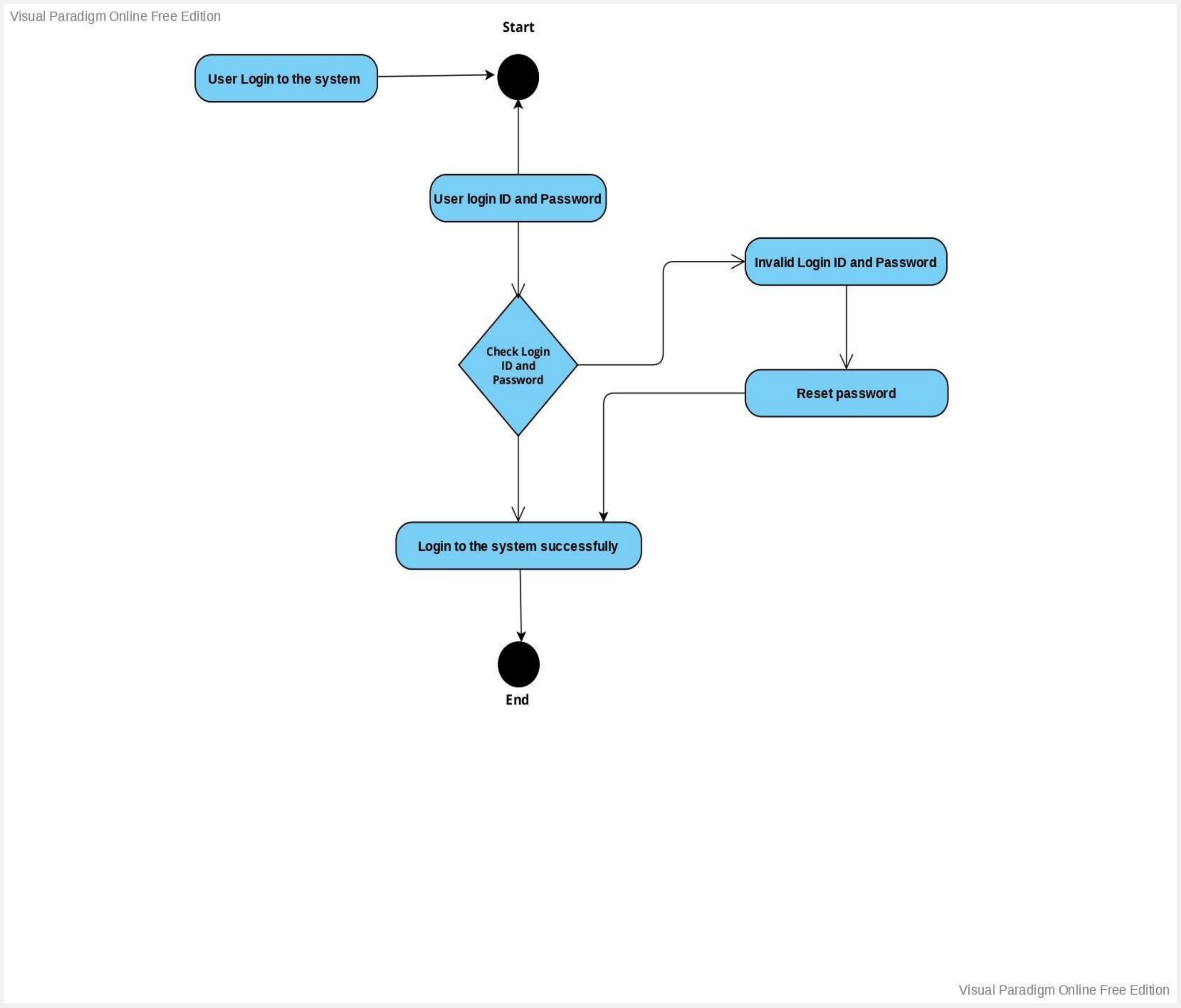


Control and Data Flows through Activity Diagrams

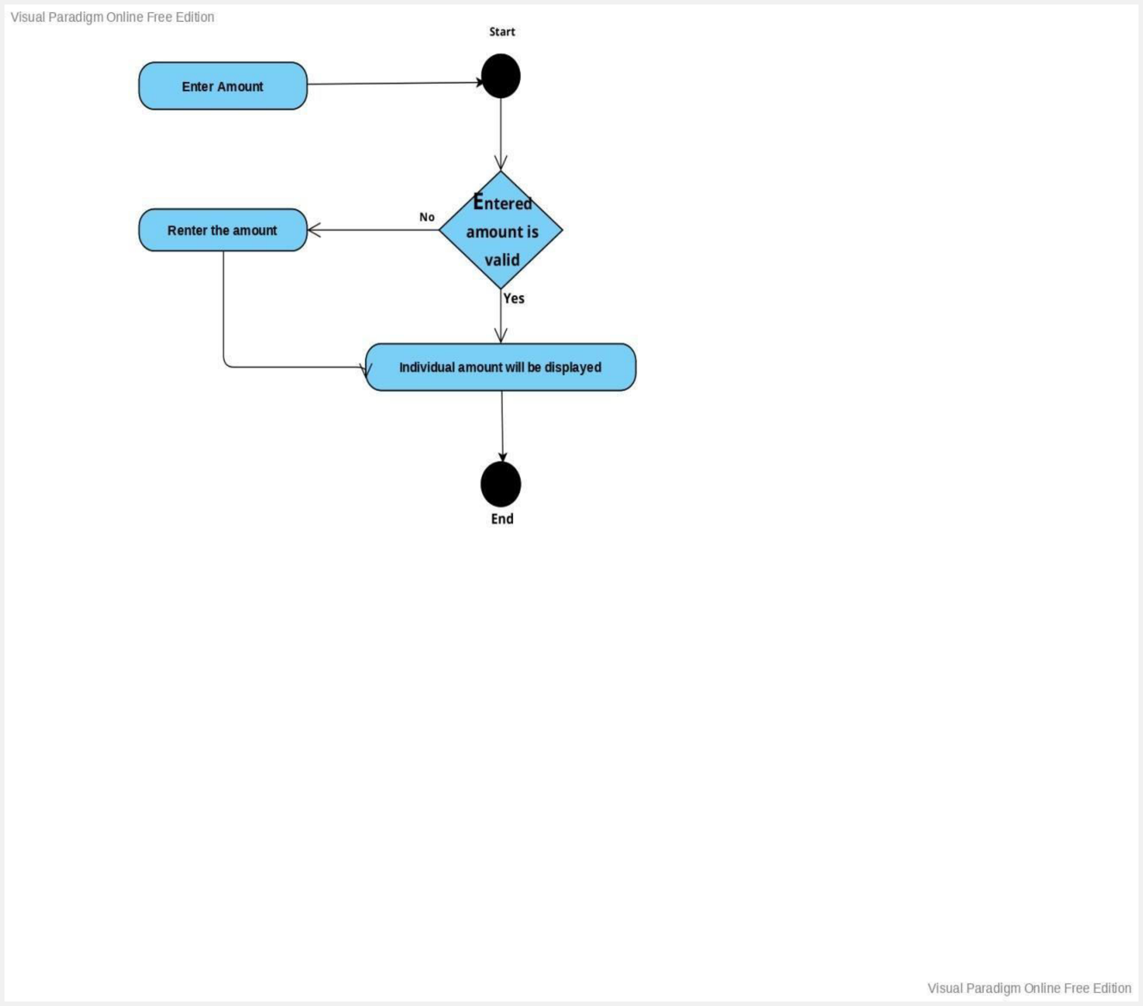
Complete Activity Diagram


Visual Paradigm Online Free Edition
Start
User login to the system
User login ID and Password
Check Login ID and Password
Invalid Login ID/Password
Login to the system successfully
Create Events
Add / Edit members and groups
Add amount
Request bill splitting
Valid amount
No
Yes
Individual amount will be displayed
End
Give Feedback
Visual Paradigm Online Free Edition

Login Authentication Activity Diagram

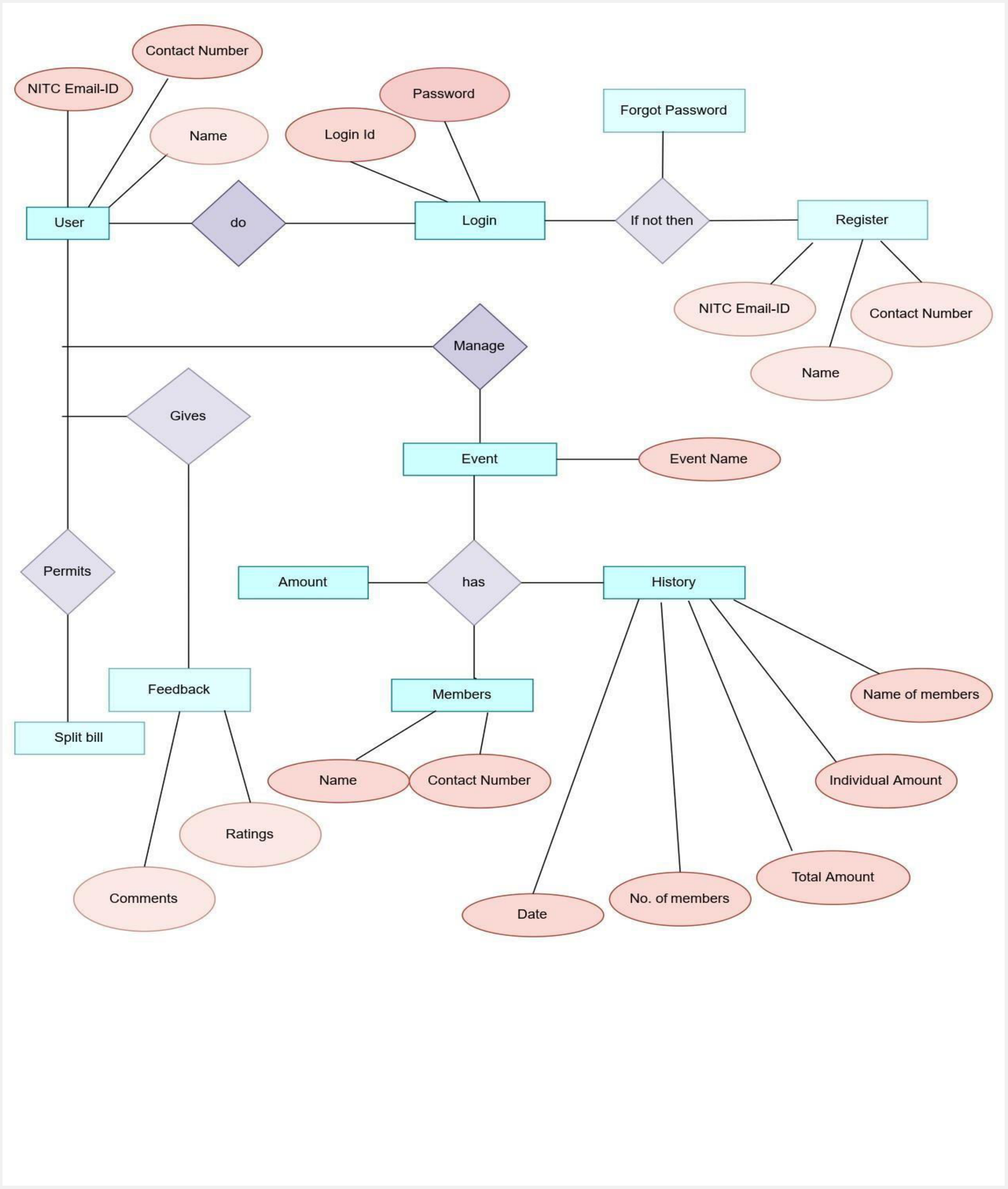

Visual Paradigm Online Free Edition
Start
User Login to the system
User login ID and Password
Invalid Login ID and Password
Check Login ID and Password
Reset password
Login to the system successfully
End
Visual Paradigm Online Free Edition

Amount Validation Activity Diagram


Visual Paradigm Online Free Edition
Start
Enter Amount
Entered amount is valid
No
Renter the amount
Yes
Individual amount will be displayed
End
Visual Paradigm Online Free Edition

2. Database Design

ER Diagram



3. Implementation Plans

Technology Stack

The application will be built using Android Studio, in which we will use concepts of Object-Oriented Programming by implementing it in JAVA.

Android Studio -: Android Studio is Android's official IDE. It is purpose-built for Android to accelerate the development and help us to build the highest-quality apps for every Android device.

Based on IntelliJ IDEA, Android Studio provides the fastest possible turnaround on coding and running workflow.

Some of its main and important features are-:

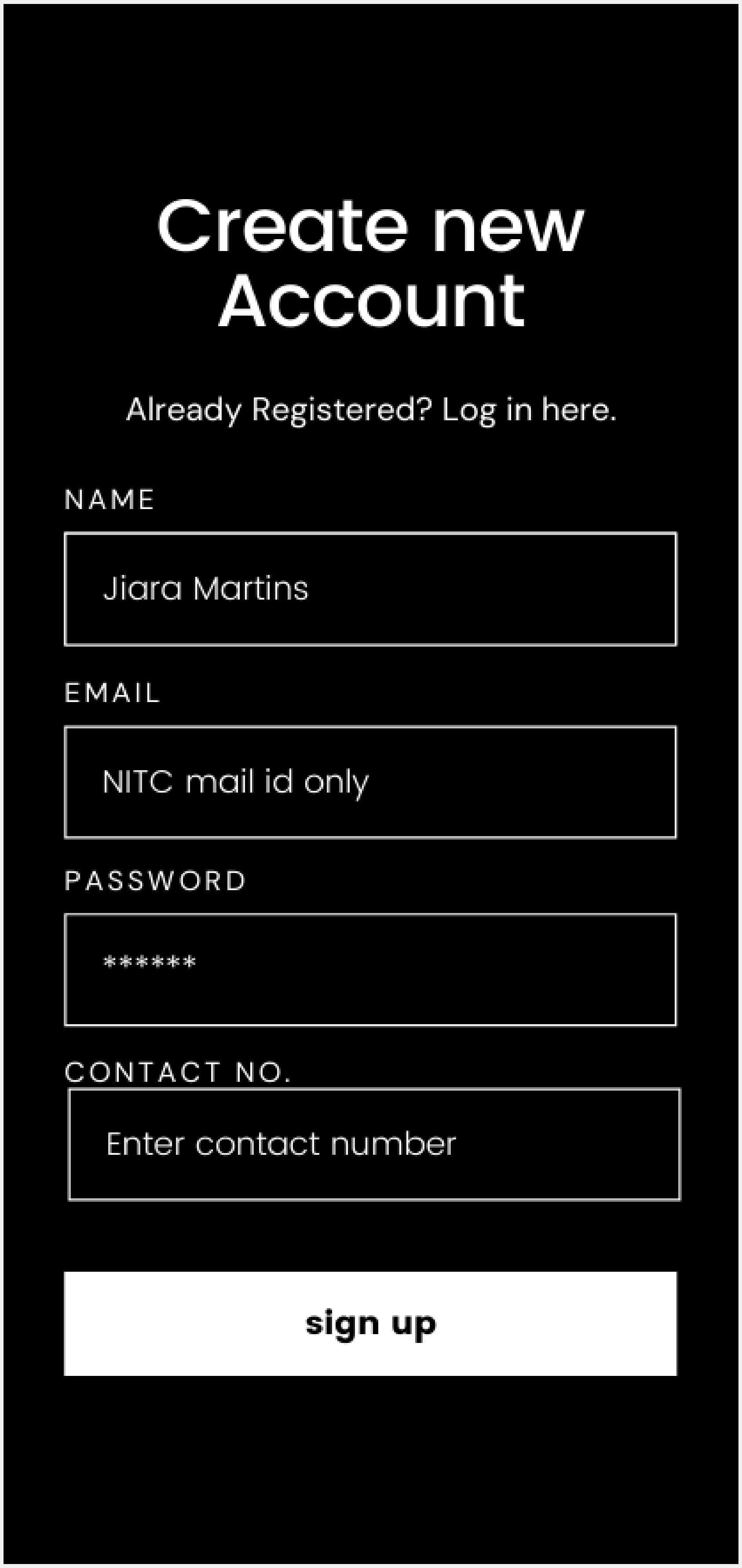
1. Apply Changes -: This feature let us push code and resource changes to our running app without restarting the app.
2. Intelligent Code Editor -: The code editor helps us to write better code, work faster, and be more productive by offering advanced code completion
3. Designed for teams -: Android Studio integrates with version control tools, such as GitHub and Subversion, that helps the team in sync with project and build changes.
4. Code Templates and Sample App -: Android Studio includes project and code templates that make it easy to add well-established patterns such as a navigation drawer and view pager.

JAVA -: Java is a popular programming language. It was created in 1995 and is operated by Oracle. Java runs on more than 3 billion computers. It is used for **mobile app development** (especially Android apps), the development of web applications, web servers and application servers, game development and database association. **Java** is the common option for building high-performance mobile applications nowadays.

Reasons why Java is more popular to develop android apps -:

1. The primary benefit of using **Java for android development** is that it provides the concepts of OOPS (Object Oriented Programming) and is more proficient because they are extensible, scalable and adaptable.
2. The main reason for choosing Java in app development is that it is easy to learn for building mobile applications.
3. Security is the integral part of any mobile app **design**. The compiler, interpreter, and runtime environment – all have been developed in the Java programming language keeping security in mind.
4. With a cost-effective **Java mobile app development agency** which supports all your requirements, building high-performance apps on low investment is possible.

User Interface Prototyping


Create new Account

Already Registered? Log in here.

NAME

Jiara Martins

EMAIL

NITC mail id only

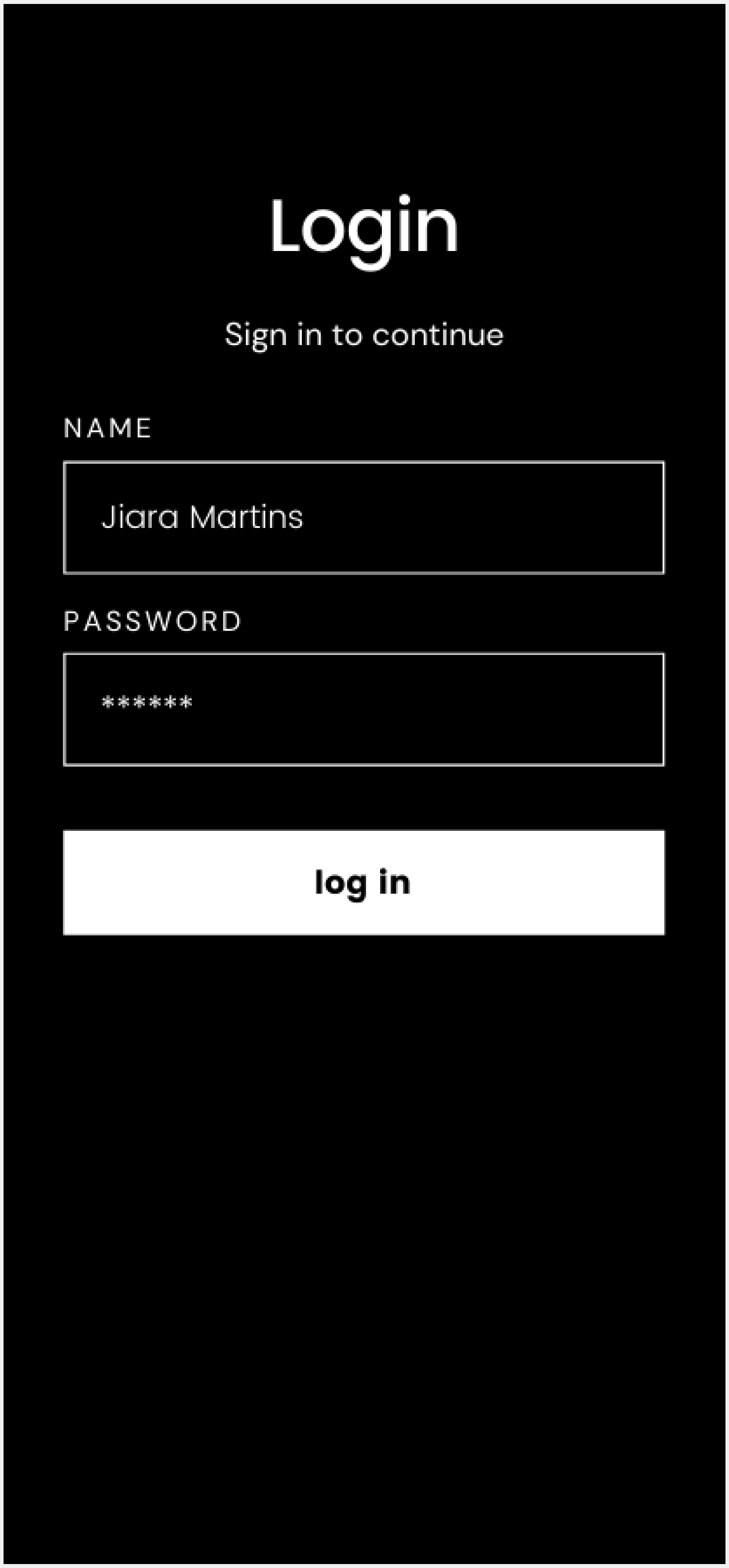
PASSWORD

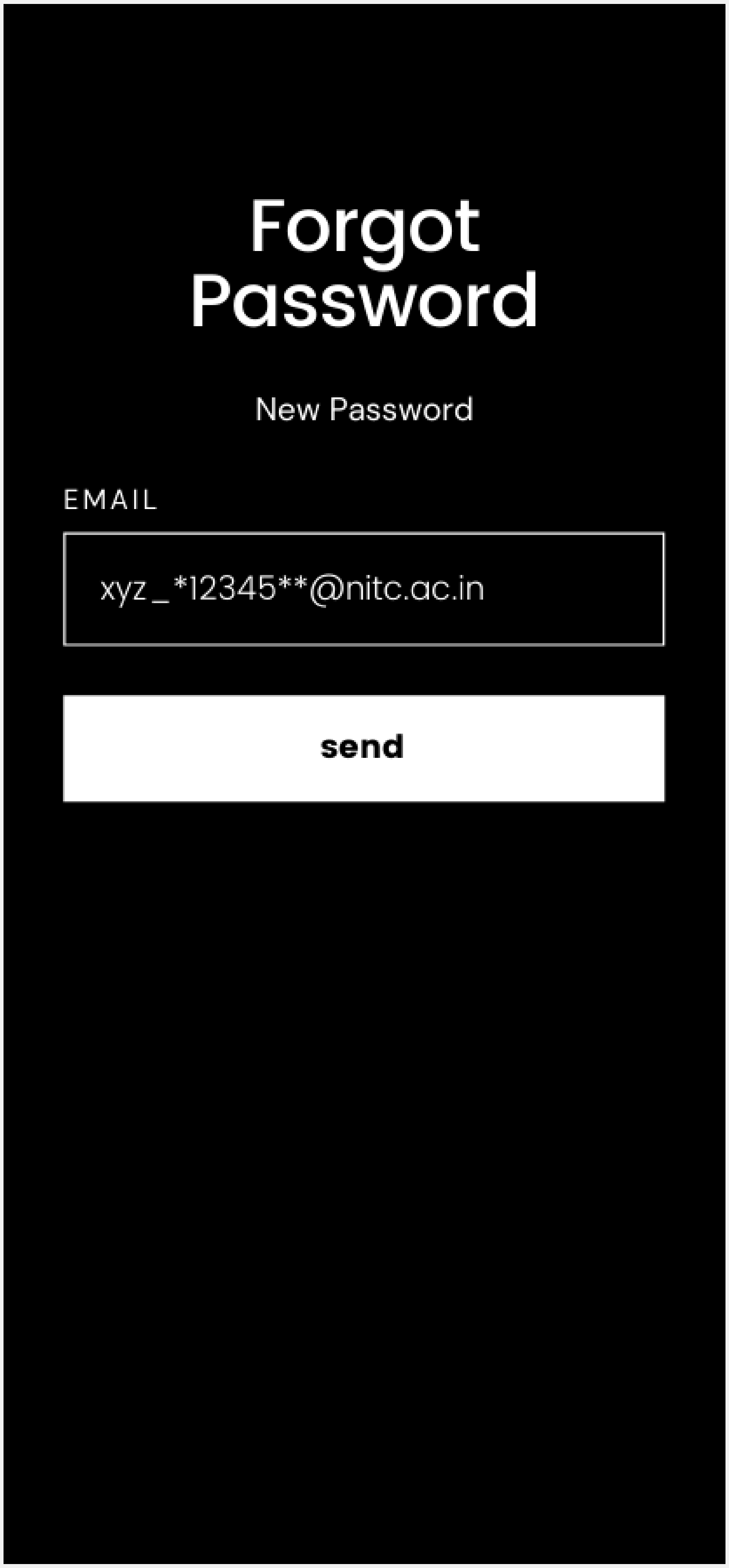
******

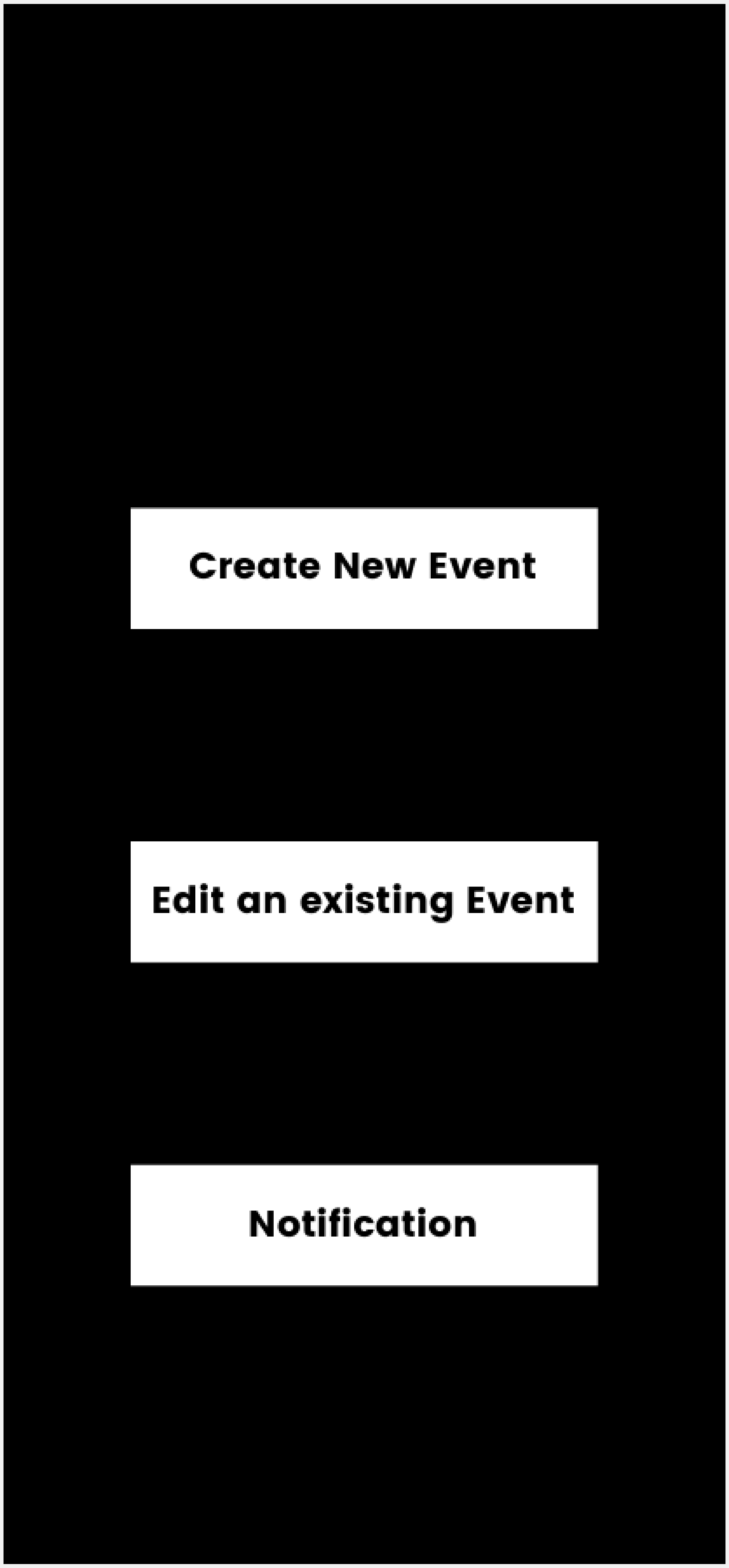
CONTACT NO.

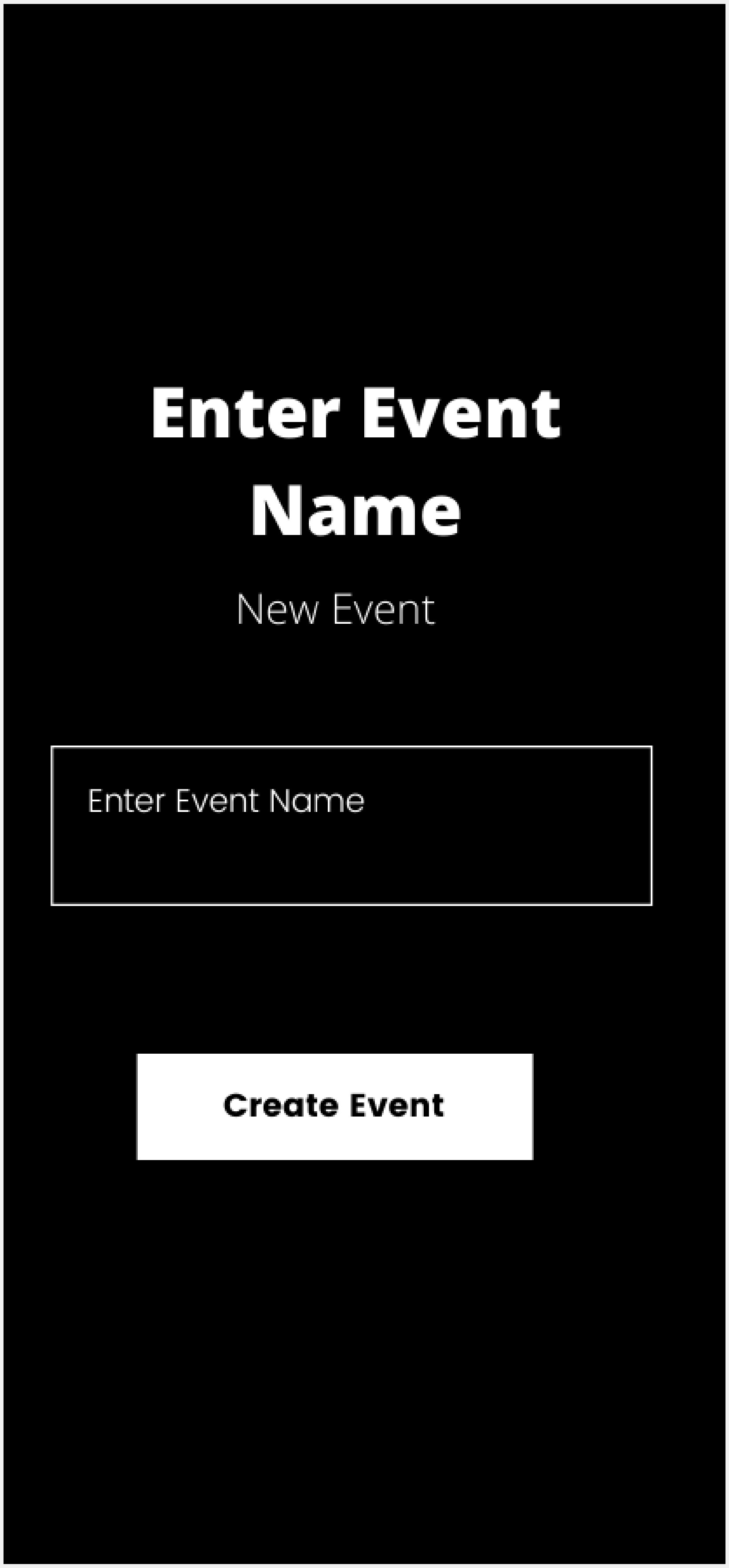
Enter contact number

sign up


Login
Sign in to continue
NAME
Jiara Martins
PASSWORD
******
log in


Forgot Password
New Password
EMAIL
xyz_*12345**@nitc.ac.in
send


Create New Event
Edit an existing Event
Notification

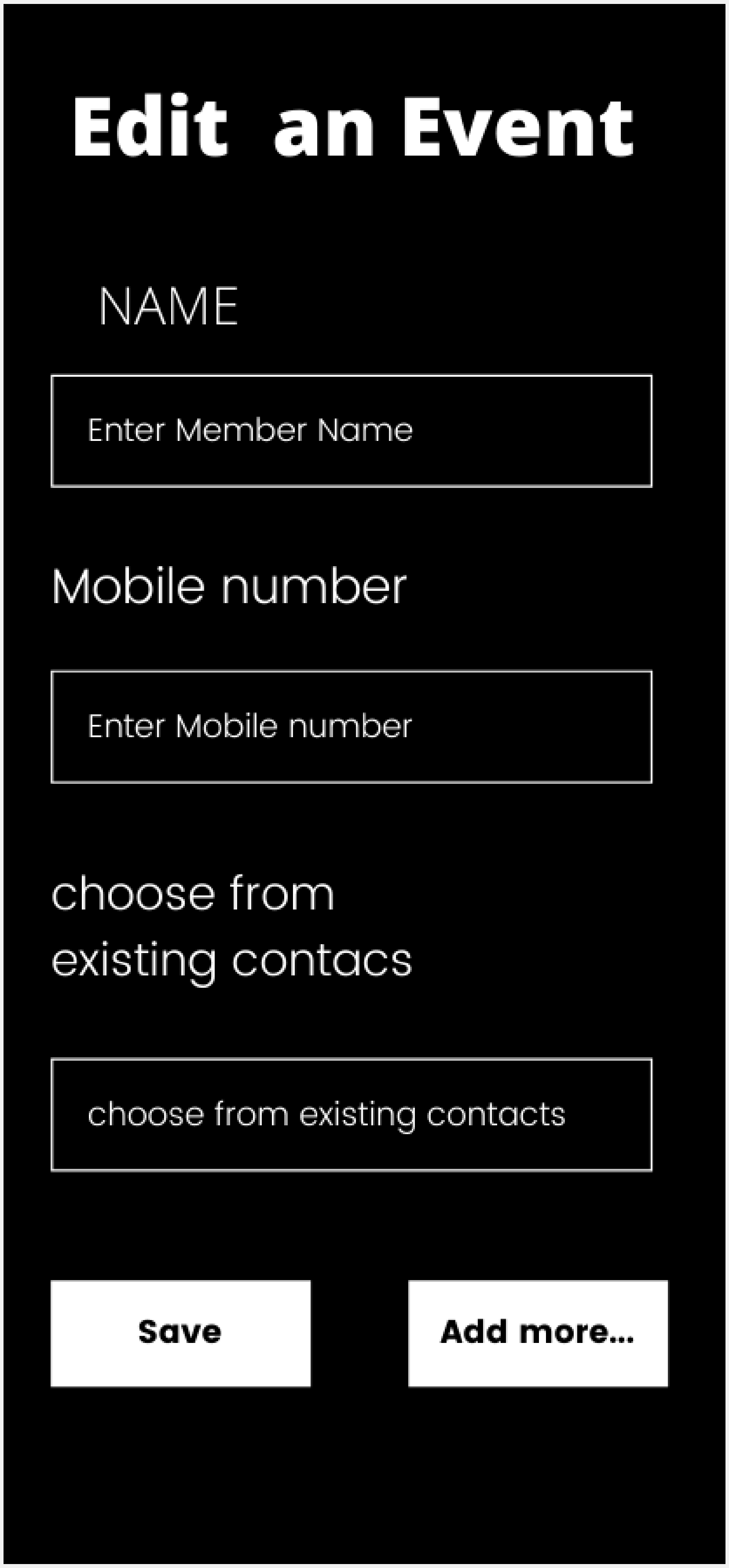


Enter Event Name

New Event

Enter Event Name

Create Event



Edit an Event

NAME

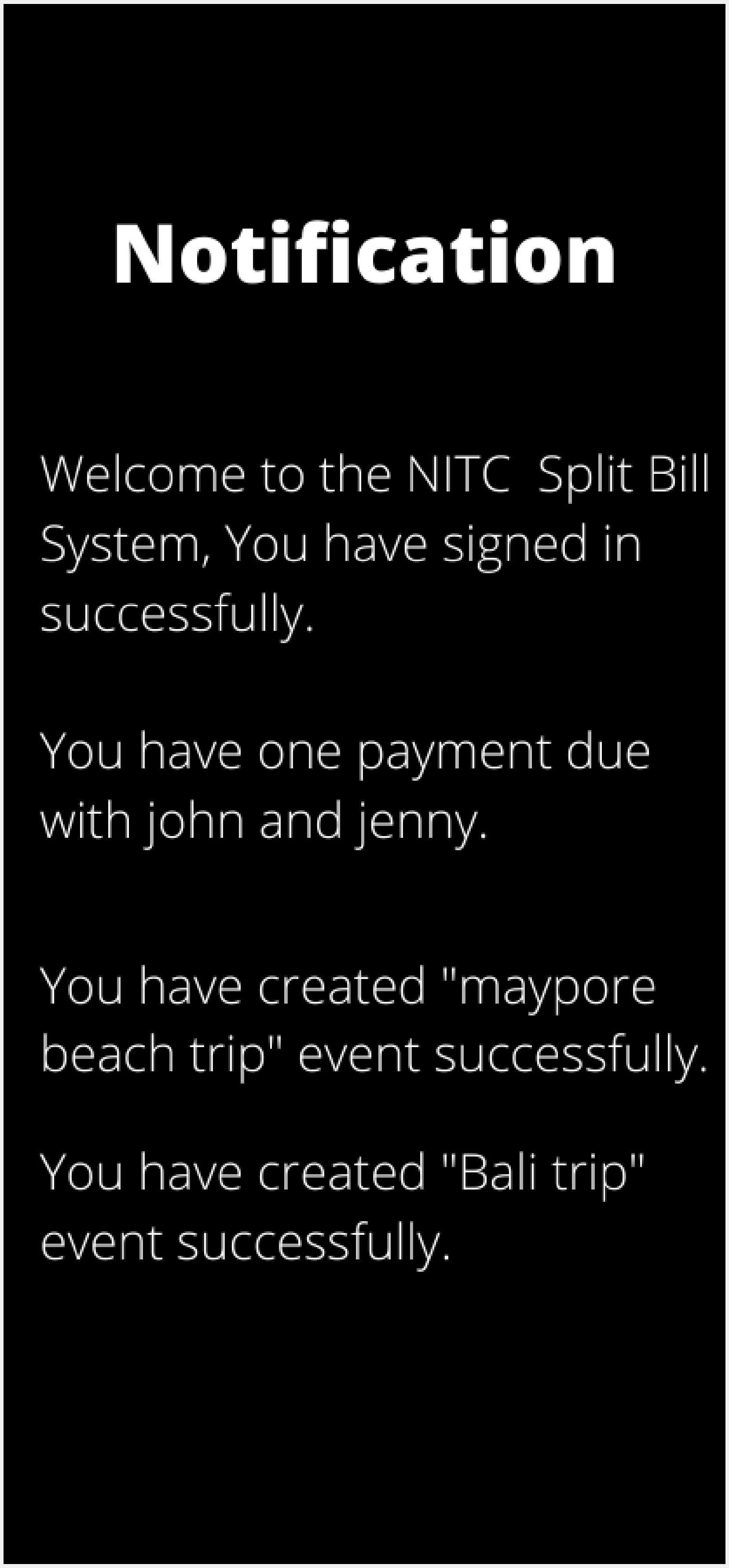
Enter Member Name

Mobile number

Enter Mobile number

choose from
existing contacs

choose from existing contacts



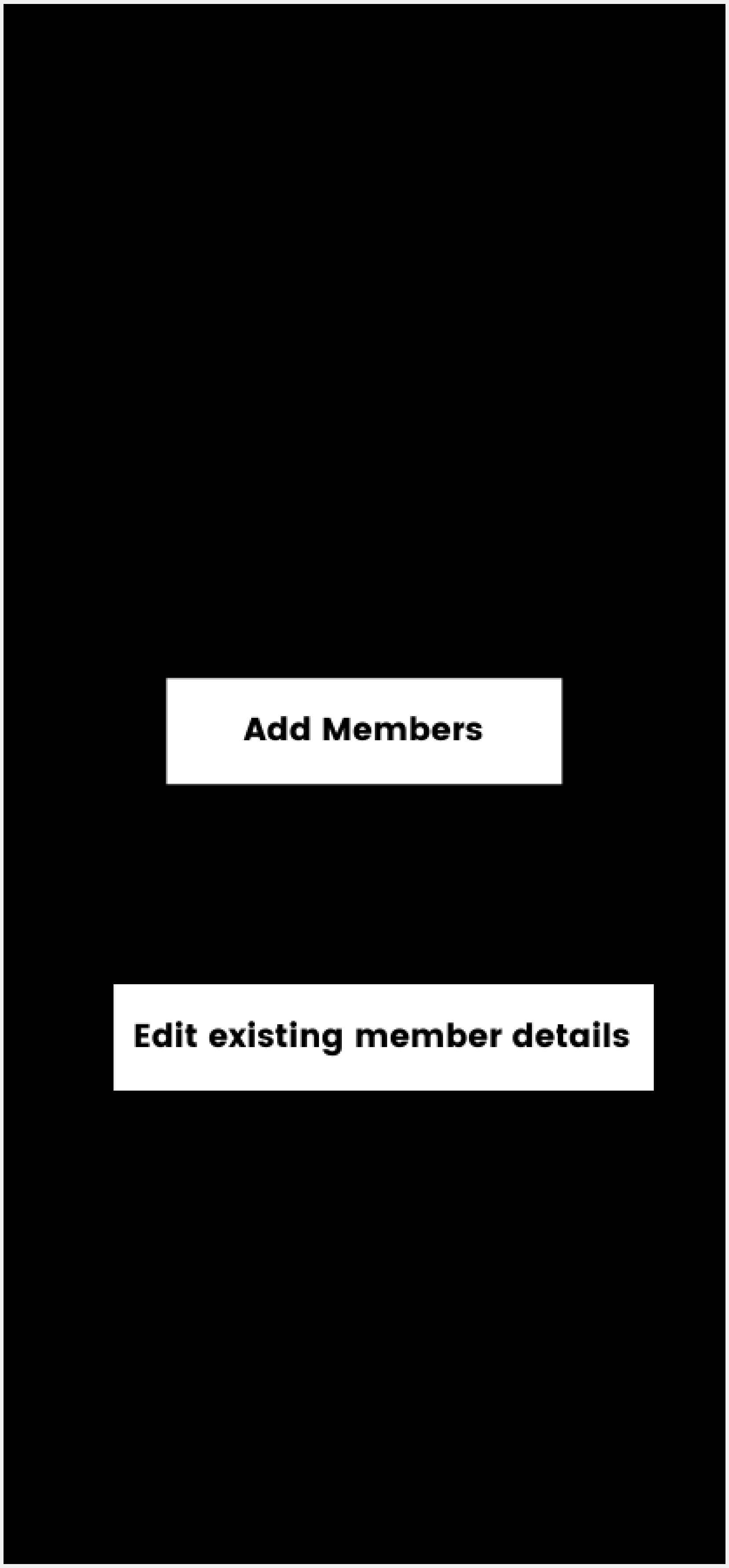
Notification

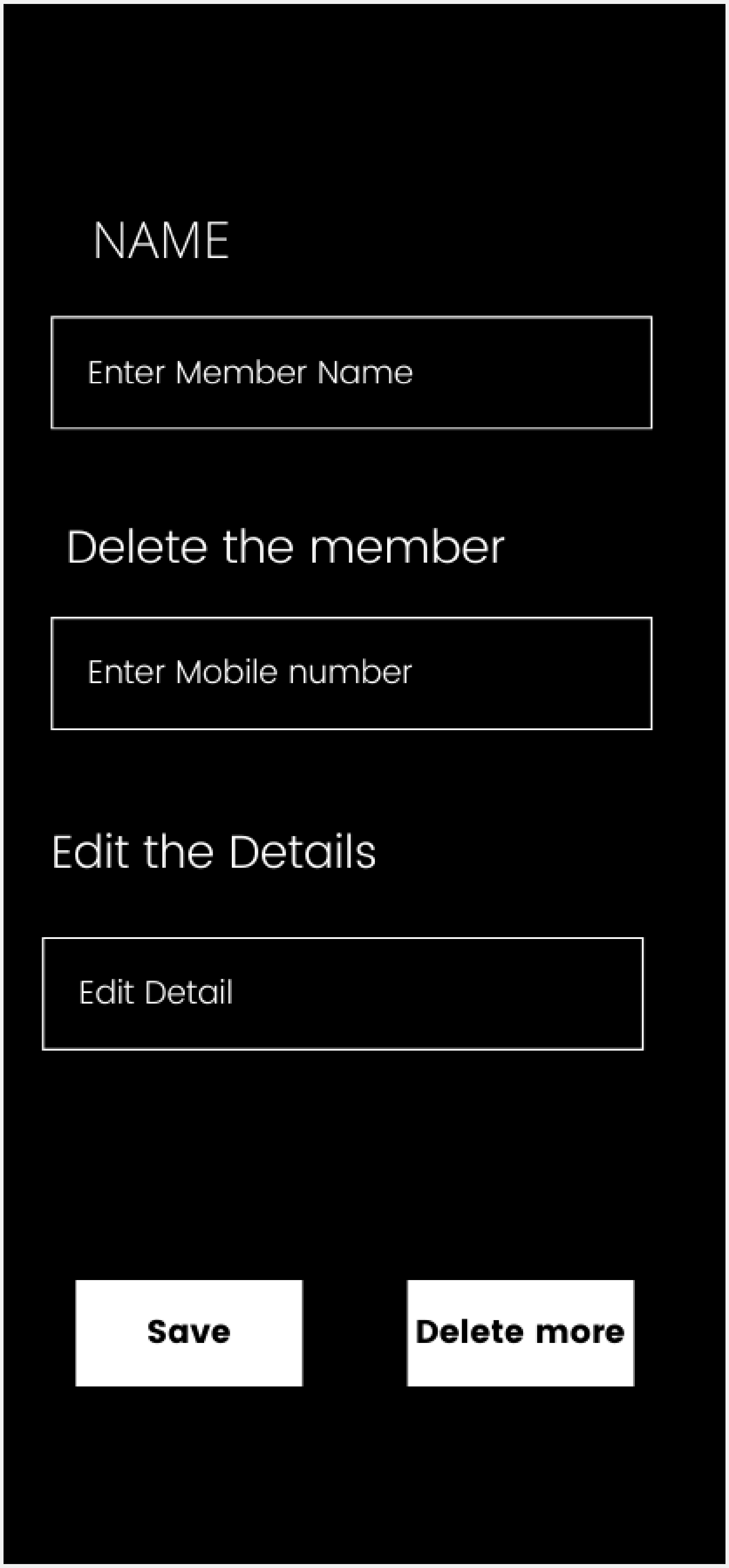
Welcome to the NITC Split Bill System, You have signed in successfully.

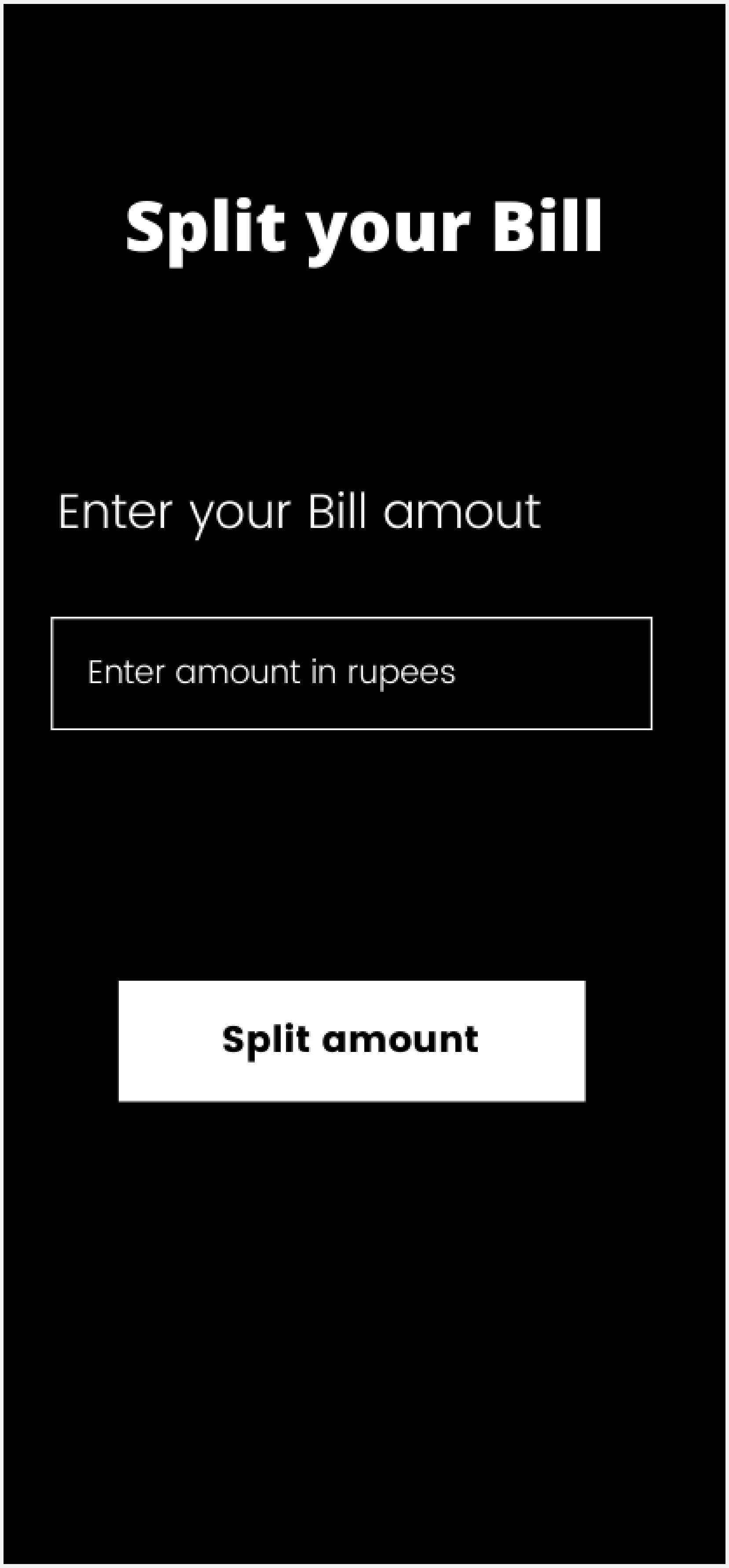
You have one payment due with john and jenny.

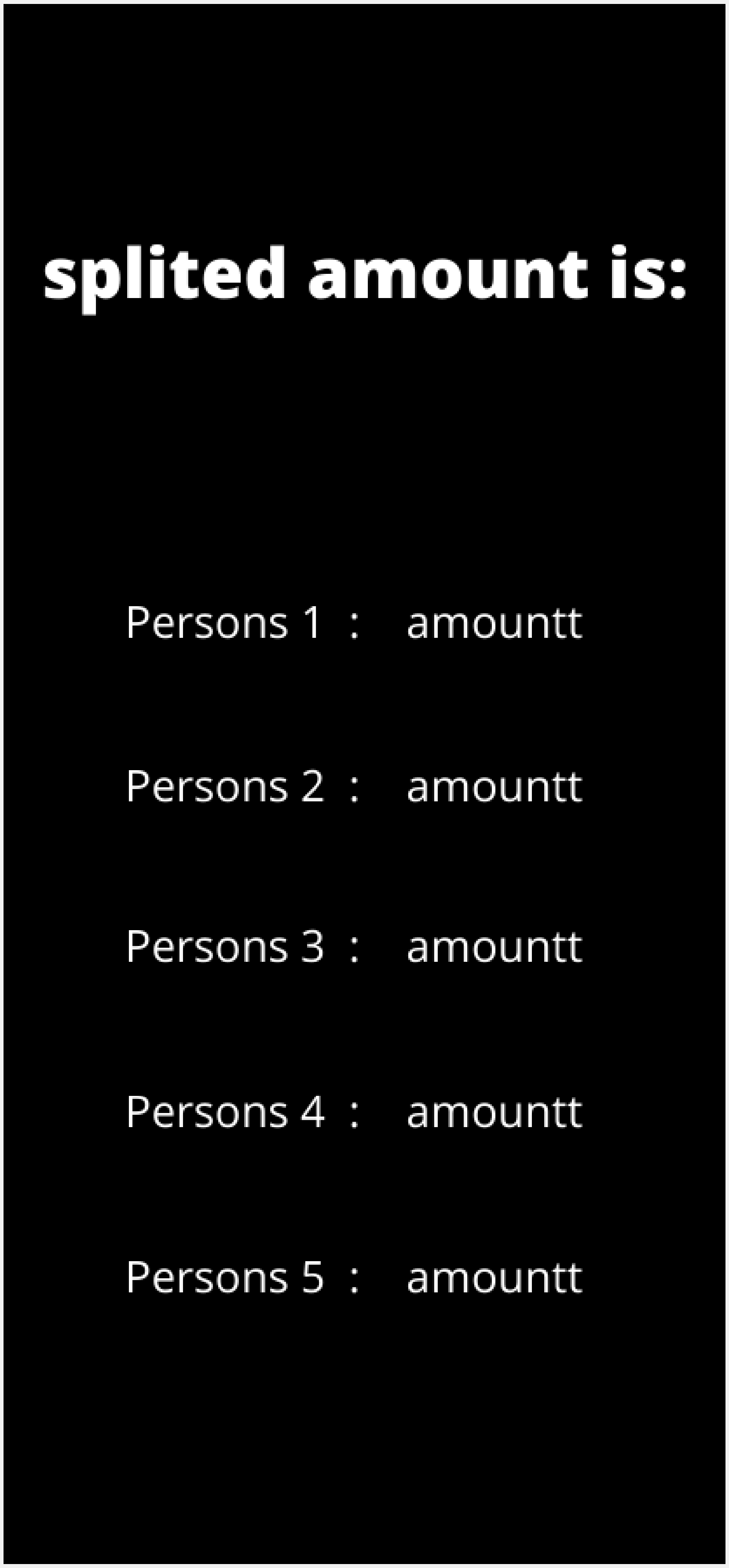
You have created "maypore beach trip" event successfully.

You have created "Bali trip" event successfully.


Add Members
Edit existing member details


NAME
Enter Member Name
Delete the member
Enter Mobile number
Edit the Details
Edit Detail
Save
Delete more


Split your Bill
Enter your Bill amout
Enter amount in rupees
Split amount



splited amount is:

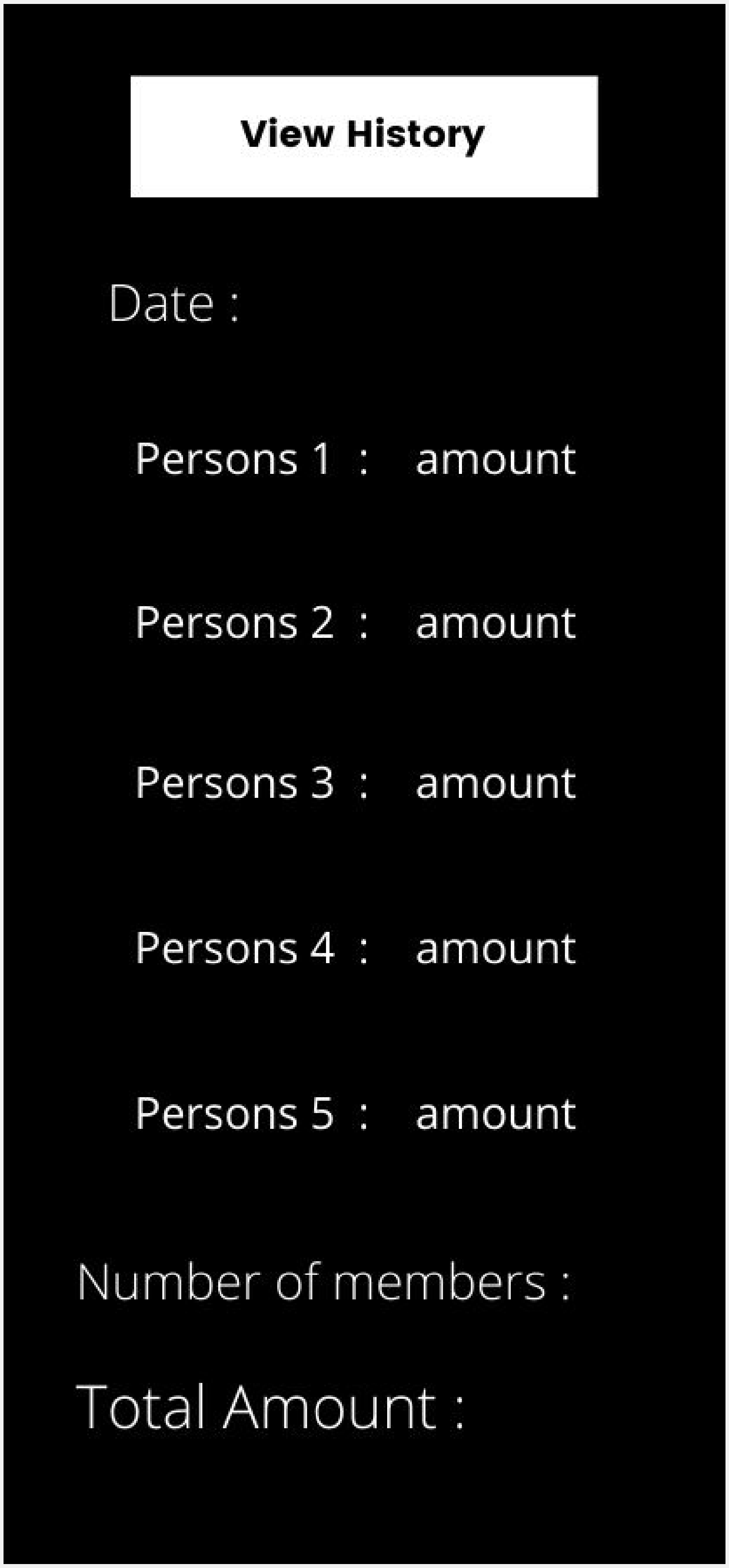
Persons 1 : amountt

Persons 2 : amountt

Persons 3 : amountt

Persons 4 : amountt

Persons 5 : amountt



Date :

Persons 1 : amount

Persons 2 : amount

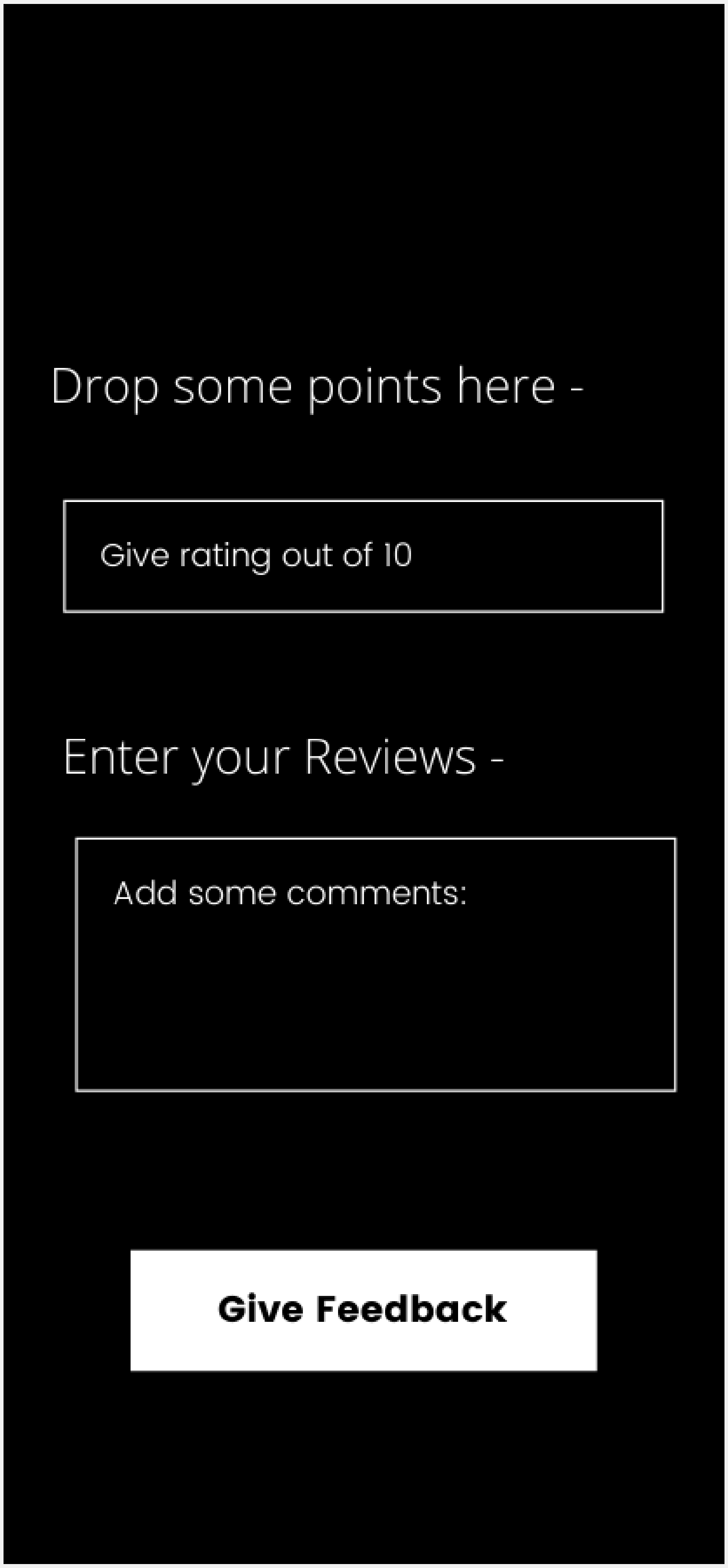
Persons 3 : amount

Persons 4 : amount

Persons 5 : amount

Number of members :

Total Amount :


Drop some points here -

Give rating out of 10

Enter your Reviews -

Add some comments:

Give Feedback

4. Test Cases

Test Case #1 (TC_REG_001)

Author: Ayushi Kumari
Test Case Description:
Test Scenario: Verify Registration
Test Case: Enter a valid NITC mail ID.
Preconditions: User must have a valid NITC mail ID.
Test Steps:
Step 1: Enter Username.
Step 2: Enter User mail.
Step 3: Enter Contact.
Step 4: Enter password.
Step 5: Click on Submit.
Test Data:
Username: Ayushi Kumari.
User_mail: ayushi_m200687ca@nitc.ac.in.
Contact: 9876543210
Password: <1234nitc>
Expected Result:
If user registration is successful, then a pop-up message will be showed to the user that "You are registered successfully".
But if registration failed, then a pop-up message will be showed to the users that "Please enter a valid NITC Mail ID".
Post conditions: User must be shown a message that "You are registered successfully".

Test Case #2 (TC_LOGIN_002)

Author: Isha Gupta
Test Case Description:
Test Scenario: Verify the application login.
Test Case: Enter valid mail and password.
Preconditions: User must be a registered user.
Test Steps:
Step 1: Enter User_mail.
Step 2: Enter Password.
Step 3: Click on Login Button.
Test Data:
User_mail: ayushi_m200687ca@nitc.ac.in.
Password: <1234nitc>
Expected Result: Login Successfully.
Post conditions: Event page will be shown in which user can create his events.

Test Case #3 (TC_LOGIN_003)

Author: Aman Kumar Sankhwar
Test Case Description:
Test Scenario: Verify forgot password function.
Test Case: Enter valid mail ID.
Preconditions: User must be a registered user.
Test Steps:
Step 1:Click on Forgot Password Button.
Step 2: Enter Mail ID.
Step 3: Click on Submit.
A message of reset password will be sent to that mail ID.
Step 4: Click on Reset Password.
Step 5: Enter new Password.
Step 6: Click on OK.
Test Data:
User_mail: ayushi_m200687ca@nitc.ac.in.
New Password: <0000>
Expected Result: Password reset is completed.
Post conditions: User can now login to the application using his new password.

Test Case #4 (TC_EVENT_004)

Author: Amarjeet Budhsaini
Test Case Description:
Test Scenario: Verify if the event function for creating new event is working perfectly or not.
Test Case: Enter event name.
Preconditions: User must login to the application.
Test Steps:
Step 1: Click on create event.
Step 2: Enter Event name.
Step 3: Click on save Button.
Test Data:
Event name: Travelling.
Expected Result: Successfully created new event.
Post conditions: Add group page will be shown to the user where he can add groups into the event.

Test Case #5 (TC_GROUP_005)

Author: Ayushi Kumari
Test Case Description:
Test Scenario: Verify if we can add group into the event or not.
Test Case: Enter group name.

Preconditions: User must have created an event.

Test Steps:

Step 1: Click on create group.
Step 2: Enter Group name.
Step 3: Click on save Button.

Test Data:

Group name: Fly High.

Expected Result: Successfully added group into the event.

Post conditions: Add member page will be shown to the user where he can add members into the group.

Test Case #6 (TC_AddMembers_006)

Author: Ayushi Kumari

Test Case Description:

Test Scenario: Verify if user can add members in the group.
Test Case: Enter member name and contact.

Preconditions: User must have created a group.

Test Steps:

Step 1: Click on Add Members.
Step 2: Enter Member's name.
Step 3: Enter Member's Contact/ choose member from contact list.
Step 4: Click on save Button.

Test Data:

Member's name: Isha Gupta
Member's Contact: 1234567890

Expected Result: Member is added successfully into the group.

Post conditions: Two options will be shown.

1. Add more members.
2. Enter amount.

Test Case #7 (TC_EditGroup_007)

Author: Isha Gupta.

Test Case Description:

Test Scenario: Verify if user can edit group or not.
Test Case: Enter group name.

Preconditions: User must login to the application.

Test Steps:

Step 1: Click on existing group.
Step 2: Click on a particular group.
Step 3: Click on edit group Button.
Step 4: Click on add Member.
Step 5: Enter member name and contact.
Step 6: Click on Save Button.

Step 7: Click on delete Members
Step 8: Click on delete button near to the name of the member.

Test Data:
Member's name: Aman Kumar Sankhwar.
Member's Contact: 9867543423
Expected Result: Successfully add and delete members from an existing group.
Post conditions: Group page will be shown.

Test Case #8 (TC_BILL _008)

Author: Aman Kumar Sankhwar.
Test Case Description:
Test Scenario: Verify enter bill option.
Test Case: Enter amount.
Preconditions: User must have created an event and added members into it.
Test Steps:
Step 1: Click on enter amount option.
Step 2: Enter amount.
Step 3: Click on Split Bill Button.
Test Data:
Enter Amount: 1000
Expected Result: Successfully read the bill.
Post conditions: Individual Bill will be shown.

Test Case #9 (TC_HISTORY _009)

Author: Amarjeet Budhsaini
Test Case Description:
Test Scenario: Verify if History is showing.
Test Case: Enter date and event name.
Preconditions: The user should have the previous history of splitting the bill.
Test Steps:
Step 1: Click on History Option.
Step 2: Enter Date.
Step 3: Enter Event name.
Step 4: Click on Show Button.
Test Data:
Enter Date: 20-02-2022
Enter Event name: Canteen Payment.
Expected Result: Individual Bill will be shown and also user can download it.
Post conditions: The data should be fetched from the database and will be shown to the user.

Test Case #10 (TC_FEEDBACK _010)

Author: Ayushi Kumari
Test Case Description:
Test Scenario: Verify feedback function.
Test Case: Enter Comments.
Preconditions: If the user wants to convey his views about the performance of the application.
Test Steps:
Step 1: Click on Send Feedback Option.
Step 2: Enter Comments.
Step 3: Click on save Button.
Test Data:
Feedback: Splitting the bill is very helpful with this application.
Expected Result: User can send his views about the app.
Post conditions: The feedback must be saved into the database.

5. Traceability

Requirements

R1: System shall allow new users to register to the application.
R2: System shall allow the existing users to login.
R3: Automatic Calculation of money owed by each member of the group.
R4: Customizable formula for splitting costs.
R5: Transaction history (including all payments made and costs incurred by each individual).
R6: Support for multiple transactions which can take place over extended period of time.
R7: Automatic notifications, alerting uses of pending debts and/or otherrelevant info.
R8: Transaction synchronization, keeping each group member up-to-date at all times.

R9: Receipt storage, which allows the user(s) to save photos of receipts associated with particular purchase/expenses.

		Class Diagram	ER Diagram	Sequence Diagram	Proto Typing	Test Cases	Use Case
REQUIREMENTS	R1		X	X	X	4.1	X
	R2	X	X	X	X	4.2, 4.3	X
	R3		X		X	4.7	X
	R4				X	4.7	
	R5	X	X		X	4.8	X
	R6			X		4.4, 4.5	
	R7					4.7, 4.8	X
	R8			X	X	4.7, 4.8	
	R9		X		X	4.8	X

References

- Draw.io
- Visual paradigm
- developer.android.com
- freeprojectz.com
- javatpoint
- geeksforgeeks

●