RoomieBuddy: A mobile application that will facilitate shared living

A Project Report
Presented to
The Faculty of the College of
Engineering

San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree

Master of Science in Software Engineering

By

Divya Mohan
Lasya Bheemendra Nalini
Shilpi Soni
Varsha Suresh

December 2022

APPROVED

Professor Gopinath Vinodh, Project Advisor

Professor Dan Harkey, Director, MS Software Engineering

Dr. Rod Fatoohi, Department Chair

ABSTRACT

RoomieBuddy: A mobile application that will facilitate shared living

By Divya Mohan, Lasya Bheemendra Nalini, Shilpi Soni, Varsha Suresh

According to a study published by Pew Research Center, nearly 79 million American adults lived in shared accommodation in 2017. Sharing accommodation can minimize living costs and help widen the social circle. Despite its financial and social appeal, shared living can lead to conflicts regarding the division of space and expenses. Transparent tracking of the shared costs, division of household chores, and coordination of roommates' schedules will help to keep conflicts at bay.

Addressing common roommate issues through in-person meetings has its challenges. To begin with, people may find it challenging to track the distribution of money and chores in person. Secondly, missing details during discussions can lead to unfair splits. Finally, this process is time-consuming and inefficient. Today, these tasks can be performed electronically with the help of mobile applications. However, multiple applications must be installed, as it is hard to find a single application that acts as a one-stop shop for all roommates' needs.

The goal of our application, RoomieBuddy, is to address these issues by providing an expense tracker that performs a fair split of the everyday household bills that occupants share. It will delegate chores to roommates, remind them when it is their turn and notify everyone on completion of tasks. It will allow the creation of shared grocery lists to stay on top of everyday needs. We seek to develop an easy-to-use mobile application to manage and address common cohabitation issues.

Acknowledgments

We would like to express our heartfelt gratitude to our Project Advisor, Professor Gopinath Vinodh, who mentored us throughout our project by providing valuable feedback and guidance.

We would also like to thank Professor Dan Harkey for his insightful comments and guidance.

Table of Contents

List of Figures

List of Tables

Chapter 1. Project Overview

1.1 Introduction

In today's fast paced life, a lot of our tasks like shopping for groceries and conversations with loved ones are performed while on the go, using mobile applications. Similarly, an application for the management of cohousing necessities can abstract away the details of sharing chores and expenses and lead to fewer awkward social situations between roommates. Oral and tacit agreements between roommates regarding household responsibilities are oftentimes forgotten and hard to keep track of. Using an application to record such information can therefore encourage reliability and enforce transparency between all members of the household.

The RoomieBuddy application will be a single, multi-use mobile application that can help roommates manage expenses, chore schedules and room related to-do lists thereby reducing the amount of time spent in navigating different applications to track details. Further, communication pertaining to the need for certain chores or grocery list items can be done effectively by adding notes and comments. A shared view of the roommates' availability can also make planning group events easier. All details of the same household can effectively be tracked in a single place.

1.2 Proposed Areas of Study and Academic Contribution

Shared living among young people is a pragmatic approach to adults facing housing insecurities. Natalier K. (2003), mentions that despite the advantages to such a model like decreased rent, the absence of written rules for the shared space leads to tensions and conflicts. Conflicts in households arise when there are differences in interests and opinions (Curseu P.L et al. 2012). Different expectations of cleanliness, freeloading by unequal task and cost division are primary sources of conflicts found in a shared household (Clark V., Tuffin K., & Bowker N., 2020). One of the ways to keep the tensions at bay is to identify equitable methods to manage disagreements. Nicolò A., et al. (2017), state that a divide-and-choose mechanism that advocates for a fair division of resources can be used to combat this problem.

As stated by Clark V. et al. (2018), shared money and costs allocated for shared property can be misused by unethical roommates, which leads to distrust in the relationship. These can be mitigated by setting boundaries and proper communication. Mause K. (2008), also mentions the struggles of shared housing such as overconsumption of shared utilities and unequal division of expense. This can be solved by creating written agreements. However, digital methods to track budget, expenses, and investments are proven to be more effective than the handwritten method according to Lewis M. & Perry M. (2019). Hence managing expenses digitally can mitigate the problem of unfair division of bills.

One of the most significant issues in shared space is ensuring that all the roommates do their fair share of keeping the apartment clean. Many otherwise happy roommate relationships have quickly gone cold when all the parties are not on the same page about doing chores. The chore division problem deals with the fair and equitable allocation of duties among all the participants (Farhadi A. & Hajiaghayi M., 2017). Aziz H. et al. (2021), have proposed a Double Round-Robin procedure to efficiently analyze the fair allocation problems when the chores are indivisible. According to Bei X., Huzhang G., & Suksompong W. (2020) , the errands that all participants dread should not be eliminated but instead shared in a truthful and envy-free manner. Apart from chores and expense management, having a shared log of the activities helps to better track the chores. This can be achieved by having a shared calendar which can help make members aware of the schedule and is easily modifiable as highlighted by Plaisant C., et al. (2006). Additionally, assigning rewards to chores enhances the house members' desire to perform chores.

The existing literature shows the need to build a tool that can track chores and expenses. For ease of use, we aim to build a mobile application that will solve conflicts by tracking chores, expenses and allowing management of grocery lists and shared calendars.

1.3 Current State of the Art

Today, most people own and use their smartphones daily for various purposes, including connecting with friends and family, working, tracking schedules, and even entertainment. It is hoped that this trend will continue to grow and create a plethora of opportunities for mobile application development.

Developing advanced applications to solve target audience issues has become an integral part of mobile application development. In this project, we will be building our application using React Native. React Native is a modern mobile application development language that is extremely fast and flexible. Mobile applications built using React Native are synonymous with high-quality standards and cutting-edge technologies. React Native enables developers to build cross-platform applications on Android and iOS platforms. The look and feel of these applications are similar to that of native applications. In addition, React Native platform saves time and cost as separate codebases are not needed for multiple platforms.

Additionally, we will be using the Expo framework to build, deploy and accelerate iterations on Android or iOS applications. Expo contains a suite of tools and services built on the React Native platform, using which React Native applications can be easily published on application stores.

Applications like Dwell and Splitwise are currently available in the market for users to perform expense and other chore-related functions. However, the disadvantage is downloading multiple applications for comprehensively managing cohousing requirements. By combining the functionality available across these different applications, we provide a unified solution.

Chapter 2. Project Requirements

2.1 Functional Specifications

2.1.1 Login/Signup

- Using the Sign-up functionality, the users can create their accounts. Based on the admin privileges, the users will have a different level of access to the features. To sign up, the users must enter a valid first name, last name, email address, password and role.
- Using the log-in functionality, the existing users can return to this application. The users must enter a valid email address and password to access their account.
- Once the sign up or login is successful, the user will land up on a welcome page that will prompt the user to create a new account or join an existing group.
- The user details such as first name, last name, email id, password, and isAdmin will be stored in the MySQL database.

2.1.2 Expense Management

- A roommate will be able to create expenses with fellow roommates.
- They will be able to choose roommates with whom he/she/they want to share an expense.
- A roommate will be able to choose if they want to split expenses equally or by specifying the amount explicitly with selected roommates.
- Ability to edit expenses created by a roommate.
- Ability to delete expenses created by a roommate.
- A roommate will be notified on expense creation or edit if they are involved in an expense.
- A roommate can view all active and past expenses.
- A roommate can view the amount they owe and are owed by other roommates.
- A roommate can settle due payment to another roommate by entering a cash amount.
- A roommate will receive push notifications when others settle their payments or record expenses.

2.1.3 Chore Management

- A roommate will be able to create a chore.
- The roommate should be signed in using valid credentials to create/view/edit/delete chores.
- A chore can be created by adding description, due date, assignee, duration, and frequency of the chore. In the Dashboard, the roommate can view the chores as a list with a checkbox.
- Additionally, they can also view the chore details and mark it as complete. Completion of the chore does not remove the chore from the list.
- The roommate can also edit the chore to change the due date, description, duration etc. Editing a chore can be done from the dashboard or by going to the chore detail page.

- When a roommate adds/views/ edits a chore it is reflected on the calendar.
- The roommate will also receive reminders before the chore is due to make sure that the chore is not overlooked.

2.1.4 To Do and Grocery Lists

- Roommates will be able to create to-do lists within the application. List items can be added to each list. The individual list items can be marked as completed or deleted. Each list item can also have comments or notes attached to them. All the aforementioned actions can be performed by all roommates.
- During the creation of a list, a roommate can choose to model its type as a to-do or grocery. Grocery and to-do lists are implemented using the same entity in the backend database and are distinguishable in the front end by a difference in look and feel. This difference in styling is enabled by maintaining a “type” attribute for each list, the value of which can be set to grocery or to-do.
- Grocery lists will also display a list of recommendations of products that the roommates usually buy. This list will be informed by a backend Python based recommendation system.
- Push notifications will be generated when the lists are modified to alert all roommates of an update.

2.1.5 Room Management

- Using this feature, the users can create or join an existing room to communicate and coordinate with other roommates. If the create room button on the welcome page is clicked, the user will be redirected to the “create room” page. a The room name will be a mandatory field whereas description field will be optional. Once the room is created successfully, the user will be redirected to the home page.
- The user can also enter a valid room link in the welcome page and click on the join button. If the link is valid, the user will be redirected to the home page.
- In the home page, an overview of expenses will be displayed in the top section. Below this, all the upcoming chore lists will be displayed. At the bottom, the icons for accessing chores page, to-do list, calendar and expenses will be displayed. The UI of this functionality will be easy to navigate.
- At the top right corner of the home page, an icon for settings page will be present. On clicking, the user will be redirected to the settings page where the user can update first name, last name and password. Room and room related details can be accessed by clicking on the visit room button present in this page.
- In the room page, the user can edit the room name, description and group image and click on the update button. Below this, the user can view the list of members of this group along with options to view privilege, leave room and remove a member. The user can leave the group if there are no dues or chores assigned. Similarly, the user can remove a member if he/she has admin access.
- An invite link will be present at the bottom of the page. This link can be shared with friends or family members to invite them to the room.
- The users can log out of the application by clicking on the log out button in the settings page.

2.1.6 ***User Management***

- Users can choose to be an admin while signing up.
- Users with admin privileges can invite another user to join the room by using a shareable invite code.
- User can leave the room if all his dues are clear.
- Users can create rooms.
- Users can join rooms by using an invitation link.
- User can also delete his account when all his dues are paid.

2.2 Non-Functional Specifications

- Application to be made available on a mobile platform(iOS).
- Request made by user to be served within 2-3s response time.
- Data of each user should be secured.
- Application should have a backup mechanism to prevent any user data loss.
- Data of each group must be visible only to the group members.

Chapter 3. Project Architecture

3.1 Introduction

The RoomieBuddy application is built on a three layered architecture: User facing presentation layer, server side business logic layer and data layer.

3.2 Architecture Subsystems

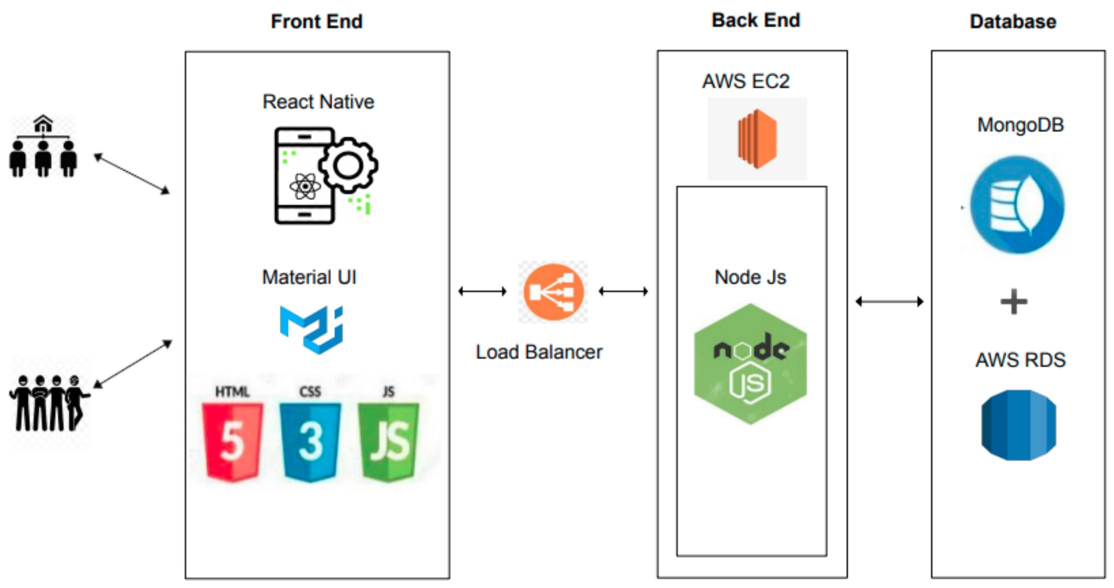


Figure 1. Architecture Diagram

3.2.1 FrontEnd subsystem

A single page mobile application built upon React Native will be developed and deployed as the front end. This application will serve as the user interface through which roommates will be able to perform various actions. It will interact with backend REST endpoints by sending json data over SSL. The application will also include local storage to support most frequently used functionalities. The application and updates to it will be downloadable from the App Store.

3.2.2 BackEnd subsystem

A Node JS web server will be deployed on AWS instances to serve as a backend for this application. The backend server will perform business logic and will be used to interact with the relational and non-relational databases storing the application data. The queries to the backend will be routed through load balancers to distribute load evenly.

3.2.3 ***Database subsystem***

Relational: User details and expense information to enforce ACID properties. Transactions will be enabled for updates to all expense and payment related record updates.

Non-relational: Group, Chore and Schedule details will be stored as no ACID properties need to be ensured. BASE properties will suffice for this.

Redis Caching (Optional): We will introduce a cache so that frequently accessed data can be fetched sooner and the database will not be hit for each operation performed on the application.

Chapter 4. Technology Descriptions

4.1 Client Technologies

4.1.1 React Native v0.68.2

In this project, we are building our application using React Native v0.68.2. React Native is a modern mobile application development language that is extremely fast and flexible. Mobile applications built using React Native are synonymous with high-quality standards and cutting-edge technologies. React Native enables developers to build cross-platform applications on Android and iOS platforms. The look and feel of these applications are similar to that of native applications. In addition, React Native platform saves time and cost as separate codebases are not needed for multiple platforms.

Additionally, we will be using the Expo framework to build, deploy and accelerate iterations on Android or iOS applications. Expo contains a suite of tools and services built on the React Native platform, using which React Native applications can be easily published on application stores.

4.1.2 React Paper v4.12.5

React Native Paper enables the creation of appealing mobile and web interfaces using high-quality cross-platform components. It is responsive, quick, and reliable regardless of platform. The library is always being updated and expanded.

When required, the components preserve native platform-specific functionality. It increases readability, reduces user confusion, and assures a better UX.

We choose React paper library because it offers ready to use components which can also be customizable.

4.2 Middle-Tier Technologies

4.2.1 NodeJS v16.13.1

Node.js is a free and open source server environment that runs javascript on the server.

It enables developers to quickly transfer data between the server-side and client-side. It is a popular framework for developing quick and scalable mobile and online apps. Node is an open-source server environment built on Google Chrome's JavaScript runtime.

We choose node.js because our applications has more I/O operations and does not require heavy computation.

4.2.2 PassportJWT v4.0.0

A Passport strategy for authenticating with a JSON Web Token. This module lets you authenticate endpoints using a JSON web token. It is intended to be used to secure RESTful endpoints without sessions.

4.2.3 *Hapi Joi v17.2.1*

Hapi Joi is an object schema description language and validator for JavaScript objects. With Hapi Joi, we create blueprints or schemas for JavaScript objects to ensure validation of key information. We are using this schema here to validate the request body.

4.3 Data-Tier Technologies

4.3.1 *SQL - AWS RDS*

User details and expense data will be persisted in an AWS RDS to enforce ACID properties. These data needed to have ACID properties to ensure the correctness and consistency of a database. We also wanted consistent results for expense and user related data hence we chose SQL for these features.

4.3.2 *NOSQL - MongoDB Atlas v5.0.13*

For functionalities such as Chore, Room and schedule we decided to use Mongodb as no ACID properties were essential. These features focus on eventual consistency, hence we decided to go with BASE properties here. Also we chose Mongodb as it can easily store structured and unstructured data.

4.4 Cloud Technologies

4.4.1 *AWS EC2 Instance:*

Amazon EC2 is a web service that provides secure, resizable compute capacity in the cloud. EC2 offers many options that enable you to build and run virtually any application.

For the deployment of the backend servers, multiple AWS EC2 instances will be created using the AWS console.

4.4.2 *Amazon S3 Bucket:*

Amazon S3 acts as a virtual device for storing and retrieving data. S3 is an object-based storage service for images, videos and documents etc.

We are using an Amazon S3 bucket to handle static files in our application such as profile images.

4.4.3 *Elastic Load Balancer:*

Elastic Load Balancing automatically distributes your incoming traffic over numerous targets in one or more Availability Zones, such as EC2 instances, containers, and IP addresses. It checks the health of its registered targets and sends traffic only to those that are in decent health.

Load Balancers, created on the AWS console, will be enabled for the multiple backend servers to share the load.

Chapter 5. Project Design

The Roomie Buddy application can be viewed as a collection of scalable modules. The application is divided into separate functional modules such as Signup, Login, Group Creation, Expense and payment management, Chore management, Grocery, to-do list, and calendar. This approach is helpful for a server system to achieve the required level of performance, scalability, security, availability, and connectivity. We have kept our cognitive load to a minimum in our UI design. Also, visual and functional consistency is followed across the application pages. To achieve optimal performance and scalability, we have divided our data into the following entities: User, Room, Expenses, Chores, Payment, ExpenseDetails, CalenderEvent, List, ListItem, and Comment. Multi-threaded MySQL Database is used to store User, Expenses, ExpenseDetails, and Payment entities. Entities such as Room, Chores, List, ListItem, Comment, and CalenderEvent are stored in a flexible MongoDB database.

5.1 Client Design

5.1.1 UI Mockups

Login and Signup page:

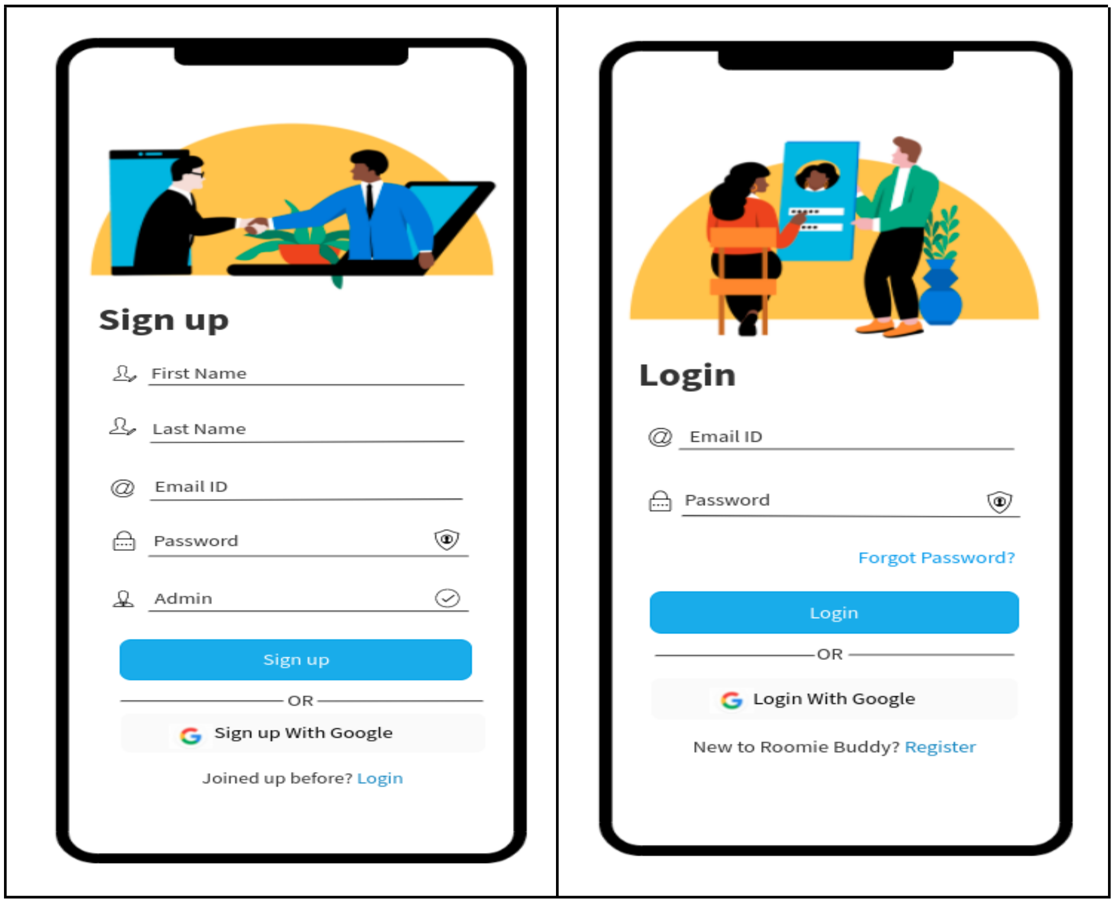
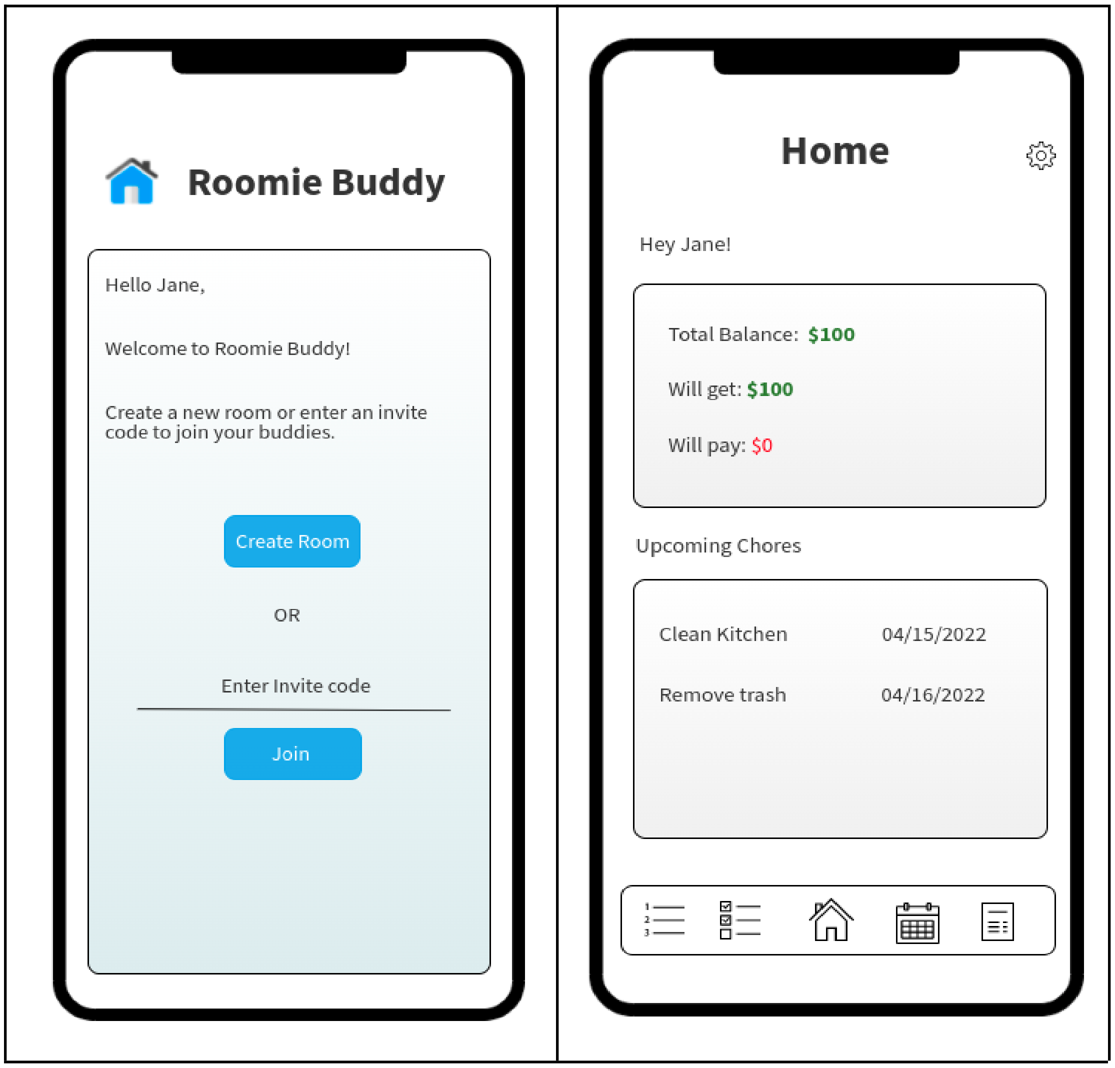


Figure 2. Login and SignUp Page

Create room and Home page:



Hello Jane,

Welcome to Roomie Buddy!

Create a new room or enter an invite code to join your buddies.

Hey Jane!

Total Balance: **$100**

Will get: **$100**

Will pay: $0

Clean Kitchen	04/15/2022
Remove trash	04/16/2022

Figure 3. Create Room and Home Page

Expense Management:

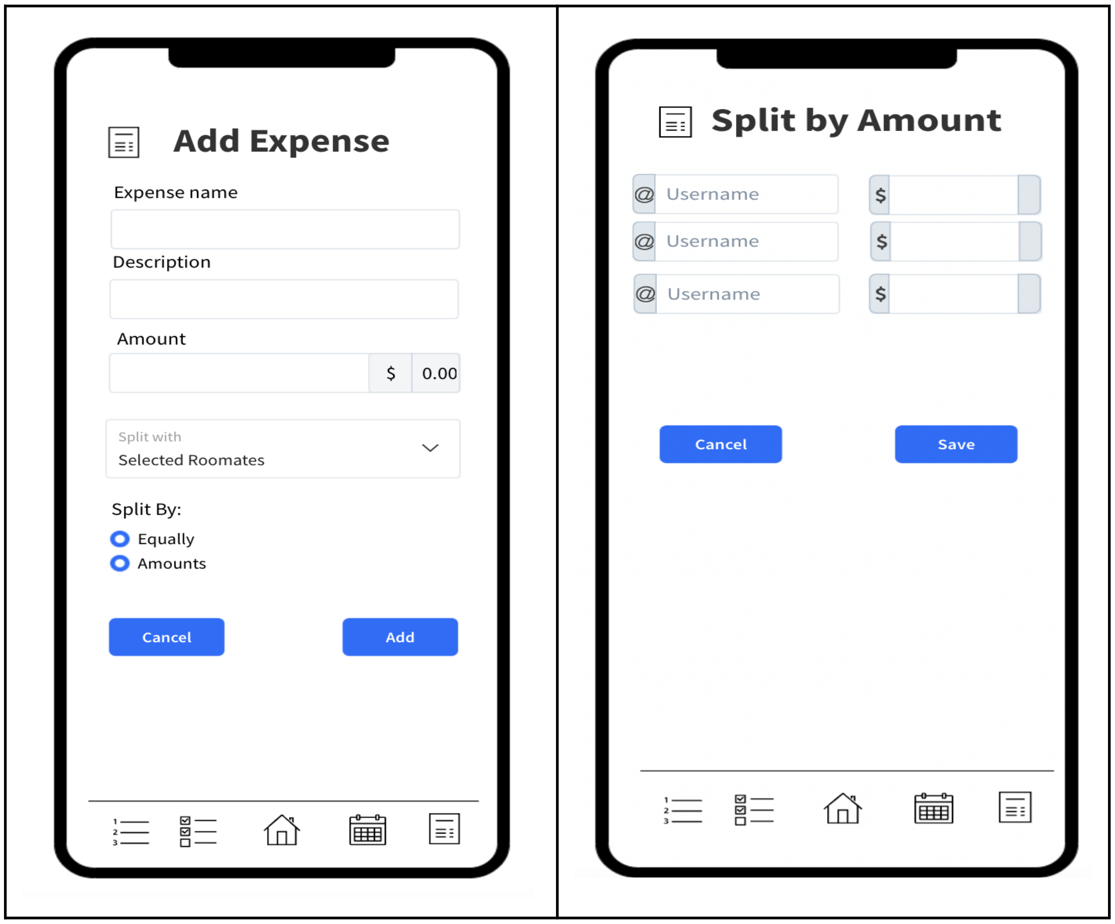


Figure 4. Add Expense and Split by Amount Page

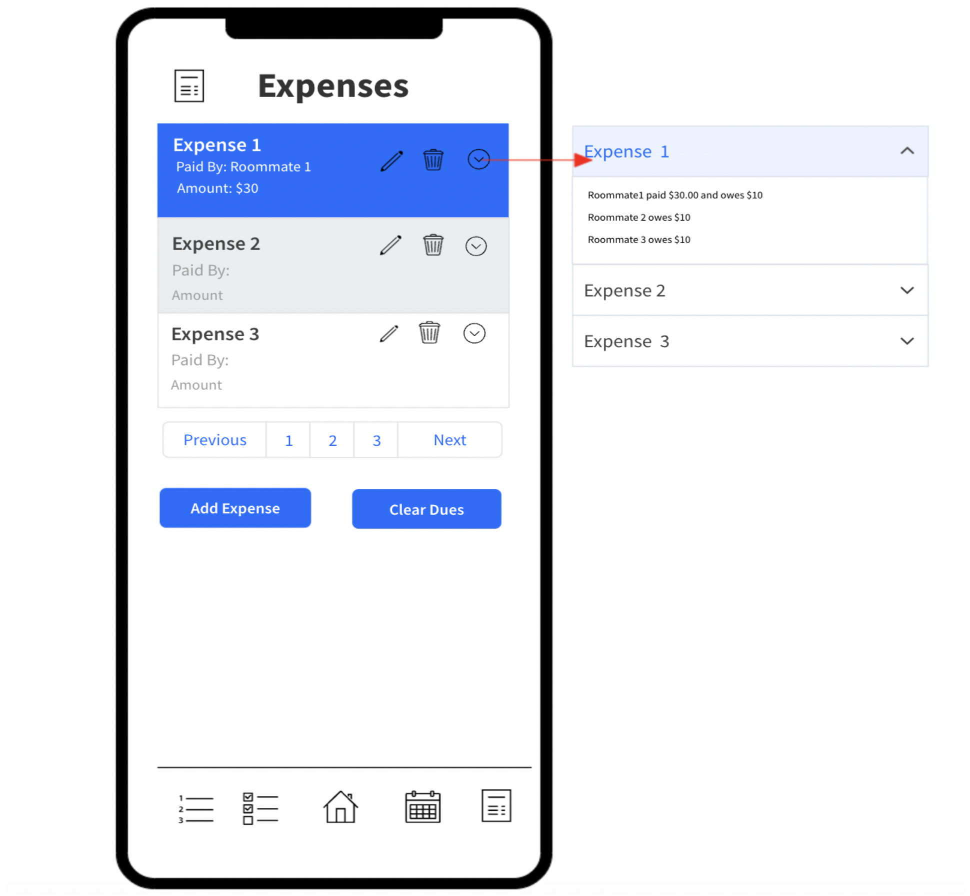
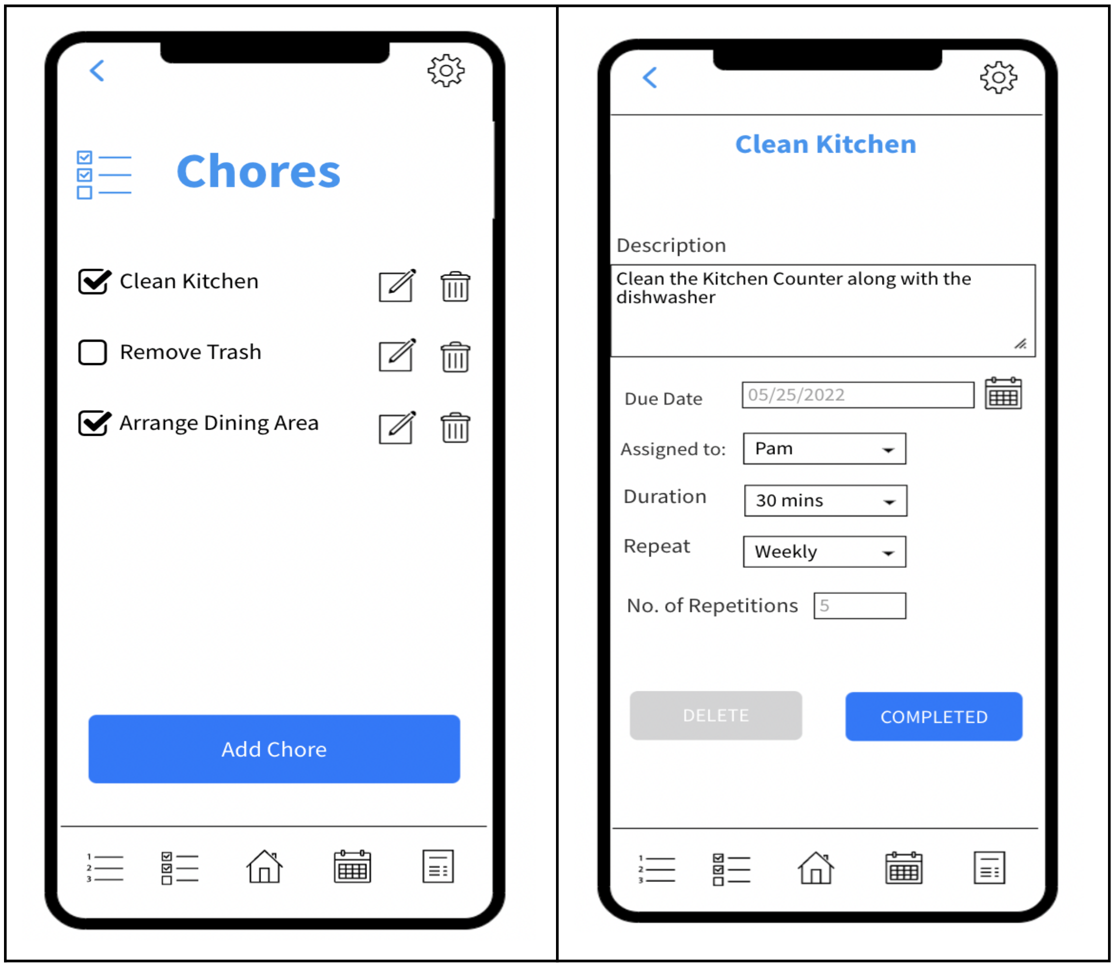


Figure 5. Expense Main Page

Chore Management:



- [✓] Clean Kitchen
- [ ] Remove Trash
- [✓] Arrange Dining Area

Description: Clean the Kitchen Counter along with the dishwasher

Due Date: 05/25/2022

Assigned to: Pam

Duration: 30 mins

Repeat: Weekly

No. of Repetitions: 5

Figure 6. Chores Home and Edit Chores page

To-Do List:

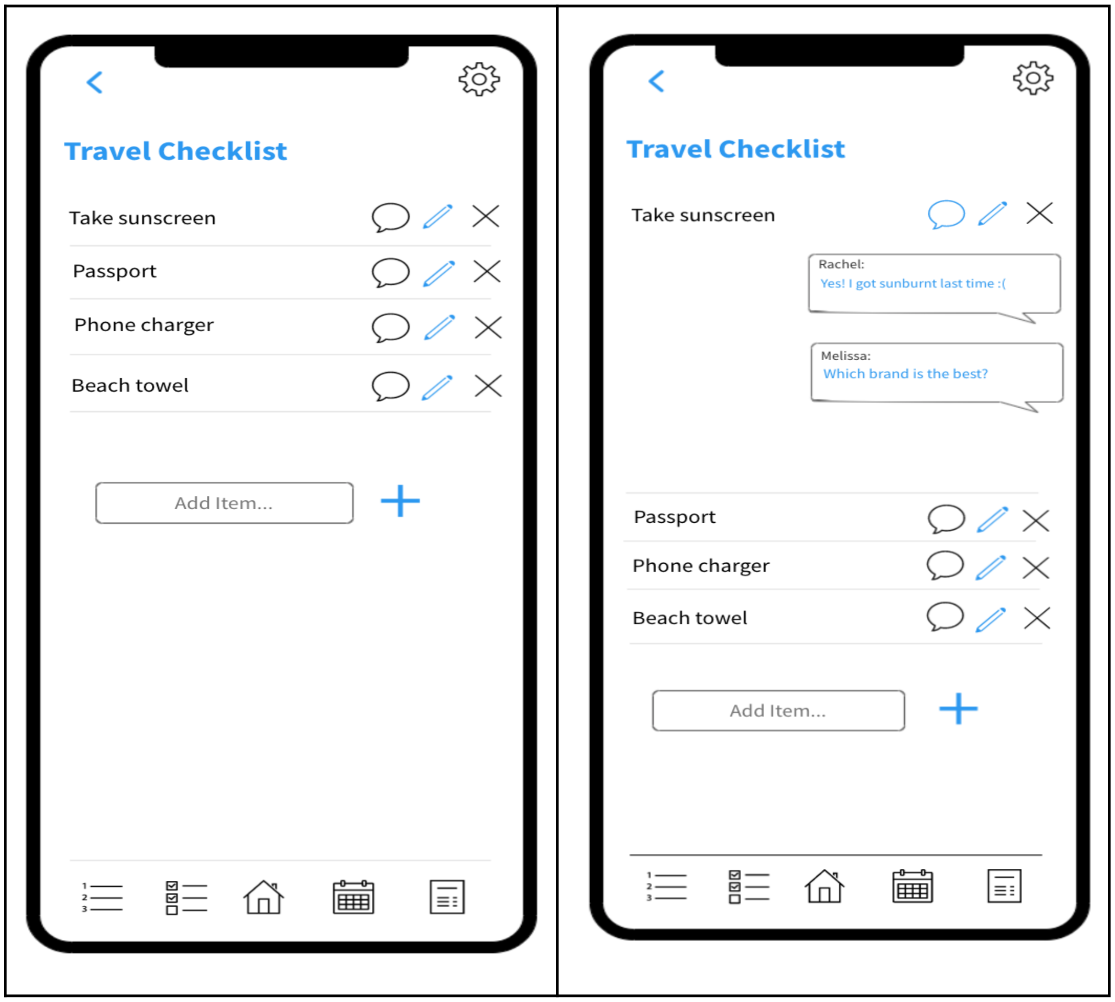
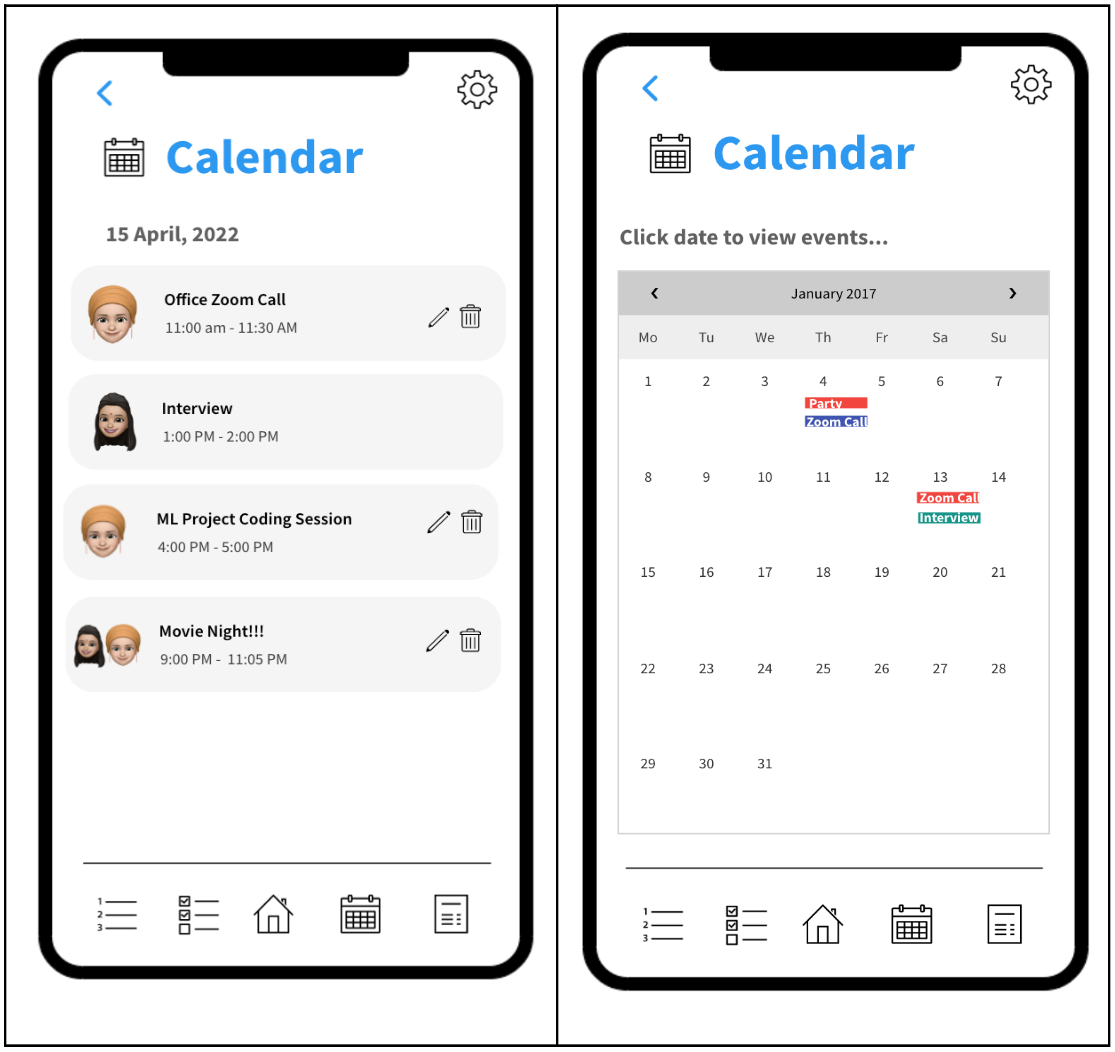


Figure 7. ToDo List Home and Edit List page

Calendar home page



15 April, 2022

Office Zoom Call
11:00 am - 11:30 AM

Interview
1:00 PM - 2:00 PM

ML Project Coding Session
4:00 PM - 5:00 PM

Movie Night!!!
9:00 PM - 11:05 PM

January 2017

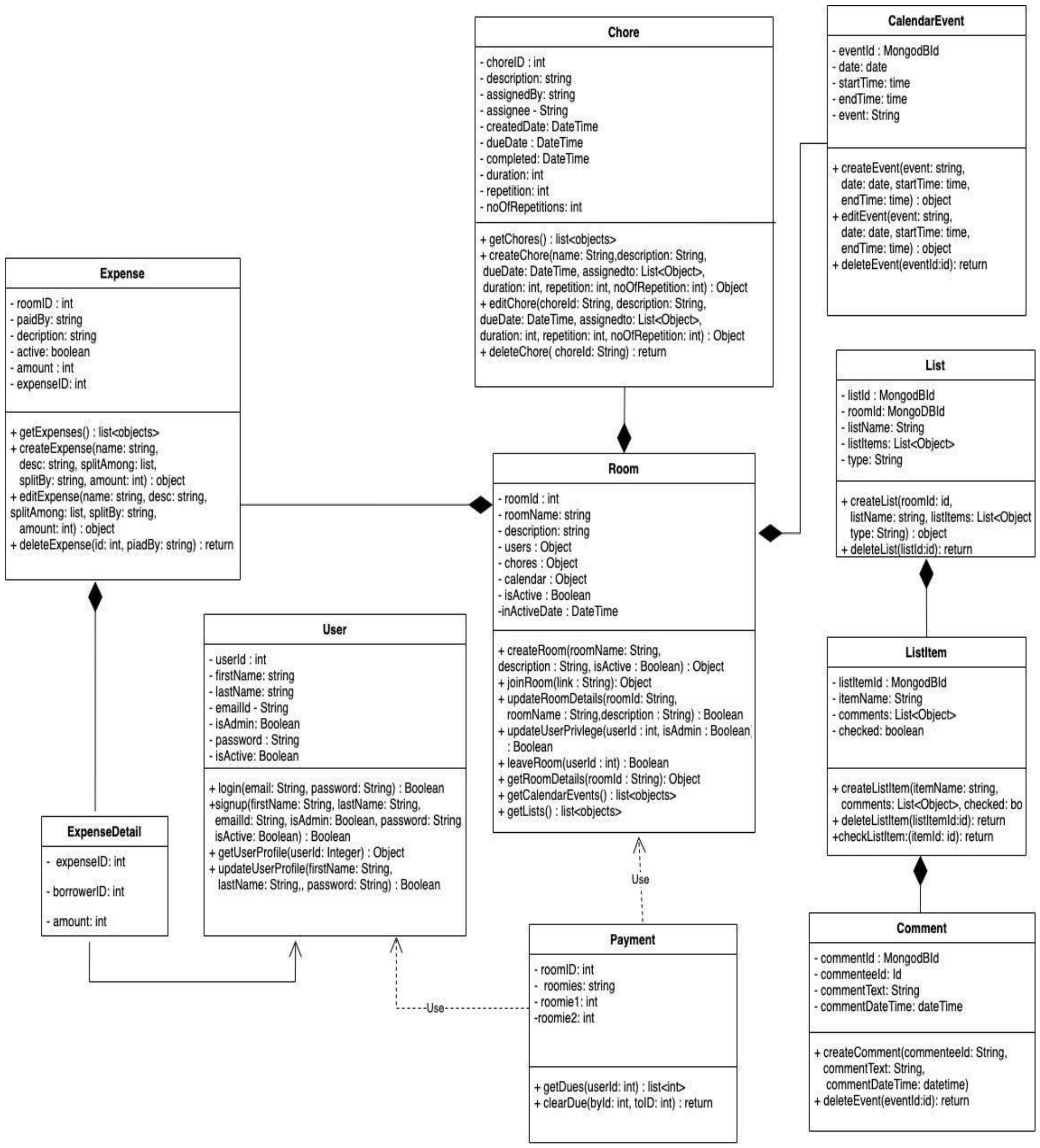
Mo	Tu	We	Th	Fr	Sa	Su
1	2	3	4 Party Zoom Call	5	6	7
8	9	10	11	12	13 Zoom Call Interview	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Figure 8. Calendar Home page

5.2 Middle-Tier Design

5.2.1 Class Diagram

The class diagram shows details about various objects that make up the system and the relationship between the objects and describes the object's functionality. We have identified essential classes for our application: User, Room, Expense, ExpenseDetail, Payment, Chore, List, ListItem, Comment, and CalendarEvent. The below-shown class diagram depicts the relationship between all these classes.



Chore
- choreID : int
- description: string
- assignedBy: string
- assignee - String
- createdDate: DateTime
- dueDate : DateTime
- completed: DateTime
- duration: int
- repetition: int
- noOfRepetitions: int
+ getChores() : list<objects>
+ createChore(name: String,description: String, dueDate: DateTime, assignedto: List<Object>, duration: int, repetition: int, noOfRepetition: int) : Object
+ editChore(choreId: String, description: String, dueDate: DateTime, assignedto: List<Object>, duration: int, repetition: int, noOfRepetition: int) : Object
+ deleteChore(choreId: String) : return
CalendarEvent
- eventId : MongodBId
- date: date
- startTime: time
- endTime: time
- event: String
+ createEvent(event: string, date: date, startTime: time, endTime: time) : object
+ editEvent(event: string, date: date, startTime: time, endTime: time) : object
+ deleteEvent(eventId:id): return
Expense
- roomID : int
- paidBy: string
- decription: string
- active: boolean
- amount : int
- expenseID: int
+ getExpenses() : list<objects>
+ createExpense(name: string, desc: string, splitAmong: list, splitBy: string, amount: int) : object
+ editExpense(name: string, desc: string, splitAmong: list, splitBy: string, amount: int) : object
+ deleteExpense(id: int, piadBy: string) : return
Room
- roomId : int
- roomName: string
- description: string
- users : Object
- chores : Object
- calendar : Object
- isActive : Boolean
-inActiveDate : DateTime
+ createRoom(roomName: String, description : String, isActive : Boolean) : Object
+ joinRoom(link : String): Object
+ updateRoomDetails(roomId: String, roomName : String,description : String) : Boolean
+ updateUserPrivlege(userId : int, isAdmin : Boolean) : Boolean
+ leaveRoom(userId : int) : Boolean
+ getRoomDetails(roomId : String): Object
+ getCalendarEvents() : list<objects>
+ getLists() : list<objects>
List
- listId : MongodBId
- roomId: MongoDBId
- listName: String
- listItems: List<Object>
- type: String
+ createList(roomId: id, listName: string, listItems: List<Object type: String) : object
+ deleteList(listId:id): return
User
- userId : int
- firstName: string
- lastName: string
- emailId - String
- isAdmin: Boolean
- password : String
- isActive: Boolean
+ login(email: String, password: String) : Boolean
+signup(firstName: String, lastName: String, emailId: String, isAdmin: Boolean, password: String isActive: Boolean) : Boolean
+ getUserProfile(userId: Integer) : Object
+ updateUserProfile(firstName: String, lastName: String,, password: String) : Boolean
ListItem
- listItemId : MongodBId
- itemName: String
- comments: List<Object>
- checked: boolean
+ createListItem(itemName: string, comments: List<Object>, checked: bo
+ deleteListItem(listItemId:id): return
+checkListItem:(itemId: id): return
ExpenseDetail
- expenseID: int
- borrowerID: int
- amount: int
Use
Payment
- roomID: int
- roomies: string
- roomie1: int
-roomie2: int
+ getDues(userId: int) : list<int>
+ clearDue(byId: int, toID: int) : return
Use
Comment
- commentId : MongodBId
- commenteeId: Id
- commentText: String
- commentDateTime: dateTime
+ createComment(commenteeId: String, commentText: String, commentDateTime: datetime)
+ deleteEvent(eventId:id): return

Figure 9. Class Diagram

5.2.2 *Sequence Diagram*

The sequence diagram shows the interaction between various objects in a given time sequence. It shows the messages passed between the participants and the objects in the order in which they occur.

Sequence Diagram for Login:

The Login sequence diagram shows the series of actions when a user logs in to the application. Once the user enters the details, the credentials are validated, and the user receives access only if the credentials are valid. Else, the user is notified with an error message.

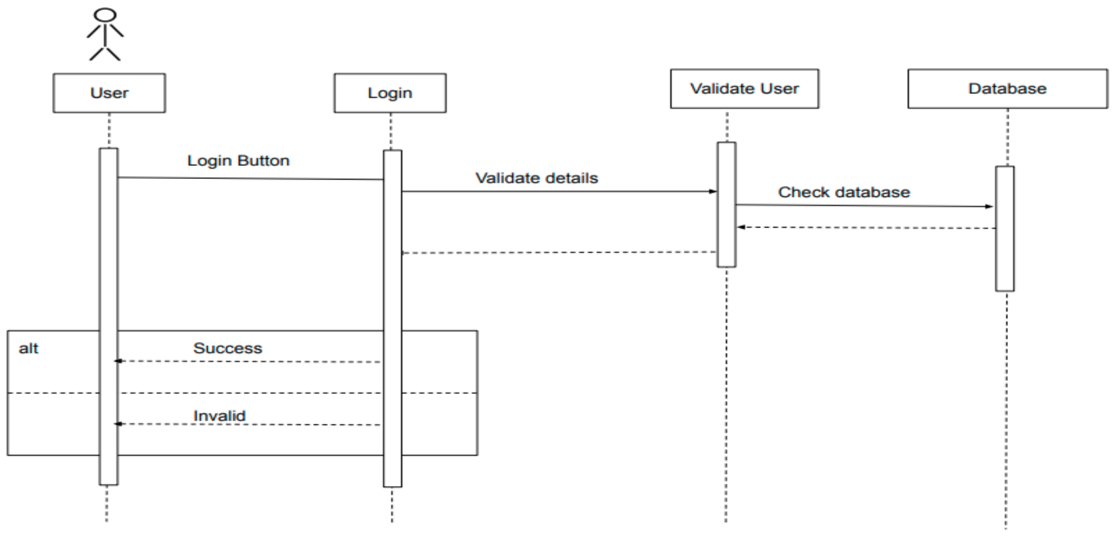


Figure 10. Sequence diagram for User Login

Sequence Diagram for Signup:

The signup sequence diagram shows the sequence of activities when the user clicks the signup button. If the details are valid, the user will successfully sign up for the application. Else, the user will be notified with an error message.

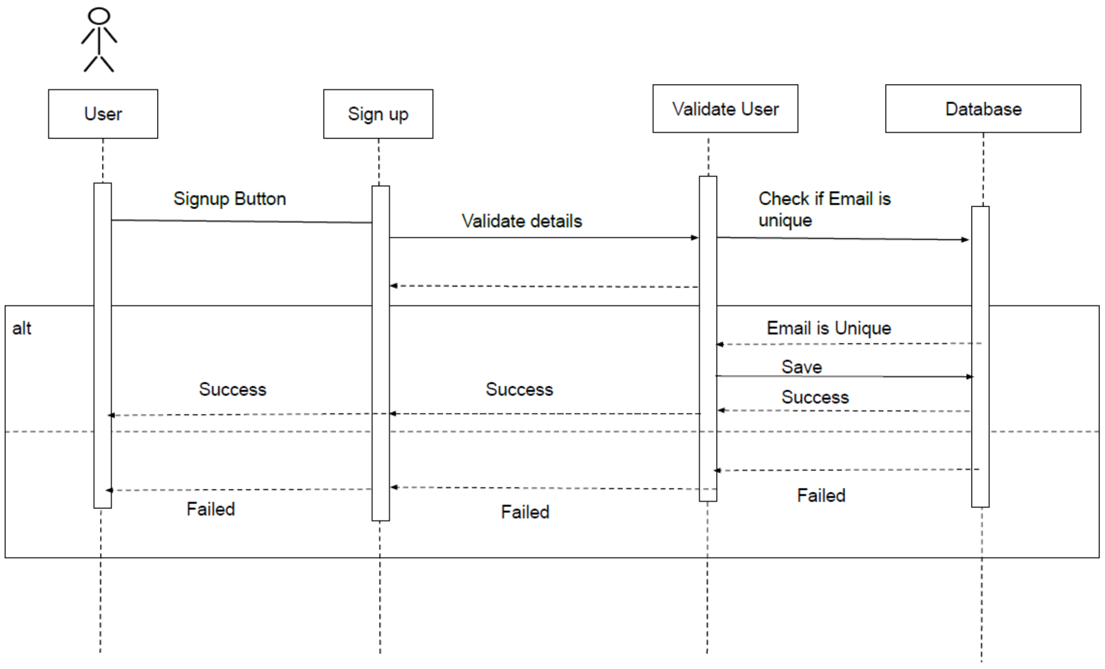


Figure 11. Sequence diagram for User Signup

Sequence Diagram for Chore Management:

The Chore management sequence diagram shows the flow of activities when the user performs chore-related activities. On navigating to the chore page, the user can see a list of chores assigned to all the roommates. The user can also add a new chore/ delete a chore/ update a chore.

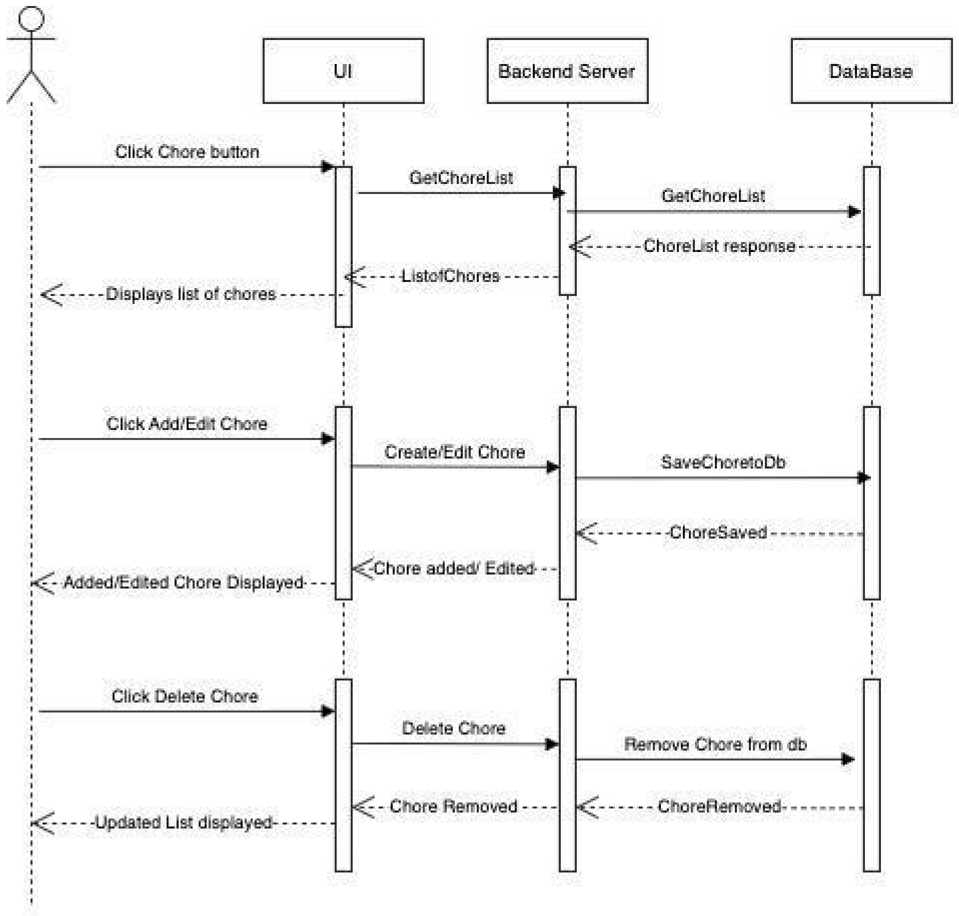


Figure 12. Sequence diagram for Chore management

Sequence Diagram for Expense Management:

The Expense management sequence diagram shows the flow of activities when the user performs expense-related activities. On navigating to the expense page, the user can see all the expenses associated with the group members. The user can also create a new expense/ delete an expense/ update an expense.

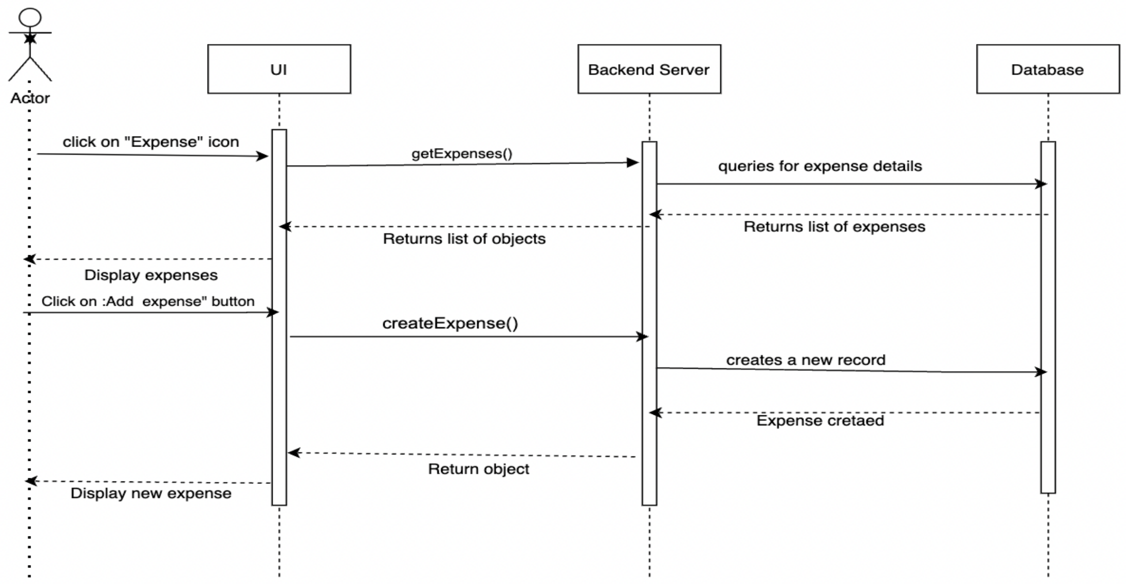


Figure 13. Sequence diagram for Expense management

Sequence Diagram for To-Do List:

The To-Do List sequence diagram shows the series of actions when the user navigates to the to-do list page. The user can view the existing list/ create a new list/ add comments to a list.

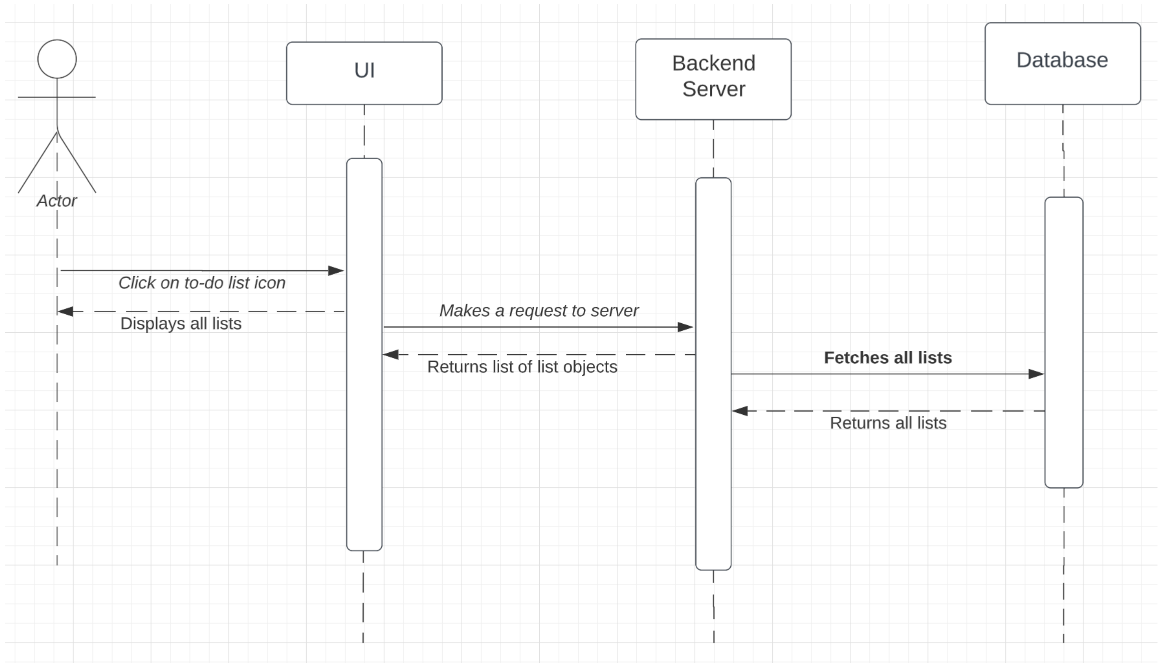


Figure 14. Sequence diagram for ToDo List

Sequence Diagram for viewing calendar with room events:

The calendar sequence diagram depicts the series of events happening when the calendar icon is clicked. The user can see calendar events/ add new events/ delete events.

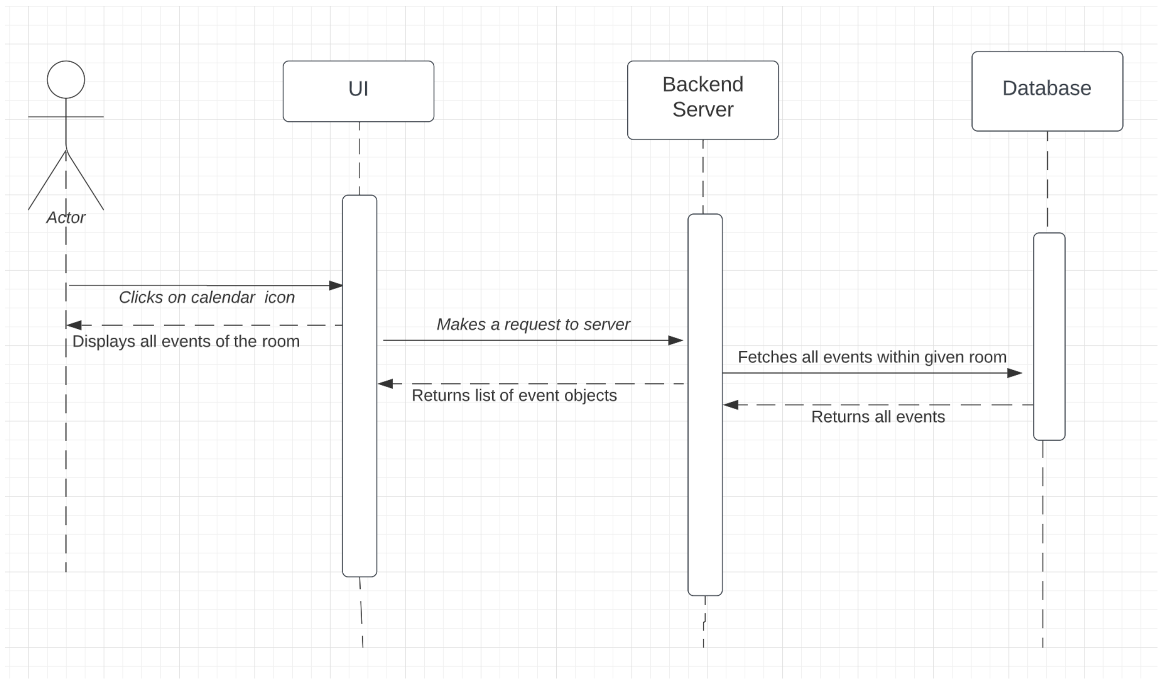

UI
Backend Server
Database
Actor
Clicks on calendar icon
Displays all events of the room
Makes a request to server
Fetches all events within given room
Returns list of event objects
Returns all events

Figure 15. Sequence diagram for viewing calendar with room events

5.3 Data-Tier Design

5.3.1 Database Entity Diagram

The Database entity diagram shows the logical structure of the database and the relationship between different entities in the database. The following database entity diagram shows the relationship between tables: User, Room, Expense, ExpenseDetails, Payment, Chore, List, ListItem, Comment, and CalendarEvent.

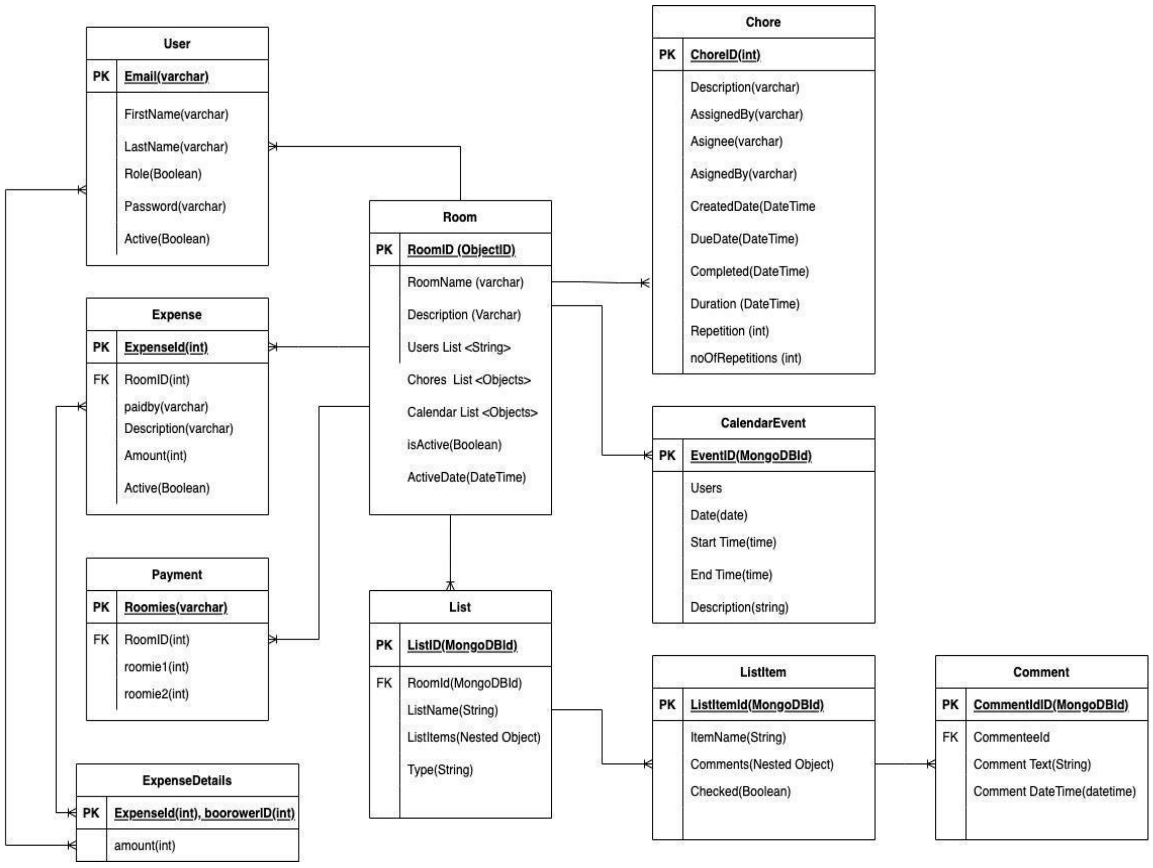


Figure 16. Database Entity Diagram

Chapter 6. Project Implementation

6.1 Client Implementation

6.1.1 Modules

6.1.1.0.Login/Sign Up:

This module handles registering new roomies to the application, and allowing current roomies access.

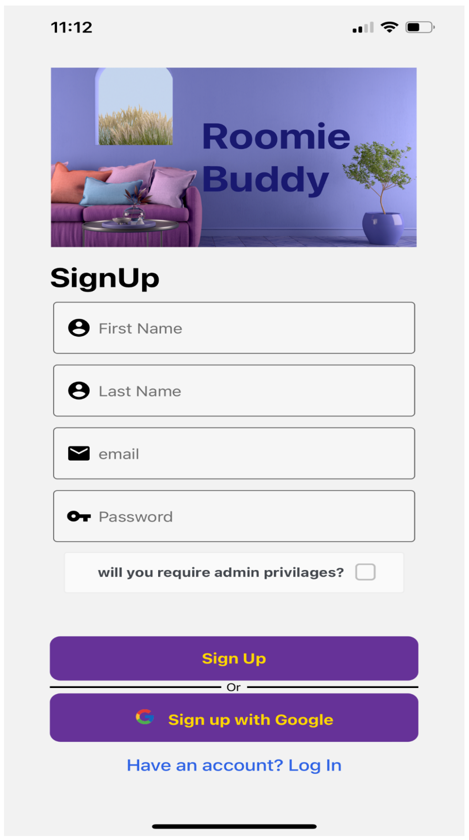
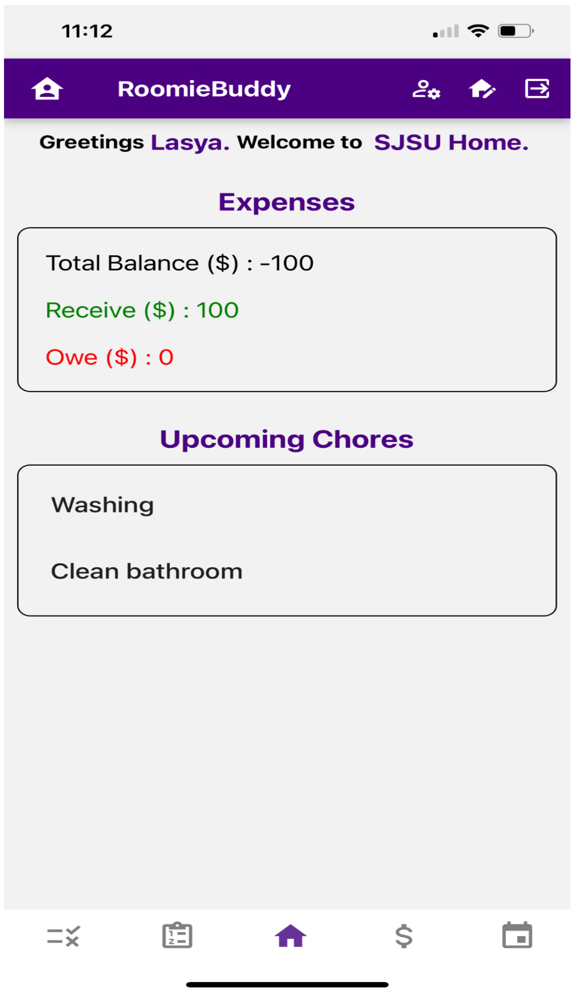


Figure 17. Login/ Signup page

6.1.1.1. Room:

This module controls the sub components that are available to all roomies within a room. Roomies belonging to different rooms cannot see the details of rooms they are not a part of.


11:12

RoomieBuddy

Greetings Lasya. Welcome to SJSU Home.

Expenses

Total Balance ($) : -100

Receive ($) : 100

Owe ($) : 0

Upcoming Chores

Washing

Clean bathroom

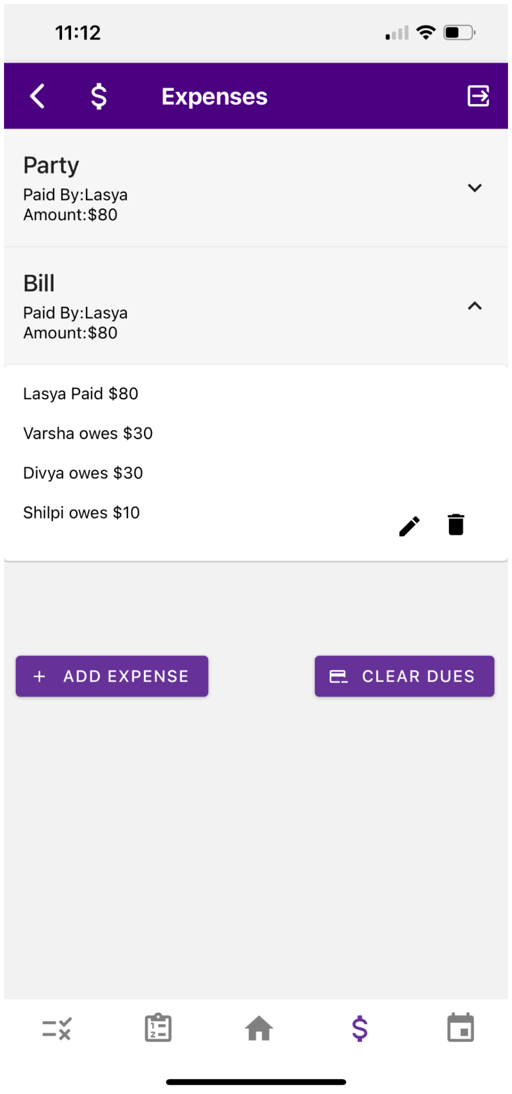
Figure 18. Room Main page

6.1.1.2. User:

This user holds all the user/roomie information.

6.1.1.3. Expense/Payment:

This module handles the expenses that roomies within a room make and log in the application. In addition it is used to track payments and dues between roommates within a room.



Party
Paid By:Lasya
Amount:$80

Bill
Paid By:Lasya
Amount:$80

Lasya Paid $80
Varsha owes $30
Divya owes $30
Shilpi owes $10

Figure 19. Expense Main page

6.1.1.4. Chores:

This module tracks the list of chores that are assigned to the roommates within the room. The chores can be reassigned.

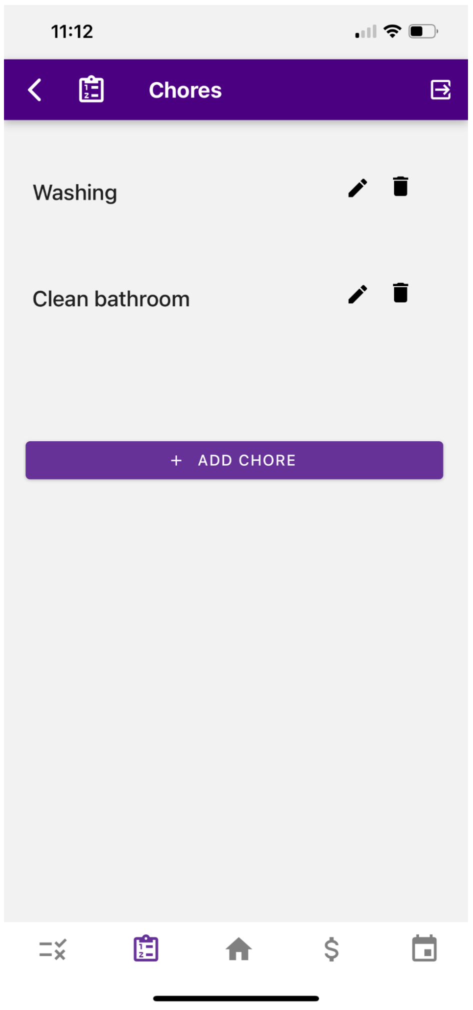
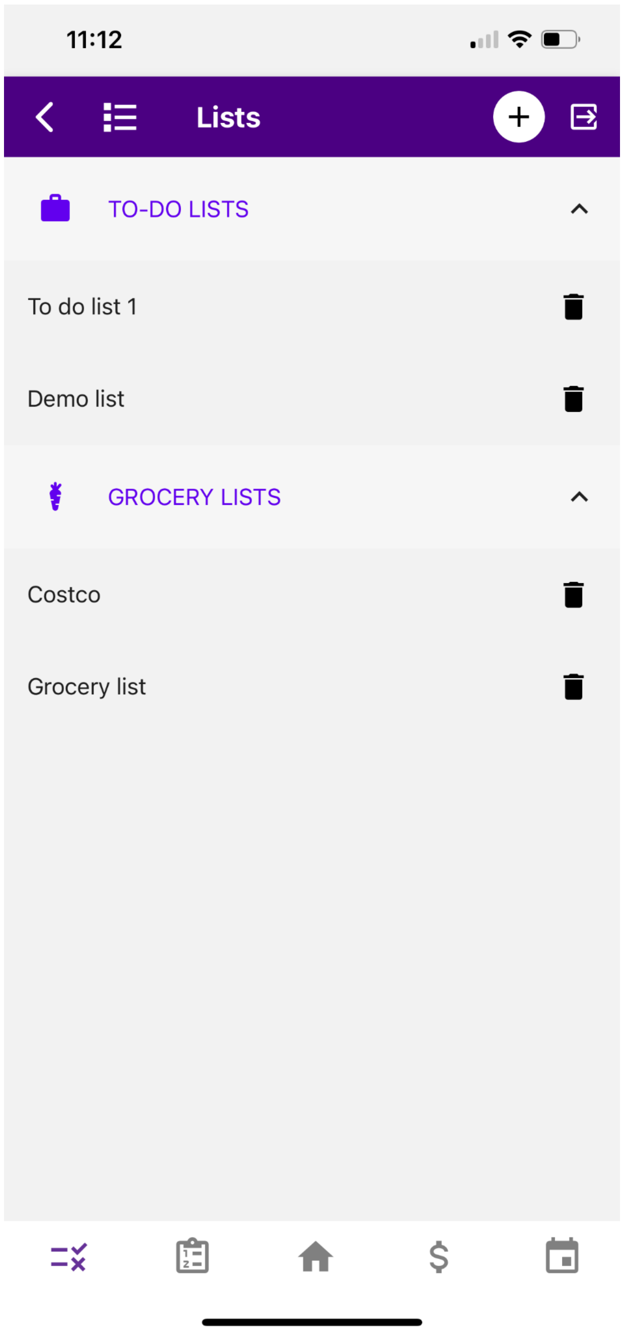


Figure 20. Chore Main page

6.1.1.5. To-do/Grocery Lists:

This module enables users of the application to create and share to-do and grocery lists with all their roommates. Roommates are able to add comments to have a conversation within the application itself.



11:12

Lists

TO-DO LISTS

To do list 1

Demo list

GROCERY LISTS

Costco

Grocery list

Figure 21. Lists Main page

6.1.1.6. Calendar:

This module helps roommates share their schedules with each other and also plan events together.

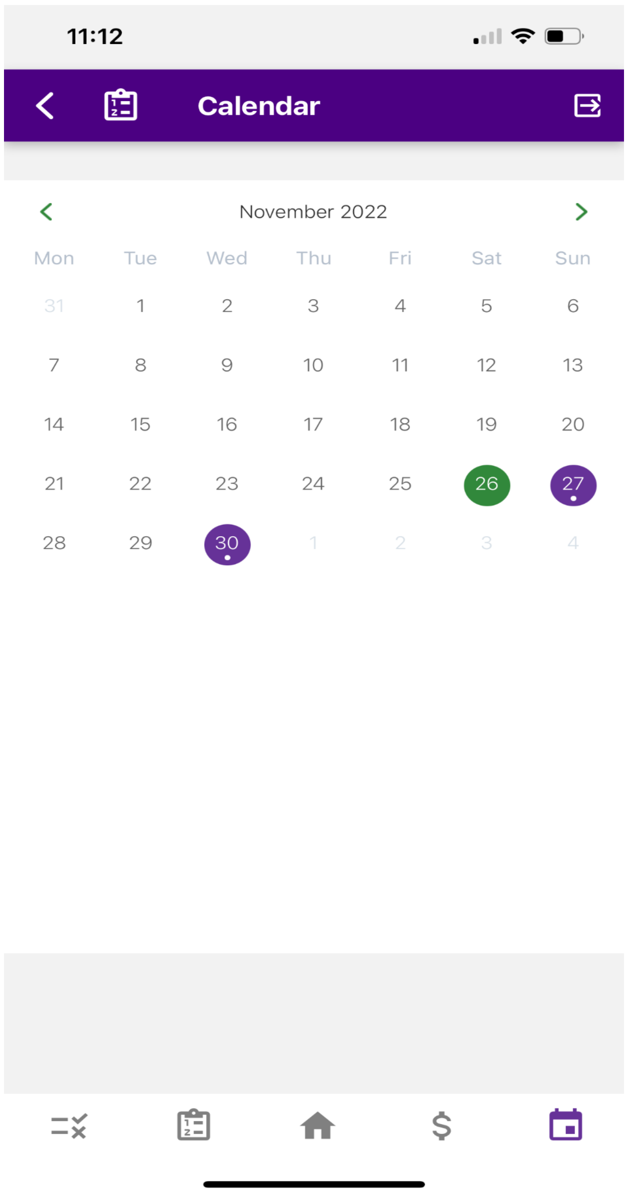


Figure 22. Calendar Main page

6.1.2 Local Storage

Local storage will be created to enable offline data persistence to make the application available even without connectivity to the internet over short periods of time.

6.2 Middle-Tier Implementation

6.2.1 *API Design*

We have created REST endpoints which will serve the data to the client facing side. The requests will contain bearer tokens for authorization of users registered with the application. The REST endpoints which make up the middle tier of our application are listed below and a detailed documentation including the response from the requests is present in the github repository.

Table 1. API Design Table

Module	**Action**	**Endpoint**	**HTTP Method**
Login	Login	/login	POST
Sign Up	Sign Up	/user/signup	POST
Room	CREATE	/room	POST
Room	GET	/room/getRoomDetails	GET
Room	UPDATE	/room/updateRoomDetails	PUT
Room	DELETE	/room/deleteRoom	PUT
Room	Leave Room	/room/leaveRoom	PUT
Room	GET roomie details	/room/memberDetails	GET

Room	Join room	/room/joinRoom	POST
User	GET profile	/profile	GET
User	UPDATE profile	/profile	PUT
Expense	Create expense	/expense/create	POST
Expense	Update	/expense/update	POST
Expense	Delete	/expense/delete	POST
Expense	Active Roomies	/expense/:user_id/active	GET
Expense	All Roomies	/expense/:user_id/all	GET
Payment	Roomies Dues	/payment/:user_id/mydues	GET
Payment	All Dues	/payment/getdues	GET
Payment	Settlement	/payment/settle	POST
Chores	GET chores	/chore/:user_id	GET
Chores	CREATE	/chore/create	POST
Chores	UPDATE	/chore/update	POST
Chores	DELETE	/chore/delete	DELETE
Chores	CREATE	/chore/create	POST
Calendar	CREATE	/room/calendar	POST
Calendar	DELETE	/room/calendar/:eventId	DELETE
Calendar	GET events	/room/calendar	GET
Calendar	UPDATE event	/room/calendar/:eventId	PUT

List	GET lists	/room/list	GET
List	CREATE	/room/list/addList	POST
List	CREATE list item	/room/list/listitem	POST
List	DELETE list item	/room/list/:listId/listItem/:listItemId	DELETE
List	DELETE list	/room/list/:listId	DELETE
List	CREATE comment	/room/list/comment	POST
List	DELETE comment	/room/list/:listId/listItem/:listItemId/comment/:commentId	DELETE
List	Check/Uncheck a listItem	/room/list/:listId/listItem/:listItemId/check	PUT
Email	Send an email	/email/sendInvite	POST

6.2.2 Authorization

We made use of passport JWT tokens to authenticate users. Upon signup, and login, a JWT token will be returned to the client side. This token must be passed as Authorization header in all future requests.

6.3 Data-Tier Implementation

6.3.1 User Table

The class diagram contains the details of the table structure followed across AWS RDS and MongoDB databases.

Chapter 7. Testing and Verification

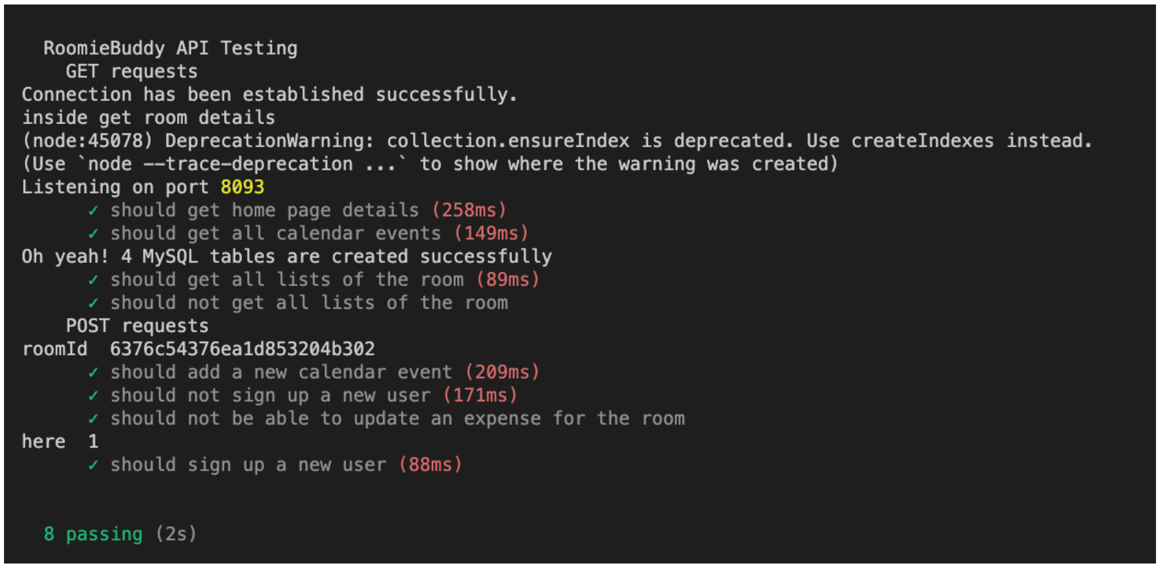
7.1 Testing

7.1.1 UX Testing

Ease of use, navigation, the intuitiveness of the application, look and feel of the layout, clarity of error messages are tested. We asked multiple users to test our application and give feedback on the ease of navigation. We adjusted the look and feel based on the suggestions received.

7.1.2 API testing

In this testing, all the APIs are evaluated. For the purpose of testing our Node.js APIs, we will be using the Mocha and Chai framework. A screenshot with execution of limited unit tests is shown below.



```
  RoomieBuddy API Testing
    GET requests
Connection has been established successfully.
inside get room details
(node:45078) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
Listening on port 8093
      ✓ should get home page details (258ms)
      ✓ should get all calendar events (149ms)
Oh yeah! 4 MySQL tables are created successfully
      ✓ should get all lists of the room (89ms)
      ✓ should not get all lists of the room
    POST requests
roomId  6376c54376ea1d853204b302
      ✓ should add a new calendar event (209ms)
      ✓ should not sign up a new user (171ms)
      ✓ should not be able to update an expense for the room
here  1
      ✓ should sign up a new user (88ms)


  8 passing (2s)
```

Figure 23. API testing

7.1.3 Functionality testing

We are ensuring if all the features are working as expected. We have created test cases for all the functionalities, provide sample input and finally compare the result with our expected outcome.

7.1.4 Integration testing

With all the modules of our application grouped, we performed integration testing to ensure if all the functionalities are working properly when combined together. We ensured the flow of data is maintained and does not disrupt the functionalities that the user performs from the mobile application.

7.2 Test Case Table

Table 2. Test Case Table

Functionality	Test Case	Test Values	Expected Result	Actual Result	Result (Pass/ Fail)
Sign up	Enter valid First name, Last name, Email ID, Password, and select the check box for Admin.	First Name: Jane Last Name: Doe Email Id: Jane@gmail.com Password: ValidP@sswor	The user must be able to sign up successfully.	User was able to sign up successfully.	Pass

		d			
Sign up	Enter an Email ID that already exists in the database.	Email Id: Jane@gmail.com	The application must throw the error: “An account for this Email Id already exists”	Working as per expected result.	Pass
Sign up	Enter valid First name, Last name, Email ID, Password, and do not check the box for Admin.	First Name: John Last Name: Doe Email Id: John@gmail.com Password: ValidP@ssword	The user must be able to sign up successfully without admin privileges. .	Working as per expected result.	Pass
Login	Enter a valid	Email Id:	The user must be	Working	Pass

	Email id and password	John@gmail.com Password: ValidP@ssword	able to login successfully	as per expected result.	
Login	Enter an invalid email id	Email Id: Jgmail.com	The application must throw the error: “Enter a valid Email id”	Working as per expected result.	Pass
Welcome page	Sign up into the application		The user must be redirected to the welcome page. The welcome page must display options to create a new room or join an existing room	Working as per expected result.	Pass
Welcome page	Enter a valid room name and	Room Name: Home Description:	The user must be able to enter room name and description	Working as per expecte	Pass

	description and click on the create button.	Mission Pointe Apartment	to create room page and A room will be successfully created and the user will be navigated to Home page	d result.	
Welcome page	If you have an invite link, paste it into the invite link text box and click on join button		The user must be able to join an existing group	Working as per expecte d result.	Pass
Home Page	Navigate to Home Page after Login/Sign up		The home page must display Total balance, payment dues, upcoming chores, and other icons.	Working as per expecte d result.	Pass

Settings Page	Update First Name, Last Name and Password with valid values and click on update button	First Name: Mike Last Name: Jackson Password: ValidP@ssword	The user must be able to update the details	Working as per expected result.	Pass
Settings Page	Update First Name and Last Name with special characters and click on update button	First Name: @@@ Last Name: !!!	The application must throw the error: “Only Alphanumeric values are allowed”	Working as per expected result.	Pass
Settings Page	Enter a password	Password: 12345	The application must throw the error:	Working as per	Pass

	that has less than 8 characters and click on the update button		“Enter a strong password with at least 8 characters”	expected result.	
Log out	Click on Log out button		The user must be successfully signed out of the application	Working as per expected result.	Pass
Your Room Page	Enter valid Room Name and Description and click on update button	Room Name: SJSU Description: 101	The details will be updated successfully	Working as per expected result.	Pass
Your Room Page	Click on Leave		The user will be able to leave the room	Working as per	Pass

	button for the user not having any dues or chores assigned on his/her name		successfully	expected result.	
Your Room Page	Click on remove button		The user will be successfully removed from the group if there are no dues or chores assigned on his/her name	Working as per expected result.	Pass
Push Notification	Create/update a chore, Add expense/record		Push notification must be sent to the user	Working as per expected result.	Pass

	payments, Add events to calendar and upcoming events				
Expense creation	Select split type as “Split by Amount” and check if able to enter a split amount against each roommate.		Should be able to enter a split amount against each roommate.	Working as per expected result.	Pass
Expense creation	Enter expense	expense name: “Party”	A new expense is created with an equal	Working as per	Pass

	name, description, amount, select roommates and split type as ”Equally” and click on “Add”.	Description: “Fun Expense” Select any two roommates. Amount: 60	$20 split among selected roommates.	expected result.	
Expense creation	1. Enter expense name, description, amount as 100, select roommates and split type as ”Amounts” .		A new expense is created with split according to entered amounts against each roommate selected.	Working as per expected result.	Pass

	2. Select any two roommates and enter 20,30 and 50 against each roommate displayed. 3. click “Save” 4. click”Add”				
Add Chore	Save chore without entering name		Chore cannot be added and user should be alerted	Working as per expected result.	Pass
Add Chore	Create chore with name,	ChoreName: Clean Kitchen Description:	Chore created successfully	Working as per expecte	Pass

	description , due date, assignee, duration, repetition, no of repetitions	Clean Kitchen and dishwasher DueDate: 05/25/2022 Assigned to: Pam Duration: 30 mins Repeat: Weekly till date: 10/25/2023		d result.	
Add Chore	Create chore without due date	ChoreName: Clean Room Description: Clean Room Assigned to: Pam	Chore will not be created as due date is a required field	Working as per expecte d result.	Pass

		Duration: 20 mins Repeat: Weekly no . of repetitions: 7			
Add Chore	Create chore and press cancel button		Chore should not be saved	Working as per expected result.	Pass
AddChore	Create a chore while assigning the chore to everyone in the household	ChoreName: Clean Kitchen Description: Clean Kitchen and dishwasher DueDate:	Chore should be created and should be added in the chore list of all the members	Working as per expected result.	Pass

		05/25/2022 Assigned to: Pam, Phyllis, Angela, Holly Duration: 30 mins Repeat: Weekly no . of repetitions: 5			
ShowChores	Press the chore button on the homepage		Display a list of all the chores for the given signed in user	Working as per expecte d result.	Pass
DeleteChore	Delete any chore from main page of Chores		Chore should be deleted and removed from the list	Working as per expecte d result.	Pass

EditChore	Click on the chore to go to the chore detail page and edit the details such as description, due date, assignees, duration, repetition, no. of repetitions. Click on the save button	ChoreName: Clean Kitchen Description: Clean Kitchen, buy dishwasher liquid. DueDate: 05/25/2022 Assigned to: Pam, Phyllis, Angela, Holly Duration: 30 mins Repeat: Weekly no . of repetitions: 5	Chore should be updated successfully	Working as per expecte d result.	Pass
EditChore	Edit the		Editing of chore	Working	Pass

	chore name		name is not possible.	as per expected result.	
To-Do List	Verify the presence of all existing lists	RoomId to fetch lists only belonging to the specified room	All lists created by roommates of selected room must display	Working as per expected result.	Pass
To-Do List	Click on the “Add a new list” button	NA	A new list must get created and must be viewable by all roommates	Working as per expected result.	Pass
To-Do List	Click on the “+” within a list and add a list item	itemName: “phone charger”	A new list item must get created in the selected list and with the name “phone charger” and it must be viewable by all roommates	Working as per expected result.	Pass

To-Do List	Modify an existing list item by clicking on the pencil button in the right corner	itemName: “laptop charger”	The list item must get modified and should reflect in the UI with new value “laptop charger”	Working as per expected result.	Pass
To-Do List	Click on the “+” within a list and add a list item with no itemName	itemName: “”	The new list item should not be created and an error message must be shown to the user indicating that the name itemName is a mandatory value.	Working as per expected result.	Pass
Calendar	Click on the calendar icon in the navigation	NA	The shared calendar for the room should be visible and must be populated with all	Working as per expected result.	Pass

	page		personal and common events of the roommates		
Calendar	Delete the event added.	id: eventid1	You should be able to delete the event with eventid1.	Working as per expected result.	Pass
Calendar	Modify the time of a personal event by clicking on the pencil icon	id: event id 2, startTime: 9AM	The event with eventid2 should be modified and now have a startTime of 9AM instead of the older startTime. The updated event should be visible on the appropriate date and time within the calendar.	Working as per expected result.	Pass
Calendar	Edit an	id: event id 4,	The event should get	Working	Pass

	event to include another room member who was not previously added.	newUserId: 5	updated to include the user with id 5.	as per expected result.	
Calendar	Try to edit a calendar event to have a date in the past	NA	The past dates should be grayed out in the dropdown for calendar event update and the edit should not be possible.	Working as per expected result.	Pass

Chapter 8. Performance and Benchmarks

8.1 Performance Evaluation

The user experience is evaluated based on the performance of the application. To ensure speed and reliability of the application, we will be performing load and stress testing. For load tests, we will be using Webload to evaluate dynamic resources and determine the maximum number of concurrent uses that our application can handle. We will be conducting tests from 100-1000 concurrent users. Similarly, we will be performing stress tests to determine maximum load capacity. When the load goes beyond the limit, the web server will start to respond slowly and will produce errors. Based on the throughput value, we can determine the heavy load threshold range.

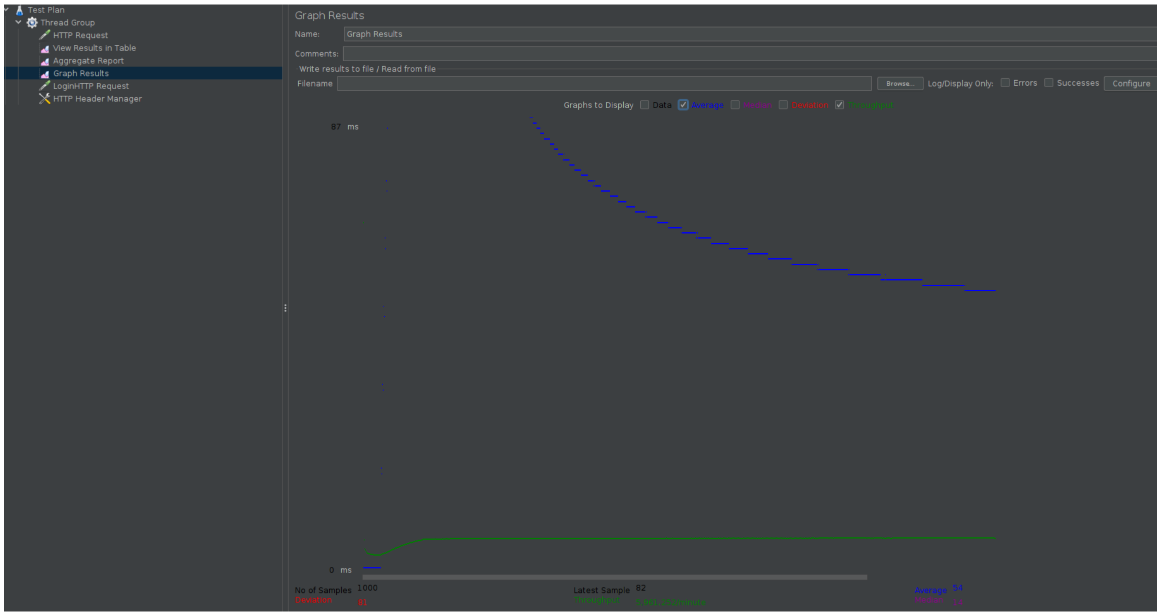
8.1.1 API response standards

The API response time is a critical statistic to assess since it measures how long it takes the API to reply to requests.If a user wishes to interact with the API, they will submit a request to the API endpoint, and the server will respond with the appropriate status code based on the activity and content of the request.

We used Postman to measure both API response time and API status code. We targeted a response time of 2-3 seconds and have verified that our APIs return well within this time limit, mostly in the range of 0.8-1.2s.

8.1.2 System performance under heavy load

The load test evaluates the performance of the application in the presence of a high number of concurrent requests over a certain period of time. For our application, we have performed load testing using Jmeter to determine how many users our application can handle. The below figures show Jmeter results for 500 and 1000 users respectively:



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Log/Display Only: ☐ Errors ☐ Successes

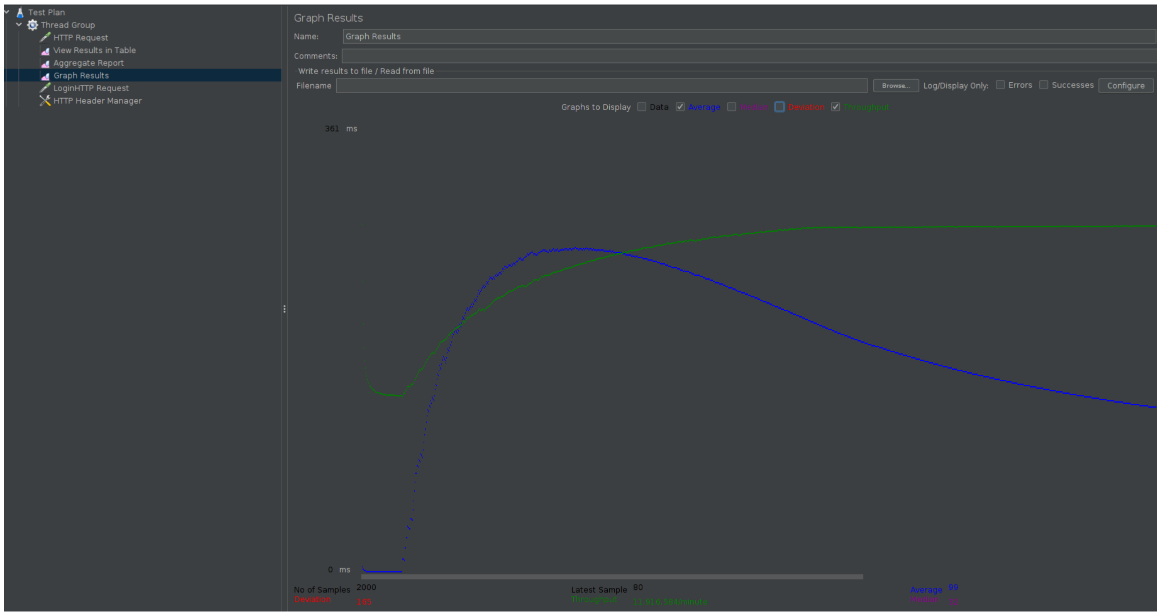
Graphs to Display ☐ Data ☑ Average ☐ Median ☐ Deviation ☑ Throughput

No of Samples 1000
Deviation 81

Latest Sample 82
Throughput 5,961.252/minute

Average 54
Median 14

Figure 24. Jmeter result for 500 users (Throughput: 5,961.252/minute)



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Log/Display Only: ☐ Errors ☐ Successes

Graphs to Display ☐ Data ☑ Average ☐ Median ☐ Deviation ☑ Throughput

No of Samples 2000
Deviation 165

Latest Sample 80
Throughput 11,916.584/minute

Average 99
Median 32

Figure 25. Jmeter result for 1000 users (Throughput: 11,909.488/minute)

From the results it can be seen that as the number of users increase, the throughput has increased exponentially. This is a clear indication that the application is configured to handle high load.

Chapter 9. Deployment, Operations, Maintenance

9.1 Deployment

9.1.1 Deployment Steps

- The data persistence layer will be implemented first.
  - MySQL AWS RDS will be created to store entities of the application which must preserve ACID properties. This RDS will be created from the AWS console and the link to it will be seeded within the application backend code. In the Expense functionality, any updates to expenses will be implemented with transaction control.
  - MongoDB database will be created manually to house the entities which follow BASE properties.
  - The databases will be configured with thread pooling to improve performance.
- For the deployment of the backend servers, multiple AWS EC2 instances will be created using the AWS console. A shell script containing commands to be run within the instances will be moved using the scp command.
- Load Balancers, created on the AWS console, will be enabled for the multiple backend servers to share the load.
- For the front end react native application, apk/aab files will be generated and then will be deployed to the Store.

9.1.2 Deployment Diagram

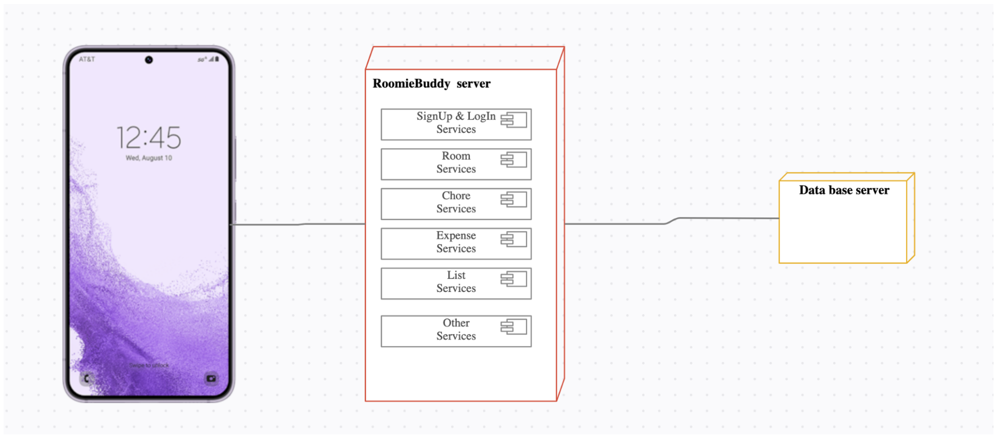


Figure 26. Deployment Diagram

Chapter 10. Summary, Conclusions, and Recommendations

10.1 Summary

Our project's main objective was to create a user-friendly smartphone application to manage and handle typical cohabitation concerns. Our program includes an expense tracker that performs equal or unequal splits based on user requirements. It can assign duties to roommates and provide notifications when a task has been completed. To keep up with home necessities, we have created to-do and grocery lists as well.

By transferring all infrastructure requirements to cloud environments, no hardware was used at any level. REST api endpoints and object-oriented programming concepts help to ensure effective communication between the client and the server.

Building a quick-loading application with rich UI interfaces allowed us to keep the program's appearance and feel in mind. To guarantee a seamless transition, extensive testing is done on both the client and server sides.

10.2 Conclusion

This mobile application is useful when dividing domestic chores fairly among family members. When living in a shared household, features like an expense tracker, chore management, lists, and calendar event tracking come in handy, and our app provides them all.

For the students to manage their household demands, an easy-to-use and free mobile application is required. Our application offers a free integrated platform for managing all household tasks. This application is robust and can be used to organize and track housework by all types of students and working professionals.

10.3 Recommendations for Further Implementation

Based on our current research and implementation, as a next step we would like to incorporate Machine Learning within our application to make suggestions to roomies based on their current habits. We are confident that suggestions for grocery items based on roommates' current purchases will prove to be a feature well liked by the target audience. Similarly, we can add suggestions for chores the roommates log regularly as required within their dwelling unit.

Glossary

- API: Application Programming Interface
- JSON: JavaScript Object Notation
- REST: Representational State Transfer.
- AWS: Amazon Web Services
- ACID: Refers to the four key properties of a transaction: atomicity, consistency, isolation, and durability
- UI: User Interface
- UX: User Experience
- iOS: an operating system used for mobile devices.
- ExpressJS: NodeJS web application framework
- React Native: Open source UI software framework
- NodeJS: Open source server environment
- Load Balancer: Networking solution to distribute load across servers
- JMeter: Tool for load testing application server
- Mocha: Javascript Test Framework
- Chai: Assertion library used alongside Mocha

References

Natalier, K. (2003). `I'm not his Wife'. In Journal of Sociology (Vol. 39, Issue 3, pp. 253–269). SAGE Publications. https://doi.org/10.1177/00048690030393003

Curşeu P. L., Boroş S., & Oerlemans L. A. G. (2012). Task and relationship conflict in short-term and long-term groups. In R. Posthuma (Ed.), International Journal of Conflict Management (Vol. 23, Issue 1, pp. 97–107). Emerald. https://doi.org/10.1108/10444061211199331

Clark V., Tuffin K., & Bowker N. (2020). Managing Conflict in Shared Housing for Young Adults. New Zealand Journal of Psychology, 49(1), 4+. https://link.gale.com/apps/doc/A645154873/PPCJ?u=csusj&sid=bookmark-PPCJ&xid=3df84b7b

Nicolò A., & Velez R. A. (2017). Divide and compromise. In Mathematical Social Sciences (Vol. 90, pp. 100–110). Elsevier BV. https://doi.org/10.1016/j.mathsocsci.2017.04.004

Clark V., Tuffin K., Frewin K., & Bowker N. (2018). Housemate desirability and understanding the social dynamics of shared living. In Qualitative Psychology (Vol. 5, Issue 1, pp. 26–40). American Psychological Association (APA). https://doi.org/10.1037/qup0000091

Mause, K. (2008). The Tragedy of the Commune: Learning from worst-case scenarios. In The Journal of Socio-Economics (Vol. 37, Issue 1, pp. 308–327). Elsevier BV. https://doi.org/10.1016/j.socec.2007.01.015

Lewis, M., & Perry, M. (2019). Follow the Money. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. CHI '19: CHI Conference on Human Factors in Computing Systems. ACM. https://doi.org/10.1145/3290605.3300620

Farhadi, A., & Hajiaghayi, M. (2017). On the Complexity of Chore Division (Version 2). arXiv. https://doi.org/10.48550/ARXIV.1710.00271

Aziz, H., Caragiannis, I., Igarashi, A., & Walsh, T. (2021). Fair allocation of indivisible goods and chores. In Autonomous Agents and Multi-Agent Systems (Vol. 36, Issue 1). Springer Science and Business Media LLC. https://doi.org/10.1007/s10458-021-09532-8

Bei, X., Huzhang, G., & Suksompong, W. (2020). Truthful fair division without free disposal. In Social Choice and Welfare (Vol. 55, Issue 3, pp. 523–545). Springer Science and Business Media LLC. https://doi.org/10.1007/s00355-020-01256-0

Plaisant, C., Clamage, A., Hutchinson, H. B., Bederson, B. B., & Druin, A. (2006). Shared family calendars. In ACM Transactions on Computer-Human Interaction (Vol. 13, Issue 3, pp. 313–346). Association for Computing Machinery (ACM). https://doi.org/10.1145/1183456.1183458