DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

GharSeva

A PROJECT REPORT

Submitted by

Himanshu Gupta (23BCS10889)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Chandigarh University

June, 2025

Table of Content

Abstract

GharSeva is a modern web-based platform designed to simplify the process of finding, listing, and booking rental accommodations such as flats and PGs (Paying Guests). In today's fast-paced urban environment, students and working professionals often struggle to locate verified and affordable housing options. Traditional offline methods or scattered online listings lack transparency, proper filtering, and a seamless booking experience. GharSeva aims to bridge this gap by providing a centralized, user-friendly solution that connects property owners and tenants directly.

The system enables property owners to register and list their accommodations with details such as location, price, amenities, and availability. Users can browse, search, and filter listings according to their preferences, view property details, and proceed with booking requests securely. The platform also integrates review and rating features to enhance trust and reliability between users.

The frontend of GharSeva is developed using **React.js** and **Tailwind CSS**, ensuring a responsive and visually appealing interface. The backend and database components are integration with modern technologies such as **Spring boot** and **PostgreSQL**, ensuring scalability and performance.

Once fully implemented, GharSeva will streamline the rental discovery process, reduce dependency on middlemen, and promote a transparent housing ecosystem for students and professionals alike.

Introduction

Background & Motivation

Finding reliable rental accommodations such as flats and PGs is a common challenge faced by students and working professionals, especially in university towns and urban areas. The process often involves multiple intermediaries, unverified listings, and misleading information, which lead to wasted time and effort. With the increasing digitization of services, there is a growing need for a dedicated, transparent, and efficient online platform that simplifies the accommodation search and booking process.

GharSeva was conceptualized to address these issues by creating a one-stop platform where users can easily find verified properties, interact directly with owners, and make informed decisions. By leveraging modern web technologies, GharSeva ensures a smooth user experience while maintaining data integrity and security.

Problem Statement

Existing property rental systems are often fragmented, unreliable, or lack crucial features like real-time availability, user reviews, and secure booking workflows. Students and professionals spend considerable time browsing through multiple sources or dealing with brokers who charge high commissions. Property owners, on the other hand, lack a simple interface to manage listings and reach potential tenants directly.
GharSeva aims to eliminate these inefficiencies by providing a unified platform that connects property owners and seekers in a transparent, verified, and user-friendly environment.

Objectives

The main objectives of the GharSeva system are:

- To develop a digital platform for **user and owner registration and authentication**.
- To enable **property listing, search, and filtering** based on location, price, and amenities.
- To allow **secure booking and review functionalities** for better transparency.
- To design an intuitive and responsive user interface that enhances accessibility.
- To ensure **data consistency and security** through robust backend and database integration.

Scope & Limitations

The current scope of GharSeva includes the core modules

- User authentication
- Property listing
- Property Addition
- Search
- Booking

Implemented through a **React.js frontend**, a **Spring Boot backend**, and a **PostgreSQL database**. The system focuses on providing a web-based solution accessible across devices. However, certain features such as **reviews, advanced admin configurations, online payment integration**, **AI-based recommendation systems**, and **mobile application deployment** are planned for future versions.

Additionally, while the platform ensures secure user interaction, it currently assumes the accuracy of property data provided by owners, which may be enhanced later through verification mechanisms.

System Analysis

Requirement Gathering: Functional Requirements

The GharSeva platform supports the following functional requirements:

- User Registration & Login:
  Users (tenants, owners, and admins) can register and log in securely through the system. Authentication is handled using Spring Boot with JWT.
- Property Management:
  Owners can create, update, and delete their property listings with details such as title, location, price, and description.
- Search & Filter:
  Tenants can search and filter listings based on budget and in future amenities etc. ensuring a quick and relevant property search experience.
- Admin Management:
  Admins can monitor user activity, verify listings, and remove fraudulent accounts or properties.

Non-Functional Requirements

- Performance: The system should respond within 3 seconds for most operations, even under concurrent usage.
- Scalability: The backend and database should handle growing data volumes and users.
- Security: Data is encrypted, passwords hashed, and API endpoints protected using JWT authentication.
- Usability: A responsive and accessible interface is built using React.js and Tailwind CSS.
- Reliability: PostgreSQL ensures data integrity and ACID compliance.
- Maintainability: The project should follow a layered architecture (Controller → Service → Repository) for easy updates and testing.

Use Case Diagram / Actor Description

Actors:

1. Tenant: Registers, searches, books properties, and posts reviews.
2. Owner: Lists, edits, and manages properties, and handles booking requests.
3. Admin: Oversees the platform, verifies users and listings, and ensures smooth operation.

Use Cases:

- Tenant → Register/Login → Search Property → Book Property → Write Review
- Owner → Register/Login → Add Property → Manage Bookings → View Feedback
- Admin → Manage Users → Manage Listings → Monitor Activity

Data Flow Diagram (DFD)

Level 0 – Context Diagram

- Actors: Tenant, Owner, Admin
- System: GharSeva Platform
- Data Flow:
    - Tenant sends login and booking requests.
    - Owner manages listings and booking responses.
    - Admin validates and monitors overall operations.

Level 1 – Process Flow

1. User interacts with the React.js Frontend.
2. Requests are handled by Spring Boot REST APIs.
3. Data is stored/retrieved from PostgreSQL Database.
4. Admin dashboard fetches analytics and management data.

Entity–Relationship Diagram (ERD) / Database Schema

Below is the proposed relational schema for the PostgreSQL database used in the GharSeva system:

Entities & Attributes:

1. **User**
   - user_id (PK)
   - name
   - email
   - password
   - role (Tenant/Owner/Admin)
   - phone

2. **Property**
   - property_id (PK)
   - owner_id (FK → User.user_id)`
   - title
   - description
   - address
   - city
   - price_per_month
   - amenities
   - availability_status

3. **Booking**
   - booking_id (PK)
   - tenant_id (FK → User.user_id)`
   - property_id (FK → Property.property_id)`
   - booking_date
   - status (Pending/Accepted/Rejected)

4. **Review**
   - review_id (PK)
   - tenant_id (FK → User.user_id)`
   - property_id (FK → Property.property_id)`
   - rating (1–5)
   - comment
   - review_date

Data Relationships

- One **Owner** can have multiple **Properties**.
- One **Tenant** can make multiple **Bookings**.
- One **Property** can have multiple **Reviews**.
- Each **Booking** connects one Tenant to one Property.

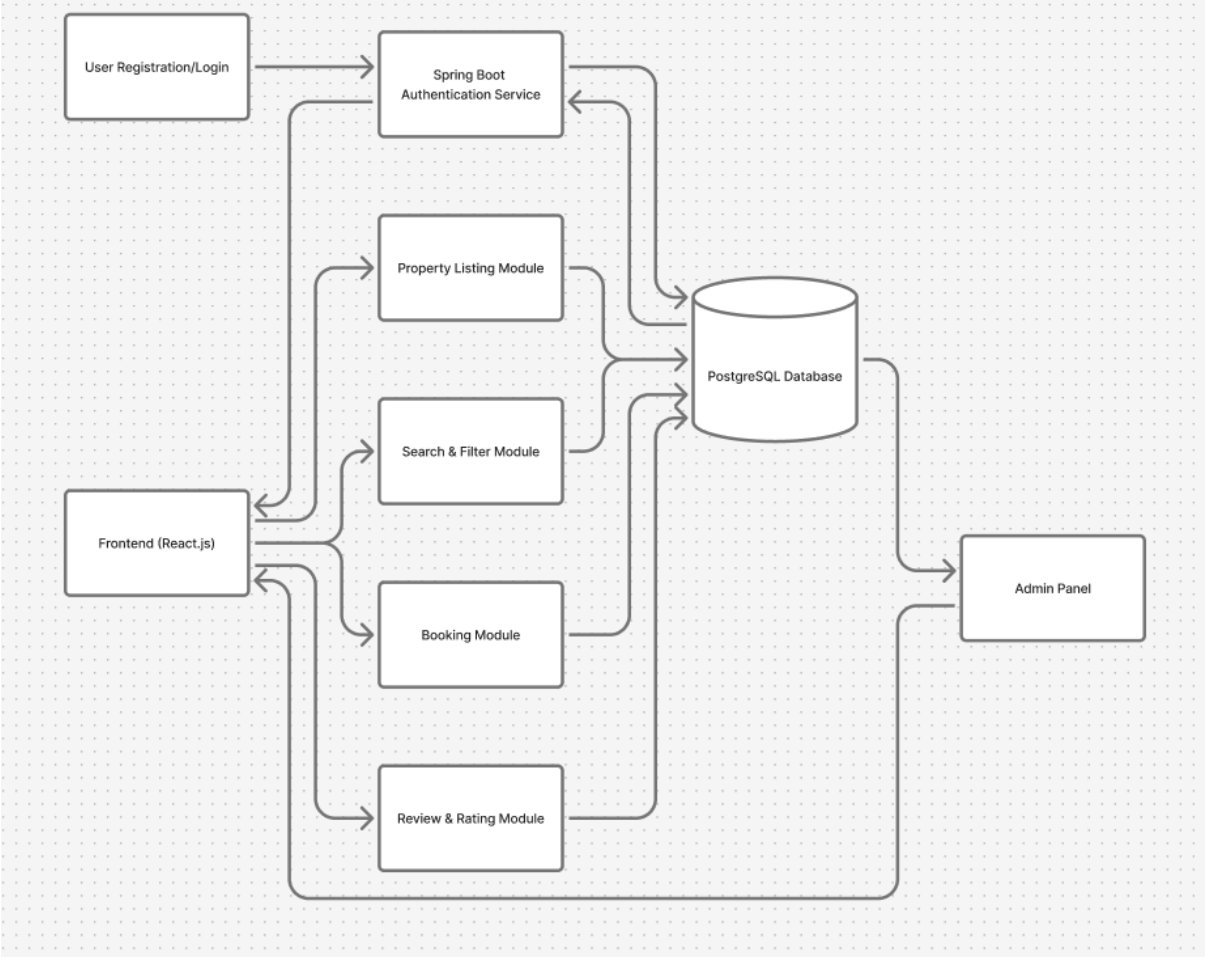

Figure 1: Relationship Diagram

Implementation

The **GharSeva** application follows a **three-tier architecture** consisting of the presentation layer (frontend), application logic layer (backend), and data storage layer (database).

Frontend (React.js + Tailwind CSS)

The frontend was implemented using **React.js** to ensure a responsive, component-based user interface. The major components developed include:

- **Home.jsx:** Displays featured properties and search bar.
- **PropertyList.jsx:** Fetches and renders property cards using API data.
- **PropertyDetail.jsx:** Shows complete details of a selected property with images, description, and amenities.
- **BookingForm.jsx:** Allows tenants to request property bookings.
- **Auth.jsx:** Handles login and registration for both tenants and owners.

State Management:

React Hooks such as useState, useEffect, and useContext were used for managing application state and side effects.
Routing between pages was managed using **React Router DOM**.

Third-Party Libraries:

- react-router-dom – For navigation.
- tailwindcss – For responsive styling.
- react-icons – For UI icons.

Backend (Spring Boot)

The backend was implemented using **Spring Boot**, following a modular layered architecture:

1. **Entity Layer:**
   Defines models like User, Property, Booking, and Review.
2. **Repository Layer:**
   Interfaces extending JpaRepository handle CRUD operations.
3. **Service Layer:**
   Contains business logic for user authentication, property management, and bookings.
4. **Controller Layer:**
   Defines REST APIs for frontend communication using endpoints such as:
   - POST /api/auth/register
   - GET /api/properties
   - POST /api/bookings

Technologies Used:

- Spring Boot
- Spring Data JPA
- Spring Security (with JWT)
- PostgreSQL (via JDBC)

Database (PostgreSQL)

A **PostgreSQL** database was used for structured data management. Entities were mapped using **JPA annotations**.

Future Work / Enhancements

Several advanced modules are planned for future development:

1. **Online Payment Integration:**
   Integration of Razorpay or Stripe API to enable secure rent and booking payments.
2. **AI-Based Recommendations:**
   Personalized property suggestions using machine learning models based on user preferences.
3. **Mobile Application:**
   A React Native version to provide a mobile-first experience.
4. **Push Notifications:**
   Real-time alerts for booking confirmations and property updates.
5. **Roommate Matching:**
   Smart algorithm to match compatible tenants based on preferences and habits.
6. **Admin Analytics Dashboard:**
   Visualization of platform activity and insights using charting libraries.

Conclusion

The **GharSeva** project successfully demonstrates the development of a scalable, secure, and user-friendly accommodation management platform. By combining a responsive **React.js** frontend, a robust **Spring Boot** backend, and a reliable **PostgreSQL** database, the system achieves its core objectives of simplifying the property rental process, connecting owners and tenants, and ensuring data transparency.

This project not only strengthened technical skills in **full-stack development technologies (ReactJS, Java, DBMS, SpringBoot, REST APIs, Postman etc.)** but also improved understanding of **system design, API communication, and database modeling**.

With future enhancements like payment integration and mobile support, GharSeva can evolve into a comprehensive real-estate management platform benefiting students and professionals alike.