Section 1 - Project Description

1.1 Project
SplitIt

1.2 Description
Splitit aims to allow users to split fairly when among a group of people without the hassle of performing complex computations. It could do a quick split that will have a basic division or a more detailed split of a bill that goes more in-depth and will be fairer.

1.3 Revision History

Date	Comment	Author
10/8/23	0.1 Initial Draft	Sydney Khiev Ren Hao Wong Brandon Becerra Zhen Wei Ng Roberto Cruz
10/28/23	0.2 Updating multiple sections	Sydney Khiev Ren Hao Wong Brandon Becerra Zhen Wei Ng Roberto Cruz
11/12/23	0.3 Revision/Adding to more sections	Sydney Khiev Ren Hao Wong Brandon Becerra Zhen Wei Ng Roberto Cruz
11/15/23	0.4 Revision and updating multiple sections	Sydney Khiev Ren Hao Wong Brandon Becerra Zhen Wei Ng Roberto Cruz
12/8/23	0.5 Final Revision	Sydney Khiev Ren Hao Wong Brandon Becerra Zhen Wei Ng Roberto Cruz

Contents

Section 2 - Overview

2.1 Purpose

The app's purpose is to create and manage a group of people within the application to split the bill, allow users to track bills, and track any pending bills that are yet to be split or resolved. The app is intended to serve people who gather together or are alone and want to split their expenses among themselves. The goal is to split the bill and remove the hassle of splitting it up manually.

Main Focus:
Provide a user-friendly interface that is accessible for anyone who needs to split a bill on a regular basis like family, friends, parties, students, etc. by providing a fast, simple, and easy way to split a bill.

Product Value:
Bill splitting amongst a large group can be a bothersome process. To circumvent the dilemma, bill-splitting provides a simplified solution to split bills based on the user's parameters. Essentially, most of the labor will be abstracted away, saving time and mental burden.

Intended audience:
Anyone who needs to split a bill on a regular basis like family, friends, parties, couples, roommates, and students. The product is intended to serve the people who gather together or even the people who are alone and want to split their expenses among themselves.

Benefits:
Providing users a fast, simple, and easy way to split a bill among other people Objectives & Goals: Easy way to split the bill and remove the hassle of splitting the bill up manually.

The benefits of having a bill-splitting application like this, are the capabilities anyone would have with keeping track of their own expenses when splitting a bill. The objective of our project is to create a platform where you can manage your own spending while also seeing the spending of everyone else that the bill is split among.

2.2 Scope

The scope of the bill splitter application is to accommodate features for recording bill information and specify friends that are involved with the bill. Users will be able to view, edit, and delete bills that are associated with their accounts. The application should also support item declaration/specification, split rule modification, tax splitting logic, and tip splitting logic. With all necessary information supplied for the billing record, the application should be able to calculate the appropriate amount owed by each involved individual.

2.3 Requirements

Design Requirements:

- Intuitive and visually appealing navigation for the app.
- Responsive and smooth usage throughout the app.
- Reliable and Protected User Data.

Graphic Requirements:

- Graphic demands on a device should be low to ensure performance
- Appealing and intuitive user interface
- Responsiveness: Able to scale seamlessly at different screen sizes

OS Requirements:

- Major mobile operating systems like Android and IOS
- APK versions may be shipped for Huawei or other OS devices
- OS supported version: Android 6/8 and above, IOS 13/15 and above

Constraints:

- The product meets target audience expectations
- Compliant with Android and IOS regulations
- Developed prototype within the time frame of 11 weeks

2.3.1 Estimates

Parallel work

#	Description	Weeks. Est.
1	Initial Data Setup	2
2	Initial Service Testing	2
3	Login Service	1
4	Registration Service	2
5	Data Refactoring	2
6	Friend Creation	2
7	Preliminary UI	2
8	Bill Display	1
9	Friend Display	2
10	Home Display	1
11	Logo Design	1
12	UX Design	2
13	Other unspecified effort	?
	TOTAL:	11

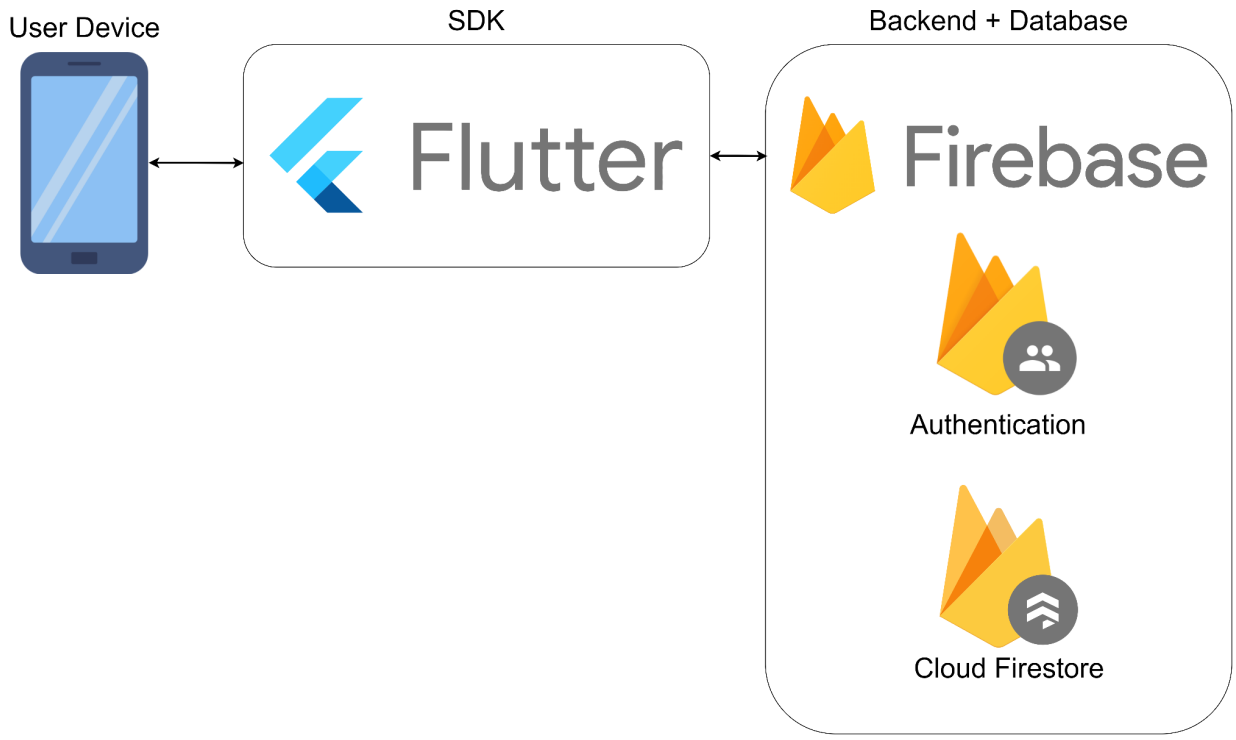
2.3.2 Traceability Matrix

Cross-reference this document with your requirements document and link where you satisfy each requirement

SRS Requirement	SDD Module
Req 3.1, 3.2, 3.3	2.3 (Requirements), 7.1 (User Interface Design Overview)

Section 3 - System Architecture

The application is seamlessly implemented in Flutter which is a versatile and efficient cross-platform framework, ensuring a smooth and consistent user experience across various devices. It concurrently leverages the strong Firebase service to securely store and manage a wide array of client data like emails, passwords, and user IDs but also adeptly handles the storage as well as retrieval of billing information for enhanced functionality and data management.

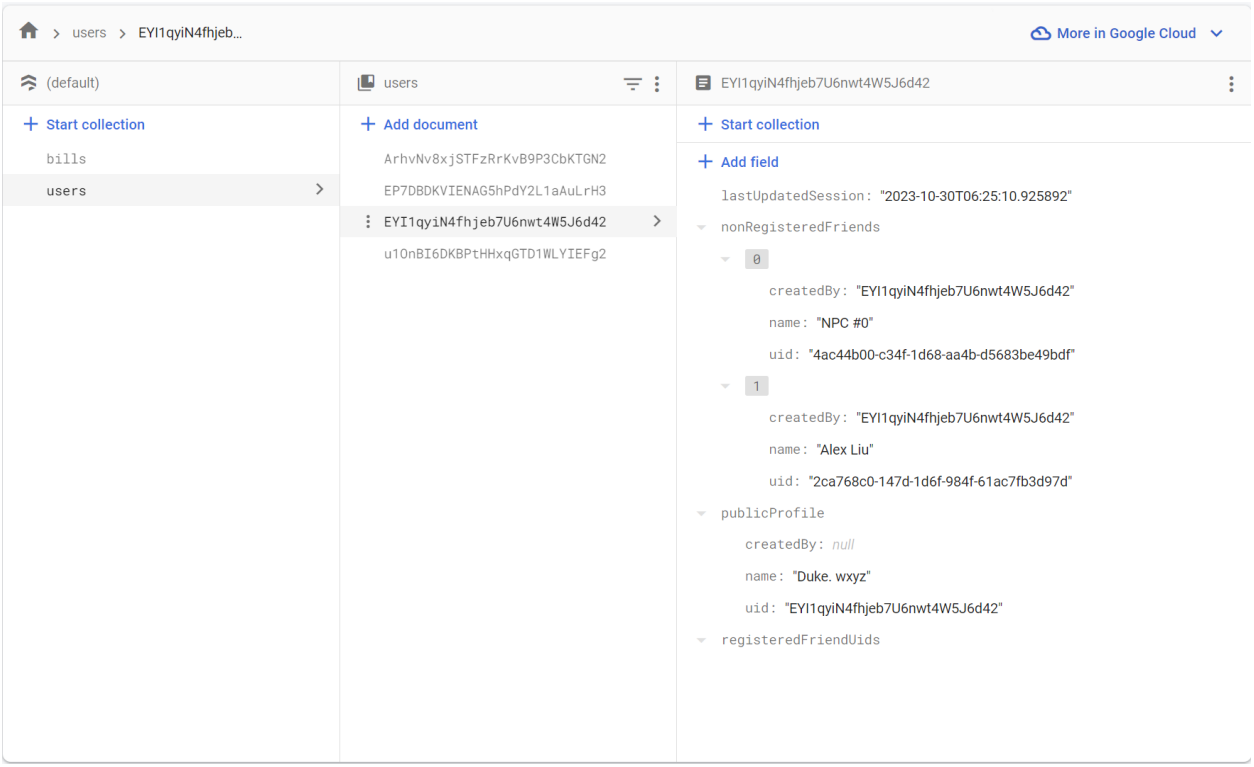


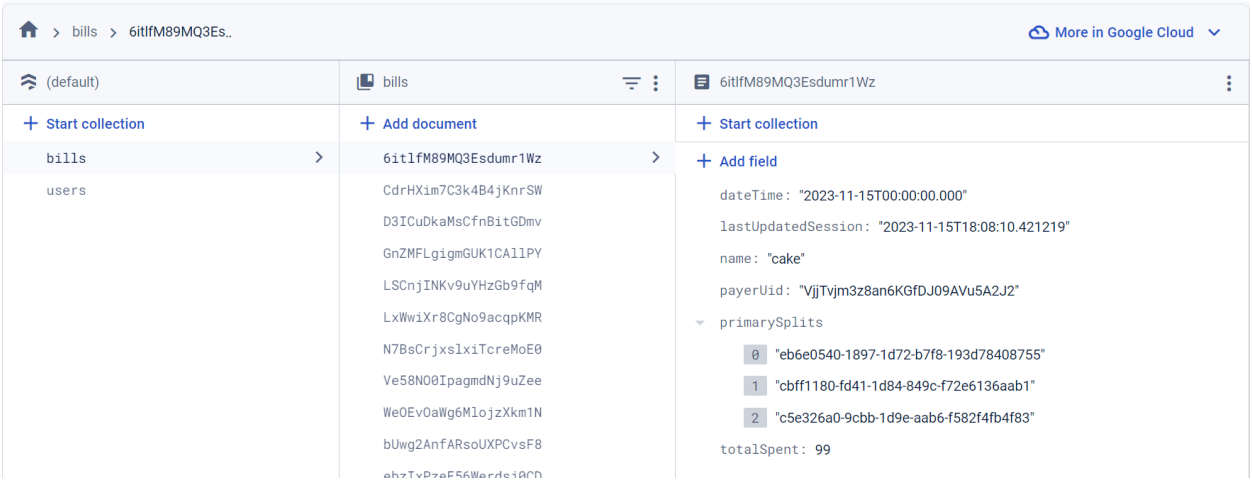
Section 4 - Data Dictionary

PublicProfile		
Field	**Notes**	**Type**
uid	The unique identifier of the profile	string
name	The name of the profile	string
createdBy	The uid of the user that created this profile (Null if belonged to an active user)	string

UserData		
Field	**Notes**	**Type**
lastUpdatedSession	The latest time the user interacted with the app	string
publicProfile	The profile of the user	JSON map
nonRegisteredFriends	The list of inactive friend profiles	List<JSON map>
registeredFriends	The list of uid of friends	List<string>

BillData		
Field	**Notes**	**Type**
dateTime	The date when the bill was created	string
name	The name of the bill	string
payerUid	The uid of the user who paid for the bill	string
totalSpent	The total amount spent on the bill	double
primarySplits	The list of uid of involved users	List<string>

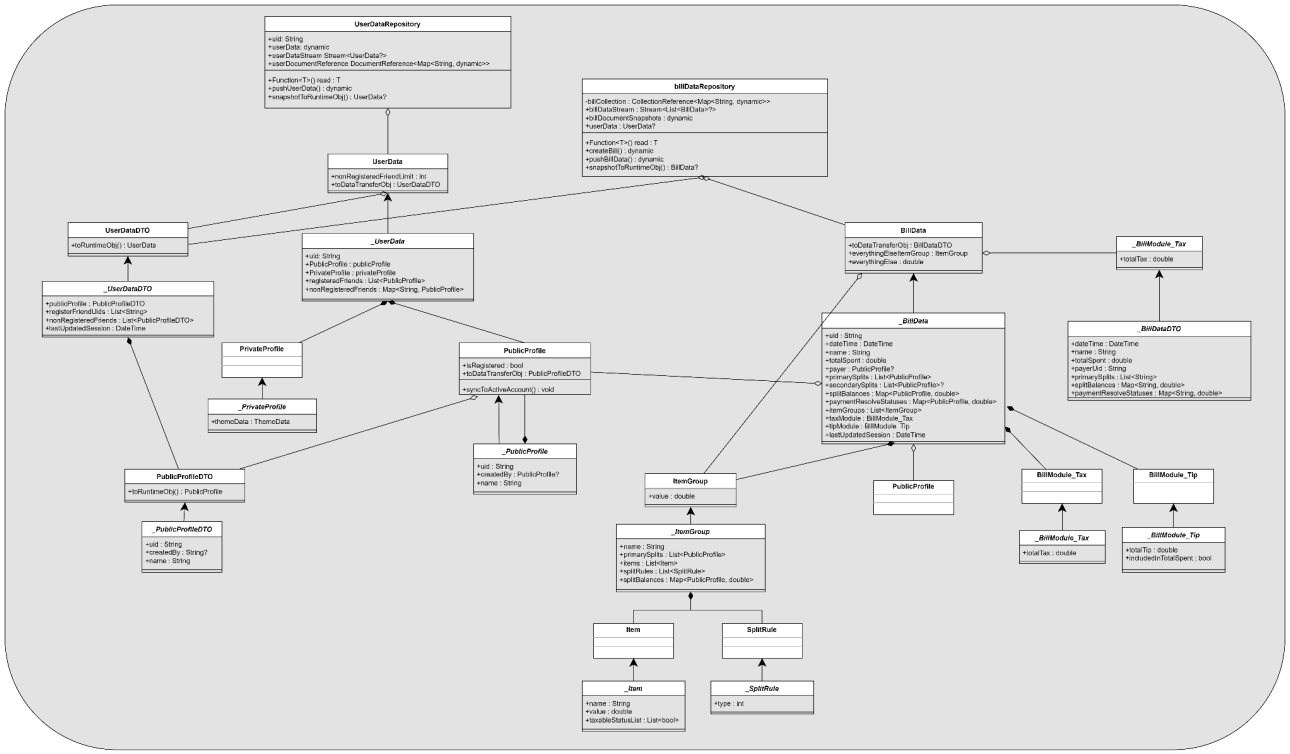

users > EYI1qyiN4fhjeb...
More in Google Cloud
(default)
users
EYI1qyiN4fhjeb7U6nwt4W5J6d42
Start collection
Add document
Start collection
bills
users
ArhvNv8xjSTFzRrKvB9P3CbKTGN2
EP7DBDKVIENAG5hPdY2L1aAuLrH3
EYI1qyiN4fhjeb7U6nwt4W5J6d42
u1OnBI6DKBPtHHxqGTD1WLYIEFg2
Add field
lastUpdatedSession: "2023-10-30T06:25:10.925892"
nonRegisteredFriends
0
createdBy: "EYI1qyiN4fhjeb7U6nwt4W5J6d42"
name: "NPC #0"
uid: "4ac44b00-c34f-1d68-aa4b-d5683be49bdf"
1
createdBy: "EYI1qyiN4fhjeb7U6nwt4W5J6d42"
name: "Alex Liu"
uid: "2ca768c0-147d-1d6f-984f-61ac7fb3d97d"
publicProfile
createdBy: null
name: "Duke. wxyz"
uid: "EYI1qyiN4fhjeb7U6nwt4W5J6d42"
registeredFriendUids


bills > 6itlfM89MQ3Es..
More in Google Cloud
(default)
bills
6itlfM89MQ3Esdumr1Wz
Start collection
Add document
Start collection
bills
users
6itlfM89MQ3Esdumr1Wz
CdrHXim7C3k4B4jKnrSW
D3ICuDkaMsCfnBitGDmv
GnZMFLgigmGUK1CAllPY
LSCnjINKv9uYHzGb9fqM
LxWwiXr8CgNo9acqpKMR
N7BsCrjxslxiTcreMoE0
Ve58NO0IpagmdNj9uZee
WeOEvOaWg6MlojzXkm1N
bUwg2AnfARsoUXPCvsF8
Add field
dateTime: "2023-11-15T00:00:00.000"
lastUpdatedSession: "2023-11-15T18:08:10.421219"
name: "cake"
payerUid: "VjjTvjm3z8an6KGfDJ09AVu5A2J2"
primarySplits
0 "eb6e0540-1897-1d72-b7f8-193d78408755"
1 "cbff1180-fd41-1d84-849c-f72e6136aab1"
2 "c5e326a0-9cbb-1d9e-aab6-f582f4fb4f83"
totalSpent: 99

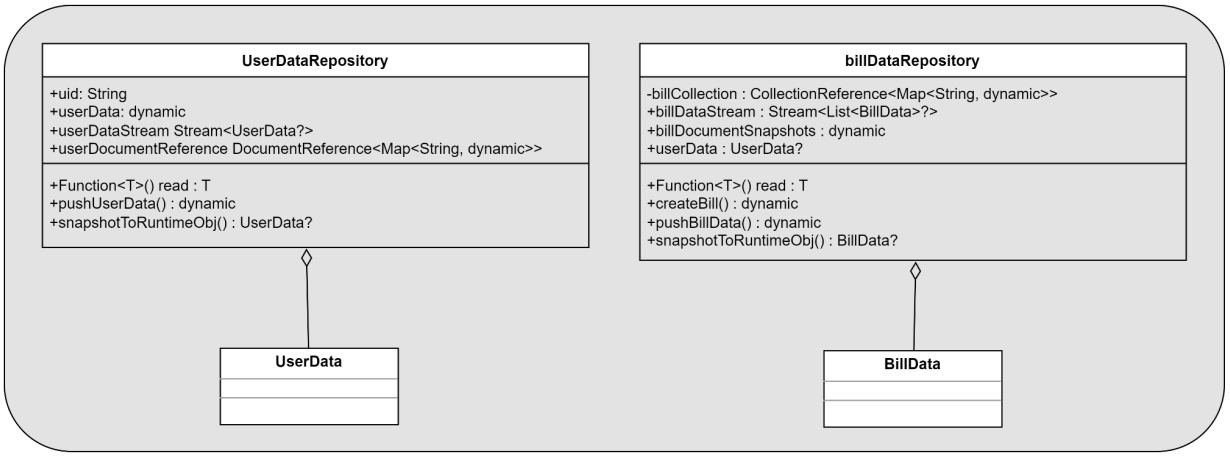
Section 5 - Software Domain Design

5.1 Software Application Domain Chart

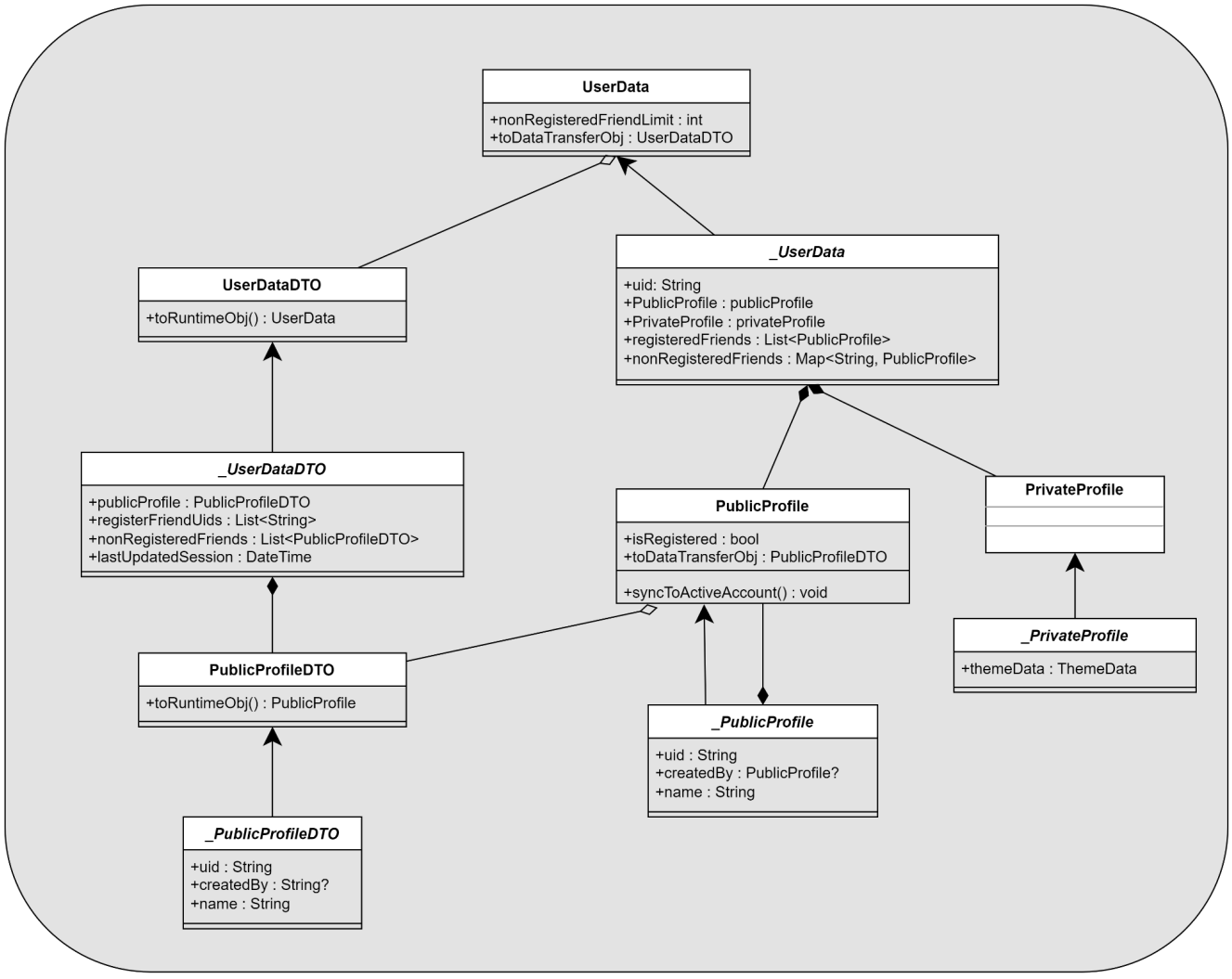
The first image below is the overall UML diagram and the following images are a closer look into each part.



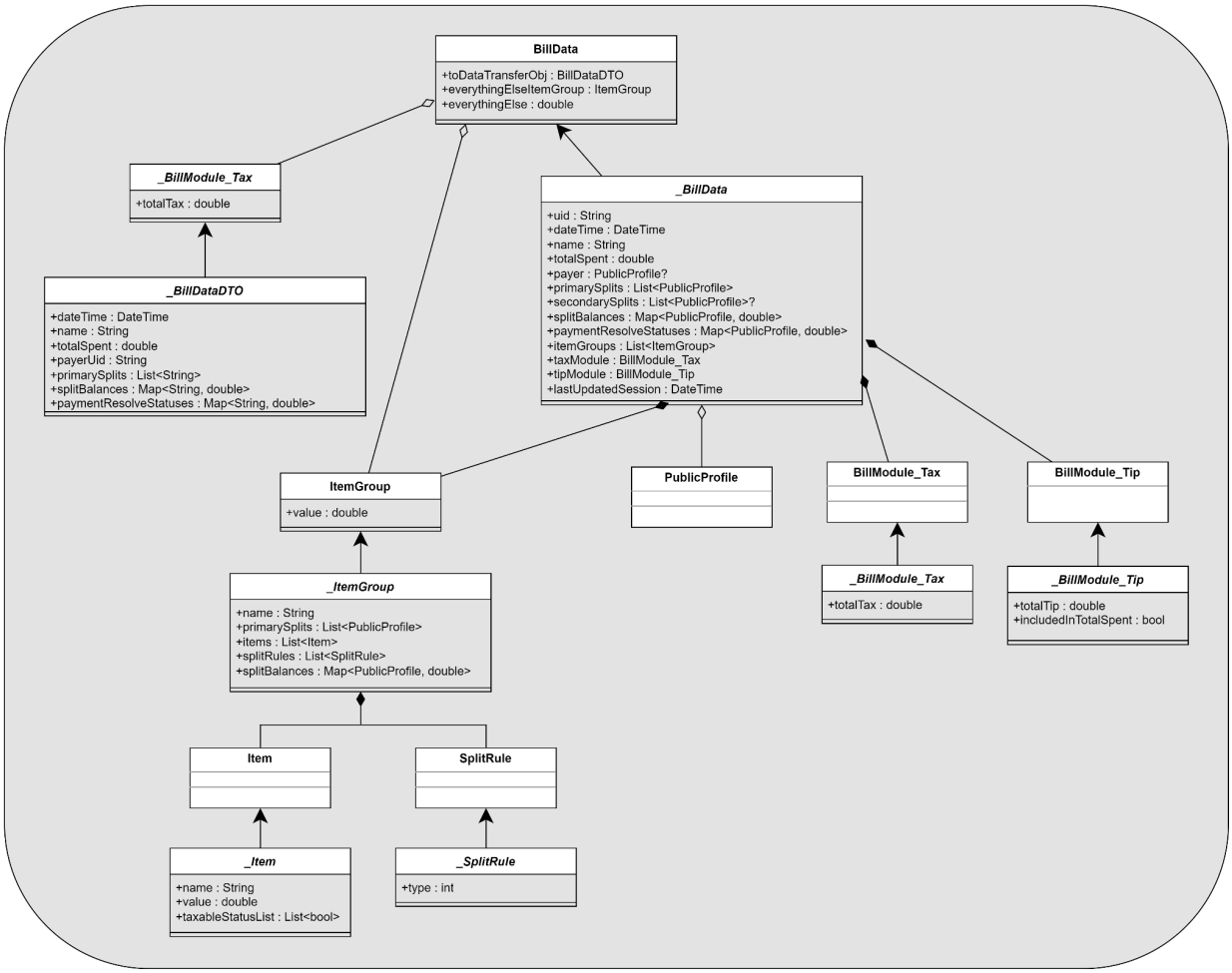
These 2 classes serve as boundary classes that communicate with Cloud Firestore to fetch data from the database. The UserDataRepository is responsible for UserData objects whereas the BillDataRepository is responsible for BillData objects. Both classes supply a real-time stream of data of their respective data, allowing fast change responses on devices without the need to submit fetch requests.



The UserData class is an entity class that handles user information such as their uid, name, and friends. UserData composes a PublicProfile and a PrivateProfile property, where the PublicProfile class serves as a data packet containing only the user's publicly visible information, without carrying excessive information on personal preferences. The UserData is also able to convert itself into a UserDataDTO object which is simplified for database storage.



The BillData class is an entity class that handles bill information such as its uid, name, date, total spent, and the payer. BillData is also composed of a list of item groups, a tax module, and a tip module property. BillData also keeps track of involved individuals of the bill through references of their PublicProfile in the primarySplits property. Similar to UserData, BillData is also able to convert itself into a BillDataDTO object which is simplified for database storage. In this data configuration, all references of PublicProfile are mapped into their uid, reducing the complex objects into strings.



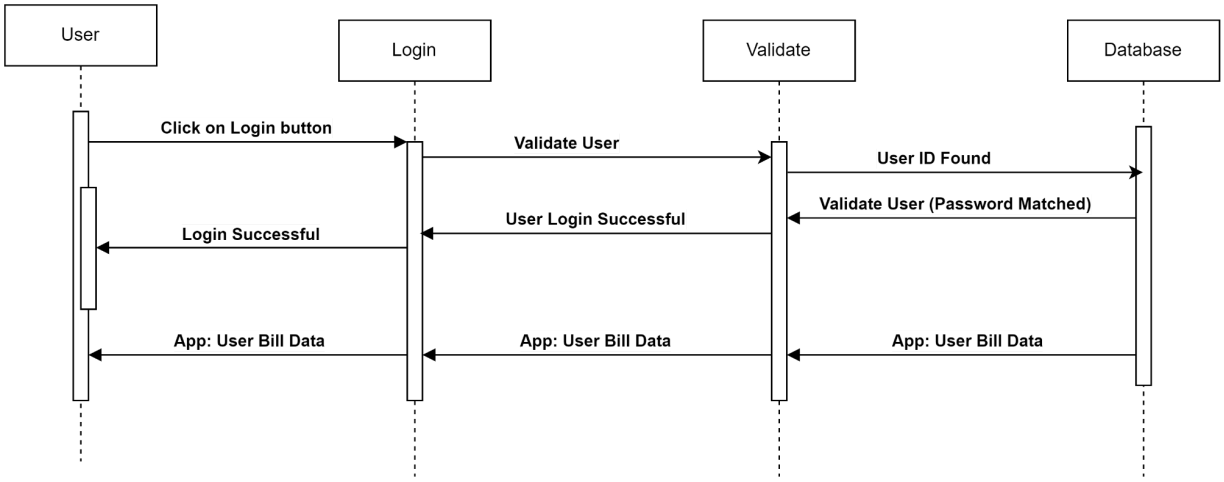
5.2 Software Application Domain

SplitIt mobile application ensures seamless navigation to the bills allowing users to effortlessly initiate bill splits or add friends to the list, thereby facilitating easy identification of participants in the expense-sharing process. The application is empowered by Firebase, incorporating a streamlined login and registration process within the app.

The bill data is connected to the Firebase where it keeps the user's transaction history so that the user may never forget about a bill or remember who participated in splitting the bill.

5.2.1 User Authentication

User authentication is currently achieved through the FireAuth API which connects the authentication service to the Firebase project, enabling access to the Cloud Firestore database.



5.2.1.1 User Registration

New user accounts can be created through FireAuth. Upon creation, Firebase will keep track of the user identifier, usually an email address, and the authentication provider. Upon creation, an uid will be designated to the user for identification within the system. FireAuth also ensures that the user identifier remains unique so that the same identifier cannot be reused to create another account. After registering, the application will route to the home page.

5.2.1.2 User Sign-In

The project currently only implemented email and password as the only sign-in providers. FireAuth is used to identify the sign-in credentials with existing user accounts in Firebase. After a successful login, the application will route to the home page.

5.2.2 Friend

Friends are categorized into registered friends and non-registered friends. Non-registered friends represent artificial personas that the user can create to be included in a bill-splitting session. This allows the split to proceed even if the friend is not a user of the application.

5.2.1.1 Friend Creation

Non-registered friends are created as a PublicProfile object which is then stored within a list in the user document on Firebase. Only a limited number of non-registered friends is allowed for each user, due to the storage constraints of a Firebase document, with the limit currently being 30.

5.2.3 Bill

Bills are stored as documents in the database under the bill collection with their own UID and data dictionary under bill data. This allows us to track any interactions the user might have with the bill. Each bill stores user uids for the payer and the involved individuals, as well as all the items specified that support the bill split calculation.

5.2.1.1 Bill Creation

Bills are primarily created in two ways, namely quick split and advanced split. Quick split offers the user a fast way to create a bill by just indicating the name, date, amount, and people involved in the bill. Advanced split allows the user to modify any bill down to the item level and makes it highly tailored to each user's need.

Section 6 – Data Design

6.1 Persistent/Static Data

6.1.1 Dataset

This application uses a no-SQL approach, and its data is dependent on 2 collections: users and bills. Each document in the collections is labeled with its uid (unique identifier) to allow search queries through uid reference. The no-SQL data structure supports conversion from the JSON format, where the document properties are related via maps with string keys.

6.1.2 Static Data

- **Unregistered Friend Limit (50)**: Each user can only create a maximum of 50 fake friends.
- **Username and User ID:** A fixed and unique user identification that is generated on account creation.
- **Emails and Passwords**

6.1.3 Persisted data

- **Friends**
- **Bills**
- **Bill Summary:** Transactional history of all bill-splitting activities, friends involved, name of bills, and bill date.

6.2 Transient/Dynamic Data

User

The user data consists of a public profile, a private profile, and a list of friends. The public profile contains common shareable data such as name and profile picture which are intended to be viewed by other users. The private profile will contain personal preferences such as app themes, which will not be intended for sharing.

6.3 External Interface Data

As of now, our application has no integration of external APIs. The app itself is currently utilizing the Dart SDK that accompanies the Flutter Framework with Firebase as the backend service.

6.4 Transformation of Data

Data classes that need to be uploaded to the online database through Firestore services have alternate data classes with the "DTO" suffix, as in *Data Transfer Object*. Runtime Data classes are able to convert into Data Transfer classes to simplify the data for storage, and the reverse operation exists too so that the stored data can be integrated more smoothly into the system. Complex object references are reduced to their uid to avoid redundant storage on the database, while still retaining the possibility to query a search to the data source through its uid.

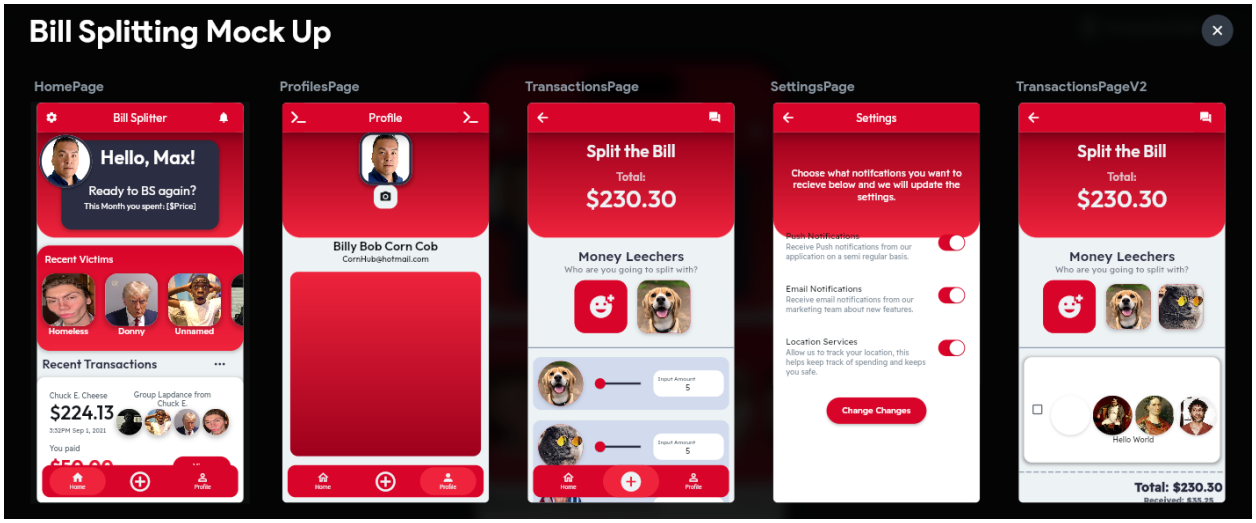
Section 7 - User Interface Design

7.1 User Interface Design Overview

All UI/UX designs (color, components, styling) follow the latest Material 3 design guidelines to ensure a smooth user experience. The home page follows a common interface layout featuring an app bar and a bottom navigation bar to enable a smooth transition of user familiarity with our app.

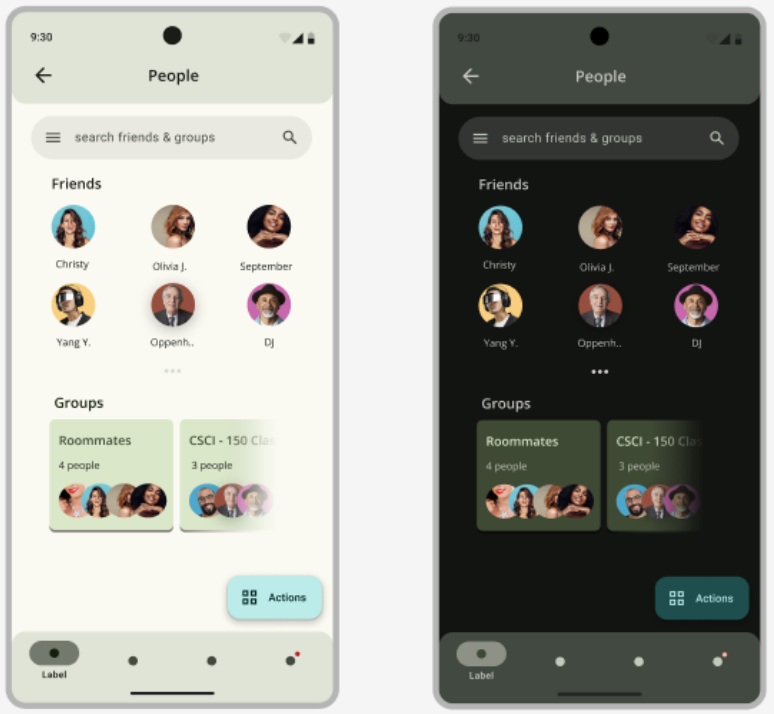
Mock-Up UI/UX Design

Initial mock-ups were designed to provide visibility on interface feasibility and considerations for additional requirements. These designs are made corresponding to an early requirement discussion.

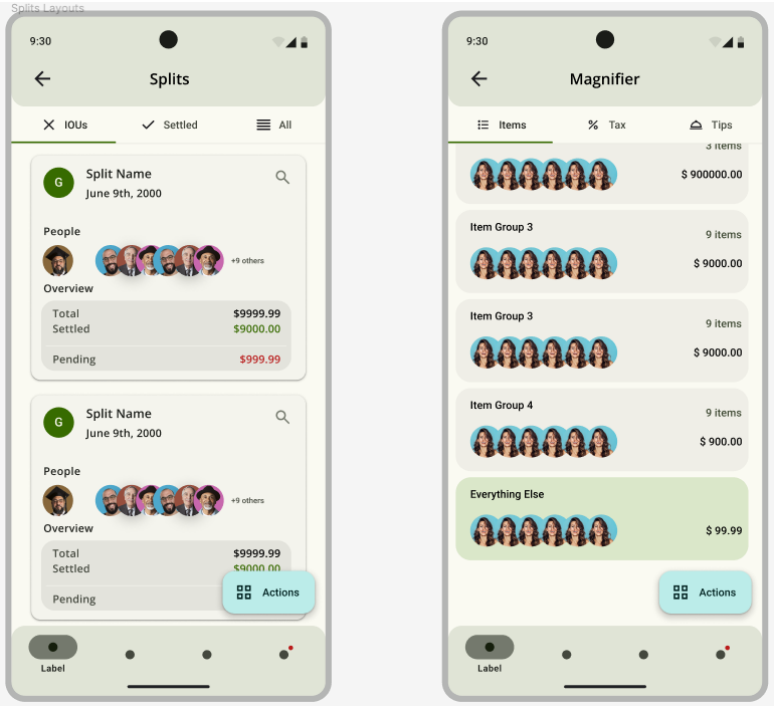


Finalized UI/UX Design
link: splitit figma file

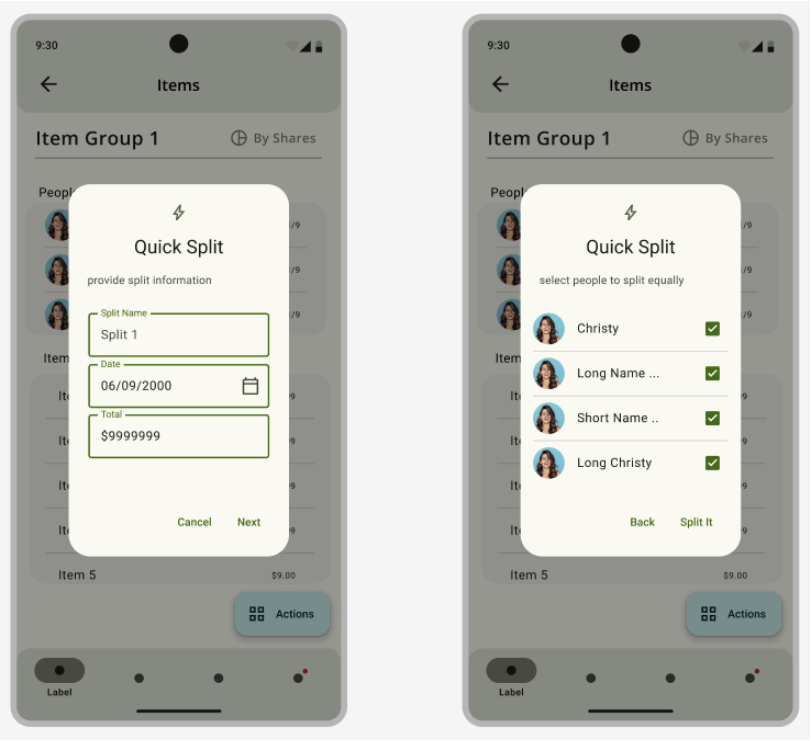
People page (friends):



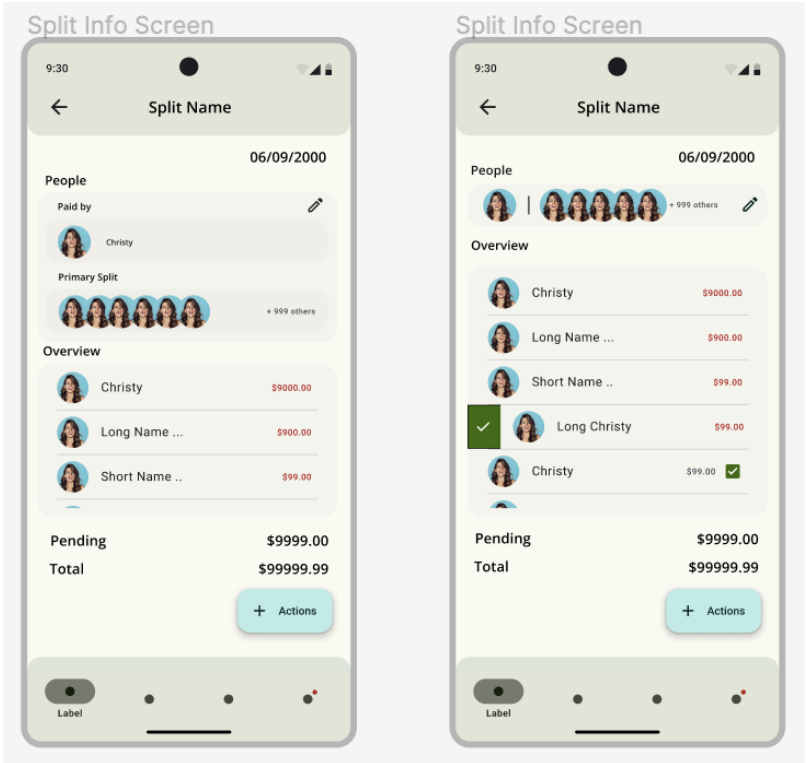
Splits page (bills):



Quick split dialog(pop-up):



Detailed Bill View:



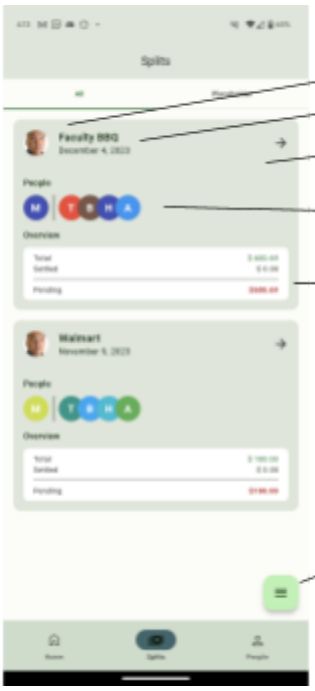
Home
(Landing page)

Top App Bar
Logout button (for developer & testing)
Username
User profile picture
User quote
Add friend button (lets the user add a friend instantly
Split bill button (lets the user input split details)
Add group button (lets the user create a group instantly
See all transactions (goes to bills page)
Transaction summary (goes to the "transaction's" page)
Home button (goes to home page)
Bills button (goes to bills page)
People button (goes to people/friends page)
Bottom Nav Bar

Quick Start
(Shortcuts)

Recent Transactions
(Quick View of Bills)

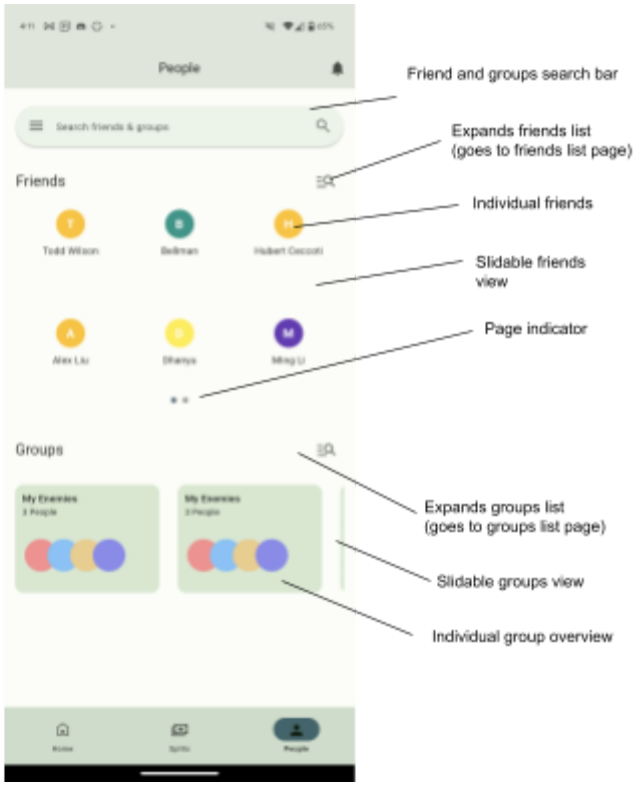
All Transaction History
(scrollable list)

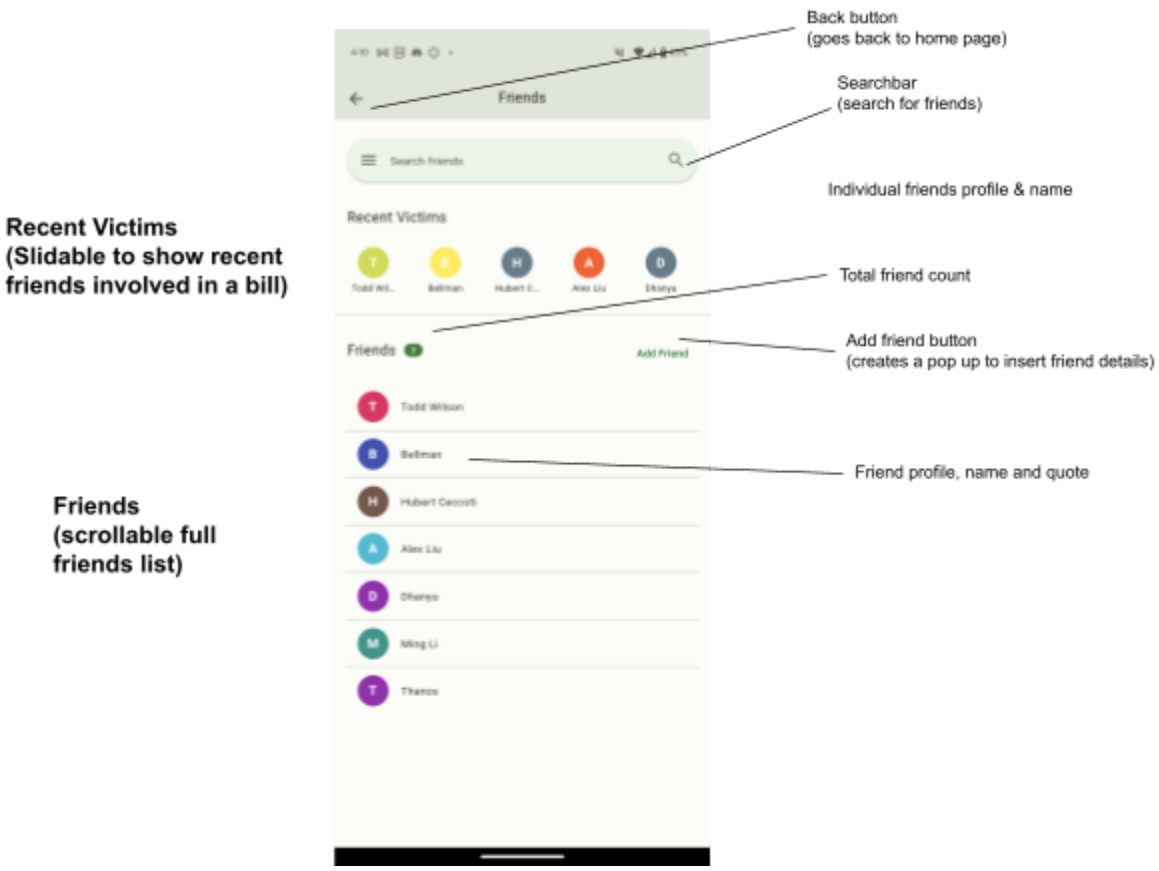


Bill photo
Split name and Date
Clickable bill card (goes to "transaction's" page)
People involved
Summary of Split Status
Floating action button (create bills)

Friends Section

Groups Section

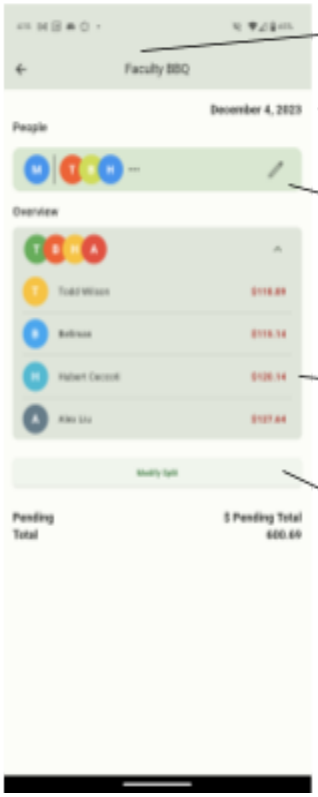



Back button
(goes back to home page)
Searchbar
(search for friends)
Individual friends profile & name
Recent Victims
(Slidable to show recent friends involved in a bill)
Total friend count
Add friend button
(creates a pop up to insert friend details)
Friends
(scrollable full friends list)
Friend profile, name and quote
Friends
Recent Victims
Friends
Add Friend
Todd Wilson
Bellman
Hubert Cecotti
Alex Liu
Dhanya
Ming Li
Thanos

People
(shows people involved)

Overview
(Shows how much each people owes)

Summary
(Shows how much is pending, paid, total)



Bill Name

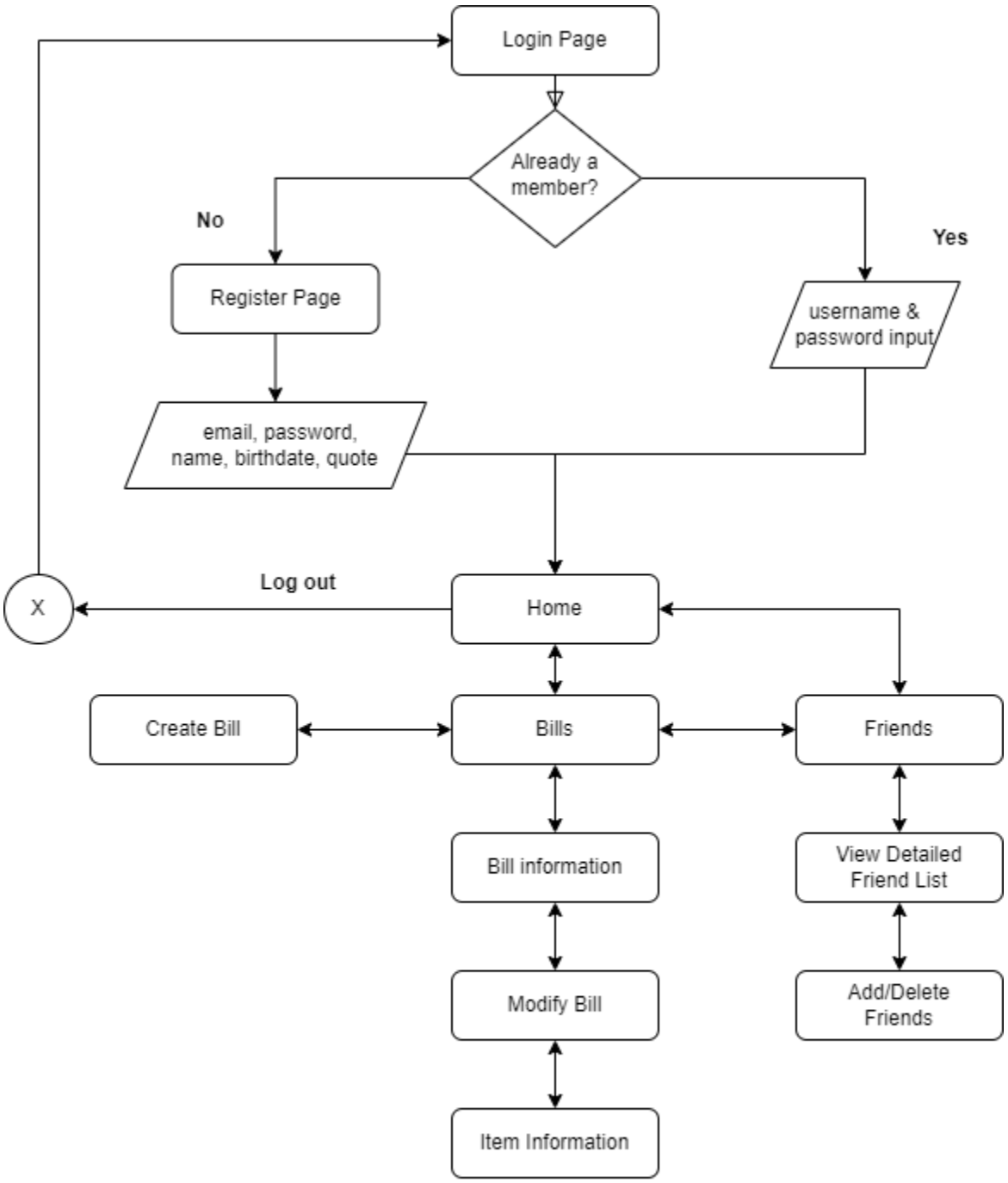
Bill Date

Edit button
(goes to people manage page)

Amount owed for each person

Navigates to advanced bill editing

7.2 User Interface Navigation Flow



User Navigation Flow:

On the first entry to the application, the user is directed to the login page. The user is given the choice of creating an account (Registration Page) or logging in (Providing email and password to be authenticated). Following the authentication process, the user is routed to the Home Page. From there, they could switch between 3 main pages: the home page, the splits page, or the people page.

People Page:

Once on the people page the user can view the list of friends or groups that were already created as well as the capability to add or delete friends to the application.

Bill Page:

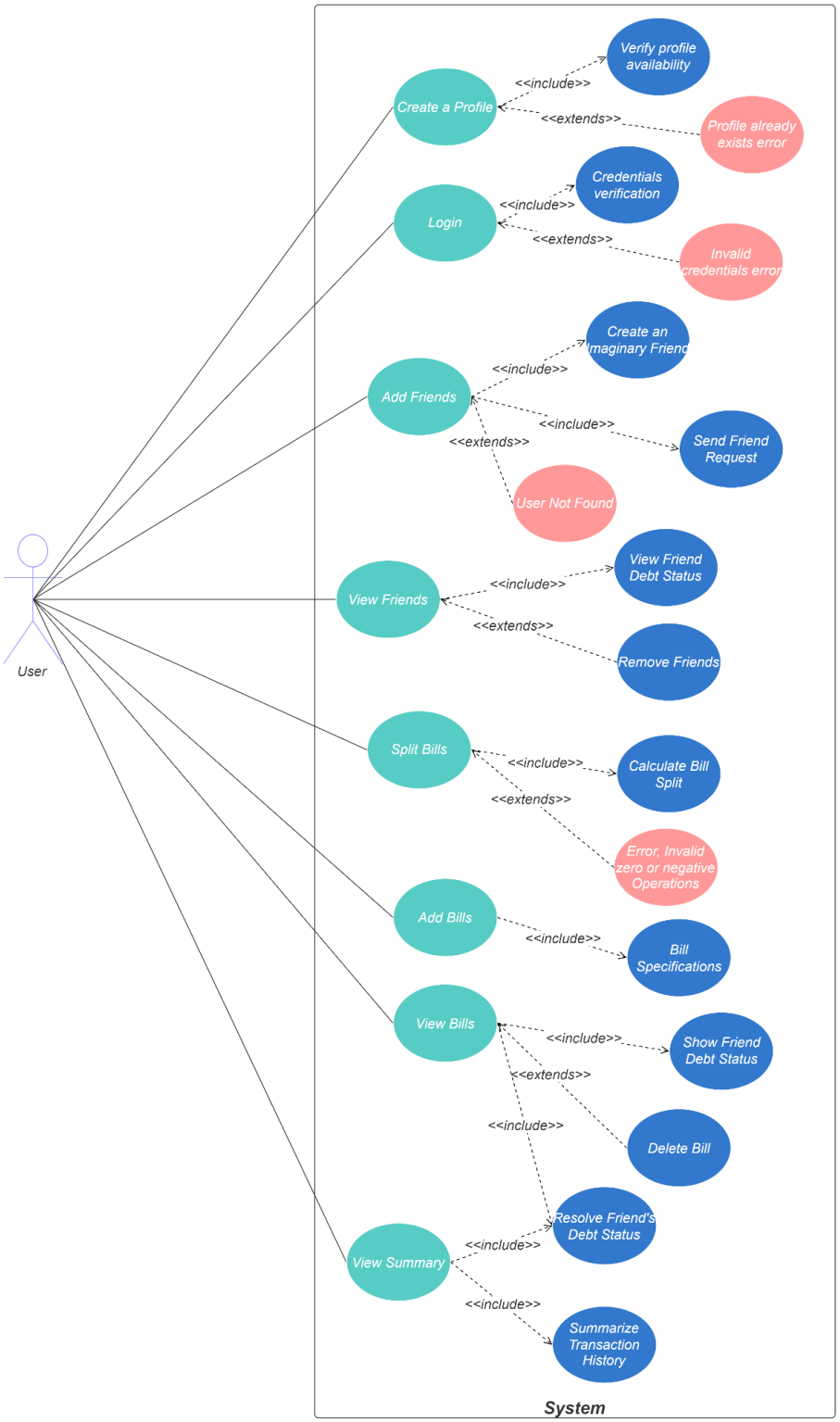
The bill page displays the history of the bills sorted by recency as well as a floating action button for the user to create a new bill to do a quick split. Each bill record is displayed as a bill card, which contains the name of the bill, the date it was created, the total cost, and a summary of the amount settled. The user would also see the individuals from the friend list that are participating in splitting the bill. Once the user presses on the bill they would be taken to the bill information page which will show more details, such as how much each person owes, and what items are in the bill.

Quick Split:

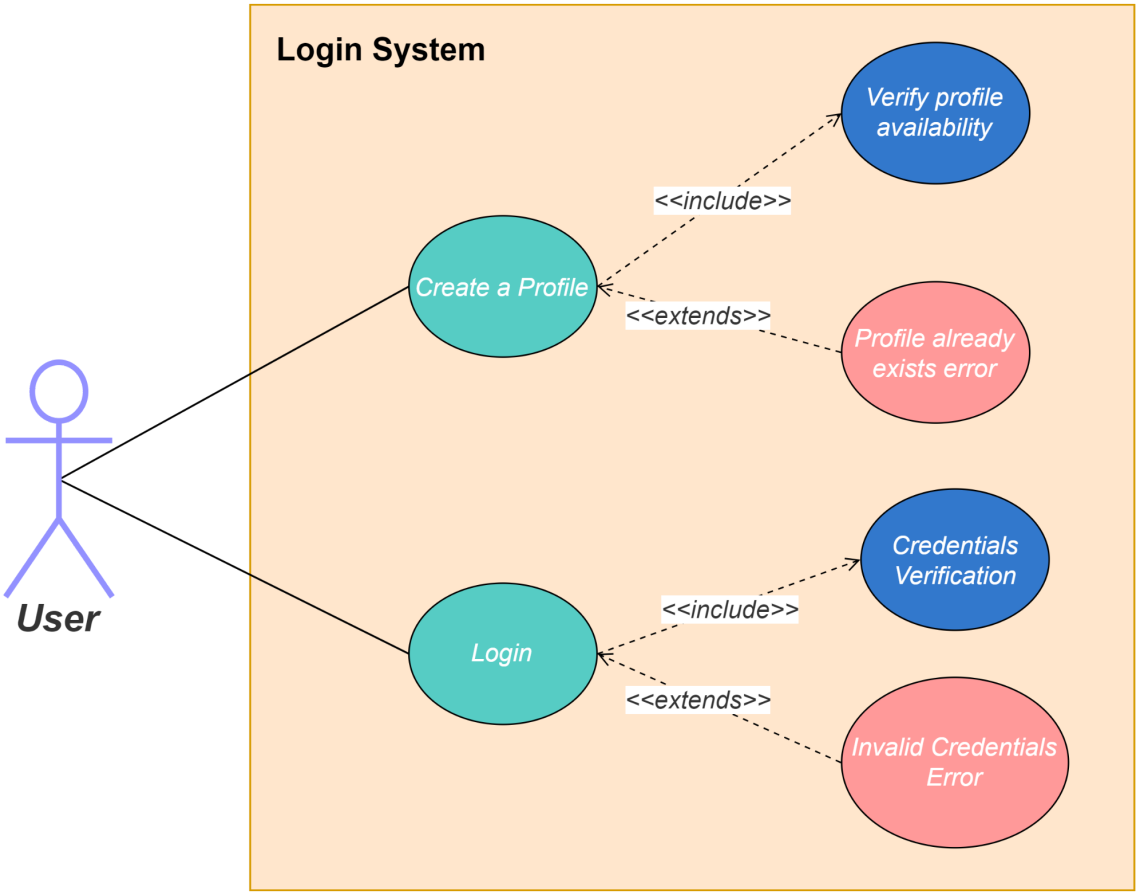
Through the quick split action, a pop-up dialog would appear where the user can fill out information for creating a new bill record, where such information includes the name, the date, as well as the total money spent. After that, the user will then select friends who are partaking in the split via a list of checkboxes.

7.3 Use Cases / User Function Description

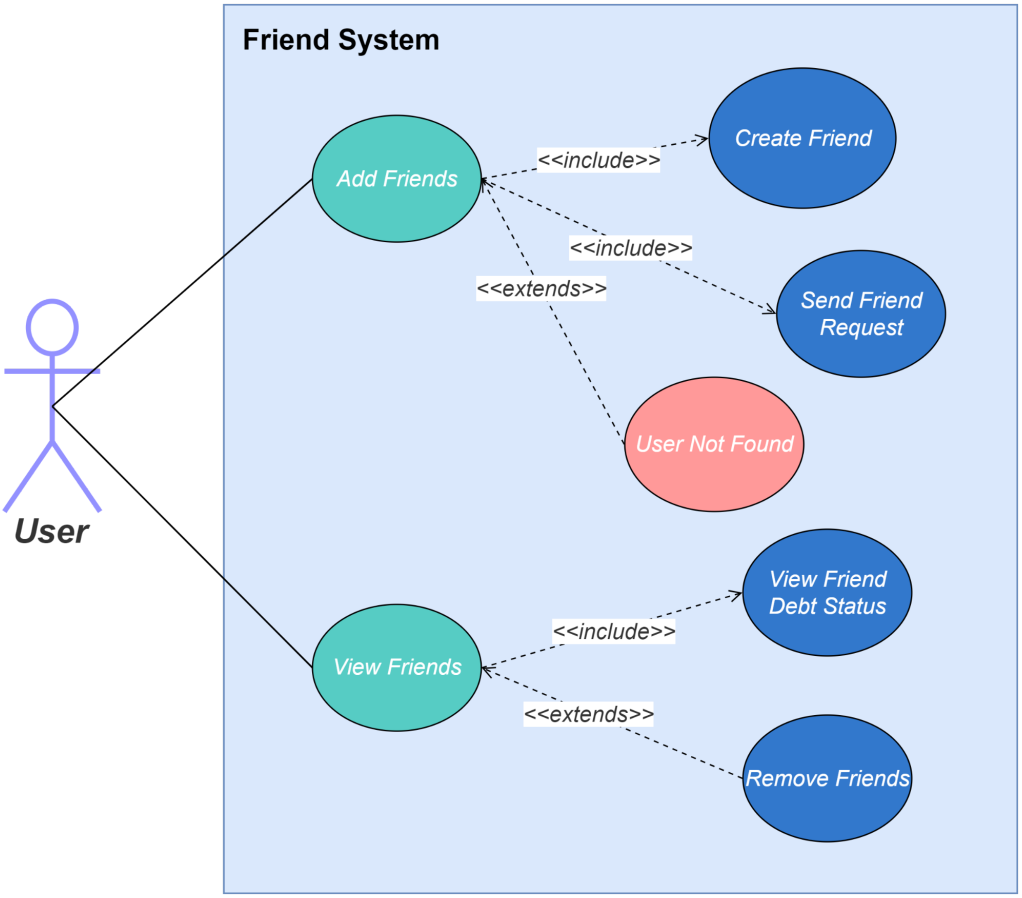
Describe screen usage/function using use cases, or on a per-function basis



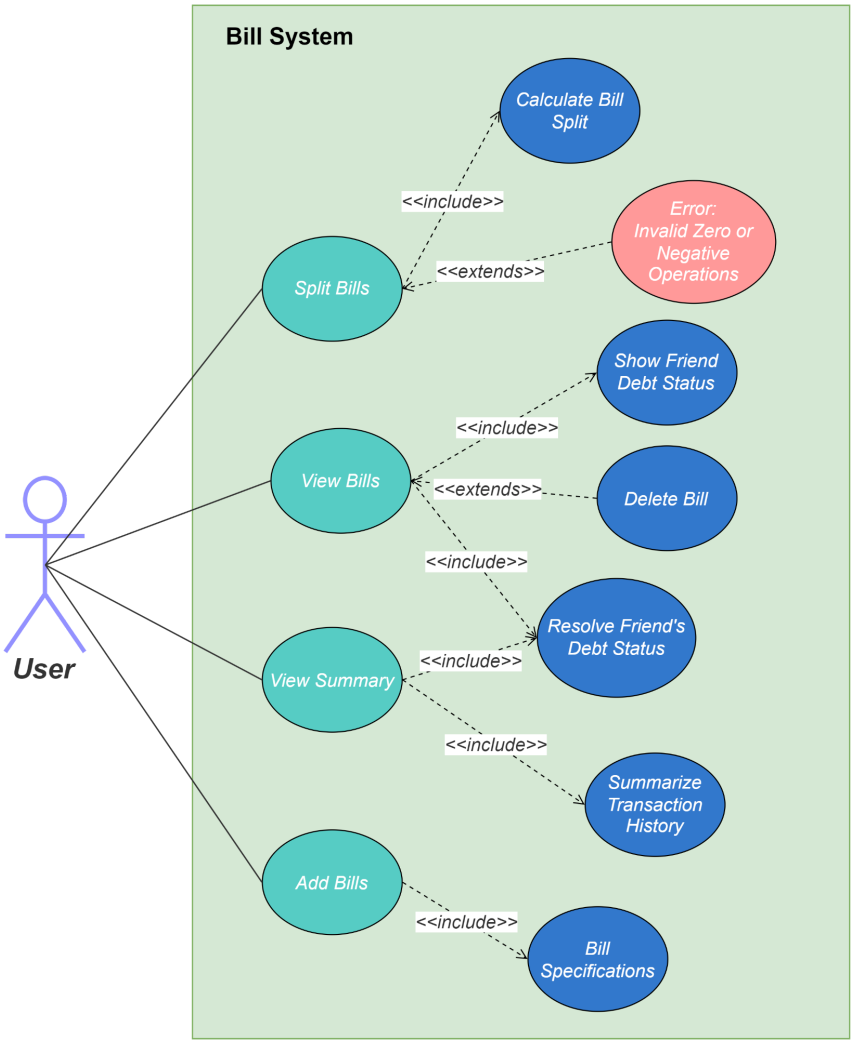
Use Case:



- **Sign-Up/Login:**
  - Profile Creation: Verifies that the account to be created is available. If not, then an error is displayed indicating that the account already exists.
  - Login: Authenticates the profile (Email and Password information).



- **Friends:**
  - Adding Friends: Creates an unregistered friend or sends a friend request.
  - View Friends: Scrolling through the friends list, with information such as debt status and recent transactional involvements displayed. Additionally, the option to remove the friend is available.



- **Bills:**
  - Bill Splitting: Calculates the bill-splitting based on the given parameters. If the operation is invalid, then an error is displayed.
  - View Bills: Showcases the bill information such as the total, amount pending from involved friends, and bill deletion option.
  - View Summary: Displays the transaction history of all interactions with the application, and the ability to resolve any pending payments from friends.

Section 8 - Other Interfaces

8.1 Firebase

This project will connect to a Firebase project running on the Spark billing plan.

8.1.1 FireAuth:

The FireAuth feature of Firebase provides secure user authentication services and supports various providers for signing in such as email and password, Google, Microsoft, Facebook,

Apple, etc.

8.1.2 Cloud Firestore:

Cloud Firestore offers a lightweight no-SQL database service with real-time data stream, enabling fast and responsive data changes on active listeners.

8.2 Flutter/Dart Packages:

Flutter/Dart dependencies, packages available through pub.dev:

- freezed_annotation: ^2.4.1
- json_annotation: ^4.8.1
- firebase_core: ^2.16.0
- firebase_auth: ^4.10.0
- provider: ^6.0.5
- cloud_firestore: ^4.9.2
- flutter_bloc: ^8.1.3
- flutter_svg: ^2.0.7
- firestore_cache: ^2.3.1
- material_symbols_icons: ^4.2706.0
- intl: ^0.18.1
- smooth_page_indicator: ^1.1.0
- flutter_slidable: ^3.0.0
- uuid: ^4.1.0
- flutter_expandable_fab: ^2.0.0
- flutter_material_symbols: ^0.0.4
- expansion_tile_card: ^3.0.0
- assorted_layout_widgets: ^8.0.5

Section 9 - Extra Design Features / Outstanding Issues

- The Flutter SDK has yet to accurately follow the Material 3 design guidelines with default widget properties; i.e: incorrect icons, incorrect colors. Requires a workaround solution or wait for a Flutter update that resolves this issue.
- The friend feature is currently restricted to user-generated friends only, connecting with active users will introduce additional logic complexity for shared data.
- Introduce additional sign-in providers
- Improve user experience
- Eliminate bugs

Section 10 – References

SplitIT Github Repo

SplitIT Figma File

Section 11 – Glossary

Glossary of terms/acronyms

DTO: Data Transfer Object
Flutter: Open-Source Software Development Framework.
Dart: Programming language used to develop Web and Mobile applications.

BS: Bill Splitting
SDK: Software Development Kit

JSON: JavaScript Object Notation
OS: Operating System
UI: User Interface
BS: Bill Splitting
HTTPS: HyperText Transfer Protocol Secure.