Student Housing System

CSC640 Fall 2023

**Team Members**

* Varsha Annamalai
* Sony Annem
* Mostafa Abdelmegeed
* Keerthi Devireddy

# Preface

Welcome to the technical documentation for the Student Housing System web application. This document serves as a comprehensive guide for all stakeholders involved in the development, implementation, and maintenance of this innovative solution designed to simplify the process of finding suitable housing for students in our community.

As the project progresses, this document may be subject to updates and revisions to reflect changing requirements, technology updates, and stakeholder feedback.

# Purpose

This document provides a detailed overview of the Student Housing System, outlining its objectives, functionalities, and the requirements necessary for its development. It serves as a comprehensive guide for the team members in the project.

# Introduction

When entering a new college town, students often find themselves grappling with many housing-related dilemmas. One of the foremost issues is the trade-off between proximity to the campus and the cost of rent. Many students wish to be as close as possible to their academic institutions for convenience, yet they are often confronted with exorbitant rental rates in such prime locations. This financial strain can significantly impact a student's budget and overall college experience. Conversely, opting for more affordable housing options on the outskirts of town may lead to different complications. Safety becomes a paramount concern, as these areas may not offer the same security and infrastructure as those closer to campus. Students may find themselves navigating unfamiliar neighborhoods with uncertain reputations, raising apprehensions about personal safety and well-being.

Furthermore, the process of searching for housing can be overwhelming due to the sheer volume of available options, each with its unique set of pros and cons. Without a reliable source of information, students may struggle to differentiate between housing alternatives, often relying on word-of-mouth recommendations or unverified online reviews, which can be unreliable and biased. Hygiene and cleanliness are additional aspects that can become problematic. Students may encounter housing that appears suitable on the surface but lacks proper maintenance and cleanliness standards, leading to unpleasant living conditions that can impact their academic performance and overall well-being.

Considering these challenges, our software solution emerges as a valuable resource for students. It empowers them with a comprehensive and unbiased overview of available housing options, considering factors like cost, proximity, safety, and cleanliness. By harnessing the collective wisdom of Google reviews and the expertise of specialized assessors, our platform aims to alleviate the housing-related stress that students face, ensuring they can make well-informed decisions that align with their unique needs and circumstances.

This document will provide a comprehensive understanding of the system's functionalities, user roles, and the specific requirements necessary to bring the Student Housing System to life. It includes detailed descriptions of each module, use case scenarios, and system architecture to ensure successful project planning and development.

In the following sections, we will delve into the specific requirements for each module, detailing user interactions, system behaviors, and the underlying data structures. The goal is to create a robust and user-centric platform that simplifies the student housing experience, making it easier for students to find suitable accommodations and landlords to manage their properties efficiently.

# Glossary

Hereby after in this document **Admin** and **Reviewer** terms will be relating to the same actor, the justification for using these two terms for the same actor was that in some use cases “Reviewer” was descriptive of the nature of the use case.

The terms **Property**, **Listing**, and **Unit**, all relate to the same thing.

# Scope

The Student Housing System encompasses four key modules:

|  |  |
| --- | --- |
| Module | Assignee |
| Admin/Reviewer | Keerthi Devireddy |
| User | Sony Annem |
| Landlord | Varsha Annamalai |
| Google API | Mostafa Abdelmegeed |

# Use Cases

## Use Case 1

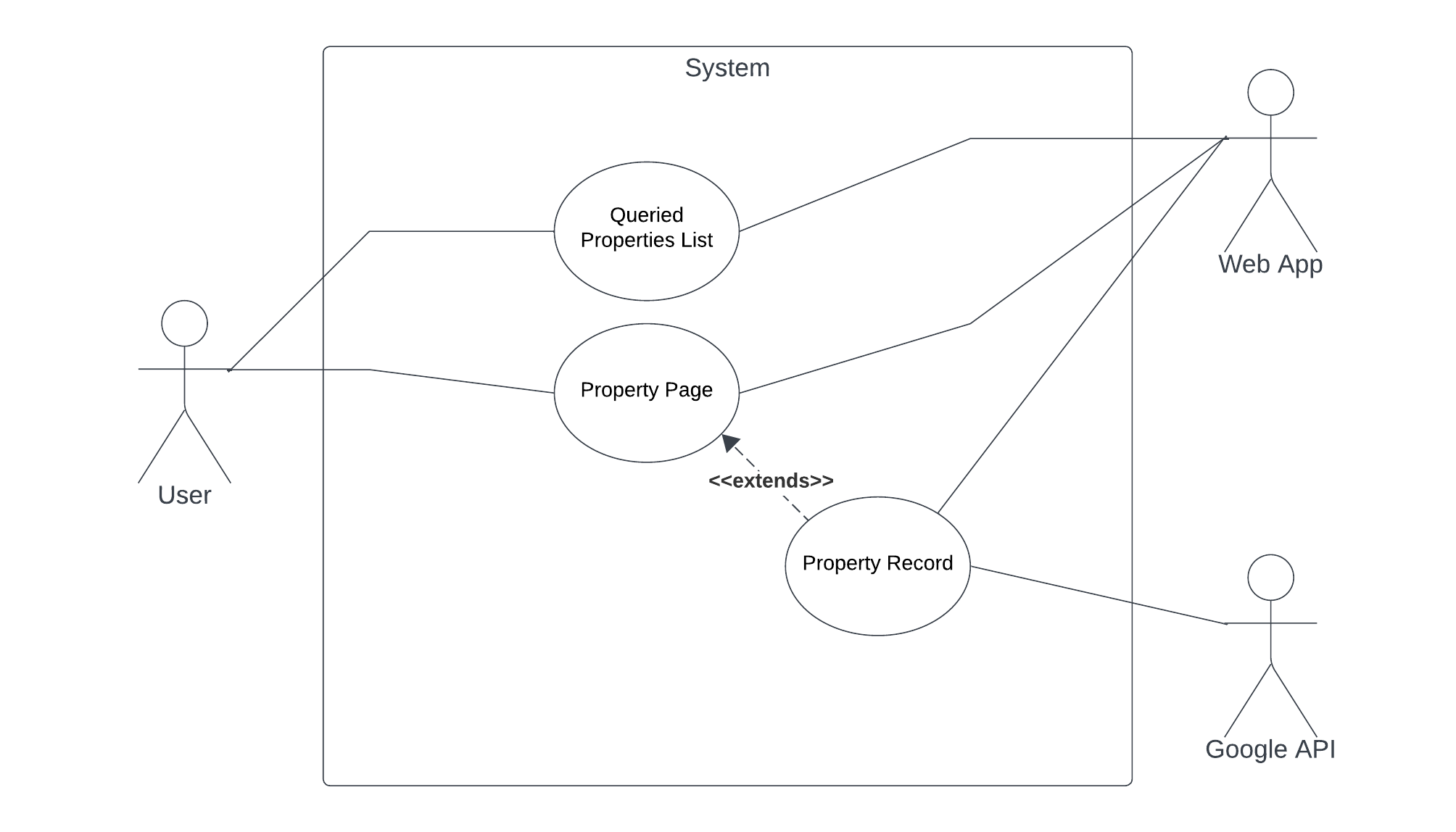
[](https://lucid.app/lucidchart/2e9707c3-e0ec-4b8a-a293-493d5766fa87/edit?crop=content&page=0&signature=76d850f120c99ead2a9aa2bb784fcaad81b4921b7d178aeb17960e5d6b4fad75)

Figure 1 Use Case Diagram by Mostafa A.

### Description

A user starts utilizing the system by triggering a search query, the web application is responsible for retrieving and showing the search results as a list of records for the user.

The user can choose one of the search results records to get a detailed description of the property selected, at the user’s request, the web application starts retrieving the necessary details from the database, and communicate with Google API, Google API compliments the retrieved data with any additional information about the property (e.g., location, reviews, ratings), the final page is sent to the user by the web application.

### Scenario

#### Initial Assumption

User knows the region of interest to search for housing information in.

#### Normal

The user will enter the region of interest that they want to search for housing options at, this could be a city, town, or college, once this is done, the web application queries the database using the input from the user, and retrieves a list of all possible properties, while doing that, the web application may also retrieve data from Google API to show on the property page.

On completion of the retrieval process, a detailed page of information about the property is shown to the user.

#### What can go wrong?

The search input keywords are non-existent in the database, or are not a known region, then a message will be shown to the user telling them there is not enough data about this region. The user then should be asked if they want to continue navigating other regions or if it was a typo from their end.

Google API could have little to no information about the selected property, if such scenario occurs, no information from Google API will be inserted, and only the information from the database is retrieved, while showing a message to the user that this property no longer exists on google maps, and a notification should be sent to the property owner to update property specifics on google maps.

#### System State on Completion

A retrieved informative property page is shown to the user.

## Use Case 2

A diagram of a user

Description automatically generated

Figure 2: Use Case Diagram by Sony Annem

## Use Case 3

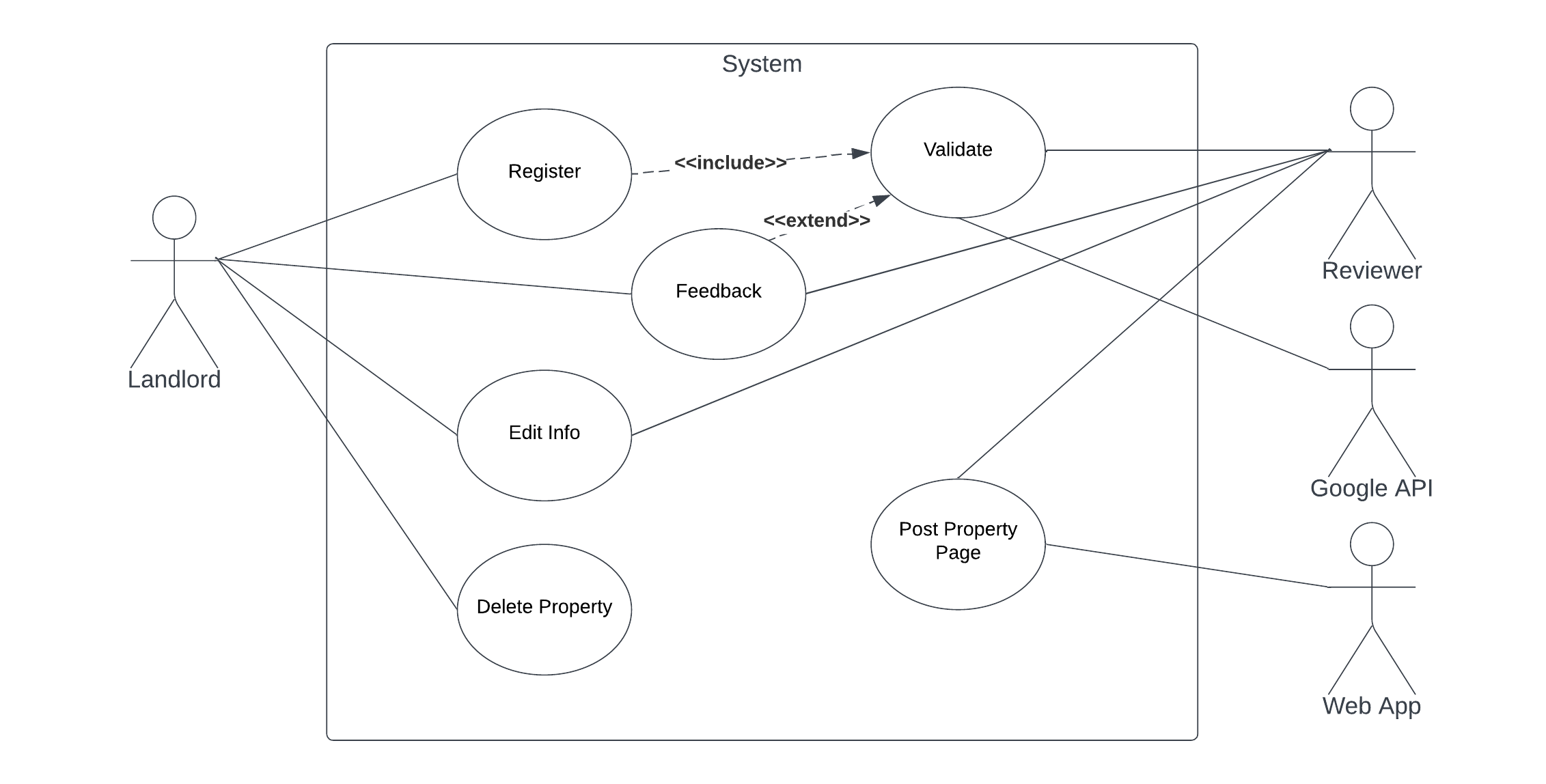
[](https://lucid.app/lucidchart/2460b50c-3c7d-4924-804a-1f2b8643f486/edit?crop=content&page=0&signature=8e1fd6639b52404defff58a0526d8ca0a3c7b67d8b92031023a44cf9d77be5ed)

Figure 3: Use Case Diagram by Mostafa A.

### Description

The Landlord can register their property on the platform; however, a reviewer must confirm the information to be posted. First the landlord registers by submitting a form. The Reviewer then starts the validation process by cross checking the details sent by the landlord and Google API.

If the request were denied, the Reviewer will send detailed feedback to the Landlord, on what was missing, and what could be done. If it was accepted, it will be posted to the system by the reviewer.

The Landlord can also add edits to their property page (if existing); however, it will still need approval from the Reviewer to be posted to the property page.

The Landlord can delete their property page without prior notice to the platform.

### Scenario

#### Initial Assumption

Landlord is a registered user on the system and has the required information to submit a form to the reviewer.

#### Normal

The landlord starts by initiating a request to submit their property on the system, the form should indicate all required information, from address, images, google maps URL, amenities, etc.

Once the form is filled and submitted, it is passed to the reviewer, whom is responsible for validating the data, and cross checking with google maps, once the information is validated, the property could be posted on the system; however, if there are any missing, or incorrect information, a feedback is sent back to the landlord, and the property is waitlisted until the landlord return with the missing information.

If the landlord already has registered properties, they can edit the information for such properties; however, the updated page will not be posted until the added information is validated by the reviewer.

Lastly, a landlord with properties registered can delete any posted property on demand.

#### What can go wrong?

Overflowing of requests, and incomparable number of reviewers could cause a bottle neck in approving properties.

If you create multiple pages for the same property, the reviewer then ignores the replicated requests, and only approves one of them.

Multiple trials with false information, reviewer could have the ability to blacklist the landlord.

#### System Stage on Completion

A posted property page on the system, or feedback sent to the landlord.

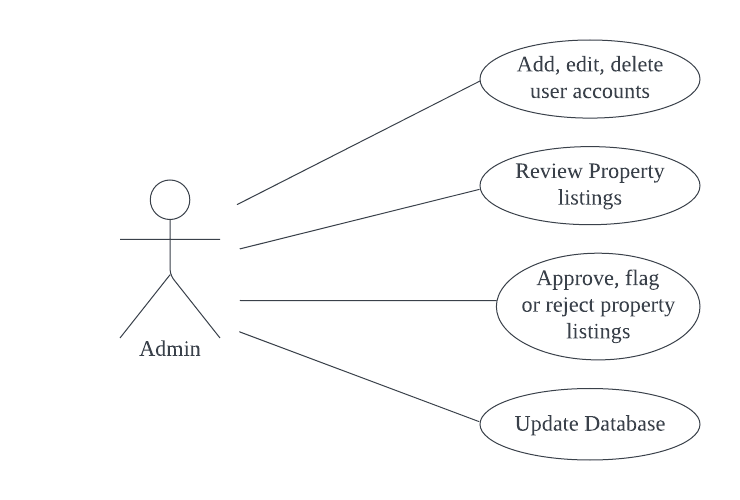
## Use Case 4

A diagram of a person with text

Description automatically generated

Figure 4: Use Case Diagram by Varsha Annamalai

## Use Case 5

  
Figure 4: Use Case Diagram by Keerthi Devireddy

### Description

The admin is responsible for overseeing and managing property listings within the Student Housing System. It has the highest level of access and control. Admin's role in managing property listings is critical to ensuring that the Student Housing System maintains a database of reliable, up-to-date, and policy-compliant property listings for the benefit of both students and property owners. This process helps create a trustworthy and efficient platform for all users.

This use case outlines the actions and responsibilities of the admin in maintaining a high-quality and accurate database of available properties.

Scenario

#### Initial Assumption

The admin is logged into the Student Housing System as an authorized administrator. The system is running without any technical issues or errors. The admin has access to a user-friendly administrative dashboard.

#### Normal

The admin reviews the property listings. The admin checks the accuracy of the property details, such as property name, description, location, amenities, and availability. The admin assesses whether the property adheres to the system’s policies and guidelines.

If a property listing is accurate and complies with policies, the admin approves the listing, making it visible to users. If a listing requires modification, the admin can edit the listing information, if necessary. However, if there are any missing, or incorrect information, feedback is sent back to the landlord, and the property is waitlisted until the landlord return with the missing information.

If a listing contains inaccurate information, the admin communicates with the property owner regarding the issues. The admin may reject the listing if the issues are not resolved.

After approving or making edits to the existing listings, the admin updates the database to reflect the changes.

#### What can go wrong?

Property owners may not respond promptly to admin communications, leading to delays in resolving issues with listings.

The system may encounter technical issues, such as database errors or server downtime, preventing admin from accessing or updating property listings.

#### System Stage on Completion

Upon completion, the system’s database reflects the following:

Accurate and compliant property listings are approved and visible to users. Listings requiring edits have been updated with necessary corrections. Rejected listings or no longer visible to students, pending potential corrections by property owners.

# User Requirements

1. User Registration and Authentication
   1. Users must be able to register using their personal information.
   2. Users should be able to log in securely to access the application’s features.
2. Search and Filtering
   1. Users can search for housing units based on criteria like price range, proximity to campus, number of bedrooms, and amenities.
   2. With a pre-existing database of housings around campuses in the US, a correct query should be sufficient to retrieve multiple entries.
   3. The list of records presented to the user should include simple and clean information about the property (e.g., name, address, price)
   4. Results should be displayed with relevant property details.
   5. Fully verified properties with functional Google API link should be shown on the top of the list.
3. View Property Details
   1. Users can view comprehensive information about a selected property, including photos, contact details for the landlord, and a map showing the property's location.
4. Property Submission
   1. Any registered user can access the platform and submit a request to post their property on the system.
   2. Landlords can submit their property details, including price, location, number of rooms, and amenities.
   3. The portal should provide clear instructions on how to fill the required fields, and the specifications required for images to be registered (size < 2 MB) per image.
   4. The landlord should be able to add multiple amenities in a separate part of the portal, each amenity is selected out of a list.
   5. Landlord should enter a proximity to different campuses around their property in miles.
   6. Prices are presented per month, and any other fees should be included in the overall price.
   7. Submitted properties must be reviewed and approved by an admin or reviewer before being added to the database.
5. Location and Mapping Integration
   1. The application should integrate with Google Maps to display the location of each property on a map.
   2. Users should be able to see the proximity of a property to the campus and other landmarks.
   3. Users can use map-based features like zooming, panning, and street view to explore the neighborhood.
6. Property Reviews and Location Information
   1. Property listings should display Google Maps' location data, including nearby restaurants, grocery stores, public transportation, and educational institutions.
   2. Users can access reviews and ratings for the neighborhood and nearby services using Google Maps data.
   3. In the property detailed page, a couple of reviews from google should be shown to the user, preferably 2 of the highest reviews, and another 2 from the worst reviews.
   4. Use Google API to retrieve average rating of properties and display it next to every property record.
   5. In case Google API does not offer enough information, or no information at all, a notification and an email shall be sent to the landlord to update the record on Google, notifying a landlord give them a month to update their record, or their property will be unverified and will not show as much in queries.
7. Landlord Notification System
   1. Landlord must be informed once their property was removed or no longer accessible via google maps.
   2. A weekly warning will be sent to the landlord in case the property remains inaccessible via google maps.
8. Up to Date System
   1. If a listing remains inaccessible via google maps for more than a month, it should be removed from the system and reported to the landlord.

# Non-functional Requirements

## Product Requirements

* The Student Housing System should be available to all users 24/7, a notice of a week at least should be put on the landing page in case a planned maintenance was coming.
* In case of failures and unplanned downtime, the system should not be down for more than 30 minutes at a time, with an overall total of 5 hours downtime per year.
* Queried results should not take more than 2 seconds to present to the user.
* In case of failures in communication with Google, a suitable reason should be shown to the user wherever possible and visible.
* The system should be designed to handle a growing database of properties and users.
* The user interface should be intuitive and user-friendly.

## Organizational Requirements

* Reviewers base should be diverse when it comes to demographics, trained, and locals to their region whenever possible.
* Landlords will need to authenticate themselves as property owners in cases of conflicts, or when multiple requests are done for the same property.
* Database shall be hosted on a cloud-based service.
* Maintain comprehensive technical documentation for the application's architecture, APIs, and data schema.

## External Requirements

* The system shall implement users’ data privacy provisions, never exposing users' data to landlords without their consent.
* Ensure compliance with local housing regulations and privacy laws.