Software Engineering (Sessional) Final Report

Course: CSE-434

RentSpot: Home Near CUET (Web Application)

Team Members

Mahshar Yahan
ID: 1804007

Tanzim Rahman
ID: 1804015

Md. Shimul Mahmud
ID: 1804021

Department of Computer Science and Engineering (CSE)
Chittagong University of Engineering & Technology (CUET)
Chattogram- 4349, Bangladesh

Executive Summary

The HomeNearCUET platform represents a paradigm shift in the student housing landscape, offering a revolutionary solution designed to address the evolving needs and preferences of students and homeowners near the CUET campus. Through a relentless pursuit of excellence, innovation, and user-centric design, our platform sets a new standard for convenience, accessibility, and engagement in the housing search process. With a comprehensive suite of advanced features and functionalities meticulously tailored to cater to the unique requirements of both students and homeowners, HomeNearCUET empowers students to seamlessly discover, evaluate, and secure accommodation options that meet their specific criteria and preferences, thereby enhancing their overall university experience. Concurrently, homeowners benefit from a robust set of property management tools that streamline the listing process, optimize occupancy rates, and facilitate seamless communication with potential renters, ultimately driving revenue growth and operational efficiency. By fostering a culture of transparency, collaboration, and innovation, HomeNearCUET not only simplifies the housing process but also fosters a sense of community, belonging, and mutual support within the CUET ecosystem. As a catalyst for positive change, HomeNearCUET is poised to revolutionize the student housing experience, ushering in a new era of connectivity, efficiency, and satisfaction for students and homeowners alike. Here, is the GitHub link of our project `https://github.com/MdShimulMahmud/software-engineering-lab`.

Contents

List of Figures

Introduction

The HomeRent System tailored for our university community stands as an integrated solution to facilitate seamless home rental services for students and homeowners alike. Our system encompasses key modules such as User Management, Property Listing, Booking Management, and Task Control. These modules collectively offer an efficient mechanism for students to find suitable rental homes, connect with homeowners, and manage their rental experience effectively.

Our technology-driven platform empowers students to effortlessly browse and book rental properties while providing homeowners the means to list their properties and manage bookings. The user-friendly interface ensures a smooth experience, allowing users to submit and track service requests related to their rented homes, covering aspects such as maintenance, cleaning, and utilities.

Administrators have the capability to oversee property listings, update information, and ensure the platform's adaptability to evolving needs. The Task Control module plays a pivotal role in managing the lifecycle of service requests, fostering effective communication between students, homeowners, and service providers.

Accessible through web browsers on various devices, our HomeRent System becomes an integral part of the daily lives of both students and homeowners. Prioritizing user interactions, the system contributes to a more organized and supportive living environment within our university community. Beyond simplifying the rental process, our platform aims to cultivate a harmonious atmosphere, akin to a collaborative community promoting shared resources and optimized living experiences.

1.1 Goals and Objectives of the project

The primary goal of the HomeRent System is to optimize and elevate the management of home rental services, establishing a user-friendly platform for students to effortlessly submit property requests, monitor their progress, and obtain prompt resolutions. Our

system strives to enhance communication channels and streamline task management between students, homeowners, and service providers, ultimately fostering a well-organized and supportive living environment for the university community.

1.2 Scope of the work

The HomeRent System encompasses a range of modules to establish a comprehensive and efficient platform for managing home rental services. The User Management Module is responsible for creating, authenticating, and maintaining user profiles, distinguishing between roles such as Homeowner, Student, and Administrator. The Property Listing Module empowers users to list and manage rental properties, facilitating a smooth process for both students and homeowners. Additionally, the Booking Management Module enables users to submit, track, and manage property booking requests through an intuitive dashboard.

The system caters to diverse rental concerns, including property maintenance, cleaning, utility management, and booking services. Meanwhile, the Admin Dashboard allows administrators to dynamically manage property listings, user profiles, and resolve issues effectively.

The Complaint Category Management Module allows administrators to create, update, and delete service categories, ensuring an adaptable system that can evolve with changing needs. The Task Control and Tracking Module orchestrate the entire lifecycle of service requests, promoting seamless communication and task management between users and administrators for efficient issue resolution and real-time tracking.

Together, these modules enhance the overall functionality of the HomeRent System, providing a user-friendly and organized approach to addressing diverse concerns within the home rental community.

1.3 System overview

The *HomeRent System* is a robust platform tailored for streamlined home rental service management. It comprises three primary panels—*Homeowner*, *Student*, and *Administrator*—ensuring efficient coordination and swift issue resolution.

1.3.1 Homeowner Panel

- **Central Control Hub:** Overseeing all activities within the platform.
- **Property Management:** Homeowners can create and manage property listings.

- **Booking Approval:** Review and approve student booking requests and manage property availability.
- **Task Assignment:** Assign tasks to service providers based on availability, skills, and workload.

1.3.2 Student Panel

- **Booking Request Submission:** Students can submit property booking requests with details and urgency levels.
- **Notification System:** Receive alerts for booking approval and property availability updates.

1.3.3 Service Provider Panel

- **Task List:** View assigned tasks with details and priority.
- **Task Completion:** Confirm task completion with notes or comments.
- **Record Keeping:** Administrators maintain task records and generate reports for efficient tracking.

This system overview ensures a cohesive and user-friendly experience for all stakeholders involved in the home rental process, promoting effective communication and task management.

Executive Summary

The HomeNearCUET platform represents a paradigm shift in the student housing landscape, offering a revolutionary solution designed to address the evolving needs and preferences of students and homeowners near the CUET campus. Through a relentless pursuit of excellence, innovation, and user-centric design, our platform sets a new standard for convenience, accessibility, and engagement in the housing search process. With a comprehensive suite of advanced features and functionalities meticulously tailored to cater to the unique requirements of both students and homeowners, HomeNearCUET empowers students to seamlessly discover, evaluate, and secure accommodation options that meet their specific criteria and preferences, thereby enhancing their overall university experience. Concurrently, homeowners benefit from a robust set of property management tools that streamline the listing process, optimize occupancy rates, and facilitate seamless communication with potential renters, ultimately driving revenue growth and operational efficiency. By fostering a culture of transparency, collaboration, and innovation, HomeNearCUET not only simplifies the housing process but also fosters a sense of community, belonging, and mutual support within the CUET ecosystem. As a catalyst for positive change, HomeNearCUET is poised to revolutionize the student housing experience, ushering in a new era of connectivity, efficiency, and satisfaction for students and homeowners alike. Here, is the GitHub link of our project `https://github.com/MdShimulMahmud/software-engineering-lab`.

Project Management Plan

2.1 Project Organization

Efficient project organization plays a vital role in software development, involving the distribution of specialized tasks among team members according to their strengths and capabilities. In a three-person team, the project is segmented among members, taking into account the time needed for each stage and leveraging the unique strengths of each individual. The objective is to establish a pragmatic organization that optimizes the likelihood of project success.

2.1.1 Individual Contribution to the project

Individual Contribution to the project are shown in Table-2.1

Member Name	Requirement Specification	Planning	Designing	Front End	Integration	Back End	Testing
Mahshar Yahan	✓	✓	✓	✓			✓
Tanzim Rahman	✓	✓	✓		✓		✓
Md. Shimul Mahmud	✓	✓	✓			✓	✓

Table 2.1: Individual Contribution to the project

2.2 Process Model used

In the development of our home rental system, we employ the Prototype Process Model to ensure agility, rapid iteration, and refinement of the application. This paradigm enables

quick planning, iterative development, and constant feedback from stakeholders, ensuring a user-centric prototype.

- Communication
- Quick plan
- Quick Design
- Construction of Prototype
- Deployment and Feedback

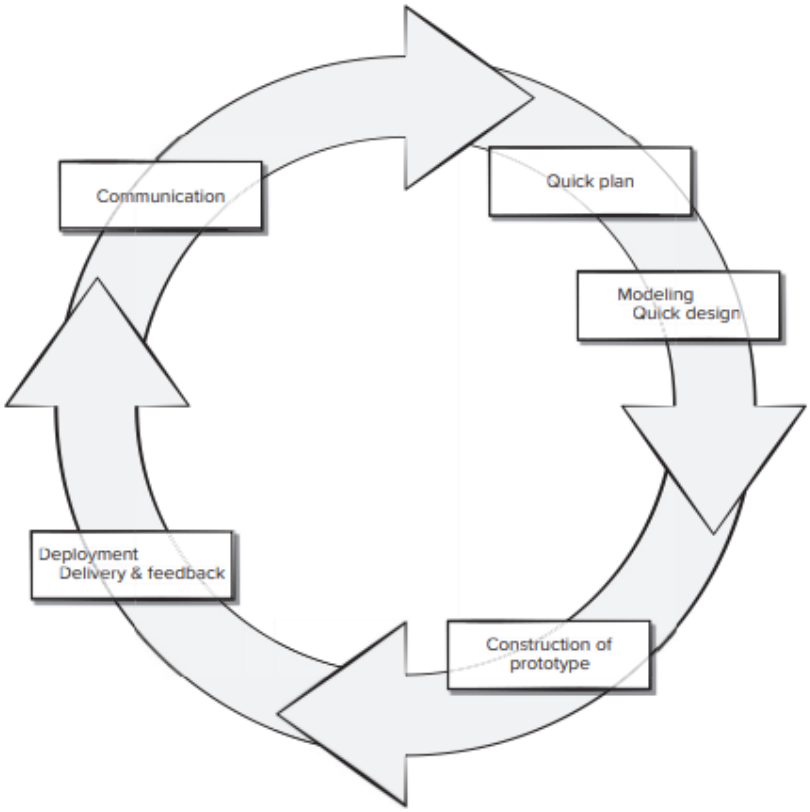

Figure 2.1: Prototyping Process Model

By using the prototype process model, the development of our home rental system remains responsive to evolving requirements, and iterative feedback from users makes this model closely aligned with our project idea.

2.2.1 Rationale for choosing lifecycle model

The selection of the Prototype Process Model for our home rental system development is grounded in several key considerations:

- **Iterative Development:** Our home rental system embraces prototyping, allowing flexibility to adapt and refine based on evolving requirements.
- **User-Centric Design:** Prioritizing an optimal user experience by engaging potential users early, ensuring an intuitive and user-friendly interface.
- **Feedback and Involvement:** Regular stakeholder feedback, including students, homeowners, and administrators, ensures active involvement and aligns the system with diverse user needs.
- **Flexibility to Changes:** Adaptable to dynamic market trends and user feedback, the prototype process model accommodates changes effectively.
- **Continuous Improvement:** The iterative nature promotes continuous enhancement, allowing seamless incorporation of new features and optimizations based on lessons learned.
- **Risk Mitigation:** Early user feedback in the prototyping phase helps effectively mitigate risks related to misunderstandings, unclear requirements, or changes in stakeholder expectations.

2.3 Risk Analysis

The risk analysis for the Home Rental Service project involves a comprehensive process encompassing risk identification, projection, refinement, and mitigation. In risk identification, potential threats such as scalability concerns, regulatory changes, and stakeholder engagement issues are systematically outlined. Risk projection evaluates the likelihood and consequences of each identified risk, facilitating effective resource allocation. The risk refinement process transforms broad risks into detailed scenarios for more targeted mitigation. A risk table categorizes identified risks, assigning probabilities and impact values.

Risk	Category	Probability (%)	Impact
Size may be small	PS	50	2
Larger number of users than planned	PS	60	1
Technology will not meet expectations	TR	10	3
Lack of experience	ST	30	2
Less users than planned	PS	60	3
Customer will change requirements	PS	80	2
Lack of maintenance	BU	60	2

Table 2.2: Developing a Risk Table

Risk mitigation strategies for scalability, data security, regulatory compliance, and technological dependencies are outlined. Risk monitoring involves periodic reviews and real-time monitoring, while risk management includes prioritization, response planning, and continuous improvement through regular retrospectives. This comprehensive approach ensures a focused and adaptive risk management strategy throughout the project's lifecycle.

2.4 Constraints to project implementation

(Schedule, Budget, Software & Hardware constraints)

- **Schedule Constraints:** Creating the CUET Home Rental System website involves making improvements through several rounds of testing, which might take extra time.
- **Hardware Constraints:** For the website to work well, we need good computer power. If our computers can't handle it, the website might be slow and not work properly.
- **Budget Constraints:** Making the CUET Home Rental System website fast and reliable means spending money on powerful servers. But, this might lead to budget issues because good servers can be expensive. So, we need to plan our budget carefully.

2.5 Hardware and Software Resource (Tools/Language) Requirements

1. **Software Requirements**:

- Development Environment: Node.js, React.js, Express.js
- Version Control: Git
- IDE: VS code
- Database: PostgreSQL
- API Documentation: Swagger

2. **Hardware Requirements**: We have moderate-performance machines with semi-sufficient RAM, decent processor and SSD storage is recommended for faster build and compilation times.

2.6 Project Timeline and Schedule

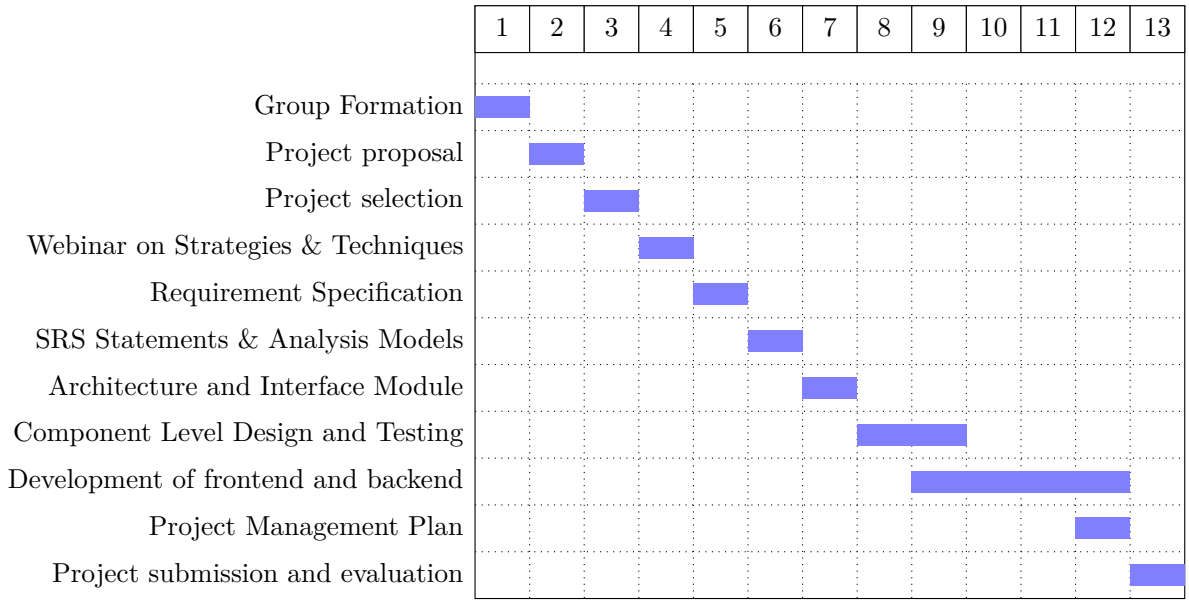

Figure 2.2: Project Gantt Chart

2.7 Estimated Budget

The estimated budget for the RentSpot: Home near CUET website is a comprehensive evaluation, taking into consideration various factors to ensure the successful development and deployment of the application within the specified scope and requirements. The following sections outline key aspects influencing the budget for Home Rental Service website:

- **Type of Software Project:**
  - RentSpot: Home near CUET website is categorized as a property rental platform project.

- The project involves creating a website for posting, booking, and managing rental properties.

- **Size of Software Project:**
  - The project size is considered medium, taking into account the people who are going to use it and the complexity associated with development.
  - As a medium-sized project, it includes features such as property listing, booking functionality and user profile management.

- **Development Team Size:**
  - The team size for RentSpot: Home near CUET website will be determined based on the specific requirements of the project.

- **Other Considerations:**
  - Additional factors, including the choice of technology stack, platform considerations (web development), and infrastructure requirements, will contribute to the overall budget estimation.

In our project there here are several mandatory roles and some optional roles. They are given in the following table.

Table 2.3: Estimated Budget Table

Area of Cost	**Cost (in TK)**
Laptop x 3	1,50,000
Resource Cost - Other Software	2,000
Server	2,000
UI/UX Designer	3,000
Developer Cost - Developer x 3	60,000
Documentation and Others	3000
Total	2,20,000

2.8 Social/Cultural/Environmental impact of the project

1. **Social Impact:**
   - Convenience and Accessibility: RENTSPOT provides easy access to a variety

of rental properties, catering to diverse housing preferences, enhancing convenience for users.

- Income Opportunities: Creates economic opportunities for property owners, landlords, and real estate professionals, contributing to income sources within the real estate industry.
- Community Well-being: Fosters community well-being by connecting property seekers with suitable homes, contributing to a sense of belonging and stability within the CUET community.

2. **Cultural Impact:**
    - Housing Diversity: Introduces users to a variety of housing options, promoting diversity in residential living experiences.
    - Preservation of Local Housing Styles: Plays a role in preserving and promoting regional architectural traditions by featuring local housing styles and preferences.
3. **Environmental Impact:**
    - Reduced Commuting and Emissions: Contributes to reducing the need for extensive property visits, leading to decreased commuting and lower carbon emissions associated with property hunting.
    - Sustainable Practices: Encourages property owners to adopt sustainable practices, minimizing the environmental impact of residential properties listed on the platform.
    - Waste Reduction: Minimizes paper-based processes, promotes digital transactions, and encourages eco-friendly property management practices, contributing to overall waste reduction.

Requirement Specifications

3.1 Stakeholders for the system

Students

- Primary users of the HomeNearCUET platform, seeking suitable accommodation options near the CUET campus.
- Require easy access to property listings, reservation functionality, and communication channels to connect with homeowners.

Homeowners

- Property owners offering accommodation options to students near the CUET campus.
- Utilize the platform to list their properties, manage bookings, and communicate with potential renters.

Web Developers

- Responsible for the development, maintenance, and enhancement of the HomeNearCUET platform.
- Ensure the platform's functionality, security, and scalability to meet the needs of stakeholders.

UI/UX Designers

- Design and optimize the user interface and experience of the HomeNearCUET platform.
- Focus on creating intuitive navigation, visually appealing designs, and seamless interactions to enhance user satisfaction and engagement.

3.2 Software Requirement Specification (SRS)

3.2.1 SRS: User Profile Management in Home Rental System

Introduction

Purpose

The purpose of this document is to delineate the prerequisites for the "User Profile Management" module within the system. This module is responsible for facilitating the management of user profiles for students (customers) and landowners.

Scope

The "User Profile Management" module will serve as a vital component of the system, allowing users to update personal information, preferences, and security settings within their profiles.

Overall Description

Product Perspective

The "User Profile Management" module interacts with other system modules like Authentication, Dashboard, and Database Management. It relies on a secure database to store user profile information.

User Classes and Characteristics

- Students (Customers): Users seeking rental properties.
- Landowners: Users owning properties available for rent.

Operating Environment

The system will be accessible via web browsers on PCs or mobile devices connected to the internet. It should support prevalent web technologies and database systems.

Functional Requirements

Profile Information Management

1. Users shall be able to view and modify personal information such as name, contact details, and profile picture.

2. The system shall allow users to update preferences like language, notification settings, and visibility of profile details.

Security Settings

1. Users should have the capability to change passwords and update authentication details.
2. The system must support secure transmission of user interactions.

Profile History

1. The system shall maintain a history log of profile updates for each user.
2. Users shall have access to review past changes made to their profiles.

Non-functional Requirements

Performance

- Profile management actions should have a response time of less than 3 seconds.
- The system should support concurrent profile updates without performance degradation.

Security

- User profile information must be encrypted and securely stored.
- All user interactions must be transmitted over HTTPS for data security.

Usability

- The user interface for profile management should be intuitive and easy to navigate.
- Clear and descriptive error messages should guide users during profile management.

Constraints

- The system must be developed using specific technologies like ReactJS, NodeJS, and SQL.
- Compliance with relevant data protection regulations and standards is mandatory.

Assumptions and Dependencies

- Users are assumed to have internet access and a compatible web browser for system interaction.
- Profile management is reliant on accurate information provided by users.

Future Enhancements

- Integration of additional security measures such as two-factor authentication for user accounts.
- Enhancements to allow users to track and revert to previous versions of their profiles.

Conclusion

This Software Requirements Specification outlines the requirements for the "User Profile Management" module in the system. Implementing these requirements will ensure an efficient and secure management system for user profiles.

3.2.2 SRS: Post Property for Rent Module in Home Rental System

Introduction

Purpose

The purpose of this document is to provide a detailed description of the requirements for the "Post Property for Rent" module of the Home Rental System. This module is responsible for allowing landowners to list their properties for rent on the website.

Scope

The "Post Property for Rent" module will be a crucial part of the Home Rental System, facilitating landowners in advertising and making their properties available for rent. It covers the property listing process, including details such as location, rent amount etc.

Overall Description

Product Perspective

The "Post Property for Rent" module interacts closely with other modules of the Home Rental System, including the User Management, Search, and Negotiation modules. It relies on a database to store property information securely.

User Classes and Characteristics

- Landowners: Users who own properties and want to list them for rent.

Operating Environment

The system will be web-based and accessible through standard web browsers. It should support common web technologies and databases.

Functional Requirements

Property Listing

1. The system shall provide a form for landowners to post a property for rent.
2. Landowners must provide details about the property, including location, rent amount, room features, etc.
3. The system shall store the posted property listing in the database.

Property Details Validation

1. The system shall validate the information provided by the landowner for the property listing.
2. If the information is incomplete or invalid, the system shall prompt the landowner to correct it.

View Property Listing

1. Users, including customers searching for rental properties, shall be able to view property listings.
2. The system shall display detailed information about each property, including location, rent amount, room features, etc.
3. The system shall provide a user-friendly interface for easy navigation through property listings.

Search and Filter Properties

1. The system shall allow customers to search for rental properties based on various criteria such as location, rent amount, and amenities.
2. customers shall be able to filter search results to find properties that meet their specific requirements.

Property Status

1. The system shall track the status of each property, indicating whether it is available for rent or not.
2. Landowners shall have the ability to update the status of their properties (e.g., available, rented).

Property Images

1. Landowners shall be able to upload images of the property during the listing process.
2. The system shall display these images along with the property details for users to visualize the rental space.

Notification System

1. The system shall notify landowners when there is user interest or inquiries about their listed property.
2. customers shall receive notifications regarding the status of their booking requests or negotiations.

Non-functional Requirements

Performance

- The system should support a large number of simultaneous property listings.
- Property listing processes should be completed within 5 seconds.

Security

- Property information must be securely stored.
- All user interactions must be transmitted over HTTPS.

Usability

- The user interface for property listing should be intuitive and user-friendly.
- Error messages should be clear and helpful.

Constraints

- The system must be developed using ReactJS, NodeJS, SQL.
- The system must comply with relevant data protection regulations and standards.

Assumptions and Dependencies

- The system assumes that landowners have access to the internet and a web browser.
- The system depends on the accuracy of property information provided by landowners.

Future Enhancements

- The system will allow landowners to add more detailed property features.
- The system will provide a preview of the property listing before submission.

*nConclusion This Software Requirements Specification outlines the detailed requirements for the "Post Property for Rent" module of the Home Rental System. The successful implementation of these requirements will ensure a seamless and efficient property listing experience for landowners on the platform.

3.2.3 SRS: Book Property for Rent Module in Home Rental System

Introduction

Purpose

The purpose of this document is to provide a detailed description of the requirements for the "Book a Property" module of the Home Rental System. This module is responsible for enabling customers to book properties for rent.

Scope

The "Book a Property" module will be an integral part of the Home Rental System, allowing customers to browse available properties and initiate the booking process. It covers property selection, confirmation, and notification processes.

Overall Description

Product Perspective

The "Book a Property" module interacts closely with other modules of the Home Rental System, including the User Management, Search, and Negotiation modules. It relies on a database to store booking information securely.

User Classes and Characteristics

- customers: Users who want to book properties for rent.

Operating Environment

The system will be web-based, accessible through standard web browsers. It should support common web technologies and databases.

Functional Requirements

Property Selection

1. Customers shall have option to initiate the booking process for a selected property.
2. The system shall display available properties based on the customer's search criteria.
3. customers shall be able to choose a property and proceed with the booking process.

Booking Confirmation

1. The system prompts customer for booking confirmation, including rental period.
2. Customers shall enter the required information for booking confirmation.
3. The system shall validate the information provided for completeness and accuracy.

Booking Process

1. Upon successful validation, the customer confirms the booking.
2. The system notifies the landowner of the successful booking and updates the status.
3. The system sends a confirmation email to the customer with details of the booked property.

Booking Cancellation

1. customers shall have the option to cancel a booking.
2. The system updates the booking status accordingly and notifies the landowner about the cancellation.

Booking Renewal

1. Users (customers) shall be able to renew the booking for an additional rental period.
2. The system shall prompt customers to confirm the renewal details.

3. The system updates the booking status and notifies the landowner about the renewed booking.

Booking History

1. The system shall maintain a history of bookings for each user.
2. Users (customers) shall be able to review past booking details for reference.

Booking Notification

1. The system shall send notifications to both customers and landowners for successful bookings, cancellations, and renewals.
2. Notifications shall include details such as booking confirmation, cancellation notices, and renewal confirmations.

Non-functional Requirements

Performance

- The system should support a large number of simultaneous booking processes.
- Booking processes should be completed within 5 seconds.

Security

- Booking information must be securely stored.
- All user interactions must be transmitted over HTTPS.

Usability

- The user interface for property selection and booking should be intuitive and user-friendly.
- Error messages should be clear and helpful.

Constraints

- The system must be developed using ReactJS, NodeJS, SQL.
- The system must comply with relevant data protection regulations and standards.

Assumptions and Dependencies

- The system assumes that customers have access to the internet and a web browser.
- The system depends on the accuracy of property information and availability provided by landowners.

Future Enhancements

- The system could be enhanced to allow customers to provide specific preferences during the booking process.
- The system could be enhanced to include a review and rating feature for booked properties.

Conclusion

This Software Requirements Specification outlines the detailed requirements for the "Book a Property" module of the Home Rental System. Implementing these requirements successfully will ensure a seamless and secure booking experience for customers using the platform.

3.3 Use case diagram with Graphical and Textual Description

3.3.1 Use Case diagram for User Profile Management

Textual Description of Use Case

User Profile Management

Use Case:

User Profile Management System

Iteration:

3

Primary Actors:

Student, Landowner

Goal in Context:

To provide a comprehensive platform for managing user profiles for students (customers) and landowners within the system.

Preconditions:

- The User Profile Management System is functional and accessible.
- Users (students and landowners) are registered with valid account credentials.

Trigger:

A user (student or landowner) wants to manage their profile information or preferences.

Scenario:

1. **User Authentication and Access:**
   - User accesses the User Profile Management System through login credentials.
   - The system validates and grants access to the user's profile management dashboard.
2. **Viewing and Editing Profile Information:**
   - Upon successful login, the user can:
     - View their current profile information (name, contact, profile picture, etc.).
     - Modify personal details like name, email, contact number, address, etc.
     - Upload or update their profile picture.
3. **Customization and Preferences:**
   - Users have the capability to:
     - Set preferences such as language, notification settings, theme, or layout preferences.
     - Customize visibility settings for specific profile details (public, private, etc.).
4. **Security and Authentication Settings:**
   - Users can manage security-related aspects:
     - Change passwords or update authentication information.
     - Configure two-factor authentication settings (if available).

Exceptions:

- System displays appropriate error messages in case of system errors or connectivity issues.
- Stringent measures in place to secure sensitive user data and prevent unauthorized access.

Priority:

Integral part of system functionality, ensuring an efficient and personalized user experience.

When Available:

Implemented as part of the system launch and continuously accessible for profile management.

Frequency of Use:

Frequent

Channels to Actors:

Accessible through web browsers on PCs or mobile devices with an internet connection.

Secondary Actors:

Server

Channels to Secondary Actors:

Via a web browser on a PC or mobile device using internet connection

Open Issues:

- Implementing robust security measures to safeguard user data.
- Designing an intuitive and user-friendly interface for seamless profile management.
- Developing a notification system for timely updates and changes in profile information.
- Establishing protocols for handling system errors or interruptions during profile management actions.

Graphical Description of Use Case

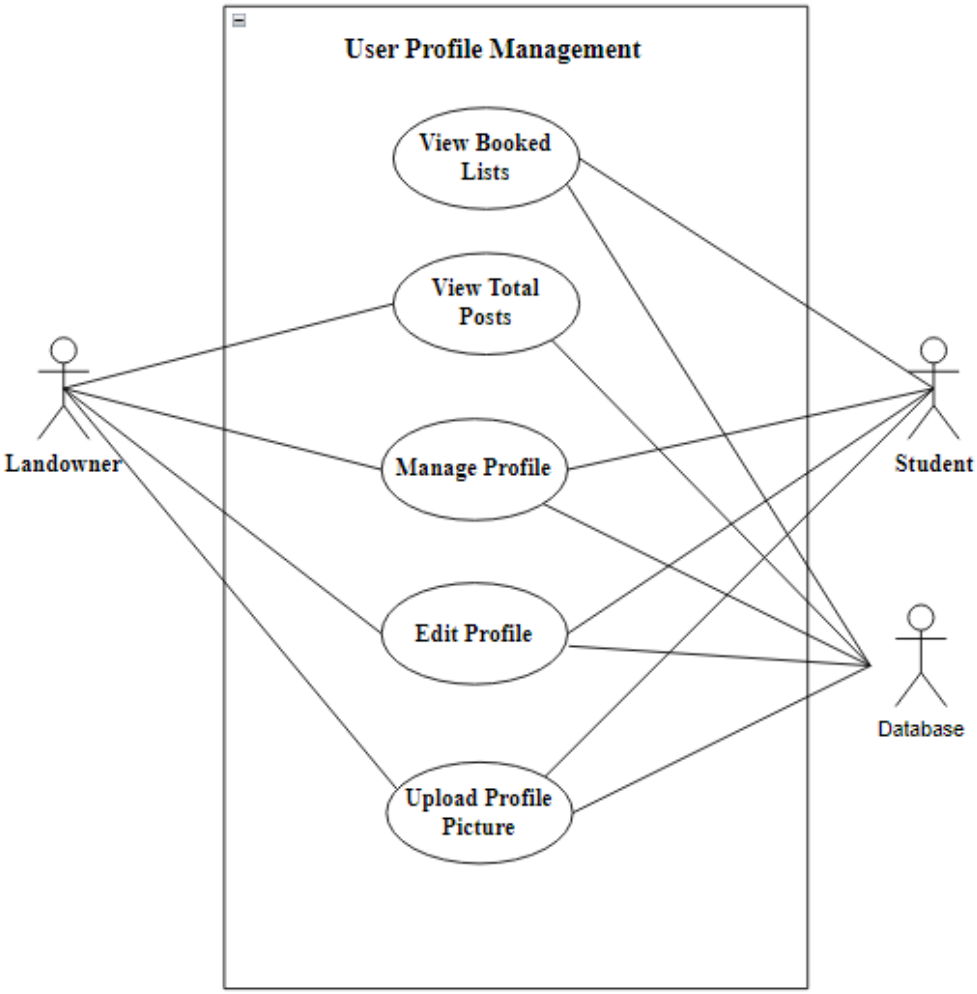

Figure 3.1: Use case diagram for user profile management.

3.3.2 Use Case diagram for Post Property

Textual Description of Use Case

Post Property for Rent

Use Case:

Post Property for Rent

Iteration:

1

Primary Actor:

Landowner

Goal in Context:

To create a new property listing and make it available for rent on the website.

Preconditions:

- The system must be fully configured.
- The landowner must have a registered account with appropriate user ID and password.

Trigger:

The landowner decides to list a property for rent.

Scenario:

1. The landowner navigates to the HomeRent website.
2. The landowner logs in using their user ID and password.
3. The system validates the credentials and displays the landowner's dashboard.
4. The landowner selects the option to "Post Property for Rent."
5. The system prompts the landowner to provide details about the property, including location, rent amount, room features, etc.
6. The landowner enters the required information.
7. The system validates the information provided.
8. The landowner confirms the property listing.
9. The system updates the database with the new property listing.

Exceptions:

- If the information provided is incomplete or invalid, the system prompts the landowner to correct it.
- If the landowner's credentials are incorrect, the system displays an error message and prompts for valid credentials.
- If there is a technical failure during the submission process, the system informs the landowner about the issue and advises them to try again later.

Priority:

High priority, to be implemented as part of the core functionality.

When Available:

First increment.

Frequency of Use:

Frequent

Channel to Actor:

Via a web browser on a PC or mobile device using internet connection

Secondary Actors:

Server

Channels to Secondary Actors:

Via a web browser on a PC or mobile device using internet connection

Open Issues:

- Security measures to prevent unauthorized access to the property listing functionality.
- Ensuring the security of the property details provided by the landowner.
- User interface considerations for an easy and intuitive property listing process.
- System scalability to handle a growing number of property listings.

Graphical Description of Use Case

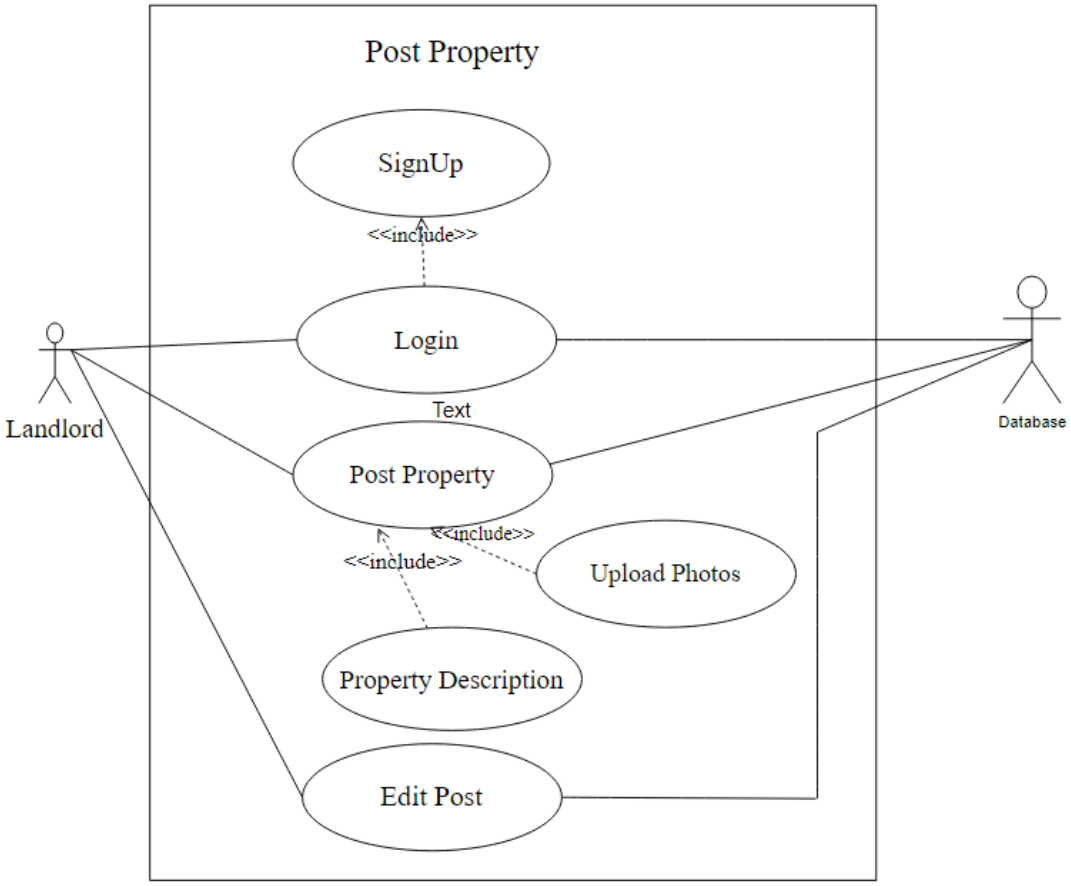

Figure 3.2: Use case diagram for Post Property.

3.3.3 Use Case diagram for Book Property

Textual Description of Use Case

Book a Property

Use Case:

Book a Property

Iteration:

2

Primary Actor:

Student

Goal in Context:

To enable a student to successfully book a property for rent.

Preconditions:

- The system must be fully configured.
- The student must have a registered account with appropriate user ID and password.
- A property listing must be available for booking.

Trigger:

The student decides to book a property for rent.

Scenario:

1. The student logs into the HomeRent website.
2. The student searches and selects a property available for rent.
3. The system displays the property details, including the option to "Book Now."
4. The student selects the "Book Now" option.
5. The system prompts the student to confirm booking details, including rental period.
6. The student enters the required information.
7. The system validates the information provided.
8. The student confirms the booking.
9. The system updates the booking status and notifies the landowner.
10. The system sends a confirmation to the student with details of the booked property.

Exceptions:

- If the booking confirmation is canceled by the student, the system updates the status accordingly.
- If the provided information is incomplete or invalid, the system prompts the student to correct it.
- If there is a system error or connection issue, an appropriate error message is displayed.

Priority:

Moderate priority, to be implemented as part of the core functionality.

When Available:

Second increment

Frequency of Use:

Frequent.

Channel to Actor:

Via a web browser on a PC or mobile device using internet connection.

Secondary Actors:

Landowner, Server

Channels to Secondary Actors:

Notification to the landowner upon successful booking.

Open Issues:

- Security measures to protect the booking process and payment information.
- User interface considerations for a seamless booking experience.
- Confirmation and notification system for timely updates after a successful booking.
- Handling system errors or interruptions during the booking process.

Graphical Description of Use Case

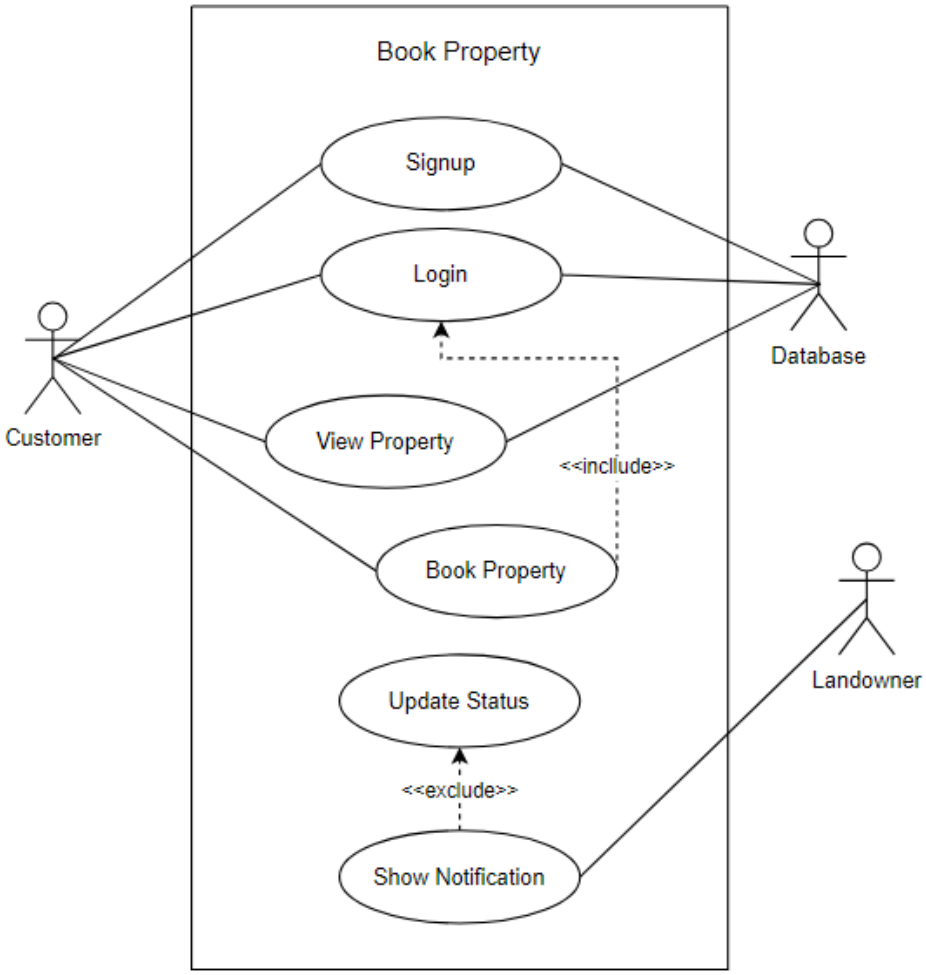

Figure 3.3: Use case diagram for Book Property.

3.4 Activity Diagram

3.4.1 Activity diagram for User Profile Management

The activity diagram for user profile management delineates steps from user authentication to role-based functionalities. Landowners manage profiles, update listings, and finalize deals, while students maintain profiles, search for properties, negotiate, and book.

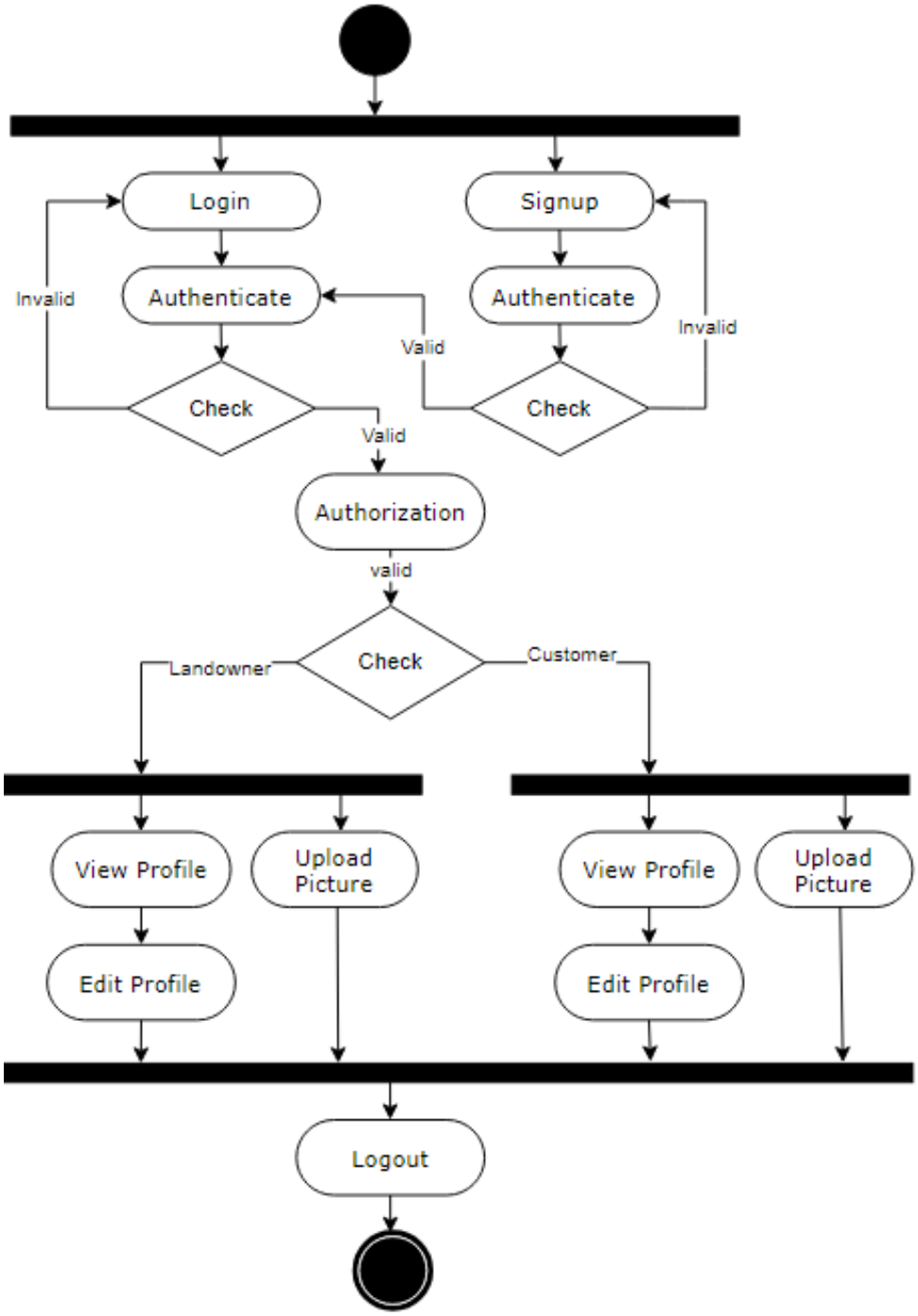

Figure 3.4: Activity diagram for user_profile() operation

3.4.2 Activity diagram for Post Property

The activity diagram for posting a property outlines the sequential steps involved in the process, from authentication to the completion of property listing. This diagram represents the activities performed by both landowners and students, capturing their distinct roles in the property posting process.

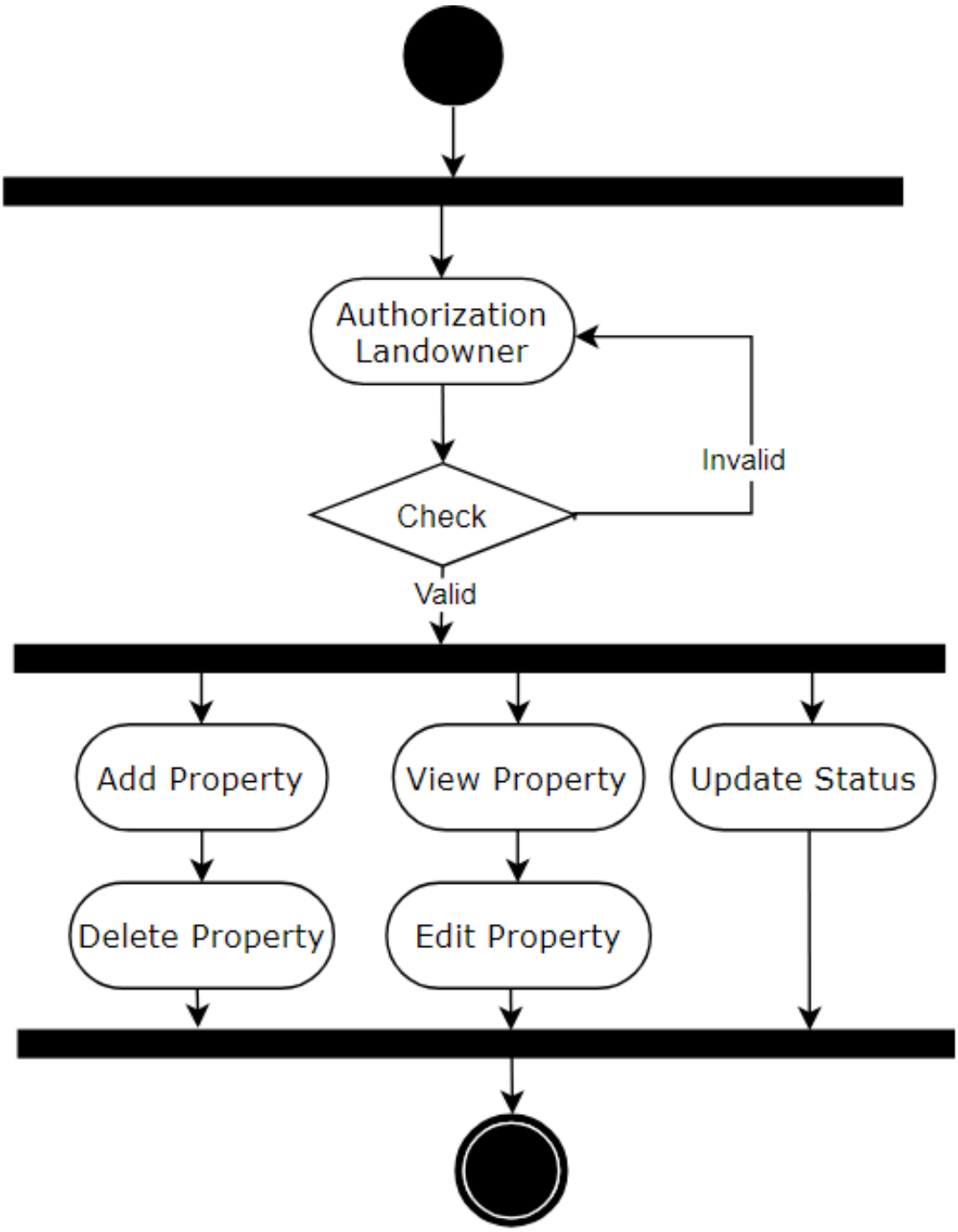

Figure 3.5: Activity diagram for post_property() operation

3.4.3 Activity diagram for Book Property

An activity diagram for ”Book Property” visually outlines the workflow involved in securing a rental. It demonstrates the user's activities: initiating a property search, selecting a listing, entering booking details, and proceeding to payment. Simultaneously, it illustrates system actions: verifying availability, processing payments, confirming reservations, and sending notifications. The diagram clarifies the sequential steps and decision points in the booking process.

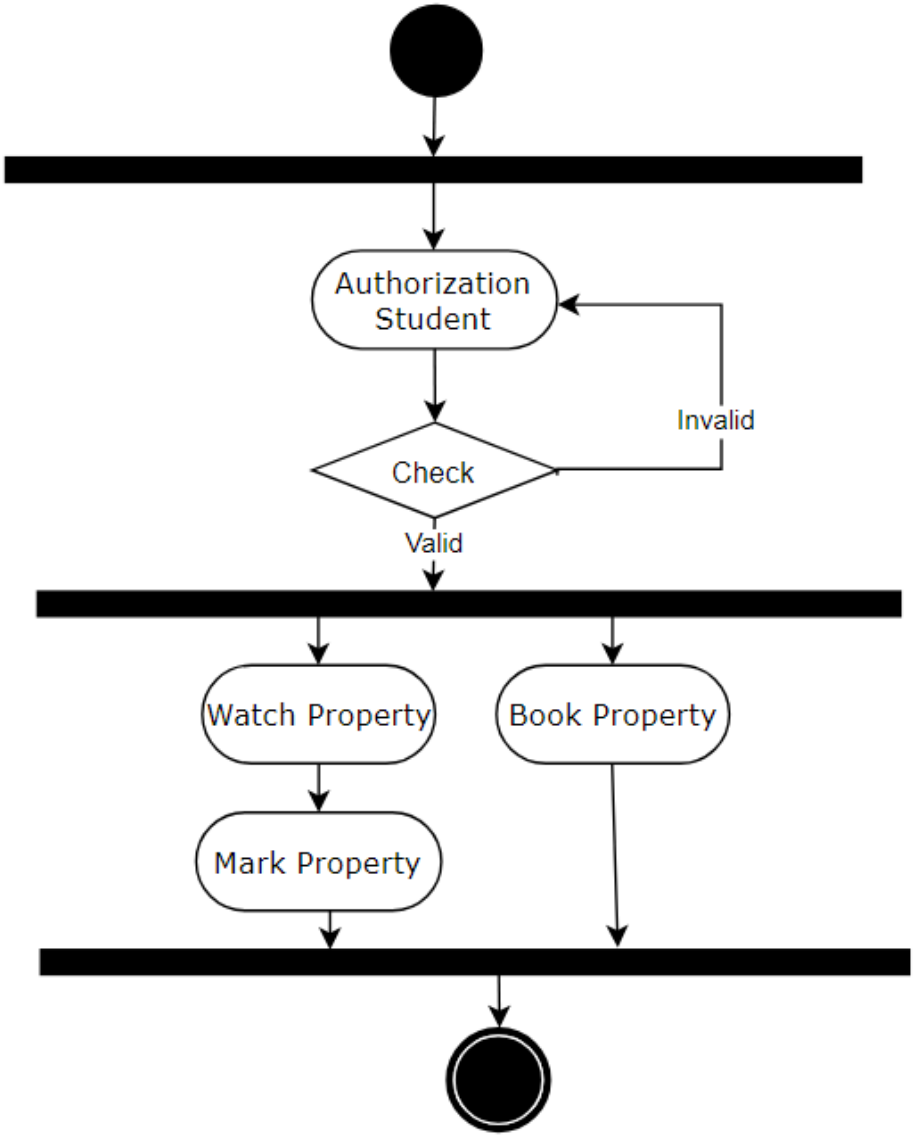

Figure 3.6: Activity diagram for book_property() operation

3.5 Dynamic model – sequence Diagram

3.5.1 Sequence diagram for User Profile Management

In the sequence diagram for user profile management, the process initiates with a user attempting authentication through login or sign-up. Upon successful authentication, the system verifies permissions and grants access to role-specific functionalities. Landowners engage in managing profiles, updating listings, and negotiating deals, while students access profiles, search properties, negotiate terms, and book. Throughout, the system facilitates secure communication and offers landowners tools for property management, ensuring a seamless user experience.

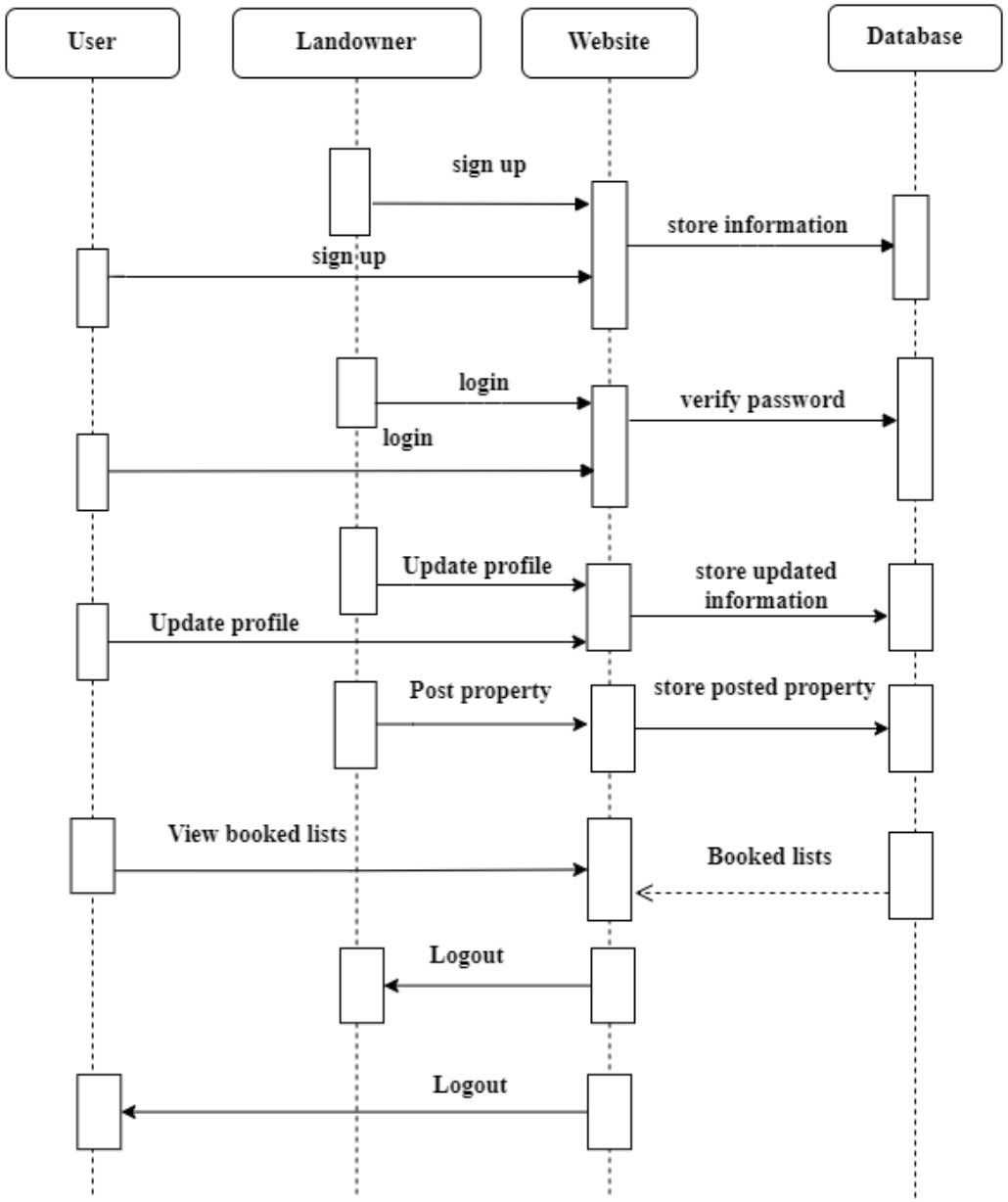

Figure 3.7: Sequence diagram for user profile management.

3.5.2 Sequence diagram for Post Property

The sequence diagram for posting a property illustrates the dynamic interactions between users, the authentication process, and the system functionalities involved in creating a property listing. The sequence begins with a user attempting authentication through login or sign-up and proceeds through role-specific actions for both landowners and students.

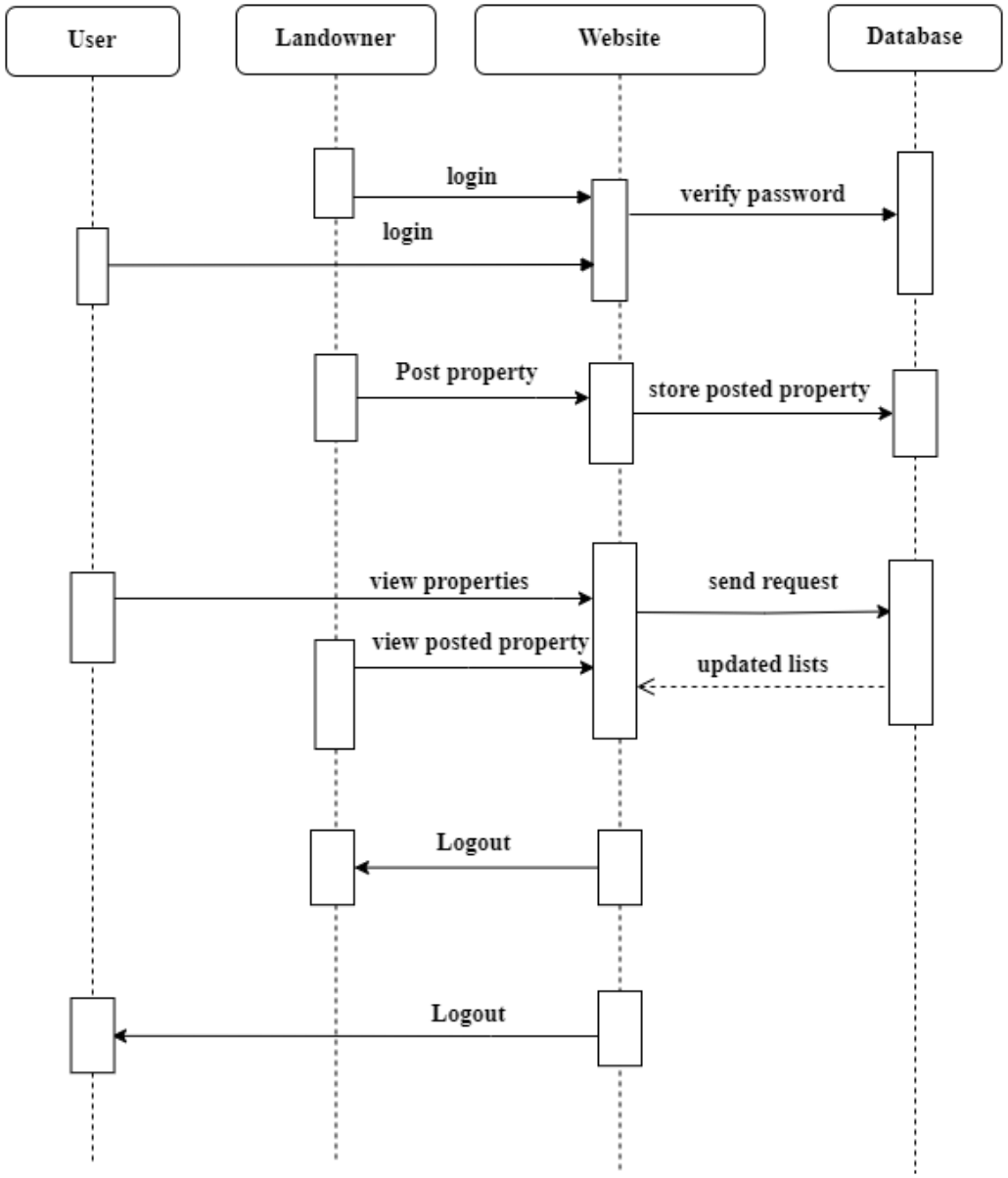

Figure 3.8: Sequence diagram for Post Property.

3.5.3 Sequence diagram for Book Property

A sequence diagram for "Book Property" showcases the chronological interaction between user actions and system components. It delineates the step-by-step process: The user searches for a property, selects it, submits booking details, the system checks availability, confirms the reservation, and notifies both user and the property owner. The diagram captures the dynamic exchange of messages and actions among these entities.

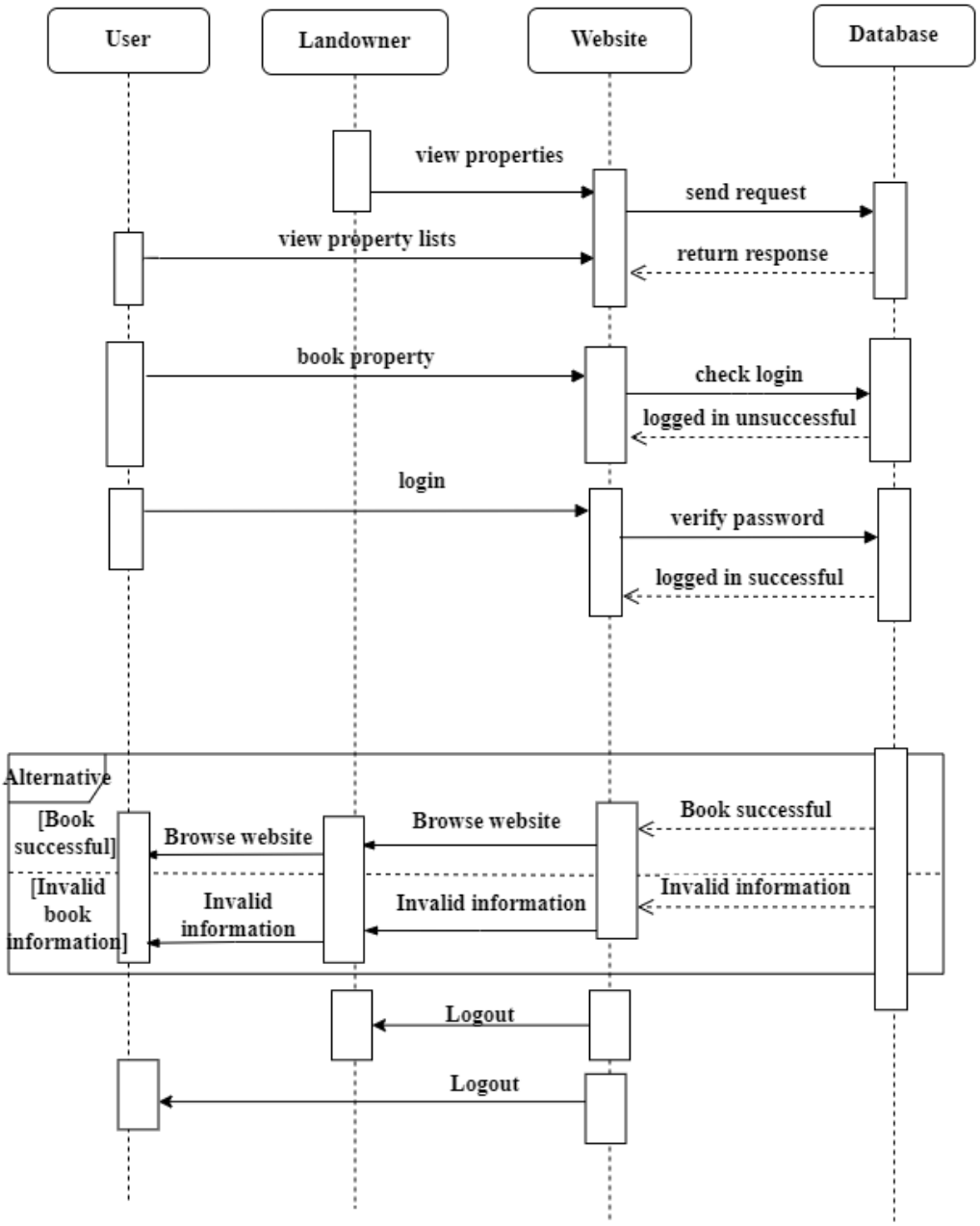

Figure 3.9: Sequence diagram for book Property.

3.6 Swimlane diagram

3.6.1 Swimlane diagram for User Profile Management

In the swimlane diagram for user profile management, distinct lanes represent users' roles: students and landowners. Landowners navigate activities such as profile management, listing updates, negotiation responses, and deal finalization. Meanwhile, students engage in profile maintenance, property searches, negotiation initiation, and booking.

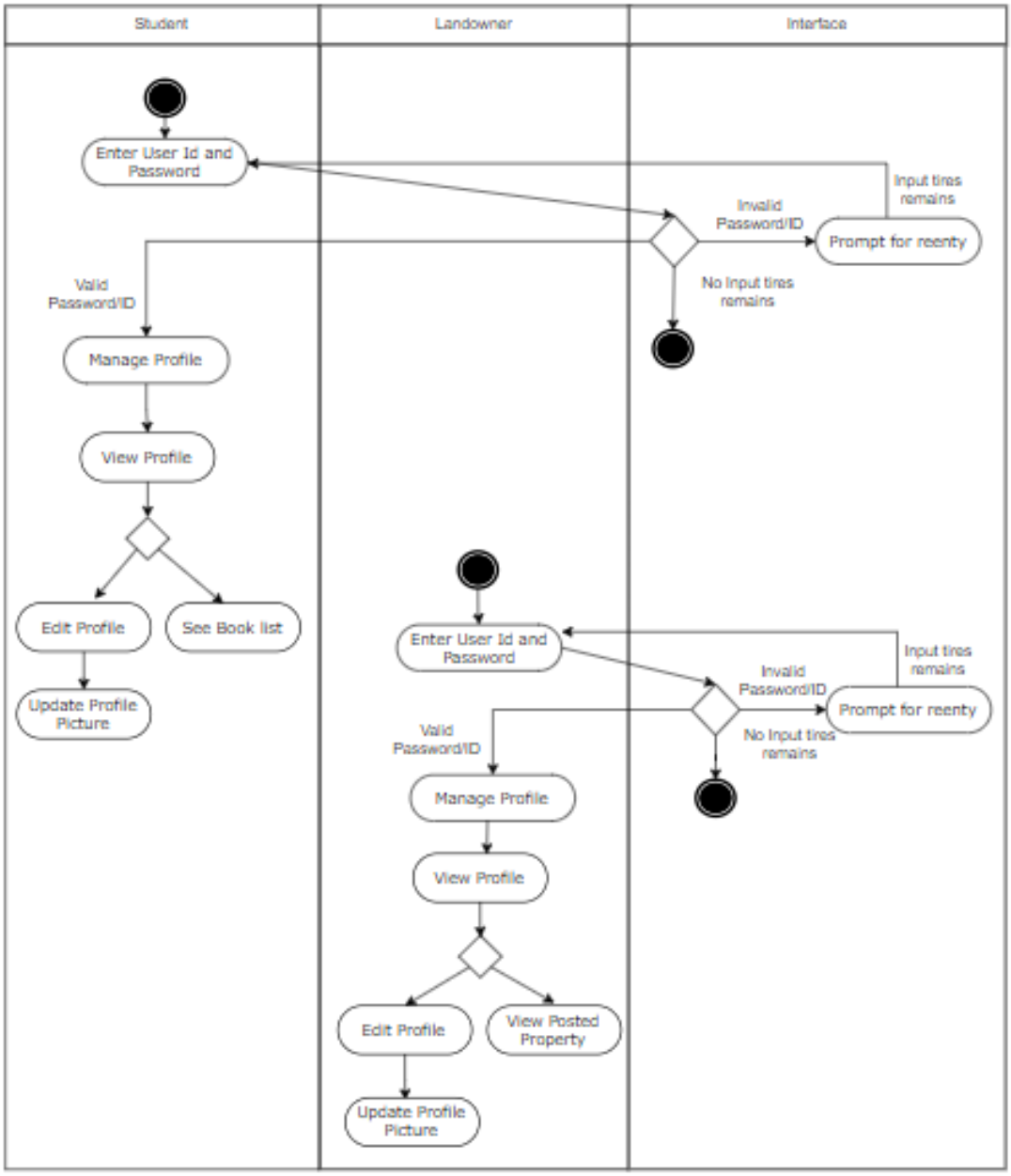

Figure 3.10: Swimlane diagram for user profile management.

3.6.2 Swimlane diagram for Post Property

The swimlane diagram depicts property posting, showcasing landowners' actions like property detail submission, negotiation responses, and deal finalization, while students engage in property search, negotiation initiation, and booking. It highlights the distinct roles and collaborative efforts of both parties in the property posting process.

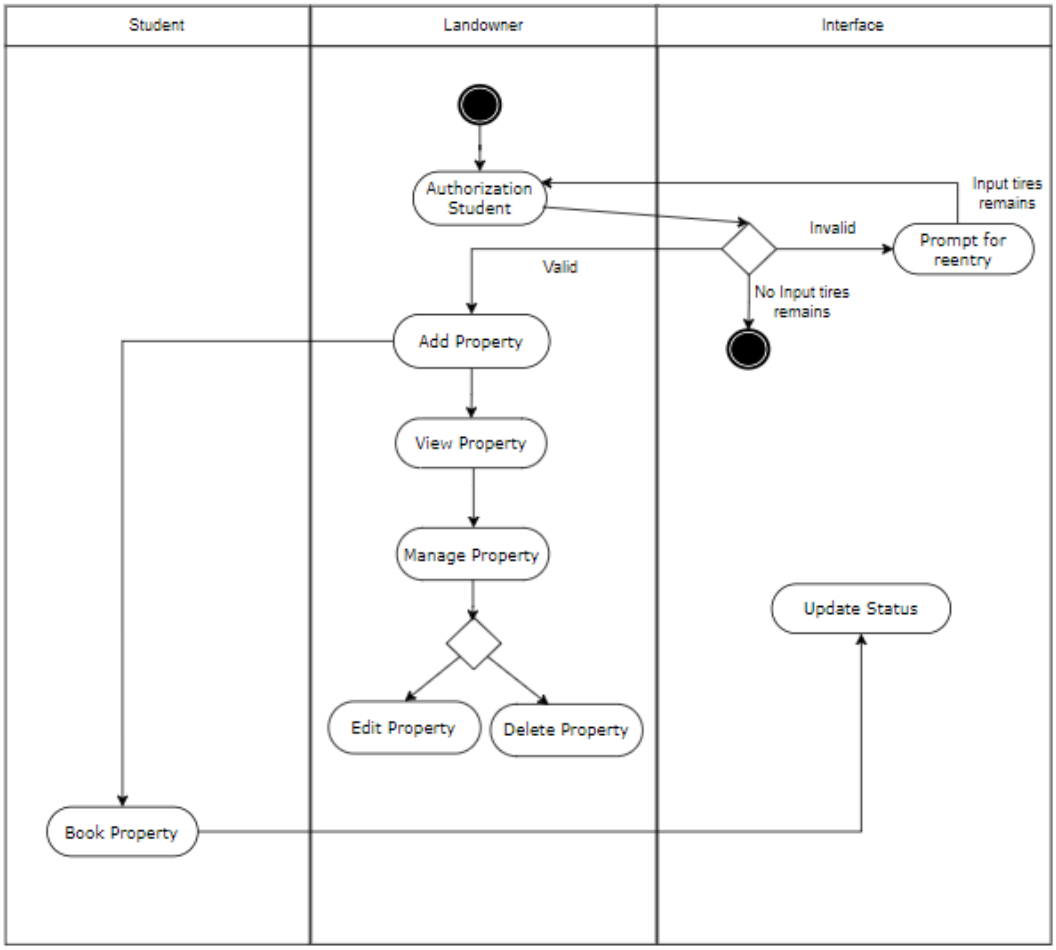

Figure 3.11: Swimlane diagram for Post Property.

3.6.3 Swimlane diagram for Book Property

In the swimlane diagram for book property, distinct lanes represent users' roles: interface, students, and landowners. The process typically begins with the user initiating a search for available properties. The website's interface interacts with the user, displaying property options based on their preferences. Once a property is selected, the user fills out booking details, such as rent duration and the user's permanent address.

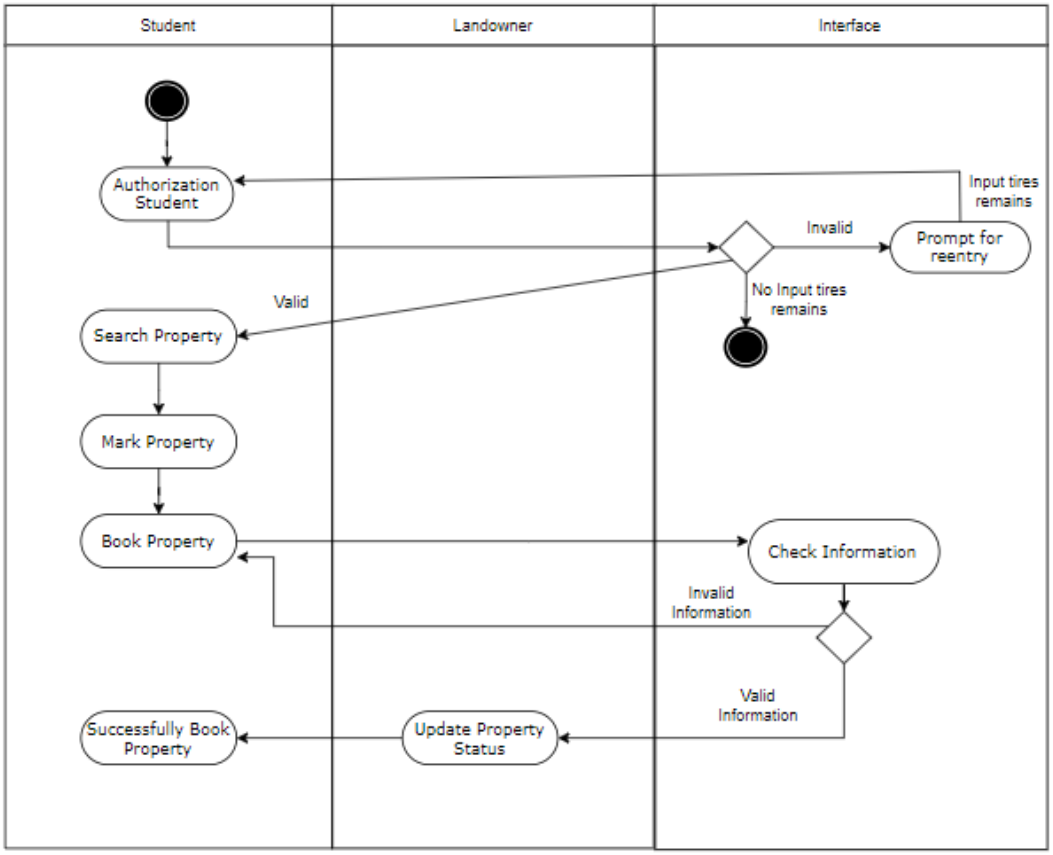

Figure 3.12: Swimlane diagram for user Book Property.

3.7 Static Model – Class Diagram

Class diagram visually represents the static structure of a system, depicting relationships and attributes of classes. The system diagram of RentSpot: Home Near CUET is given below drawn according to the reference of [1] book chapter-8 Requirements Modeling.

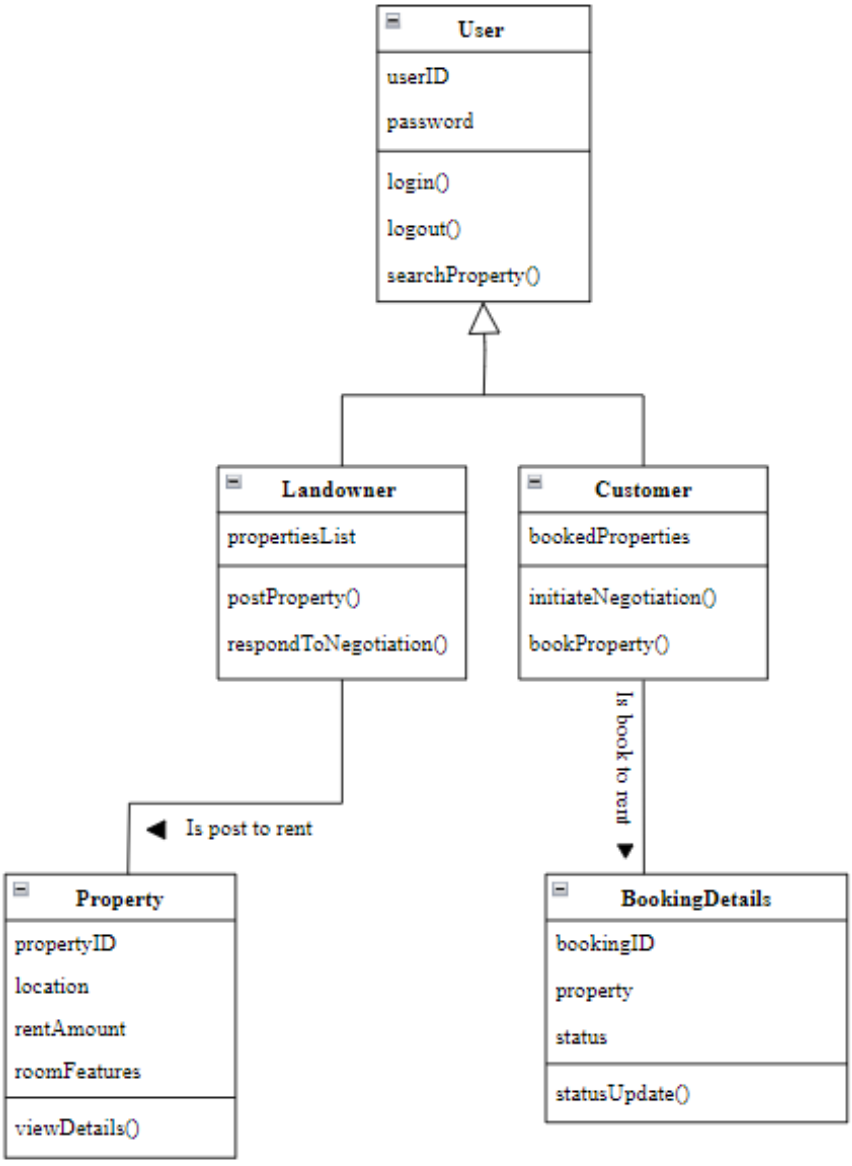

Figure 3.13: Class diagram for RentSpot: Home near CUET.

Architecture

4.1 Data Centered Architecture for Overall System

A data-centered architecture for a home rental system efficiently manages property bookings, user profiles, and property postings. It relies on a robust database system that integrates seamlessly with a user interface, enabling swift property searches, secure bookings, and easy profile management. Through structured data organization, it optimizes property listings, user information, and booking records, ensuring a streamlined and user-friendly experience for both property owners and renters.

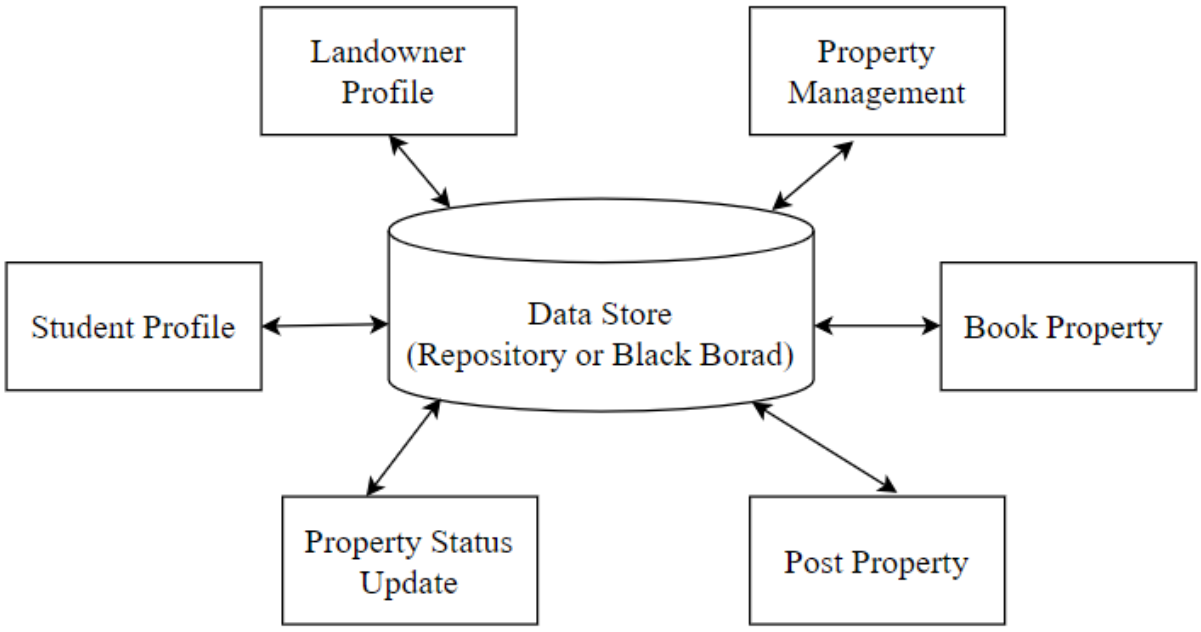

Figure 4.1: Data Centered Architecture for the overall system.

4.1.1 Rationale for Choosing Data Centered Style

- **Scalability:** Data-centered architecture allows for easy scaling as the system grows, accommodating increased property listings and user profiles effortlessly.

- **Efficiency:** Centralizing data improves system efficiency by enabling quick access, retrieval, and manipulation of information, enhancing overall system performance.
- **Consistency:** With a centralized data approach, maintaining data integrity and consistency across different functionalities becomes more manageable, ensuring reliability.
- **Flexibility:** This architecture facilitates easy integration of new features or updates, enabling adaptability to changing user needs and technological advancements.
- **Security:** Centralized data management enhances security measures, enabling better control and safeguarding of user information and transactions.

4.2 MVC Architecture for the System

In an MVC (Model-View-Controller) architecture for a home rental system, the emphasis is on separating concerns between user management, booking, and posting property.

4.2.1 MVC Architecture for User Management System

In the User Management System using MVC architecture, the Model stores user data, the View is the user interface, and the Controller manages interactions between them. This modular structure enhances organization and simplifies user-related operations.

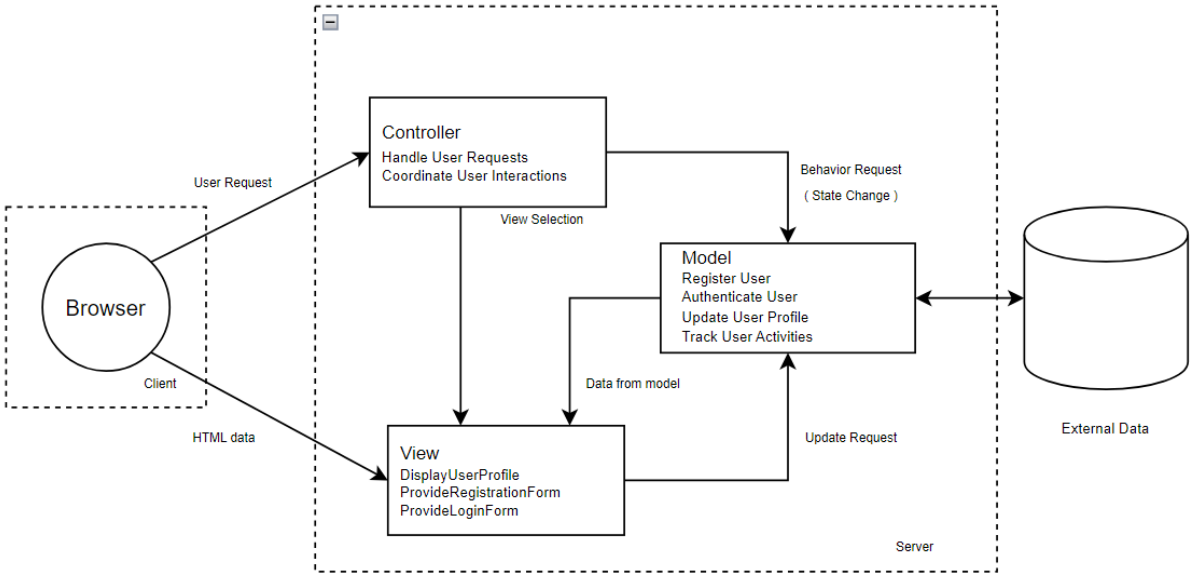

Figure 4.2: MVC Architecture for Booking System

4.2.2 MVC Architecture for Posting System

In the Posting System with MVC architecture, the Model stores post data, the View displays posts, and the Controller manages interactions between them for a streamlined posting experience.

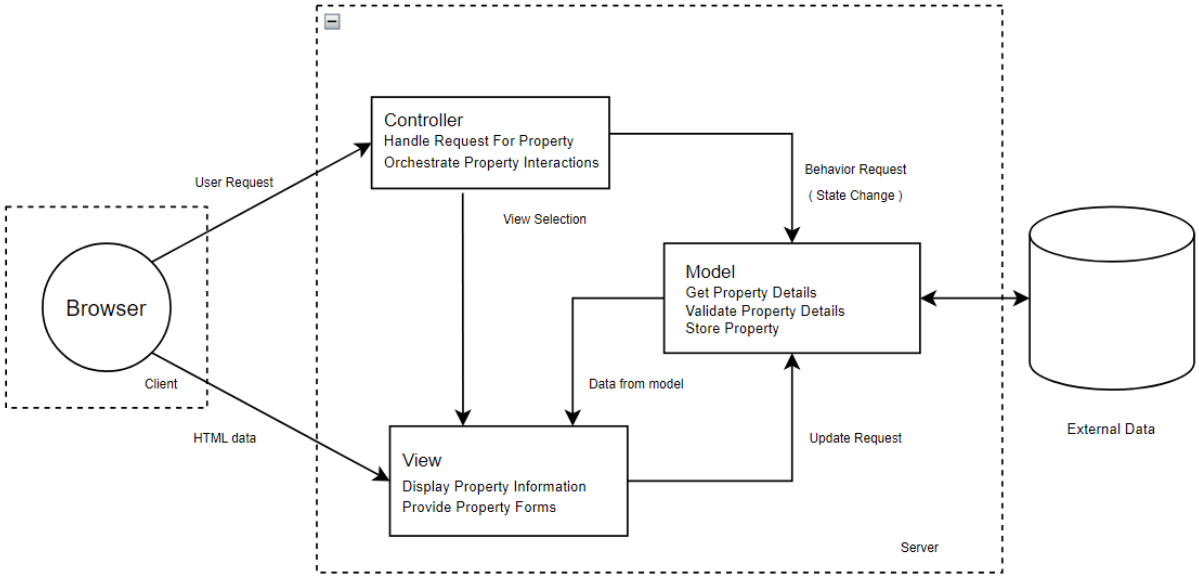

Figure 4.3: MVC Architecture for Booking System

4.2.3 MVC Architecture for Booking System

In the Booking System employing MVC architecture, the Model stores booking data, the View provides the user interface for booking interactions, and the Controller manages the communication between the Model and View, facilitating seamless booking processes.

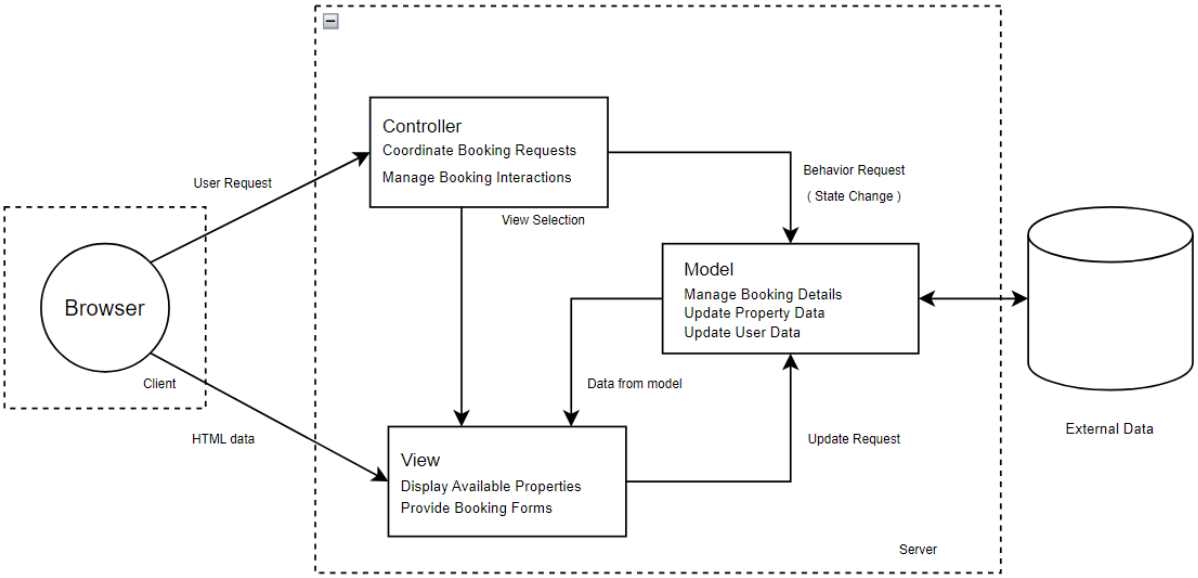

Figure 4.4: MVC Architecture for Booking System

4.2.4 Rationale for Choosing MVC Architecture

For our home rental system project, we opt for the Model-View-Controller (MVC) architecture due to its inherent benefits that align with the specific needs of our application. MVC provides a modular structure, allowing us to compartmentalize the different aspects of the system. This modularity enhances maintainability, as updates or modifications can be made to one component without affecting others. Additionally, MVC's clear distinction between the Model (data and business logic), View (user interface), and Controller (interaction and processing) ensures a scalable and adaptable architecture. This separation not only streamlines development but also promotes code reusability and ease of testing. Ultimately, choosing MVC for our project enhances the efficiency, flexibility, and collaborative aspects of our development process, aligning well with the dynamic nature of a home rental system.

4.3 Technology, software, and hardware used

The booking system utilizes a modern technology stack. On the front end, it employs React for dynamic user interfaces, CSS for styling, and Bootstrap and React Strap for responsive design. The back end is powered by Node.js for efficient server-side processing. The MySQL database ensures robust data management.

- **Frontend:** React, CSS, Bootstrap
- **Backend:** Node.js
- **Database:** MySQL

Design

5.1 Component level design pattern

A software component is an encapsulated, interchangeable unit offering specific functionalities, designed for seamless interaction with other components. It's a modular, portable, and reusable package defining clear interfaces and dependencies. It's a building block that ensures well-defined functionality within a software system. RentSpot's website features a modular, encapsulated system for property booking, listing, and user profile management. It offers replaceable, reusable components with clear interfaces, ensuring seamless interaction between customers, landowners, and the platform. Each component defines specific functionalities, fostering a cohesive and easily adaptable user experience.

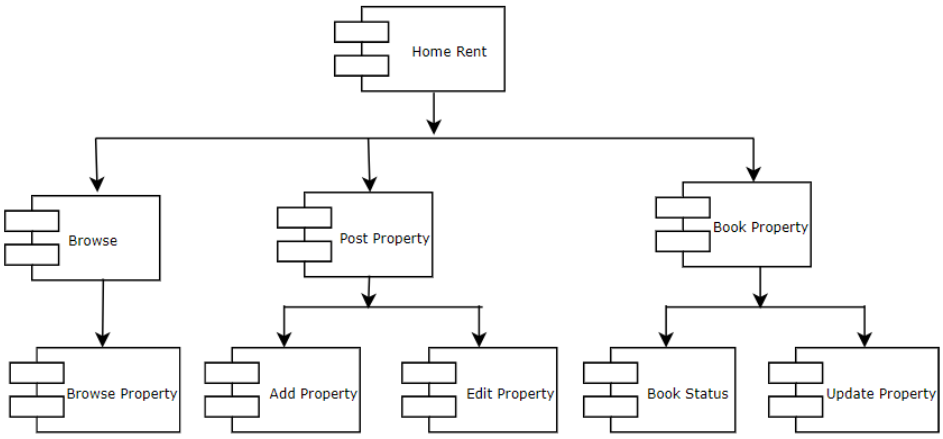

Figure 5.1: Structure Chart for Rentspot

5.1.1 Home Reservation

The component-level design outlines the Home Reservation feature using PlantUML. It comprises two components: PropertyDetails for displaying property information and BookingManager for handling the booking process. The PropertyBooking class interacts with these components to view property details and initiate bookings. This design pro-

motes modularity and encapsulation, enhancing the maintainability of the Home Reservation feature.

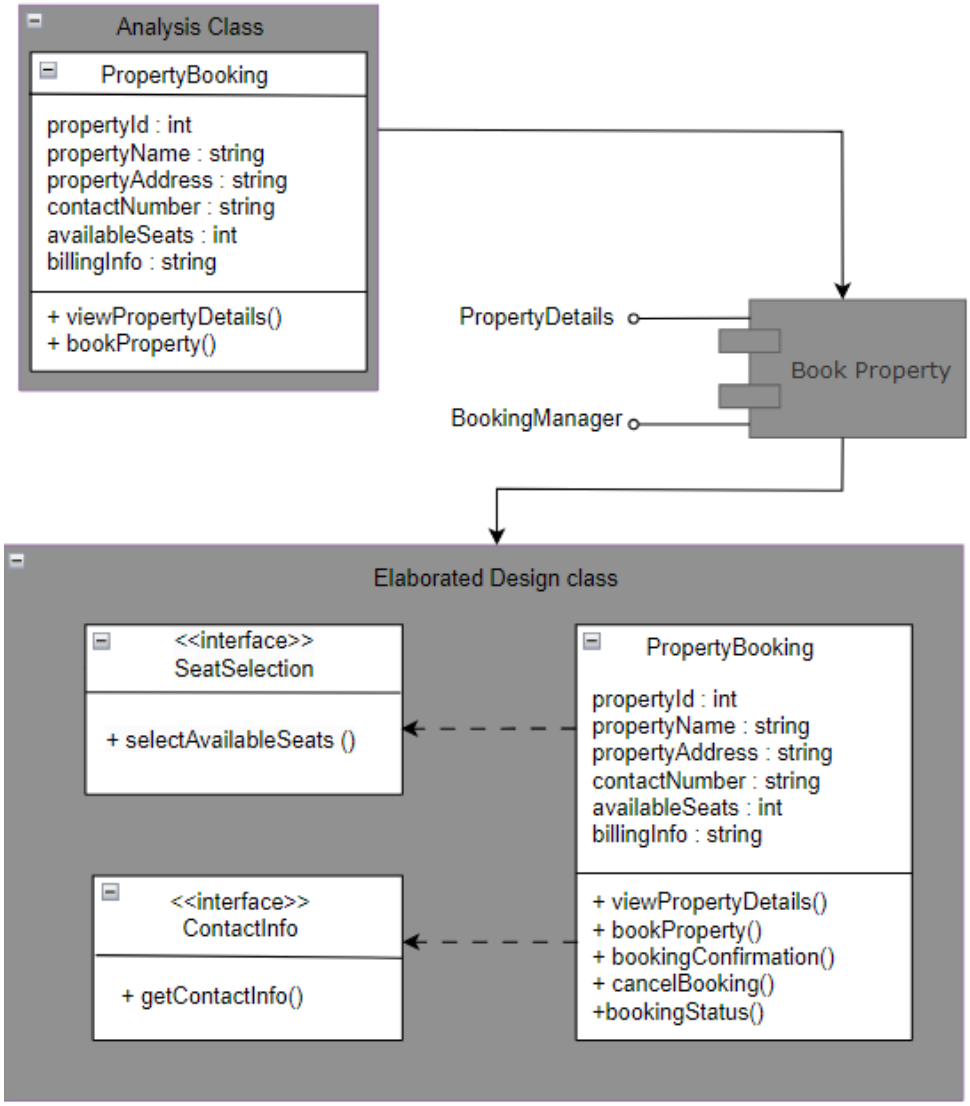

Figure 5.2: Component level design for Home Reservation.

5.1.2 Property Management

Our apartment renting platform empowers landlords to post their empty apartments with detailed descriptions and images effortlessly. Through a streamlined interface, landlords can manage their listings while users browse available apartments, creating a seamless experience for both renters and property owners.

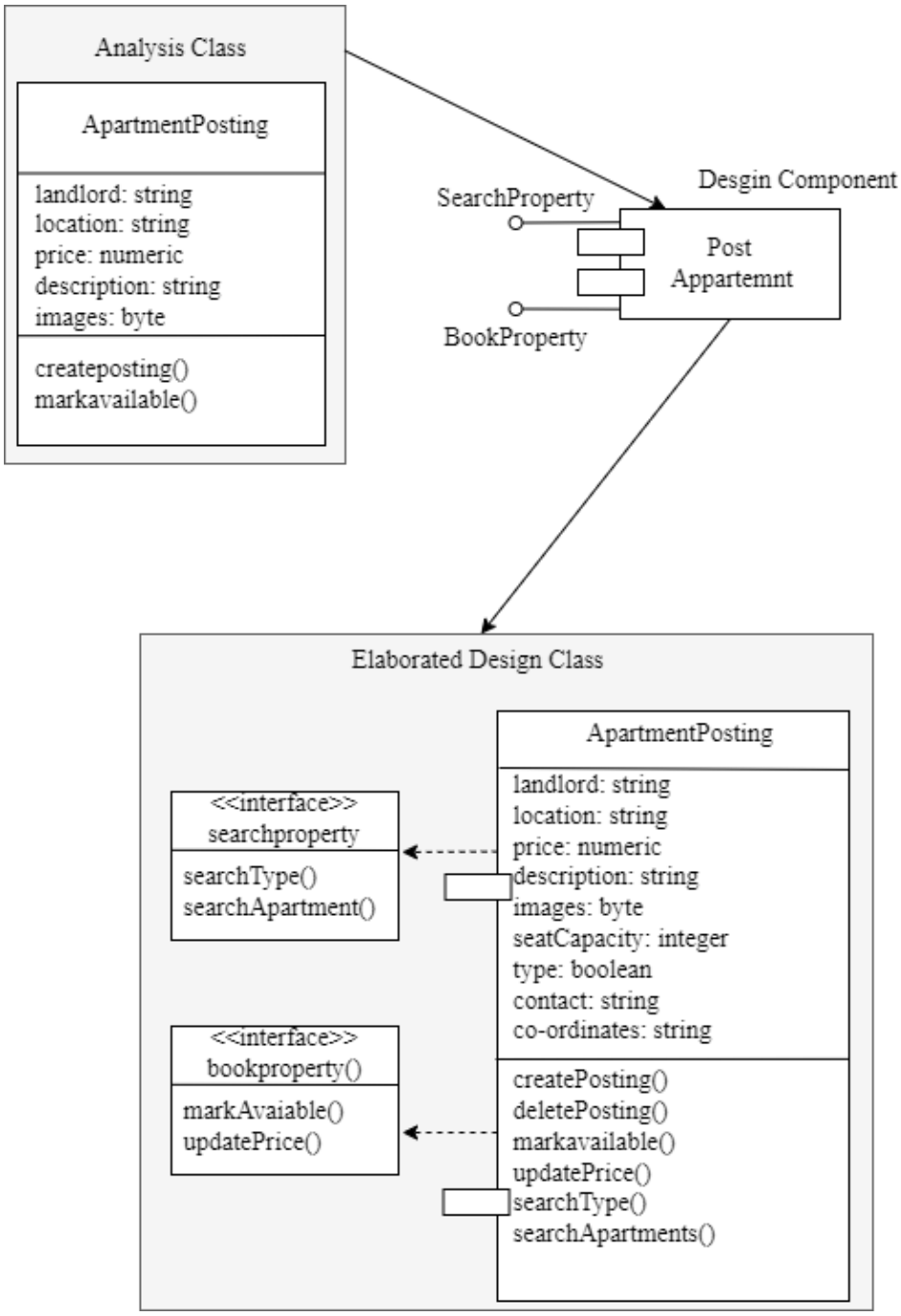

Figure 5.3: Component level design for Property Management.

5.1.3 User Profile Management

Our user profile management system offers a centralized platform for users to effortlessly update personal information, manage privacy settings, and adjust preferences. With a clean and intuitive interface, users can navigate through sections like security settings, notification preferences, and connected accounts seamlessly. Leveraging robust backend integration and stringent testing, our system ensures data security, reliable functionality, and a user-centric experience.

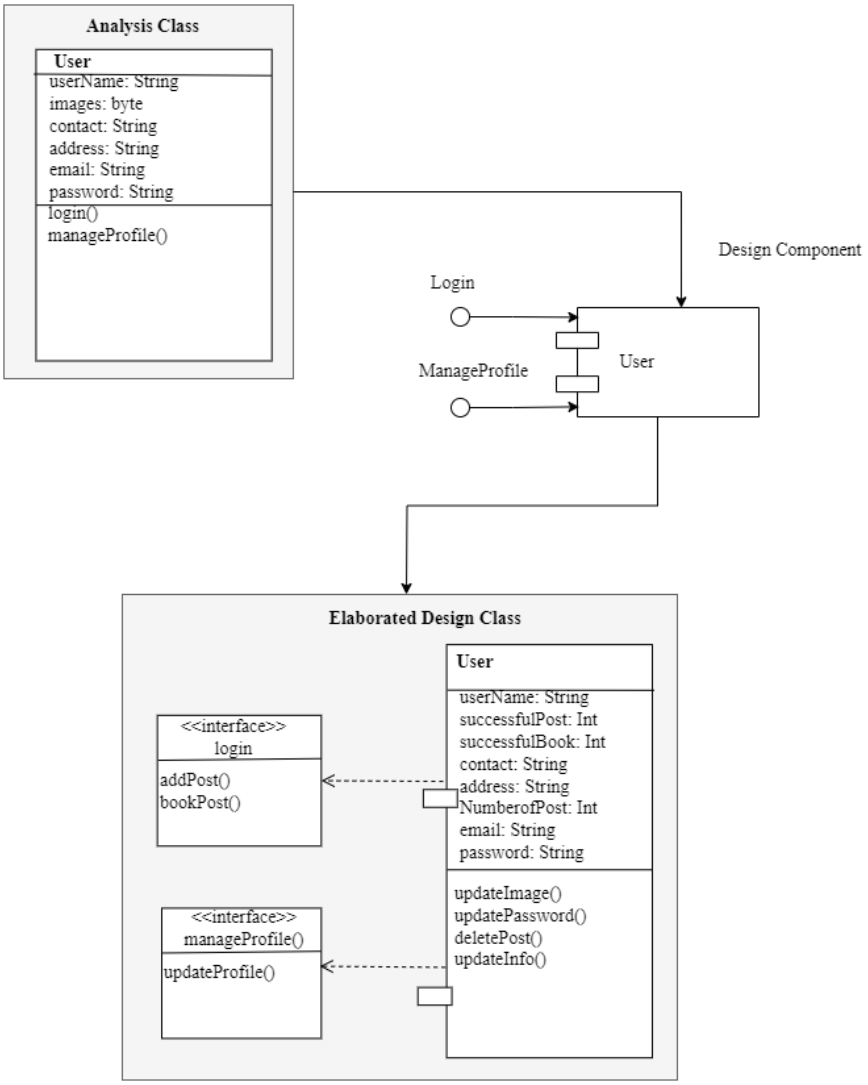

Figure 5.4: Component level design for user profile management.

5.2 Demo GUI(Graphical User Interface) Design

A graphical user interface (GUI) allows users to interact with a computer device. Here are the interfaces of our System.

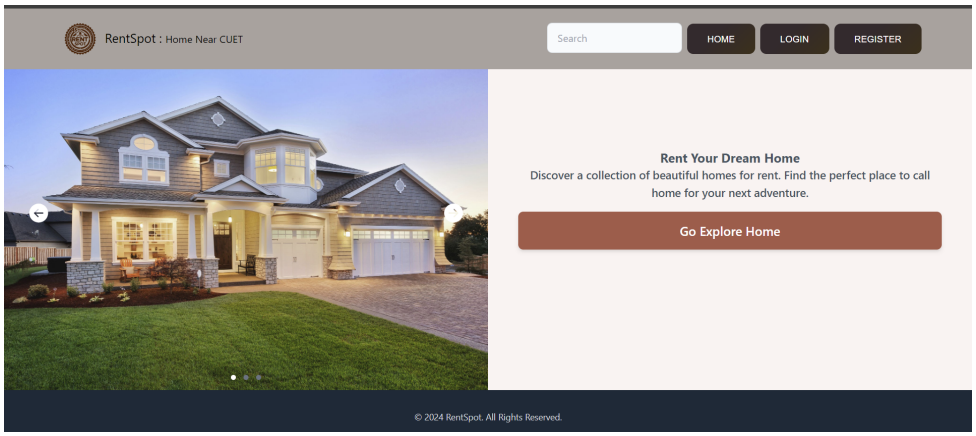

Figure 5.5: Home page of our webiste

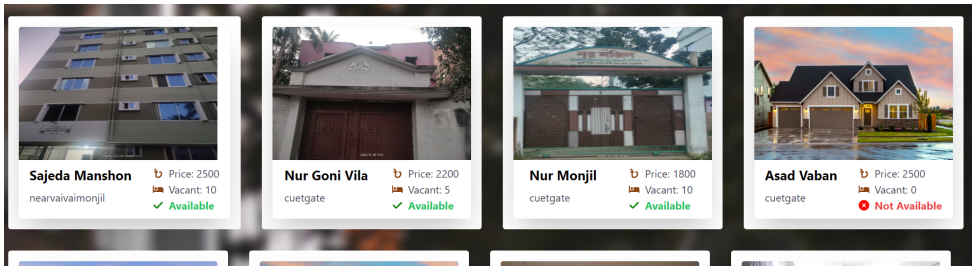

Figure 5.6: Appartment list of of our webiste

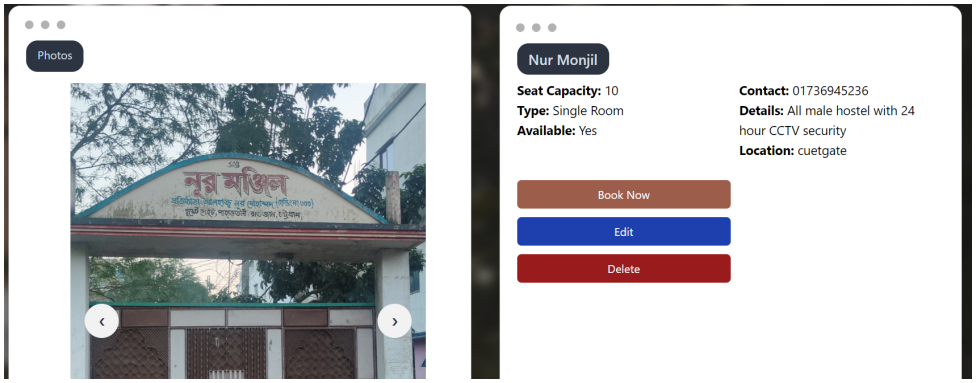

Figure 5.7: Appartment details of our webiste

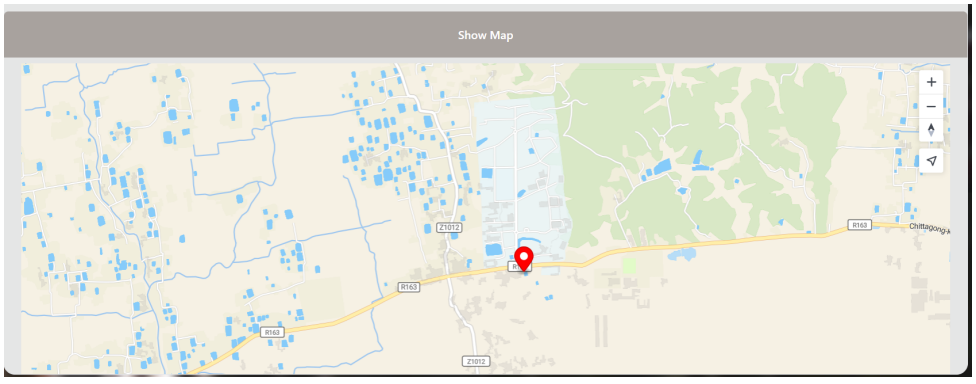

Figure 5.8: Location of specific apartment

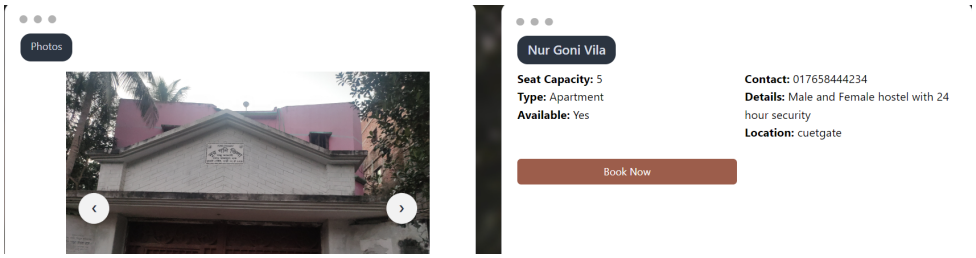

Figure 5.9: Booking apartment page for the website

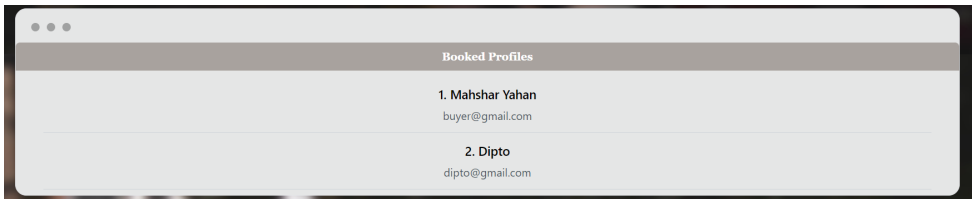

Figure 5.10: List of booked students in our website

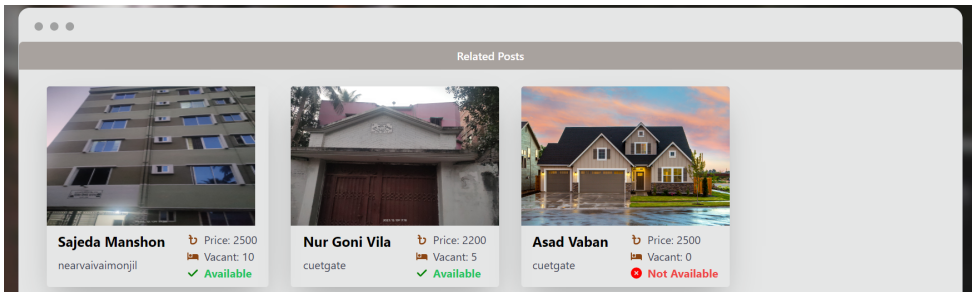

Figure 5.11: Related post of the apartment in our website

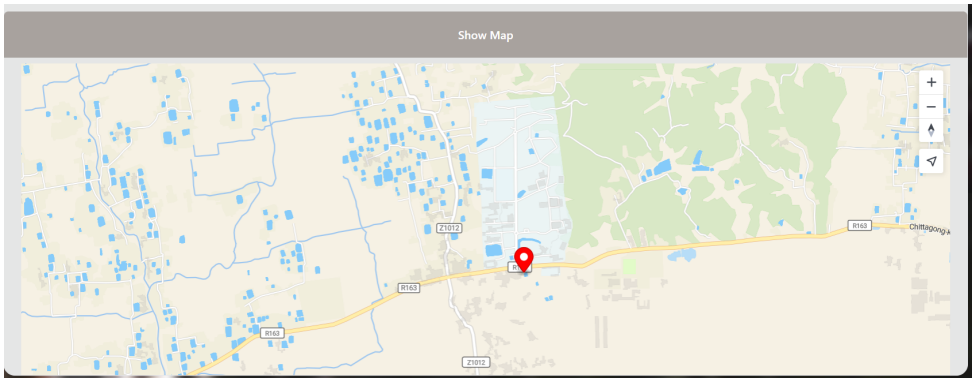

Figure 5.12: Location of specific apartment

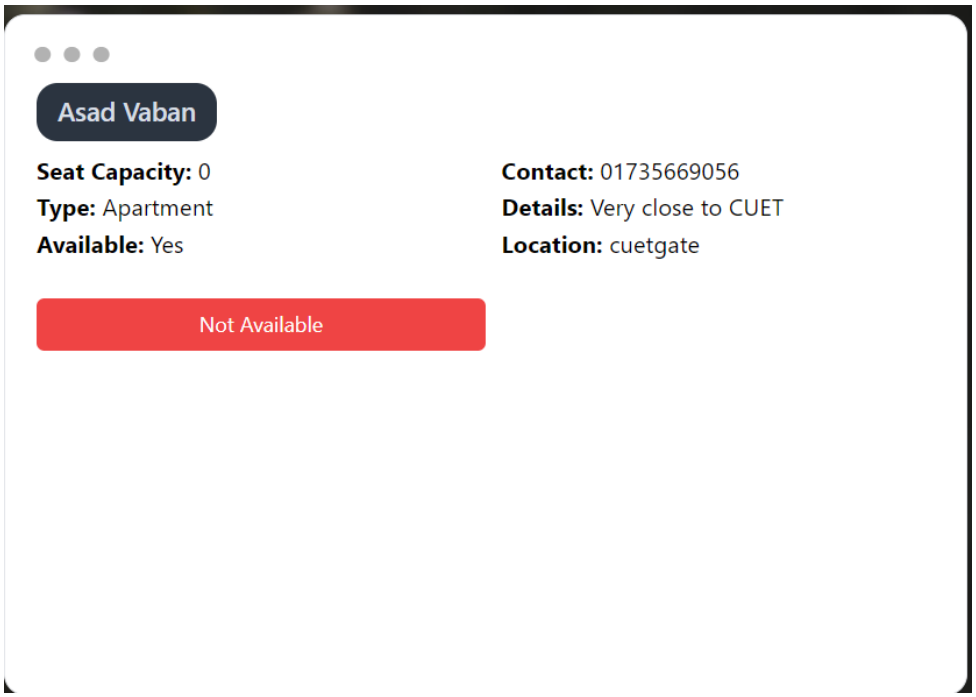

Figure 5.13: No seat remains in apartment in the website

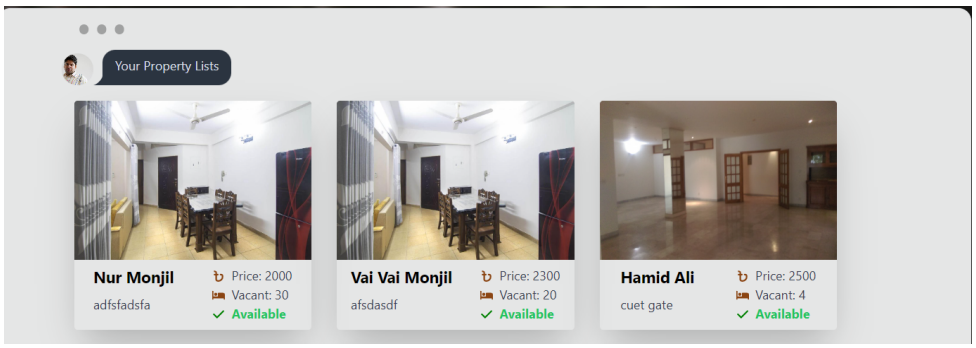

Figure 5.14: Posted property list of the website

5.3 Dataflow Diagram (DFD)

5.3.1 First Level DFD for the System

The Level 1 Data Flow Diagram (DFD) for the system includes entities like ”User,” ”Property,” and ”Booking.” It demonstrates how users interact with the system by posting properties and making bookings, illustrating data flow among these components for property management and booking processes.

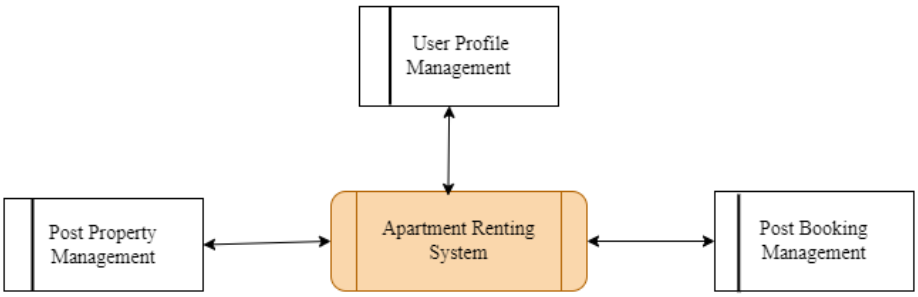

Figure 5.15: First Level Dataflow diagram for Home Reservation System.

5.3.2 DFD for Booking Property

The Data Flow Diagram (DFD) illustrates the home reservation feature. Users interact with the system through the user interface, entering booking details. The Booking System processes the booking, updates the User on confirmation, and adjusts the property's availability. The flow emphasizes simplicity, with data moving from user input to system processes and updates.

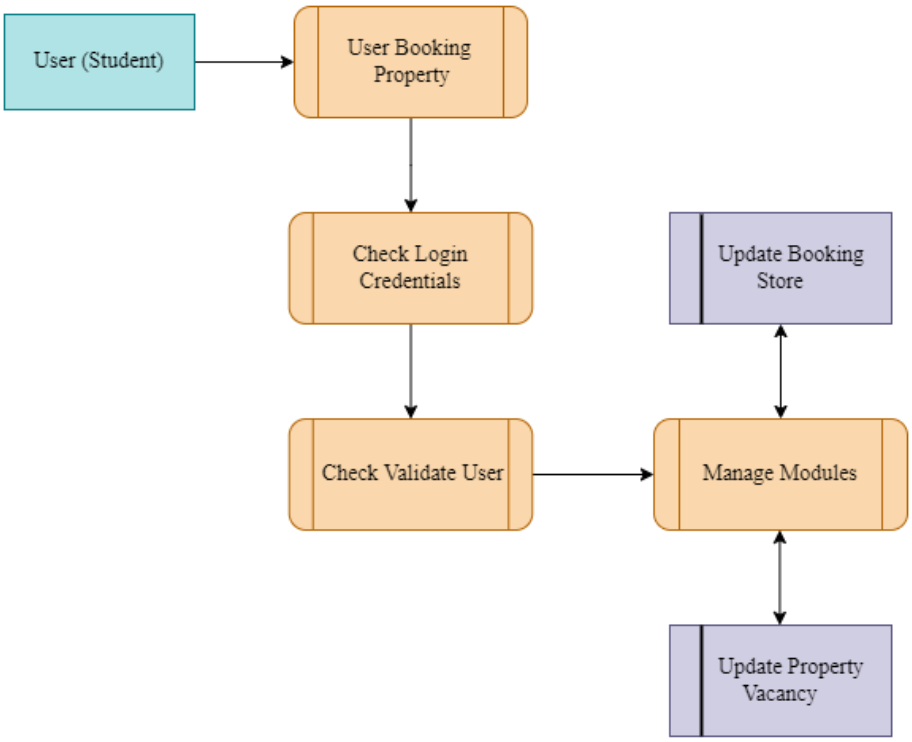

Figure 5.16: First Level Dataflow diagram for Home Reservation.

5.3.3 DFD for Post Property Management

Property Management dataflow diagram shows how the landowner inputs apartment details, funneling information into the Property Listings data store. Published via the listing platform, potential tenants search listings. Inquiries submitted by tenants are handled, and responses are directed back to potential tenants.

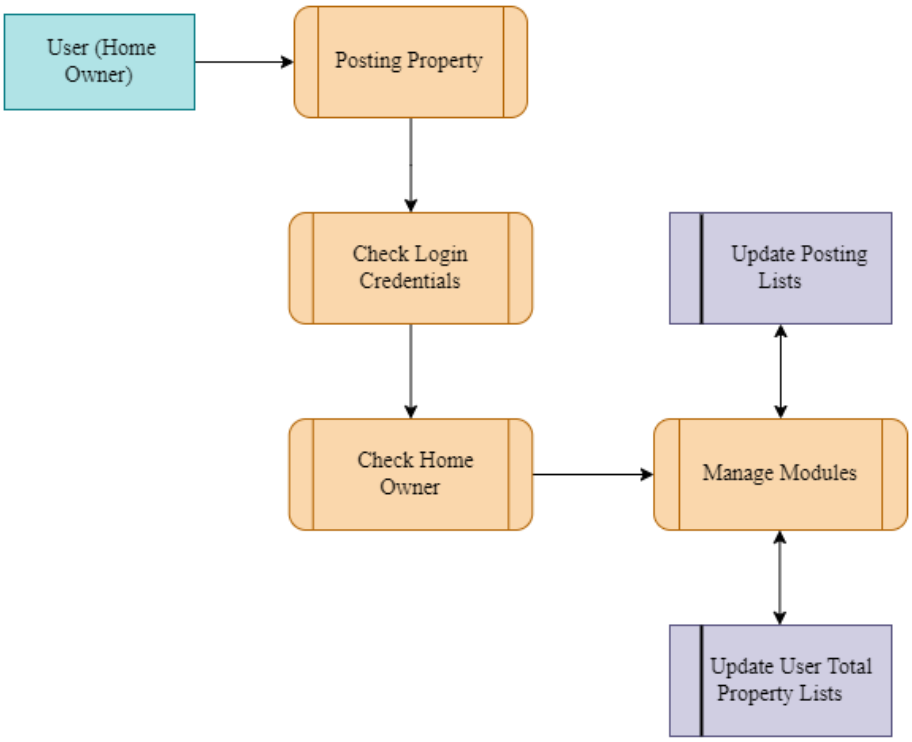

Figure 5.17: First Level Dataflow diagram for Property Management.

5.3.4 DFD for User Profile Management

The Level 0 DFD for user profile management showcases interactions between users and the system. Users input data for profile updates, which undergo authentication before updating the User Profile Database. At Level 1, the process delves deeper: user inputs are validated, profiles updated, and identity verified before retrieving or storing data in the central User Profile Database, ensuring a secure and accurate user profile management system.

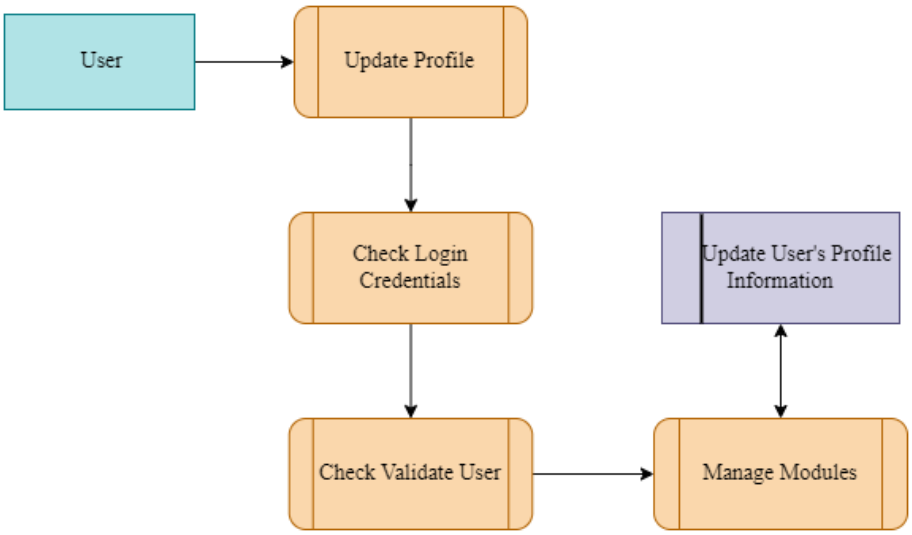

Figure 5.18: First Level Dataflow diagram for user profile management

5.4 Entity Relationship Diagram (ERD)

The ER (Entity-Relationship) diagram for a website involving booking and property posting includes entities like User, Property, and Booking and their relationships. Users can post properties, which can have multiple bookings, establishing associations among these entities to facilitate property management and booking functionalities. Bookings include details such as booking date, duration, and total cost. The relationships indicate that a user can make multiple bookings, and each property can be booked multiple times.

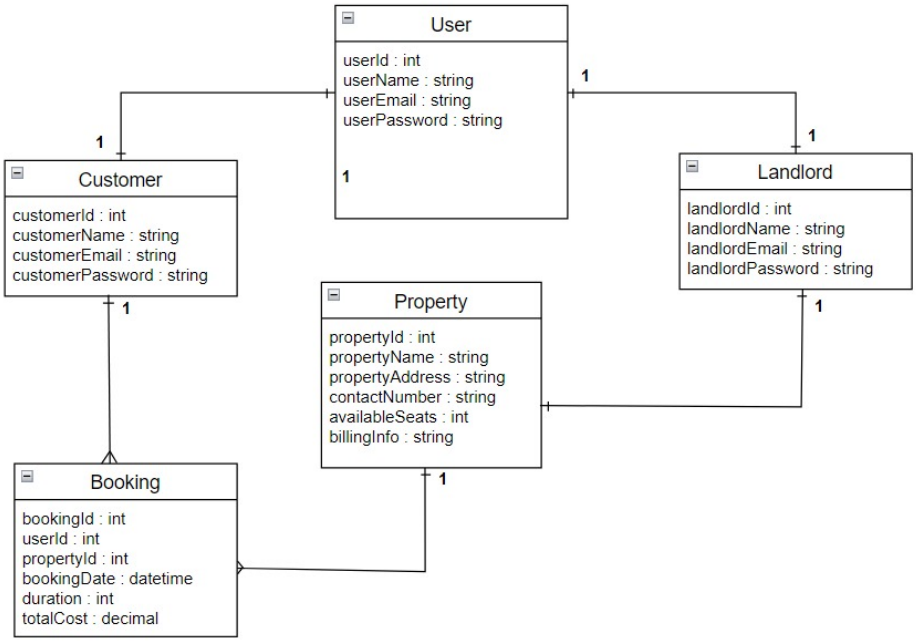

Figure 5.19: ER Diagram for Booking

API Documentation Manual

6.1 User Management

The User Management API documentation in Swagger provides comprehensive details on endpoints for user creation, retrieval, update, and deletion. It outlines request and response formats, authentication methods, and error handling, facilitating seamless integration and management of user-related operations within applications.

6.1.1 Register New User

Request

Parameter	Values	Condition	Expiations	Example
Email	string	Mandatory	Unique email of user	yahan@gmail.com
Name	string	Mandatory	Name of user	Mahshar Yahan
Password	string	Mandatory	Password for account	abc123
Role	string	Mandatory	Landowner or student	BUYER

Figure 6.1: INPUT Parameters for the request JSON.

Curl

```
curl -X 'POST' \
  'http://localhost:5000/users/register' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "email": "shimul@gmail.com",
  "name": "Shimul",
  "password": "123456",
  "role": "BUYER"
}'
```

Request URL

```
http://localhost:5000/users/register
```

Figure 6.2: Header and Request URL

Response

Server response

Code	Details
201	Response body

```
{
  "id": "5f938600-e4a0-4b99-924f-e8aeab6160fe",
  "name": "Shimul",
  "email": "shimul@gmail.com",
  "password": "$2a$10$3iqBriQJOoj4tz.mpKhusem5TQHmWXzS4WffsPnc0OUQPJ.6tKraK",
  "role": "BUYER",
  "createdAt": "2024-01-02T12:08:18.820Z",
  "updatedAt": "2024-01-02T12:08:18.820Z"
}
```

Download

Response headers

```
connection: keep-alive
content-length: 255
content-type: application/json; charset=utf-8
date: Tue,02 Jan 2024 12:08:18 GMT
etag: W/"ff-3oNuvvXN4utA8x+S7MMnU4n9qZE"
keep-alive: timeout=5
x-powered-by: Express
```

Figure 6.3: Response Body and response header

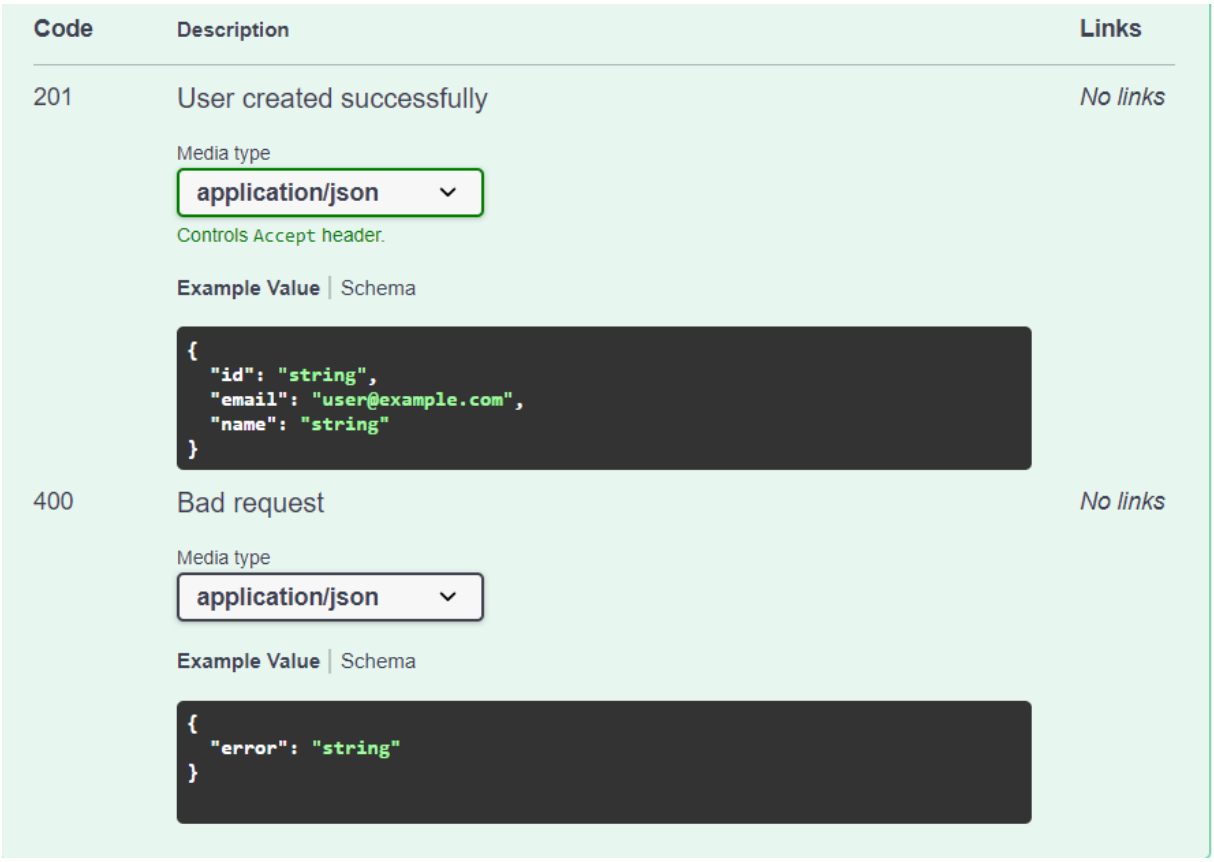

Figure 6.4: Confirmation of responses

6.1.2 Login User

Request

Parameter	Values	Condition	Expiations	Example
Name	string	Mandatory	Name of user	Mahshar Yahan
Password	string	Mandatory	Password for account	abc123

Figure 6.5: INPUT Parameters for the request JSON.

Figure 6.6: Header and Request URL

Response

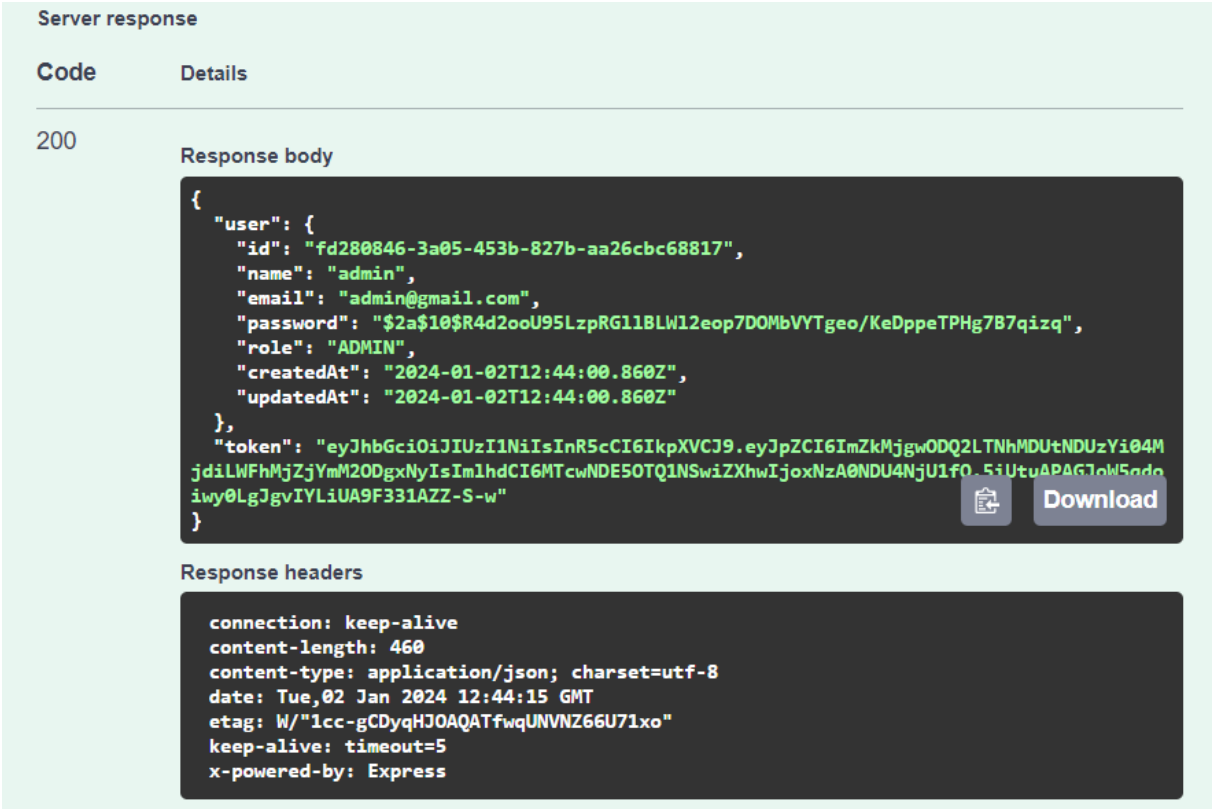

Figure 6.7: Response Body and response header

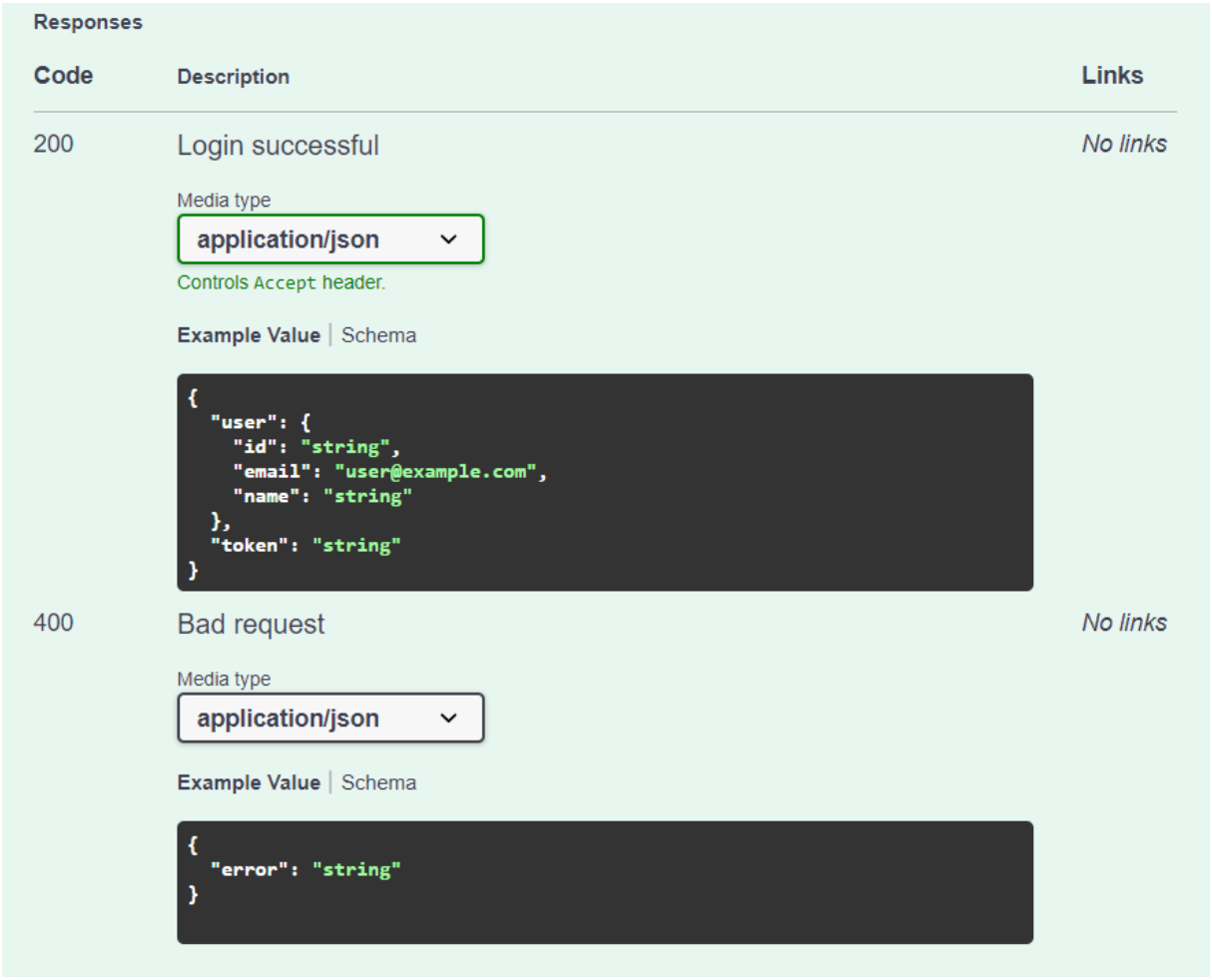

Figure 6.8: Confirmation of responses

6.1.3 Update

Request

Parameter	Values	Condition	Expiations	Example
Name	string	Mandatory	Name of user	Mahshar Yahan
Password	string	Mandatory	Password for account	abc123

Figure 6.9: INPUT Parameters for the request JSON.

Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:5000/users/7e56e41c-9e1d-4548-bfbc-5d071a71bd33' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "email": "dipto@gmail.com",
  "name": "Tanzim Dipto",
  "password": "123456",
  "role": "BUYER"
}'
```

Request URL

```
http://localhost:5000/users/7e56e41c-9e1d-4548-bfbc-5d071a71bd33
```

Figure 6.10: Header and Request URL

Response

Server response

Code	Details
200	Response body

```
{
  "id": "7e56e41c-9e1d-4548-bfbc-5d071a71bd33",
  "name": "Tanzim Dipto",
  "email": "dipto@gmail.com",
  "password": "123456",
  "role": "BUYER",
  "createdAt": "2024-01-02T08:36:10.189Z",
  "updatedAt": "2024-01-02T12:44:32.088Z"
}
```

Download

Response headers

```
connection: keep-alive
content-length: 206
content-type: application/json; charset=utf-8
date: Tue,02 Jan 2024 12:44:32 GMT
etag: W/"ce-T0Bn6muNSSS0wi1gKEoCwK8MDXA"
keep-alive: timeout=5
x-powered-by: Express
```

Figure 6.11: Response Body and response header

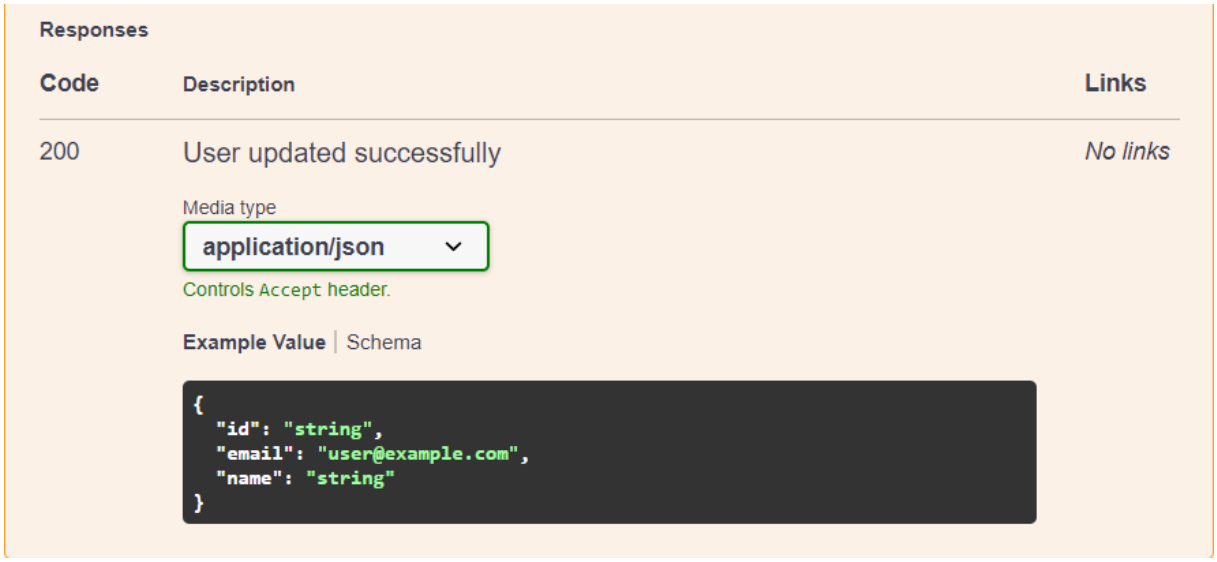

Figure 6.12: Confirmation of responses

6.1.4 Delete

Request

Parameter	Values	Condition	Expiations	Example
User ID	string	Mandatory	Unique ID for users	8c375985-b683-4c16-a06a-47b7d78a84c5

Figure 6.13: INPUT Parameters for the request JSON.

Figure 6.14: Header and Request URL

Response

Server response

Code	Details
200	Response body

```
{
  "id": "8c375985-b683-4c16-a06a-47b7d78a84c5",
  "name": "Yahan",
  "email": "abc@gmail.com",
  "password": "$2a$10$EjkHDyJwi2FzTr9FXA3S..vO2nloIZrqW.TedOFJkNr8Vcpxv4LNi",
  "role": "BUYER",
  "createdAt": "2023-12-28T17:54:48.536Z",
  "updatedAt": "2023-12-28T17:54:48.536Z"
}
```

Download

Response headers

```
connection: keep-alive
content-length: 251
content-type: application/json; charset=utf-8
date: Tue,02 Jan 2024 13:01:37 GMT
etag: W/"fb-fYKSpyYkvz8/2K914G8gj+Ab7Ww"
keep-alive: timeout=5
x-powered-by: Express
```

Figure 6.15: Response Body and response header

Responses

Code	Description	Links
200	User deleted successfully	*No links*

Media type

application/json

Controls Accept header.

Example Value | Schema

```
{
  "id": "string",
  "email": "user@example.com",
  "name": "string"
}
```

Figure 6.16: Confirmation of responses

6.2 Post

6.2.1 Create Post

Request

Parameter	Values	Condition	Expiations	Example
title	string		Title of Property	Nur Monjil
price	integer	Mandatory	Rent of Property	2000
seatCapacity	integer		Capacity of seats	4
type	integer		Type of Property	Apartment
available	boolean	Mandatory	Availability of Property	true
contact	string	Mandatory	Contact number of Landlord	01735996049
details	string		Information about Property	Two storied building with 24/7 CCTV monitoring
location	string	Mandatory	Location of the property	CUET gate opposite
coordinates	string	Mandatory	GPS location	22.461694, 91.970939

Figure 6.17: INPUT Parameters for the request JSON.

Figure 6.18: Header and Request URL

Response

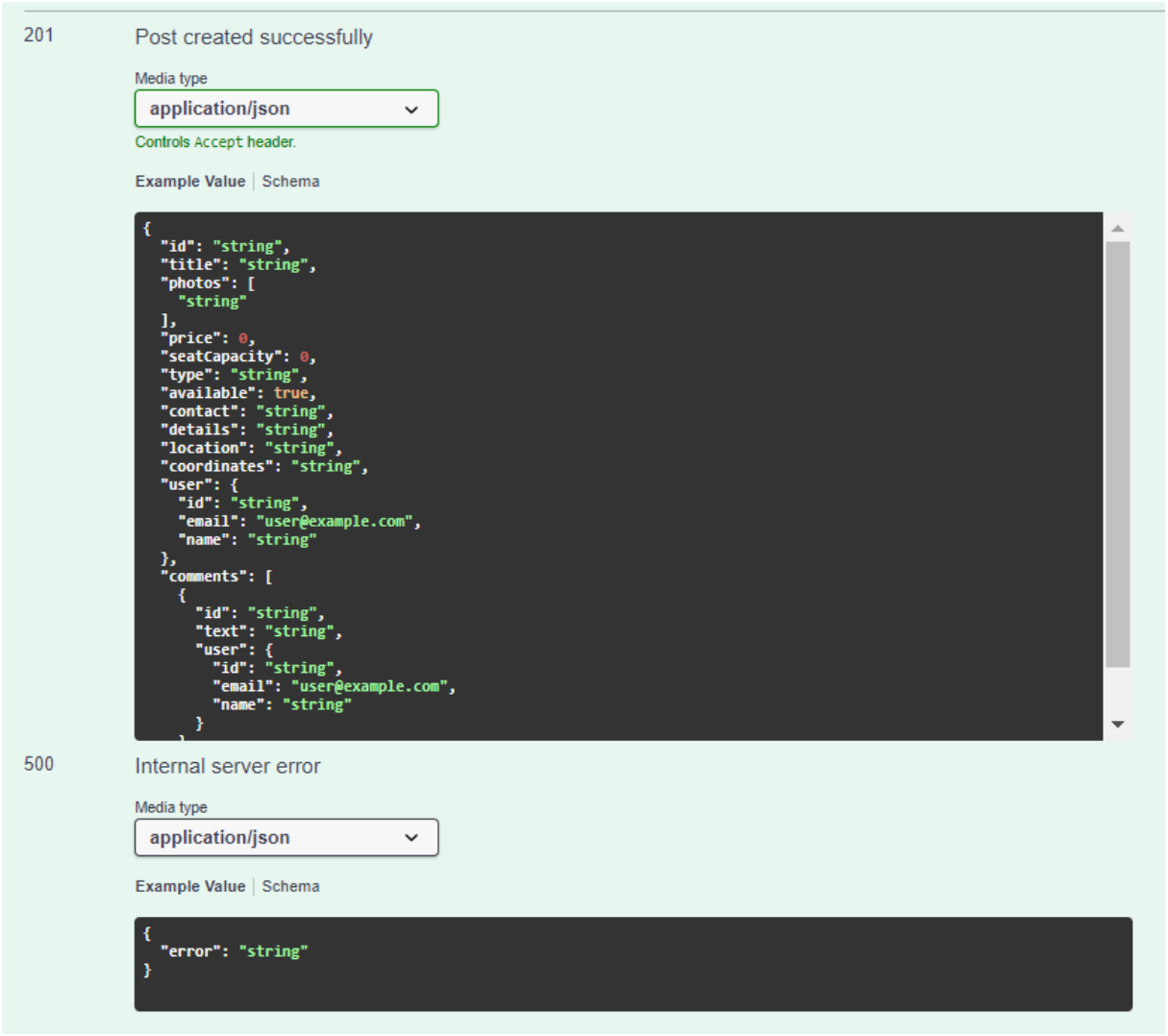

Figure 6.19: Confirmation of responses

6.2.2 Delete Post

Testing and Sustainability Plan

7.1 Requirements/Specifications-based System Level Test Cases

In the property renting project, system testing relies on a comprehensive specification document. It involves isolating each component for thorough examination, ensuring alignment with end-user expectations. A concise tabulated representation of system-level test cases, organized by requirements, ensures systematic validation, affirming the system's reliability and adherence to specified criteria. Table for Requirements/Specifications-based system-level test cases is given below:

Requirement ID	Requirement Statement	Must/ Want	Comment
R-Upload-01	Users can't post a property listing without providing the address	Must	N/A
R-Upload-02	Display an error message if the user attempts to post a property without specifying the available seats	Want	N/A
R-Search-01	Users must be able to search for properties based on location and other relevant criteria	Must	N/A
R-Search-02	Display a notification if there are no matching properties for the user's search criteria	Want	N/A
R-Booking-01	Users cannot book a property without providing necessary details such as duration and contact information	Must	N/A

Table 7.1: Requirements/Specifications-based System Level Test Cases (Home Rental System)

Project Name	RentSpot: Home Near CUET (Web Application)					
Module Name	Upload Post					
Created By	Tanzim Rahman					
Reviewed By	Shimul Mahmud					
Date of Creation	15-01-24					
Date of Review	18-01-24					
Test Case ID	**Scenario**	**Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
R-Upload-01	Check upload image option with invalid file extension.	1. Go to the user interface 2. Click 'choose file' option 3. Upload file from PC's directory	File: new.pdf	Redirect to the initial state of user interface	Redirect to the initial state of user interface	Passed
R-Upload-02	Attempt to post a property listing without specifying the available seats.	1. Navigate to property upload section. 2. Skip entering the available seats field. 3. Attempt to submit the listing.	Property details without specifying available seats.	Display an error message indicating that specifying available seats is mandatory.	Error message appears, indicating that specifying available seats is mandatory.	Passed

Table 7.2: Requirements/specifications-based system level test cases (01)

Project Name	RentSpot: Home Near CUET (Web Application)					
Module Name	Search Posts					
Created By	Mahshar Yahan					
Reviewed By	Shimul Mahmud					
Date of Creation	20-01-24					
Date of Review	22-01-24					
Test Case ID	**Scenario**	**Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
R-Search-01	Search for properties based on location and relevant criteria.	1. Navigate to the search section. 2. Enter location and other relevant criteria. 3. Initiate the search.	Location: ”City A”, Criteria: ”Pet-friendly”	Display a list of properties matching the specified location and criteria.	List of properties appears, matching the specified location and criteria.	Passed
R-Search-02	Search for properties with criteria that have no matching results.	1. Navigate to the search section. 2. Enter criteria with no matching results. 3. Initiate the search.	Criteria: ”Swimming Pool, Sauna”	Display a notification indicating that there are no matching properties.	Notification appears, indicating that there are no matching properties.	Passed

Table 7.3: Requirements/specifications-based system level test cases (02)

Project Name	RentSpot: Home Near CUET (Web Application)					
Module Name	Booking					
Created By	Mahshar Yahan					
Reviewed By	Tanzim Rahman,					
Date of Creation	23-01-24					
Date of Review	25-01-24					
Test Case ID	**Scenario**	**Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
R-Booking-01	Attempt to book a property without providing necessary details such as duration and contact information.	1. Select a property to book. 2. Skip entering the contact information. 3. Attempt to submit the booking.	Duration: Not provided, Contact Information: Not provided.	Display an error message indicating that providing necessary details such as duration and contact information is mandatory.	Error message appears, indicating that providing necessary details such as duration and contact information is mandatory.	Passed

Table 7.4: Requirements/specifications-based system level test cases (03)

7.2 Techniques used for test generation

At RentSpot: Home Near CUET, we utilize Black Box Testing methods, including Equivalence Partitioning. Equivalence Partitioning for our platform entails segmenting the input range into separate data categories and using these categories to generate test cases. This technique involves examining equivalence classes associated with a specific input condition to distinguish between valid and invalid states. The input condition is Boolean, delineating one valid category and one invalid category.

7.3 Assessment of the goodness of your test suite

In evaluating the reliability and accuracy of our RentSpot: Home Near CUET platform during the testing phase, we adopt a meticulous and systematic approach. We utilize black

box testing methodology, primarily focusing on functional requirements. Specifically, we employ equivalence partitioning to evaluate different input conditions thoroughly. This technique categorizes input conditions into valid and invalid equivalence classes, facilitating a comprehensive assessment of the platform's functionality. Through these black box testing methods, we ensure the robustness and dependability of the RentSpot: Home Near CUET system.

7.4 Sustainability Plan

The components that uphold the sustainability of our RentSpot: Home Near CUET platform collectively constitute a sustainability strategy. The main objective of this strategy is to foster a balanced and sustainable pace of development that can be sustained over the long term. Optimal development here doesn't solely prioritize speed, but rather seeks a pace that balances immediate requirements with future objectives. Sustainable development involves achieving efficiency and preserving a harmonious equilibrium between short-term necessities and long-term aspirations within our project.

7.4.1 Scalability

Scalability stands as a pivotal factor in evaluating the efficiency of our RentSpot: Home Near CUET platform. A scalable software system can adeptly manage varying user loads, maintaining its performance and effectiveness whether dealing with a small or large user base. Our RentSpot: Home Near CUET platform, built with ReactJs, NodeJs, and PostGreSQL framework, places a strong emphasis on scalability by offering robust features and enhancing adaptability. This strategy reduces maintenance costs and boosts user satisfaction. The platform is engineered to seamlessly accommodate a growing user base and effectively handle increased workloads or expanded functionalities. Scalability ensures the sustained efficiency and effectiveness of our software platform, adapting seamlessly to changing conditions or user requirements.

7.4.2 Flexibility / Customization

The RentSpot: Home Near CUET platform is engineered to be adaptable, leveraging technologies such as TailwindJs and ReactJs. It integrates PostGreSQL and NodeJs to ensure flexibility in response to evolving user requirements. With a straightforward and dependable design, the platform facilitates swift adjustments, thereby saving time and resources. Striking a balance between a robust design and adaptability is crucial, and RentSpot: Home Near CUET appears to have successfully achieved this, demonstrating attentiveness to the changing needs of its users.

Acknowledgement

We want to express our heartfelt gratitude for the tireless dedication and precious time invested by our course teachers.
Mir. Md. Saki Kowsar
Assistant Professor
Dept. of CSE, CUET
Moumita Sen Sarma Lecturer
Dept. of CSE, CUET
Their valuable guidance and feedback have helped us in completing this project.

References

[1] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*

[2] "Soft Roboties", `https://www.softroboties.com.bd/`.

[3] "Swagger",*https://swagger.io/*

[4] GeeksforGeeks, `https://www.geeksforgeeks.org/software-engineering-architecture`