Splitsmart Term Project

Requirements Document

Team 6: Shahd Mustafa, Mike Nasser, Hasan Radwan, Hussen Al-Jubury, Nathaniel Leonardo

1. Introduction

1.1 purpose of document

The Purpose of this document is as follows, We will be analyzing the requirements given for this project and using these we will be grounding those requirements in how it will be practically used and to outline the structure of the project itself. We will be providing the goals for the project and its relationship to the developers, client, and potential future users; as such we will be going in depth into the functionality and interface of the project.

1.2 project summary

Project name: Split Smart
Project Members:

- Hasan Radwan
- Shahd Mustafa
- Hussein Aljubury
- Nathaniel Leonardo
- Mike Nasser

1.3 Project scope

The scope of this project is a Web based application that can support potentially millions of users simultaneously and provide fast and accurate services. The project will allow all of said users to be able to log in and have accounts along with the other functionalities. The application will be supported by both mobile and desktop applications for a wide range of users

1.4 System purpose

1.4.1 Users

The project will be available for a wide range of customers both on the web version of the application as well as a potential mobile version.

1.4.2 Location

The application will have a web version as the default, and potentially a mobile friendly version of the application will be available. Internet access will be necessary in order for the application to communicate changes within the app as well as enable login features.

1.4.3 Responsablites

The responsibilities of the software we are developing are as follows:

- Calculate expenses given a variety of user inputs
- Allow users to create accounts to maintain consistency within the application
- Allow users to edit and manage multiple different expenses and “events”
- Give users the ability to allow friends to contribute to “events”
- Allow users to generate reports on expenses and payment contributions
- Allow users with accounts to save expenses and reports via their account
- Deliver a user friendly interface

1.4.4 need

The system we are creating is a necessity for users to properly track and manage their expenses for events, both the user themselves and their friends and family participating with them. Expense management is getting increasingly difficult as the modern age has a lot of expenses that become difficult to track, the application aims to allow users the ease of access to do so.

1.5 overview of the document

- Section 2: Functional objectives: the objectives here are the intended behaviors for the application we wish to make, they will be sorted by the following priorities: High, medium and low.
- Section 3:Non-functional objectives: this section covers various technical requirements and constraints of the application.
- Section 4: Context model: the context model provides the goal of the system as a whole and how entities outside the system will interact with it in the context of real world applicability.
- Section 5: Use-case model: this section will go over the specific behavioral requirements of the application presented via use cases of specific scenarios that relate to the application.
- Section 6: The class model:A class is a collection of objects in the system that have the same data and behavior. All analysis classes and their relationships are shown on the class diagram.

2. Functional objectives

2.1 high priority

1. The system must allow the creation of user accounts.
2. The system must allow users to add each other on a friends list, using either user account names or phone numbers.
3. The system must allow for the creation of expense groups among users.
4. The system must allow each user within a group to add balances to said group.
5. The system must split the total balance among all users in the group and then accurately display to the nearest whole number, the total amount each user owes in that group.

2.2 medium priority

1. When a user adds a balance to a group it shall provide different options of splitting the bill, such as equally among users, or by individual percentages. It will also choose to split balances among only specific users in the group.
2. The users will be able to settle debts in groups.

2.3 low priority

1. Must allow users to see a group balance sheet, which includes information like the total amount of spending between each user in the group and the total amount of spending in the group in general.
2. The system should have an activities tab which shows all recent activity in groups that the user belongs to.
3. The system will notify the user when a new expense has been added to any group they are a part of.

3. Non-Functional Objectives

3.1 Reliability

- The Software should have 100% accuracy with respect to expense information and balance adjustments.
- Under normal circumstances, the software should have a less than 3 second response time for any user interactions (expense creation, balance adjustment, etc.).

3.2 Usability

- The interface should have obvious, simple labels and clear instructions for each field of the software process.

- The interface should use widely standardized visual cues to provide the user with confirmation/more information (asterisk for required fields, red for error, green for success, etc.).

3.3 Performance

- Under normal conditions, the software should be able to support an average number of concurrent users.
- The software should be available 24/7 with less than 5% downtime under non adverse conditions.
- Downtime should be scheduled whenever possible, and communicated to users via frontpage announcements.

3.4 Security

- All transactions should be encrypted to industry standards.
- System should be password protected, with password requirements meeting industry standards.

3.5 Supportability

- All modern web browsers (2010 or later) should be able to access the site without compatibility issues.
- Webapp software should work on various devices (mobile, desktop, etc.)
- Software should be designed to leave room to add on to it at a later stage.

3.6 Online User Documentation and Help

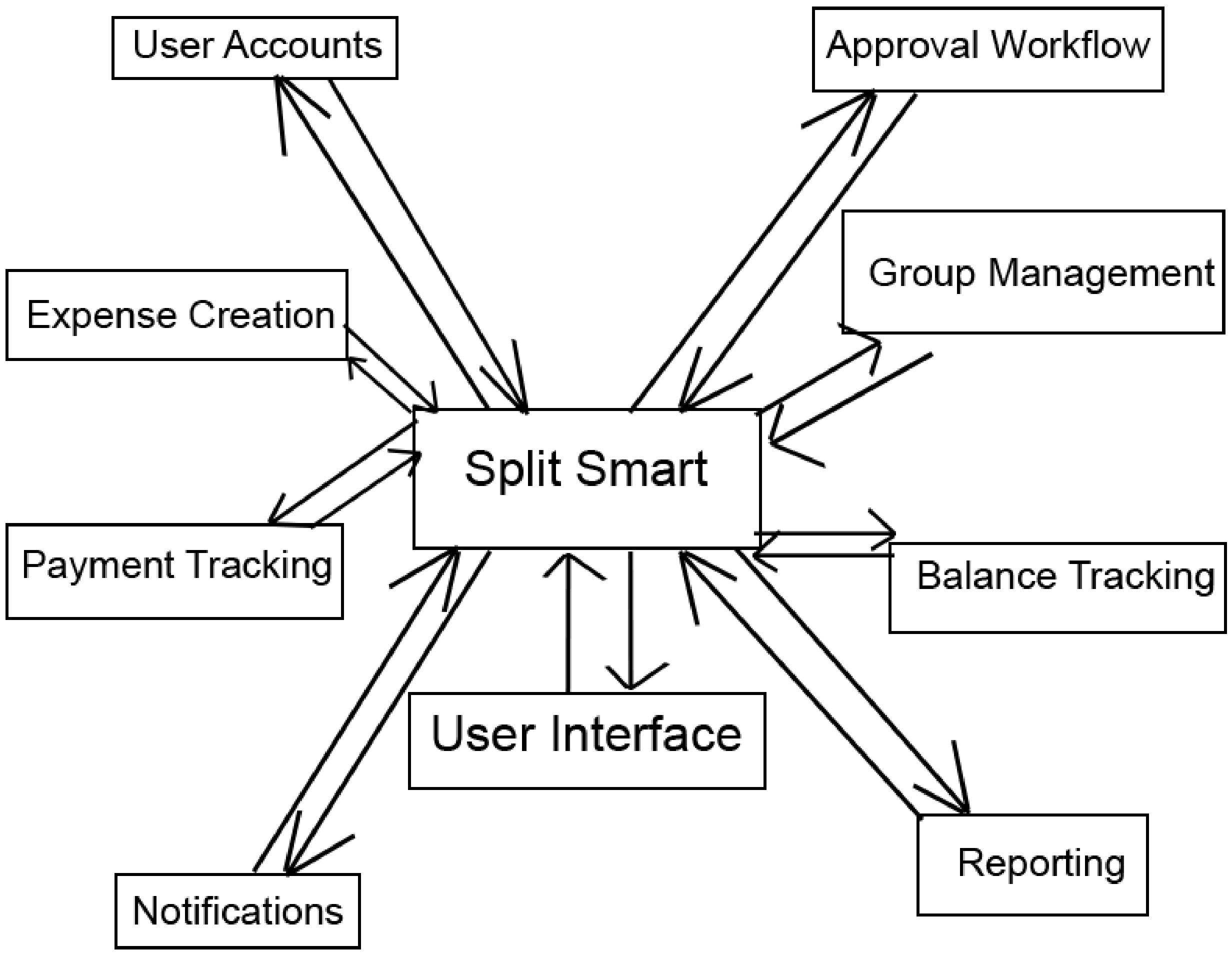
- There should be an easily accessible help page that can be accessed from any other page the user is on (help link in header).
- Help page should explain how to utilize and navigate the web app in simple terms.

4. The context model

4.1 goal statement

The goal of the SplitSmart software is to provide individuals and groups with a user-friendly and efficient solution for sharing expenses and managing balances. The software aims to streamline the process of expense creation, group management, balance tracking, and payment settlement, while also offering robust reporting capabilities to provide users with a clear overview of their financial activities. The software can be accessed through a standalone application, web browser, or mobile app.

4.2 Context diagram



4.3 system externals

- User Interface
    - The software should have a user interface that allows users to interact with the system, create accounts, log in, and access various features such as expense creation, group management, balance tracking, payment recording, and report generation.
- User Accounts

    - The system should support the creation and management of user accounts, allowing individuals to log in and access their own expense and balance information. User accounts should include necessary details such as name, email, and password.
- Group Management
    - The system should provide functionality for creating and managing groups, enabling users to invite and add members to specific groups. This feature allows for easier tracking of shared expenses and balances within those groups.
- Expense Creation
    - The system should allow users to create new expenses, specifying the amount, date, description, shared manner (equally split or split by percentage), and optionally attaching a receipt image. Users should also be able to select the involved users or group members for each expense.
- Approval Workflow
    - The system should include an approval workflow for reviewing and approving expenses before they are added to the system and used to adjust balances. This feature ensures accuracy and accountability in expense tracking.
- Notifications
    - The system should have a notification system in place to alert users when other group members create new expenses. Notifications can be in the form of push notifications, email notifications, or in-app notifications.
- Balance Tracking
    - The system should track the balance owed by each user to every other user, considering all expenses that have been entered. It should update and reflect the accurate balances in real-time.
- Payment Tracking
    - The system should allow users to record when payments have been made to settle balances owed. Users should be able to indicate the date and amount of the payment, which will result in updating the balances accordingly.
- Reporting

- The system should generate reports and summaries of expenses, balances, and payments. Users should be able to access pre-defined reports as well as generate custom reports with specific date ranges and other parameters to gain insights into their financial activities.

5. The use case model

5.1 system use case description (for selected cases)

- **User login**

Use case name:	User login
Summary:	To get into the “SplitSmart” site users need to login to access their own expenses and view balance.
Basic Flow:	1. The use case begins when the user chooses to login. 2. The system requests username and password. 3. The user enters their username and password. 4. The system undergoes a verification process for the username and password. 5. The system begins the session, displaying the home page where users can view their account information, balance, view/create groups and examine reports.
Alternative Flows:	Step 4: invalid username or password entered results in going back to step 2 in the use case to re-enter the username and password. When the user enters a valid username and password the use case moves on to step 4.
Extension Points:	None.
Preconditions:	User has previously created an account.
Postcondition:	The user is able to create groups, expenses, payments and access their information.

- **User registration**

Use case name:	User registration
Summary:	New users register a username and password.
Basic Flow:	1. The use case begins when the user chooses to register for an account. 2. The system request for username and password. 3. The user enters username and password. 4. The system verifies the entered username and password for duplicates, username and password can only use letters and numbers with no symbols. 5. The system requests the following information (all are required): name, address (street name, city, zip code, state, country), phone number, email address. 6. The user enters their information. 7. The system verifies information. 8. The system executes the use case *Payment information*.. 9. The system begins the login session and displays the home page.
Alternative Flows:	Step 4: the user enters a pre-existing username, the system will display a message and the use case goes back to step 2. Step 4: the user enters an invalid username and password format, the system will display a message and the use case goes back to step 2. Step 6: the user does not enter all required information, the system will display a message and the use case goes back to step 5.
Extension Points:	*Payment information*.
Preconditions:	None.
Postcondition:	Users are able to view their account profile.

- **Payment information**

Use case name:	Payment information
Summary:	This use case allows the user to enter payment information.

Basic Flow:	1. The use case begins after the user enters their profile information. 2. The system requests for card information. 3. The user enters their card information. 4. The system verifies that the card information is valid.
Alternative Flows:	Step 4: the user entries invalid card information, the use case goes back to step 2.
Extension Points:	None.
Preconditions:	The user is logged in to the system.
Postcondition:	None.

- **Manage groups**

Use case name:	Manage groups
Summary:	The use case allows users to add other users to join their group.
Basic Flow:	1. The use case begins when the user indicates that they want to create a group. 2. The system requests a name for the group created. 3. The user enters a name for the group. 4. The system displays slots to add friends/family to the group. 5. The user enters in the members name and email. 6. The system verifies that the entered members have an account. 7. The system sends invites to selected users. 8. The system goes back to the home page, the newly created group is displayed.
Alternative Flows:	Step 4: the user enters members that do not have a registered account. The use case goes back to step 3.
Extension Points:	None.
Preconditions:	All members have a registered account.

Postcondition:	Users are able to view and manage groups.

- **Create new expenses:**

Use case name:	Create new expenses
Summary:	The user creates new expenses.
Basic Flow:	1. The use case begins when the user indicates that they want to create a new expense. 2. The system requests for users to choose which group to assign the expense to. 3. The user indicates the group for the expense to be sent to. 4. The system requests for expense information: amount, date, description, split (system executes *split expense*) and optional image. 5. The system requests which member is charged, the system executes *assign expense*. 6. The user enters the expense information and chooses members to charge. 7. The system undergoes approval process for the expense.
Alternative Flows:	Step 4: if the user enters invalid expense information. The use case goes back to step 2. Step 4: if user enters invalid unknown users. The use case goes back to step 3. Step 5: if the approval process fails, the use case goes back to step 2.
Extension Points:	*Split expense ,assign expense*
Preconditions:	User has created a group.
Postcondition:	The user can view the new expense created.

- **Split Expense**

Use case name:	Split Expense
Summary:	The user is able to specify how the expense is split among the members.
Basic Flow:	1. The use case begins when a new expense a creation has been made,

	while in the use case *create new expense*. 2. The user is able to specify how the expense is split among all group members (equally or %) 3. The system will execute the use case, assign *expense*, and display the expense based on how it was split.
Alternative Flows:	None.
Extension Points:	*Assign expense*
Preconditions:	The system had executed the use case *create new expense.*
Postcondition:	The members are charged the split amount.

- **Assign Expense**

Use case name:	Assign Expense
Summary:	The user selects members to assign the expense to, thus chagrin them the amount.
Basic Flow:	1. The use case begins when a new expense a creation has been made, while in the use case *create new expense*. 2. The system will charge the members chosen within the expense. 3. The system requests for the member to select a payment option such as credit card or paypal. 4. The system uses pre entered card information to pay the expense. 5. The system stores the payment details (executing *Expense report)* and displays a message to the user. 6. The system updates the balance, the use case *update balance* is executed.
Alternative Flows:	None.
Extension Points:	*Expense report, update balance*
Preconditions:	The system had executed the use case *create new expense*

Postcondition:	The customer is charged, and the use case *expense report* is updated.

- **Customer report**

Use case name:	Customer report
Summary:	The user is able to view summaries of all their expenses, balance, and payments.
Basic Flow:	1. The use case begins when the user indicates they want to view their report or see a custom report. 2. The system uses the use case *update balance* and *assign expense* to create a report. 3. If a custom report is chosen the system will request the date range and produce a report. 4. The user is able to view the report.
Alternative Flows:	Step 3: if the system cannot find any information for the specified date range chosen, the use case goes back to step 1.
Extension Points:	None.
Preconditions:	The user has previously entered information in the use case *update balance* and *assign expense.*
Postcondition:	The user is able to view their report.

- **Expense Notification**

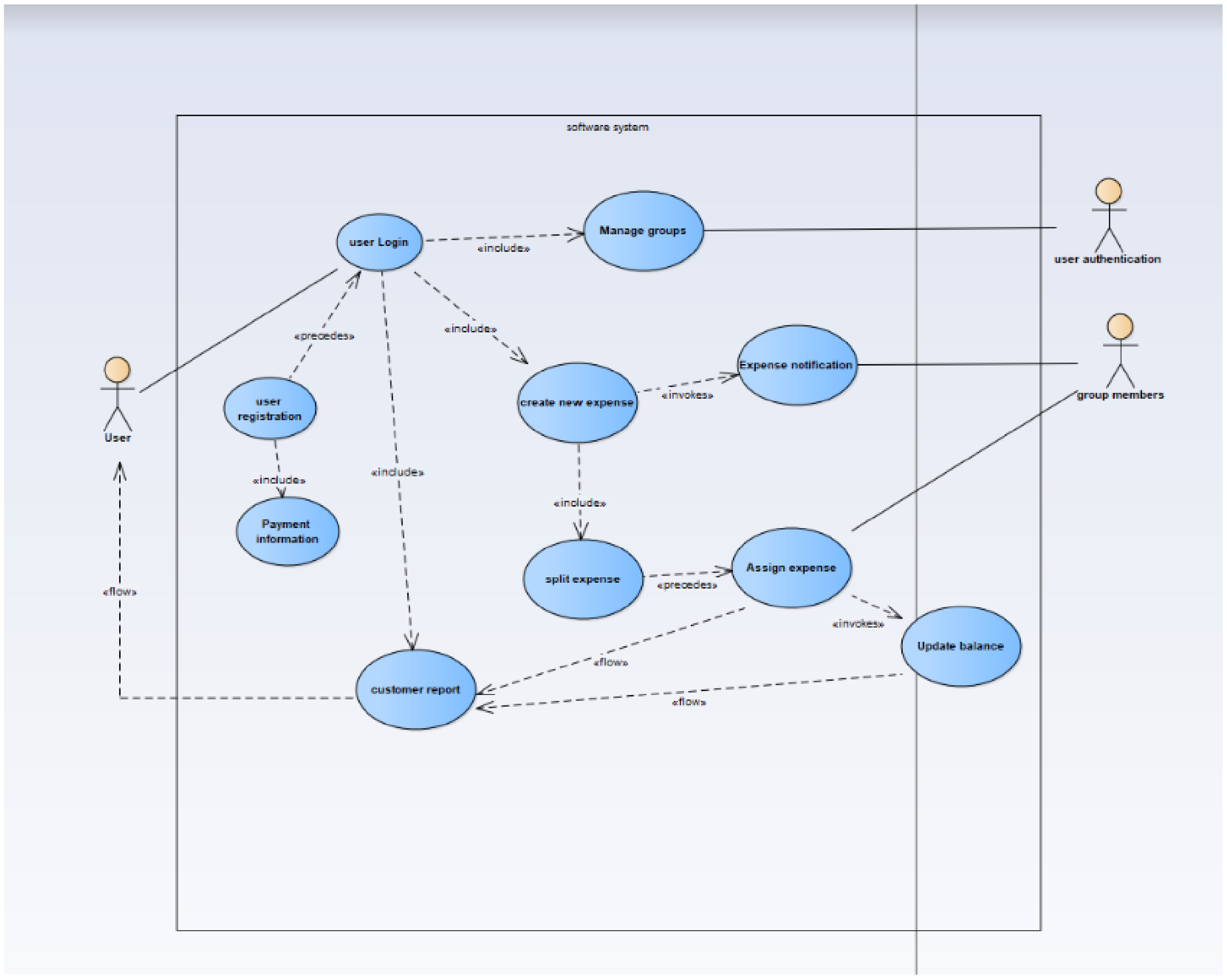
Use case name:	Expense Notification
Summary:	The system sends user notifications when new expenses are made.
Basic Flow:	1. The use case begins when a new expense a creation has been made, while in the use case *create new expense*. 2. The system sends a notification to all members that were chosen.
Alternative	None.

Flows:	
Extension Points:	None.
Preconditions:	One user creates a new expense.
Postcondition:	none.

- **Update Balance**

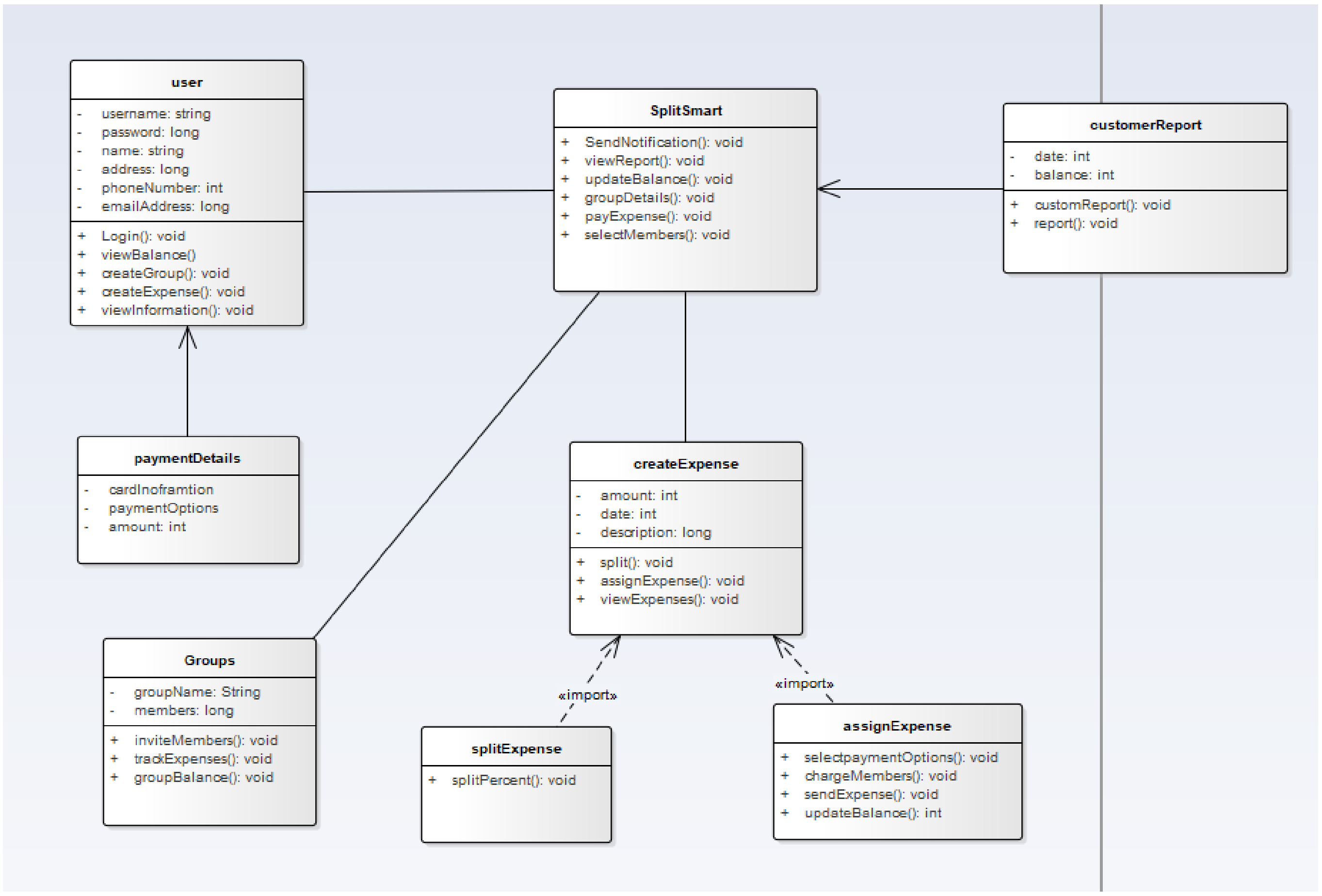
Use case name:	Update Balance
Summary:	The system tracks and updates the user balance when new expenses and payments are done.
Basic Flow:	1. The use case begins when the user executes the use cases *create new expense and assign expense*. 2. The system updates the balance owed by each user to all other users. 3. The system displayed the updated balance to the user. 4. The system stores changes made in the use case *expense report.*
Alternative Flows:	None.
Extension Points:	*Expense report*
Preconditions:	The user had previously had expenses made or made payments.
Postcondition:	The new balance is displayed to the user.

Behavioral model:



software system
user Login
Manage groups
«include»
user authentication
«precedes»
«include»
Expense notification
group members
User
user registration
create new expense
«invokes»
«include»
Payment information
«include»
«include»
split expense
Assign expense
«flow»
«precedes»
«invokes»
Update balance
«flow»
customer report
«flow»

6. The class model (shahd)


user
- username: string
- password: long
- name: string
- address: long
- phoneNumber: int
- emailAddress: long
+ Login(): void
+ viewBalance()
+ createGroup(): void
+ createExpense(): void
+ viewInformation(): void
SplitSmart
+ SendNotification(): void
+ viewReport(): void
+ updateBalance(): void
+ groupDetails(): void
+ payExpense(): void
+ selectMembers(): void
customerReport
- date: int
- balance: int
+ customReport(): void
+ report(): void
paymentDetails
- cardInoframtion
- paymentOptions
- amount: int
createExpense
- amount: int
- date: int
- description: long
+ split(): void
+ assignExpense(): void
+ viewExpenses(): void
Groups
- groupName: String
- members: long
+ inviteMembers(): void
+ trackExpenses(): void
+ groupBalance(): void
«import»
«import»
splitExpense
+ splitPercent(): void
assignExpense
+ selectpaymentOptions(): void
+ chargeMembers(): void
+ sendExpense(): void
+ updateBalance(): int