DALHOUSIE
UNIVERSITY

Technical Design Document

CSCI5308
Advanced Topics in Software Development

Table of Contents

Table of Figures

Group Details

Group 7

Members

- Kenee Ashok Patel – B00969805
- Jay Sanjaybhai Patel – B00982253
- Jay Alpeshkumar Patel – B00969013

Introduction

Purpose of the Document

The purpose of this technical design document is to outline the components, technical decisions, and architecture of the full-stack web application [1] RoomM8. It details the high-level and low-level architectural decisions, component descriptions, and interactions within the system, ensuring that anyone wanting to understand the codebase has a clear understanding of how the system operates. Additionally, this document serves as a reference for all future developments, providing developers with information on data and interface design, technology stack, and testing strategies. This will help developers streamline their development processes according to the document's standards. Furthermore, the document includes detailed information on testing strategies, such as unit testing, integration testing, system testing, and user acceptance testing.

Scope of the Document

This document's scope covers a number of crucial aspects of the application’s overview and design choices.
System Overview: This section provides an overview of the software system, describing its main features and the operating environment. It provides a comprehensive grasp of the system's function and the factors influencing its growth by outlining the project's broad goals and restrictions.
Architecture and Component Design: This section gives the high-level architecture [2], including diagrams showing the overall system organization. Each component or module of the system is described with its functions, interactions, and data flows including a sequence or flow diagram.
Data Design: The section covers the organization and management of data inside the system. It shows the connections between various data items, and also displays entity-relationship (ER) [3] diagrams and data models.
Interface Design: This section covers internal interfaces and their interactions, external interfaces and their interactions, and API documentation covering details about the Application Programming Interfaces (APIs).
Security Design: This section covers the security measures taken by the system, covering points like user authentication and Access Control, data encryption, and compliance measures.
Performance Considerations: This section covers the system's load, fault tolerance, and scalability management strategies.
Testing Strategy: This section covers the approach and strategies implemented for testing the system, including unit testing, integration testing, system testing, and user acceptance testing.
Conclusion and Future Work: This section covers key points by summarising them and discusses potential future enhancements to the application.

Target Audience

The target audience includes:
Developers: The developers [6] can take these documents as a reference for developing new features or understanding older features and design choices.

End Users: Although not the primary audience, end users [7] can occasionally refer to this document for a high-level understanding of how the system operates, especially if they are involved in user acceptance testing.

System Overview

System Description and Context

roomM8 is a web-based application designed to enhance the living experience for roommates by streamlining communication, task management, and grocery planning. The system operates in a client-server environment where the backend is developed using Spring Boot [8] and the frontend is built with Next.js [9]. The primary users of the system are roommates who need to coordinate daily activities, manage shared tasks, and communicate effectively.

The application offers several key features which includes:

1. **Authentication:** Secure user registration, login, and password reset functionalities.
2. **Room Setup:** Association of users with rooms using unique room codes.
3. **Task Management:** Creation, modification, and deletion of tasks assigned to room members.
4. **Announcements:** Posting and managing announcements for room members.
5. **Grocery Lists:** Managing shared grocery lists and items.
6. **Expenses:** Managing the daily expenses that occur for the household.
7. **User Profiles:** Viewing and updating user profile details.

The application ensures that each user can interact with their assigned room and its associated tasks, announcements, and grocery lists in a secure and user-friendly manner.

Design Goals and Constraints

The design of roomM8 is driven by the following goals:
Usability: Ensure the application is intuitive and easy to use for all users. This involves designing a clean and straightforward user interface with a consistent look and feel, using Shadcn/UI [10] components extensively.
Scalability: Design the system to handle an increasing number of users and data without performance degradation. This means using efficient algorithms and data structures, as well as scalable technologies like Spring Boot and Next.js.
Security: Implement robust security measures to protect user data and ensure secure authentication and authorization. This includes using JWT [11] tokens for session management and encrypting sensitive data.
Maintainability: Follow clean code principles and SOLID design principles to facilitate easy maintenance and future enhancements. This involves writing modular, well-documented code and using best practices for version control and code reviews.
Performance: Optimize the system to provide quick response times and a smooth user experience. This involves minimizing the use of heavy resources, optimizing database queries, and using caching where appropriate.
Testability: Design the system to be easily testable, supporting Test Driven Development (TDD). This includes writing unit tests, integration tests, and end-to-end tests to ensure the system works as expected.

Accessibility: Ensure the application is accessible to users with disabilities by following accessibility guidelines and best practices in UI design.

Constraints include:
Technological: The application must use Spring Boot for the backend, Next.js for the frontend, and MySQL for the database. These technologies are chosen for their robustness, scalability, and community support.
Time: Development must adhere to a fixed timeline, ensuring timely delivery of features. This requires careful planning and prioritization of tasks.
Resource: Development resources, including developers and tools, are limited and must be used efficiently. This involves optimizing workflows and making the best use of available resources.
Compliance: The application must comply with relevant data protection regulations, such as GDPR, ensuring that user data is handled and stored securely.

Stakeholder Identification

The primary stakeholders for roomM8 include:
Roommates (End Users): The main users of the application who will rely on roomM8 to manage their shared living spaces. Their satisfaction and feedback are crucial for the success of the application.
Landlords and Property Managers: Individuals who might use the application to manage multiple rooms or properties. They are interested in features that help them oversee and coordinate their tenants' activities.
Developers: The team responsible for building and maintaining the application. They need clear and maintainable code to work efficiently and effectively.
Product Managers: Individuals who define the application's features and ensure they meet user needs and business goals. They require detailed documentation to guide development and track progress.

By addressing the needs and concerns of these stakeholders, roomM8 aims to provide a valuable and efficient solution for roommates to manage their shared living spaces.

Architecture and Component Design

High-Level Architecture

The high-level architecture of roomM8 is based on a client-server model. The backend is implemented using Spring Boot, which handles business logic, data management, and user authentication. The frontend is built using Next.js, Tailwind CSS, and Shadcn/UI for a responsive and user-friendly interface. MySQL is used as the database to store all persistent data.

High-Level Architecture Diagram

The High level architecture diagram can be found below:

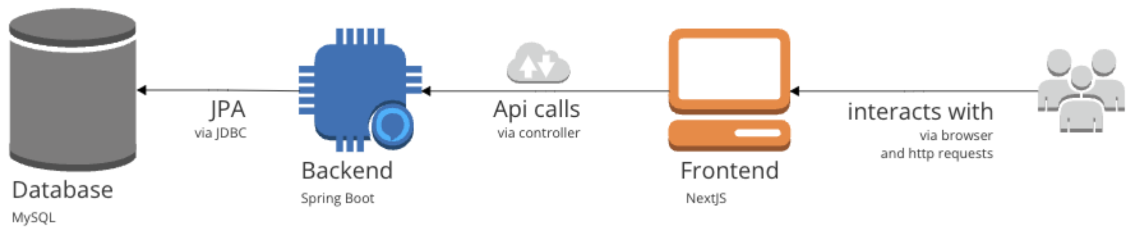


Figure 1 High level Architecture Diagram

Client (Users): Interacts with the application through a web browser.
Frontend (Next.js, Tailwind CSS, Shadcn/UI): Provides the user interface and communicates with the backend via API calls.
Backend (Spring Boot): Handles business logic, authentication, and data management.
Database (MySQL): Stores persistent data, such as user information, rooms, tasks, announcements, and grocery lists.

Component or Module Description

1. Authentication Module: Manages user registration, login, password reset, and session handling. Interacts with the user module to manage user data.
2. User Module: Manages user profiles and associated data. Interacts with the authentication module for user creation and updates.
3. Room Module. Interacts with the user module to associate users with rooms.
4. Task Management Module: Manages task creation, updates, assignments, and completion status. Interacts with the room module to associate tasks with rooms and users.
5. Announcement Module: Manages creation, updates, and deletion of announcements. Interacts with the room module to associate announcements with rooms.
6. Grocery List Module: Manages creation, updates, and deletion of grocery lists and items. Interacts with the room module to associate grocery lists with rooms.

7. Expense Module: Manages creation, updates, and deletion of expenses and settle ups with users in the room. Interacts with the room module to associate Expenses, Settle ups, and activities

Sequence Diagrams or Flow Diagrams

Authentication Flow Diagram

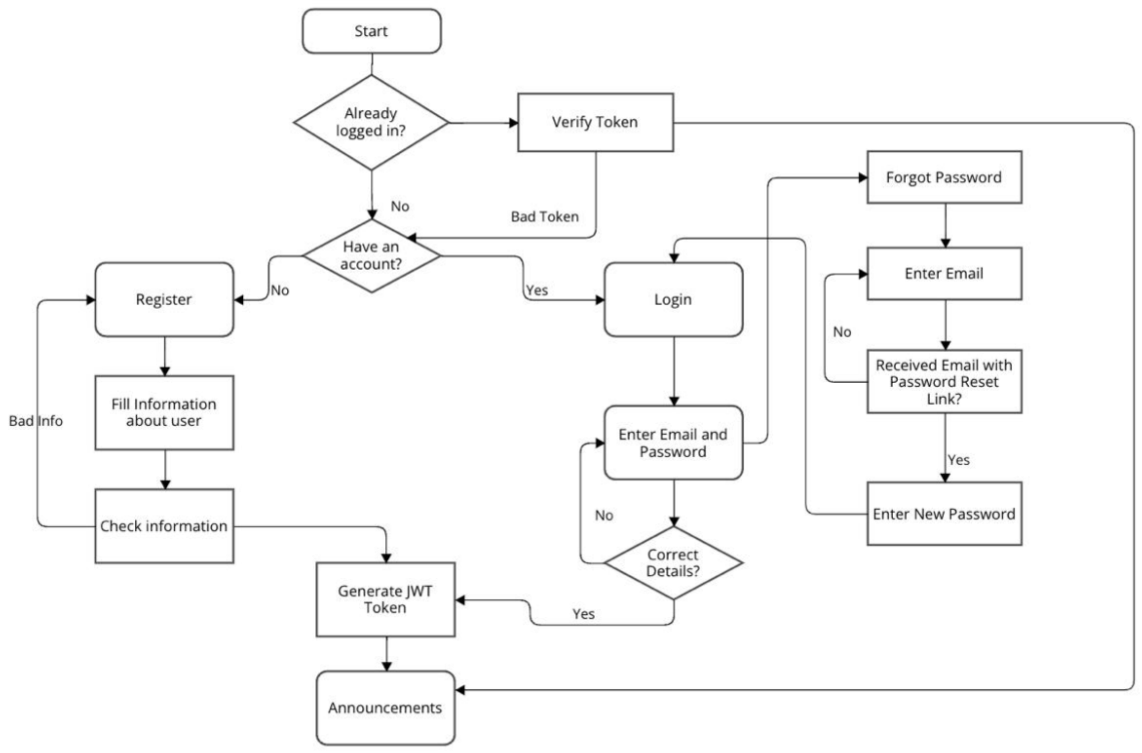


Figure 2 Authentication Flow Diagram

Tasks Flow Diagram

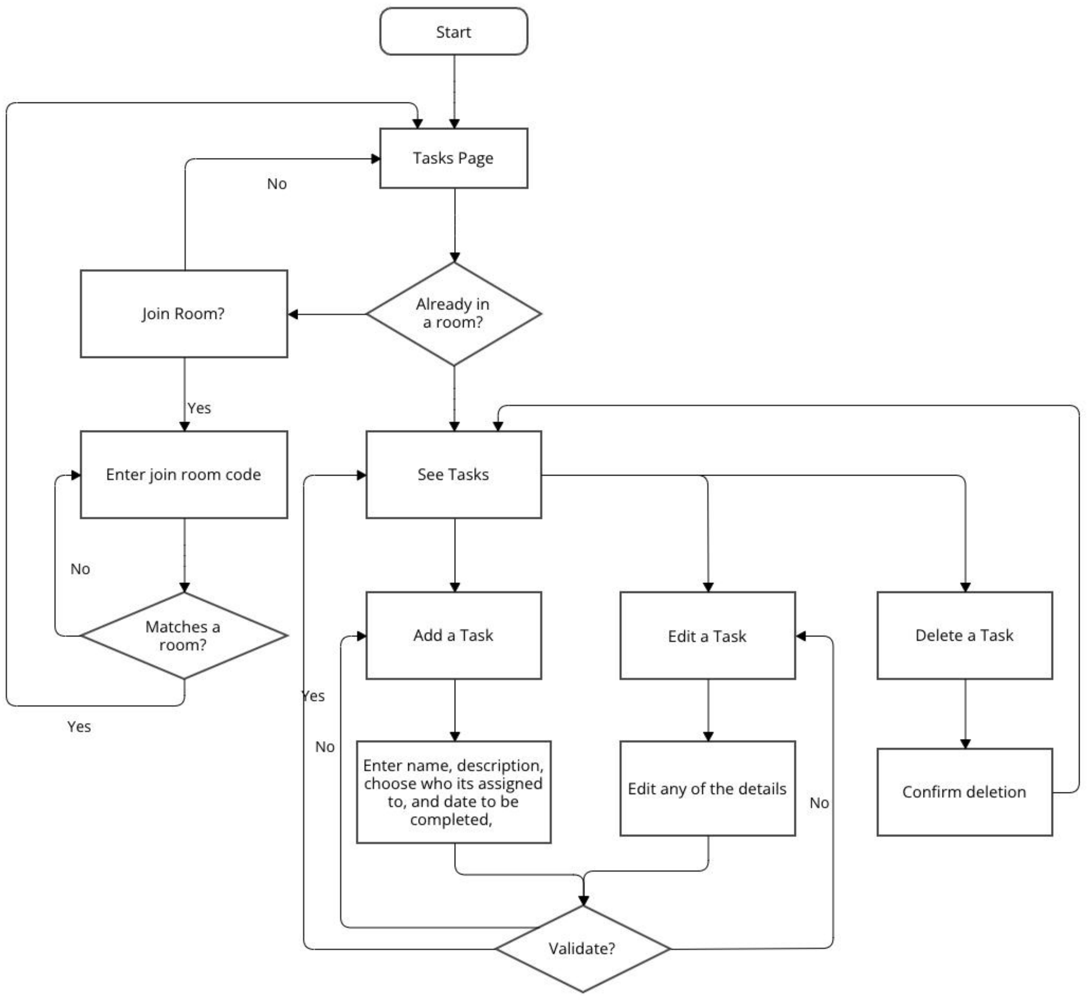


Figure 3 Task Flow Diagram

Profile and Room Setup Flow Diagram

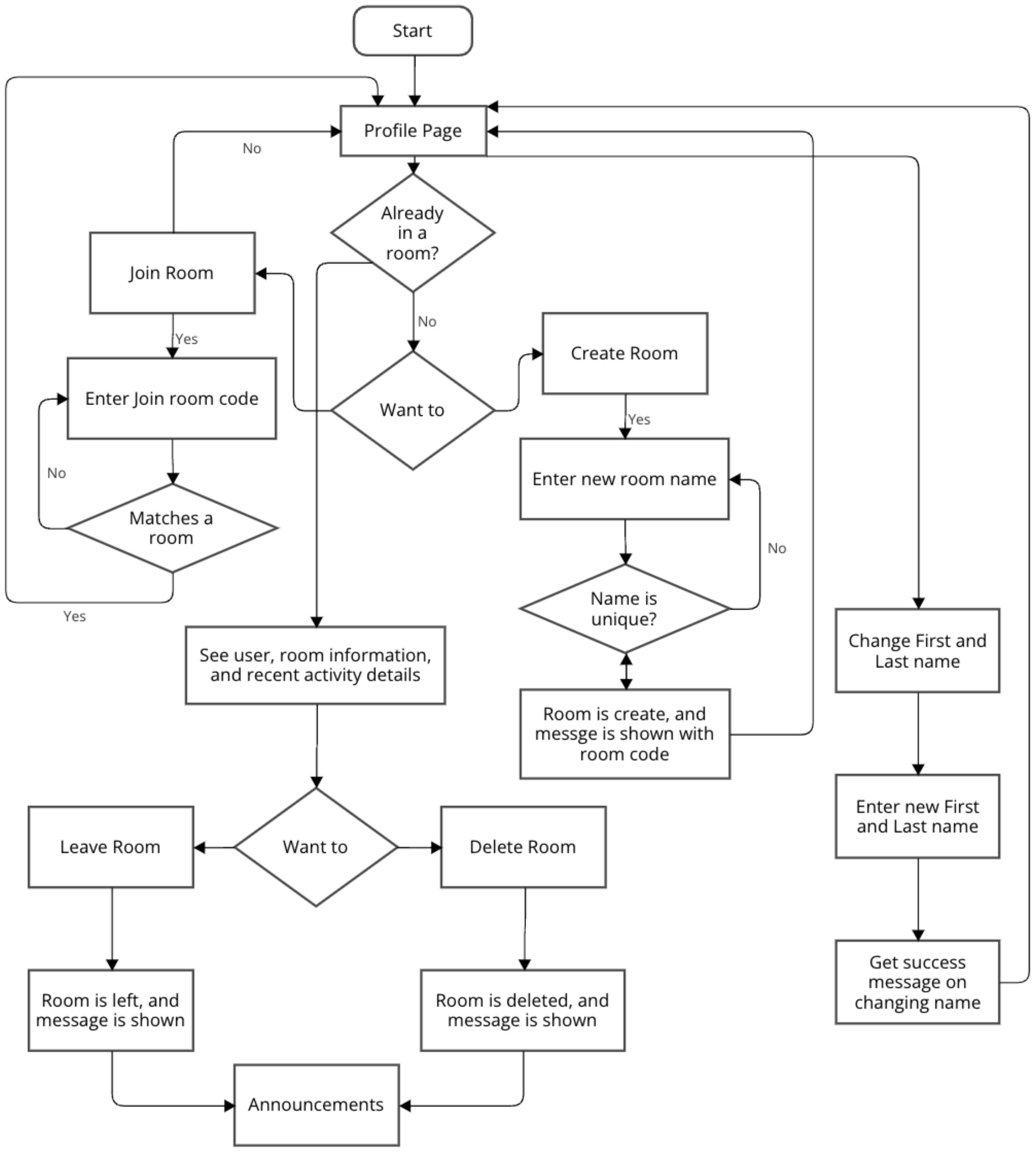

Start
Profile Page
Already in a room?
No
Join Room
Yes
Enter Join room code
No
Matches a room
Yes
Want to
Create Room
Yes
Enter new room name
No
Name is unique?
Room is create, and messge is shown with room code
Change First and Last name
Enter new First and Last name
Get success message on changing name
See user, room information, and recent activity details
Want to
Leave Room
Delete Room
Room is left, and message is shown
Room is deleted, and message is shown
Announcements

Figure 4 Profile and Room setup Flow Diagram

Expense Flow Diagram

The Expense Flow diagram can be found here

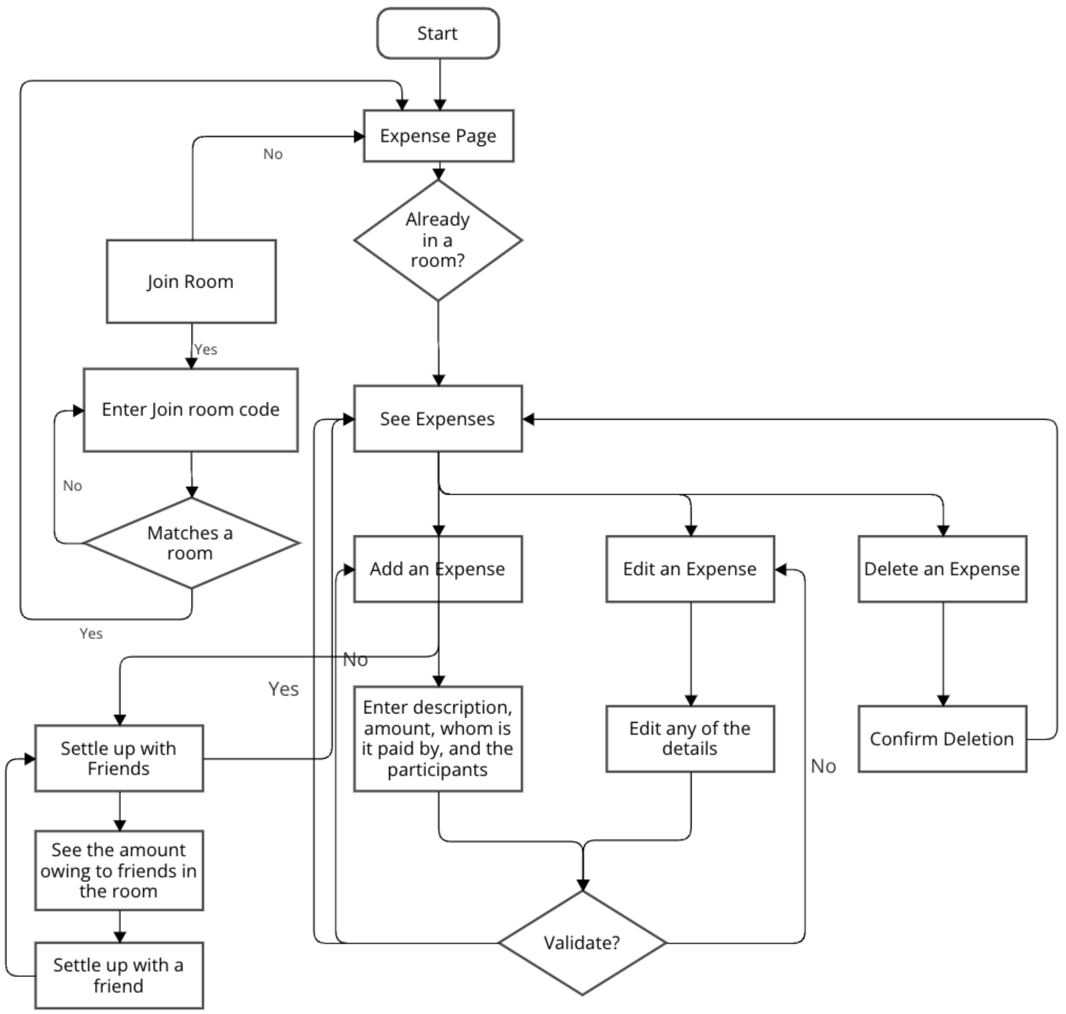


Figure 5 Expenses Flow Diagram

Data Design

Here we look at the overall design and also each data element, detailing its role, type, length, constraints (if any), a short description, and an example for each particular data element.

First, we see the figure that we obtain by performing reverse engineering in Workbench.

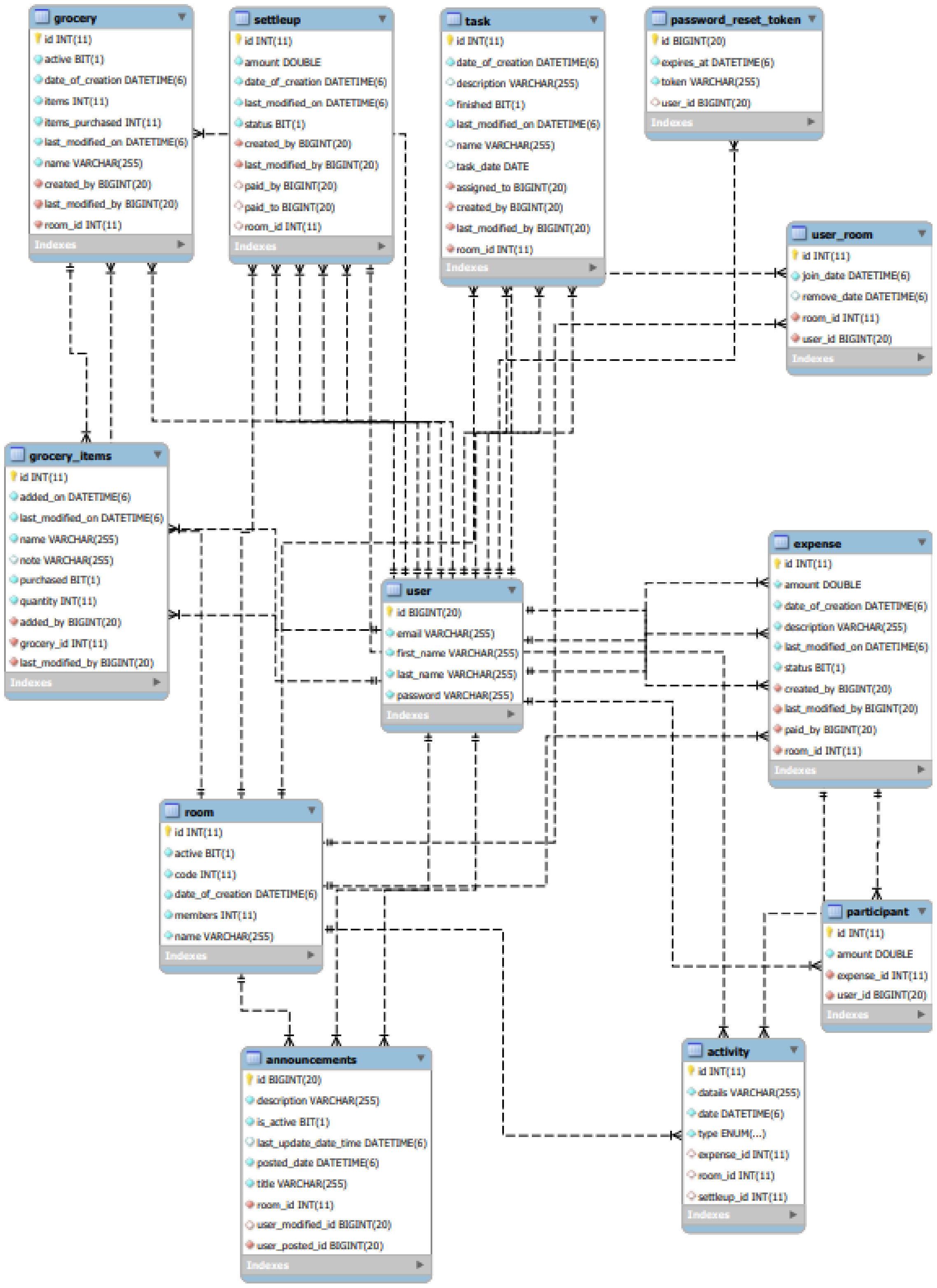


Figure 6 Reverse Engineered MySQL Workbench figure

User

Column Name	Data Type	Length	Constraints	Description	Example
id	BIGINT	20	PRIMARY KEY, NOT NULL, AUTO_INCREMENTED	Unique identifier for the user	1
email	VARCHAR	255	NOT NULL, UNIQUE	User Email	"test@gmail.com"
first_name	VARCHAR	255	NOT NULL	User First Name	"Jay"
last_name	VARCHAR	255	NOT NULL	User Last Name	"Patel"
password	VARCHAR	255	NOT NULL	Store in encrypted form	"$2a$10$token"

Room

Column Name	Data Type	Length	Constraints	Description	Example
id	INT	11	PRIMARY KEY, NOT NULL, AUTO_INCREMENTED	Unique identifier for room	1
active	BIT	1	NOT NULL	Indicates if the room is active or inactive	1
date_of_creation	DATETIME	6	NOT NULL		2024-08-02 13:56:15.000000
code	INT	11	NOT NULL, UNIQUE	Room code	56172

members	INT	11	NOT NULL	Number of members in the room.	4
name	VARCHAR	255	NOT NULL, UNIQUE	Name of the room	“South Park”

User_Room

Column Name	Data Type	Length	Constraints	Description	Example
id	INT	11	PRIMARY KEY, NOT NULL, AUTO_INCREMENTED	Unique identifier for user room	1
join_date	DATETIME	6	NOT NULL	Join date of the user in the room	2024-08-02 13:56:15.000000
remove_date	DATETIME	6		leave date of the user from the room	2024-08-02 13:56:15.000000
user_id	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	Associated user	1
room_id	INT	11	FOREIGN KEY (room.id), NOT NULL, UNIQUE	Associated room	1

Announcements

Column Name	Data Type	Length	Constraints	Description	Example
id	BIGINT	20	PRIMARY KEY, NOT NULL, AUTO_INCREMENTED	Unique identifier for announcement	1

description	VARCHAR	255	NOT NULL	Announcement Description	"Meeting at 5 PM"
is_active	BIT	1	NOT NULL	Indicates if the announcement is active or inactive	1
last_update_date_time	DATETIME	6		Last update date and time	2024-08-02 13:57:13.000000
posted_date	DATETIME	6	NOT NULL	Date when the announcement was posted	2024-08-02 13:57:13.000000
title	VARCHAR	255	NOT NULL	Title of the announcement	"Team Meeting"
room_id	INT	11	FOREIGN KEY (room.id), NOT NULL	Associated room	1
user_posted_id	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	User who posted the announcement	1
user_modified_id	BIGINT	20	FOREIGN KEY (user.id)	User who modified the announcement	1

Task

Column Name	Data Type	Length	Constraints	Description	Example

id	INT	11	PRIMARY KEY, NOT NULL, AUTO_INCREMENTED	Unique identifier for the task	1
date_of_creation	DATETIME	6	NOT NULL	Date of creation	2024-08-02 13:57:13.000000
description	VARCHAR	255	NOT NULL	Task description	"Clean the kitchen"
finished	BIT	1	NOT NULL	Indicates if the task is finished or pending	0
last_modified_on	DATETIME	6	NOT NULL	Last modified date	2024-08-02 13:57:13.000000
name	VARCHAR	255	NOT NULL	Name of the task	"Kitchen Cleaning"
task_date	DATE	6	NOT NULL	Date of the task	2024-08-02
created_by	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	User who created the task	1
last_modified_by	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	Last update date and time	1
assigned_to	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	User assigned to the task	2
room_id	INT	11	FOREIGN KEY (room.id), NOT NULL	Associated room	1

Expense

Column Name	Data Type	Length	Constraints	Description	Example
id	INT	11	PRIMARY KEY, NOT NULL, AUTO_INCREMENTED	Unique identifier for expense	1
amount	DOUBLE		NOT NULL	Amount for the expense	20.0
date_of_creation	DATETIME	6	NOT NULL	Date of creation	2024-08-02 13:57:13.000000
description	VARCHAR	255	NOT NULL	Expense description	"Groceries"
last_modified_on	DATETIME	6	NOT NULL	Last modified date	2024-08-02 13:57:13.000000
active	BIT	1	NOT NULL	Indicates if the expense is active or inactive	1
created_by	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	User who created the expense	1
last_modified_by	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	User who update the expense	1
paid_by	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	User who paid for the expense	1
room_id	INT	11	FOREIGN KEY (room.id), NOT NULL	Associated room	1

Participant

Column Name	Data Type	Length	Constraints	Description	Example
id	INT	11	PRIMARY KEY, NOT NULL, AUTO_INCREMENTED	Unique identifier for the participant	1
amount	DOUBLE		NOT NULL	Amount for the participant	10.0
expense_id	INT	11	FOREIGN KEY (expense.id), NOT NULL	Associated expense	1
user_id	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	Associated user	1

Activity

Column Name	Data Type	Length	Constraints	Description	Example
id	INT	11	PRIMARY KEY, NOT NULL, AUTO_INCREMENTED	Unique identifier for activity	1
details	VARCHAR	255	NOT NULL	Activity description	"Jay added 'Milk'"
type	ENUM (EXPENSE_ADD, EXPENSE_EDIT,		NOT NULL	Type of Activity	"EXPENSE_ADD"

	EXPENSE_DELETE, SETTLEUP_ADD, SETTLEUP_EDIT, SETTLEUP_DELETE)				
date	DATETIME	6	NOT NULL	Date and time of the activity	2024-08-02 13:57:13.000000
expense_id	INT	11	FOREIGN KEY (expense.id)	Associated expense	1
settleupr_id	INT	11	FOREIGN KEY (settleup.id)	Associated settleup	1
room_id	INT	11	FOREIGN KEY (room.id)	Associated room	1

SettleUp

Column Name	Data Type	Length	Constraints	Description	Example
id	INT	11	PRIMARY KEY, NOT NULL, AUTO_INCREMENTED	Unique identifier for settleup	1
amount	DOUBLE		NOT NULL	Amount for settleup	50.0
date_of_creation	DATETIME	6	NOT NULL	Date of creation	2024-08-02 13:57:13.000000
last_modified_on	DATETIME	6	NOT NULL	Last modified date	2024-08-02 13:57:13.000000

status	BIT	1	NOT NULL	Indicates if the settleup is active or inactive	1
created_by	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	User who created the settleup	1
last_modified_by	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	User who update the settleup	1
paid_by	BIGINT	20	FOREIGN KEY (user.id)	User who paid for the settleup	1
paid_to	BIGINT	20	FOREIGN KEY (user.id)	User who received the payment	1
room_id	INT	11	FOREIGN KEY (room.id)	Associated room	1

Grocery

Column Name	Data Type	Length	Constraints	Description	Example
id	INT	11	PRIMARY KEY, NOT NULL, AUTO_INCREMENTED	Unique identifier for grocery	1
active	BIT	1	NOT NULL	Indicates if the grocery is active or inactive	1
date_of_creation	DATETIME	6	NOT NULL	Date of creation	2024-08-02 13:57:13.000000

items	INT	11	NOT NULL	Total grocery items present in this list	5
items_purchased	INT	11	NOT NULL	Number of items purchased in this list	2
last_modified_on	DATETIME	6	NOT NULL	Last modified date	2024-08-02 13:57:13.000000
name	VARCHAR	255	NOT NULL, UNIQUE	Name of the grocery	"Weekly Grocery"
room_id	INT	11	FOREIGN KEY (room.id), NOT NULL	Associated room	1
created_by	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	User who created the grocery	1
last_modified_by	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	User who last modified the grocery	1

Grocery_Items

Column Name	Data Type	Length	Constraints	Description	Example
id	INT	11	PRIMARY KEY, NOT NULL, AUTO_INCREMENTED	Unique identifier for grocery item	1
added_on	DATETIME	6	NOT NULL	Date when the item was added	2024-08-02 13:57:13.000000

last_modified_on	DATETIME	6	NOT NULL	Last modified date	2024-08-02 13:57:13.000000
name	VARCHAR	255	NOT NULL, UNIQUE	Name of the item	"Apple"
note	VARCHAR	255		Note for the item	
quantity	INT	11	NOT NULL	Quantity of the item	10
purchased	BIT	1	NOT NULL	Indicates if the item is purchased or pending	0
grocery_id	INT	11	FOREIGN KEY (grocery.id), NOT NULL	Associated grocery	1
added_by	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	User who added the item	1
last_modified_by	BIGINT	20	FOREIGN KEY (user.id), NOT NULL	User who last modified the item	1

Password_Reset_Token

Column Name	Data Type	Length	Constraints	Description	Example
id	BIGINT	20	PRIMARY KEY, NOT NULL, AUTO_INCREMENTED	Unique identifier for the token	1

expires_at	DATETIME	6	NOT NULL	Expiry date of the token	2024-08-02 13:57:13.000000
token	VARCHAR	255	NOT NULL, UNIQUE	Token string	"f9a9fa2a-9193-4727-983e-3d73713500e1"
user_id	BIGINT	20	FOREIGN KEY (user.id)	Associated user	1

Interface Design

Internal Interfaces

Internal interfaces describe how different components within the system interact with each other to provide the desired functionalities. These interactions are primarily through APIs and direct method calls within the backend, and API calls from the frontend to the backend.

Authentication Module

- API Endpoints:
  - /api/v1/auth/register: Accepts user registration data and creates a new user.
  - /api/v1/auth/login: Authenticates the user and returns a JWT token and room information.
  - /users/password/forgot: Handles forgot password requests.
  - /users/password/reset: Handles password reset requests.
- Interactions:
  - Communicates with the User Module to create and manage user records.
  - Generates JWT tokens for authenticated sessions.

User Module

- API Endpoints:
  - /api/v1/profile: Retrieves and updates user profile information.
  - /api/v1/rooms: Manages user-room associations.
- Interactions:
  - Works with the Authentication Module for user-related operations.
  - Interacts with the Room Module to manage user-room relationships.

Room Module

- API Endpoints:
  - /api/v1/room/create: Creates a new room.
  - /api/v1/room/update: Updates room details.
  - /api/v1/room/join: Handles requests to join a room using a unique room code.
  - /api/v1/room/delete/{id}: Deletes a room with the given id.
  - /api/v1/room/leave: Handles requests to leave a room.
  - /api/v1/room/get/all/users: Retrieves all users from the room.
- Interactions:
  - Coordinates with the User Module to manage room memberships.
  - Interfaces with the Task Management, Announcement, and Grocery List Modules to associate tasks, announcements, expenses, settle ups, and grocery lists with rooms.

Task Management Module

- API Endpoints:
  - /task/add: Creates a new task.
  - /task/update: Updates task details.
  - /task/delete: Deletes a task.
  - /task/get/all: Retrieves tasks associated with a room.

- Interactions:
  - Works with the Room Module to associate tasks with specific rooms and users.

Announcement Module

- API Endpoints:
  - /api/v1/announcements/create: Creates a new announcement.
  - /api/v1/announcements/update/{id}: Updates announcement details.
  - /api/v1/announcements/delete/{id}: Deletes an announcement.
  - /api/v1/announcements/get/all: Retrieves announcements associated with a room.
  - /api/v1/announcements/get/{id}: Retrieves announcement for a particular id
- Interactions:
  - Interacts with the Room Module to manage announcements for specific rooms.

Grocery List Module

- API Endpoints:
  - /api/v1/grocery-list/add: Creates a new grocery list.
  - /api/v1/grocery-list/get/all: Gets all the grocery list.
  - /api/v1/grocery-list/get/{id}: Gets a particular grocery list.
  - /api/v1/grocery-list/delete/{id}: Deletes a grocery list with the given id.
  - /api/v1/grocery-list/item/add: Adds an item to a grocery list.
  - /api/v1/grocery-list/item/get/{id}: Gets a particular grocery item.
  - /api/v1/grocery-list/item/get/all/{id}: Gets all the grocery items for the given id of the grocery list.
  - /api/v1/grocery-list/item/update: Updates an item's details.
  - /api/v1/grocery-list/item/update/purchased:Marks the given grocery item as purchased.
  - /api/v1/grocery-list/item/delete/{id}: Deletes an item from a grocery list.
- Interactions:
  - Coordinates with the Room Module to associate grocery lists with specific rooms.

Expense Module

- API Endpoints:
  - /api/v1/expense/add: Creates a new expense.
  - /api/v1/expense/update: Updates an existing expense.
  - /api/v1/expense/delete/{id}: Deletes an expense by its id.
  - /api/v1/expense/get/{id}: Gets an expense with the given id.
  - /api/v1/expense/get/all/{type}: Gets all expense by its type which can be ‘all’, ‘active’, ‘inactive’.
- Interactions:
  - Coordinates with the Room Module to associate expenses with specific rooms and users.

Settle up Module

- API Endpoints:

    - /api/v1/settleup/calculations: Gets all settle-up calculations for the current user and room.
    - /api/v1/settleup/get/all/{type}: Get all settle-ups of a specific type for the current user and room.
    - /api/v1/settleup/add: Creates a new settle-up.
    - /api/v1/settleup/update: Update an existing settle-up.
    - /api/v1/settleup/delete/{id}: Delete a specific settle-up by its ID.
- Interactions:
    - Coordinates with the Room Module to associate settle ups with specific rooms and users.

Activity Module

- API Endpoints:
    - /api/v1/activity/get/all/expense: Get all expense activities for the current room.
    - /api/v1/activity/get/all/settleup: Get all settle-up activities for the current room.
    - /api/v1/activity/all: Get all activities for the current room.
- Interactions:
    - Coordinates with the Room Module to associate activities with specific rooms and users.

External Interfaces

External interfaces describe how the system interacts with external entities, primarily users and third-party services.

User Interfaces

- **Web Interface:** The primary interface for users to interact with roomM8 is through a web application. The frontend, built with Next.js, Tailwind CSS, and Shadcn/UI, provides a responsive and intuitive user experience.
    - Login and Registration Pages: Allow users to create accounts and log in to the system.
    - Profile Page: Enables users to view and update their profile information, and see the recent Expense activities.
    - Room Management: Allows users to create, join, and manage rooms.
    - Task Management: Enables users to create, update, and delete tasks.
    - Announcement Management: Allows users to create, update, and delete announcements.
    - Grocery List Management: Enables users to create, update, and delete grocery lists and items.
    - Expense Management: Enables users to add, edit, and delete expenses along with allowing them to settle up with friends in the same room.

Third-Party Services

Email Service: Used for sending verification emails, password reset emails, and other notifications which sends an email to a specified recipient.

Authentication Service: Manages user authentication and session management using JWT tokens.

API Documentation

Authentication API

Register User

- Endpoint: POST /api/v1/auth/register

  Request Body:
  ```
  {
    "firstName": "User",
    "lastName": "Example",
    "email": "user@example.com",
    "password": "user@123"
  }
  ```

  Response:
  ```
  {
    "id": 10,
    "firstName": "User",
    "lastName": "Example",
    "email": "user@example.com",
    "password":
  "$2a$10$hY1M6X9LXz4mZ3bA4diHPOQbsErRLR7LT.VZRsLqIHy56azuT3QYu"
  }
  ```

Login User

- Endpoint: POST /api/v1/auth/login

  Request Body:
  ```
  {
    "email": "user@example.com",
    "password": "user@123"
  }
  ```

  Response:
  ```
  {
    "roomDTOResponse": null,
    "token":
  "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ1c2VyQGV4YW1wbGUuY29tIiwiaWF0IjoxNzIyODE2O
  TQ1LCJleHAiOjE3MjI5MDMzNDV9.Z3sTj1D-iC-
  JRSIqAv4o5tjfsqn8_ieYP8TOdiIAaIbOXK6Qyb7kKcqhi7bzsmmoLdfxmpSNRBr_B_71PDwc0
  Q",
  ```

  "email": "user@example.com"
}

Verify Token

- Endpoint: POST /api/v1/auth/verify-token

  Request Header:
  Authorization: Bearer
  eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ1c2VyQGV4YW1wbGUuY29tIiwiaWF0IjoxNzIyODE2OTQ1LCJleHAiOjE3MjI5MDMzNDV9.Z3sTj1D-iC-JRSIqAv4o5tjfsqn8_ieYP8TOdiIAaIbOXK6Qyb7kKcqhi7bzsmmoLdfxmpSNRBr_B_71PDwc0Q

  Response:
  ```
  {
    "tokenValid": "true"
  }
  ```

Forgot Password

- Endpoint: POST /users/password/forgot

  Request Body:
  ```
  {
    "email": "user@example.com"
  }
  ```

  Response:
    - Status of the response will be 200 if the link is sent, else 400

Reset Password

- Endpoint: POST /api/v1/password/reset

  Request Body:
  ```
  {
    "token":
  "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ1c2VyQGV4YW1wbGUuY29tIiwiaWF0IjoxNzIyODE2OTQ1LCJleHAiOjE3MjI5MDMzNDV9.Z3sTj1D-iC-JRSIqAv4o5tjfsqn8_ieYP8TOdiIAaIbOXK6Qyb7kKcqhi7bzsmmoLdfxmpSNRBr_B_71PDwc0Q",
    "password": "newpassword",
    confirmPassword: "newpassword"
  }
  ```

  Response:

    - Status of the response will be 200 if the password is reset, else 400

Expense API

- Add Expense

  Endpoint : /api/v1/expense/add (POST)

  Request

```
{
   "description":"Milk",
   "amount":200,
   "paidBy":"pateljaysanjaybhai8@gmail.com",
   "participantEmails":["kenee@gmail.com", "testing123@gmail.com"]
}
```

  Response

```
{
   "id": 2,
   "room": {
      "id": 1,
      "name": "Ogilvie",
      "members": 4,
      "code": 56172,
      "active": true
   },
   "description": "Milk",
   "amount": 200.0,
   "paidBy": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
   },
   "participants": [
      {
         "id": 2,
         "user": {
            "id": 1,
            "firstName": "Kenny",
            "lastName": "Patel",
            "email": "kenee@gmail.com"
            },
            "amount": 66.66666666666667
        },
        {
            "id": 3,
            "user": {
                "id": 8,
                "firstName": "test",
                "lastName": "test",
                "email": "testing123@gmail.com"
            },
            "amount": 66.66666666666667
        }
    ],
    "status": true,
    "dateOfCreation": "2024-08-04T20:39:12.0397534",
    "lastModifiedOn": "2024-08-04T20:39:12.0397534",
    "createdBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
    },
    "lastModifiedBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
    }
}
```

- Get Expense
  Endpoint : /api/v1/expense/get/2 (GET)

  Response

```
{
    "id": 2,
    "room": {
        "id": 1,
        "name": "Ogilvie",
        "members": 4,
        "code": 56172,
        "active": true
```

```
},
"description": "Milk",
"amount": 200.0,
"paidBy": {
  "id": 9,
  "firstName": "Jay",
  "lastName": "Patel",
  "email": "pateljaysanjaybhai8@gmail.com"
},
"participants": [
  {
    "id": 2,
    "user": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "amount": 66.66666666666667
  },
  {
    "id": 3,
    "user": {
      "id": 8,
      "firstName": "test",
      "lastName": "test",
      "email": "testing123@gmail.com"
    },
    "amount": 66.66666666666667
  }
],
"status": true,
"dateOfCreation": "2024-08-04T20:39:12",
"lastModifiedOn": "2024-08-04T20:39:12",
"createdBy": {
  "id": 9,
  "firstName": "Jay",
  "lastName": "Patel",
  "email": "pateljaysanjaybhai8@gmail.com"
},
"lastModifiedBy": {
  "id": 9,
  "firstName": "Jay",
  "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
    }
}
```

- Delete Expense

  Endpoint : /api/v1/expense/delete/1 (DELETE)

  Response

  “Success”

- Get All expense

  Endpoint : /api/v1/expense/get/all/all (GET)

Response

```
[
  {
    "id": 1,
    "room": {
      "id": 1,
      "name": "Ogilvie",
      "members": 4,
      "code": 56172,
      "active": true
    },
    "description": "Milk",
    "amount": 100.0,
    "paidBy": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "participants": [
      {
        "id": 1,
        "user": {
          "id": 3,
            "firstName": "Jay",
            "lastName": "Patel",
            "email": "jay@gmail.com"
          },
          "amount": 50.0
        }
      ],
      "status": false,
      "dateOfCreation": "2024-08-02T12:00:33",
      "lastModifiedOn": "2024-08-04T20:49:46",
      "createdBy": {
        "id": 3,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "jay@gmail.com"
      },
      "lastModifiedBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      }
    },
    {
      "id": 2,
      "room": {
        "id": 1,
        "name": "Ogilvie",
        "members": 4,
        "code": 56172,
        "active": true
      },
      "description": "Milk",
      "amount": 200.0,
      "paidBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      },
      "participants": [
        {
          "id": 2,
          "user": {
                "id": 1,
                "firstName": "Kenny",
                "lastName": "Patel",
                "email": "kenee@gmail.com"
            },
            "amount": 66.66666666666667
        },
        {
            "id": 3,
            "user": {
                "id": 8,
                "firstName": "test",
                "lastName": "test",
                "email": "testing123@gmail.com"
            },
            "amount": 66.66666666666667
        }
    ],
    "status": true,
    "dateOfCreation": "2024-08-04T20:39:12",
    "lastModifiedOn": "2024-08-04T20:39:12",
    "createdBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
    },
    "lastModifiedBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
    }
  }
]
```

- Get All Active Expense

  Endpoint : /api/v1/expense/get/all/active (GET)

Response

```
[
  {
```

```
"id": 2,
"room": {
  "id": 1,
  "name": "Ogilvie",
  "members": 4,
  "code": 56172,
  "active": true
},
"description": "Milk",
"amount": 200.0,
"paidBy": {
  "id": 9,
  "firstName": "Jay",
  "lastName": "Patel",
  "email": "pateljaysanjaybhai8@gmail.com"
},
"participants": [
  {
    "id": 2,
    "user": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "amount": 66.66666666666667
  },
  {
    "id": 3,
    "user": {
      "id": 8,
      "firstName": "test",
      "lastName": "test",
      "email": "testing123@gmail.com"
    },
    "amount": 66.66666666666667
  }
],
"status": true,
"dateOfCreation": "2024-08-04T20:39:12",
"lastModifiedOn": "2024-08-04T20:39:12",
"createdBy": {
  "id": 9,
  "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      },
      "lastModifiedBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      }
    }
]
```

- Get Inactive Expense

  Endpoint : /api/v1/expense/get/all/inactive (GET)

  Response:

```
[
  {
    "id": 1,
    "room": {
      "id": 1,
      "name": "Ogilvie",
      "members": 4,
      "code": 56172,
      "active": true
    },
    "description": "Milk",
    "amount": 100.0,
    "paidBy": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "participants": [
      {
        "id": 1,
        "user": {
          "id": 3,
          "firstName": "Jay",
          "lastName": "Patel",
          "email": "jay@gmail.com"
        },
                "amount": 50.0
            }
        ],
        "status": false,
        "dateOfCreation": "2024-08-02T12:00:33",
        "lastModifiedOn": "2024-08-04T20:49:46",
        "createdBy": {
            "id": 3,
            "firstName": "Jay",
            "lastName": "Patel",
            "email": "jay@gmail.com"
        },
        "lastModifiedBy": {
            "id": 9,
            "firstName": "Jay",
            "lastName": "Patel",
            "email": "pateljaysanjaybhai8@gmail.com"
        }
    }
]
```

- Update Expense

  Endpoint : /api/v1/expense/udpate (PUT)

  Request

```
{
    "id":2,
    "description":"Milk",
    "amount":400,
    "paidBy":"pateljaysanjaybhai8@gmail.com",
    "participantEmails":["kenee@gmail.com"]
}
```

  Response

```
{
    "id": 2,
    "room": {
        "id": 1,
        "name": "Ogilvie",
        "members": 4,
        "code": 56172,
        "active": true
      },
      "description": "Milk",
      "amount": 400.0,
      "paidBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      },
      "participants": [
        {
          "id": 2,
          "user": {
            "id": 1,
            "firstName": "Kenny",
            "lastName": "Patel",
            "email": "kenee@gmail.com"
          },
          "amount": 200.0
        }
      ],
      "status": true,
      "dateOfCreation": "2024-08-04T20:39:12.0397534",
      "lastModifiedOn": "2024-08-04T20:40:09.0542167",
      "createdBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      },
      "lastModifiedBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      }
    }
```

- Get Expense Calculations

  Endpoint : /api/v1/settleup/calculations (GET)

  Response:

```
[
  {
    "from": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "to": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
    },
    "amount": 66.66666666666667,
    "status": true
  },
  {
    "from": {
      "id": 8,
      "firstName": "test",
      "lastName": "test",
      "email": "testing123@gmail.com"
    },
    "to": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
    },
    "amount": 66.66666666666667,
    "status": true
  }
]
```

- Add Settleup

  Endpoint : /api/v1/settleup/add (POST)

  Request

  ```
  {
  ```

```
    "to":"Jay@gmail.com",
    "amount":100
}
```

Response

```
{
    "id": 1,
    "room": {
        "id": 1,
        "name": "Ogilvie",
        "members": 4,
        "code": 56172,
        "active": true
    },
    "paidBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
    },
    "paidTo": {
        "id": 3,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "jay@gmail.com"
    },
    "amount": 100.0,
    "status": true,
    "dateOfCreation": "2024-08-04T20:57:27.1022872",
    "lastModifiedOn": "2024-08-04T20:57:27.1022872",
    "createdBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
    },
    "lastModifiedBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
    }
}
```

- Get Settleup

  Endpoint : /api/v1/settleup/get/1 (GET)

  Response

```
{
  "id": 1,
  "room": {
    "id": 1,
    "name": "Ogilvie",
    "members": 4,
    "code": 56172,
    "active": true
  },
  "paidBy": {
    "id": 9,
    "firstName": "Jay",
    "lastName": "Patel",
    "email": "pateljaysanjaybhai8@gmail.com"
  },
  "paidTo": {
    "id": 3,
    "firstName": "Jay",
    "lastName": "Patel",
    "email": "jay@gmail.com"
  },
  "amount": 100.0,
  "status": true,
  "dateOfCreation": "2024-08-04T20:57:27",
  "lastModifiedOn": "2024-08-04T20:57:27",
  "createdBy": {
    "id": 9,
    "firstName": "Jay",
    "lastName": "Patel",
    "email": "pateljaysanjaybhai8@gmail.com"
  },
  "lastModifiedBy": {
    "id": 9,
    "firstName": "Jay",
    "lastName": "Patel",
    "email": "pateljaysanjaybhai8@gmail.com"
  }
```

}

- Delete Settleup

  Endpoint : /api/v1/delete/2 (DELETE)

  Response

  “Success”

- Get All Settleup

  Endpoint : /api/v1/settleup/get/all/all (GET)

  Response

```
[
  {
    "id": 1,
    "room": {
      "id": 1,
      "name": "Ogilvie",
      "members": 4,
      "code": 56172,
      "active": true
    },
    "paidBy": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
    },
    "paidTo": {
      "id": 3,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "jay@gmail.com"
    },
    "amount": 100.0,
    "status": true,
    "dateOfCreation": "2024-08-04T20:57:27",
    "lastModifiedOn": "2024-08-04T20:57:27",
    "createdBy": {
      "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      },
      "lastModifiedBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      }
    },
    {
      "id": 2,
      "room": {
        "id": 1,
        "name": "Ogilvie",
        "members": 4,
        "code": 56172,
        "active": true
      },
      "paidBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      },
      "paidTo": {
        "id": 1,
        "firstName": "Kenny",
        "lastName": "Patel",
        "email": "kenee@gmail.com"
      },
      "amount": 150.0,
      "status": false,
      "dateOfCreation": "2024-08-04T21:01:03",
      "lastModifiedOn": "2024-08-04T21:01:38",
      "createdBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      },
      "lastModifiedBy": {
        "id": 9,
            "firstName": "Jay",
            "lastName": "Patel",
            "email": "pateljaysanjaybhai8@gmail.com"
        }
    }
]
```

- Get All Active Settleup List

  Endpoint : /api/v1/settleup/get/all/active (GET)

  Response:

```
[
    {
        "id": 1,
        "room": {
            "id": 1,
            "name": "Ogilvie",
            "members": 4,
            "code": 56172,
            "active": true
        },
        "paidBy": {
            "id": 9,
            "firstName": "Jay",
            "lastName": "Patel",
            "email": "pateljaysanjaybhai8@gmail.com"
        },
        "paidTo": {
            "id": 3,
            "firstName": "Jay",
            "lastName": "Patel",
            "email": "jay@gmail.com"
        },
        "amount": 100.0,
        "status": true,
        "dateOfCreation": "2024-08-04T20:57:27",
        "lastModifiedOn": "2024-08-04T20:57:27",
        "createdBy": {
            "id": 9,
            "firstName": "Jay",
            "lastName": "Patel",
            "email": "pateljaysanjaybhai8@gmail.com"
        },
        "lastModifiedBy": {
            "id": 9,
            "firstName": "Jay",
            "lastName": "Patel",
            "email": "pateljaysanjaybhai8@gmail.com"
        }
    }
]
```

- Get All Inactive Settleup

  Endpoint : /api/v1/settleup/get/all/inactive (GET)

  Response:

```
[
    {
        "id": 2,
        "room": {
            "id": 1,
            "name": "Ogilvie",
            "members": 4,
            "code": 56172,
            "active": true
        },
        "paidBy": {
            "id": 9,
            "firstName": "Jay",
            "lastName": "Patel",
            "email": "pateljaysanjaybhai8@gmail.com"
        },
        "paidTo": {
            "id": 1,
            "firstName": "Kenny",
            "lastName": "Patel",
            "email": "kenee@gmail.com"
        },
        "amount": 150.0,
        "status": false,
        "dateOfCreation": "2024-08-04T21:01:03",
        "lastModifiedOn": "2024-08-04T21:01:38",
        "createdBy": {
            "id": 9,
            "firstName": "Jay",
            "lastName": "Patel",
            "email": "pateljaysanjaybhai8@gmail.com"
        },
        "lastModifiedBy": {
            "id": 9,
            "firstName": "Jay",
            "lastName": "Patel",
            "email": "pateljaysanjaybhai8@gmail.com"
        }
    }
]
```

- Update Settleup
  Endpoint : /api/v1/settleup/update (PUT)

  Request

```
{
    "id":2,
    "to":"jay@gmail.com",
    "amount":250
}
```

  Response

```
{
    "id": 2,
    "room": {
        "id": 1,
        "name": "Ogilvie",
        "members": 4,
        "code": 56172,
        "active": true
    },
    "paidBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
    },
    "paidTo": {
        "id": 3,
        "firstName": "Jay",
    "lastName": "Patel",
    "email": "jay@gmail.com"
  },
  "amount": 250.0,
  "status": false,
  "dateOfCreation": "2024-08-04T21:01:03",
  "lastModifiedOn": "2024-08-04T21:05:38.0708689",
  "createdBy": {
    "id": 9,
    "firstName": "Jay",
    "lastName": "Patel",
    "email": "pateljaysanjaybhai8@gmail.com"
  },
  "lastModifiedBy": {
    "id": 9,
    "firstName": "Jay",
    "lastName": "Patel",
    "email": "pateljaysanjaybhai8@gmail.com"
  }
}
```

- Get All Expense Activity

  Endpoint : /api/v1/activity/get/all/expense (GET)

  Response:

```
[
 {
  "id": 1,
  "date": "2024-08-02T11:48:05",
  "details": "Jay added 'Milk'",
  "type": "Expense_ADD",
  "expense": {
   "id": 1,
   "room": {
    "id": 1,
    "name": "Ogilvie",
    "members": 4,
    "code": 56172,
    "active": true
   },
   "description": "Milk",
   "amount": 100,
    "paidBy": {
     "id": 1,
     "firstName": "Kenny",
     "lastName": "Patel",
     "email": "kenee@gmail.com"
    },
    "participants": [
     {
      "id": 1,
      "user": {
       "id": 3,
       "firstName": "Jay",
       "lastName": "Patel",
       "email": "jay@gmail.com"
      },
      "amount": 50
     }
    ],
    "status": false,
    "dateOfCreation": "2024-08-02T12:00:33",
    "lastModifiedOn": "2024-08-04T20:49:46",
    "createdBy": {
     "id": 3,
     "firstName": "Jay",
     "lastName": "Patel",
     "email": "jay@gmail.com"
    },
    "lastModifiedBy": {
     "id": 9,
     "firstName": "Jay",
     "lastName": "Patel",
     "email": "pateljaysanjaybhai8@gmail.com"
    }
   },
   "room": {
    "id": 1,
    "name": "Ogilvie",
    "members": 4,
    "code": 56172,
    "active": true
   }
  },
  {
   "id": 10,
"date": "2024-08-04T20:49:47",
"details": "Jay removed expense 'Milk'",
"type": "Expense_DELETE",
"expense": {
  "id": 1,
  "room": {
    "id": 1,
    "name": "Ogilvie",
    "members": 4,
    "code": 56172,
    "active": true
  },
  "description": "Milk",
  "amount": 100,
  "paidBy": {
    "id": 1,
    "firstName": "Kenny",
    "lastName": "Patel",
    "email": "kenee@gmail.com"
  },
  "participants": [
    {
      "id": 1,
      "user": {
        "id": 3,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "jay@gmail.com"
      },
      "amount": 50
    }
  ],
  "status": false,
  "dateOfCreation": "2024-08-02T12:00:33",
  "lastModifiedOn": "2024-08-04T20:49:46",
  "createdBy": {
    "id": 3,
    "firstName": "Jay",
    "lastName": "Patel",
    "email": "jay@gmail.com"
  },
  "lastModifiedBy": {
    "id": 9,
    "firstName": "Jay",
    "lastName": "Patel",
    "email": "pateljaysanjaybhai8@gmail.com"
   }
  },
  "room": {
   "id": 1,
   "name": "Ogilvie",
   "members": 4,
   "code": 56172,
   "active": true
  }
 }
]
```

- Get SettleUp Activity

  Endpoint : /api/v1/activity/get/all/settleup (GET)

  Response:

```
[
 {
  "id": 11,
  "date": "2024-08-04T20:57:27",
  "details": "Jay paid to 'Jay'",
  "type": "SETTLE_UP_Add",
  "settleup": {
   "id": 1,
   "room": {
    "id": 1,
    "name": "Ogilvie",
    "members": 4,
    "code": 56172,
    "active": true
   },
   "paidBy": {
    "id": 9,
    "firstName": "Jay",
    "lastName": "Patel",
    "email": "pateljaysanjaybhai8@gmail.com"
   },
   "paidTo": {
    "id": 3,
    "firstName": "Jay",
    "lastName": "Patel",
    "email": "jay@gmail.com"
    },
    "amount": 100,
    "status": true,
    "dateOfCreation": "2024-08-04T20:57:27",
    "lastModifiedOn": "2024-08-04T20:57:27",
    "createdBy": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
    },
    "lastModifiedBy": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
    }
  },
  "room": {
    "id": 1,
    "name": "Ogilvie",
    "members": 4,
    "code": 56172,
    "active": true
  }
},
{
  "id": 14,
  "date": "2024-08-04T21:05:38",
  "details": "Jay paid to(edited) 'Jay'",
  "type": "SETTLE_UP_EDIT",
  "settleup": {
    "id": 2,
    "room": {
      "id": 1,
      "name": "Ogilvie",
      "members": 4,
      "code": 56172,
      "active": true
    },
    "paidBy": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      },
      "paidTo": {
        "id": 3,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "jay@gmail.com"
      },
      "amount": 250,
      "status": false,
      "dateOfCreation": "2024-08-04T21:01:03",
      "lastModifiedOn": "2024-08-04T21:05:38",
      "createdBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      },
      "lastModifiedBy": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      }
    },
    "room": {
      "id": 1,
      "name": "Ogilvie",
      "members": 4,
      "code": 56172,
      "active": true
    }
  }
]
```

Grocery List

- Add Grocery List

  Endpoint : /api/v1/grocery-list/add (POST)

  Request
  {

```
   "name": "Testing List"
}
```

Response

```
{
   "id": 4,
   "name": "Testing List",
   "items": 0,
   "items_purchased": 0,
   "date_of_creation": "2024-08-04T21:37:37.5712894",
   "last_modified_on": "2024-08-04T21:37:37.5712894",
   "active": true,
   "grocery_items": [],
   "room": {
      "id": 1,
      "name": "Ogilvie",
      "members": 4,
      "code": 56172,
      "active": true
   },
   "created_by": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
   },
   "last_modified_by": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
   }
}
```

- Get Grocery List

  Endpoint : /api/v1/grocery-list/get/4 (GET)

  Response:

```
{
   "id": 4,
   "name": "Testing List",
   "items": 0,
   "items_purchased": 0,
    "date_of_creation": "2024-08-04T21:37:37",
    "last_modified_on": "2024-08-04T21:37:37",
    "active": true,
    "grocery_items": [],
    "room": {
      "id": 1,
      "name": "Ogilvie",
      "members": 4,
      "code": 56172,
      "active": true
    },
    "created_by": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
    },
    "last_modified_by": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
    }
}
```

- Get All Grocery List

  Endpoint : /api/v1/grocery-list/get/all (GET)

  Response:

```
[
  {
    "id": 1,
    "name": "Walmart",
    "items": 5,
    "items_purchased": 0,
    "date_of_creation": "2024-08-02T10:59:07",
    "last_modified_on": "2024-08-02T11:01:15",
    "active": true,
    "grocery_items": [
      {
        "id": 1,
        "name": "Milk",
        "quantity": 2,
    "note": "Scotsburn",
    "added_on": "2024-08-02T10:59:59",
    "last_modified_on": "2024-08-02T10:59:59",
    "purchased": false,
    "added_by": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "last_modified_by": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "groceryListName": null
  },
  {
    "id": 2,
    "name": "Eggs",
    "quantity": 2,
    "note": "dozen sized",
    "added_on": "2024-08-02T11:00:15",
    "last_modified_on": "2024-08-02T11:00:15",
    "purchased": false,
    "added_by": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "last_modified_by": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "groceryListName": null
  },
  {
    "id": 3,
    "name": "Chips",
    "quantity": 2,
    "note": "200g Cheetos",
    "added_on": "2024-08-02T11:00:32",
    "last_modified_on": "2024-08-02T11:00:32",
    "purchased": false,
    "added_by": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "last_modified_by": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "groceryListName": null
  },
  {
    "id": 4,
    "name": "Soda",
    "quantity": 4,
    "note": "1l for the party",
    "added_on": "2024-08-02T11:00:54",
    "last_modified_on": "2024-08-02T11:00:54",
    "purchased": false,
    "added_by": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "last_modified_by": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "groceryListName": null
  },
  {
    "id": 5,
    "name": "Napkins",
    "quantity": 2,
      "note": "pack of 100 pieces each",
      "added_on": "2024-08-02T11:01:15",
      "last_modified_on": "2024-08-02T11:01:15",
      "purchased": false,
      "added_by": {
        "id": 1,
        "firstName": "Kenny",
        "lastName": "Patel",
        "email": "kenee@gmail.com"
      },
      "last_modified_by": {
        "id": 1,
        "firstName": "Kenny",
        "lastName": "Patel",
        "email": "kenee@gmail.com"
      },
      "groceryListName": null
    }
  ],
  "room": {
    "id": 1,
    "name": "Ogilvie",
    "members": 4,
    "code": 56172,
    "active": true
  },
  "created_by": {
    "id": 1,
    "firstName": "Kenny",
    "lastName": "Patel",
    "email": "kenee@gmail.com"
  },
  "last_modified_by": {
    "id": 1,
    "firstName": "Kenny",
    "lastName": "Patel",
    "email": "kenee@gmail.com"
  }
},
{
  "id": 4,
  "name": "Testing List",
  "items": 0,
  "items_purchased": 0,
  "date_of_creation": "2024-08-04T21:37:37",
  "last_modified_on": "2024-08-04T21:37:37",
  "active": true,
  "grocery_items": [],
  "room": {
   "id": 1,
   "name": "Ogilvie",
   "members": 4,
   "code": 56172,
   "active": true
  },
  "created_by": {
   "id": 9,
   "firstName": "Jay",
   "lastName": "Patel",
   "email": "pateljaysanjaybhai8@gmail.com"
  },
  "last_modified_by": {
   "id": 9,
   "firstName": "Jay",
   "lastName": "Patel",
   "email": "pateljaysanjaybhai8@gmail.com"
  }
 }
]
```

- Delete Grocery List

  Endpoint : /api/v1/grocery-list/delete/3 (GET)

  Response

  “Success”

Grocery Item

- Add Grocery Items in List

  Endpoint : /api/v1/grocery-list/item/add (POST)

Request

```
{
   "name": "Aloo 2",
   "quantity":12,
   "note":"Freah",
   "groceryListName":"Testing List"
}
```

Response

```
{
   "id": 11,
   "name": "Aloo 2",
   "quantity": 12,
   "note": "Freah",
   "added_on": "2024-08-04T21:43:10.5899899",
   "last_modified_on": "2024-08-04T21:43:10.5899899",
   "purchased": false,
   "added_by": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
   },
   "last_modified_by": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
   },
   "groceryListName": null
}
```

- Get Item in the List

  Endpoint : /api/v1/grocery-list/item/get/1 (GET)

```
{
   "id": 1,
   "name": "Milk",
    "quantity": 2,
    "note": "Scotsburn",
    "added_on": "2024-08-02T10:59:59",
    "last_modified_on": "2024-08-02T10:59:59",
    "purchased": false,
    "added_by": {
        "id": 1,
        "firstName": "Kenny",
        "lastName": "Patel",
        "email": "kenee@gmail.com"
    },
    "last_modified_by": {
        "id": 1,
        "firstName": "Kenny",
        "lastName": "Patel",
        "email": "kenee@gmail.com"
    },
    "groceryListName": "Walmart"
}
```

- Update item in Grocery List

  Endpoint : /api/v1/grocery-list/item/update (PUT)

  Request

```
{
    "id": 1,
    "name":"Milk Updated",
    "note":"Farmer",
    "quantity":4,
    "groceryListName":"Testing List"
}
```

  Response

```
{
    "id": 1,
    "name": "Milk Updated",
    "quantity": 4,
    "note": "Farmer",
    "added_on": "2024-08-02T10:59:59",
    "last_modified_on": "2024-08-04T21:45:18.0864085",
    "purchased": false,
    "added_by": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "last_modified_by": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
    },
    "groceryListName": null
}
```

- Get All Grocery Items of List

  Endpoint : /api/v1/grocery-list/item/all/1 (GET)

```
[
  {
    "id": 1,
    "name": "Milk Updated",
    "quantity": 4,
    "note": "Farmer",
    "added_on": "2024-08-02T10:59:59",
    "last_modified_on": "2024-08-04T21:45:18",
    "purchased": false,
    "added_by": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "last_modified_by": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
    },
    "groceryListName": "Walmart"
  },
  {
    "id": 2,
    "name": "Eggs",
    "quantity": 2,
    "note": "dozen sized",
    "added_on": "2024-08-02T11:00:15",
    "last_modified_on": "2024-08-02T11:00:15",
    "purchased": false,
    "added_by": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "last_modified_by": {
      "id": 1,
      "firstName": "Kenny",
      "lastName": "Patel",
      "email": "kenee@gmail.com"
    },
    "groceryListName": "Walmart"
  }
]
```

- Update status

  Endpoint : /api/v1/grocery-list/item/update/purchaesd (PUT)

  Request

  ```
  {
     "id": 1,
     "name":"Milk Updated",
     "note":"Farmer",
     "quantity":4,
     "groceryListName":"Testing List"
  }
  ```

  Response

  “true”

Tasks

- Add task
  Endpoint : /task/add (POST)

Request

```
{
  "name": "Clean Out Garbage",
  "taskDate": "2024-08-05",
  "assignedTO": "Jems Patel",
  "description": "Already overflowing!!"
}
```

Response

```
{
  "id": 3,
  "room": {
    "id": 1,
    "name": "Ogilvie",
    "members": 4,
    "code": 56172,
    "active": true
  },
  "name": "Clean Out Garbage",
  "date_of_creation": "2024-08-04T22:00:07.940411",
  "last_modified_On": "2024-08-04T22:00:07.940411",
  "taskDate": "2024-08-05",
  "created_by": {
    "id": 9,
    "firstName": "Jay",
    "lastName": "Patel",
    "email": "pateljaysanjaybhai8@gmail.com"
  },
  "last_modified_by": {
    "id": 9,
    "firstName": "Jay",
    "lastName": "Patel",
    "email": "pateljaysanjaybhai8@gmail.com"
  },
  "description": "Already overflowing!!",
  "assignedTo": {
    "id": 4,
    "firstName": "Jems",
    "lastName": "Patel",
    "email": "jems@gmail.com"
  },
  "finished": false
```

}

- Get task
  Endpoint : /task/get?id=3 (GET)

  Request

```
{
   "id": 3,
   "room": {
      "id": 1,
      "name": "Ogilvie",
      "members": 4,
      "code": 56172,
      "active": true
   },
   "name": "Clean Out Garbage",
   "date_of_creation": "2024-08-04T22:00:07",
   "last_modified_On": "2024-08-04T22:00:07",
   "taskDate": "2024-08-05",
   "created_by": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
   },
   "last_modified_by": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
   },
   "description": "Already overflowing!!",
   "assignedTo": {
      "id": 4,
      "firstName": "Jems",
      "lastName": "Patel",
      "email": "jems@gmail.com"
   },
   "finished": false
}
```

- Add task
  Endpoint : /task/update (PUT)

Request

```
{
   "name": "Clean Out Garbage",
   "taskDate": "2024-08-07",
   "assignedTO": "User Example",
   "description": "Already overflowing Update!!"
}
```

Response

```
{
   "id": 3,
   "room": {
      "id": 1,
      "name": "Ogilvie",
      "members": 4,
      "code": 56172,
      "active": true
   },
   "name": "Clean Out Garbage",
   "date_of_creation": "2024-08-04T22:00:07",
   "last_modified_On": "2024-08-04T22:02:41.3288204",
   "taskDate": "2024-08-07",
   "created_by": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
   },
   "last_modified_by": {
      "id": 9,
      "firstName": "Jay",
      "lastName": "Patel",
      "email": "pateljaysanjaybhai8@gmail.com"
   },
   "description": "Already overflowing Update!!",
   "assignedTo": {
      "id": 10,
      "firstName": "User",
      "lastName": "Example",
      "email": "user@example.com"
   },
   "finished": false
```

}

- Update status of task
  Endpoint : /task/update/finished?id=3 (PUT)

  Response

  “true”

- Get all tasks
  Endpoint : /task/get/all (GET)

  Response

```
[
   {
      "id": 1,
      "room": {
         "id": 1,
         "name": "Ogilvie",
         "members": 4,
         "code": 56172,
         "active": true
      },
      "name": "Wash dishes",
      "date_of_creation": "2024-08-02T11:13:55",
      "last_modified_On": "2024-08-02T11:13:55",
      "taskDate": "2024-08-02",
      "created_by": {
         "id": 3,
         "firstName": "Jay",
         "lastName": "Patel",
         "email": "jay@gmail.com"
      },
      "last_modified_by": {
         "id": 3,
         "firstName": "Jay",
         "lastName": "Patel",
         "email": "jay@gmail.com"
      },
      "description": "Please clean all dishes .",
      "assignedTo": {
         "id": 1,
         "firstName": "Kenny",
        "lastName": "Patel",
        "email": "kenee@gmail.com"
      },
      "finished": false
    },
    {
      "id": 3,
      "room": {
        "id": 1,
        "name": "Ogilvie",
        "members": 4,
        "code": 56172,
        "active": true
      },
      "name": "Clean Out Garbage",
      "date_of_creation": "2024-08-04T22:00:07",
      "last_modified_On": "2024-08-04T22:03:54",
      "taskDate": "2024-08-07",
      "created_by": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      },
      "last_modified_by": {
        "id": 9,
        "firstName": "Jay",
        "lastName": "Patel",
        "email": "pateljaysanjaybhai8@gmail.com"
      },
      "description": "Already overflowing Update!!",
      "assignedTo": {
        "id": 10,
        "firstName": "User",
        "lastName": "Example",
        "email": "user@example.com"
      },
      "finished": true
    }
  ]
```

- Delete task
  Endpoint : /task/delete?id=1 (DELETE)

Response

“true”

Announcements

- Get all announcements
  Endpoint : /api/v1/announcements/get/all (GET)

  Response

```
[
  {
    "id": 1,
    "title": "Monthly Rent Due",
    "description": "Just a reminder that the monthly rent is due by the 5th of each
month. Please ensure that your payment is made on time to avoid any late fees.",
    "userPostedFirstName": "Kenny",
    "userPostedLastName": "Patel",
    "userModifiedFirstName": null,
    "userModifiedLastName": null,
    "postedDateTime": "2024-08-02T10:57:13",
    "lastUpdatedDateTime": null
  },
  {
    "id": 2,
    "title": "New Roommate Welcome Party",
    "description": "Join us in welcoming our new roommate, Sarah, with a small get-
together this Friday at 7 PM in the living room. Snacks and drinks will be provided!",
    "userPostedFirstName": "Kenny",
    "userPostedLastName": "Patel",
    "userModifiedFirstName": null,
    "userModifiedLastName": null,
    "postedDateTime": "2024-08-02T10:57:39",
    "lastUpdatedDateTime": null
  },
  {
    "id": 3,
    "title": "WiFi Maintenance",
        "description": "The internet service provider will be performing maintenance on our WiFi network this Wednesday from 2 PM to 4 PM. There may be intermittent outages during this time.",
        "userPostedFirstName": "Kenny",
        "userPostedLastName": "Patel",
        "userModifiedFirstName": null,
        "userModifiedLastName": null,
        "postedDateTime": "2024-08-02T10:58:06",
        "lastUpdatedDateTime": null
    },
    {
        "id": 4,
        "title": "Utilities Bill Share",
        "description": "The utilities bill for this month has arrived. Please check your share in the finance section and make sure to pay your part by the end of the week.",
        "userPostedFirstName": "Kenny",
        "userPostedLastName": "Patel",
        "userModifiedFirstName": null,
        "userModifiedLastName": null,
        "postedDateTime": "2024-08-02T10:58:32",
        "lastUpdatedDateTime": null
    }
]
```

- Add task

  Endpoint : api/v1/announcements/add (POST)

  Request

```
{
    "title":"Add annoucenemnt",
    "description":"Add annoucement"
}
```

  Response

```
{
    "id": 6,
    "title": "Add annoucenemnt",
    "description": "Add annoucement",
    "userPostedFirstName": "Jay",
    "userPostedLastName": "Patel",
    "userModifiedFirstName": null,
    "userModifiedLastName": null,
    "postedDateTime": "2024-08-04T23:17:40.2373428",
    "lastUpdatedDateTime": null
```

}

- Get announcement
  Endpoint : /api/v1/announcements/get/ 3 (GET)

  Response

```
{
   "id": 3,
   "title": "WiFi Maintenance",
   "description": "The internet service provider will be performing maintenance on our
WiFi network this Wednesday from 2 PM to 4 PM. There may be intermittent outages
during this time.",
   "userPostedFirstName": "Kenny",
   "userPostedLastName": "Patel",
   "userModifiedFirstName": null,
   "userModifiedLastName": null,
   "postedDateTime": "2024-08-02T10:58:06",
   "lastUpdatedDateTime": null
}
```

- Update announcement
  Endpoint : /api/v1/announcements/update/3 (PUT)

  Request

```
{
   "title":"New annoucement",
   "description":"Add annoucement"
}
```

  Response

```
{
   "id": 3,
   "title": "New annoucement",
   "description": "Add annoucement",
   "userPostedFirstName": "Kenny",
   "userPostedLastName": "Patel",
   "userModifiedFirstName": "Jay",
   "userModifiedLastName": "Patel",
   "postedDateTime": "2024-08-02T10:58:06",
   "lastUpdatedDateTime": "2024-08-04T23:21:17.9112231"
}
```

- Delete announcement
  Endpoint : /api/v1/announcements/delete/2 (DELETE)

  Response

  “true”

Profile

- Update profile
  Endpoint : /api/v1/profile/endpoint (POST)

  Request

  ```
  {
     "email": "pateljaysanjaybhai8@gmail.com",
     "firstName":"Jay_Sanjaybhai_Update",
     "lastName":"Patel_Update"
  }
  ```

  Response

  ```
  {
     "id": 9,
     "firstName": "Jay_Sanjaybhai_Update",
     "lastName": "Patel_Update",
     "email": "pateljaysanjaybhai8@gmail.com"
  }
  ```
- Get Profile
  Endpoint : /api/v1/profile/get (GET)

  Response

  ```
  {
     "userDTOResponse": {
        "id": 9,
        "firstName": "Jay_Sanjaybhai_Update",
        "lastName": "Patel_Update",
        "email": "pateljaysanjaybhai8@gmail.com"
     },
     "roomDTOResponse": {
        "id": 1,
        "name": "Ogilvie",
  ```

```
        "members": 4,
        "code": 56172,
        "active": true
    },
    "joinDate": "2024-08-04T20:37:10"
}
```

Room Setup

- Leave Room
  Endpoint : /api/v1/room/leave (GET)

  Response

  “true”

- Create Room
  Endpoint : /api/v1/room/create (POST)

  Request

  ```
  {
   "name": "Dalhousie"
  }
  ```

  Response

  ```
  {
     "id": 5,
     "name": "Dalhousie",
     "members": 1,
     "code": 90675,
     "active": true
  }
  ```
- Update Room
  Endpoint : /api/v1/room/update (PUT)

  Response
  ```
  {
     "id": 5,
     "name": "Dalhousie Update"
  ```

```
}
```

Request

```
{
   "id": 5,
   "name": "Dalhousie Update",
   "members": 1,
   "code": 90675,
   "active": true
}
```

- Delete Room
  Endpoint : /api/v1/room/delete/5 (GET)

  Response

  “true”
- Join Room
  Endpoint : /api/v1/room/join (POST)

  Request

```
{
   "joinCode": 56172
}
```

Response

```
{
   "userDTOResponse": {
      "id": 9,
      "firstName": "Jay_Sanjaybhai_Update",
      "lastName": "Patel_Update",
      "email": "pateljaysanjaybhai8@gmail.com"
   },
   "join_date": "2024-08-04T22:15:29.2355426",
   "roomDTOResponse": {
      "id": 1,
      "name": "Ogilvie",
      "members": 4,
        "code": 56172,
        "active": true
    }
}
```

- Get all Users in Room
  Endpoint : /api/v1/room/get/all/users (GET)

  Response

```
[
    {
        "id": 1,
        "firstName": "Kenny",
        "lastName": "Patel",
        "email": "kenee@gmail.com"
    },
    {
        "id": 9,
        "firstName": "Jay_Sanjaybhai_Update",
        "lastName": "Patel_Update",
        "email": "pateljaysanjaybhai8@gmail.com"
    }
]
```

- Get Room by ID
  Endpoint : /api/v1/room/get/3 (GET)

  Response

```
{
    "id": 3,
    "name": "South Park",
    "members": 2,
    "code": 80956,
    "active": true
}
```

Security Design

This section outlines the security measures implemented in the application. It includes sections about User Authentication and Access Control, Data Encryption, and Protection and Compliance Measures.

User Authentication and Access Control

User Identification: The users of the application will be uniquely identified through their emails. Every service that requires the user's details will be found out using their email address. There are also room level details which can be found out by fetching the email address of the user and then finding out the room details from there.
Authentication Process
Sign Up: The user can sign up to the platform using their first name, last name, email address, and password.
Login: The user can log in to their accounts using their email address and password.
Access Control: Only specific parts of the application will be exposed to users who are not signed in. This configuration is provided in the file WebConfig in the backend code. The text files include_endpoints and exclude_endpoints have the endpoints which are excluded and included for protection.

Data Encryption and Protection

The following encryption and protection techniques will be used to secure sensitive data:

Data Encryption

Sensitive data such as password of the user will be first encrypted and then stored in the database. Protecting sensitive information, including user passwords, against illegal access and possible breaches requires the usage of encryption. Passwords that are encrypted stay safe and unreadable to unauthorized parties, even if the database is compromised.
The flow of how the password will be encrypted:

- Step 1: During registration, the user provides their plaintext password along with their first name, last name, username, and email.
- Step 2: A unique salt is generated for each user. The salt is made to ensure uniqueness of the hashed password.
- Step 3: A cryptographic hash function like BCrypt is used to process the plaintext password in conjunction with the salt. This function creates a hashed password value.
- Step 4: The hashed password is then stored in the database.

JWT Authentication

To guarantee safe application access and efficiently handle user sessions, token-based authentication is used. JSON Web Tokens (JWT) are utilized in this system for user authentication and access control management.

- Token-Based Authentication: Token-based authentication will be implemented by the system using JWT. The client will store and include the JWT that the server issues after a successful login in the Authorization header of any subsequent requests.

- Data in the Token: The token consists of information about the room of the suer and the email address of the user. The room details include name , members etc.
- Token Expiry: The JWT token will have a defined expiration time to enhance security. In this system, the token will expire 24 hours after issuance. This ensures that even if a token is compromised, it will only be valid for a limited period, reducing the risk of long-term misuse.
- Token Storage: The JWT token will be securely stored in the local storage of the user's browser. Local storage is a preferable option as it persists across page reloads and browser tabs, providing a seamless user experience.

A specialized user service is in charge of managing user credentials and loading user-specific information throughout the authentication procedure. To increase security, user credentials are encrypted using the BCrypt hashing method. The BCrypt hashing technique, which is robust and flexible, protects against brute-force attacks by using a salt.

Route Protection
The system protects its API routes and endpoints by validating the JWT on each request. Middleware is used to check the presence and validity of the token before granting access to protected resources. If the token is missing, invalid, or expired, the request is denied, and an appropriate error response is returned. This ensures that only authenticated users with valid tokens can access many parts of the system.

Compliance Measures
To ensure that the system adheres to relevant regulations and standards, the following compliance measures will be implemented:

- Regulatory Requirements: The system is designed to comply with applicable regulations such as the General Data Protection Regulation (GDPR) for data protection and privacy. Only required data is collected from the user and stored in the database. The data is also encrypted and stored safely.
- Compliance Strategy: An all-encompassing method for preserving security and compliance with regulations is part of a compliance strategy. This includes carrying out routine audits and security evaluations to pinpoint any potential weaknesses.

Performance Considerations

Scalability

The system's capacity to accommodate growing demands by increasing resources is referred to as scalability. When a system is effectively scalable, it can handle increasing user and data volumes without seeing a decrease in its overall performance.

Horizontal Scaling

Decoupled Architecture:

- Different functionalities of the application are managed by distinct services or containers thanks to its decoupled architecture. Because of this modular design, each component can be scaled separately.
- More containers for particular services can be added as demand grows without affecting other areas of the application. Because of its flexibility, focused scaling of components according to their individual loads is made possible, which promotes effective horizontal scaling.

Service-Based Design:

- The program is designed using a service-oriented architecture, in which different features are divided up into distinct services. This strategy complements Docker's ability to handle numerous, separate containers.
- Scalable deployment is made possible by service-based design, as additional containers enable the autonomous scaling of individual services. This division facilitates the management of various workloads and maximizes the use of resources.

Stateless Design:

- A stateless design means there is no data stored in the server. In our application we follow a stateless design. This design ensures that any instance of the application can handle any request without needing to have knowledge of other interactions, making it easier to distribute requests across multiple instances.
- Client-Side Storage: For managing user sessions, our application relies on client-side storage mechanisms such as local storage. User authentication tokens, such as JWT (JSON Web Tokens), are stored in these client-side locations. When a request is made, the token is sent with the request, and the server validates it to authenticate the user.
- JWT (JSON Web Tokens): JWTs are used for authentication and are sent with every request. These tokens are validated with each request and contain user data. The server does not need to keep track of a session state in between requests because the token has all the user's information that is required.

Token Validation:

- On each request, the application extracts the token, validates it, and retrieves user information from it. This process ensures that the server does not need to keep session data in memory.

Load Handling

The techniques and mechanisms used in load handling make that the application runs smoothly even during periods of high demand. Although we have prioritized the essential functions in our current implementation, we have built the system with principles that will allow for future improvements in load handling.

- The application adheres to good coding practices, ensuring that the codebase is clean, well-structured, and efficient. This includes optimizing algorithms as well easy to refactoring code.
- For both the front end and back end, we generate and bundle application images using Docker. Containerization makes deployment easier and guarantees that our application operates reliably in various environments. Docker containers facilitate horizontal scaling by making it simple to deploy and manage many instances.

Fault Tolerance and Recovery Measures

Fault Tolerance and Recovery Measures are really important for any application since they help applications withstand failures.

In our application, we utilize Docker containerization to manage application versions and ensure fault tolerance. This approach allows us to quickly revert to a stable state in case of errors or issues with new deployments.

Below are some points which the application follows that ensure fault tolerance and recovery:

- Versioned Docker Images: After implementation and testing, every feature is integrated into a Docker image. Because these images are versioned, we can keep track of previous stable and unstable iterations.
- Rollback to Previous Versions: If a newly tested Docker image reveals faults, we may immediately fall back to the previous stable image. In doing so, downtime is reduced and continuous service availability is guaranteed.
- Testing: Before releasing any feature we test it thoroughly so that we can catch errors or exceptions beforehand.

Technology Stack

This section details the technologies used to build and operate the roomM8 application. The chosen stack ensures a robust, scalable, and maintainable system.

Programming Languages

Java:
Usage: Backend development.
Rationale: Java is a powerful, widely-used language known for its performance, scalability, and extensive ecosystem. It's well-suited for building robust server-side applications and integrates seamlessly with Spring Boot.
TypeScript:
Usage: Frontend development.
Rationale: TypeScript is a superset of JavaScript that adds static typing, making code easier to understand and maintain. It helps catch errors at compile-time, improving code quality and reliability.

Frameworks and Libraries

Spring Boot:
Usage: Backend framework.
Rationale: Spring Boot simplifies the development of production-ready applications. It provides a comprehensive set of features for building web applications, including dependency injection, security, and data access.
Next.js:
Usage: Frontend framework.
Rationale: Next.js is a popular React framework that offers server-side rendering, static site generation, and API routes out of the box. It enhances performance and SEO for the frontend application.
Tailwind CSS:
Usage: CSS framework.
Rationale: Tailwind CSS is a utility-first CSS framework that allows developers to style their applications quickly and consistently. It promotes a modular approach to styling, reducing the need for custom CSS.
Shadcn/UI:
Usage: UI components library.
Rationale: Shadcn/UI provides a set of reusable UI components that integrate seamlessly with Tailwind CSS. It helps maintain a consistent look and feel across the application and speeds up the development process.

Tools and Environments

MySQL:

- Usage: Database management.

- Rationale: MySQL is a reliable, widely-used relational database management system. It supports ACID transactions, ensuring data integrity and consistency, and is well-suited for handling structured data in roomM8.

JWT (JSON Web Tokens):

- Usage: Authentication and session management.
- Rationale: JWTs are a compact, secure way to represent user identity and session information. They are stateless and can be easily transmitted between the client and server, simplifying authentication processes.

Docker:

- Usage: Containerization.
- Rationale: Docker enables the creation of lightweight, portable containers that package the application and its dependencies. This ensures consistent environments across development, testing, and production, reducing deployment issues.

Postman:

- Usage: API testing.
- Rationale: Postman is a powerful tool for developing and testing APIs. It allows developers to create, test, and document API endpoints, ensuring they work as expected.

JUnit:

- Usage: Unit testing.
- Rationale: JUnit is a widely-used testing framework for Java. It facilitates the creation and execution of automated tests, ensuring that individual components of the backend are working correctly.

VSCode:

- Usage: Integrated Development Environment (IDE).
- Rationale: Visual Studio Code (VSCode) is a versatile, lightweight IDE with extensive plugin support. It enhances productivity with features like syntax highlighting, code completion, and integrated terminal.

This technology stack ensures that roomM8 is built with reliable, scalable, and maintainable technologies. Each component has been chosen for its strengths and compatibility with the overall system, ensuring a cohesive and efficient development process.

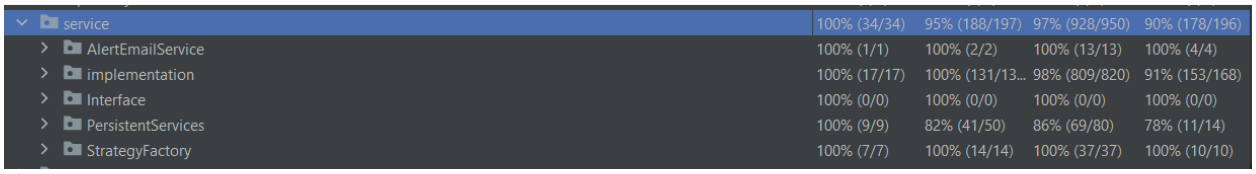
Testing Strategy

A comprehensive testing strategy is essential to ensure the roomM8 application functions correctly, meets user requirements, and maintains high quality. This section outlines the various testing methodologies that will be employed.

Unit Testing

Unit Testing focuses on testing individual components or units of the application to ensure they function as expected.

- JUnit: We have used JUnit for testing Java components. Each class and method has corresponding test cases to verify functionality, edge cases, and error handling. JUnit is our primary testing framework for writing and executing unit tests in Java. For all the business logic written test cases to handle normal and edge conditions.



service	100% (34/34)	95% (188/197)	97% (928/950)	90% (178/196)
AlertEmailService	100% (1/1)	100% (2/2)	100% (13/13)	100% (4/4)
implementation	100% (17/17)	100% (131/13...	98% (809/820)	91% (153/168)
Interface	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
PersistentServices	100% (9/9)	82% (41/50)	86% (69/80)	78% (11/14)
StrategyFactory	100% (7/7)	100% (14/14)	100% (37/37)	100% (10/10)

Figure 7 Code Coverage

- Mockito: A well-liked Java framework for building mock objects for unit testing is called Mockito. It makes it simpler to test components separately by enabling developers to isolate the unit under test and mimic the behavior of complicated dependencies.

In our project, Mockito was utilized extensively to facilitate unit testing by creating mock objects and defining their behaviors. As a result, we were able to successfully isolate and test the application's different components, making sure that each one operates as intended when separated from outside dependencies.

Integration Testing

Integration Testing: Integration testing ensures that different modules or services within the application work together as expected.

Backend:

- Spring Boot Test: Use Spring Boot's testing support to create integration tests. This involves setting up the application context and testing the interactions between different layers (e.g., controllers, services, and repositories).
- Testcontainers: Employ Testcontainers to spin up temporary instances of MySQL and other services for integration testing. This ensures tests run in an environment that closely resembles production.

System Testing

System Testing involves testing the complete, integrated application to verify it meets the specified requirements.

- **Manual Testing:** Conducting manual testing to identify issues that automated tests might miss. This includes exploratory testing, usability testing, and testing on different devices and browsers.

User Acceptance Testing (UAT)

User Acceptance Testing (UAT) ensures the application meets the needs and expectations of its users.

- Git Flow: We can implement the Git Flow methodology for User Acceptance Testing. Group members worked on personal branches and feature branches. The development branch was used for CI/CD runs to ensure all tests pass. The main branch was used for final testing and deployment.
- Beta Testing: We can release a beta version of roomM8 to a select group of users. gather feedback on functionality, usability, and performance and address any issues or feature requests identified during this phase.

Continuous Integration and Continuous Deployment (CI/CD)

CI/CD practices are essential to maintain code quality and ensure smooth deployments.

- GitLab CI/CD: The application usesGitLab for the CI/CD pipeline. The development branch runs automated tests to ensure code quality. Successful tests lead to deployments from the main branch.
- Docker: The application uses Docker to containerize the application. This ensures consistency across development, testing, and production environments.
- University VM: The application is deployed on a VM provided by the university. This setup ensures that the environment is stable and controlled.

Logging

Logging is crucial for identifying and diagnosing issues in both development and production environments.

- SLF4J and Log4j: The application uses SLF4J as the logging facade and Log4j as the logging implementation. This combination provides a flexible and powerful logging system for the backend.

This testing strategy ensures that roomM8 is thoroughly tested at every stage of development, from individual components to the complete system. By combining automated and manual testing, we can ensure the application meets its requirements and provides a high-quality user experience.

Conclusion and Future Work

Summary

We have included a thorough description of the technological architecture and execution plan for our web application in this document. System architecture, data design, interface design, security precautions, performance concerns, and adherence to industry standards were among the crucial topics we addressed.

Our application is built on a robust and scalable architecture that accommodates current needs and anticipates future growth. We have implemented security measures, including data encryption to securely store user's data.Performance concerns have been taken into account using a design that facilitates horizontal scaling, which enables the system to effectively manage growing demand. As part of our fault tolerance strategy, we deploy our applications using Docker, which enables quick rollbacks to earlier iterations as needed, guaranteeing system dependability and resilience.

The application's business logic has been tested using JUnit and Mockito.Unit tests as well as other test plans. The technology stack has also been discussed in detail but in summary for frontend Next.js is used and for backend Spring boot is used.

To summarize , this document covers important parts of the application ann can server as a guide for someone who wants to learn more about the application.

Future Improvements or Enhancements

Currently, our project includes essential features developed within a constrained timeline. However, we have outlined several enhancements to further elevate the application's functionality and user experience.

1. **Shared Docs & Photos Feature:** Our foremost enhancement will be the introduction of the "Shared Docs & Photos" feature. This addition will enable roommates to effortlessly access and manage shared documents and images within their living space. This feature aims to streamline the management of communal resources, fostering better organization and collaboration among users.
2. **Enhanced Expense Customization and Analysis:** We plan to expand the expense management functionality by incorporating advanced customization options. Users will be able to tailor their expense tracking according to individual preferences and requirements. Furthermore, we will introduce comprehensive analytical tools that provide visual insights into users' spending patterns. By leveraging graphs and visualizations, we will enable users to gain deeper insights into their financial habits, facilitating more informed budgeting and expense planning.
3. **Role-Based Authentication:** Finally, we aim to implement role-based authentication to enhance access control within the application. This feature will enable differentiated access levels, allowing us to assign specific roles to users, such as administrators, editors, or users. Each role will have defined permissions to add, edit, update, or delete items across various application features. By implementing role-based access control, we will improve security and ensure that users can only interact with data and features relevant

to their role, thus minimizing the risk of unauthorized changes and enhancing overall application integrity.

These future improvements are designed to enhance user engagement, provide more tailored functionality, and deliver actionable insights. As we continue to evolve the application, we remain committed to incorporating user feedback and staying aligned with emerging trends to ensure our platform remains relevant and valuable.

References

[1] H. Chauhan, "Master Full-Stack Monorepos: A Step-by-Step Guide," 3 Aug. 2024. [Online]. Available: https://dev.to/hardikidea/master-full-stack-monorepos-a-step-by-step-guide-2196. [Accessed 4 Aug. 2024].

[2] Lucidchart, "How to draw 5 types of architectural diagrams," [Online]. Available: https://www.lucidchart.com/blog/how-to-draw-architectural-diagrams. [Accessed 2 Aug. 2024].

[3] gliffy, "How to Draw Entity Relationship Diagrams (ERDs)," 2 Oct. 2020. [Online]. Available: https://www.gliffy.com/blog/how-to-draw-an-entity-relationship-diagram. [Accessed 2 Aug. 2024].

[4] Career Explorer, "What is an App Developer?," [Online]. Available: https://www.careerexplorer.com/careers/app-developer/. [Accessed 2 Aug. 2024].

[5] Intuit mailchimp, "End User," [Online]. Available: https://mailchimp.com/marketing-glossary/end-user/. [Accessed 2 Aug. 2024].

[6] spring, "Spring Boot," [Online]. Available: https://spring.io/projects/spring-boot. [Accessed 2 Aug. 2024].

[7] NEXT.js, "The React Framework for the Web," [Online]. Available: https://nextjs.org/. [Accessed 2 Aug. 2024].

[8] shadcn/ui, "Build your component library," [Online]. Available: https://ui.shadcn.com/. [Accessed 2 Aug. 2024].

[9] Auth0, "JWT," [Online]. Available: https://jwt.io/. [Accessed 2 Aug. 2024].