PREFACE

In today's fast-paced urban environment, finding suitable accommodation for students and working professionals can be a challenging task. With the rising demand for affordable and accessible rental spaces, digital solutions have become essential in streamlining the search for comfortable living arrangements.

MessFinder is a technology-driven platform designed to simplify the process of locating **mess rooms and rental spaces** for individuals seeking budget-friendly accommodations. By integrating **real-time listings, smart filtering, and secure communication**, MessFinder ensures users can effortlessly discover living spaces that match their needs.

This documentation provides a **comprehensive guide** to the MessFinder system, detailing its **architecture, core functionalities, and implementation strategies** to enhance the rental search experience.

ACKNOWLEDGEMENT

We, the students of **Computer Science, Bankura Sammilani College, Bankura**, proudly declare that our project, **"MESS FINDER,"** has been successfully completed under the expert guidance of **Prof. Sanjoy Sen**, Computer Science Department, Bankura Sammilani College, Bankura. We sincerely express our deepest gratitude for his invaluable support, insightful feedback, and continuous encouragement throughout the development of this project.

We would also like to extend our heartfelt appreciation to **Prof. Tapas Ghosh (Head of Department)** for his unwavering guidance, constructive suggestions, and motivation, which greatly contributed to our learning and project completion.

Furthermore, we are truly grateful to our **families and friends** for their patience, silent cooperation, and emotional support during this challenging yet rewarding journey. Lastly, we acknowledge the contributions of all individuals—known and unknown—who assisted us in various capacities throughout this project.

This project stands as a testament to teamwork, dedication, and the pursuit of technological innovation in solving real-world problems.

PROJECT PROFILE

❖ System information

System definition : MESS FINDER (**MessFinder**)
Type of the Application : Web Application
Time Duration : April 2025 – June 2025

❖ Submitted by:

	NAME	UID
1.	Aakash Shit	22023115004
2.	Moumita Dhabal	22023115036
3.	Priti Banerjee	21023115008
4.	Puja Chowdhury	22023115012
5.	Reshmi Singh	22023115031
6.	Subhadip Garai	22023115022
7.	Souvik Bhattacharya	22023115010
8.	Sujoy Bhakat	22023115015
9.	Susanta Mandi	22023115003

❖ Submitted to:

Department of Computer Science,

Bankura Sammilani College, Bankura

INTRODUCTION

Finding suitable rental accommodations—especially for students and working professionals can be a daunting challenge in urban areas. Traditional rental searches often involve manual inquiries, unreliable listings, and time-consuming processes, making it difficult to locate affordable and convenient living spaces.

MessFinder is a tech-driven solution that streamlines this process by offering a smart, intuitive, and efficient platform for discovering rental spaces. Through real-time listings, user-friendly filters, and direct owner communication, MessFinder ensures a seamless experience in finding suitable mess rooms and rental accommodations.

Designed with modern web technologies and Firebase, the platform supports secure messaging, profile management, and location-based searching, making it an ideal choice for those seeking comfortable and budget-friendly housing solutions.

This document explores MessFinder's architecture, functionalities, and implementation strategies, providing a detailed insight into its development and purpose.

SCOPE OF SYSTEM

MessFinder's scope extends across multiple sectors, enhancing **housing accessibility** through technology. Key applications include:

1. **Students & Educational Institutions** – Helping students find **hostels, PGs, and shared mess accommodations**.
2. **Working Professionals** – Assisting employees in locating **budget-friendly rental spaces** close to workplaces.
3. **Property Owners** – Offering landlords a **convenient way** to list and manage their accommodations.
4. **Security & Trust** – Enabling verified user interactions for **scam-free transactions**.
5. **Government & Public Housing** – Can potentially integrate with **public housing initiatives**.
6. **Smart City Infrastructure** – Contributing to **digitalized rental solutions** for urban areas.

FEASIBILITY STUDY

1. Introduction

Finding suitable rental accommodations is a common challenge for **students and working professionals** in urban areas. Traditional rental processes often rely on **manual inquiries**, outdated listings, and inefficient communication. A **digital solution** can enhance **accessibility, reliability, and efficiency** in finding affordable housing options.

2. Project Overview

MessFinder is a **real-time rental discovery platform** that connects **tenants with landlords**, focusing on **mess rooms, shared accommodations, and independent rental spaces**. The system aims to **eliminate manual searching** and streamline **secure owner-tenant interactions**, ensuring a seamless experience.

3. Key Components

- **Frontend:** A responsive **web application** that enables users to search, filter, and explore listings.
- **Backend:** A robust **Firebase-based infrastructure** to store and manage user data, rooms, and messaging.
- **Communication: Secure chat system** allowing users to interact directly with property owners.

- **Filtering & Search:** Smart filters based on **price, location, and accommodation type** to refine searches effectively.
- **User Authentication:** Secure **login system** to validate users and ensure genuine listings.

4. Methodology

- **User Registration:** Landlords and tenants sign up to access platform features.
- **Room Listings:** Owners add rental properties, including images, location, and pricing details.
- **Search & Filtering:** Users browse listings, apply filters, and shortlist options.
- **Direct Chat Integration:** Tenants and landlords communicate securely via an in-app messaging system.

5. Challenges and Considerations

- **Listing Verification:** Ensuring reliable and scam-free property listings.
- **User Engagement:** Encouraging landlords to keep listings up to date.
- **Privacy & Security:** Protecting user data and messages from misuse.
- **Scalability:** Supporting multiple users without compromising performance.

6. Applications

- **Students:** Quick access to hostels, mess rooms, and PG accommodations.
- **Working Professionals:** Affordable rental spaces near offices and business hubs.
- **Property Owners:** Easy-to-manage listing system with direct tenant interaction.

7. Technical Feasibility

MessFinder leverages **modern web technologies** for an optimized user experience:

- **Frontend:** React.js for dynamic UI and responsive design.
- **Backend:** Firebase for real-time database management and authentication.
- **Communication:** WebRTC-based secure chat system.
- **Hosting:** Cloud-based deployment for accessibility across devices.

8. Economic Feasibility

- **Development Cost:** Open-source and cloud-based, reducing infrastructure expenses.
- **Maintenance:** Minimal due to Firebase's automatic scaling.

- **User Benefits:** Increased efficiency, accurate listings, and reduced search time.

9. Operational Feasibility

- **User Accessibility:** Designed for **non-tech users** with a simple interface.
- **Integration:** Compatible with existing rental platforms or standalone operation.
- **Scalability:** Handles a growing number of users and property listings.

10. Schedule Feasibility

The project follows a structured timeline:

- **Phase 1:** Platform development & testing.
- **Phase 2:** User onboarding & listing population.
- **Phase 3:** Optimization & feature expansion based on user feedback.

SYSTEM PROFILE

OVERVIEW

MessFinder is a **web-based accommodation discovery platform** designed for **students and working professionals**. It helps users find suitable **mess rooms, PGs, hostels, and rental spaces**, simplifying the search process with **real-time listings, smart filters, and direct communication with property owners**.

SCOPE OF SYSTEM

- Facilitates **searching and listing** rental accommodations.
- Enables **direct messaging** between tenants and landlords.
- Provides **secure authentication** and user management.
- Allows **filtered searches** based on budget, location, and room type.
- Supports **verified listings** to prevent fraudulent postings.

MODULES AFTER LOGIN BY OWNER

- **User Authentication Module** – Secure login/signup via Firebase.
- **Room Listing & Management** – Owners can add, update, and remove listings.

- **Search & Filtering Module** – Users refine searches based on key parameters.
- **Chat Module** – Secure communication between tenants and landlords.
- **User Profile & Preferences** – Tenants can save/bookmark listings.

▪ MODULES AFTER LOGIN BY USER

- **Dashboard** – Personalized recommendations and saved listings.
- **Search & Filters** – Advanced filtering for targeted search results.
- **Chat & Interaction** – Direct messaging with property owners.
- **Profile Management** – Users update their details and preferences.
- **Booking & Inquiry System** – Ability to express interest in a room.

SYSTEM REQUIREMENTS

HARDWARE COMPONENTS:

- **Server:** Cloud-based infrastructure (Firebase) for scalability.
- **User Devices:** Any laptop, mobile, or desktop with browser and Internet support.
- **Storage:** Cloud storage for user data and images (Firebase Storage).

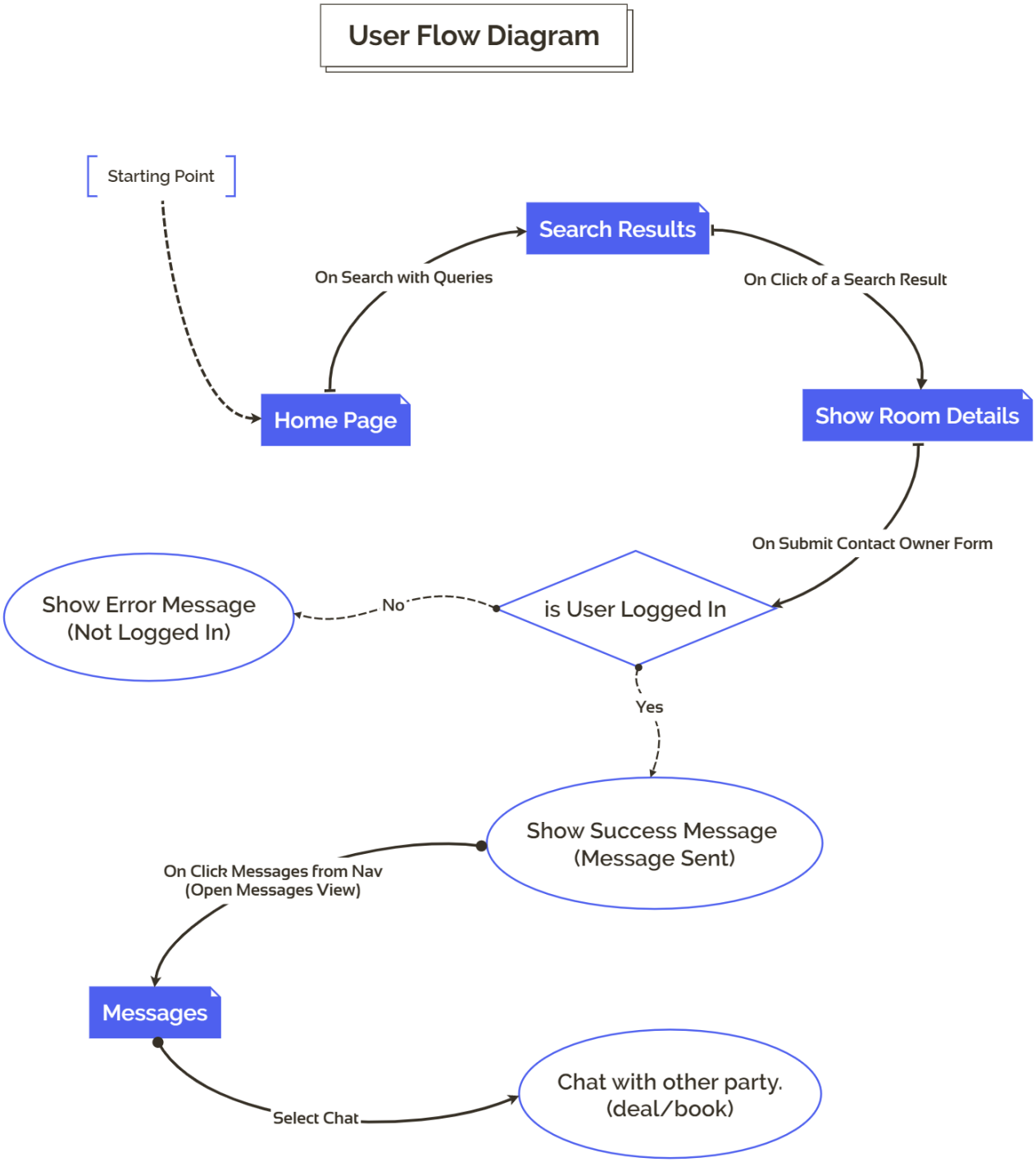
SOFTWARE COMPONENTS:

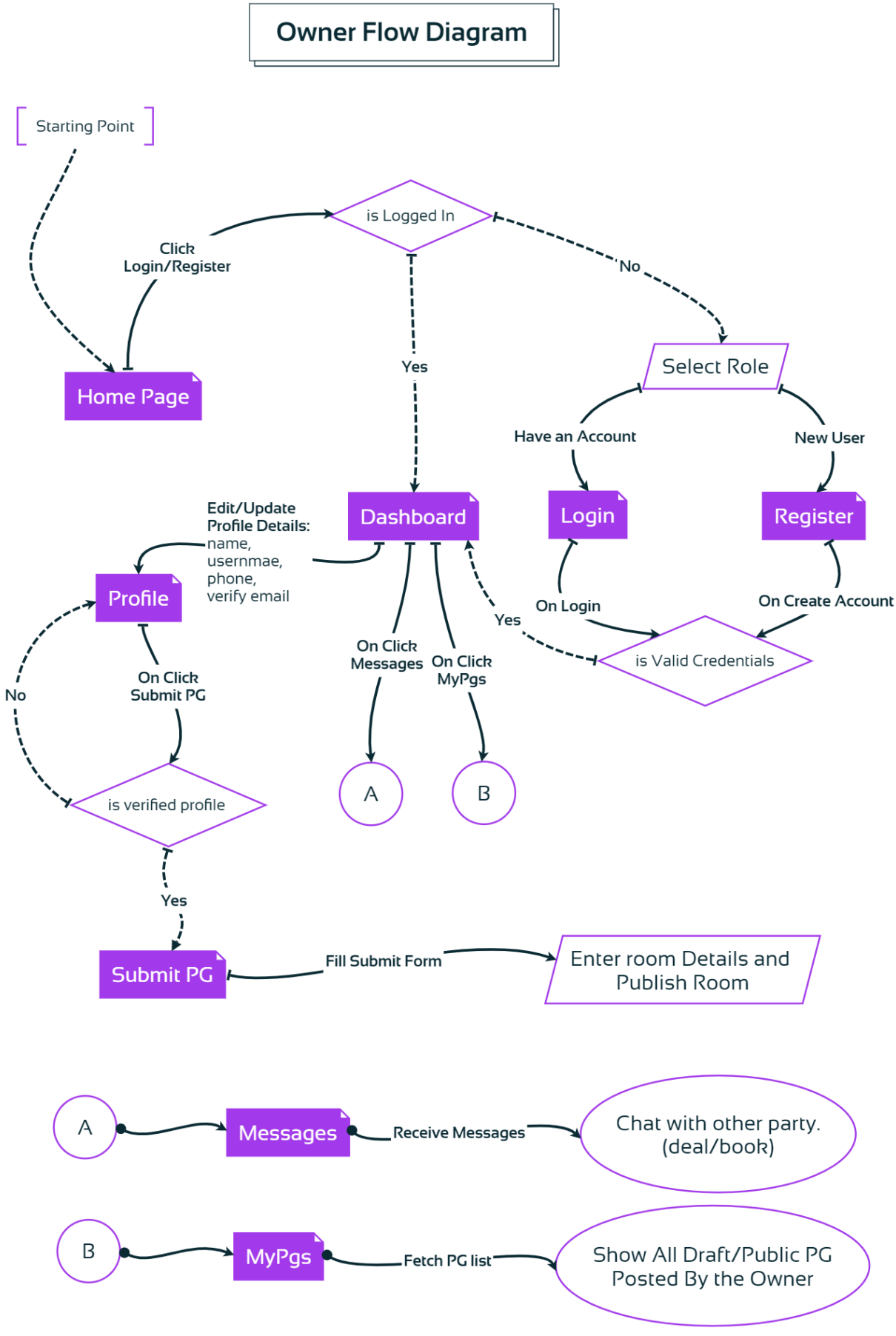
- **Frontend UI:** Designed for seamless user experience using React.js.
- **Backend Services:** Firebase for authentication, database management, storage, and messaging.
- **Search & Filtering System:** Efficient algorithms for customized search results.
- **Messaging Module:** Real-time chat system for tenant-owner interaction.

TOOLS AND TECHNOLOGY

- Programming Languages:
  - HTML,
  - CSS,
  - JavaScript (React.js for frontend),
  - Firebase (for Backend Services).
- Libraries & Frameworks:
  - React Router DOM,
  - Tailwind CSS,
  - Firebase SDK.
- Database System:
  - Firebase Realtime Database for dynamic data handling.
- Operating Systems:
  - Platform-independent (works on Windows, macOS, Linux).

SYSTEM FLOW DIAGRAM




Owner Flow Diagram
Starting Point
is Logged In
Click
Login/Register
No
Yes
Home Page
Select Role
Have an Account
New User
Dashboard
Login
Register
Edit/Update
Profile Details:
name,
usernmae,
phone,
verify email
Profile
On Login
On Create Account
Yes
is Valid Credentials
On Click
Messages
On Click
MyPgs
No
On Click
Submit PG
A
B
is verified profile
Yes
Submit PG
Fill Submit Form
Enter room Details and
Publish Room
A
Messages
Receive Messages
Chat with other party.
(deal/book)
B
MyPgs
Fetch PG list
Show All Draft/Public PG
Posted By the Owner

MODULES

MessFinder consists of several essential modules to ensure **smooth operation and user experience**:

1. **User Interface Module** –

   Intuitive dashboard for **searching, filtering, and listing accommodations**.

2. **Database Management Module** –

   Firebase backend for **user authentication, room listings, and chat records**.

3. **Search & Filtering Module** –

   Smart filters based on **budget, sharing type, and location**.

4. **Secure Chat Module** –

   Real-time **messaging** for **tenant-owner interactions**.

5. **Security & Authentication Module** –

   User verification and **secure login system**.

ROLES & RESPONSIBILITIES

1. Property Owners:

1. Register and create a verified owner profile.
2. Add new room listings.
3. Edit or remove listings as needed.
4. View and respond to messages from interested tenants.

2. Users (Tenants)

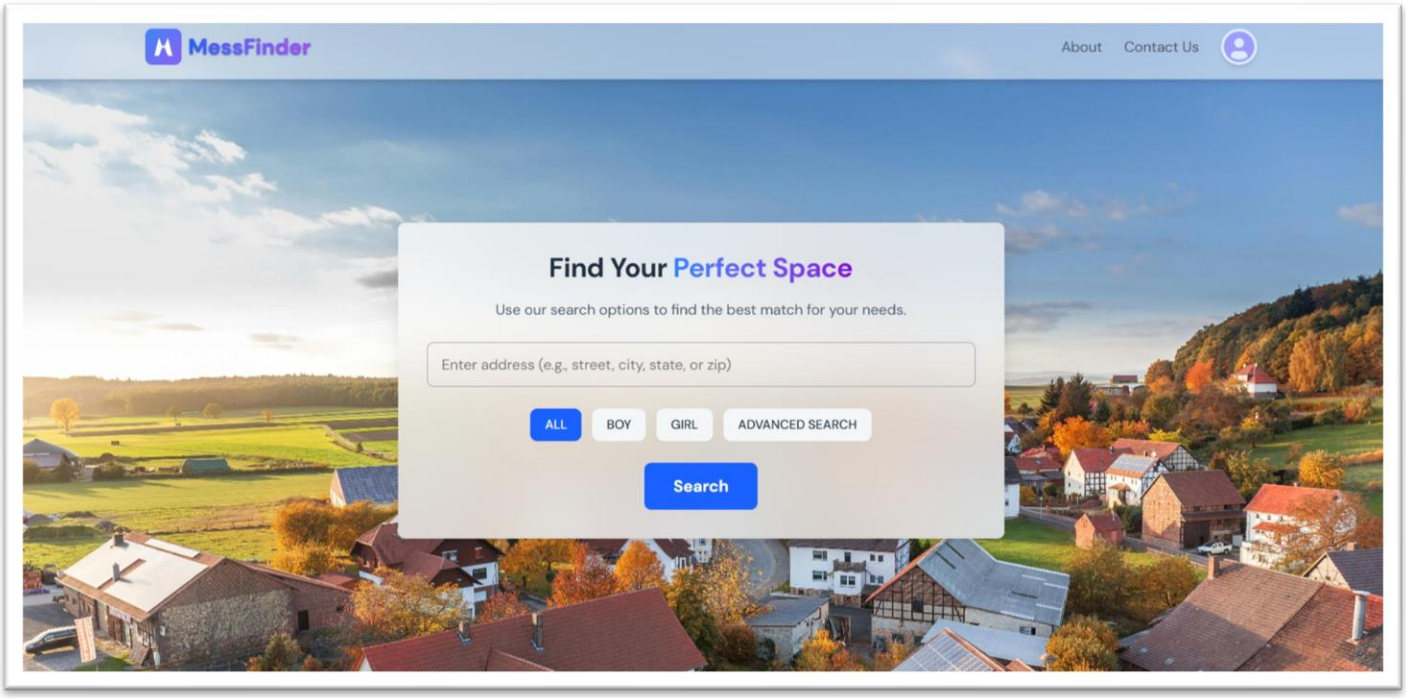
1. Sign up and complete profile for a personalized experience.
2. Search for rooms using filters like price, mess type, and location.
3. Contact property owners through secure chat.
4. Report any suspicious activity or false listings.

3. Unauthorized Users (Guests)

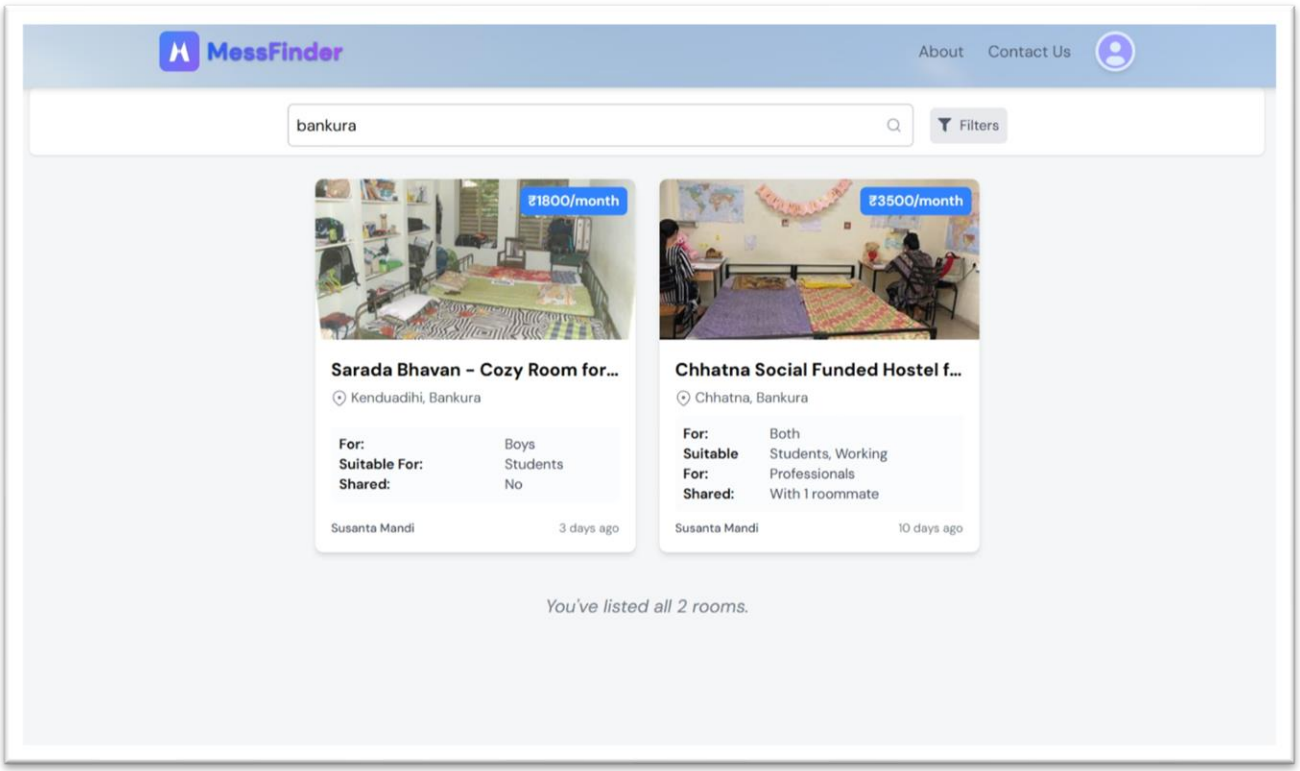
1. Can browse only basic or public listings.
2. Cannot use filters, chat, or save options.
3. No access to profile, dashboard, or private features.

ALL PAGES

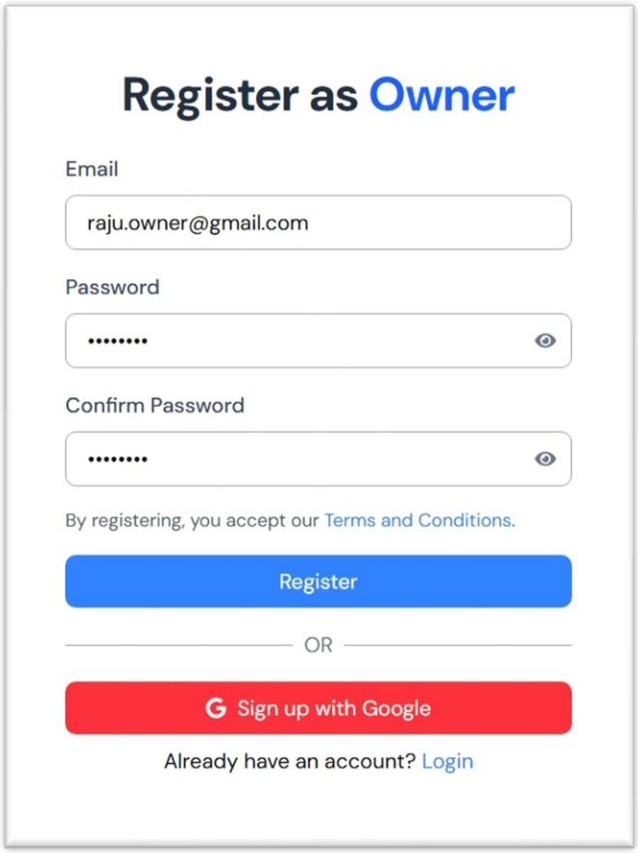
HOME PAGE



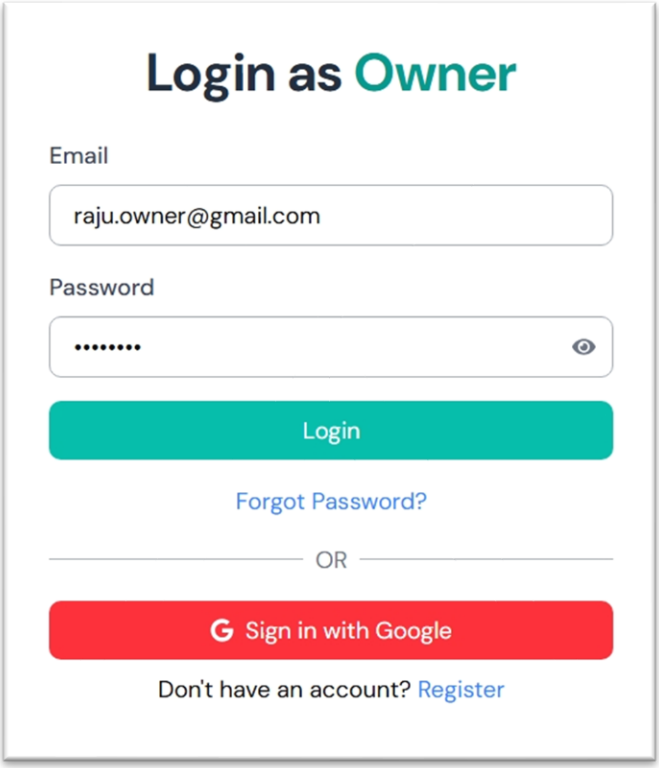
SEARCH RESULT LIST



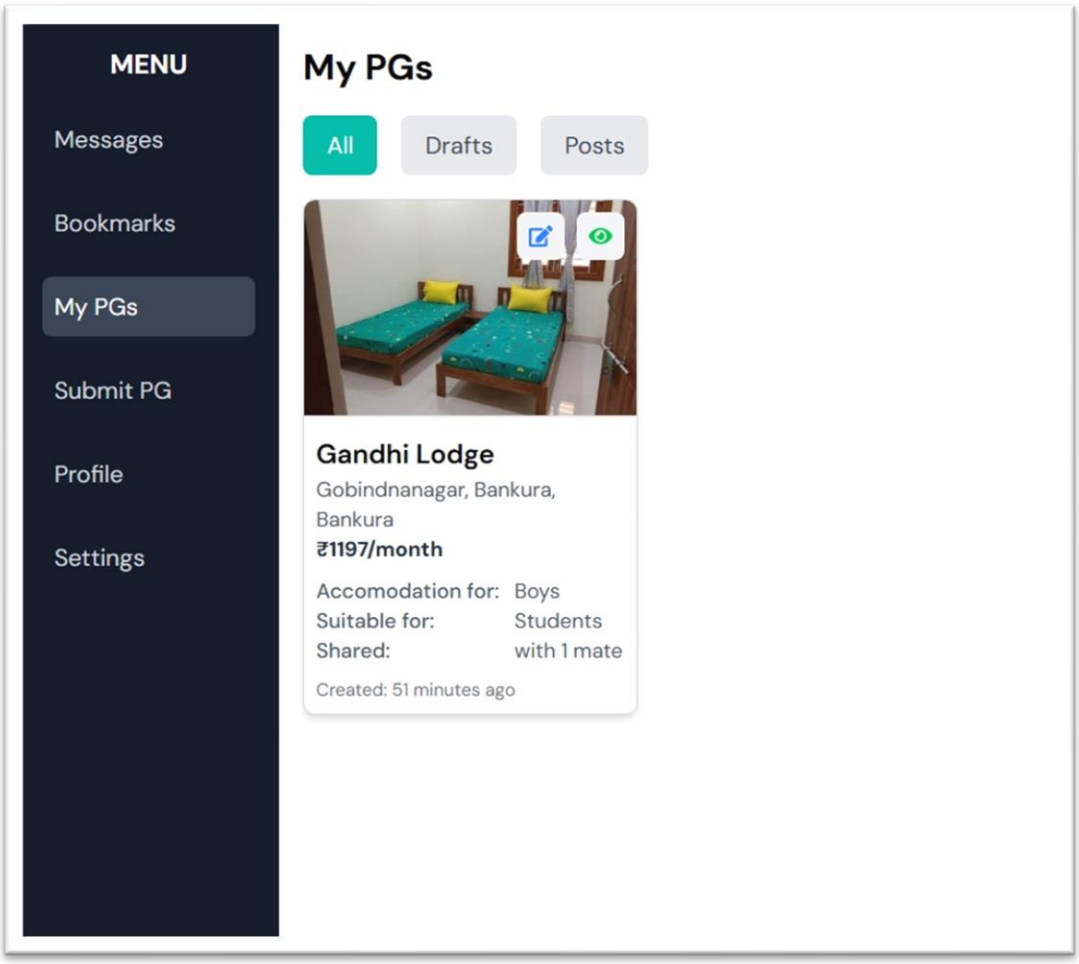
REGISTRATION FORM



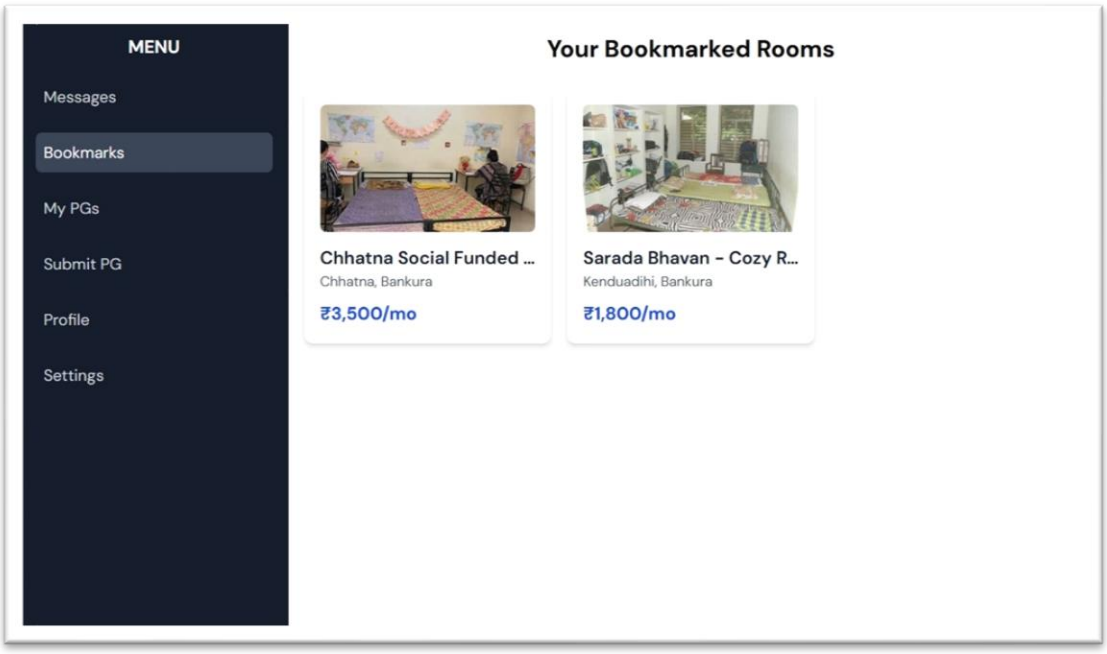
LOGIN FORM



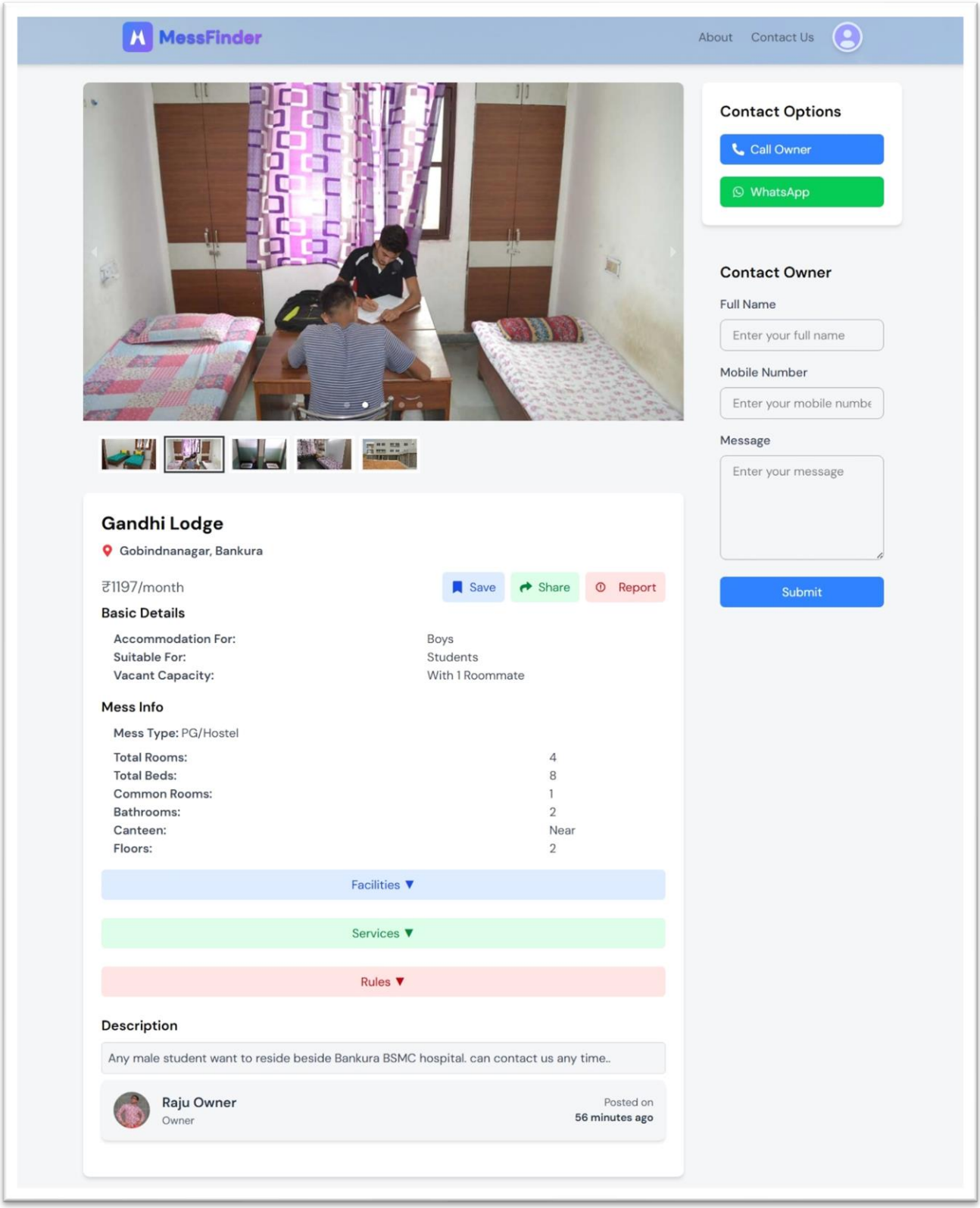
DASHBOARD



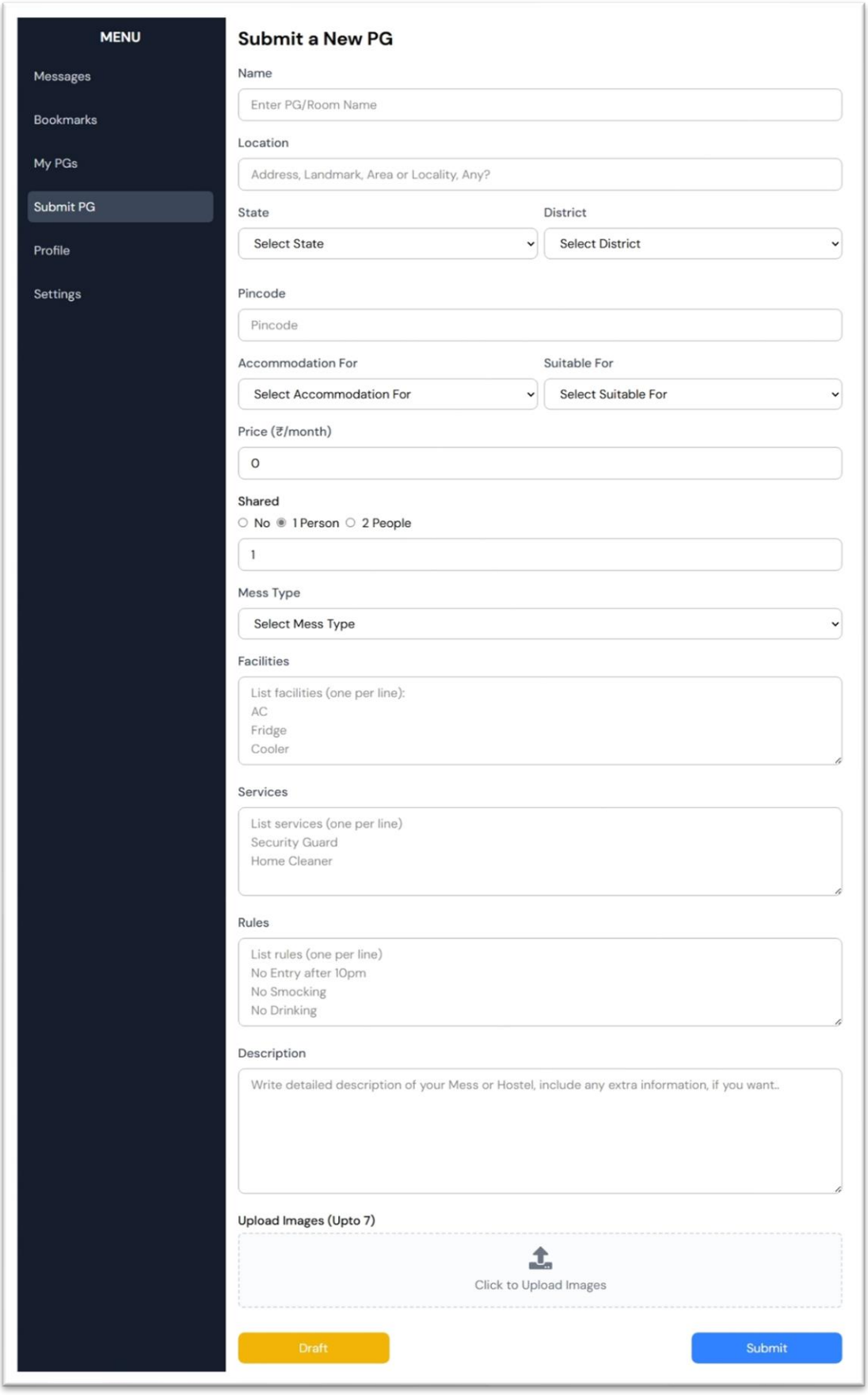
BOOKMARKS PAGE



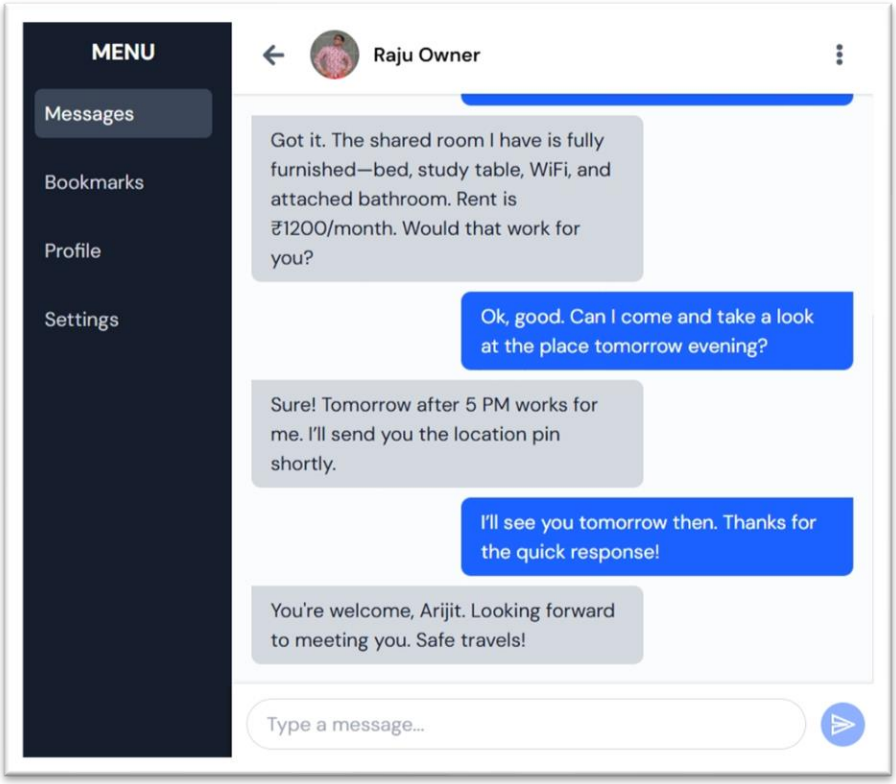
ROOM VIEW


MessFinder
About
Contact Us
Contact Options
Call Owner
WhatsApp
Contact Owner
Full Name
Enter your full name
Mobile Number
Enter your mobile numbe
Message
Enter your message
Submit
Gandhi Lodge
Gobindnanagar, Bankura
₹1197/month
Save
Share
Report
Basic Details
Accommodation For: Boys
Suitable For: Students
Vacant Capacity: With 1 Roommate
Mess Info
Mess Type: PG/Hostel
Total Rooms: 4
Total Beds: 8
Common Rooms: 1
Bathrooms: 2
Canteen: Near
Floors: 2
Facilities ▼
Services ▼
Rules ▼
Description
Any male student want to reside beside Bankura BSMC hospital. can contact us any time..
Raju Owner
Owner
Posted on
56 minutes ago

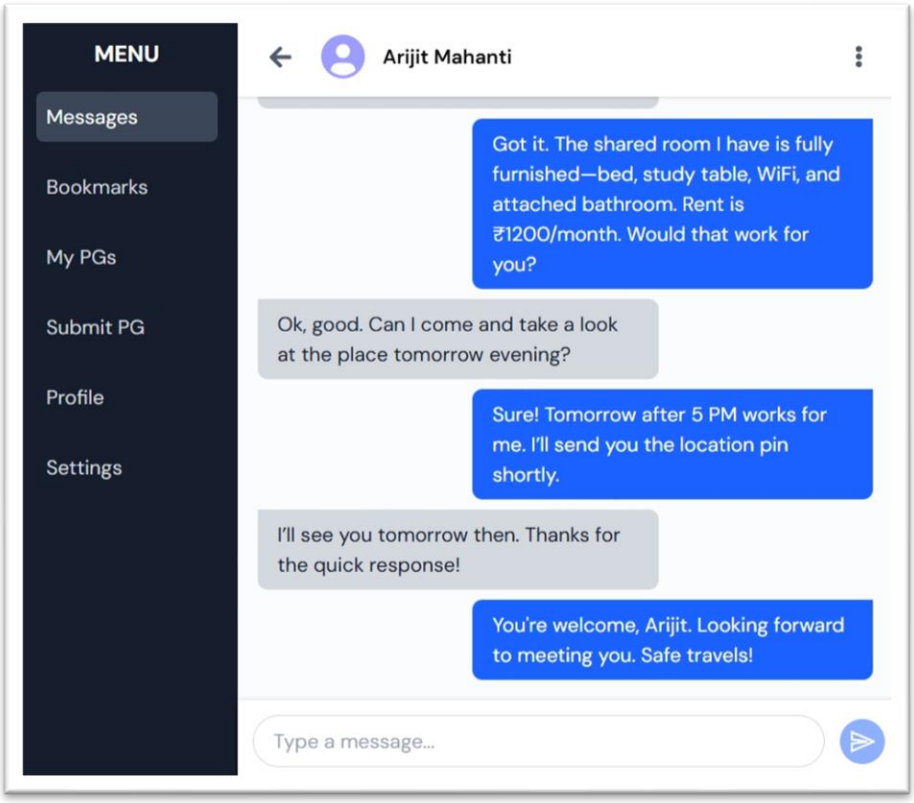
SUBMIT PG FORM



MESSAGE VIEW USER



MESSAGE VIEW OWNER



FOLDER STRUCTURE

The MessFinder project is organized into several folders to keep the code clean and manageable. Below is a simple explanation of each major folder:

```
MessFinder/                  # Main source directory
├── src/                     # Reusable UI and functional components
│   ├── components/          # Error display components
│   │   ├── error/           # Header, Footer, and layout elements
│   │   ├── layout/          # Form components for property listing
│   │   ├── owner-form/      # Generic UI elements like buttons, inputs, etc.
│   │   └── ui/              # Firebase configuration and React context providers
│   ├── context/             # Custom JavaScript modules and utilities
│   ├── module/              # Custom styles
│   │   ├── css/             # JS helpers (districts, nav items, etc.)
│   │   └── js/              # Route-based component structure
│   └── pages/               # Authentication (login, register, reset)
│       ├── auth/            # User dashboard and settings
│       ├── dashboard/       # Home and search
│       ├── home/            # Static info pages (About, Contact, FAQs)
│       ├── info/            # User and public profile
│       ├── profile/         # Room details and search results
│       └── rooms/           # Public assets (favicon, images, etc.)
├── public/                  # Global styles
├── index.css                # Base HTML file
├── index.html               # Main layout structure
├── Layout.jsx               # React app entry point
├── main.jsx                 # App routing configuration
├── routes.jsx               # Scroll-to-top functionality on route change
├── scroll-to-top.jsx        # License file
├── LICENSE                  # Project overview and instructions
├── README.md                # Additional config files (.gitignore, etc.)
└── ...
```

src/

This is the main folder that contains all the core code of the application.

- **components/** – Reusable parts of the UI, like buttons, forms, header, footer, error pages, etc.

- **context/** – Contains Firebase setup and authentication logic used across the app.
- **module/** – Stores helper functions, custom JavaScript files, and reusable logic like pin codes and string functions.
- **pages/** – Contains all the pages linked to routes like home, login, dashboard, profile, etc.

public/

This folder contains public assets like images, the favicon, and the main index.html file used by the browser.

index.css: The main stylesheet that defines global styles for the application.

Layout.jsx: Defines the overall layout used across pages, combining header, footer, and body content.

main.jsx: Entry point of the React application where the app is initialized.

routes.jsx: Defines all the routes (URLs) and links them to the corresponding pages.

DATABASE STRUCTURE

MessFinder uses **Firebase Realtime Database**, a cloud-hosted **NoSQL** database that stores data in **JSON** format. This enables real-time syncing between users and devices, making it ideal for chat, listings, and user activity tracking.

What is a NoSQL Realtime Database?

A **NoSQL** database stores data as key-value pairs, arrays, or nested objects instead of traditional tables. Firebase Realtime Database is:

- **JSON-based**: Data is stored as a tree-like JSON object.
- **Realtime**: Changes are instantly synced across all clients.
- **Scalable**: Optimized for mobile and web apps.

Structure

- **bookmarks/**: Stores saved room IDs for each user.
- **chats/**: Stores chat data between users and owners.
- **info/**:
  - contactus/: Stores user contact messages.
  - report/: Tracks reports against owners or listings.
- **rooms/**: All the property listing details.
- **userChats/**: Maps chat references for easy access by user or owner.
- **users/**: Stores user profile and role information.

Database Structure – JSON Based

```
mess-finder/
|
├── bookmarks/
|   └── [userId]/
|       └── [bookmarkId]: { createdAt, roomId }
|
|
├── chats/
|   └── [chatId]/
|       ├── messages/
|       |   └── [messageId]: { text, sent, timestamp }
|       ├── ownerId
|       └── userId
|
|
├── info/
|   ├── contactus/
|   |   └── [messageId]: { name, email, message, uid, createdAt }
|   └── report/
|       └── [reportId
]: { reason, description, ownerId, uid, createdAt }
|
├── rooms/
|   └── [roomId]: {
|         name, location, facilities, services, price,
|         ownerId, images[], rules, description,
|         messInfo: { messType, totalRooms, totalBeds, etc. }
|     }
|
├── userChats/
|   └── [userId]/
|       └── [chatRefId]: { chatId, userId or ownerId }
|
|
├── users/
|   └── [userId]: {
|         displayName, email, phoneNumber,
|         role, createdAt, updatedAt
|
|     }
```