Software Design Document

Group 8

Expense Monitoring System

Software Engineering
Fall 2023

CS673 A1

BOSTON UNIVERSITY

Abhishek Gupta
Siddharth Krishnakumar
Julia Peterson
Jiuzhou Lu
Haolong Yang

Content

1. Overview

Software Design Document (SDD) of the Expense Monitoring System provides the necessary definitions to conceptualize and further formalize the design of the software, whose requirements and functionalities were summarized in the Software Requirements Specifications (SRS) Report. The aim is to provide guidance to a design that could be easily implemented by any programmer reading this report. The document complies with the IEEE standards (IEEE Std 1016 – 2009).

1.1 Scope

This complete SDD will contain the general definition and features of the project, design constraints, the overall system architecture and data architecture, and a brief explanation of our current progress and schedule of the project. With the help of UML diagrams, the design of the system and subsystems/modules will be explained visually in order to help the programmer to understand all information stated in this document correctly and easily. For instance, we will use class diagrams, use case diagrams, and many other UML diagrams to illustrate the software design of the Expense Monitoring System.

The scope of the Expense Monitoring System encompasses

A. User-Centric Features:

- Personalized dashboard displaying real-time expenses and budget metrics.
- Ability to input, categorize, and track daily expenses.
- Customizable alerts for budget limits and irregular transactions.

B. Reporting Tools:

- Generate and export monthly and annual expense reports.
- Visual representations of spending patterns, including graphs and charts.
- Breakdown of expenses by category, vendor, and payment method.

C. Budget Management:

- Set, modify, and track monthly and yearly budgets.
- Compare actual spending to budgeted amounts, highlighting overages or underages.

D. Integration and Compatibility:

- Ability to sync with bank accounts and credit cards for automated expense tracking.
- Integration with financial planning tools or software, if required.
- Accessible via web and mobile platforms.

E. Security and Compliance:

- Robust user authentication and data encryption to ensure the security of financial data.
- Compliance with financial data handling and privacy regulations.

1.2 Purpose

This SDD is intended to provide a software system design that will satisfy functional and nonfunctional requirements stated in the SRS Document of the Expense Monitoring System. The purpose of this document is to serve as a guideline throughout the development phase of the project for developers.

The Expense Monitoring System was developed to provide an efficient and user-friendly platform for individuals and organizations to monitor, manage, and analyze their financial expenses. The system will serve as an essential tool for financial management, promoting awareness and facilitating accountability.

1.3 Intended Audience

Audience is the staff who will develop the application described in this document, all the team 8 members, and the professor.

2. Definitions

- **EMS -** Expense Monitoring System, our project.
- **SRS** - Software Requirements Specification
- **SDD** - Software Design Document
- **UI -** User Interface, the means by which a user and system interact
- **GUI -** Graphical User Interface, the method used to mediate relations between user and device
- **DBMS** - Database Management System
- **API** - Application Programming Interface
- **HTML** - HyperText Markup Language
- **UML** - Unified Modeling Language
- **FQ -** Frequency, duration in which a wave repeats
- **QA -** Quality Assurance.
- **UX -** User Experience.

3. Design Description Information Content

3.1 Introduction

The Design Description Information provides a detailed breakdown of the software components, their relationships, and their functionality within the Expense Monitoring System (EMS). This section aims to offer the intended audience a clear understanding of the system's architectural choices, data flow, and user interface designs. It will identify how this web application will be designed and implemented. Throughout the document identification, diagrams, user views, and user viewpoints are provided.

3.2 SDD Identification

After testing for the verification and validation, users could login/sign up and be directed to the Home page. Those with enough privileges will be able to click one of the tabs, including Personal, Groups, and Dashboard.

When the users click the Personal tab, EMS will display personal expense information, including spending amount, budget, and expense status.

When the users click the Groups tab, they can click “New” to create a new group or click the remaining group. For the former one, the users will be directed to the Create Group page and they can complete it after filling out the group information. Then, they will redirect to the Group page. For the latter one, EMS will display the expense information of the selected group. The users will be able to click “Invite” to invite another user to be a member of this group after filling out the new member information. Then similarly, they will redirect to the Group page.

When the users click the Dashboard tab, EMS will display a graphical and textual analysis of the user expenses. And in this page, users can click “export report” to get the analysis.

3.3 Design Stakeholders and Their Concerns

The successful design and implementation of the EMS depend on the collaboration and feedback from various stakeholders. Each group of stakeholders brings specific concerns based on their roles and expectations. Addressing these concerns is vital for ensuring the system's usability, functionality, and overall success.

A. End-Users Concerns:

- Usability and accessibility of the system.
- Data accuracy and real-time updates.
- Security and privacy of personal financial data.
- Compatibility and performance on different devices or different browsers.

B. Developers Concerns:

- Clarity in system architecture and component interactions.
- Availability of development tools and resources.
- Integration capabilities with third-party services or platforms.
- Flexibility for future enhancements or modifications.

C. Project Managers Concerns:

- Alignment of the design with project timelines and milestones.
- Resource allocation and budget adherence.
- Risks associated with the design and potential mitigation strategies.
- Stakeholder communication and expectation management.

D. Testers Concerns:

- Availability of comprehensive test cases and expected outcomes.
- Logging and reporting mechanisms to track bugs, defects, and their resolutions.
- Communication channels with developers to understand design intricacies and to report anomalies.
- Frequency and methods of software updates, ensuring that tests remain relevant for newer versions.

3.4 Design Views

This project will be implemented as a modular structure. The Architectural View includes modules and their relationships, key architectural patterns employed, and interactions between the system and external entities. Object-oriented design is chosen for this project, which gives us an advantage in integrating new features into our project and removing and replacing the components that we want.

3.5 Design Elements

Design elements are the fundamental building blocks of the system that, when combined, realize the system's functionalities and requirements. They consist of entities and the attributes associated with these entities.

3.5.1 Design Entities

- User
- Group
- Dashboard
- Expense
- Budget
- Category
- Report
- Notification
- Transaction

3.5.2 Design Attributes

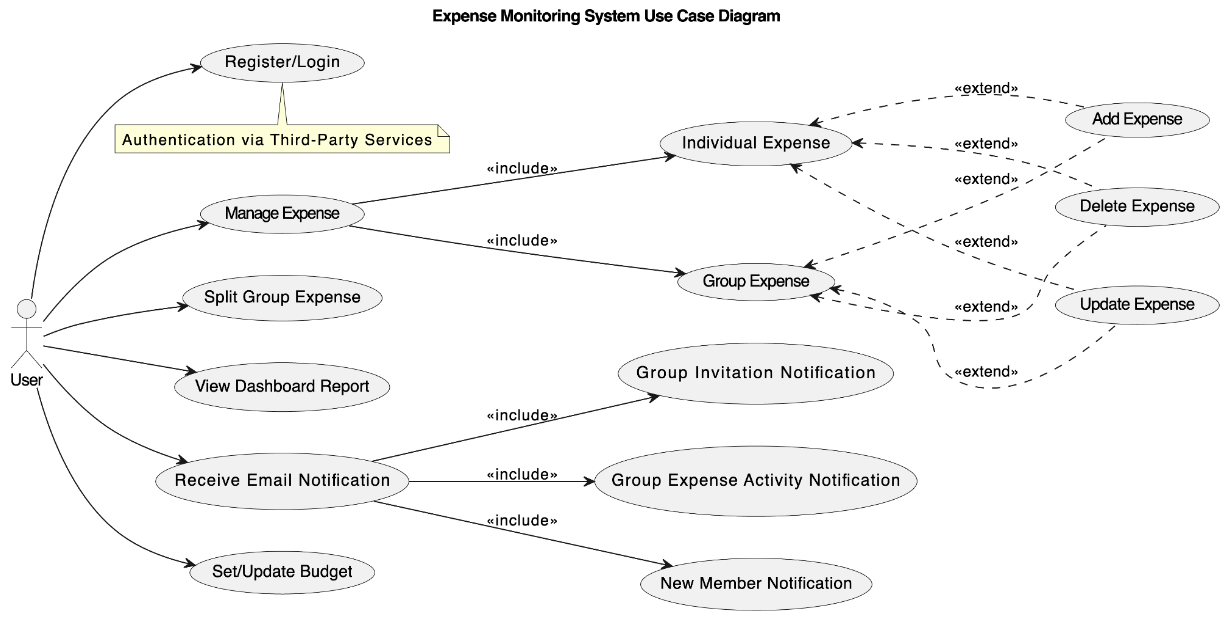
Entity	***Description***
User	Holds the information of the user.
Group	Holds the information of the group.
Dashboard	Displays graphical and textual analysis of the user expenses.
Expense	Captures specific costs or expenditures of the user.
Budget	Represents the financial limits set by the user for specific durations.
Category	Organizes expenses into different types or groups.
Report	Aggregates and presents user expenses for analysis.
Notification	Conveys system-generated alerts or messages to the user.
Transaction	Logs financial activities, either as income or expenditure.

3.6 Design Languages

Unified Modeling Language (UML) is selected as a part of the design viewpoint specification. For example, the use case diagram, class diagram and data flow diagram.

4. Design Viewpoints

4.1 Use Case Diagram



4.1.1 Register/Login

Users can register for a new account or log into the Expense Monitoring System. Registration can be facilitated through third-party authentication services. During registration, users provide username, email address, password, and upon successful registration, they gain access to the system's features.

4.1.2 Manage Expense

4.1.2.1 Add Expense

For individual expenses, users can add a new entry detailing the nature, amount, and other attributes of the expense. For group expenses, this action adds an expense to be shared among group members.

4.1.2.2 Update Expense

Users can modify the details of an existing individual or group expense. This includes changing the amount, nature, or other attributes of the recorded expense.

4.1.2.3 Delete Expense

Users can remove an individual or group expense entry from the system. Once deleted, the expense data is no longer available for review or calculations.

4.1.3. Individual Expense

Users can manage their individual expenses. The specific operations they can perform on individual expenses are detailed in the extended use cases.

4.1.4 Group Expense

Users can manage expenses that are shared within a group. Specific operations applicable to group expenses are detailed in the extended use cases.

4.1.5 Split Group Expense

For expenses that are shared within a group, users have the option to split the cost. The system will divide the expense among group members, ensuring each member knows their share.

4.1.6 View Dashboard Report

Users can view a comprehensive report on the dashboard, summarizing their expenses. This report might include trends, monthly totals, and other analytical insights.

4.1.7 Receive Email Notification

4.1.7.1 Group Invitation Notification

Users will be notified via email when they are invited to join a group.

4.1.7.2 Group Expense Activity Notification

Users will receive an email notification whenever there is an activity related to a group expense, such as when a new expense is added to the shared group.

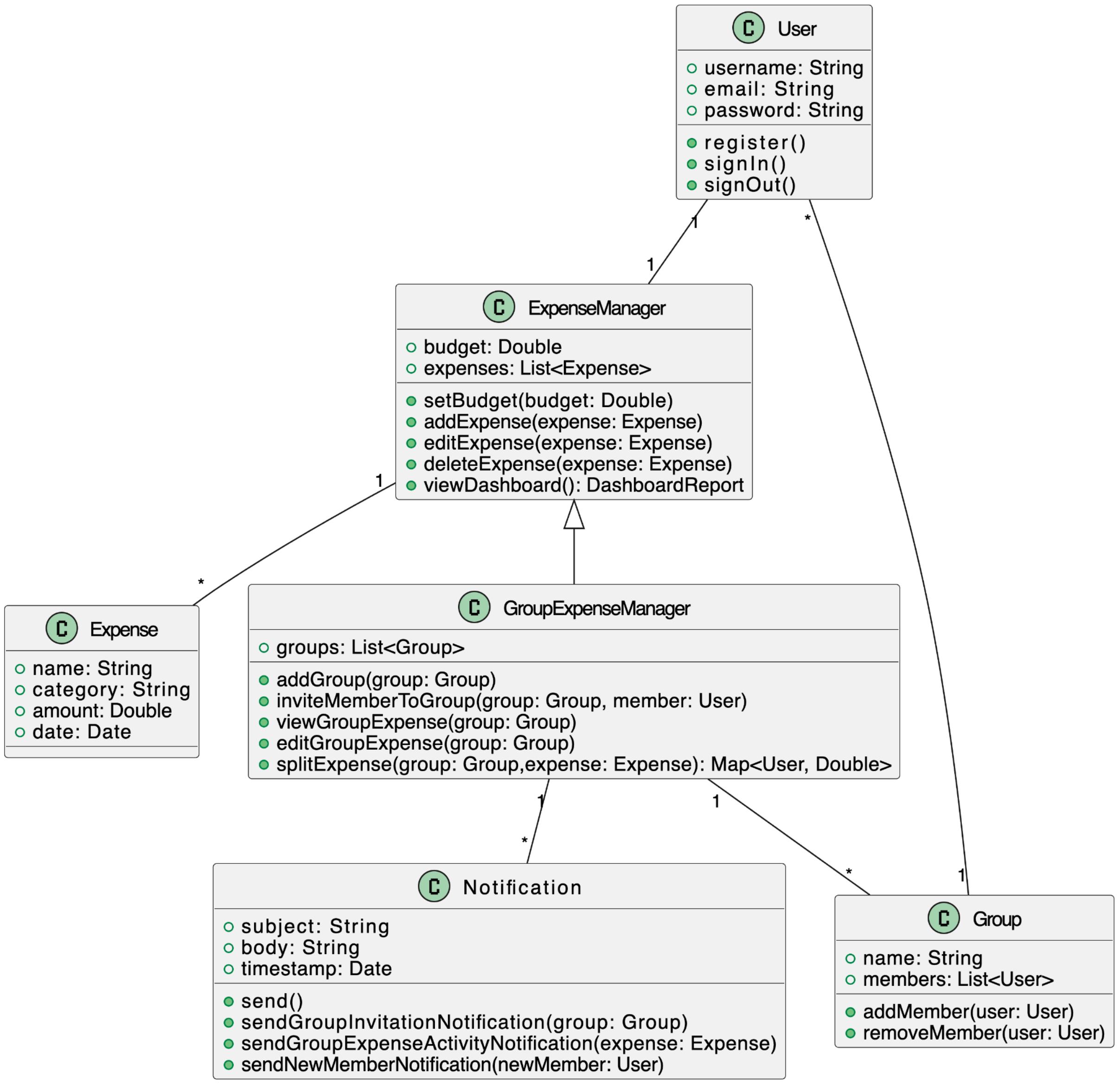
4.1.7.3 New Member Notification

Whenever a new member joins a group, existing members of the group will receive an email notification about the new member.

4.1.8 Set/Update Budget

Users can define a budget for their expenses. Once set, they can also update it as required. The system might provide warnings or insights based on the set budget and actual expenses.

4.2 Class Diagram



User
username: String
email: String
password: String
register()
signIn()
signOut()
ExpenseManager
budget: Double
expenses: List<Expense>
setBudget(budget: Double)
addExpense(expense: Expense)
editExpense(expense: Expense)
deleteExpense(expense: Expense)
viewDashboard(): DashboardReport
Expense
name: String
category: String
amount: Double
date: Date
GroupExpenseManager
groups: List<Group>
addGroup(group: Group)
inviteMemberToGroup(group: Group, member: User)
viewGroupExpense(group: Group)
editGroupExpense(group: Group)
splitExpense(group: Group,expense: Expense): Map<User, Double>
Notification
subject: String
body: String
timestamp: Date
send()
sendGroupInvitationNotification(group: Group)
sendGroupExpenseActivityNotification(expense: Expense)
sendNewMemberNotification(newMember: User)
Group
name: String
members: List<User>
addMember(user: User)
removeMember(user: User)
1
1
*
1
*
1
*
1
*
1

4.2.1 USER CLASS

This class represents an individual who interacts with the Expense Monitoring System. It captures personal details and provides methods for user-related activities such as registration and signing in/out.

- FIELDS

  username: Unique identifier for the user, chosen by the user.
  email: Email address of the user; serves as a point of contact.
  password: Secret code chosen by the user to secure their account.

- FUNCTIONS

  register(): Allows a new user to create an account in the system.
  signIn(): Permits an existing user to log into the system.
  signOut(): Allows the logged-in user to safely exit from the system.

4.2.2 EXPENSE CLASS

This class captures the details about a particular expense, whether individual or group.

- FIELDS

  name: Name or description of the expense.
  category: Type or category the expense belongs to.
  amount: The monetary value of the expense.
  date: The date the expense was incurred.

4.2.3 EXPENSE MANAGER CLASS

This class oversees the management of expenses, including adding, editing, or deleting them. It also holds a list of expenses and a set budget. It is a class directly used for personal expense management.

- FIELDS

  budget: Monetary limit set by the user to manage their expenses.
  expenses: A list of all expenses the user has added.

- FUNCTIONS

  setBudget(budget: Double): Allows the user to define or change their budget.
  addExpense(expense: Expense): Adds a new expense to the user's list.
  editExpense(expense: Expense): Modifies the details of an existing expense.
  deleteExpense(expense: Expense): Removes a selected expense from the list.
  viewDashboard(): Provides a summary or report of the user's expenses.

4.2.4 GROUP EXPENSE MANAGER CLASS

This subclass inherits from ExpenseManager and provides functionalities specific to managing group expenses.

- FIELDS

  groups: A list of all groups that a user is part of.

- FUNCTIONS

  addGroup(group: Group): Adds a new group to the list.
  inviteMemberToGroup(group: Group, member: User): Sends an invitation to a user to join a specific group.
  viewGroupExpense(group: Group): Views all expenses related to a specific group.
  editGroupExpense(group: Group): Modifies details of an existing group expense.

4.2.5 GROUP CLASS

Represents a group in which users can jointly manage and share expenses.

- FIELDS

  name: Name or identifier of the group.
  members: A list of all users that are part of the group.

- FUNCTIONS

  addMember(user: User): Adds a new member to the group.
  removeMember(user: User): Removes an existing member from the group.

4.2.6 NOTIFICATION CLASS

Handles the notifications related to group activities such as invitations, expenses, and new member additions.

- FIELDS

  subject: The main theme or title of the notification.
  body: Detailed information or message content of the notification.
  timestamp: The time when the notification was generated.

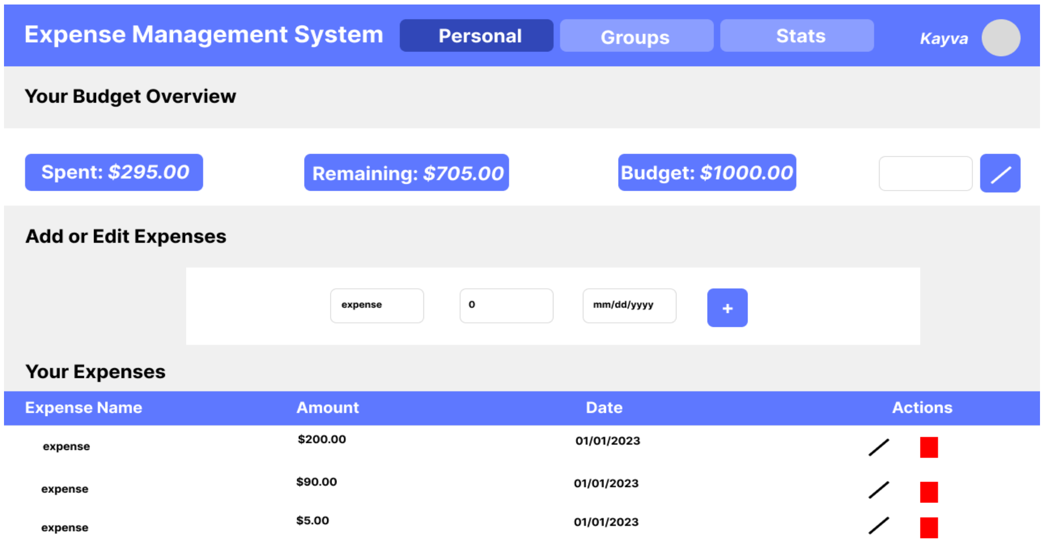
- FUNCTIONS

  send(): Sends a notification to the relevant parties.
  sendGroupInvitationNotification(group: Group): Notifies a user about an invitation to a group.
  sendGroupExpenseActivityNotification(expense: Expense): Informs group members about a new or edited expense.
  sendNewMemberNotification(newMember: User): Alerts existing group members about the addition of a new member.

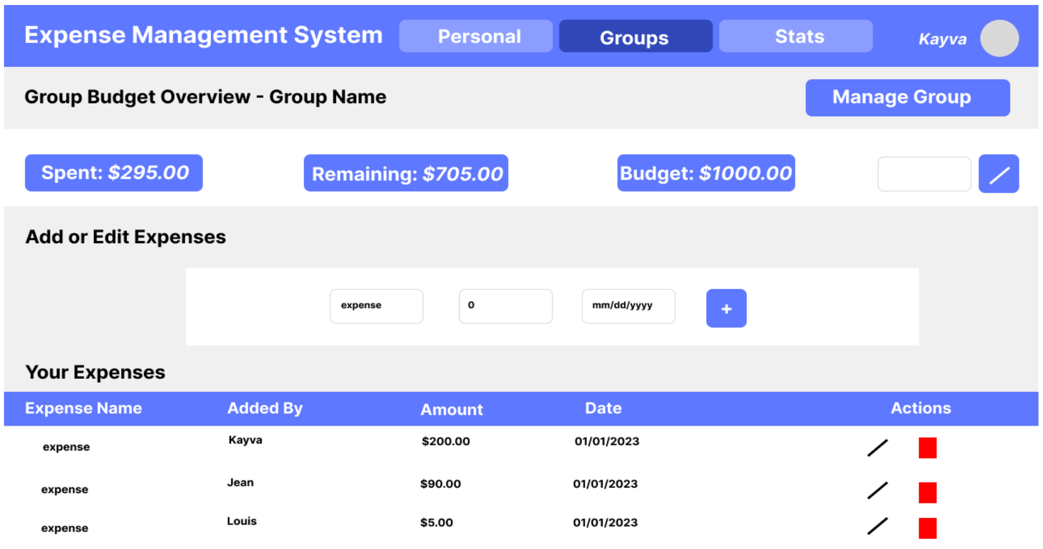
4.3 Interface Design

The interface is designed to be clean and readable, with a similar interface used between the different pages to allow for a quick understanding of the functionality without wasting time and effort learning new patterns and behaviors. Users can select between the three different pages on the top bar. The group page allows for access to a separate sub-page for managing groups, and the statistics page allows for the user to select between their personal budget or one of their groups.

On the personal and group pages, users can create an expense from the middle bar, and view a list of their expenses on the bottom bar. The pencil icon allows users to edit previous expenses and the trash can icon allows them to delete the expense. The pages also offer an overview of the budget, in both total, spent, and remaining funds.

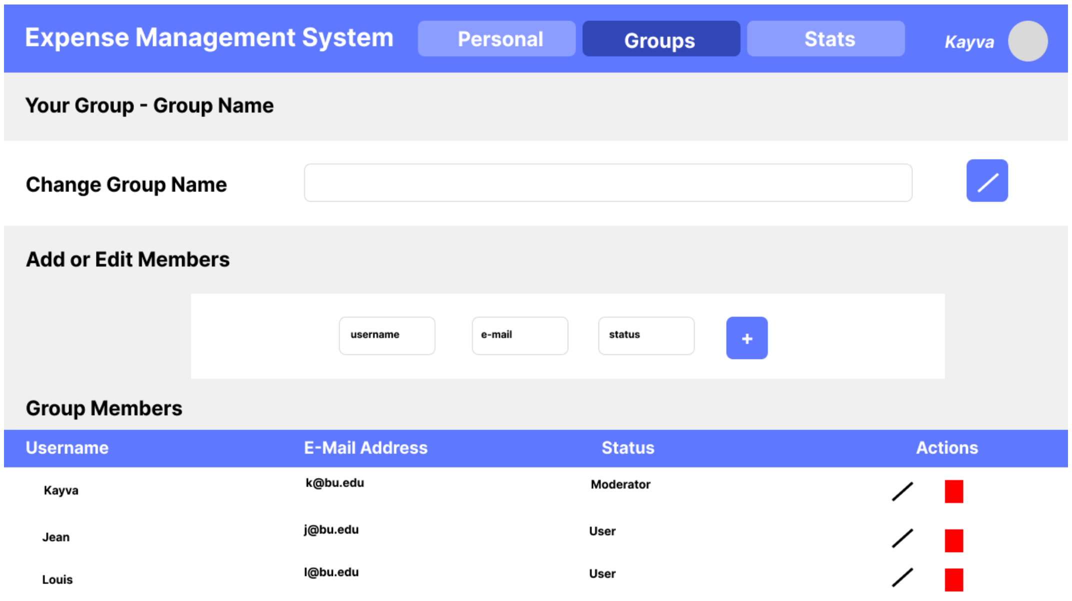


Expense Name	Amount	Date	Actions
expense	$200.00	01/01/2023	⁄ 🟥
expense	$90.00	01/01/2023	⁄ 🟥
expense	$5.00	01/01/2023	⁄ 🟥



Expense Name	Added By	Amount	Date	Actions
expense	Kayva	$200.00	01/01/2023	⁄ 🟥
expense	Jean	$90.00	01/01/2023	⁄ 🟥
expense	Louis	$5.00	01/01/2023	⁄ 🟥

The group management page allows moderator class users to add and remove members of a group, by entering their information, and the statistics page allows for the creation of line and pie charts showing the allocation of funds and spending over time.



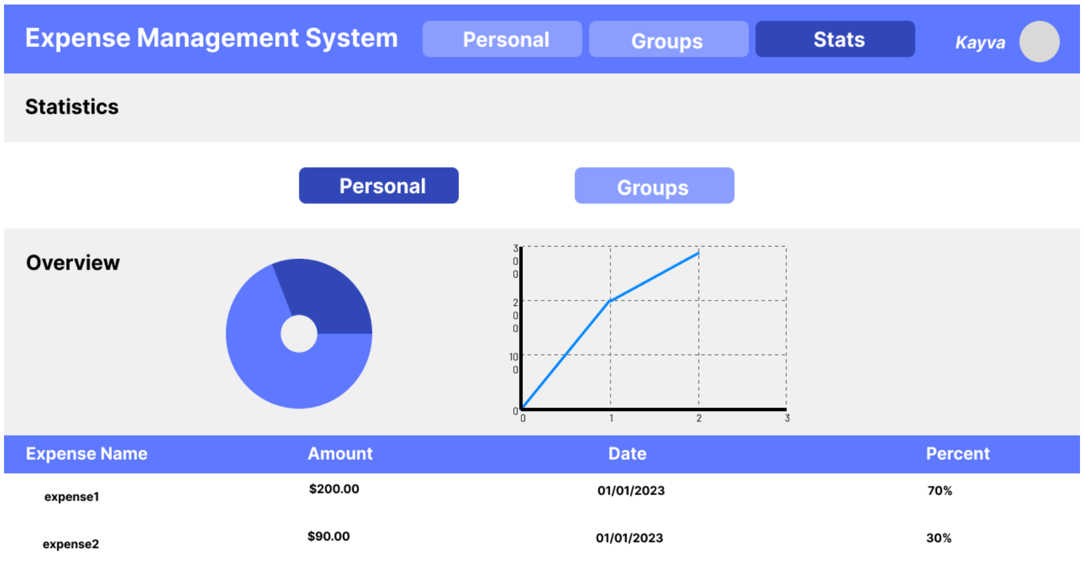
Your Group - Group Name

Change Group Name

Add or Edit Members

Group Members

Username	E-Mail Address	Status	Actions
Kayva	k@bu.edu	Moderator	/ 🟥
Jean	j@bu.edu	User	/ 🟥
Louis	l@bu.edu	User	/ 🟥

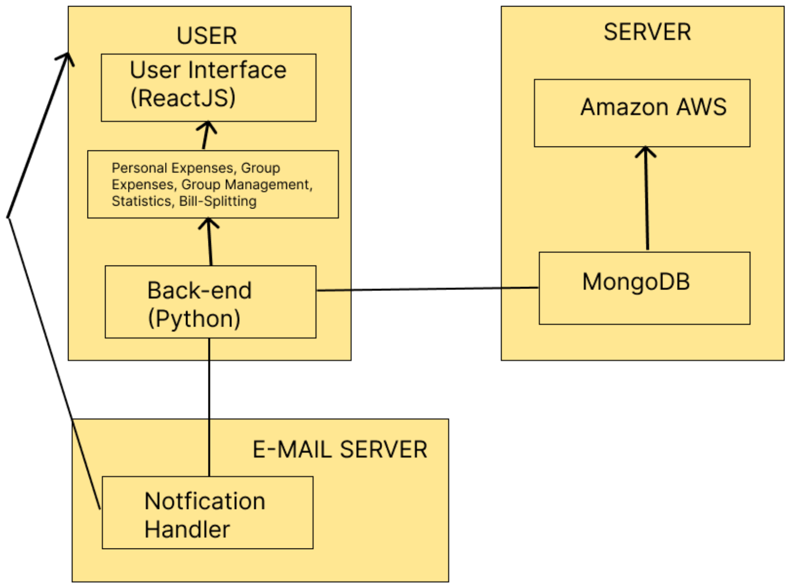


Statistics

Overview

Expense Name	Amount	Date	Percent
expense1	$200.00	01/01/2023	70%
expense2	$90.00	01/01/2023	30%

4.4 Component Diagram



The following components are being used in our project.

User Interface (ReactJS): Serving as the frontend, this component is developed using ReactJS. It offers users an interactive platform where they can engage in various operations like managing personal expenses, group expenses, bill-splitting, and reviewing expenses.

Back-end (Python): Built using Python Flask micro API, it processes all user requests, from fetching expense data to handling group management tasks. This backend system ensures smooth, efficient data flow between the user interface and the database.

E-Mail Server & Notification Handler: This component plays a crucial role in our user communication strategy. These notifications are dispatched to users via our dedicated E-Mail Server, ensuring timely delivery.

Amazon AWS: The cloud powerhouse, Amazon AWS, hosts our application, providing robust, scalable infrastructure to accommodate our system's needs. It guarantees high availability and top-tier performance, ensuring a seamless experience for our users.

MongoDB: Our choice for database management, MongoDB provides efficient, flexible data storage solutions. Directly interfacing with our Python backend, MongoDB safely stores user data, from expense details to group management configurations.

4.5 API Design

4.5.1 Personal Expense Management

4.5.1.1 Create Personal Expenses

Endpoint: /personal/addExpense
Method: POST
Description: Adds a new personal expense.

4.5.1.2 Retrieve Personal Expenses

Endpoint: /personal/getExpenses
Method: GET
Description: Fetches all personal expenses for the user.

4.5.1.3 Update Personal Expense

Endpoint: /personal/updateExpense/:expenseId
Method: PUT
Description: Updates a specified personal expense.

4.5.1.4 Delete Personal Expense

Endpoint: /personal/deleteExpense/:expenseId
Method: DELETE
Description: Deletes a specified personal expense.

4.5.1.5 Set Personal Expense Budget

Endpoint: /personalExpense/setBudget
Method: POST
Description: Sets a budget limit for personal expenses.

4.5.1.6 Retrieve Personal Expense Budget

Endpoint: /personalExpense/getBudget
Method: GET
Description: Fetches the set budget limit for personal expenses.

4.5.1.7 Edit Personal Expense Budget

Endpoint: /personalExpense/editBudget
Method: PUT
Description: Updates the budget limit for personal expenses.

4.5.2 Group Management

4.5.2.1 Create Group

Endpoint: /group/create
Method: POST
Description: Creates a new group.

4.5.2.2 Retrieve Groups

Endpoint: /group/list
Method: GET
Description: Fetches all groups the user is a part of.

4.5.2.3 Update Group Details

Endpoint: /group/update/:groupId
Method: PUT
Description: Updates details of a specified group.

4.5.2.4 Delete Group

Endpoint: /group/delete/:groupId
Method: DELETE
Description: Deletes a specified group.

4.5.2.5 Add Member to Group

Endpoint: /group/addMember
Method: POST
Description: Adds a new member to an existing group.

4.5.2.6 Remove Member from Group

Endpoint: /group/removeMember
Method: DELETE
Description: Removes a member from an existing group.

4.5.3 Group Expense Management

4.5.3.1 Add Group Expense

Endpoint: /groupExpense/add/:groupId
Method: POST
Description: Adds a new expense to a specified group.

4.5.3.2 Retrieve Group Expenses

Endpoint: /groupExpense/list/:groupId

Method: GET
Description: Fetches all expenses within a specified group.

4.5.3.3 Update Group Expense

Endpoint: /groupExpense/update/:expenseId
Method: PUT
Description: Updates details of a specified group expense.

4.5.3.4 Delete Group Expense

Endpoint: /groupExpense/delete/:expenseId
Method: DELETE
Description: Deletes a specified expense from a group.

4.5.4 Bill Settlement

4.5.4.1 View Bill Summary

Endpoint: /group/{groupID}/billSummary
Method: GET
Description: Retrieves a summary of all bill settlements within a specific group, detailing who owes whom and how much.

4.5.4.2 Settle Bill

Endpoint: /group/{groupID}/settleBill
Method: POST
Description: Settles the bill by recording a member's payment towards settling the group's dues.
Required Parameters in the body: Payer's user ID, recipient's user ID, and the amount being settled.

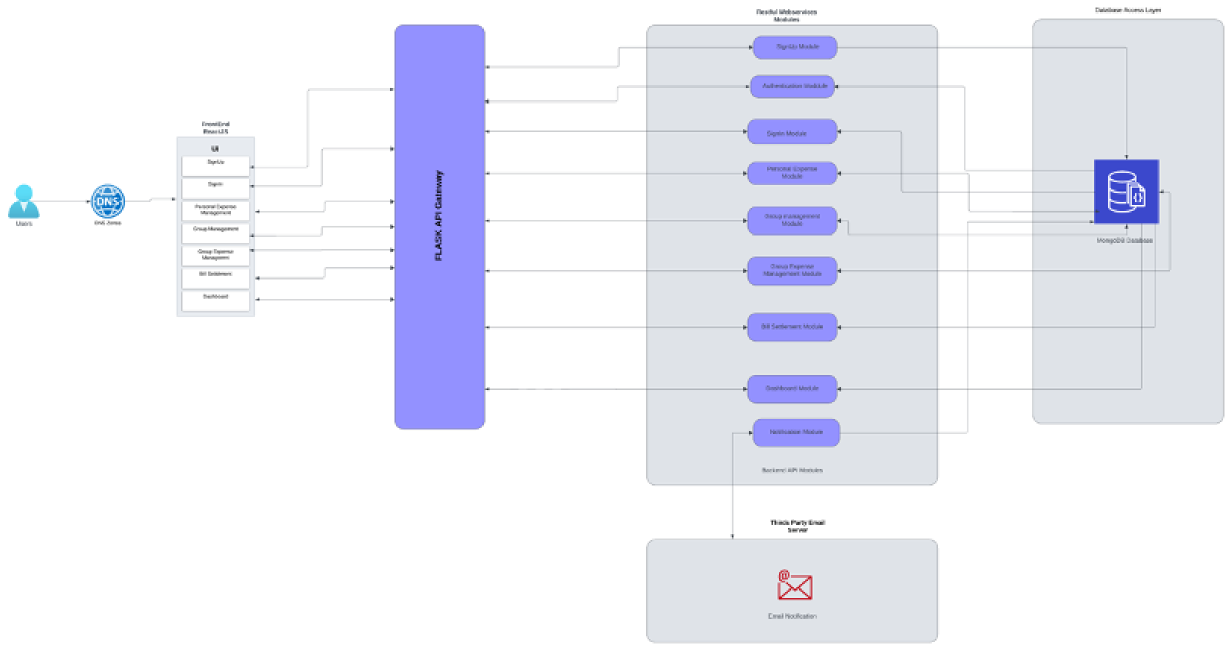
4.6 Email Services

Third-party Integrations:
Service Name: SendGrid API
Purpose: Facilitates sending email notifications to users.
Integration Points:

- User registration confirmation
- Group invitation notifications
- Bill settlement notifications

Data Shared: User email addresses, and email content (based on the type of notification being sent).
Endpoint: Example https://api.sendgrid.com/v3/mail/send

5. System Architecture and Overview



The architecture diagram presents a clear view of the system's structure, showing how different components interact to provide a seamless user experience.
Users: The users interact with the system through a frontend interface. Their requests are routed by DNS zones to ensure they reach the appropriate destination within the system.

- **Frontend ReactJS**: This is the user interface layer, built using ReactJS. It provides various features like UI for user sign-up, sign-in, personal expense management, group management, group expense management, bill settlement, and a dashboard.

- **FLASK API Gateway**: Serving as an intermediary, this layer receives requests from the frontend and routes them to the appropriate backend module. This gateway ensures that each request is directed to the correct service for processing.

- **Restful Web Services**: This layer is segmented into several modules, each responsible for a specific functionality:

  - SignUp Module: Handles user registrations.
  - Authentication Module: Validates user credentials and ensures secure access.
  - SignIn Module: Manages user logins.
  - Personal Expense Module: Enables users to manage their individual expenses.

o Group Management Module: Facilitates the creation and management of expense groups.
o Group Expense Management Module: Helps users in managing shared expenses within a group.
o Bill Settlement Module: Offers features for settling bills and clearing dues.
o Dashboard Module: Provides an overview of the user's financial activities.
o Notification Module: Responsible for sending alerts and updates to users.

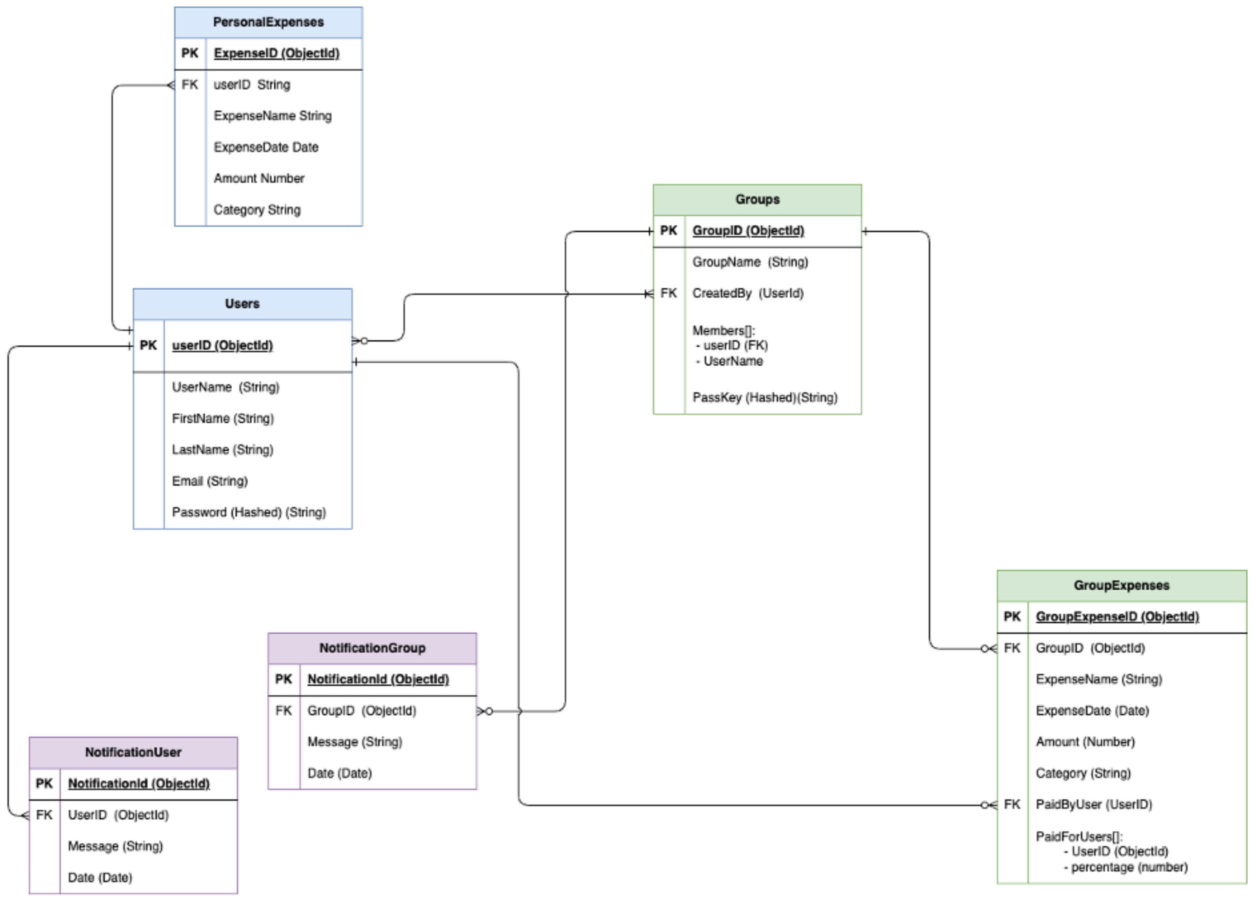
- **Third-Party Email Server**: An external service utilized for sending email notifications to users. The interaction with this service is primarily managed by the Notification Module.

- **Database Access Layer**: Acts as an abstraction layer, allowing the Backend API modules to interact with the MongoDB database without directly dealing with the intricacies of database operations.

o MongoDB Database: This is the system's primary data storage solution. It is connected to the Backend API modules, which perform CRUD (Create, Read, Update, Delete) operations based on user interactions.

6. Data Design

6.1 Database Schema



6.1.1 PersonalExpenses

Represents the expenses incurred by an individual user.
Attributes:

- ExpenseID: Unique identifier for each personal expense.
- userID: Foreign key linking to the Users table, indicating which user made the expense.
- ExpenseName: The name or description of the expense.
- ExpenseDate: The date when the expense was made.
- Amount: The cost of the expense.
- Category: The category or type of expense (e.g., food, transportation).

6.1.2 Users

Represents the registered users in the system.
Attributes:

- userID: Unique identifier for each user.
- UserName: Username chosen by the user.
- FirstName: User's first name.
- LastName: User's last name.
- Email: User's email address.
- Password: User's password, stored in a hashed format for security.

6.1.3 Groups

Represents groups that users can create or join to manage group expenses.
Attributes:

- GroupID: Unique identifier for each group.
- GroupName: Name of the group.
- CreatedBy: Foreign key linking to the Users table, indicating who created the group.
- Members[]: An array or list of user IDs, representing the members of the group.
- PassKey: A hashed string, possibly used to join the group or verify membership.

6.1.4 GroupExpenses

Represents the expenses made as a group.
Attributes:

- GroupExpenseID: Unique identifier for each group expense.
- GroupID: Foreign key linking to the Groups table, indicating to which group the expense belongs.
- Other attributes (ExpenseName, ExpenseDate, Amount, Category) are similar to PersonalExpenses.
- PaidByUser: Foreign key linking to the Users table, indicating who paid for the group expense.
- Users[]: An array or list that specifies which users are associated with this group expense, and their respective contribution percentages.

6.1.5 NotificationGroup

Represents notifications sent to groups.
Attributes:

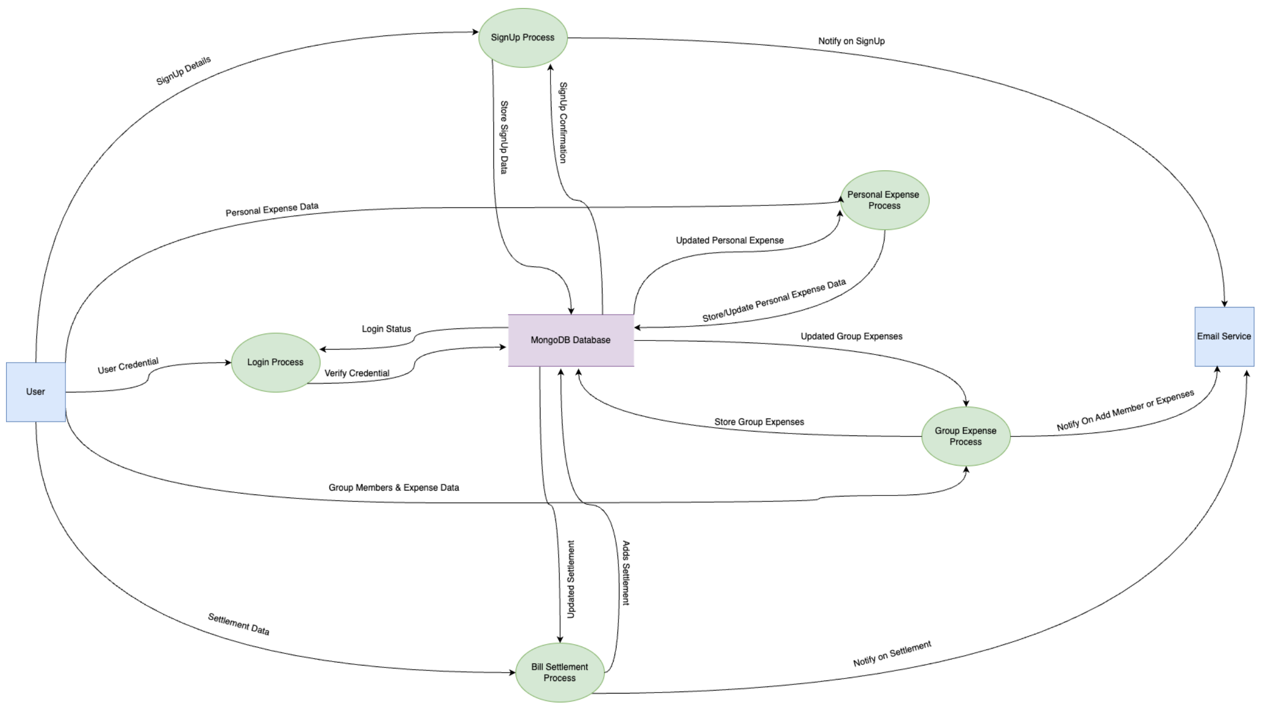
- NotificationId: Unique identifier for each notification.
- GroupID: Foreign key linking to the Groups table, indicating to which group the notification is addressed.
- Message: The content of the notification.
- Date: The date when the notification was sent.

6.1.6 NotificationUser

Represents notifications sent to individual users.
Attributes:

- NotificationId: Unique identifier for each notification.
- UserID: Foreign key linking to the Users table, indicating which user the notification is addressed to.
- Other attributes (Message, Date) are similar to NotificationGroup.

6.2 Data Flow Diagram



The DFD illustrates the flow of data within an expense management system:

- User Interaction:
    - A user can initiate a Signup Process. Once they successfully sign up, the Email Service is triggered to notify them of the successful registration.
    - Users engage with the Login Process by providing their credentials. The system verifies these credentials and provides a login status.

- Expense Tracking:
    - Users can submit Personal Expense Data, which is processed and then stored or updated in the MongoDB database.
    - There's a Group Expense Process for managing expenses in groups. When group expenses are updated, they're stored in the database. If there are any significant changes,

the Email Service sends notifications to old and new group members about these expenses.

- Data Storage:
  - The MongoDB Database is central, storing login statuses, personal expenses, group members & expense data, and other pertinent information.

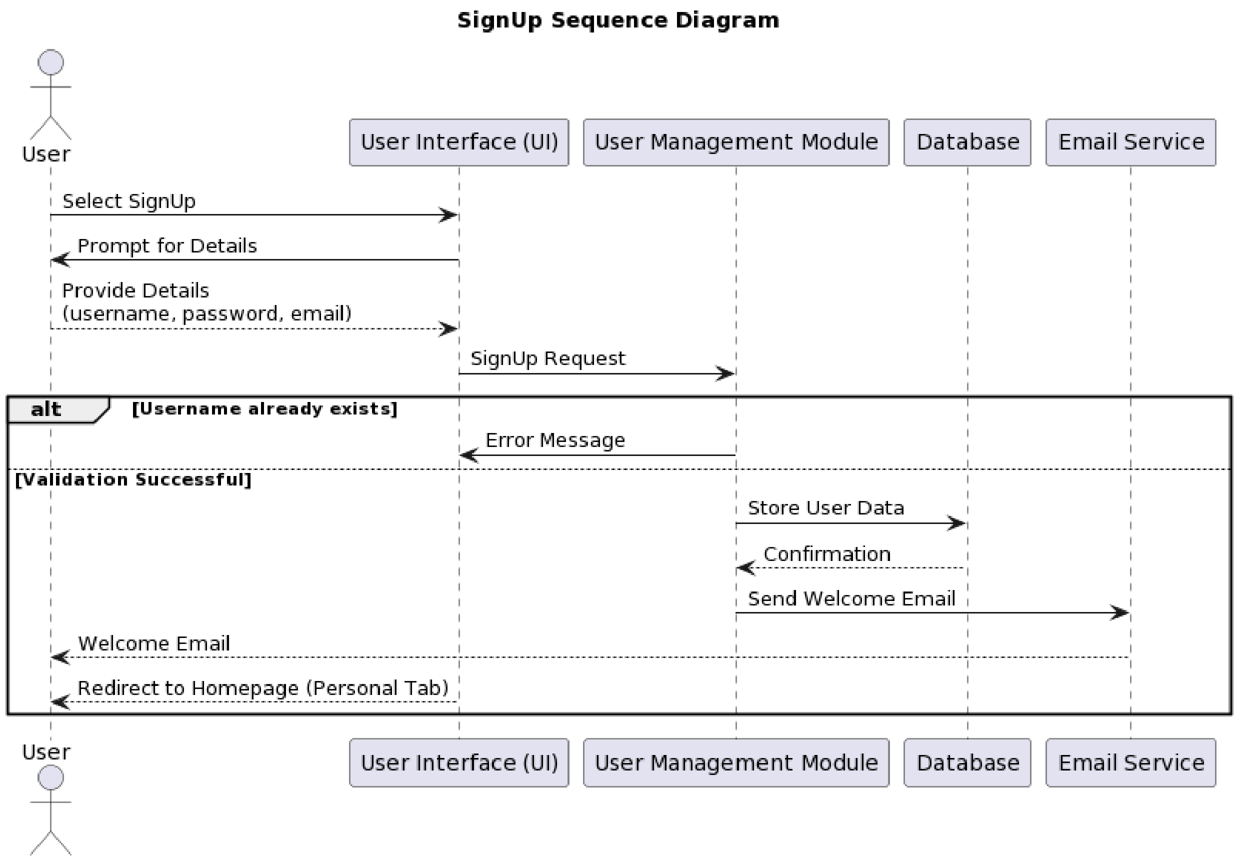
- Settlement:
  - Settlement data undergoes a Bill Settlement Process, after which the Email Service notifies users of any settlements or updates related to bills.

- Email Services
  - The Email Service plays a crucial role in this system, acting as the communication bridge between the application and the users, ensuring they're notified of essential actions or updates.

7. Interaction and Behavioral Design

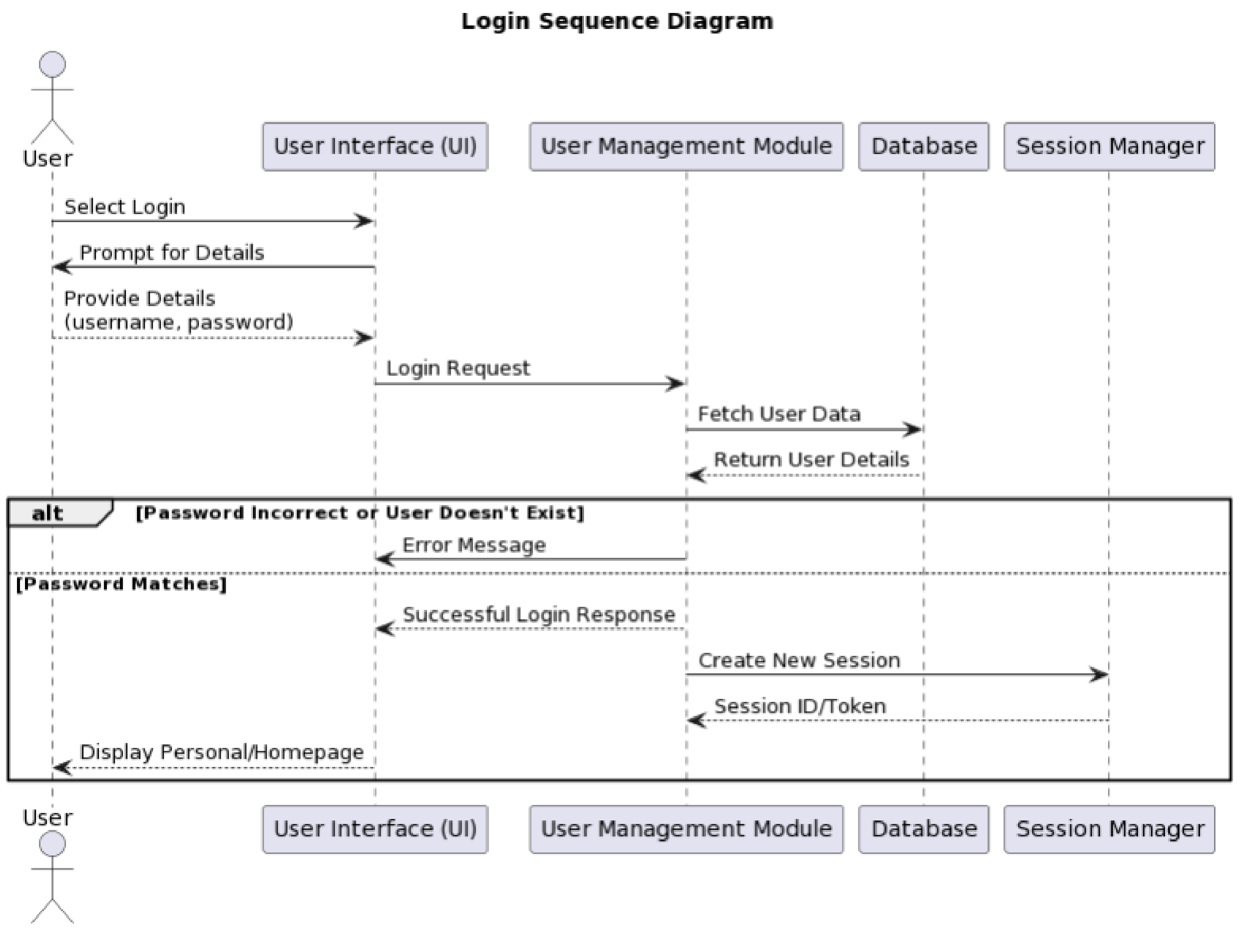
7.1 Sequence Diagram

7.1.1 SignUp Page



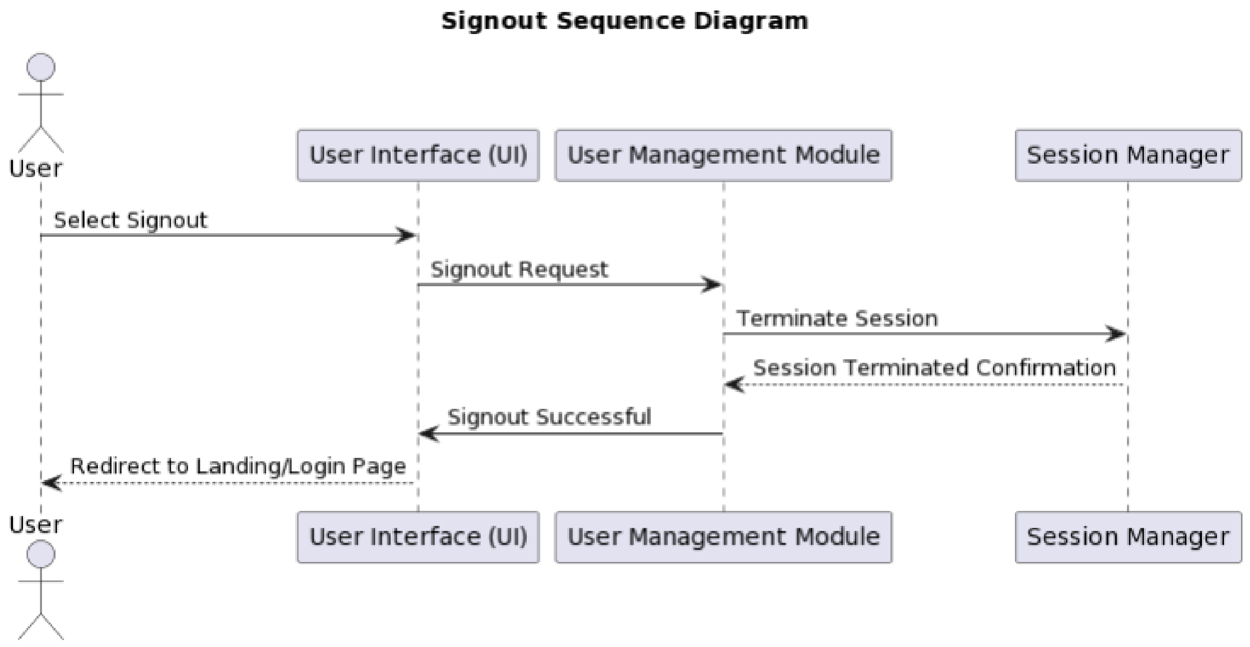
When a user initiates the signup operation, they are prompted to provide their details. These details undergo a validation process. If the validation is successful, the user's data is stored in the database, and a welcome email is dispatched to the user. However, if the validation fails, an error message is displayed to the user, prompting them to re-enter their details correctly.

7.1.2 Login operation



When a user initiates the sign-in process, they enter their credentials. These credentials are then verified. If the verification is successful, the user is granted access and redirected to their personal tab. However, if the credentials do not match or are invalid, an error message is displayed, and the user is prompted to try signing in again.

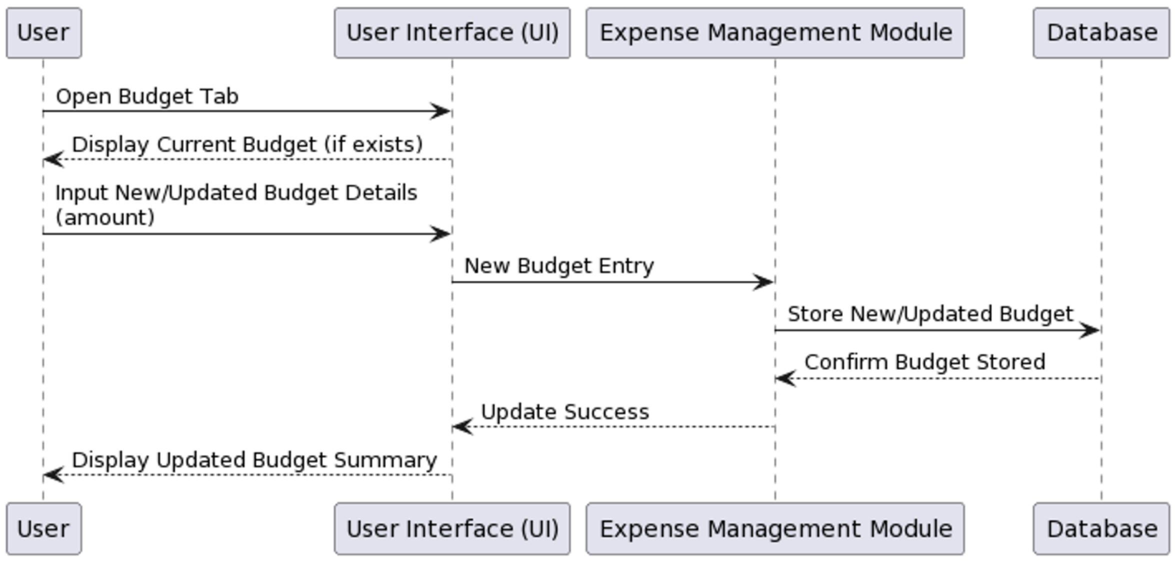
7.1.3 Sign-Out Sequence



When a user initiates the sign-out process, they select the 'Signout' option. This action triggers a signout request to the User Management Module. Subsequently, the Session Manager is instructed to terminate the user's active session. Once the session is successfully terminated, a confirmation is sent back to the User Management Module, and the user is informed of the successful sign-out. Finally, the user is redirected to the landing or login page.

7.1.4 Set Budget Sequence

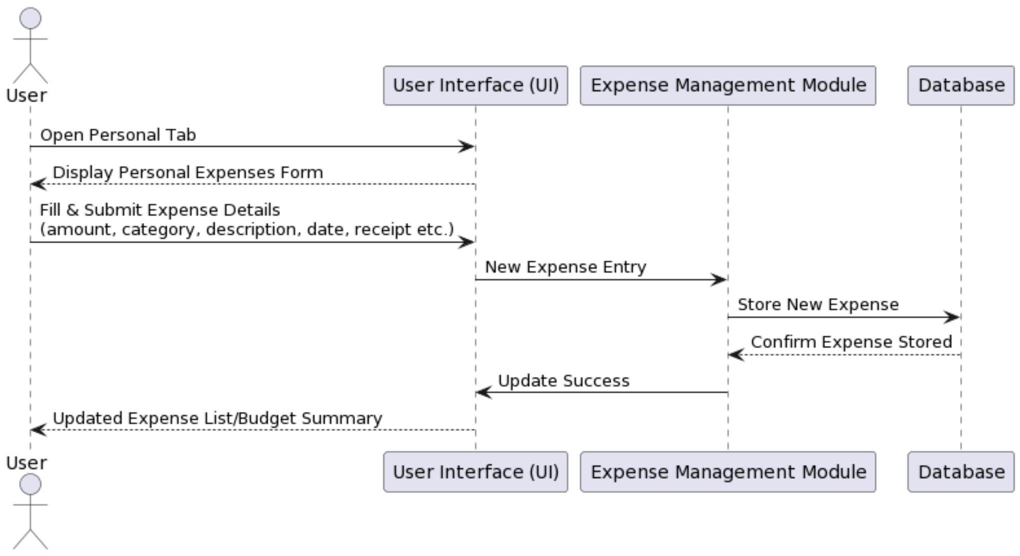
Setting and Editing Budget Sequence Diagram


User
User Interface (UI)
Expense Management Module
Database
Open Budget Tab
Display Current Budget (if exists)
Input New/Updated Budget Details (amount)
New Budget Entry
Store New/Updated Budget
Confirm Budget Stored
Update Success
Display Updated Budget Summary

A user can perform edit/set budget operations on their personal expenses using this sequence diagram. On budget update, the budget summary will be updated.

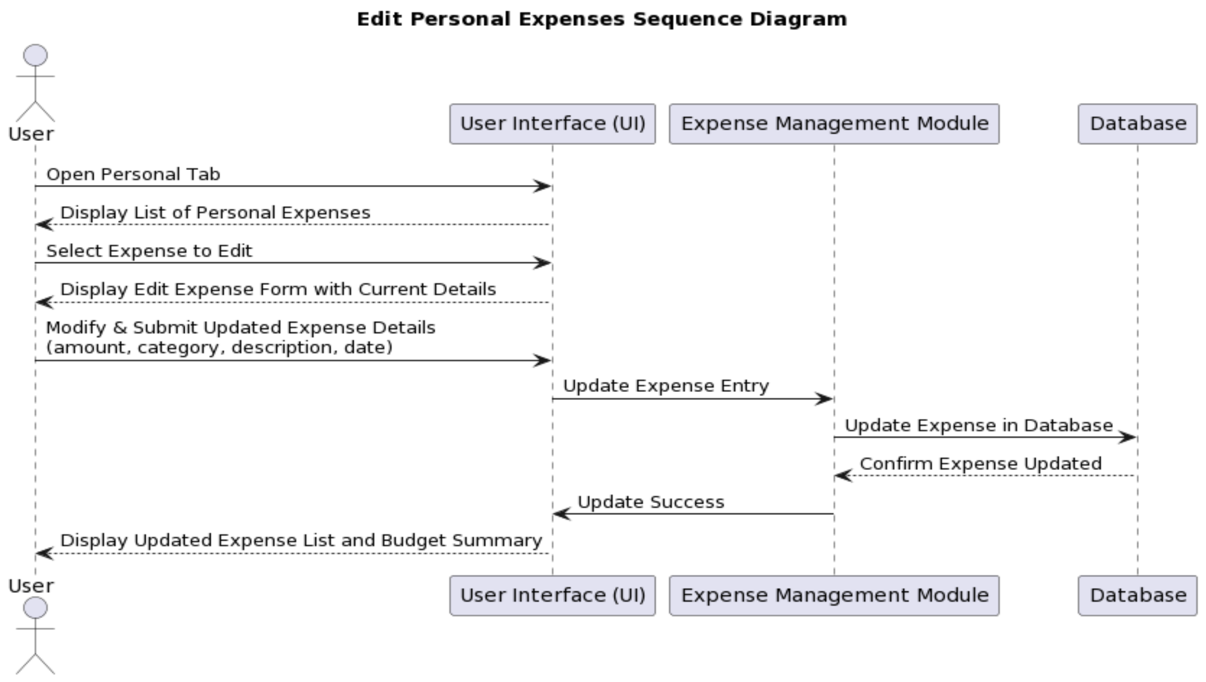
7.1.5 Adding Personal Expenses

Adding Personal Expenses Sequence Diagram


User
User Interface (UI)
Expense Management Module
Database
Open Personal Tab
Display Personal Expenses Form
Fill & Submit Expense Details (amount, category, description, date, receipt etc.)
New Expense Entry
Store New Expense
Confirm Expense Stored
Update Success
Updated Expense List/Budget Summary

The user can add personal expense operations in their personal expense tab by entering details of the expense. The details will be stored, and the updated details will be rendered in the budget summary and the expense list.

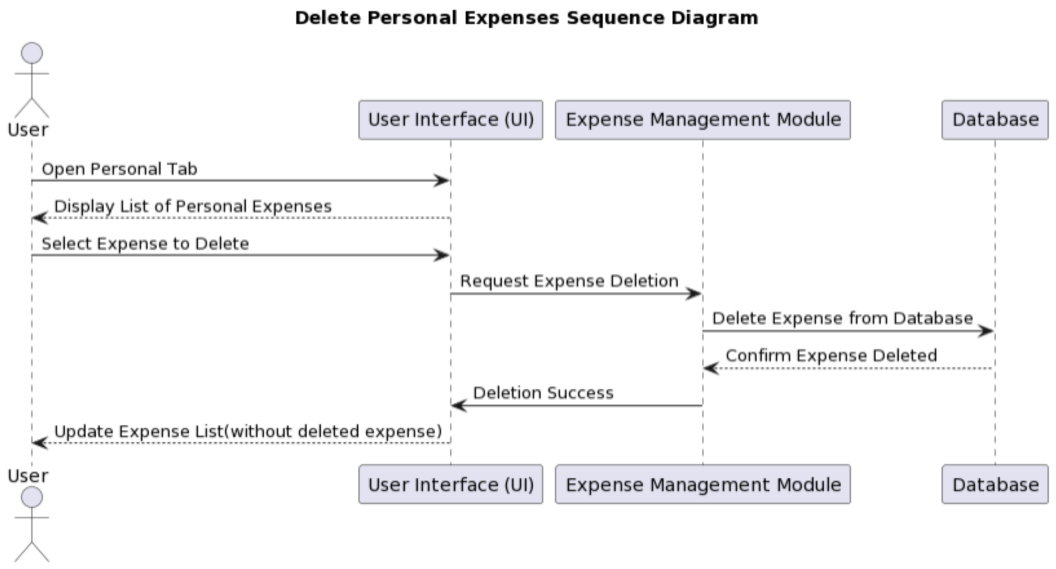
7.1.6 Edit Personal Expense



Edit Personal Expenses Sequence Diagram
User
User Interface (UI)
Expense Management Module
Database
Open Personal Tab
Display List of Personal Expenses
Select Expense to Edit
Display Edit Expense Form with Current Details
Modify & Submit Updated Expense Details (amount, category, description, date)
Update Expense Entry
Update Expense in Database
Confirm Expense Updated
Update Success
Display Updated Expense List and Budget Summary

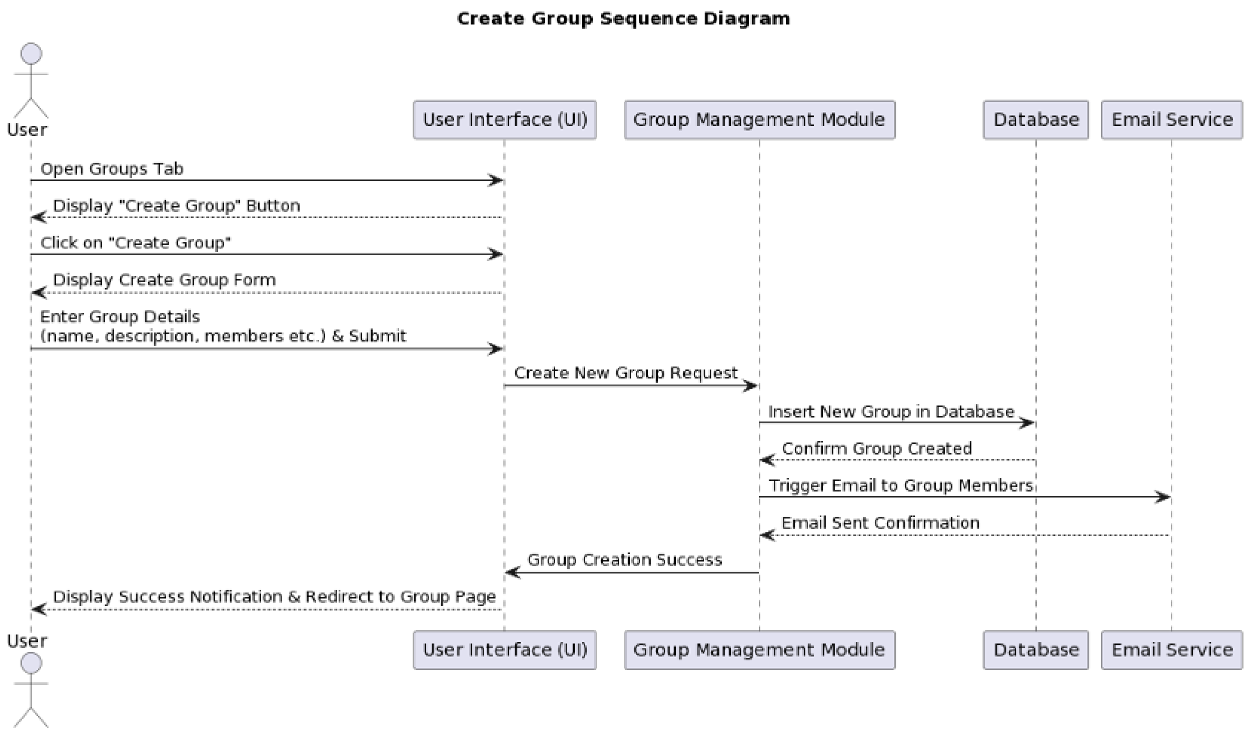
The user can perform an edit expense action by clicking on the expense to edit. The user can reenter the expense which will be updated on the database and the updated details will be rendered on the budget summary and expense list.

7.1.7 Delete Personal Expenses


Delete Personal Expenses Sequence Diagram
User
User Interface (UI)
Expense Management Module
Database
Open Personal Tab
Display List of Personal Expenses
Select Expense to Delete
Request Expense Deletion
Delete Expense from Database
Confirm Expense Deleted
Deletion Success
Update Expense List(without deleted expense)

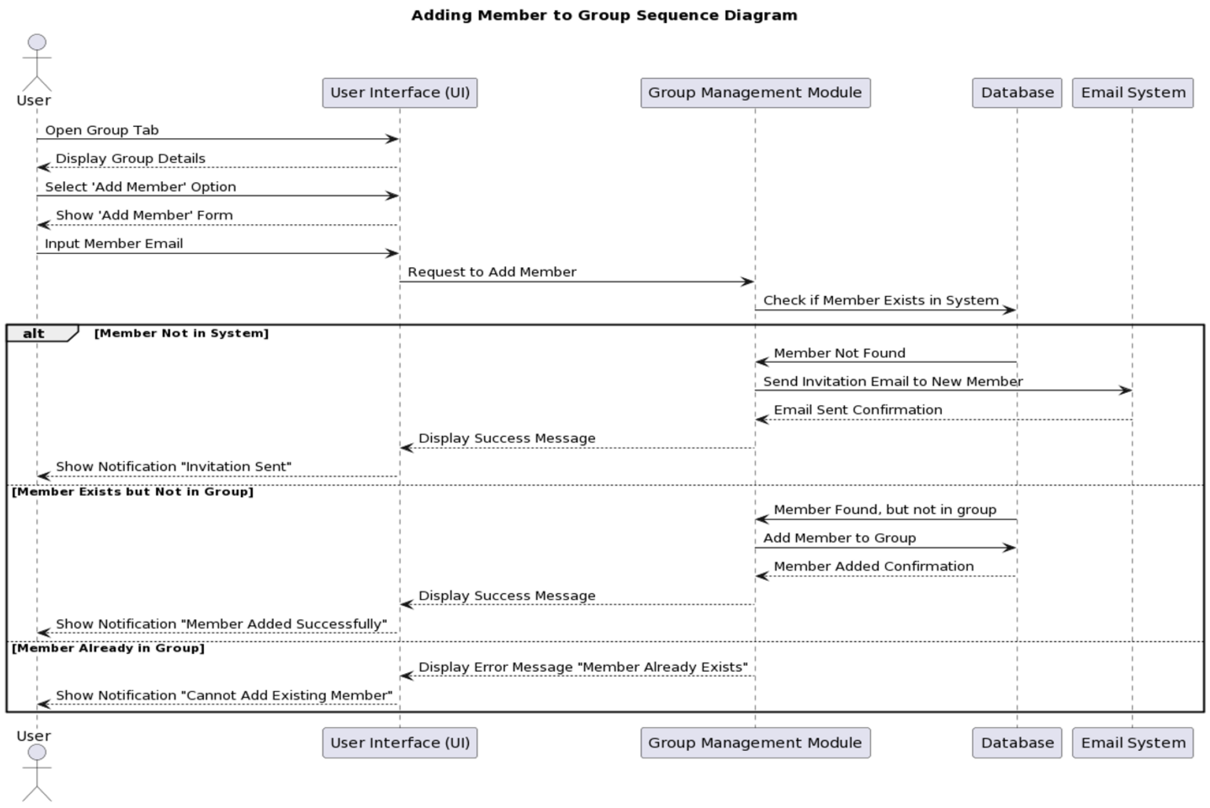
The user can delete an expense by clicking on the expense to delete. Upon the request, the delete action will be performed on the database and the updated details will be rendered in the expense list and budget summary.

7.1.8 Create Group


Create Group Sequence Diagram
User
User Interface (UI)
Group Management Module
Database
Email Service
Open Groups Tab
Display "Create Group" Button
Click on "Create Group"
Display Create Group Form
Enter Group Details
(name, description, members etc.) & Submit
Create New Group Request
Insert New Group in Database
Confirm Group Created
Trigger Email to Group Members
Email Sent Confirmation
Group Creation Success
Display Success Notification & Redirect to Group Page

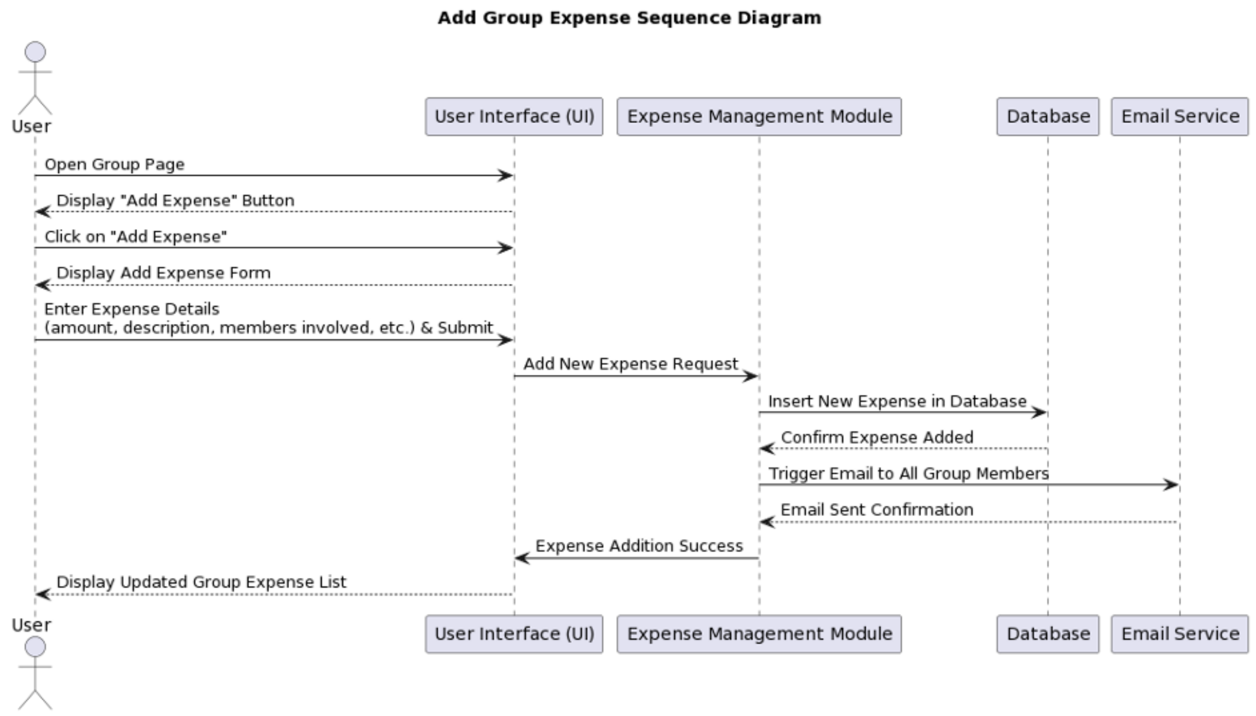
A user can create a new group by clicking on the create group button and by providing details of the group. The details will be passed to create a new group by the group management module to create a new group. A new group will be created and the user will be redirected to the new group page.

7.1.9 Adding Member To Group


Adding Member to Group Sequence Diagram
User
User Interface (UI)
Group Management Module
Database
Email System
Open Group Tab
Display Group Details
Select 'Add Member' Option
Show 'Add Member' Form
Input Member Email
Request to Add Member
Check if Member Exists in System
alt [Member Not in System]
Member Not Found
Send Invitation Email to New Member
Email Sent Confirmation
Display Success Message
Show Notification "Invitation Sent"
[Member Exists but Not in Group]
Member Found, but not in group
Add Member to Group
Member Added Confirmation
Display Success Message
Show Notification "Member Added Successfully"
[Member Already in Group]
Display Error Message "Member Already Exists"
Show Notification "Cannot Add Existing Member"

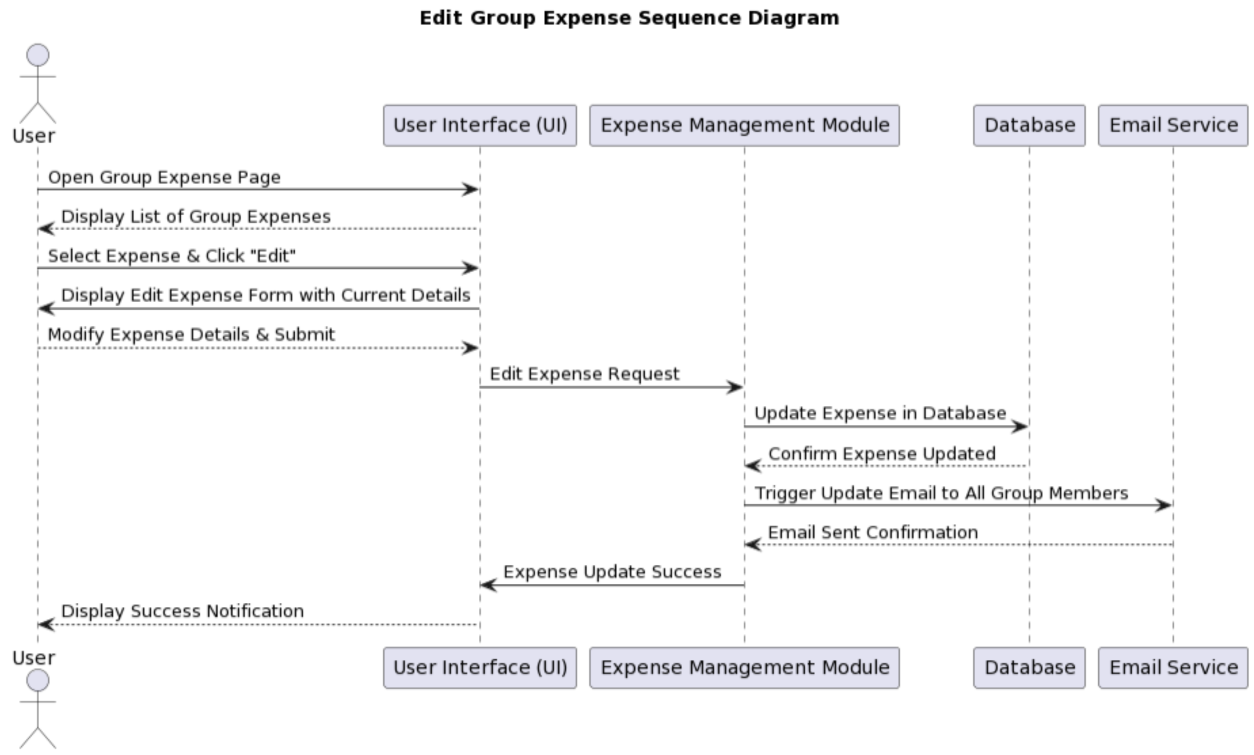
A user can add a member to a group by providing details of the member. The details will be validated on success and the member will be added to the group. However, if the member is not in the system, the member will be sent an email invite to join the group. Moreover, if the member is already in the group the user will be prompted with the error that the member is already in the group.

7.1.10 Add Group Expense


Add Group Expense Sequence Diagram
User
User Interface (UI)
Expense Management Module
Database
Email Service
Open Group Page
Display "Add Expense" Button
Click on "Add Expense"
Display Add Expense Form
Enter Expense Details
(amount, description, members involved, etc.) & Submit
Add New Expense Request
Insert New Expense in Database
Confirm Expense Added
Trigger Email to All Group Members
Email Sent Confirmation
Expense Addition Success
Display Updated Group Expense List

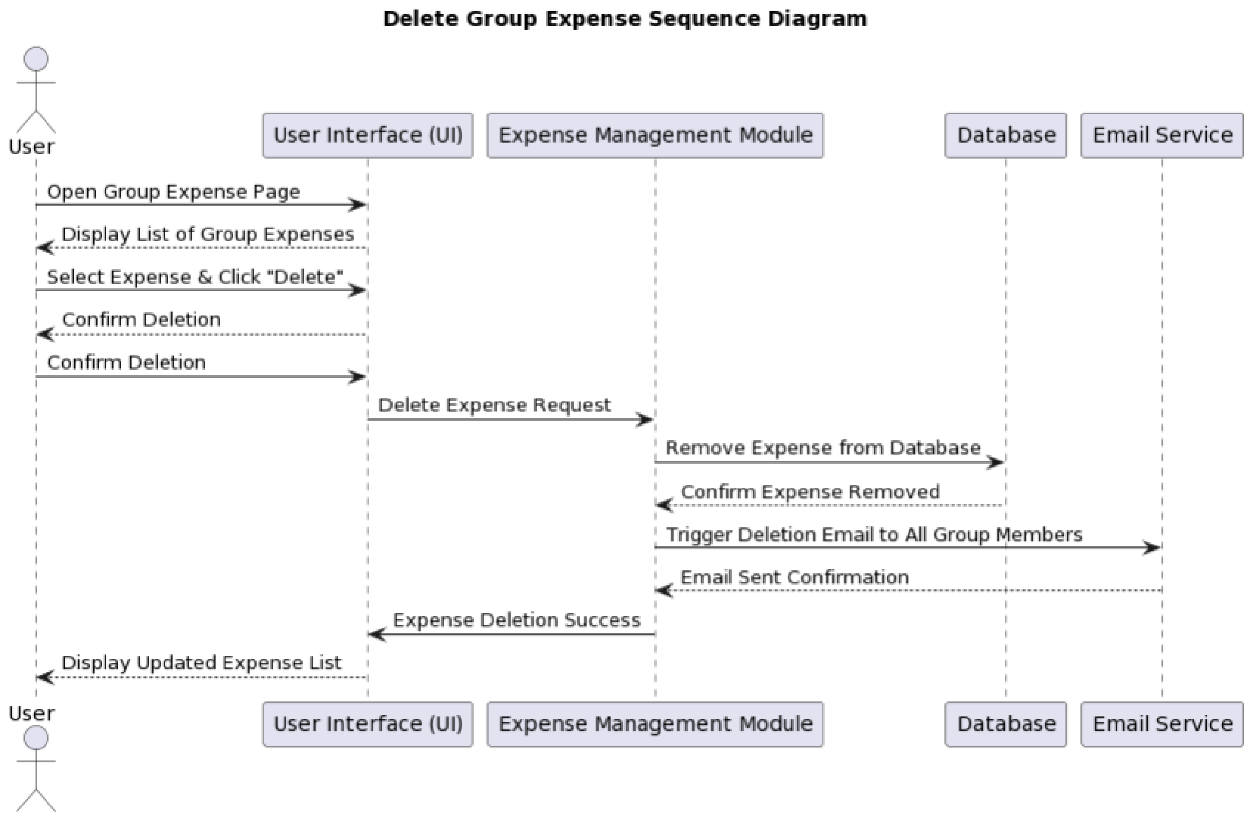
A user can add an expense to the group by entering the details in the group expense form by adding who paid, the amount, paid for whom, and their share. The expense details will be saved, and the updated details will be rendered in the expense list. All the group members will be prompted with email notification on the new expense that is added.

7.1.11 Edit Group Expense


Edit Group Expense Sequence Diagram
User
User Interface (UI)
Expense Management Module
Database
Email Service
Open Group Expense Page
Display List of Group Expenses
Select Expense & Click "Edit"
Display Edit Expense Form with Current Details
Modify Expense Details & Submit
Edit Expense Request
Update Expense in Database
Confirm Expense Updated
Trigger Update Email to All Group Members
Email Sent Confirmation
Expense Update Success
Display Success Notification

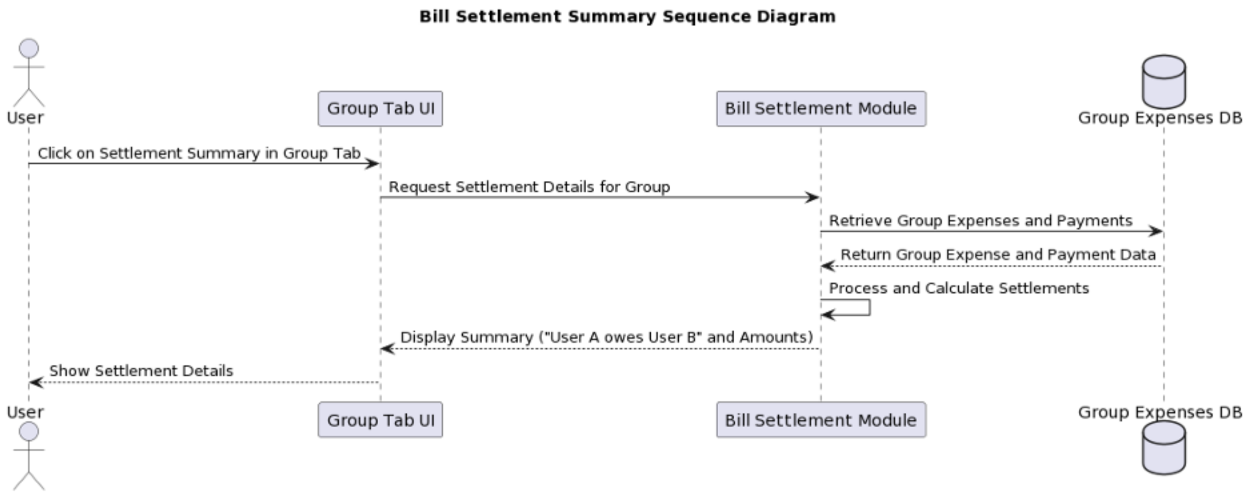
A User can edit the expense by clicking on the expense list and by filling out the update form. The updated operation will be performed, and the updated list will be rendered. An email notification will be sent for any update.

7.1.12 Delete Group Expense


Delete Group Expense Sequence Diagram
User
User Interface (UI)
Expense Management Module
Database
Email Service
Open Group Expense Page
Display List of Group Expenses
Select Expense & Click "Delete"
Confirm Deletion
Confirm Deletion
Delete Expense Request
Remove Expense from Database
Confirm Expense Removed
Trigger Deletion Email to All Group Members
Email Sent Confirmation
Expense Deletion Success
Display Updated Expense List

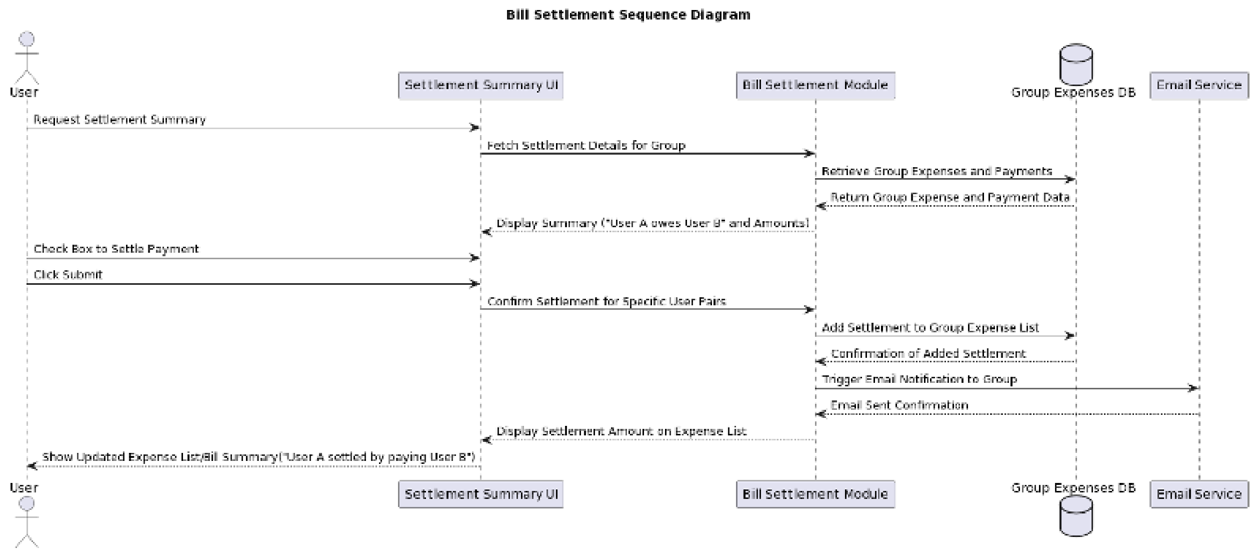
A user can delete the expense from the group by clicking on the expense list. The Delete operation will be performed and the updated expense list will be rendered. All members will be notified in the delete operation.

7.1.13 Bill Settlement Summary


Bill Settlement Summary Sequence Diagram
User
Group Tab UI
Bill Settlement Module
Group Expenses DB
Click on Settlement Summary in Group Tab
Request Settlement Details for Group
Retrieve Group Expenses and Payments
Return Group Expense and Payment Data
Process and Calculate Settlements
Display Summary ("User A owes User B" and Amounts)
Show Settlement Details

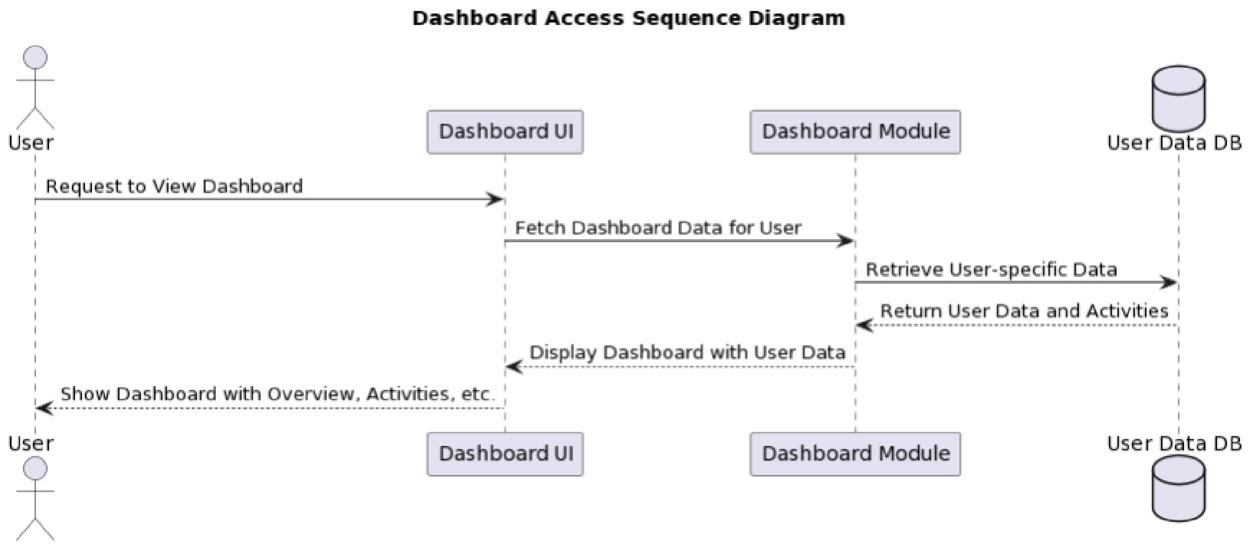
A user can request for a Settlement summary report. The summary report will be generated on who owes who and how much.

7.1.14 Bill Settlement


Bill Settlement Sequence Diagram
User
Settlement Summary UI
Bill Settlement Module
Group Expenses DB
Email Service
Request Settlement Summary
Fetch Settlement Details for Group
Retrieve Group Expenses and Payments
Return Group Expense and Payment Data
Display Summary ("User A owes User B" and Amounts)
Check Box to Settle Payment
Click Submit
Confirm Settlement for Specific User Pairs
Add Settlement to Group Expense List
Confirmation of Added Settlement
Trigger Email Notification to Group
Email Sent Confirmation
Display Settlement Amount on Expense List
Show Updated Expense List/Bill Summary("User A settled by paying User B")

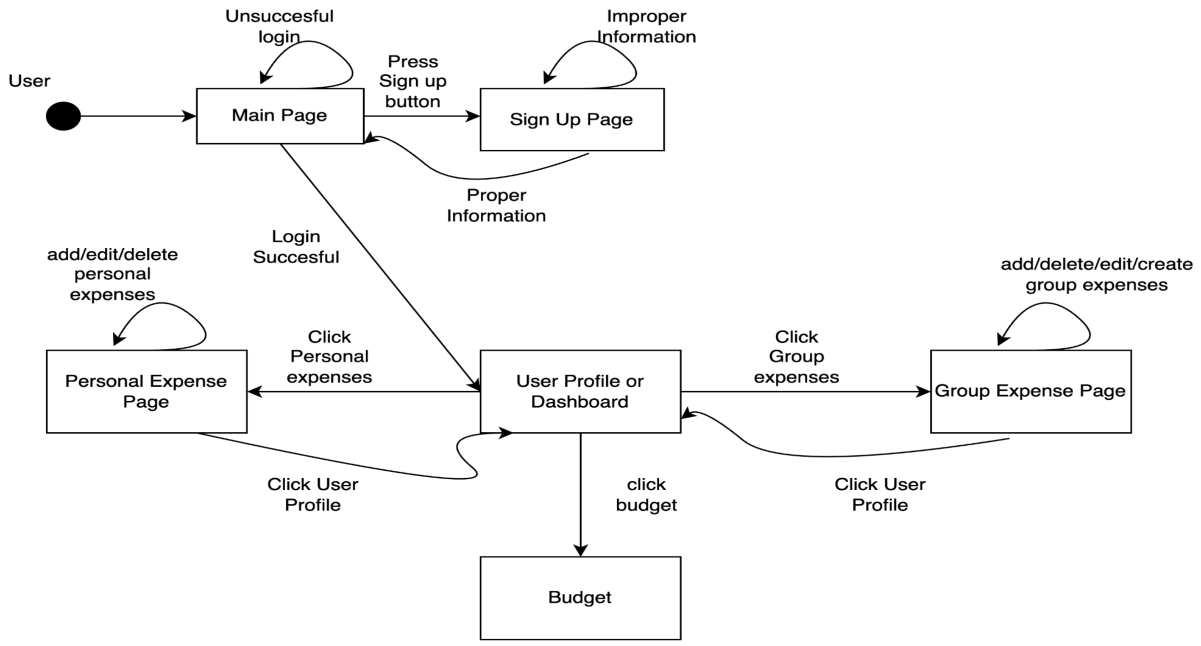
A user can settle the expenses by clicking on the check box on the bill settlement summary. Once the user clicks on submit the expense will be saved and the same will be added to the group expense list.

7.1.15 Dashboard Access


Dashboard Access Sequence Diagram
User
Dashboard UI
Dashboard Module
User Data DB
Request to View Dashboard
Fetch Dashboard Data for User
Retrieve User-specific Data
Return User Data and Activities
Display Dashboard with User Data
Show Dashboard with Overview, Activities, etc.

A user can access the dashboard for their personal and group expenses. The user can visualize their expense by different grains of data.

7.2 State Diagram



Main Page - A user when clicking the website application will land at this page. This page will display the information on login or signup activity that can be performed by the users. A user can click the sign-up button on this page and register themselves to the application for further usage. The user can also click login and login to the application with his/her credentials. They also have an option to change their password if they have forgotten their password. If the login is unsuccessful the user will land back at the same page.

Sign up Page - A user will land at this page if he clicks the sign-up button on the main page. This page will contain the information block for the user to enter for registering to access the website application. Once they give the proper information and meet the required criteria they will be sent an email notification to verify their account and be transferred to the Main page to login with their registered credentials.

User Profile or Dashboard - Once the login is successful in the main page the user will be directed to the Dashboard where the user can maintain his/her profile in the application. The user can view their aggregated expenses and their spending on this page. The user also has the option of viewing the total expenditure, Personal expenses, and Group expenses in a graphical form and also can view their budget limit.

Personal Expense Page - The user can click the personal expenses button to land on this page where they can input their expenses under different categories daily. The users can add any expense if they have missed any expense on a particular day, month, or year. They also can edit and delete their expenses at any point in time. The users can generate a report on their personal spending on the personal expense tracking page. The User can also return back to the dashboard to view any changes that were made in the personal expense tracking page.

Group Expense Page - The user can click the group expense page in the Dashboard page to be directed to the group expense tracking page. They have the option to create a group and send invitation links as admins to the people they want to add to the group. They also have access to remove any user from the group as admin. The users can perform the same functionalities that they were able to perform in the personal expense tracking page for the Group expense page. Eg add, edit, delete expenses. Apart from this, the users have an additional feature in the application for splitting the bills between each of the members to achieve simplicity in tracking his/her expenses. The user can also track back to the dashboard page to view any changes made in the group expense tracking page.

Budget Tracker - The user can set a budget limit and track the monthly budget set. The user can view this in any of the pages and can edit the budget limit whenever required.

8. Security Design

8.1 Authentication and Authorization

8.1.1 JWT-based Authentication

JSON Web Tokens (JWT) are used to authenticate users in the system. Upon successful login, the system generates a JWT for the user. This token, which contains encoded user details, is sent back to the user and must be included in the header of subsequent requests. This way, the backend can verify the identity of the user making a request.

8.1.2 Restricted Access

Only authenticated users, verified through JWT, have the capability to add expenses or perform other sensitive operations. Unauthorized attempts are promptly denied, ensuring that malicious actors cannot manipulate data or gain unauthorized insights.

8.2 Data Privacy and Protection

8.2.1 Password Security

User passwords are not stored in plain text. Instead, they are hashed using bcrypt, a robust password-hashing algorithm. This means that even if there were an unlikely breach, the actual passwords would remain undisclosed. Furthermore, bcrypt has built-in salting, which makes it computationally difficult for attackers to reverse engineer the original password from the hash.

8.2.2 Financial Data Safety

While our system facilitates expense management, we prioritize user trust and safety by not storing sensitive financial details like bank account numbers, credit card details, or other personal financial information. Instead, our focus remains on the management and tracking of expenses without delving into intricate financial data.

9. Deployment Design

9.1 Deployment Architecture

- **Cloud-Based Deployment:** Utilize cloud platforms like AWS or GCP to ensure our system's scalability and reliability.
- **Containerization:** Implement Docker for consistent deployment from development to production environments.
- **Orchestration:** Use Kubernetes for auto-scaling, load balancing, and fault tolerance(If required).
- **Database Management:** Employ managed services for databases (e.g., MongoDB Atlas on AWS) for enhanced availability and scalability.

9.2 CI/CD Processes

- **Version Control:** Implement Git for efficient code management and collaboration.
- **Continuous Integration:** Use tools like Jenkins or Travis CI for automated code compilation and integration testing post each code push.
- **Continuous Deployment:** Harness Jenkins or Spinnaker to automate deployment after successful integration, ensuring smooth updates to production.
- **Automated Testing:** Integrate automated tests to confirm system stability post-deployment.
- **Real-Time Monitoring:** Set up Prometheus and Grafana to flag performance issues instantly, fostering quick optimization actions.

10. References

- https://www.bellevuecollege.edu/wp-content/uploads/sites/135/2019/04/SDD_RoadTrip.pdf
- IEEE standards (IEEE Std 1016 – 2009)
- Week 8 - Software Architecture.pdf