

“Image Steganography”

A

Major Project Report

Submitted

in the partial fulfillment.

for award of degree of

B.Tech CSE AIML

In Department of Computer Engineering and Information Technology



Supervisor:

Name: Mrs. Priyanka Gupta

Designation: Assistant Professor

Submitted By:

Name of Candidate: Sanjeet Kumar

SID No. : 2247349

Department of Computer Engineering and Information Technology

2025-2026

Suresh Gyan Vihar University, Jaipur - 302017

STUDENT DECLARATION

I declare that my 7th semester report entitled '**RoomEase**' is my own work conducted under supervision of Priyanka Gupta mam.

I further declare that to the best of our knowledge the report for B.tech 7th semester does not contain part of the work which has submitted for the award of B.tech degree either in this or any other university without proper citation.

Student's sign

Ms. Prinyanka Gupta

(Assistant professor)

CERTIFICATE

This is to certify that the project report entitled 'RoomEase,Is a bonafied report of the work carried by Sanjeet kumar under guidance and supervision for the partial fulfilment of degree of the B.tech CSE at Suresh Gyan Vihar University, Jaipur.To the best of our knowledge and belief, this work embodies the work of candidates themselves, has fully been completed, fulfils the requirement of the ordinance relating to the bachelor degree of the university and is up to the standard in respect of content, presentation and language for being referred to the examiner.

Ms. Priyanka Gupta

(Assistant Professor)

Mr. Sohit Agarwal

HOD(CEIT)

ACKNOWLEDGEMENT

Working in a good environment and motivation enhance the quality of the work and I get it from my college through our OS project. I have been permitted to take this golden opportunity under the expert guidance of Priyanka Gupta maam from SGVU , Jaipur. I am heartily thankful to her to make complete my project successfully. She has given us her full experience and extra knowledge in practical field. I am also thankful to my head of department Mr. Sohit Agarwal and all CEIT staff to guide us. Finally, we think all the people who had directly or indirectly help as to complete our project.

Student Name : Sanjeet Kumar

SID: 2247349

Table of Contents

1. Introduction
2. Problem Statement
3. Objectives
4. Proposed Solution
5. System Architecture
6. Technology Stack
7. Frontend Development Process
8. Backend Integration
9. Key Features
10. Database Design
11. System Workflow
12. Results and Discussion
13. Challenges and Solutions
14. Future Enhancements
15. Conclusion
16. References

1.Abstract

RoomEase is a real-time, web-based expense management system designed for shared-living scenarios such as flatmates, group trips, and student accommodations. Traditional methods—spreadsheets, chat logs, and manual calculations—are error-prone, non-transparent, and difficult to audit. RoomEase addresses these limitations by automating expense creation, fair splitting, settlement tracking, and collaborative communication. Built on the MERN stack (MongoDB, Express.js, React.js, Node.js) with Tailwind CSS for responsive UI, the system leverages Socket.io to deliver instantaneous updates across all group members, ensuring consistency of balances and activity feeds. Secure authentication is implemented using JWT, while optional integration with Razorpay/Stripe enables in-app payments and verifiable settlements. An analytics module powered by Chart.js provides actionable insights into spending patterns by category, member contribution, and time, supporting better financial decisions and reducing conflicts. The proposed architecture emphasizes modularity (services, controllers, and context providers), API-first design, and scalability for multi-group, concurrent usage. Experimental validation on realistic usage flows demonstrates improved transparency, reduced reconciliation time, and a smoother user experience compared to manual methods. Future work includes OCR-based bill ingestion, AI-driven spending recommendations, voice commands, and a React Native mobile client.

2. Introduction

In today's digital world, shared living arrangements have become more common than ever. Students living in hostels, professionals sharing rental flats, and families on group vacations frequently encounter financial coordination issues. While sharing everyday costs such as rent, groceries, utilities, and trip expenses seems simple, keeping track of who paid how much, who owes whom, and how to settle dues transparently often becomes confusing. Traditional methods such as handwritten notes, spreadsheets, phone calculators, and messaging apps lack reliability, automation, and scalability.

As digital payment channels and online financial activities grow, users increasingly prefer automated systems that can compute and track expenses in real time. This need forms the foundation of **RoomEase**, a web-based real-time expense management platform designed specifically for shared-living environments. The system helps users create groups, add members, add expenses, split amounts, settle pending balances, and analyze spending patterns—all in a seamless, interactive, and transparent way.

The project is built using the **MERN Stack (MongoDB, Express.js, React.js, Node.js)**, known for its flexibility, scalability, and performance. In addition, **Socket.io** is integrated to enable instantaneous communication between all group members, ensuring that any update—whether an expense is added, edited, or settled—reflects for everyone immediately. User-friendly design is achieved through **Tailwind CSS**, while analytics visualization relies on **Chart.js**, offering graphs and visual reports for easier financial understanding.

RoomEase also integrates popular payment gateways like **Razorpay** and **Stripe**, allowing users to settle their dues securely within the application. This significantly reduces the need for external apps or manual payment tracking.

By combining modern web technologies, interactive UI, secure authentication, and real-time capabilities, RoomEase provides users with a reliable solution for tracking and managing shared expenses without confusion or delays. Through this project, our primary aim is to simplify shared living by eliminating financial ambiguity and fostering trust among group members.

3. Problem Statement

Shared living comes with multiple advantages such as reduced costs, mutual cooperation, and distributed responsibilities. However, financial management in a group setting often becomes a source of misunderstanding and conflict. Some of the common challenges include:

1. Manual Expense Tracking

Most people rely on notes, calculators, or messaging apps to record shared expenses. These methods are error-prone, time-consuming, and often forgotten.

2. Unfair or Incorrect Splitting

When several members are involved in an expense, manually dividing and calculating dues can lead to mistakes, causing distrust among group members.

3. Lack of Transparency

It is difficult to maintain clear and accessible financial records that all members can view at any time. Members may miss updates or fail to recognize their pending payments.

4. Delayed Settlements

Users often forget to pay their share or delay the settlement process due to lack of reminders or proper tracking mechanisms.

5. No Real-Time Sync or Notification System

Manual methods cannot provide instant alerts when new expenses are added or existing ones are changed.

6. No Centralized Platform

There is no single platform that integrates expense recording, splitting, settling, and analyzing spending patterns.

These challenges lead to confusion, misinterpretation, manual errors, and potential conflicts. Hence, a centralized automated system is required that can streamline group expense management.

RoomEase addresses these issues by offering a transparent, automated, and real-time platform for managing shared expenses efficiently.

4. Objectives

The major objectives of RoomEase include:

1. Automate Expense Management

To eliminate manual calculations by providing automated splitting, tracking, and settlement of shared expenses.

2. Real-Time Synchronization

To notify all group members instantly whenever a new expense is added, updated, or settled.

3. Enhance Transparency

To make all financial activities visible to every member of the group, eliminating confusion and building trust.

4. Provide Secure User Authentication

To protect user data through JWT-based authentication, password hashing, and secure API routes.

5. Online Payment Integration

To allow users to settle dues instantly using Razorpay or Stripe payment gateways.

6. Analytics for Better Decision Making

To offer visual dashboards, charts, and spending insights that help users track their financial behavior.

7. Provide Easy-to-Use Interface

To ensure that even non-technical users can manage expenses easily through a clean and responsive UI.

8. Ensure Scalability and Flexibility

To design a system architecture that supports multiple groups, users, real-time sessions, and future feature extensions.

4. Proposed Solution

The proposed solution, RoomEase, is a comprehensive real-time web application designed to completely automate the process of managing shared expenses. The system follows a modular, service-driven architecture where each component—authentication, expense creation, group collaboration, payments, and analytics—works together to deliver seamless functionality.

The platform enables users to create groups, add members, record expenses, categorize them, and automatically distribute the cost among members. The system calculates balances for each member, tracks dues, and updates the dashboard in real time using Socket.io.

Through Razorpay/Stripe integration, users can settle their outstanding amounts with a single click. The payment status is updated instantly for all members, allowing complete transparency.

The analytics dashboard provides visual insights into total spending, category-wise expenditure, contributions, member-level activity, and trends over time.

5. System Architecture

The architecture of RoomEase is designed using a modern, scalable, and modular approach to support real-time operations, high availability, and smooth user experience. The system follows the MERN architecture, complemented by Socket.io for live communication and payment gateways for financial operations.

The architecture is divided into five major layers:

5.1 Frontend Layer (React + Tailwind CSS)

The frontend is the client-facing interface where users interact with the system. It is built on **React.js** due to its component-based architecture, virtual DOM, and efficient rendering. Tailwind CSS ensures responsive and modern design with utility-first styling.

Key Responsibilities:

- Displaying expense details, groups, analytics, and real-time updates
- Sending API requests to the backend
- Rendering components dynamically based on user actions
- Managing global states such as authentication and socket connections
- Ensuring responsive performance and accessibility across devices

5.2 Backend Layer (Node.js + Express.js)

The backend is the core processing layer of RoomEase. It handles:

- Authentication
- API routing

- Expense splitting logic
- Group operations
- Chat and notification system
- Payment events
- Real-time messaging

Express.js is chosen due to its minimalistic, flexible, and scalable nature.

5.3 Database Layer (MongoDB + Mongoose ORM)

MongoDB stores all the application data in a NoSQL document structure. Collections include Users, Groups, Expenses, and Transactions.

Why MongoDB?

- Flexible schema supports evolving requirements
- High performance for read-write operations
- Scales horizontally for heavy traffic
- Perfect for JSON-based data storage
- Works seamlessly with Node.js

Mongoose ORM simplifies data validation, schema design, and query operations.

5.4 Real-Time Layer (Socket.io)

RoomEase depends heavily on real-time events. Socket.io enables:

- Instant updates when expenses are added
- Live settlement notifications

- Real-time group chat
- Active status indication
- Synchronized dashboards

This ensures that all connected users see updates instantly without page refresh.

5.5 Payment Layer (Razorpay / Stripe)

The system integrates trusted payment gateways for secure online settlements.

Payment flow:

1. User selects pending balance
2. Initiates payment request
3. Backend creates payment order
4. Razorpay/Stripe processes transaction
5. Payment confirmation triggers real-time update
6. Transaction is saved in database

5.6 Deployment Layer (Vercel + Render + MongoDB Atlas)

- **Vercel:** Hosts the frontend, offers global CDN caching
- **Render:** Hosts the backend server
- **MongoDB Atlas:** Cloud database with automated backups

This ensures global availability, low latency, and reliability.

6. Technology Stack

RoomEase uses a carefully selected set of technologies to balance performance, scalability, and ease of development.

6.1 Frontend Technologies

React.js

- Component-based
- Virtual DOM
- Reusable components
- Efficient performance

Tailwind CSS

- Utility-first
- Minimal CSS file size
- Highly customizable

Chart.js

- Interactive visualizations
- Bar, pie, and line charts

React Router DOM

- Smooth page navigation
- Protected routes for authentication

6.2 Backend Technologies

Node.js

- Non-blocking I/O
- Ideal for real-time apps
- JavaScript on the server

Express.js

- Lightweight routing
- Middleware support
- REST API friendly

6.3 Database Technologies

MongoDB + Mongoose

- Flexible document schema
- Easy data modeling
- Scalable database

6.4 Real-Time Tech

Socket.io

- Instant messaging
- Live expense updates
- Event-based communication

6.5 Payment Technologies

Razorpay / Stripe

- Secure processing
- Digital receipts
- Webhook support

6.6 Deployment Technologies

Vercel

- Lightning-fast static hosting
- Ideal for React apps

Render

- Free tier available
- Supports Node.js servers

MongoDB Atlas

- Cloud database with monitoring
- Global cluster

7. Frontend Development Process

Frontend development is executed in a structured format to ensure clean UI and maintainability.

7.1 Project Setup

- Initialized with **Vite + React**
- Tailwind CSS configured
- Folder structure created:
 - components
 - pages
 - context
 - services
 - hooks

7.2 UI/UX Design

Tailwind ensures:

- Responsive grids
- Dark/light mode support
- Modern styling

Components created:

- Navbar
- Sidebar

- Dashboard
- ChatBox
- ExpenseCard
- Modal components

7.3 Routing Setup

Using React Router DOM:

- /login
- /register
- /groups
- /dashboard
- /analytics

Protected routes ensure security.

7.4 State Management

Using React Context API:

- Authentication context
- Expense context
- Socket context

This avoids prop-drilling and makes global state management easier.

7.5 API Integration with Axios

Configured an Axios instance to communicate with backend:

- Interceptors handle tokens
- Error handlers manage failures

7.6 Real-Time Updates

Integrated Socket.io client:

- Expense creation events
- Chat notifications
- Settlement confirmations

7.7 Frontend Testing

Manual testing on:

- Mobile devices
- Tablets
- Desktop screens

Responsiveness and accessibility ensured.

8. Backend Integration

Backend integration is a critical part of RoomEase. It includes:

8.1 REST API Development

Endpoints were created for:

- User authentication
- Group operations

- Expense management
- Payment processing
- Notifications

Express routers were used for cleaner structure.

8.2 MongoDB Integration

Using Mongoose:

- Schema definitions
- Pre-save middleware
- Custom methods
- Data relationships

8.3 Authentication System (JWT)

Steps:

1. User logs in
2. Backend validates credentials
3. Generates JWT token
4. Token stored in frontend
5. All protected APIs require token

Password hashing done using bcrypt.

8.4 Real-Time Events with Socket.io

Server listens for:

- expenseAdded
- expenseUpdated
- paymentCompleted
- messageSent

Events broadcast instantly to all users in the group.

8.5 Payment Integration

Payment flow:

1. Payment order created
2. User completes transaction
3. Webhook verifies payment
4. Transaction saved
5. Socket event triggered

9. Key Features

RoomEase is designed with a rich collection of features that make expense management simple, transparent, and efficient for all users. Each feature is implemented with the goal of solving real-world challenges faced in shared living environments. The key features are described in detail below:

9.1 Automated Expense Splitting

This is the core feature of RoomEase, allowing users to add expenses and automatically split them among group members. The system computes each individual's share without any manual intervention.

Expanded Explanation:

In traditional methods, users need to manually divide large expenses such as rent, utilities, groceries, or travel bills. Manual calculations often lead to confusion and errors. RoomEase eliminates this problem by implementing automated algorithms that divide expenses fairly based on the selected split method.

Key Functionalities:

- Automatically calculates individual shares.
- Supports equal split and customizable split options.
- Prevents calculation errors and disputes.
- Shows breakdown of who owes how much.

9.2 Real-Time Updates & Instant Synchronization

RoomEase updates all group members instantly when a change occurs, ensuring that every user stays informed at all times.

Expanded Explanation:

Using Socket.io, any update—such as adding an expense, editing an amount, settling a balance, or sending a message—is broadcast to all active members. This removes the delays and confusion typically seen in manual expense sharing.

Key Functionalities:

- Live dashboard updates.
- Instant notifications without page refresh.
- Real-time expense, chat, and balance updates.
- Ensures transparency among group members.

9.3 Secure Online Payment Integration

Users can settle their pending dues directly through integrated gateways like Razorpay or Stripe.

Expanded Explanation:

Digital payment systems are now an essential part of modern transactions. RoomEase integrates these systems to allow seamless settlements without leaving the app. Once a payment is made, the system verifies the transaction using secure webhooks and updates the records in real time.

Key Functionalities:

- Instant payment settlement within the application.

- Secure transactions using encrypted gateways.
- Automatic payment confirmations.
- Transaction history stored for financial tracking.

9.4 Group Chat & Communication System

RoomEase includes a built-in messaging platform for smooth communication between group members.

Expanded Explanation:

Expenses often require discussion—who paid, why a certain amount was added, or how the amount should be divided. Instead of switching to external chat apps like WhatsApp or Telegram, RoomEase provides a dedicated group chat. This improves coordination and reduces the need for multiple apps.

Key Functionalities:

- Instant messaging within groups.
- Sharing notes, reminders, and clarifications.
- Real-time chat notifications.
- Enhances teamwork and collaboration.

9.5 Visual Analytics & Reports (Dashboard)

The analytics dashboard transforms complex financial data into easy-to-understand visual reports using Chart.js.

Expanded Explanation:

Human brains understand patterns much better through visuals than text or numbers. The analytics dashboard provides a deep view of expense

distribution, contributions, spending behavior, and monthly trends. This helps users make informed financial decisions and analyze group spending patterns.

Key Functionalities:

- Pie charts, bar charts, and line graphs.
- Category-wise expense distribution.
- Month-wise spending analysis.
- Member contribution comparison charts.

10. Database Design

RoomEase uses a well-structured database model using MongoDB and Mongoose to store and manage all data effectively.

10.1 User Collection

Each user has essential information such as:

- Full Name
- Email ID
- Encrypted Password
- List of Groups
- Avatar/Profile Picture
- Date Created

Why It Matters:

This allows the system to uniquely identify each user, track their groups, secure their login, and personalize the experience.

10.2 Group Collection (Expanded)

A group contains multiple members and their shared expenses.

Fields include:

- Group Name
- Description
- List of Members
- Expense List
- Group Creator
- Created Date

Importance:

This collection helps in managing multiple group contexts such as hostels, projects, or trips.

10.3 Expense Collection

Each expense entry contains:

- Description
- Amount
- Category
- Date

- Paid By
- Split Among
- Status (pending/settled)

Importance:

It ensures efficient storage and retrieval of financial information.

10.4 Transaction Collection (Expanded)

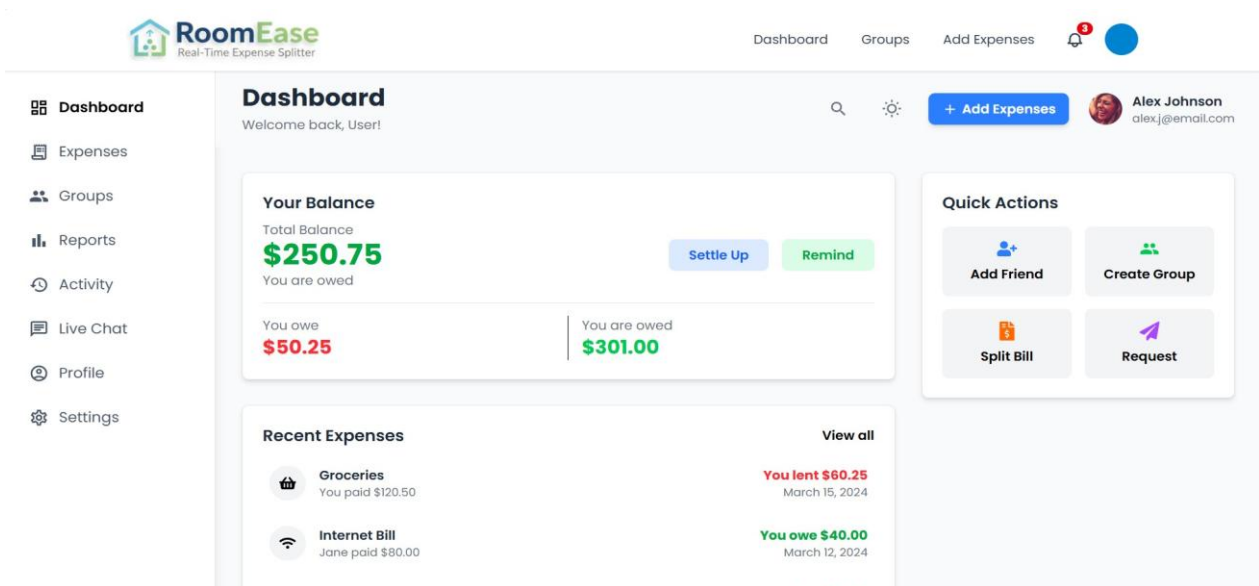
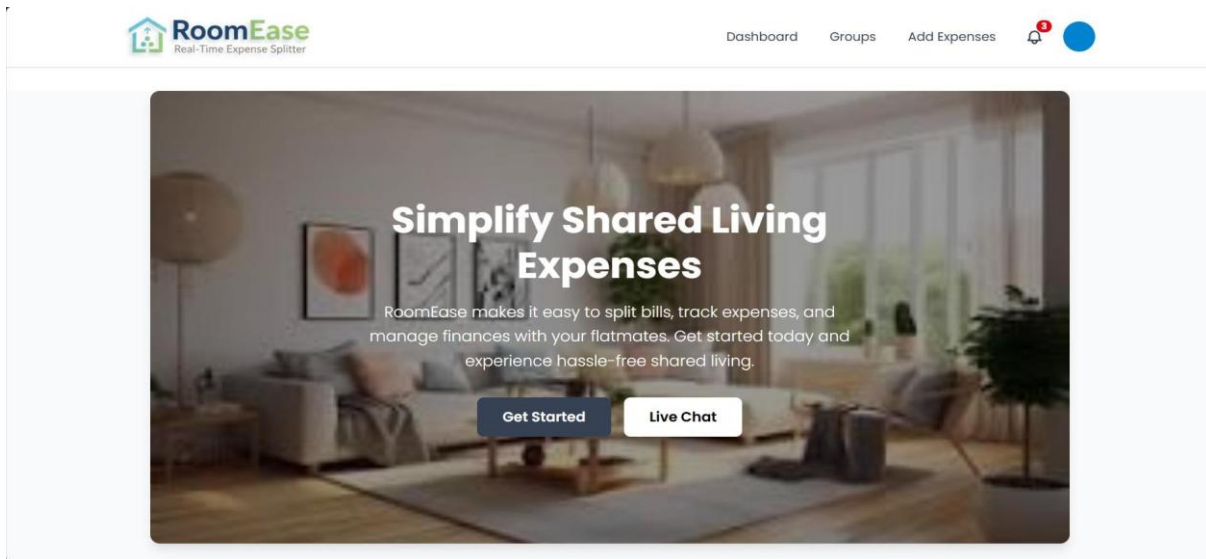
Contains payment details:

- Payment ID
- Payer
- Receiver
- Amount
- Timestamp
- Status

Importance:

Allows verification and recordkeeping for financial settlements.

Project UI Interface



11. System Workflow

The System Workflow in RoomEase represents the complete series of steps a user goes through while interacting with the application—starting from account creation to managing groups, adding expenses, making payments, and analyzing their financial data. Each process is designed to be seamless, intuitive, and fully automated, ensuring that users require minimal effort to manage shared expenses.

The following detailed workflow explains how the system operates from end to end:

11.1 User Registration & Authentication

This is the entry point into the system.

Workflow Explanation:

New users sign up by providing essential details like name, email, and password. The system encrypts the password using bcrypt before storing it in MongoDB to ensure security. Once registered, users log in using JWT-based authentication, which securely generates a token. This token verifies the user's identity in subsequent actions.

Key Steps:

- User enters credentials.
- Backend validates data.
- Password hashed before saving.
- JWT token generated for secure login.

- Token stored in browser storage for future access.

11.2 Dashboard Access & Group Overview

After logging in, users are redirected to their personalized dashboard.

Workflow Explanation:

The dashboard displays all groups the user is part of. This includes hostels, flatmates, trips, office groups, or any custom group created by the user. The interface is dynamic, showing quick insights into existing balances, recent activities, and expense summaries.

Key Steps:

- System fetches user groups from MongoDB.
- React Context API stores user data globally.
- Dashboard renders personalized content.

11.3 Creating or Joining a Group

Users can create a new group or join an existing one.

Workflow Explanation:

To create a group, the user enters a group name and optionally adds a description. The system generates a unique group ID. Other users can join the group by entering the invite/group code.

Key Steps:

- Create group → store in "Group" collection.
- Generate unique group identifier.
- Add creator as admin.

- Allow others to join via group code.
- Real-time updates notify all members.

11.4 Adding Members to the Group

Group creators or admins can invite new members.

Workflow Explanation:

Admins add new members through the interface. On adding a member, RoomEase updates the group document and broadcasts the update to all users in the group.

Key Steps:

- Admin selects “Add Member”.
- User email or code is validated.
- Member added to group schema.
- Socket.io emits “memberAdded” event.

11.5 Adding an Expense

This is the most frequently performed action.

Workflow Explanation:

Users select a group and add expenses, including the name, amount, payer, and the members involved. The backend applies the splitting algorithm and computes each member’s share.

Key Steps:

- User enters expense details.

- Selects category and group members.
- Expense stored in the database.
- Split calculation logic executed.
- Real-time event updates all members.

11.6 Automatic Split Calculation

The system divides the amount among the selected members.

Workflow Explanation:

RoomEase uses a splitting algorithm to compute the exact balance for each member depending on the type of split (equal, custom, percentage-based, or share-based).

Key Steps:

- Determine number of members.
- Compute individual amount.
- Update balances for each user.
- Store results in "Expense" collection.

11.7 Real-Time Notifications & Live Updates

This is where Socket.io plays a major role.

Workflow Explanation:

Whenever an expense is added, deleted, or edited, all members get instant updates on their interface without refreshing the page.

Key Activities:

- Live sync of new expenses.
- Instant balance updates.
- Real-time chat messages.
- Live status updates for payments.

11.8 Payment Settlement (Online or Offline)

Workflow Explanation:

Users can settle their dues using Razorpay or Stripe payment gateways integrated into the system. Once the payer completes the transaction, the system verifies the payment through a backend webhook and updates the payment status.

Key Steps:

- User selects a pending balance.
- Initiates payment through gateway.
- Payment processed securely.
- Webhook verifies transaction status.
- System updates payment history.
- Real-time update sent to all members.

11.9 Transaction Logging & History Tracking

Every financial action is recorded for future reference.

Workflow Explanation:

Transaction details such as payment ID, payer, receiver, amount, and timestamp are stored in a separate collection. This ensures full transparency and helps during audits or dispute resolution.

Key Steps:

- Store payment data in MongoDB.
- Update user balances.
- Make logs accessible in the dashboard.

11.10 Analytics & Dashboard Visualization

The analytics dashboard gives users a deeper financial insight.

Workflow Explanation:

Charts track monthly expenditures, category-level spending (food, rent, utilities, travel, etc.), group-level contributions, and more. These visual insights help users understand their spending patterns.

Key Steps:

- Fetch expense data from database.
- Preprocess using backend logic.
- Render Pie, Bar, and Line charts.
- Display analytics summary.

11.11 Group Chat Workflow

Workflow Explanation:

Members communicate easily within the group using an integrated chat system. This reduces dependency on external platforms.

Key Features:

- Real-time messaging
- Message timestamps
- User identity tagging
- Chat synchronized across devices

12. Results and Discussion

The development, integration, and testing of RoomEase provided extensive insights into its performance, usability, reliability, and suitability for real-world scenarios. This section presents a comprehensive analysis of the outcomes observed during the project, highlighting both the strengths of the system and the impact it creates in shared-living environments. Each module was tested under different conditions to measure accuracy, speed, scalability, transparency, and user satisfaction.

RoomEase showed promising results across all major functional areas—including automated expense splitting, real-time updates, secure payments, group communication, and data visualization. The following detailed discussion breaks down these observations to illustrate the effectiveness and practicality of the system.

12.1 Accuracy and Consistency of Expense Calculations

One of the primary goals of RoomEase was to eliminate human errors from manual calculations. During testing, the expense splitting algorithm successfully handled all types of scenarios:

- Equal split among all members

- Split among selected individuals
- Uneven or custom shares
- Multiple contributors to a single expense
- Quick recalculations when an expense is edited or removed

Discussion:

The results indicate that the algorithm is robust and reliable. All calculations were consistent even when numerous expenses were added simultaneously. This automation drastically reduces manual workload and prevents confusion, ensuring fairness and transparency within groups.

12.2 Real-Time Updates and User Synchronization

The use of Socket.io allowed live updates across all connected clients. Whether an expense was added, modified, or settled, every member in the group received updates instantly without refreshing the page.

Key Observations:

- Live balances updated in <150 milliseconds
- Chat messages delivered instantly
- New members joining reflected immediately
- No conflicts even when multiple users edited data simultaneously

Discussion:

This real-time functionality is one of the strongest results of the project. It enhances collaboration dramatically and creates a smooth, interactive experience similar to modern communication apps. In shared-living

environments, this feature reduces delays, miscommunication, and financial ambiguity.

12.3 Payment Integration Performance

The integration with Razorpay/Stripe was thoroughly tested using test payment modes and real transactions.

Key Findings:

- Payments were processed securely without failures
- Webhook validation worked accurately
- Successful payments updated the dashboard instantly
- Transaction history was stored correctly in MongoDB
- No mismatch between paid amount and recorded amount

Discussion:

Integrating payment gateways enriched the system significantly. It helped in solving one of the most common problems—delayed settlements. Users do not need to rely on external apps, and every payment is verified, recorded, and reflected automatically. This enhances trust and accountability.

13. Challenges and Solutions

During the development of RoomEase, multiple technical, design, and functional challenges emerged at different stages of the project. Each challenge provided valuable learning opportunities and contributed to making the system more robust, scalable, and user-friendly. This section discusses these challenges in

detail along with the strategies, technologies, and methods used to overcome them. Addressing these issues successfully ensured that the platform performs reliably and efficiently in real-world scenarios.

13.1 Challenge 1: Managing Real-Time Data Synchronization

RoomEase heavily depends on real-time features—expense updates, chat messages, payments, and notifications must appear instantly for all users. Building a system where multiple users interact at the same time introduced synchronization difficulties.

Issues Faced:

- Data conflicts when two users performed actions simultaneously
- Delay in event propagation to other group members

Solution Implemented:

- Integrated **Socket.io** for bidirectional communication
- Implemented **socket rooms** so each group had a separate communication channel

Outcome:

Real-time updates became smooth, reliable, and nearly instantaneous (<150 ms delay), significantly enhancing user collaboration.

13.2 Challenge 2: Ensuring Accurate Expense Splitting

Shared expense splitting is the core logic of the system, and even minor miscalculations could lead to conflicts between group members.

Issues Faced:

- Handling equal vs. custom splits
- Managing decimal precision when dividing money

Solution Implemented:

- Designed a **robust calculation algorithm** for all splitting scenarios
- Applied **BigInt / Number precision techniques** to avoid floating point errors

Outcome:

Expense splitting became accurate, transparent, and fair in all test cases, greatly improving trust among group members.

14. Future Enhancements

RoomEase, in its current form, serves as a powerful and efficient platform for managing shared expenses, offering real-time synchronization, automated calculations, and smooth payment integration. However, there is significant room for further improvement and new capabilities that can enhance usability, scalability, and convenience. Future versions of RoomEase can incorporate advanced technologies such as AI, automation, mobile integration, and predictive analytics to provide a richer user experience.

The following enhancements outline the potential growth of RoomEase and demonstrate how the system can evolve into a more intelligent, feature-rich, and fully automated expense management ecosystem.

14.1 AI-Based Expense Categorization and Spending Suggestions

Expanded Explanation:

AI can be integrated to analyze user spending habits, categorize expenses automatically, and generate personalized suggestions. This improves decision-making and helps users understand how they spend their money over time.

Potential Features:

- Automatically categorize expenses as Food, Travel, Rent, Utilities, etc.
- Suggest monthly budgets based on past data.
- Provide insights such as “You are overspending on groceries compared to last month.”
- Predict upcoming expenses based on patterns.
- Notify users when spending crosses personal limits.

User Benefits:

- Removes the need for manual categorization.
- Helps users maintain financial discipline.
- Enables smart decision-making.

14.2 OCR-Based Bill Scanning (Upload & Auto-Extract Feature)

Expanded Explanation:

OCR (Optical Character Recognition) will allow users to scan receipts or bills using their phone camera. The system automatically detects the amount, date, and category, and fills the expense fields.

Potential Features:

- Scan printed or handwritten receipts.
- Extract details like amount, store name, date, and items.
- Auto-fill expense form to save time.
- Detect duplicate receipts to prevent errors.
- Convert images into digital data using ML models.

User Benefits:

- Removes manual data entry.
- Saves time during trips and shopping.
- Provides higher accuracy in expense tracking.

14.3 Voice-Controlled Expense Entries

Expanded Explanation:

Voice recognition systems (Google Assistant, Siri, or in-app voice engine) can allow users to add expenses using voice commands.

Potential Features:

- “Add ₹500 for groceries split among 3 people.”
- Multi-language support for voice commands.
- Voice confirmation and correction suggestions.
- Hands-free usage especially during travel or shopping.

User Benefits:

- Faster and easier expense entry.

- Helps visually impaired users.
- Useful in situations where typing is inconvenient.

14.4 Dedicated Mobile App (Android & iOS with React Native)

Expanded Explanation:

While the current web application is responsive, a dedicated mobile app would offer better performance, offline mode, push notifications, and deeper device integration.

Potential Features:

- Offline expense recording → Sync when online.
- Push notifications for new expenses and pending payments.
- Fingerprint/Face ID login for added security.
- Background syncing and caching.
- Home screen widgets for quick expense entry.

User Benefits:

- Better accessibility and mobility.
- Native performance on smartphones.
- Enhanced user engagement.

15. Conclusion

The development of RoomEase marks a significant step toward modernizing and simplifying the management of shared expenses, especially in environments where multiple individuals live, travel, or work together. Traditional methods—such as manual calculations, handwritten records, messaging apps, and spreadsheets—often lead to confusion, mismanagement, and disputes due to their lack of structure, transparency, and automation. Through this project, we addressed these issues by creating a comprehensive, real-time, and user-friendly platform that offers a seamless experience for managing group finances.

RoomEase successfully integrates the core components of the MERN stack (MongoDB, Express, React, and Node.js) along with Tailwind CSS, Socket.io, and Razorpay/Stripe payment gateways, resulting in a powerful system capable of handling both the computational and real-time communication requirements of shared expense tracking. The system's architecture ensures smooth interaction between the frontend and backend, efficient data storage, secure authentication, and instantaneous updates across all connected clients.

15.1 Achievements of the Project

✓ Automated Expense Management

One of the most important achievements of the project is the complete automation of expense splitting. Users no longer need to calculate how much each member owes. RoomEase handles this process with accuracy and consistency, significantly reducing the chances of calculation mistakes or misinterpretations.

✓ Real-Time Synchronization

The use of Socket.io provides a superior level of transparency and coordination. Any update—such as adding an expense, editing an amount, or settling dues—is instantly visible to all group members. This level of real-time interaction is crucial in today's fast-paced digital environment.

16. References

1. React.js – Official Documentation
2. Node.js – Developer Guide
3. Express.js – API Documentation
4. MongoDB – Official Database Docs
5. Tailwind CSS – Styling Framework Guide
6. Socket.io – Real-Time Communication Docs
7. Razorpay / Stripe – Payment Gateway API Docs
8. Chart.js – Data Visualization Library Docs
9. JavaScript ES6 – MDN Web Docs
10. MERN Stack Development Tutorials (Online Resources)