

---

# **Software Project Documentation**

**for**

## **Splitsi (Group 10)**

**Prepared by**

**Linah Al Hamour Al Jarad (251009671)**

**Zhiqi Xie (Kelly) (251050986)**

**Victor Zhuoyue Shi (250853569)**

**Brendan Bain (251086487)**

**CS 4471**

**December 2, 2022**

## **Table of Contents**

<b>1. Introduction</b>	<b>1</b>
1.1 Problem Definition	1
1.2 Project Objective	1
1.3 Stakeholders List	2
1.4 Success/Acceptance Criteria for Stakeholders	2
<b>2. Diagrams</b>	<b>3</b>
2.1 Use Case Diagram	3
2.2 Use Case Descriptions	5
2.3 Sequence Diagrams	7
2.4 System Architecture	8
2.5 Detailed Class Diagram	9
2.6 State Machine Diagram	11
2.7 ER - Diagram	11
<b>3. Project Code</b>	<b>11</b>
3.1 GitHub	12
<b>4. Conclusion</b>	<b>12</b>
<b>5. References</b>	<b>12</b>
<b>Appendix A: Project WBS</b>	<b>12</b>
<b>Appendix B: Task Assignment Matrix</b>	<b>12</b>
<b>Appendix C: Sample of GitHub commits</b>	<b>12</b>

# 1. Introduction

## 1.1 Problem Definition

For many university students and roommates, splitting bills can be a vigorous task. At times, it can be difficult to split a bill between others if there was not an equal share of the expenses, so people often split the bill equally, although this is not a fair solution. Managing expenses is necessary for students and roommates as they frequently must consider paying monthly expenses and paying off student loans. According to a survey conducted by Ohio State University's 2015 National Student Financial Wellness Study, 70% of university/college students reported feeling stressed about their finances[1]. Many students must split financial expenses with their friends and roommates, such as, monthly rent, house utilities (hydro and electricity), groceries, restaurant bills, and so on. Each student has the right to pay their share of expenses, in order to manage their finances better and avoid spending too much money on items they did not contribute to as much as their friends/roommates.

Introducing Splitsi, an app designed for calculating and splitting expenses between students, roommates, or any group of people, and a place to store all the expenses in one view for users to look back at. Whether a student wants to split the monthly rent, or go out to dinner with their friends, Splitsi provides users a way to add as many items they wish to split, as well as the amount of people they wish to split the bill with. Users of Splitsi can choose to register and log in with an account to view their expenses history, but Splitsi also does not require users to register with an account, as that could be a tedious task, so they can use a Guest account instead, if they wish to make use of the application quickly. After the expenses are listed and calculated, Splitsi provides a unique link that can be shared amongst the participants of the bill, that lasts for 30 days, which provides a list of the expenses and how much each participant owes. This is to ensure a quick way to split the bills amongst roommates, without requiring each participant to register with an account on the app. It also makes sure that the participants pay the amount that they owe within the timeframe, before they lose access to their data.

## 1.2 Project Objective

In this project, Team 10 hopes to reach university students who are stressed about their finances and require splitting bills with their friends and roommates, to reduce their stress levels about their financial struggles by providing a faster way to discover the amount of expenses they must pay, and ensuring that their group is paying their share of expenses. By November 2022, Team 10 plans to accomplish this web application with all its required functionalities, to be ready for use by students. The following are required functionalities for the Splitsi system: When users access the application, they can choose to use a Guest or sign up/log in with a User account. In the Homepage, users can choose to create a new bill, view existing bills using a unique link, and log out if they are using a registered User account. If a user chooses to create a bill, they will be taken to the Create Bills page where they can add their expenses items. Once they are satisfied with all their expenses items, they can click on the "share the bill" button, which will notify the database to generate a unique link. The unique link is activated for 30 days before it will get permanently deleted from the database. A user can copy the link and send it to their friends and roommates. Once they access the unique link, they will see a description of all the expenses items that they owe, and they can add comments if they wish to discuss the bills further.

### 1.3 Stakeholders List

Stakeholder name	Role	Influence (High/Low)	Interest (High/Low)
Product team	Developer	High	High
Payee	Product user	Low	High
Payers	Product user	Low	High

### 1.4 Success/Acceptance Criteria for Stakeholders

Success Criteria	Description
Create a new bill	Users (Guest or Registered) can click a button on the homepage to create a new bill.
View existing bills	Registered users are able to view the history of previous bills that they have created under their account.
Add expenses	Users (Guest or Registered) are able to add expense items and their respective amount that will be split amongst the group members.
Add people to bills	Users (Guest or Registered) are able to add their group members to the bill and check mark which expense(s) they owe.
Calculate splitted bill	The bill form will indicate the amount of each expense after they've been split.
Generate a unique link	Once a bill is submitted, the system will generate a unique link, in which the user will be prompted to copy. The unique link will be set to expire in 30 days.
Access the unique link	Group members of the user (payers) will be able to access the unique link to view the bill form.
Write anonymous comments	Group members of the user (payers) are able to write anonymous comments on the bill form using the shared link if

	they wish to discuss the bills further with their group. Payers and Users (Guest or Registered) are able to view all the comments until the unique link expires.
Upload receipts (Optional)	Users (Guest or Registered) are able to upload an image of a receipt in the Bill expense form, for authenticity purposes, which will be displayed to the payers to see/verify when they access the unique link.

## 2. Diagrams

### 2.1 Use Case Diagram

For Figure 1, a user can register an account, create new bills, add other users' information, add expenses, add pictures of receipts, upload bills to the system, then the system generates a unique link. An anonymous user is a user. A registered user is a user. A registered user can log in, log out, view bill history, and change name.

For Figure 2, an anonymous user is a user. A registered user is a user. A user can share a unique link with other users. A user can access the unique link to view bills and add comments to the bills.

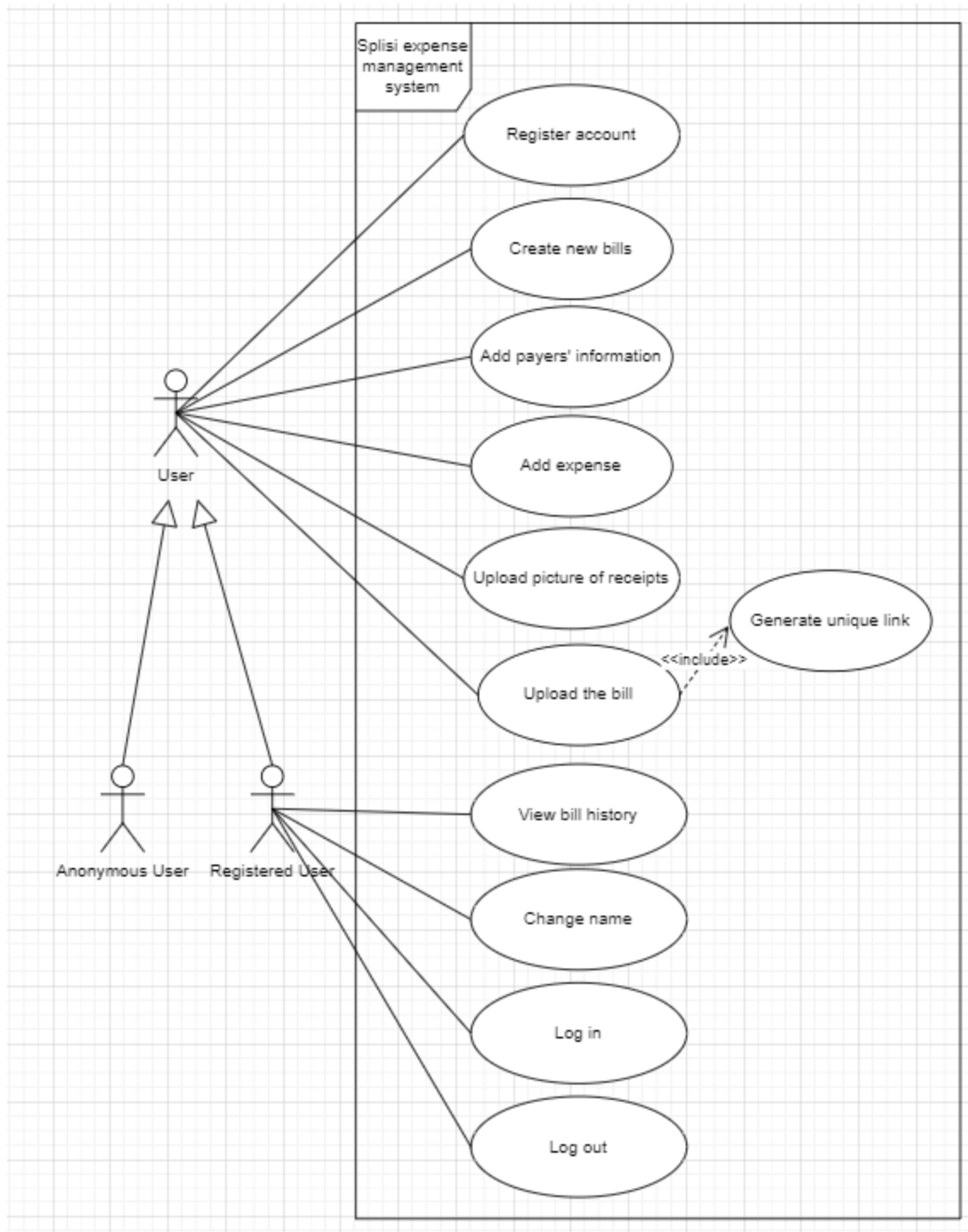


Figure 1: Use Case Diagram #1

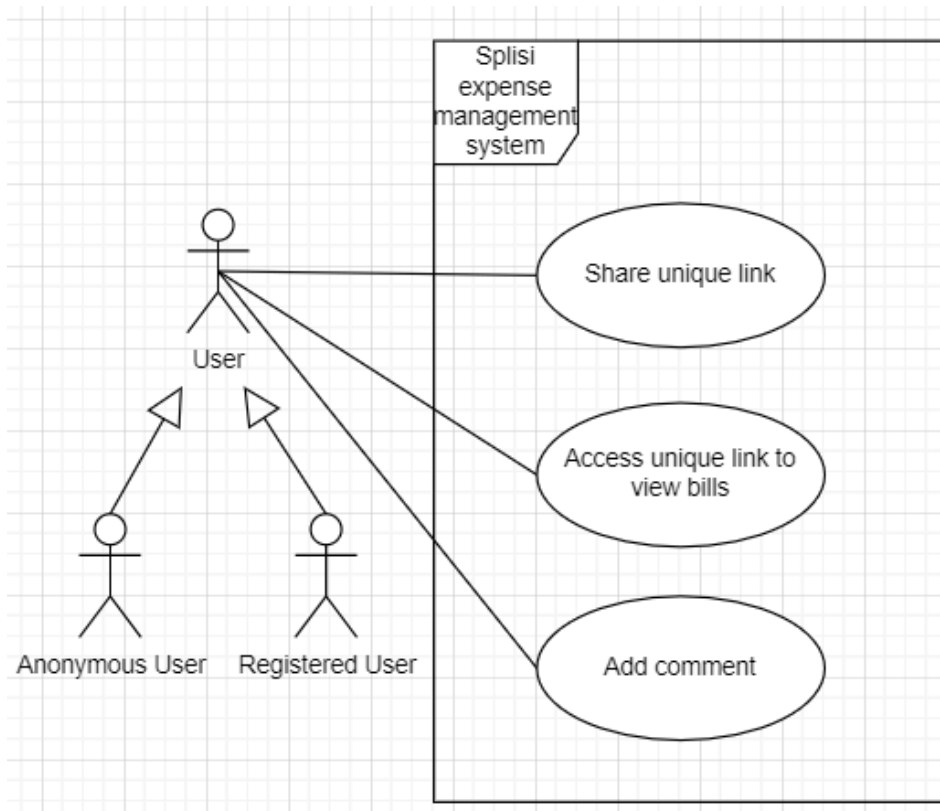


Figure 2: Use Case Diagram #2

## 2.2 Use Case Descriptions

Use Case Name	Split a bill
Scenario	Group payment
Triggering Event	Person makes payment on behalf of one or more people.
Brief Description	When a user needs to collect money, the user and system will create a new bill, add people to the bill, add expenses to the bill, add pictures of receipts to the bill, and upload the bill.
Actors	User.
Related Use Cases	N/A
Stakeholders	Payee: receiving the payment. Payer: making a payment.
Preconditions	User must exist.

	Expense(s) must exist.	
Postconditions	Bill must be saved.	
Flow of Activities	Actor	System
	<ol style="list-style-type: none"> <li>1. User creates a new bill.</li> <li>2. User adds expense(s) to the bill.</li> <li>3. User adds picture(s) of receipt(s) to the bill.</li> <li>4. User uploads the bill.</li> </ol>	<ol style="list-style-type: none"> <li>1.1. Create an empty bill.</li> <li>2.1. Calculate payments.</li> <li>4.1 Save bill.</li> </ol>
Exception Conditions	5.1. If expense(s) does not exist, then the bill is not uploaded.	

Use Case Name	Share a bill	
Scenario	Group payment	
Triggering Event	Person makes payment on behalf of one or more people.	
Brief Description	When a user uploads a bill, the system will generate a unique link, and display the bill at that link. User(s) accesses the link to view the bill, and add comments to the bill.	
Actors	User.	
Related Use Cases	Includes: <i>Split a bill</i> .	
Stakeholders	Payee: receiving the payment. Payer: making a payment.	
Preconditions	Bill must exist.	
Postconditions	The bill must be related to a unique link.	
Flow of Activities	Actor	System
	<ol style="list-style-type: none"> <li>1. System saves the bill.</li> <li>2. User shares the unique link.</li> <li>3. User accesses the unique link.</li> </ol>	<ol style="list-style-type: none"> <li>1.1. Generate a unique link.</li> <li>3.1. Display bill.</li> </ol>



	4. User adds comment(s) to the bill.
Exception Conditions	3.1. If the bill is past expiration date, then an error message is displayed.

## 2.3 Sequence Diagrams

For Figure 3, an unregistered user can register an account. A registered user with an account can log in. A user creates a bill, next adds other user's information, expense, picture of receipt, then uploads the bill to the system. The system generates the unique link.

For Figure 4, a user sends the unique link to other users. Then other users access the link to view the bill, and add comment to the bill.

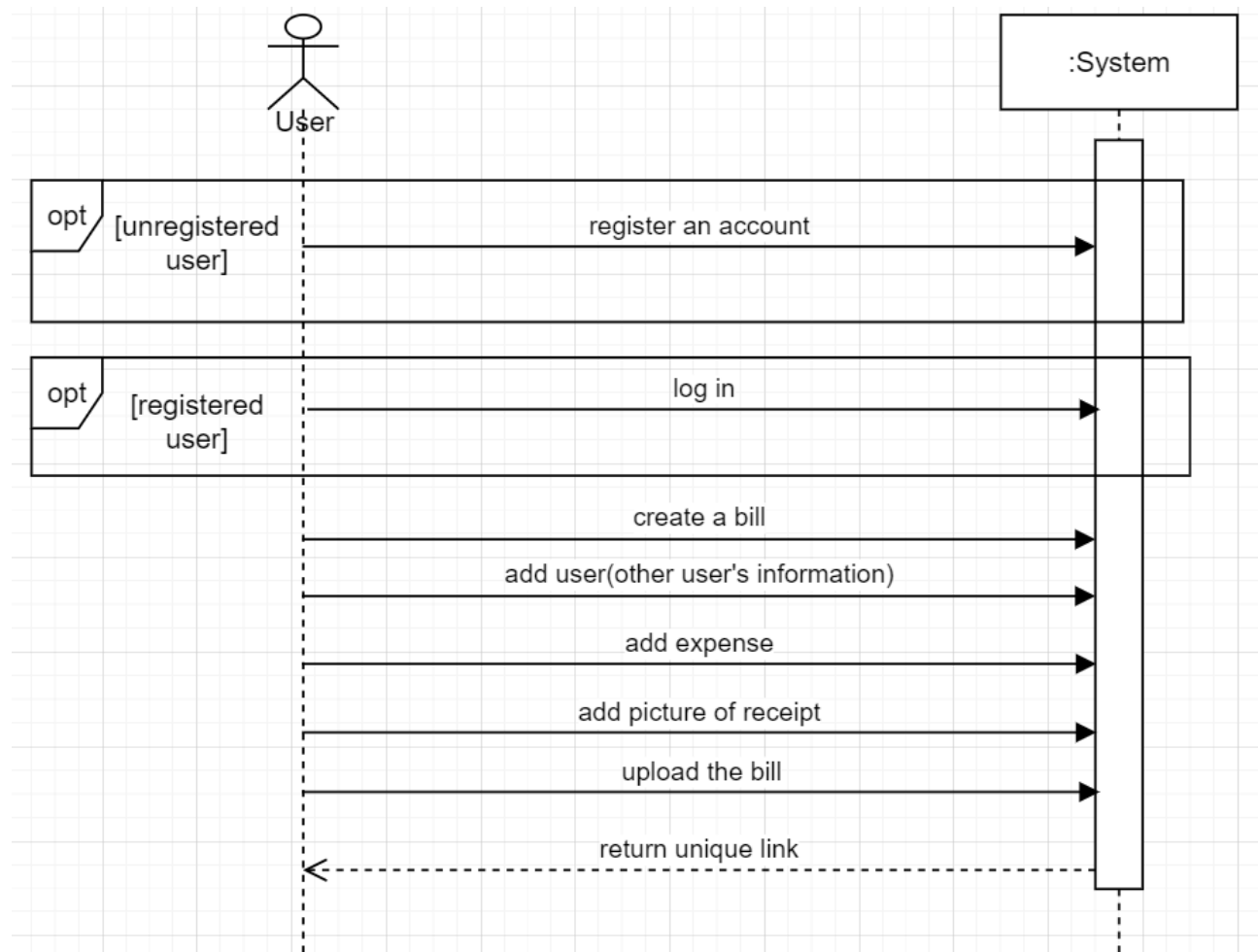


Figure 3: Sequence Diagram #1

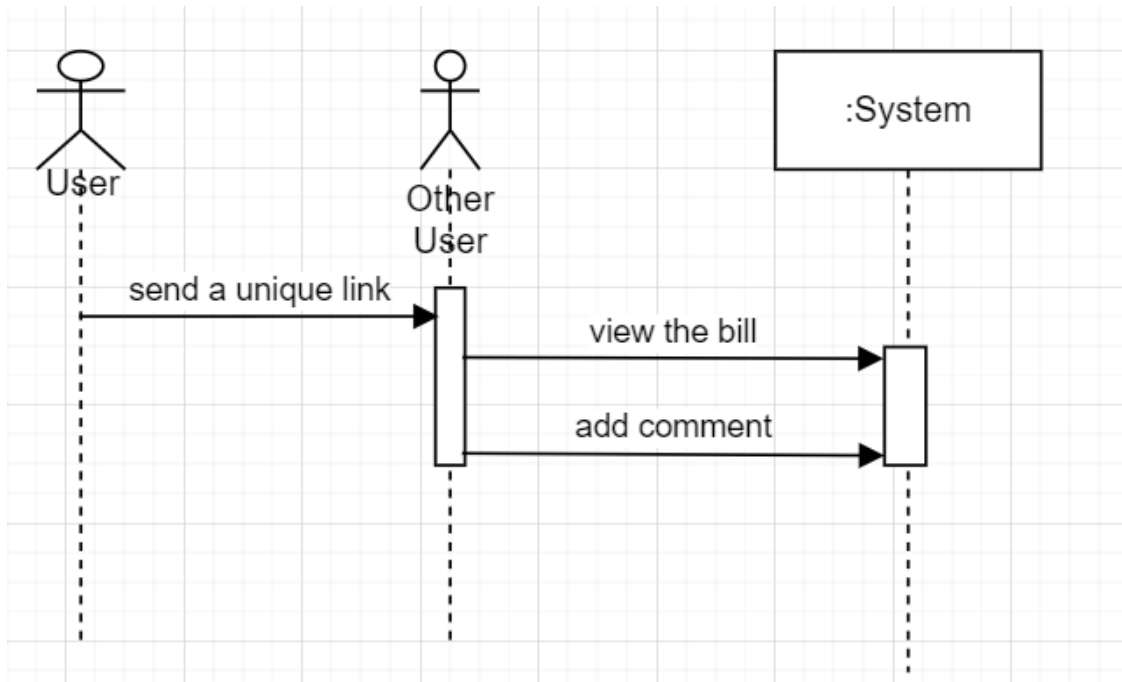


Figure 4: Sequence Diagram #2

## 2.4 System Architecture

The system architecture for Splitsi will use a closed-layered architecture model to best represent the product (see Figure 5). The closed-layered architecture model was chosen for the project because it reduces dependency for each of the layers in the project. This allows us to modify a layer without affecting the rest of the layers, making it easier to test the components and complete the project faster. The model includes four layers: the presentation layer, business layer, data access layer and the database layer. The presentation layer includes the Flutter web interface that will be viewed by users. The business layer includes the Flutter application logic, from managing expenses to the generation of unique links. The data access layer includes the Firebase application where we will access the inputted information from the expenses form. Lastly, the database layer includes the database in Firestore to store the information provided by the users as well as the unique links.

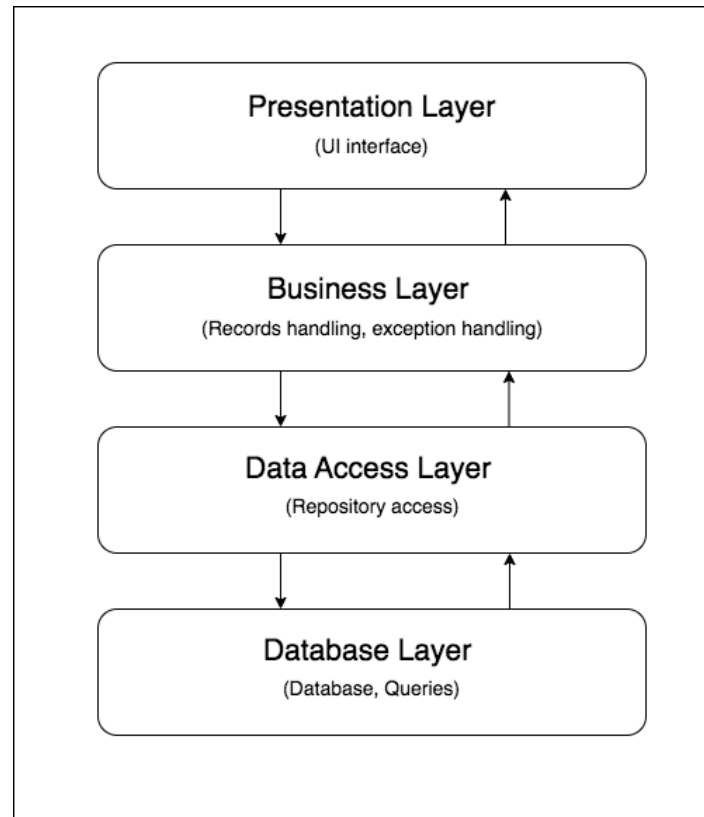


Figure 5: Closed Layered Architecture

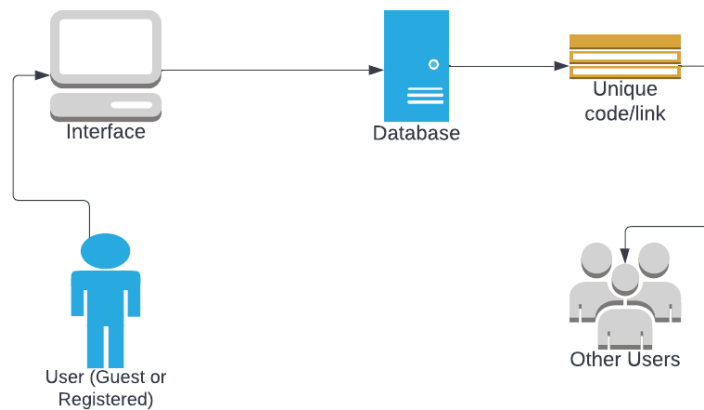


Figure 6: System Architecture

## 2.5 Detailed Class Diagram

In Figure 7, the class diagram shows how a user will create a bill. Users can be anonymous or signed in, where the major difference is that a signed-in user can view past bills. A bill is the organizational container for expenses, images (of receipts), and comments. When a bill is uploaded, it needs to have a unique code for identification and the code will also be used in the unique link. The other classes will reference the unique code so that they know the bill they are associated with. Bills will have timestamps so that the

database can remove expired data. Comments will have timestamps so that the comments can be organized by most recent.

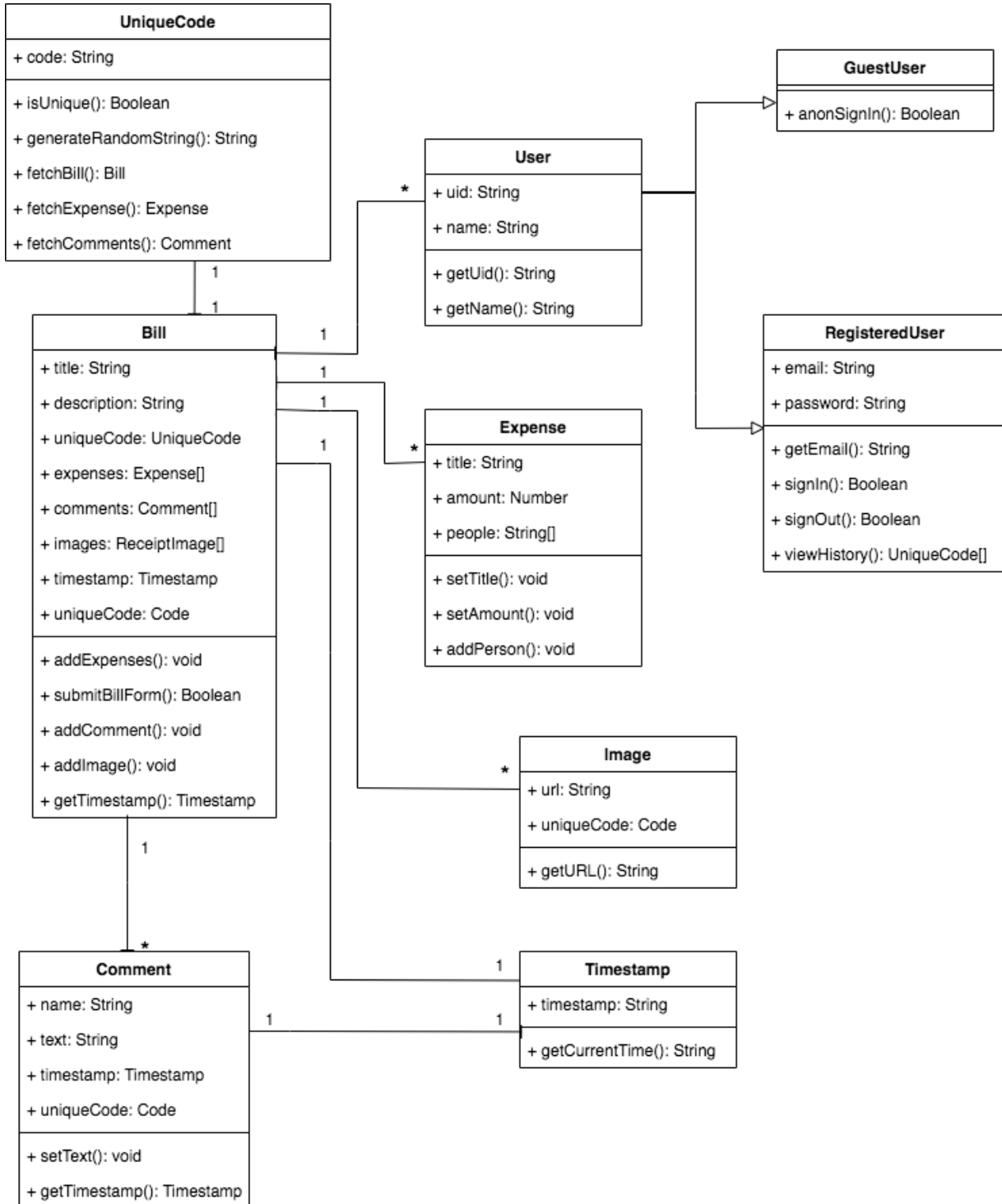


Figure 7: Class Diagram

## 2.6 State Machine Diagram

Figure 8 is a state machine depicting the functionalities and pages of Splitsi to better understand the page navigation of the website and in which order use cases can happen.

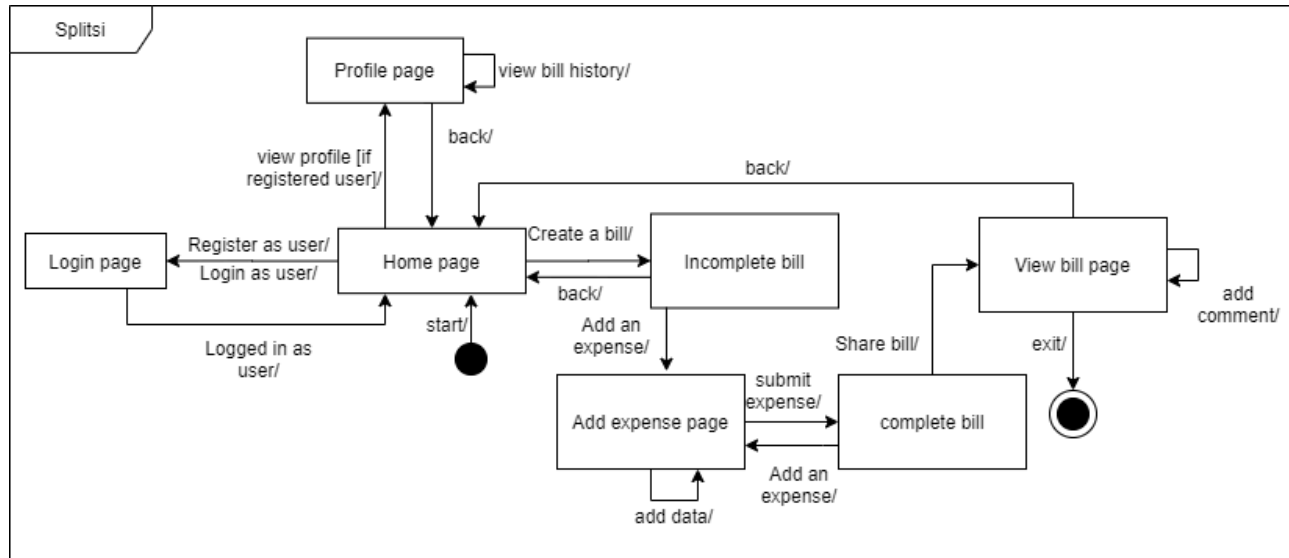


Figure 8: State Machine Diagram

## 2.7 ER - Diagram

The ER diagram in figure 9 shows which entities are being stored in Splitsi's database as well as which attributes are important to record.

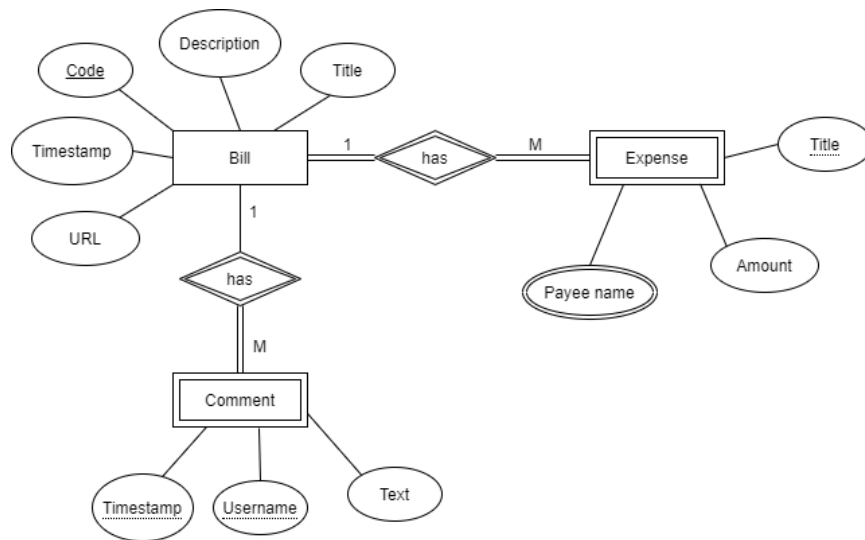


Figure 9: Entity-Relationship Diagram

## 3. Project Code

### 3.1 GitHub

<https://github.com/victorzshi/splitsi>

Live Website: <https://splitsi-3e71f.web.app/>

## 4. Conclusion

**Linah** - I learned that pair programming is an effective way to collaborate on developing the code as it allows us to share new ideas with the team and receive help/guidance in case we are stuck on something. I will continue to use and encourage others to utilize pair programming techniques when working on projects in the future.

**Kelly** - I learned that Scrum helps the development team work together closely and the developers receive feedback frequently.

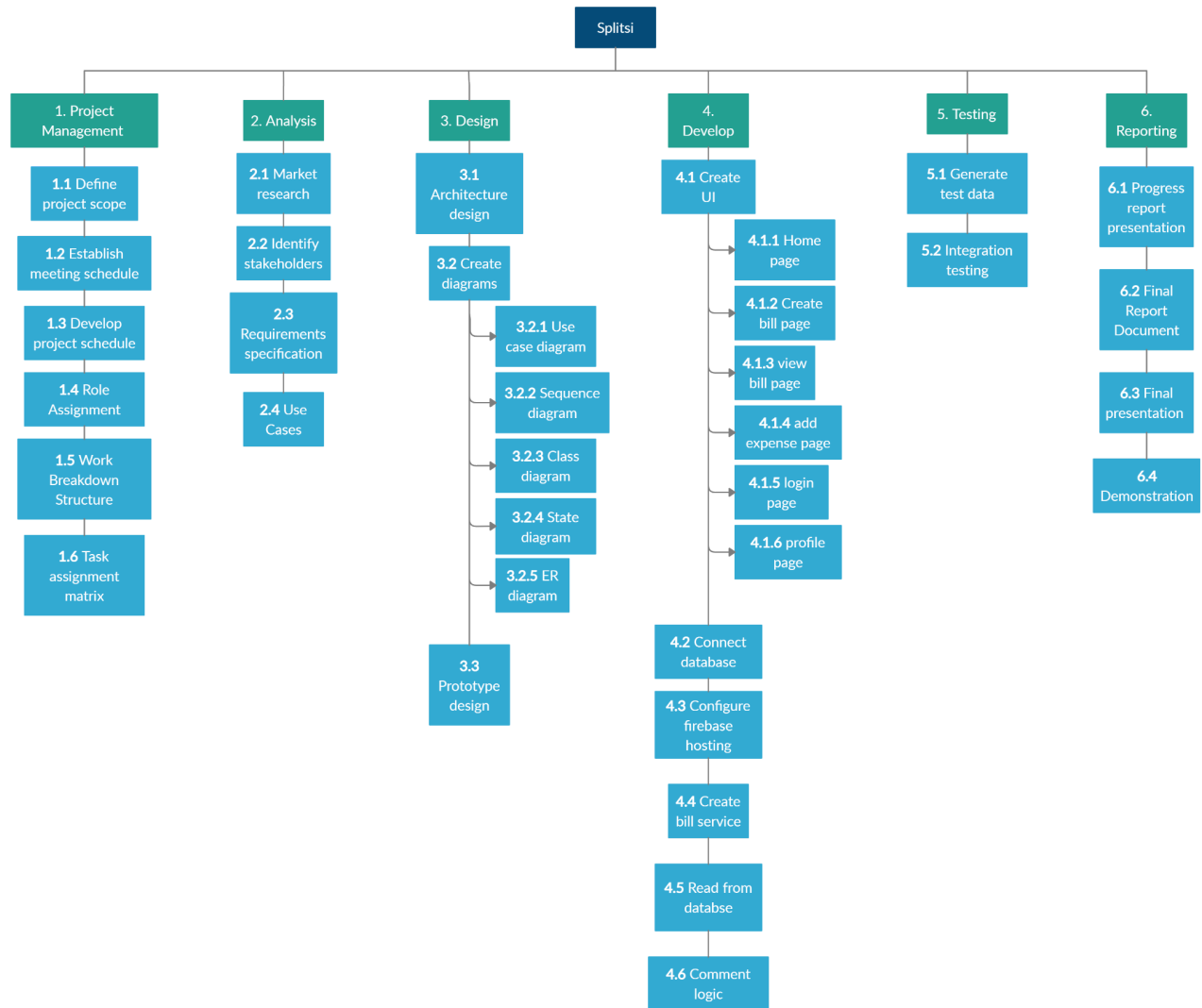
**Brendan** - I learned that version control is essential for tracking and making changes to our GitHub project.

**Victor** - I learned that it is important to manage my time in order to ensure that we are able to finish our parts on time.

## 5. References

[1] <https://www.betteryou.ai/financial-stress-in-college-students/>

## Appendix A: Project WBS



## Appendix B: Task Assignment Matrix












Responsible	performs task
Accountable	accountable/owner of task/final authority
Consulted	gives feedback/has expertise
Informed	should know about

ID	Task Description	Victor	Linah	Brendan	Kelly
1	Project Management				
1.1	Define project scope	R	A	C	C
1.2	Establish meeting schedule	R	C	C	A
1.3	Develop project schedule	C	R	A	C
1.4	Role definement	A	C	C	R
1.5	Work breakdown structure	C	A	R	C
1.6	Task assignment matrix	A	C	R	C
2	Analysis				
2.1	Market research	C	R	A	C
2.2	Identify stakeholders	I	A	I	R
2.3	Requirements specification	A	R	C	C
2.4	Use cases	R	C	C	A
3	Design				
3.1	Architecture design	C	R	C	A
3.2.1	Use case diagram	C	C	A	R
3.2.2	Sequence diagram	C	C	A	R
3.2.3	Class diagram	R	C	C	A
3.2.4	State diagram	C	A	R	C
3.2.5	ER diagram	A	C	R	C
3.3	Prototype design	R	C	A	C
4	Develop				
4.1.1	Home page	R	A	C	C
4.1.2	Create bill page	R	C	C	A
4.1.3	View bill page	C	A	C	R
4.1.4	Add expense page	C	R	A	C
4.1.5	Login page	C	R	C	A
4.1.6	Profile page	A	R	C	C
4.2	Connect database	R	C	A	C
4.3	Configure firebase hosting	A	C	R	C
4.4	Create bill service	C	A	C	R



4.5	Read from database	A	C	R	C
4.6	Comment logic	C	C	A	R
5	Testing				
5.1	Generate test data	R	I	I	A
5.2	System testing	I	A	R	I
6	Reporting				
6.1	Progress report presentation	C	A	C	R
6.2	Final report document	C	C	R	A
6.3	Final presentation	A	R	C	C
6.4	Demonstration	R	C	A	C

## Appendix C: Sample of GitHub commits

Show tooltip for copying link linahajjarad committed yesterday	 f173589 <>
Copy shareable link to clipboard linahajjarad committed yesterday	 b035392 <>
Style comments zxie86 committed yesterday	 51f332b <>
Set test data automatically zxie86 committed yesterday	 9b798ce <>
Refactor comment logic zxie86 committed yesterday	 e593473 <>
Add comment model brendan-bain committed yesterday	 053ad9d <>
Configure Firebase hosting brendan-bain committed yesterday	 6dfcb89 <>
Filter comments by code victorzshi committed yesterday	 edb5b75 <>
Commits on Nov 28, 2022	
Add comments victorzshi committed 2 days ago	 3d54c91 <>
Generate unique code in background linahajjarad committed 2 days ago	 edc3c8a <>
Create bill service zxie86 committed 2 days ago	 1f38eed <>