Final Report

# Capstone Project: RentAPlace

**Submitted by: Souvik Pradhan**

**Batch: .Net Angular**

**Instructor: Parth Shukla**

**Date: September 2025**

# Table of Contents

1. Problem Definition and Objectives

2. Frontend & Backend Architecture

3. Component Breakdown & API Design

4. Database Design & Storage Optimization

5. Screenshots

6. Deployment

7. Conclusion

# Problem Definition and Objectives

RentAPlace is an online platform for renting homes for short and long-term durations. The application serves two types of users: regular users (renters) and owners. Users can search for properties, while owners can manage their listings.

## Objectives:

- Allow users to register, login, search, and reserve properties  
- Enable owners to list, update, and manage properties  
- Provide secure communication between users and owners  
- Implement reservation, review, and messaging system  
- Ensure authentication using JWT tokens

# Frontend & Backend Architecture

The project uses Angular for the frontend, .NET Core 8 Web API for the backend, and SQL Server as the database. Authentication is managed using JWT.

## System Architecture Diagram:



# Component Breakdown & API Design

## Frontend Components:

- State management using Angular services  
- Routing with Angular Router  
- UI Components: Login/Register, Property List, Property Details, Reservation, Owner Dashboard, Messaging

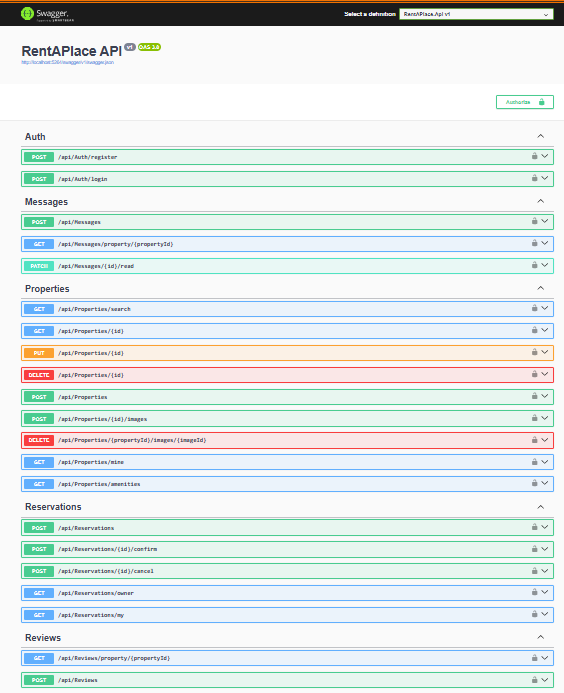
## Backend API Endpoints:

Auth:  
 - POST /api/Auth/register  
 - POST /api/Auth/login  
  
Properties:  
 - GET /api/Properties/search  
 - GET /api/Properties/{id}  
 - GET /api/Properties/mine  
 - POST /api/Properties  
 - PUT /api/Properties/{id}  
 - DELETE /api/Properties/{id}  
  
Reservations:  
 - POST /api/Reservations  
 - POST /api/Reservations/{id}/confirm  
 - GET /api/Reservations/my  
  
Messages:  
 - POST /api/Messages  
 - GET /api/Messages/property/{propertyId}  
  
Reviews:  
 - GET /api/Reviews/property/{propertyId}  
 - POST /api/Reviews

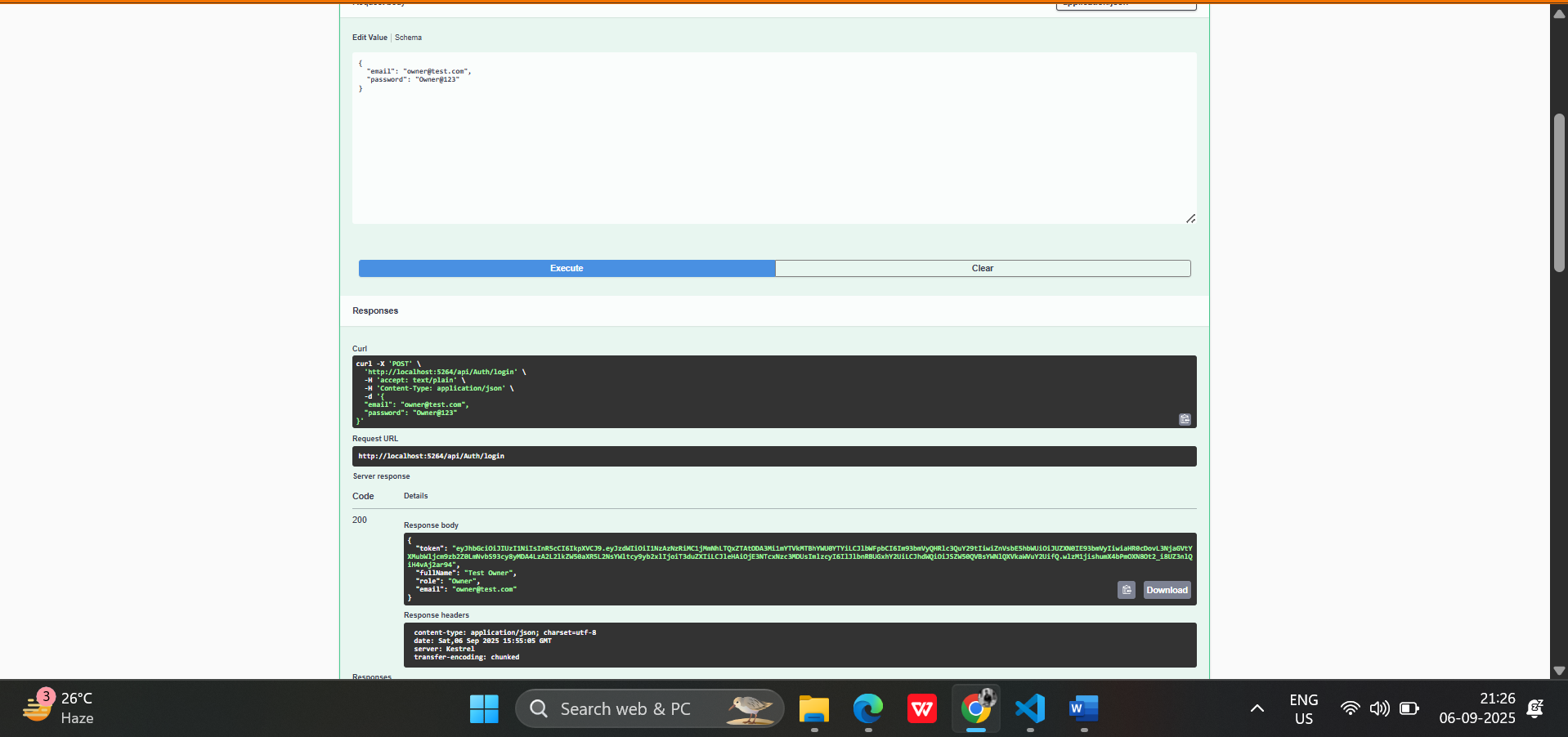
**All backend APIs were tested using Swagger UI before integrating with Angular frontend.**

Coverage:

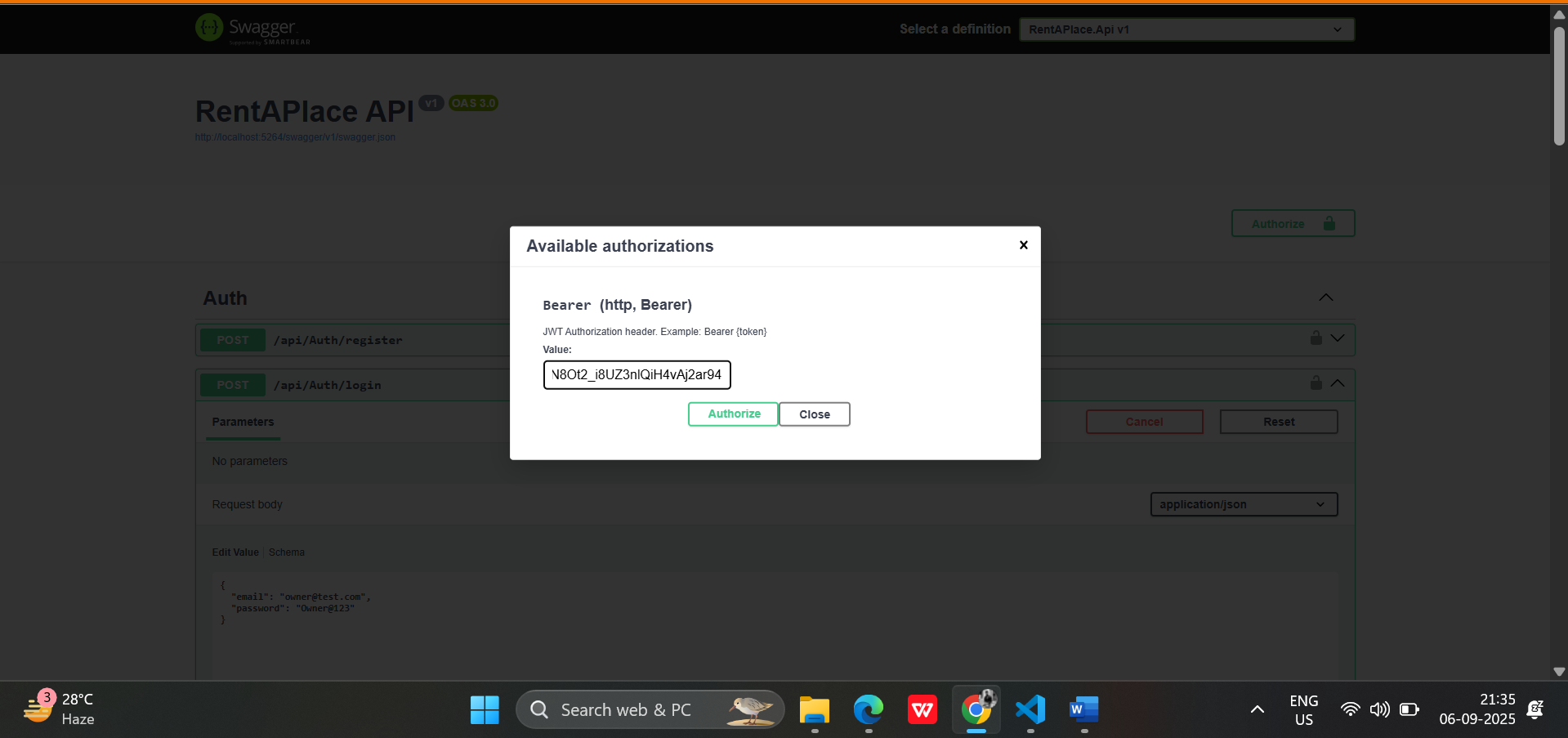
* User Authentication (Login/Register)
* Property CRUD (Add, Update, Delete, Get)
* Reservation API (Book a property)
* Messaging API (Send/Receive messages)

****

**While logging in it is generating the JWT token-**

****

This JWT token will not expire for 30 min because we set the expire timer for 30 min. In between 30 min every request will carry this JWT token as a proof of authorization & authentication.

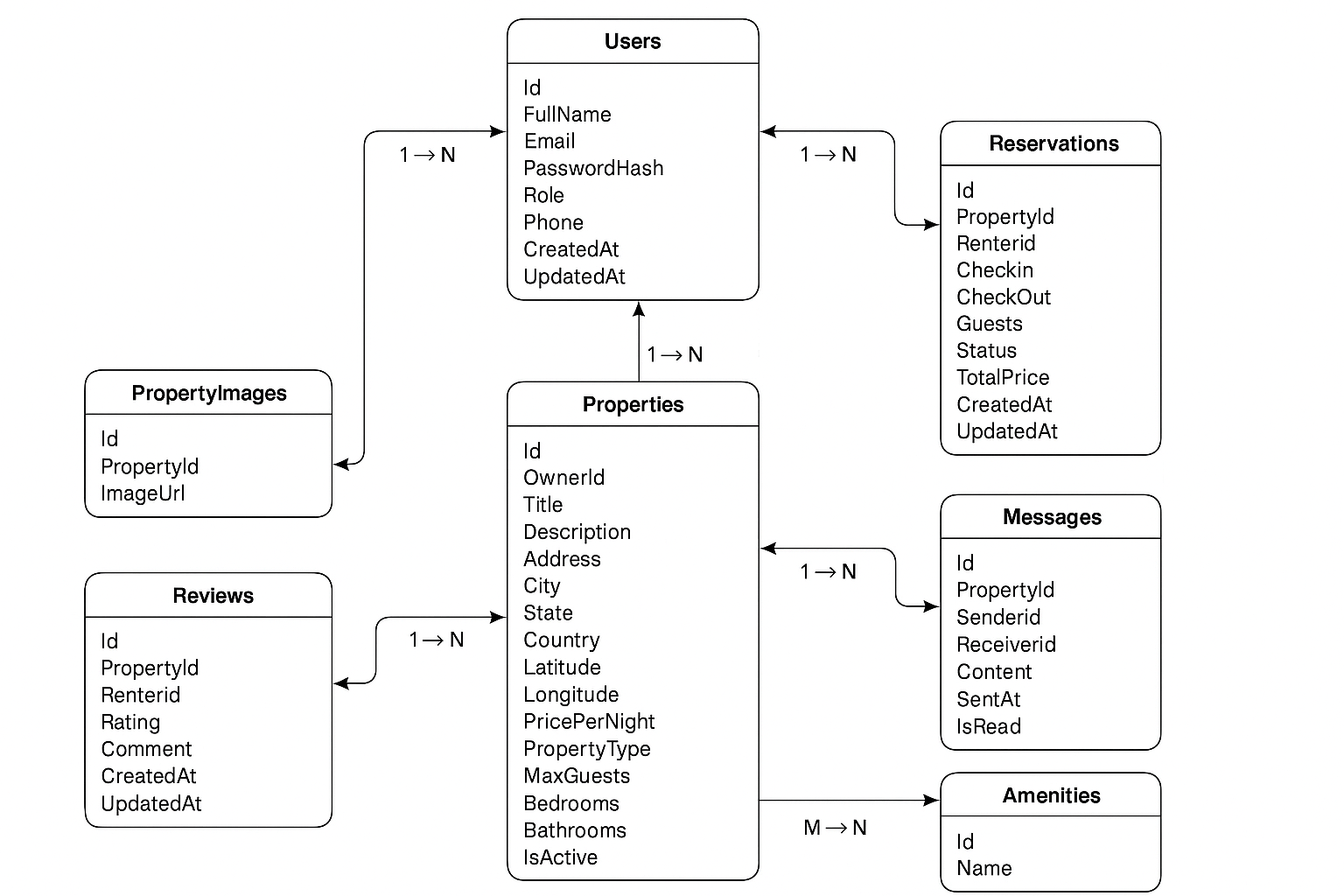


# Database Design & Storage Optimization

## The project uses SQL Server with the following major entities:

- Users: Stores renter/owner details with roles  
- Properties: Contains property details  
- Amenities & PropertyAmenities: Many-to-many relationship for property features  
- PropertyImages: Stores image URLs  
- Reservations: Tracks bookings and statuses  
- Messages: Communication between renter and owner  
- Reviews: Ratings and feedback for properties

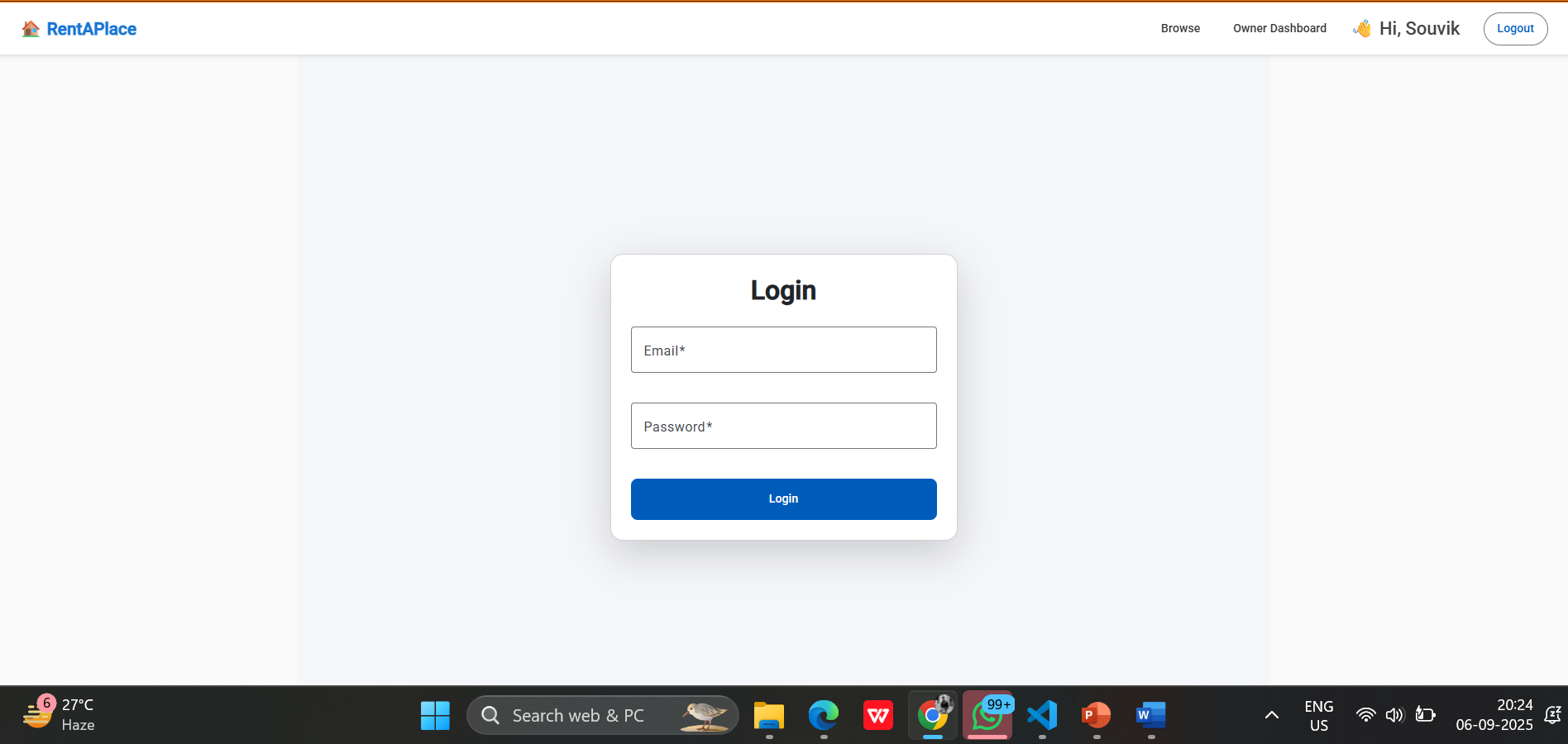
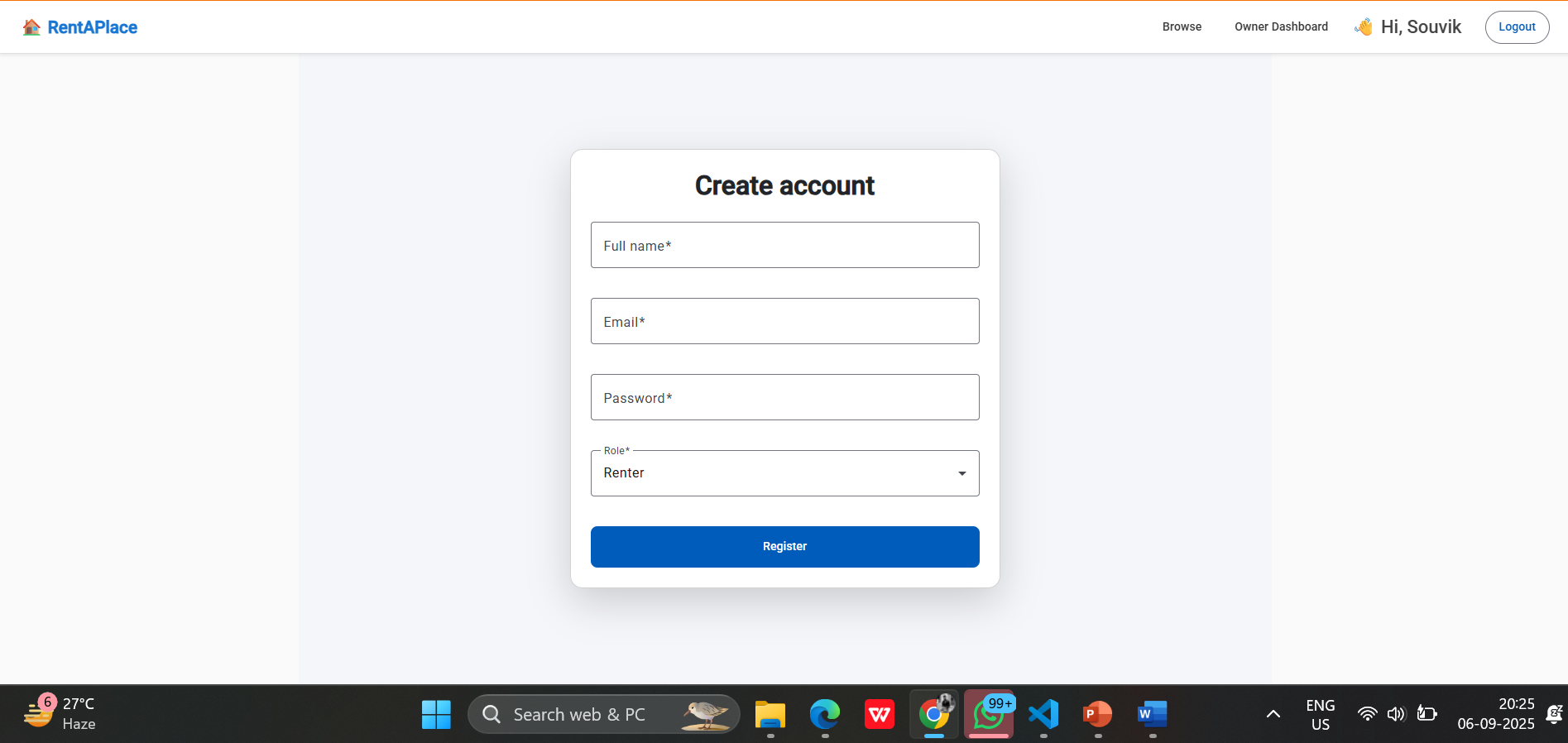
## Entity Relationship Diagram:



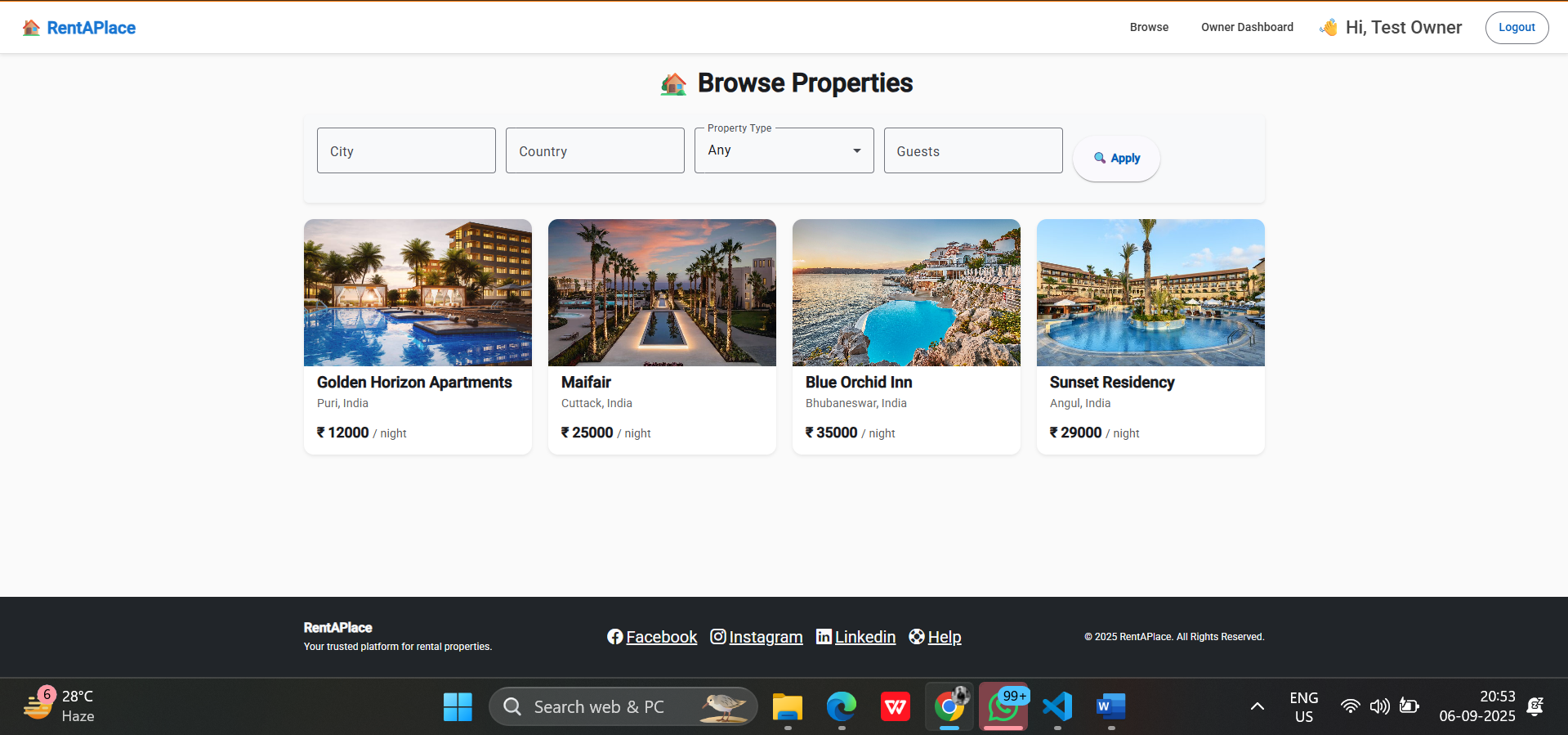
# Screenshots

## Insert screenshots for the following sections:

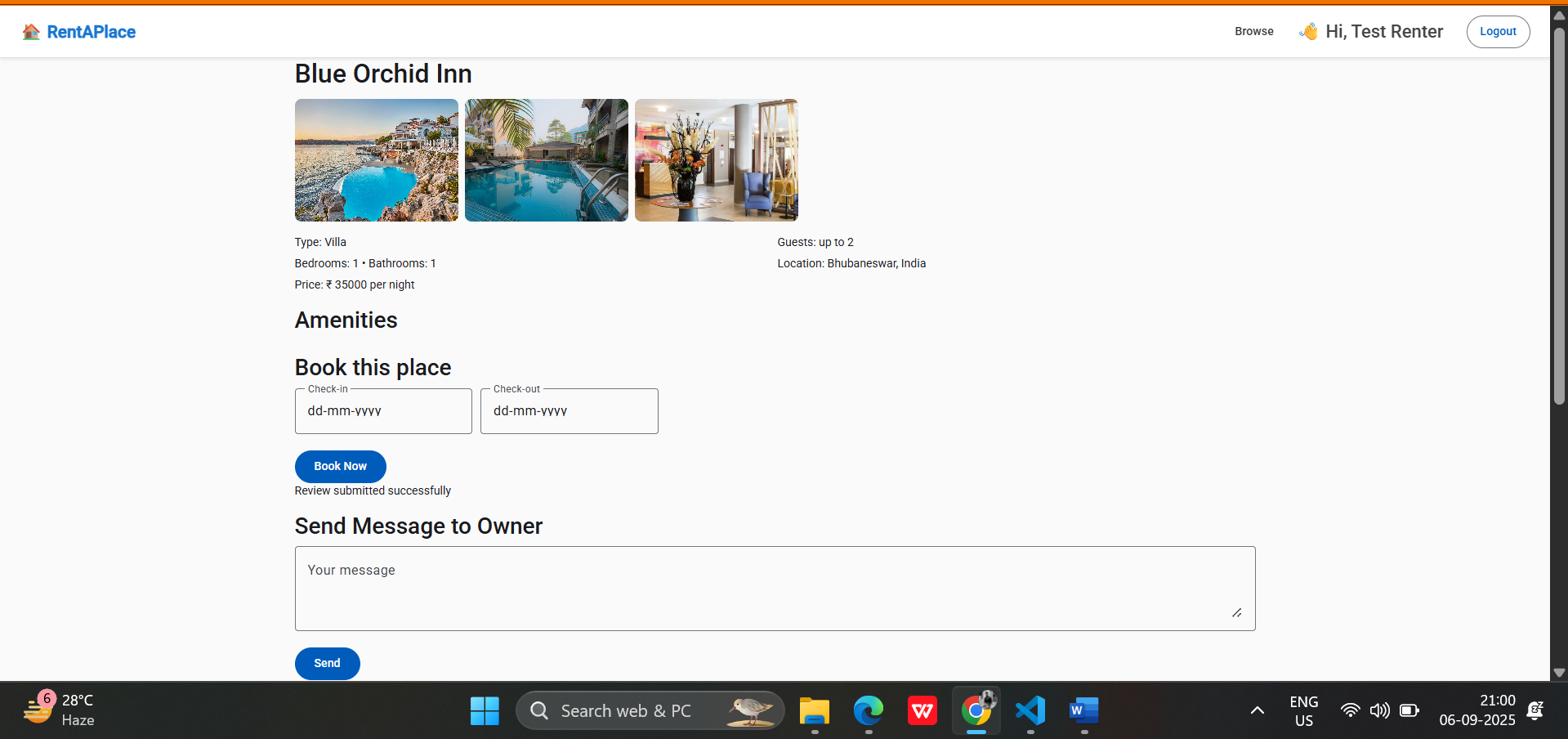
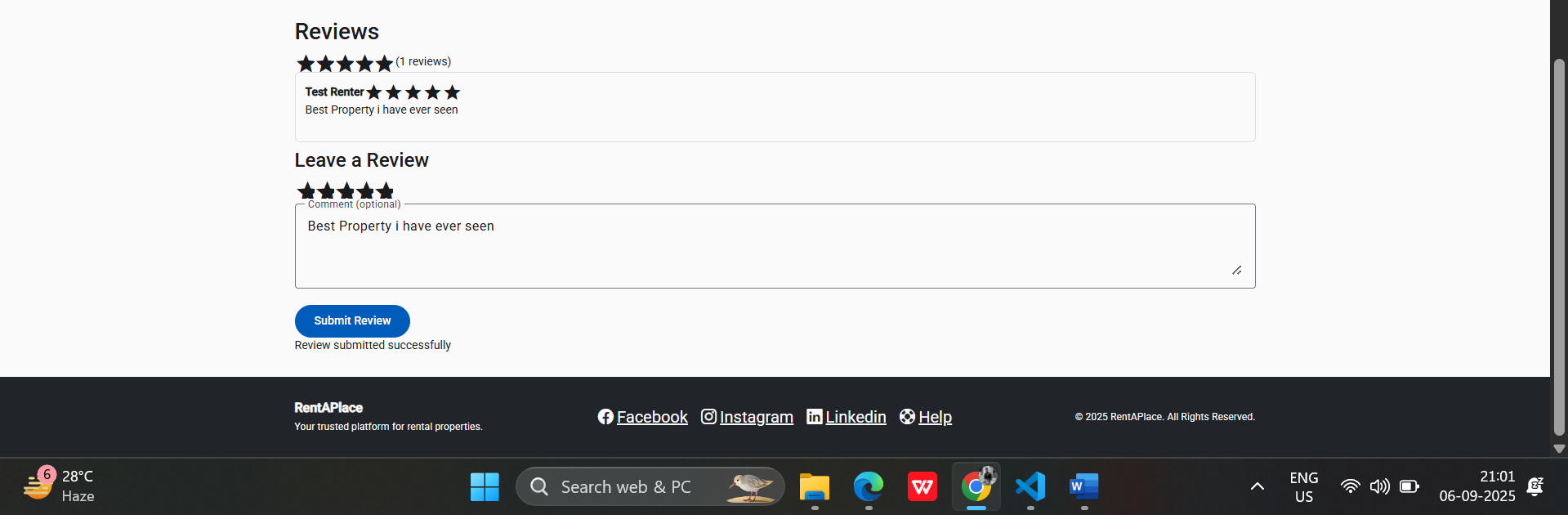
- Login/Register Page

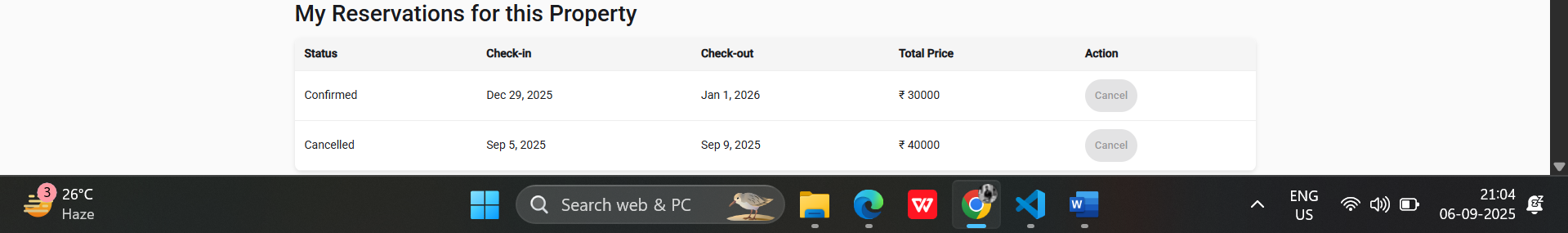
- Property Listing/Search Page



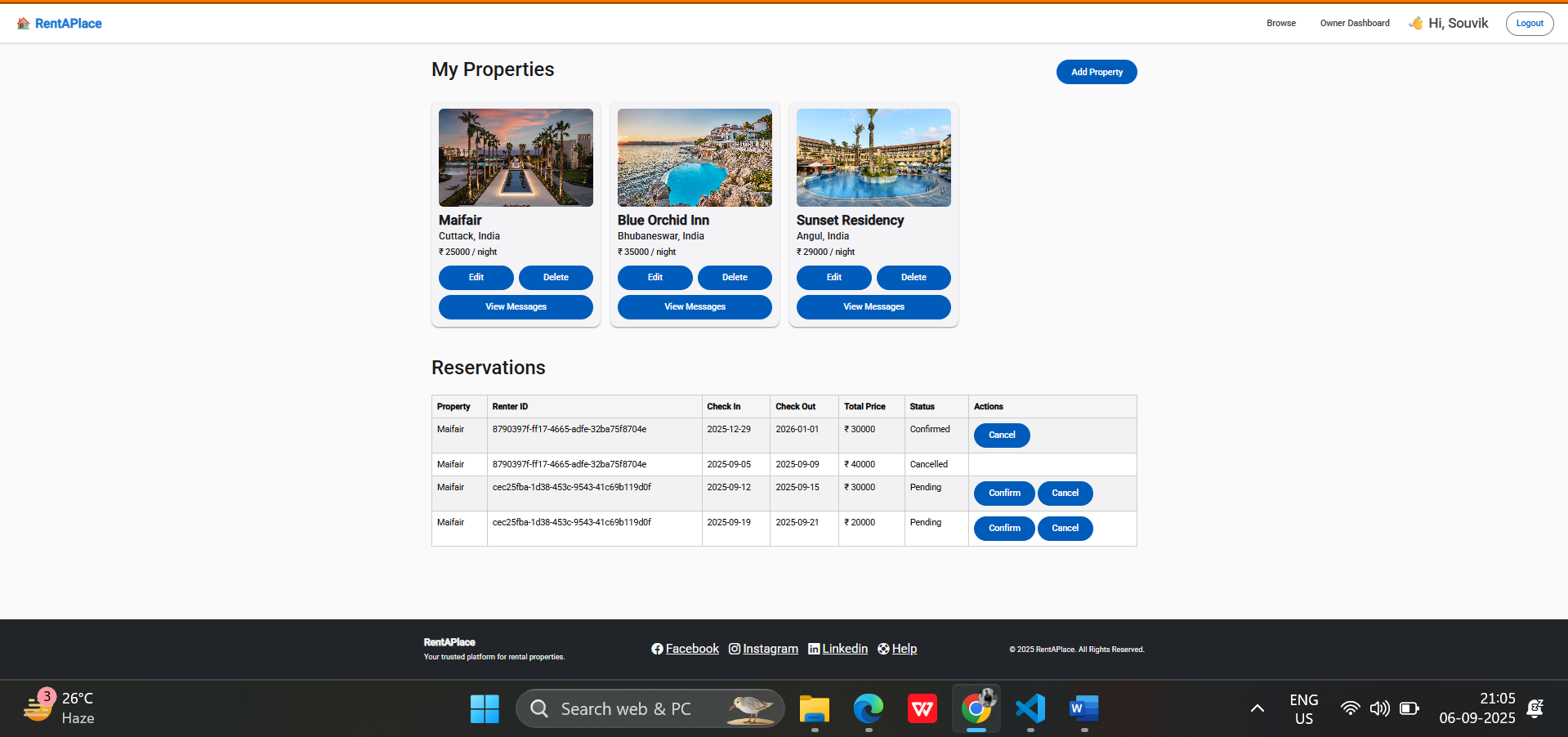
- Property Details Page

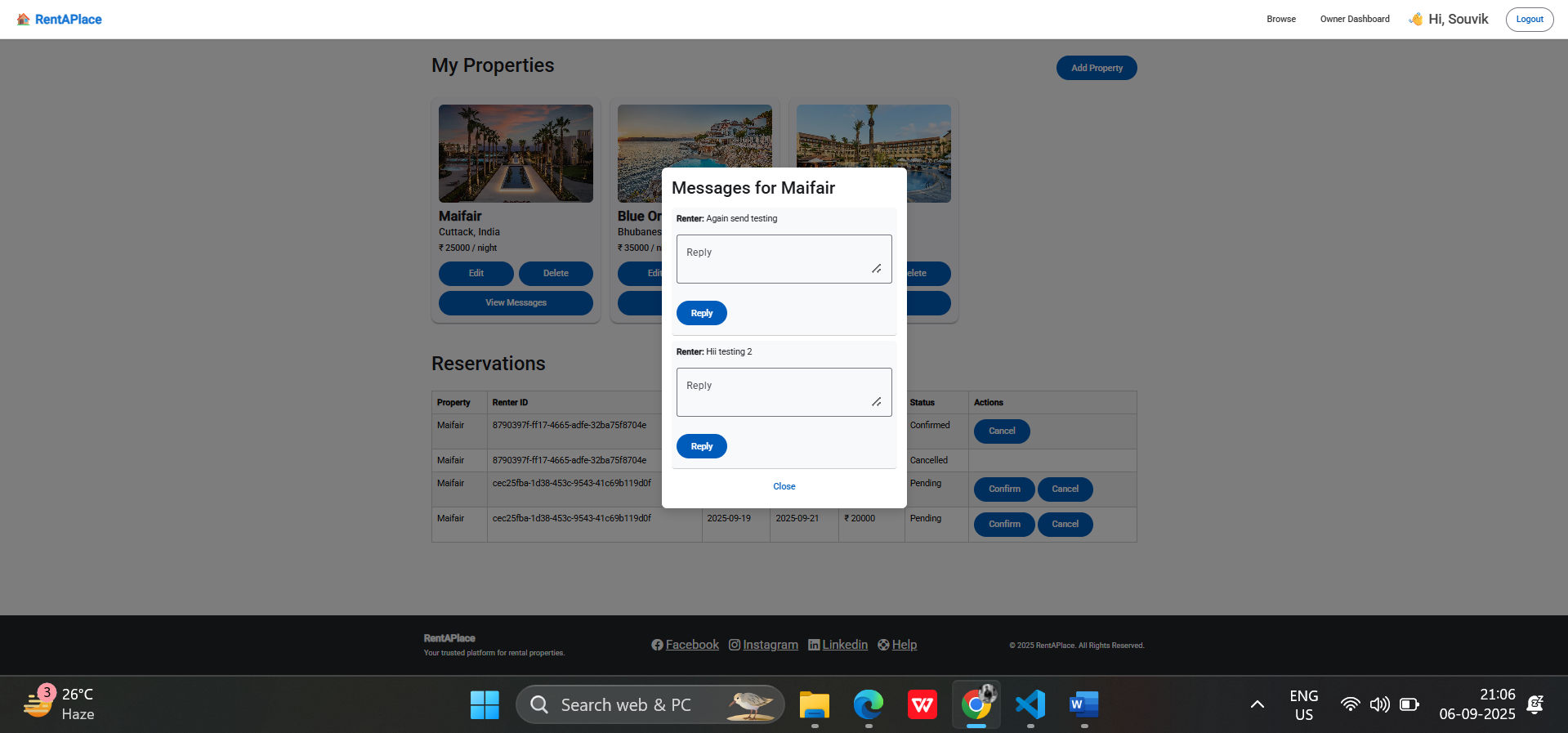
  


- Reservation Page



- Owner Dashboard

  
- Messaging System



# Deployment

The project will be deployed on GitHub (private repository).

[Insert GitHub Repository Link Here]

# Conclusion

The RentAPlace project successfully demonstrates a full-stack application built with Angular, .NET Core Web API, and SQL Server. It covers essential features such as authentication, property management, reservations, messaging, and reviews. The system ensures scalability, security, and usability for both renters and property owners.