

**CS673 Software Engineering**

**Team 2 - Rental Ninja**

**Software Design Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Yongjing Wu | Team Lead | *Yongjing wu* | 09/08/2024 |
| Jiacheng Ding | Security Lead | *Jiacheng Ding* | 09/08/2024 |
| Yueyang He | Design and Implementation Lead | *Yueyang He* | 09/08/2024 |
| Xueqi Zhou | Requirement Lead | *Xueqi Zhou* | 09/09/2024 |
| Rundong Zhong | Configuration Lead | *Rundong Zhong* | 09/08/2024 |
| Xiang Zhang | QA Lead | *Xiang Zhang* | 09/09/2024 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **v0.1** | **Rundong Zhong**  **Yueyang He**  **Jiachen Ding** | **09/09/2024** | **Initial draft** |
| **v0.2** | **Xiang Zhang** | **10/04/2024** | **Update** |

[Introduction](#_heading=h.gjdgxs)

[Software Architecture](#_heading=h.30j0zll)

[Class Diagram](#_heading=h.1fob9te)

[UI Design (if applicable)](#_heading=h.3znysh7)

[Database Design (if applicable)](#_heading=h.2et92p0)

[Security Design](#_heading=h.tyjcwt)

[Business Logic and/or Key Algorithms](#_heading=h.3dy6vkm)

[Design Patterns](#_heading=h.1t3h5sf)

[Any Additional Topics you would like to include.](#_heading=h.4d34og8)

[References](#_heading=h.2s8eyo1)

[Glossary](#_heading=h.17dp8vu)

# Introduction

In this section, we provide an overview of the front-end design goals for the rental website. This project aims to provide users with a convenient rental experience, allowing them to browse rental listings, view rental information, and filter listings based on various criteria, collect rental information, and user posted list.

# Software Architecture

**Frontend design:**

The front-end is built using the Vue 3 framework, with Vue Router managing page navigation and state management between components. The main components include:

HomePage.vue: Displays rental listings, filters, and the navigation bar.

Listing.vue: Shows a detailed list of all rental properties, with filters for users to narrow down their searches.

DetailPage.vue: Displays detailed information about a single property and allows users to favorite listings.

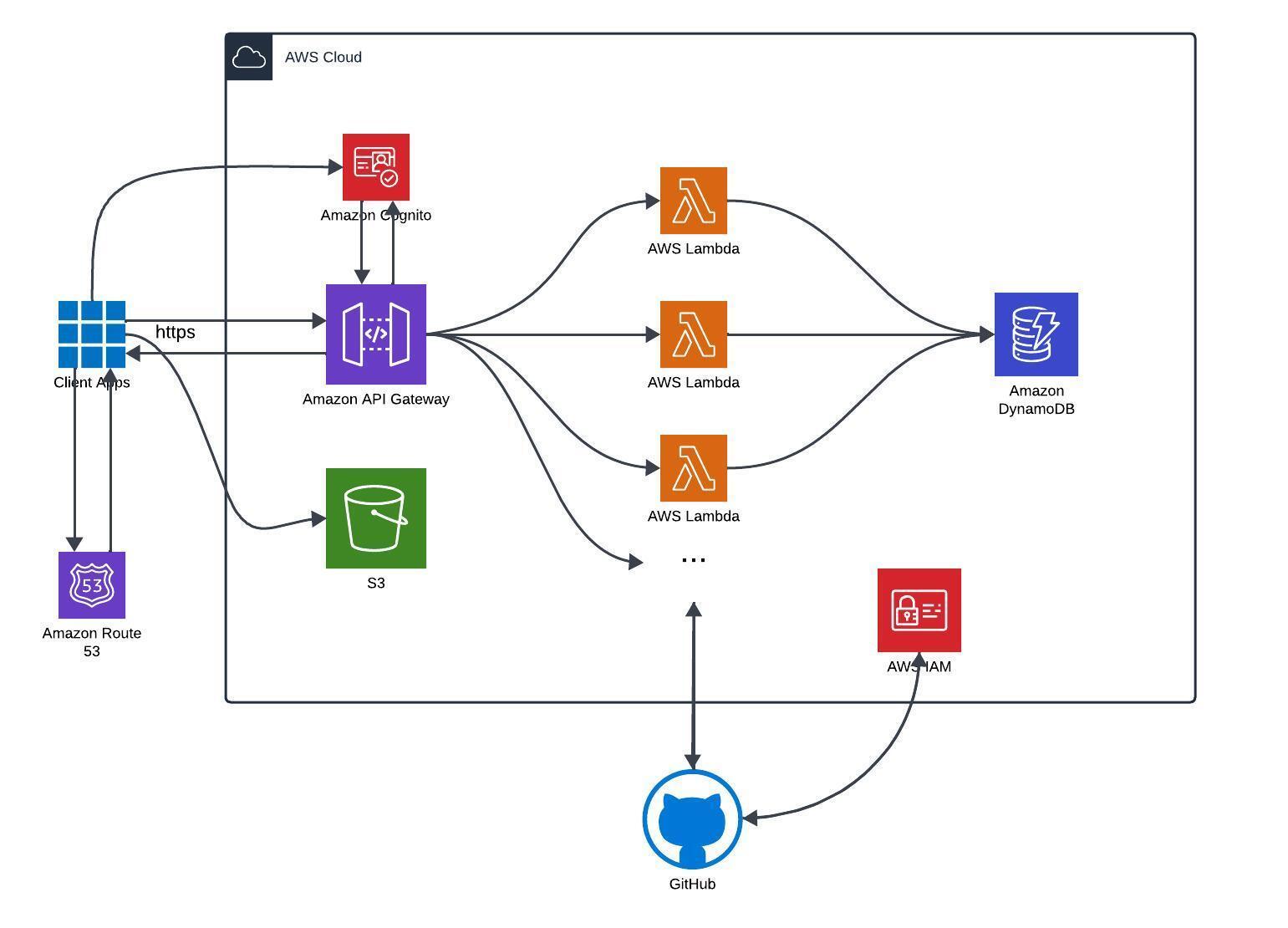
LandlordPage.vue: (Placeholder) Displays landlord information and a list of properties they manage.

Navbar.vue and Sidebar.vue: Top navigation bar and sidebar for global navigation and page switching.

The architecture diagram shows the relationships between components, with HomePage.vue, Listing.vue, and DetailPage.vue being the primary pages, while other components provide global support.

**Backend Design**

**Solution Architecture Diagram:**

****

The system follows a serverless microservices pattern using AWS Lambda. AWS API Gateway is the entry point, routing requests to Lambda functions, keeping services modular and scalable.

**Programming language:**

We will use java17 as our primary programming language, we will use amazon corretto 17 distribution.

**Solution Architecture:**

We are adopting a serverless architecture powered by AWS Lambda for executing business logic, with Amazon Route 53 serving as our DNS provider and domain registrar. AWS API Gateway will manage all incoming client traffic, routing requests to Lambda functions that integrate with Amazon S3 for object storage and Amazon DynamoDB for metadata management. This setup ensures a scalable, efficient, and cost-effective solution for handling both dynamic content and data storage. We will also integrate action to build our CI/CD pipeline.

**Deployment platform**:

Amazon Web Services

**Tools & services involved:**

Lambda, Cognito, Dynamodb, API Gateway, Route53, ACM.

**Backend APIs**:

1. Get pre-signed url
   1. Client will request for a pre-signed url for uploading image
2. upload image
   1. Upload post image once at a time
3. upload post
   1. post to the website
4. get posts list
   1. Main page get the post list based on the key word
5. get specific post info
   1. Get detail of a post
6. delete post
7. log in/sign up
8. get my post list
   1. Get my post list in my info page

# Class Diagram

**Frontend**:

The front-end consists primarily of Vue components. Below is a brief description of the class diagram for key components:

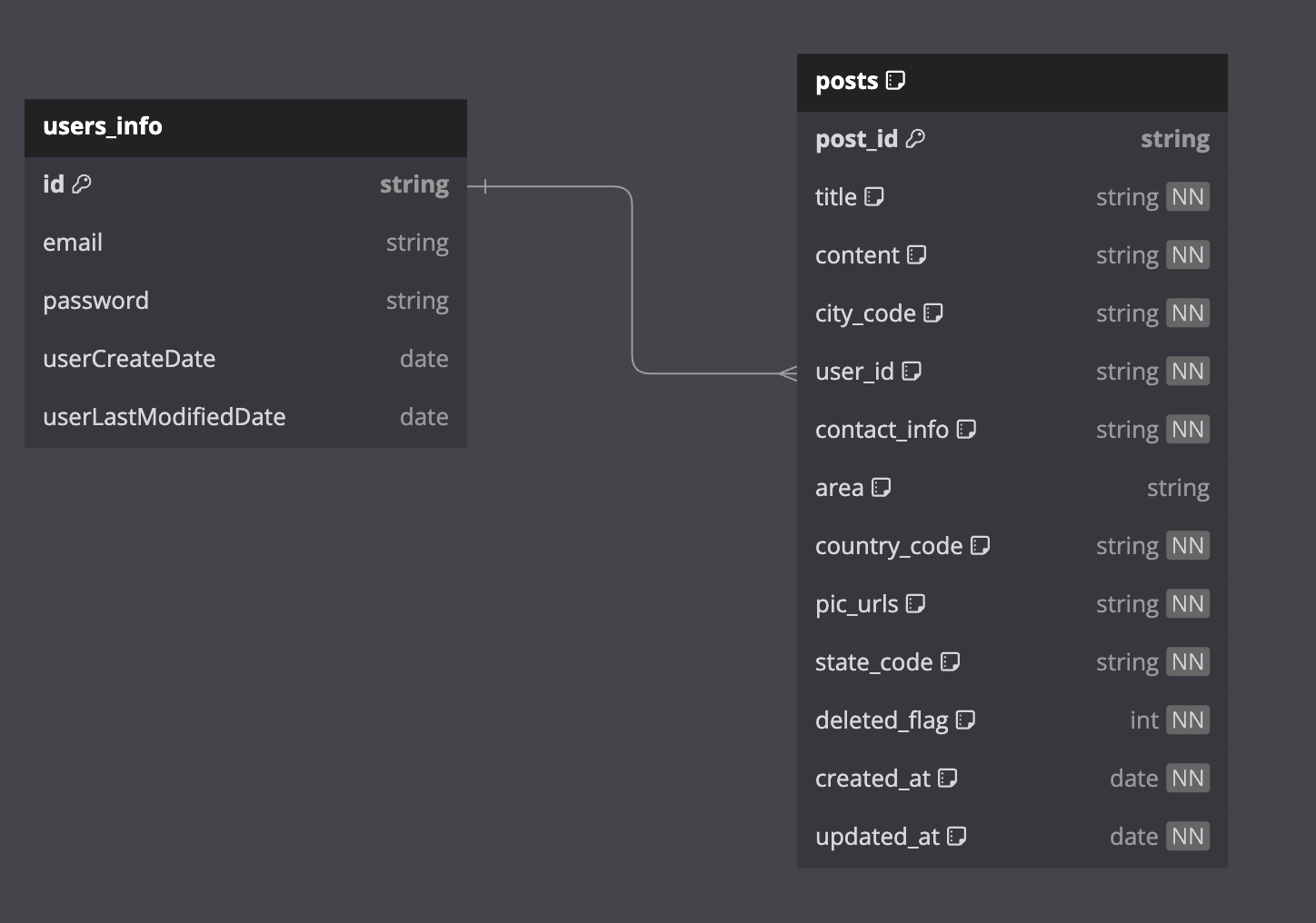
HomePage.vue: Calls the Listing.vue component to display the homepage.

Listing.vue: Responsible for fetching and displaying the rental listings in a grid, with filters for user interaction.

DetailPage.vue: Shows detailed information for a single listing, integrates the favorite feature, and links to LandlordPage.vue.

LandlordPage.vue: (Placeholder) Displays landlord information and properties managed by the landlord (properties may be displayed using Listing.vue).

**Backend UML diagram:**



# UI Design (if applicable)

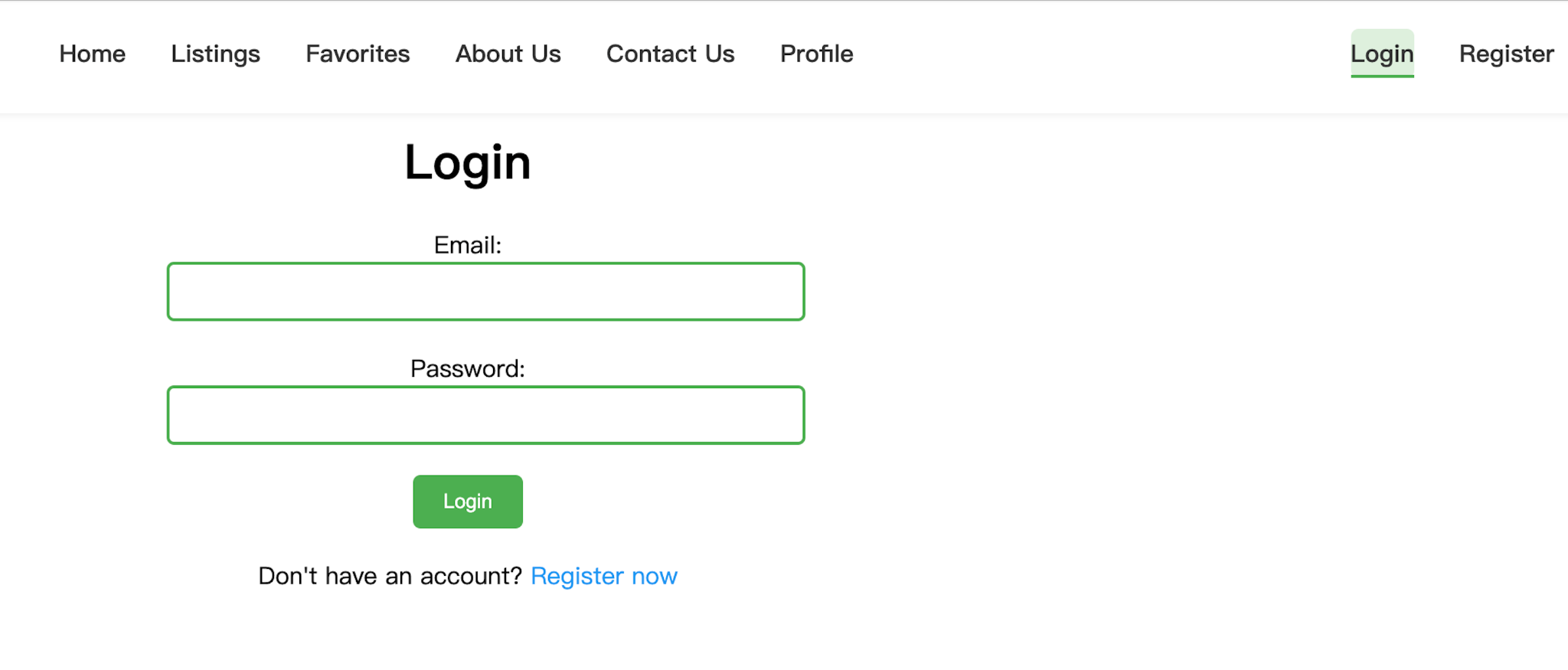
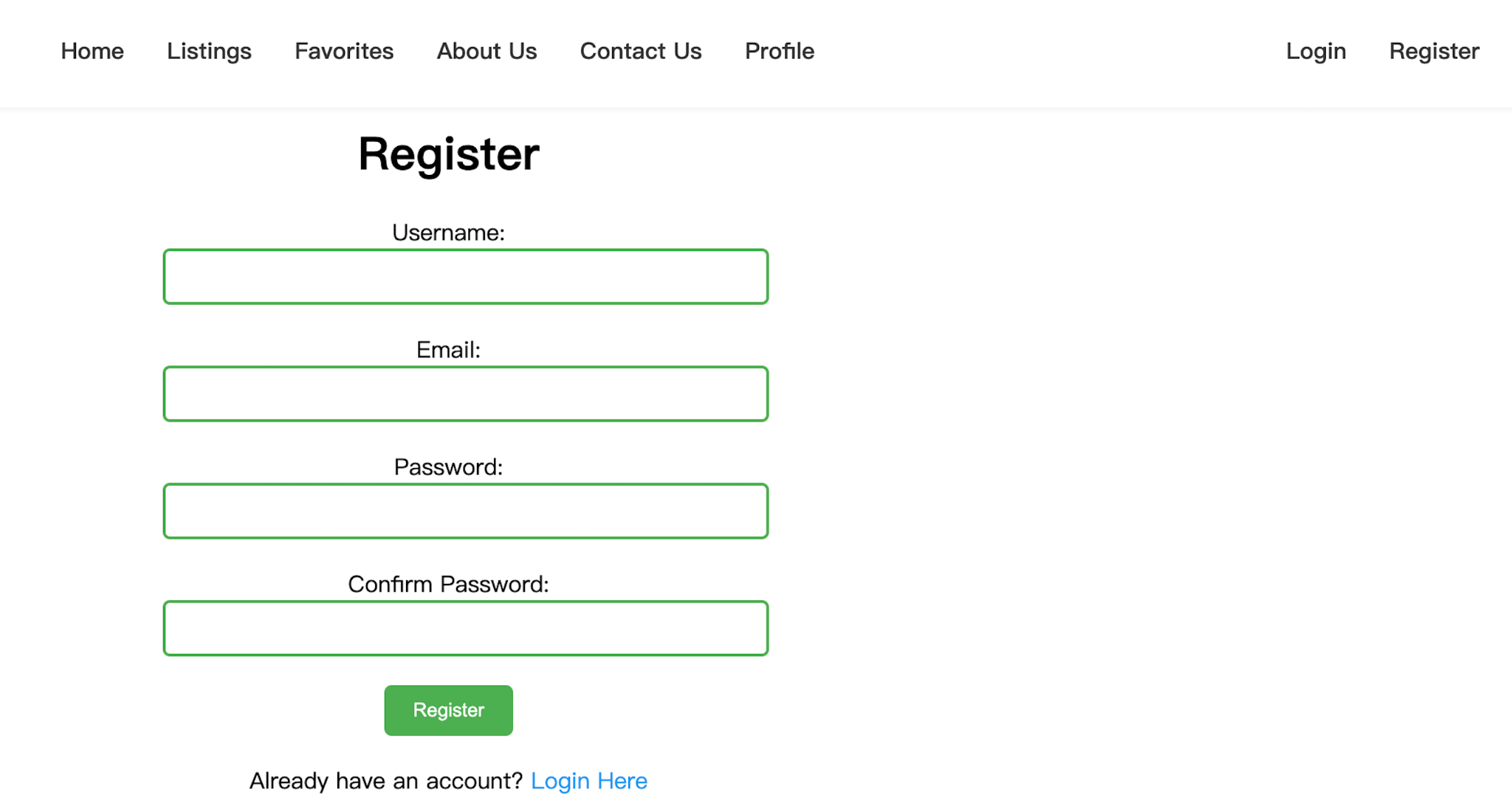
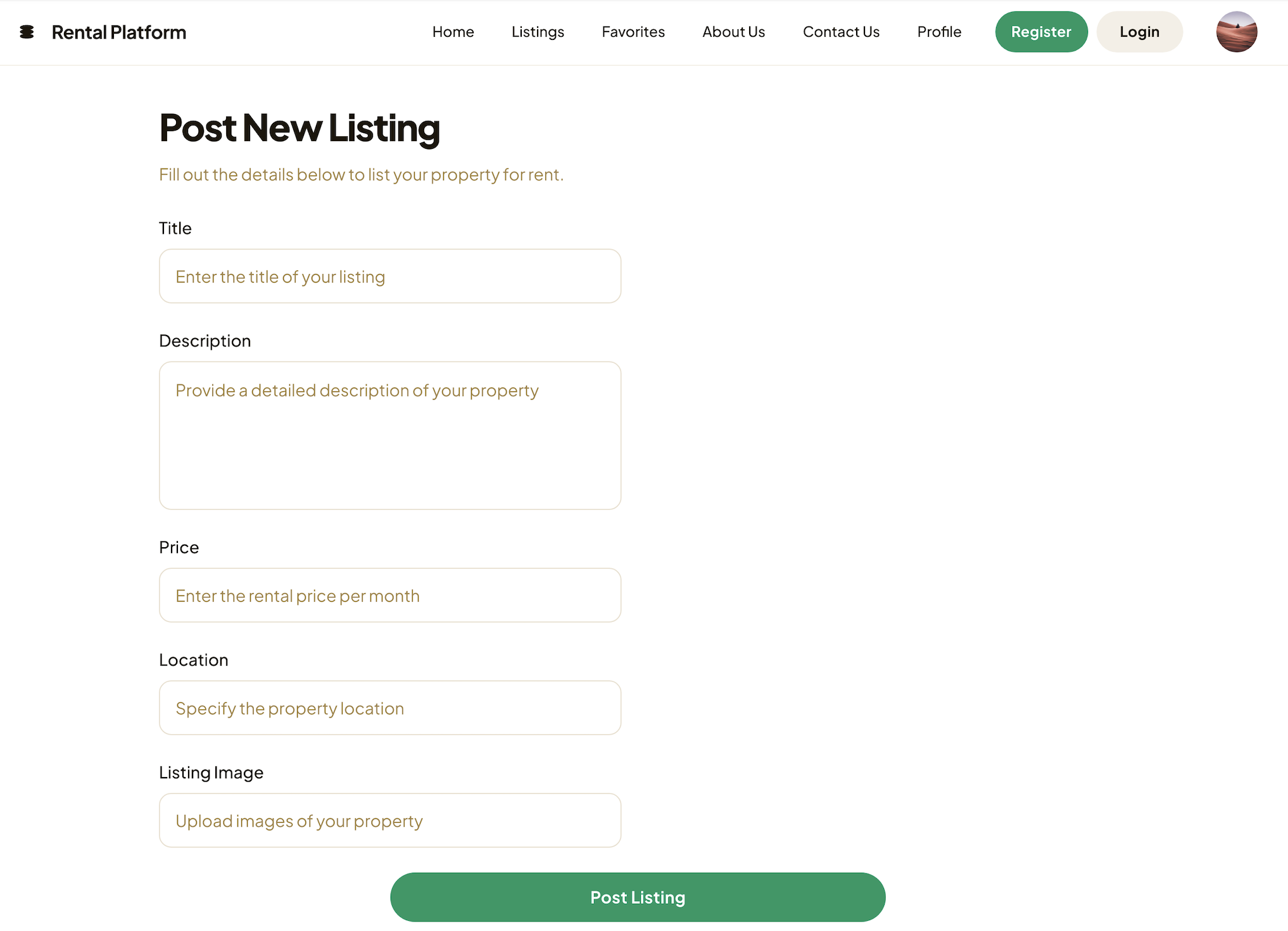
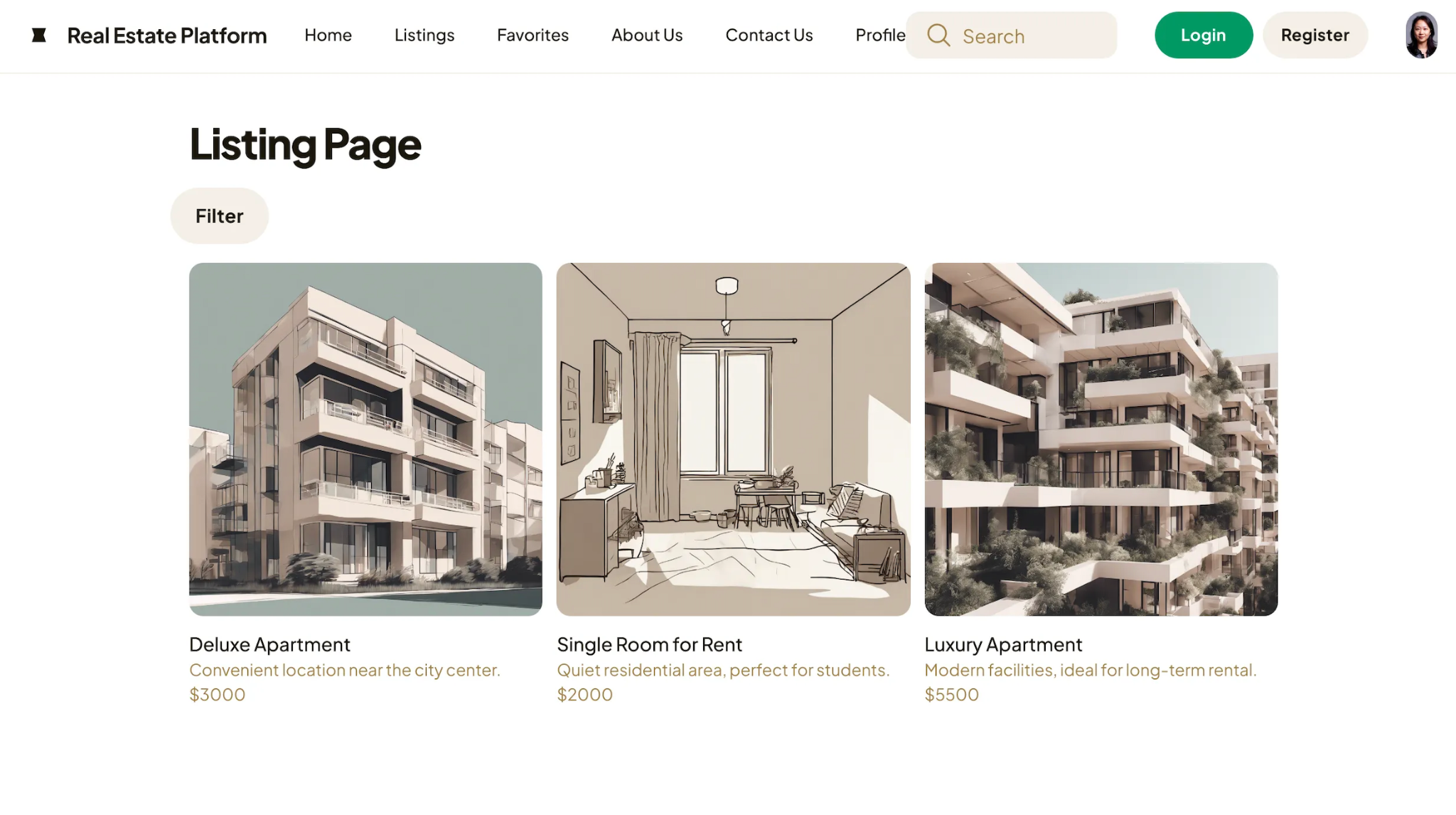
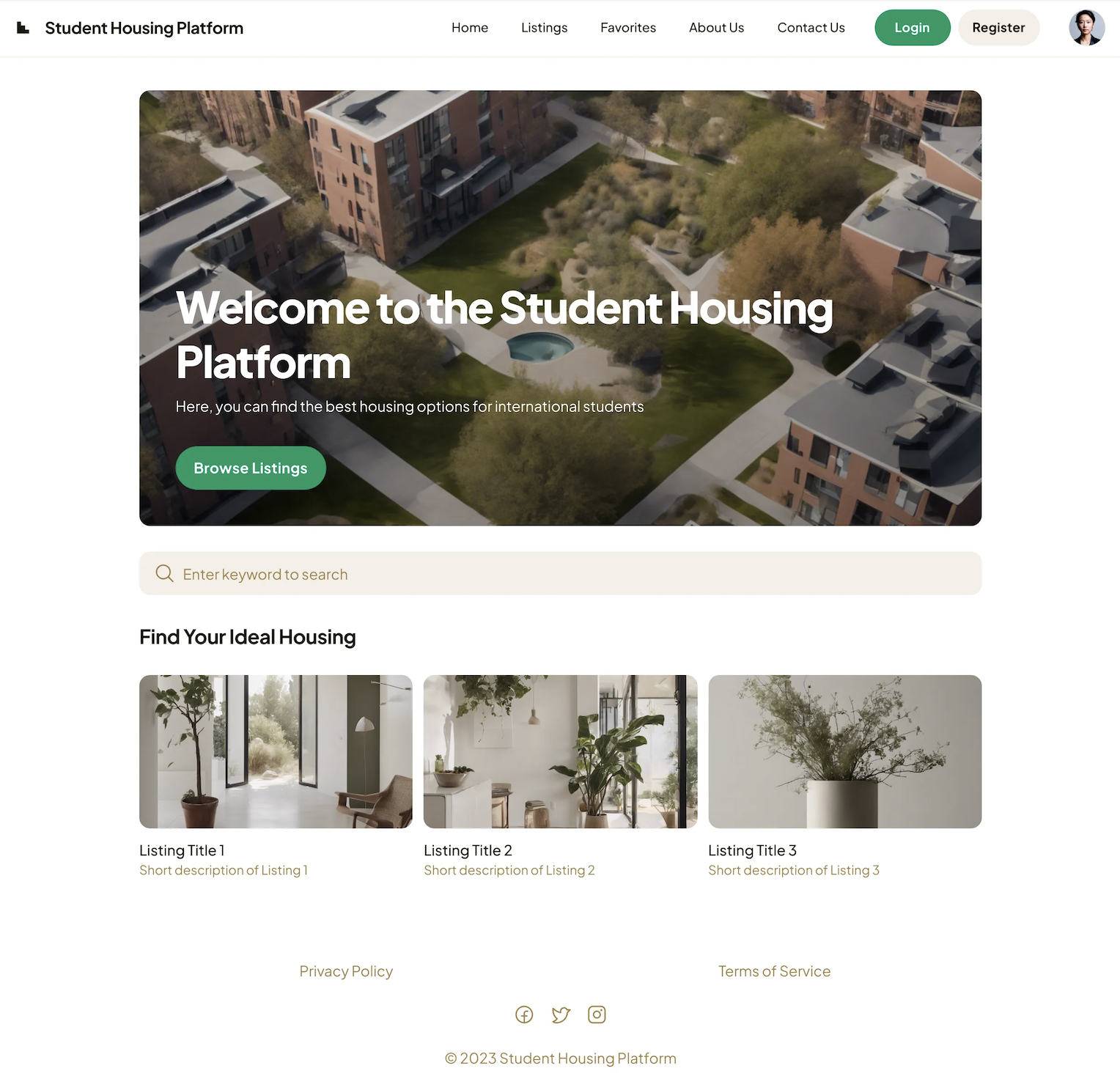
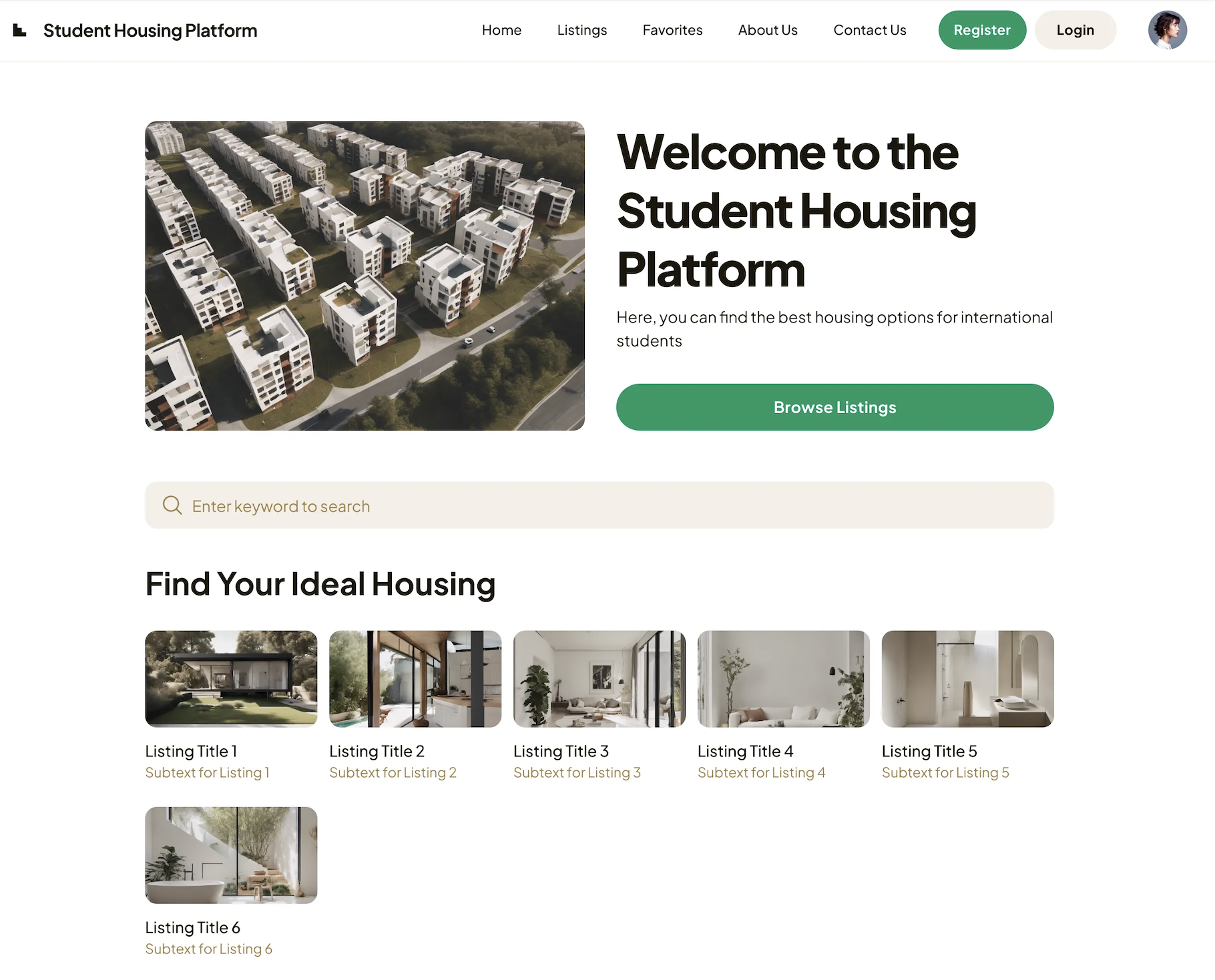
The front-end design is simple and user-friendly, focusing on the user experience. Below is a description of the UI for key pages:

Homepage: Includes the navigation bar, rental listings, and filter options. Users can filter the listings by different criteria.

Listing Page: Displays all available rentals. Users can filter the properties based on price, type, and location.

Detail Page: Shows detailed information about a specific rental, including a favorite option, and users can navigate to the landlord’s profile page.

Landlord Page (Placeholder): Displays landlord information and the listings they manage.



# Database Design (if applicable)

We will use S3 as our object storage and Dynamodb as our metadata storage.

Dynamodb will have one table named post the attributes in the table will have:

"post\_id"

"update\_time"

"area"

"city\_code"

"contact\_info"

"content"

"country\_code"

"create\_time"

"delete\_flag"

"pic\_urls"

"state\_code"

"title"

"user\_id"

# Security Design

1. **Authentication and Authorization**: Integration with the User Management Service will ensure secure user authentication and enforce Role-Based Access Control (RBAC). Stateless authentication will be implemented using JSON Web Tokens (JWT), which will eliminate the need for session storage while maintaining secure access control. JWTs will be signed and validated to ensure their integrity, preventing unauthorized access.
2. **Rate Limiting and Throttling**: To protect the system against Denial of Service (DoS) and Brute Force attacks, rate limiting mechanisms will be implemented. These will restrict the number of requests that can be made by a user or IP address within a defined time frame, reducing the risk of system overload or unauthorized access attempts.
3. **Secure Communication**: All communication between the Project Management Service and external systems will be secured via HTTPS and encrypted using TLS/SSL. This ensures data integrity and confidentiality during transmission, preventing interception and tampering by attackers through man-in-the-middle attacks.
4. **Data Backup and Recovery Plans**: Sensitive data will be regularly backed up in an encrypted format, ensuring secure storage. In the event of a data breach or accidental loss, encrypted backups can be used to quickly restore operations. A comprehensive Disaster Recovery Plan will be in place to minimize downtime and ensure business continuity.
5. **Access Logging and Monitoring**: Beyond audit logging, continuous monitoring of access patterns will be employed to detect suspicious activities in real-time. Anomaly detection tools will flag abnormal behavior and potential security breaches as they occur, allowing for timely interventions. Monitoring will include analyzing logs for unusual login attempts, unauthorized data access, or unexpected system activities.

# Business Logic and/or Key Algorithms

## 1. Overview of Business Logic

**1.1 System Objective:**

The rental website connects students with available rental properties and facilitates the management of rental information, including property listings, user inquiries, and lease agreements.

**1.2 Core Functions:**

Identify the core functions of the system, such as:

Property Listing and Search

User Registration and Authentication

Rental Application and Lease Management

Performance Optimization

## 2. Key Algorithms

**2.1 Property Search Algorithm**

This algorithm helps users find rental properties based on their criteria.

Input: Search parameters (e.g., location, price range, type of property)

Output: List of properties matching the criteria

**2.2 User Authentication Algorithm**

This algorithm handles user login and authentication.

Input: Username and password

Output: Authentication result (success/failure)

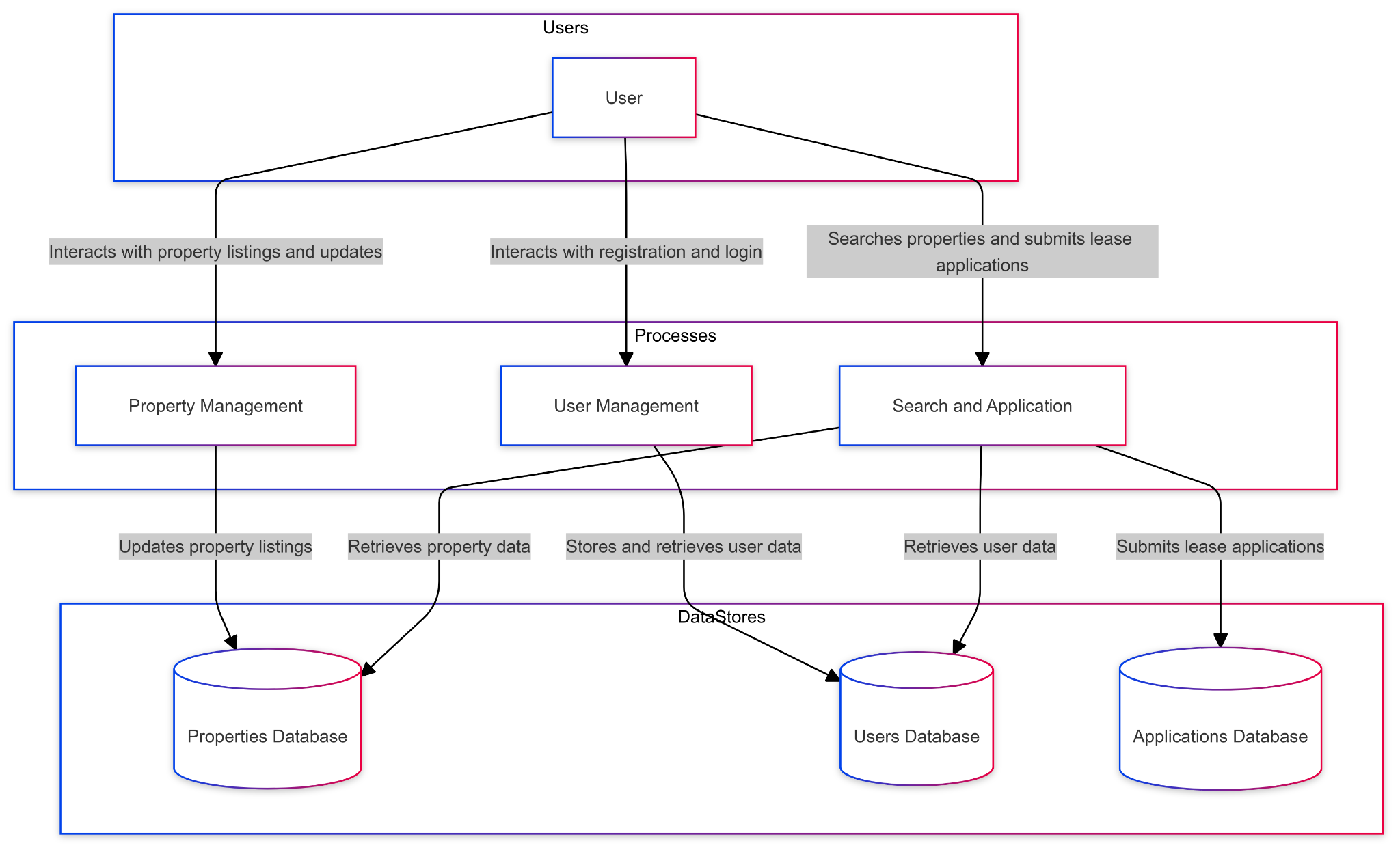
**2.3 Rental information Algorithm**

This algorithm processes rental information from users.

Input: User details, property details, application data

Output: Application status

## 3.Data Flow Diagrams (DFD)



**3.1 Context Diagram (Level 0)**

Purpose: Provides a high-level overview of the system and its interactions with external entities.

**Components**:

External Entities: Users, Database

Processes: Property Search, User Authentication, Lease Application

Data Stores: Properties Database, Users Database

**Data Flows:**

Users input search criteria, login details, or lease applications.

System retrieves data from and stores data in the databases.

**3.2 Level 1 DFD**

Purpose: Breaks down the main processes into more detailed components.

**Processes:**

Property Management: Handles property listings and updates.

User Management: Manages user registration, authentication, and profiles.

Search and Application: Facilitates property search and lease application processes.

**Data Stores:**

Properties Database: Stores property listings and details.

Users Database: Stores user profiles and authentication data.

Applications Database: Stores lease applications and statuses.

**Data Flows:**

Users interact with Property Management for listings and updates.

Users interact with User Management for registration and login.

Users use Search and Application to search for properties and submit lease applications.

## 4. Sequence Diagrams

## 

**4.1 Property Search Sequence Diagram**

Actors: User, Frontend, Backend, Database

Flow:

* User enters search criteria.
* Frontend sends search request to Backend.
* Backend queries the Database for matching properties.
* Database returns the search results to Backend.
* Backend sends the results to Frontend.
* Frontend displays the search results to the User.

**4.2 User Authentication Sequence Diagram**

Actors: User, Frontend, Backend, Database

Flow:

* User enters login credentials.
* Frontend sends credentials to Backend.
* Backend retrieves user data from Database.
* Backend verifies the credentials.
* Backend sends the authentication result to Frontend.
* Frontend displays the result to the User.

**4.3 Rental Information Sequence Diagram**

Actors: User, Frontend, Backend, Database

Flow:

* User submits rental information.
* Frontend sends data to Backend.
* Backend processes the application and checks eligibility.
* Backend stores the application data in the Database.
* Backend sends the application status to Frontend.
* Frontend displays the status to the User.

# Design Patterns

We use the **API Gateway pattern** to manage all client requests, providing a single entry point and simplifying microservice integration. It also handles authentication and response aggregation.

We also use a **Microservices pattern** to split the system into loosely coupled components, each with a specific role, such as property listing, user authentication, or rental processing.

# Any Additional Topics you would like to include.

**Landlord Page Placeholder**: The specific functionality and implementation of this page will be finalized in future iterations.

# References

# Glossary