

.Section 1 - Project Description

1.1 Project

The project name: SplitIt

1.2 Description

Brief overall description of the project:

Create and manage a group of people: allow users to create a group within the application to split the bill accordingly, ability to add, remove members. The ability to track bills and allow users to track any pending bills that are yet to be spit/resolved. User friendly, allows users to quickly split their bills based on their rules or with templates.

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/adding to multiple sections | Sydney Khiev Ren Hao Wong Brandon Becerra Zhen Wei Ng Roberto Cruz |
| | | |
| | | |
| | | |

Contents

[Section 1 - Project Description](#)

[1.1 Project](#)

[1.2 Description](#)

[1.3 Revision History](#)

[Section 2 - Overview](#)

[2.1 Purpose](#)

[2.2 Scope](#)

[2.3 Requirements](#)

[2.3.1 Estimates](#)

[2.3.2 Traceability Matrix](#)

[Section 3 - System Architecture](#)

[Section 4 - Data Dictionary](#)

[Section 5 - Software Domain Design](#)

[5.1 Software Application Domain Chart](#)

[5.2 Software Application Domain](#)

[5.2.1 Domain X](#)

[5.2.1.1 Component Y of Domain X](#)

[5.2.1.1.1 Task Z of Component Y1 of Domain X](#)

[Section 6 – Data Design](#)

[6.1 Persistent/Static Data](#)

[6.1.1 Dataset](#)

[6.1.2 Static Data](#)

[6.1.3 Persisted data](#)

[6.2 Transient/Dynamic Data](#)

[6.3 External Interface Data](#)

[6.4 Transformation of Data](#)

[Section 7 - User Interface Design](#)

[7.1 User Interface Design Overview](#)

[7.2 User Interface Navigation Flow](#)

[7.3 Use Cases / User Function Description](#)

[Section 8 - Other Interfaces](#)

[8.1 Interface X](#)

[Section 9 - Extra Design Features / Outstanding Issues](#)

[Section 10 – References](#)

[Section 11 – Glossary](#)

Section 2 - Overview

2.1 Purpose

Brief description of the focus of this module of the overall project and its intended audience.

Intended audience:

Anyone who needs to split a bill on a regular basis like family, friends, parties, couples, roommates, students.

The product is intended to serve the people, the people who gather together or even the people who are alone and want to split their expenses among themselves.

2.2 Scope

Describe the scope of the module to be produced

Benefits: Providing user 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.

2.3 Requirements

Your mileage may vary -- we typically break down the requirements to provide a ballpark estimate.

Design Requirements:

- Intuitive and visually appealing navigation for the app.
- Responsive and smooth usage throughout the app
- User data is protected

Graphic Requirements:

- Graphic demands on device should be low to ensure performance
- Appealing and easy to understand graphics
- 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:

- Product meets target audiences expectations
- Compliant with Android and IOS regulations
- Developed prototype within time frame of 11 weeks

2.3.1 Estimates

| # | 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 | 2 |
| 12 | UX Design | 2 |
| | 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

Describe/include a figure of the overall system architecture (and where this module fits in)






Section 4 - Data Dictionary

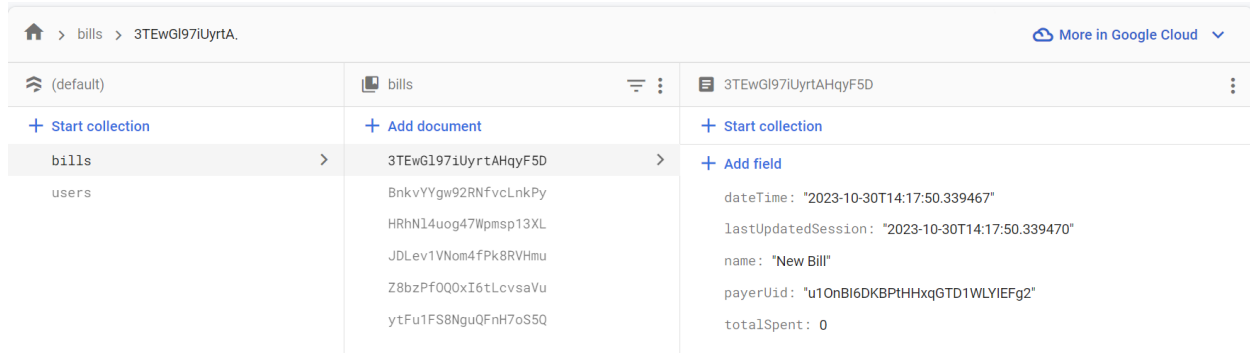
Brief description of each element in this module or a link to an actual data dictionary

(template of a database table description)

| Table | | |
|-------|--|--|
|-------|--|--|

| Field | Notes | Type |
|-------|----------------------------------|---------|
| ID | Unique Identifier from TABLE SEQ | DECIMAL |
| | | |
| NAME | The Name in Object.Name() | VARCHAR |
| VALUE | The Value output from somewhere | VARCHAR |

| | | | |
|---|--|--|--|
|  > users > EYI1qyiN4fhjeb... | | |  More in Google Cloud |
|  (default) |  users |  EYI1qyiN4fhjeb7U6nwt4W5J6d42 | |
| + Start collection | + Add document | + Start collection | + Add field |
| <div>bills</div> <div>users ></div> | <div>ArhvNv8xjSTFzRrKvB9P3CbKTGN2</div> <div>EP7DBDKVIENAGShPdY2L1aAuLrH3</div> <div>EYI1qyiN4fhjeb7U6nwt4W5J6d42 ></div> <div>u10nBI6DKBPTtHxqGTD1WLYIEFg2</div> | <div>lastUpdatedSession: "2023-10-30T06:25:10.925892"</div> <div>nonRegisteredFriends</div> <div>0</div> <div>createdBy: "EYI1qyiN4fhjeb7U6nwt4W5J6d42"</div> <div>name: "NPC #0"</div> <div>uid: "4ac44b00-c34f-1d68-aa4b-d5683be49bdf"</div> <div>1</div> <div>createdBy: "EYI1qyiN4fhjeb7U6nwt4W5J6d42"</div> <div>name: "Alex Liu"</div> <div>uid: "2ca768c0-147d-1d6f-984f-61ac7fb3d97d"</div> <div>publicProfile</div> <div>createdBy: null</div> <div>name: "Duke. wxyz"</div> <div>uid: "EYI1qyiN4fhjeb7U6nwt4W5J6d42"</div> <div>registeredFriendUids</div> | |

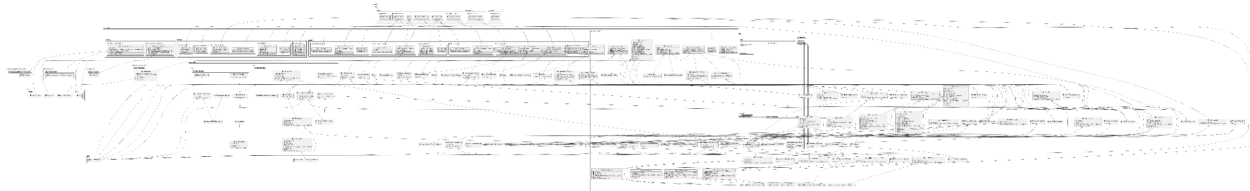


Section 5 - Software Domain Design

5.1 Software Application Domain Chart

Describe / chart each major software application domain and the relationships between objects (UML, etc)

Trust us, this is our UML diagram:



(Click link below to view full image)



[→ UML Class Diagram ←](#)

| Character | Icon for field | Icon for method | Visibility |
|-----------|----------------|-----------------|-----------------|
| - | □ | ■ | private |
| # | ◇ | ◆ | protected |
| ~ | △ | ▲ | package private |
| + | ○ | ● | public |

5.2 Software Application Domain

A Comprehensive high level description of each domain (package/object wherever it is better to start) within the scope of this module (or within the greater scope of the project if applicable)

5.2.1 Domain X

A high level description of the family of components within this domain and their relationship. Include database domain, stored procedures, triggers, packages, objects, functions, etc.

5.2.1.1 Component Y of Domain X

Define Component Y, describe data flow/control at component level

5.2.1.1.1 Task Z of Component Y1 of Domain X

Define Task Z, describe data flow/control at task level

Section 6 – Data Design

Describe the data contained in databases and other shared structures between domains or within the scope of the overall project architecture

6.1 Persistent/Static Data

Describe/illustrate the logical data model or entity relationship diagrams for the persistent data (or static data if static)

6.1.1 Dataset

Describe persisted object/dataset and its relationships to other entities/datasets

6.1.2 Static Data

Unregistered Friend Limit (30)

Each user can only create a maximum of 30 fake friends.

6.1.3 Persisted data

Describe persisted data

6.2 Transient/Dynamic Data

User

The user data consist 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 theme, which will not be intended for sharing.

6.3 External Interface Data

Any external interfaces' data goes here (this is for the data, section 8 is for the interface itself)

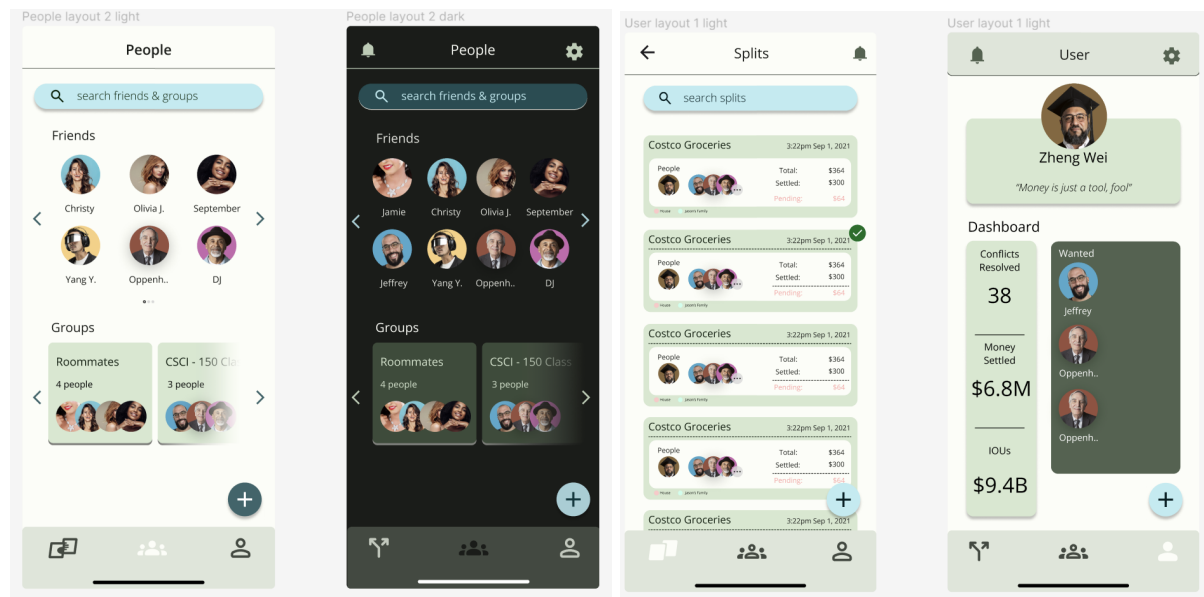
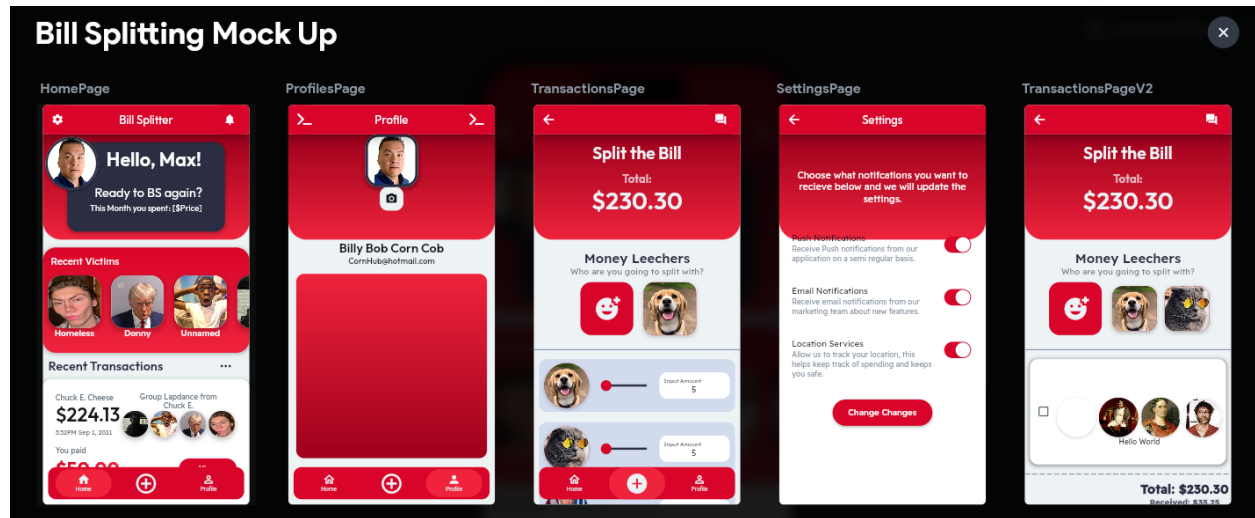
6.4 Transformation of Data

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.

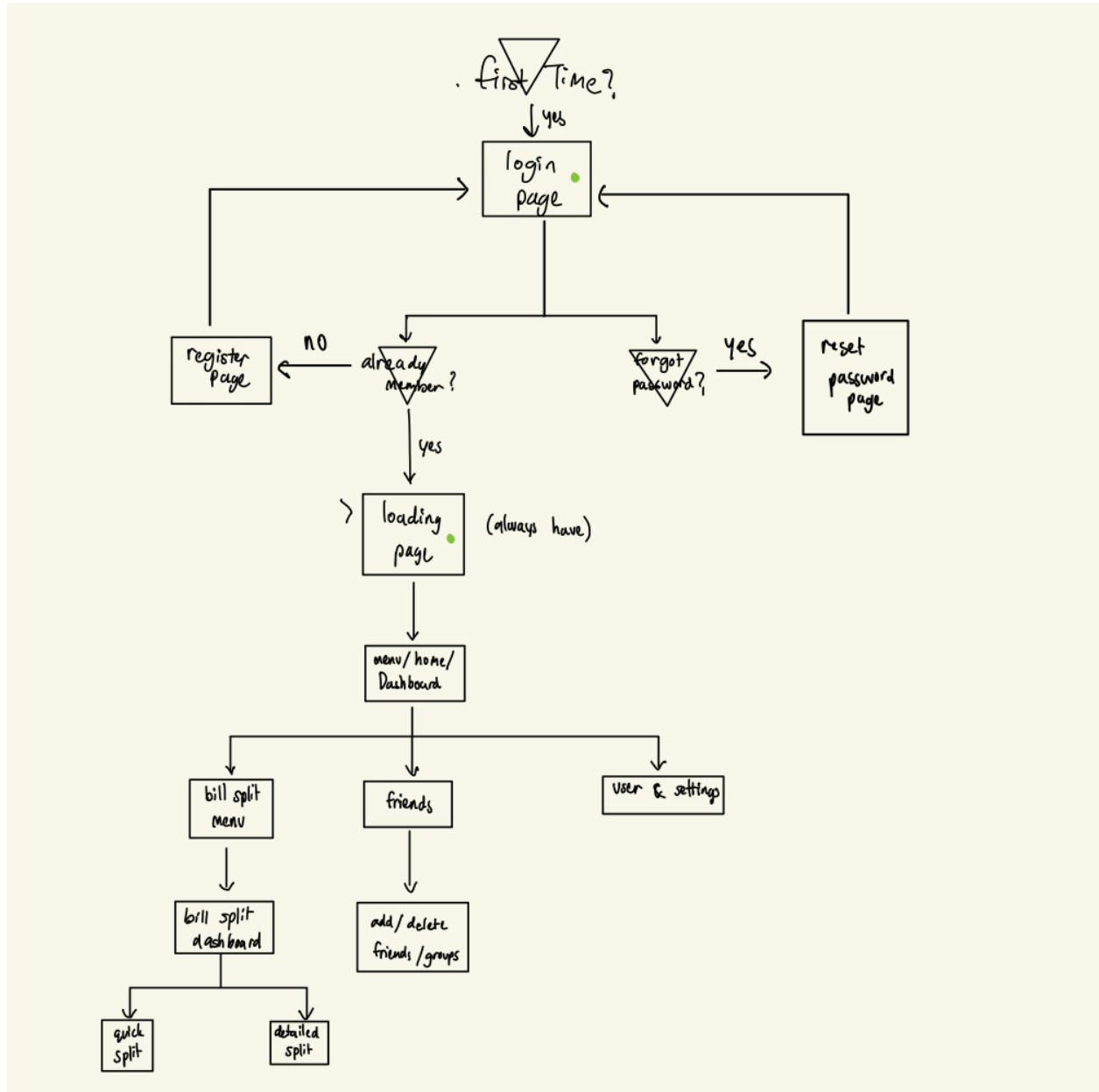
Section 7 - User Interface Design

7.1 User Interface Design Overview

Pictures, high level requirements, mockups, etc.

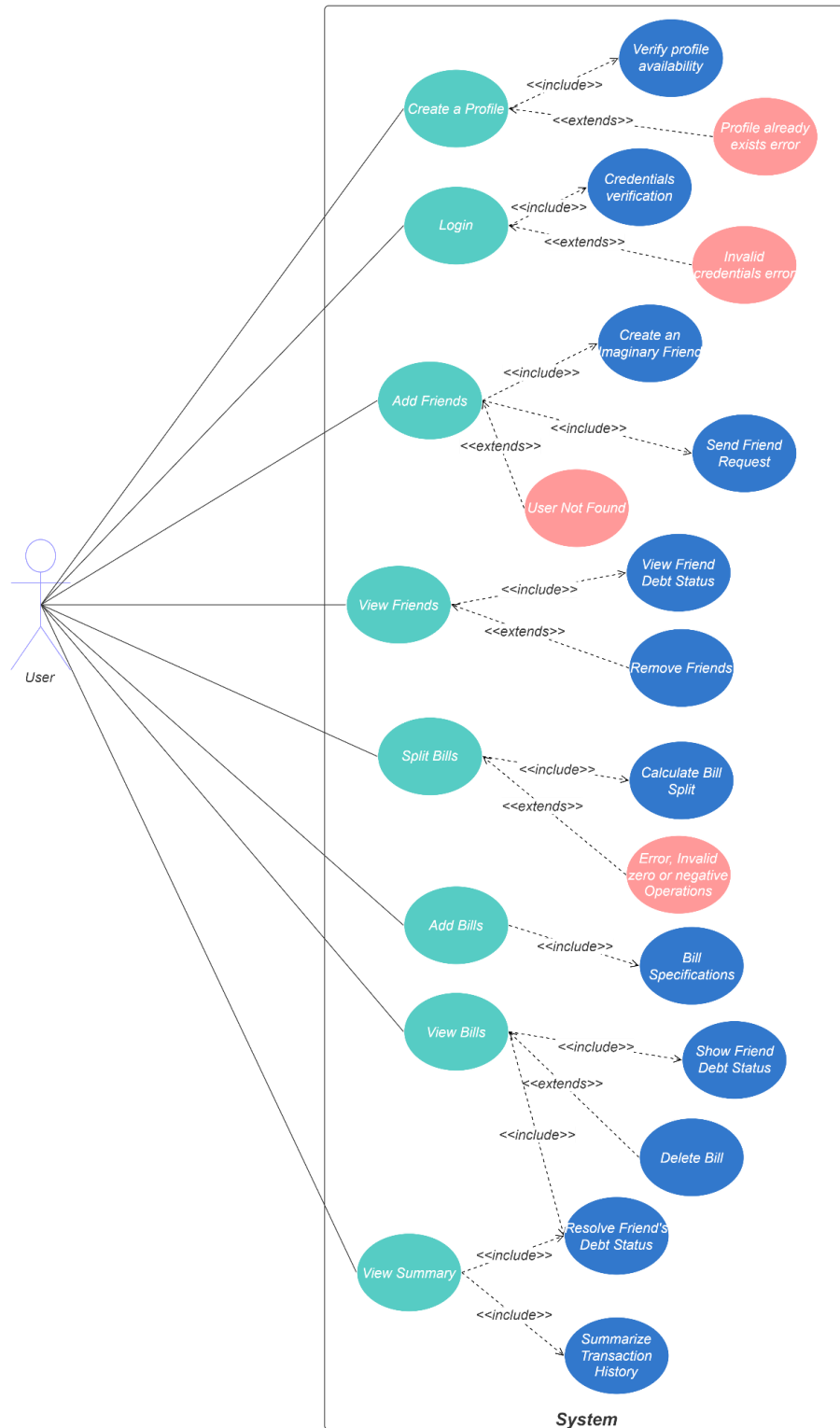


7.2 User Interface Navigation Flow



7.3 Use Cases / User Function Description

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



Section 8 - Other Interfaces

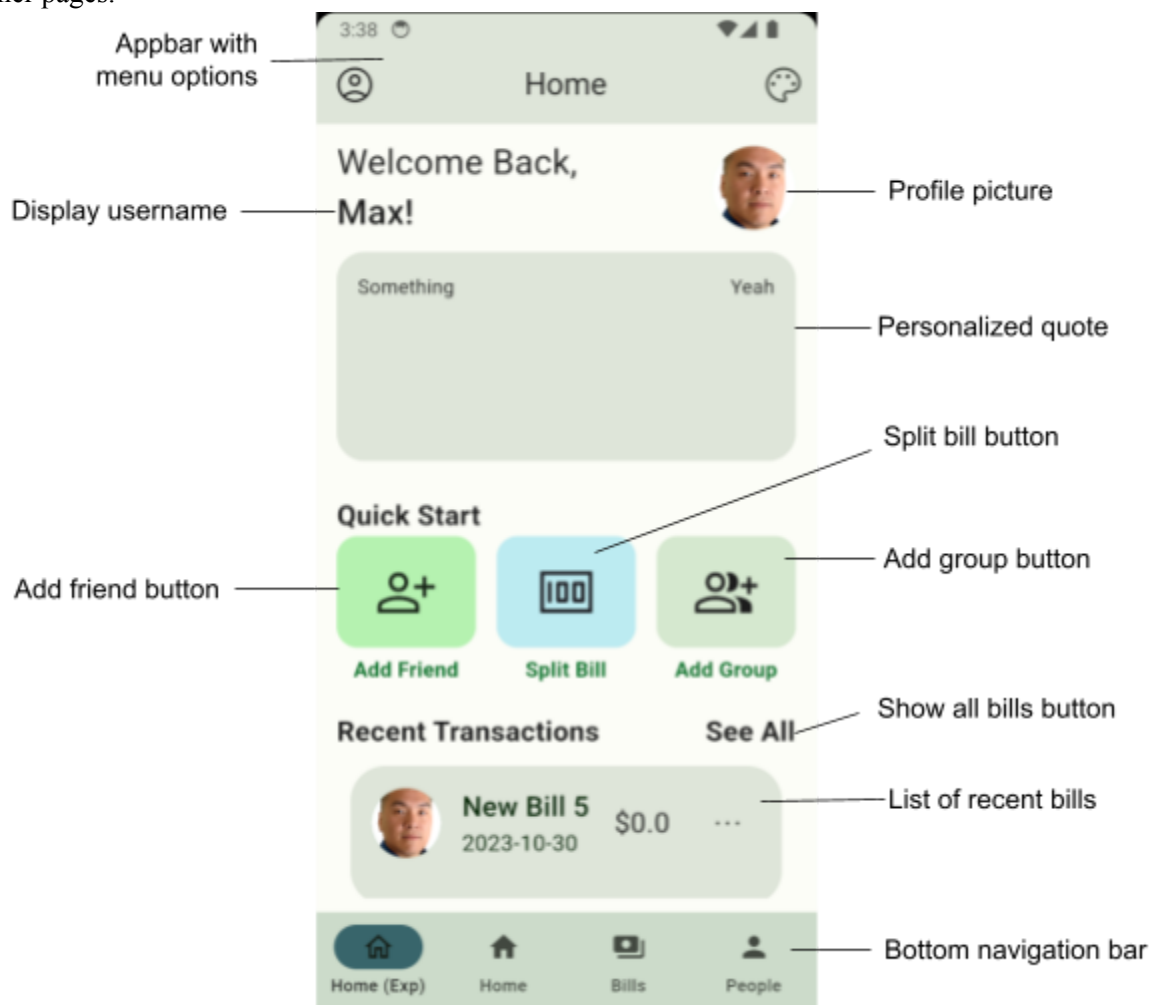
Identify any external interfaces used in the execution of this module, include technology and other pertinent data.

8.1 Interface X

Describe interactions, protocols, message formats, failure conditions, handshaking, etc.

8.1.1 Interactions:

The interactions will include adding friends, searching in the friend list, scrolling through the bills, making new bills, and changing between screens of home, bills, and people. Users are able to navigate through the application by either using the bottom navigation bar or the directive text buttons that route to other pages.



8.1.2 Protocols:

Firestore Authentication and Firestore Database (Backend)

Dart Packages (freezed_annotation, json_annotation, firebase_core, firebase_auth, cloud_firestore, firestore_cache, smooth_page_indicator, flutter_slidable)

HTTPS (Hyper-Text Transfer Protocol Secure) - Utilized by Firebase to encrypt in-transit data and isolate the customer data logically.

8.1.3 Message Formats:

- Firestore database querying with the Flutter Firebase API
- Firestore real-time data stream
- Data exchange service in JSON

8.1.4 Failure Conditions:

Our potential incompetency.

Section 9 - Extra Design Features / Outstanding Issues

Does not fit anywhere else above, but should be mentioned -- goes here

Section 10 – References

Any documents which would be useful to understand this design document or which were used in drawing up this design.

This will help understand better: [Github](#)

Section 11 – Glossary

Glossary of terms / acronyms

OS: Operating System

UI: User Interface

BS: Bill Splitting

DTO: Data Transfer Object

HTTPS: HyperText Transfer Protocol Secure.

Flutter: Open-Source Software Development Framework.

Dart: Programming language used to develop Web and Mobile applications.