[Student Property Rental System] Requirements Specification

Version 3.0

March 8th, 2025

Use this Requirements Specification template to document the requirements for your product or service, including priority and approval. Tailor the specification to suit your project, organizing the applicable sections in a way that works best, and use the checklist to record the decisions about what is applicable and what isn't.

The format of the requirements depends on what works best for your project.

This document contains instructions and examples which are for the benefit of the person writing the document and should be removed before the document is finalized.

To regenerate the TOC, select all (CTL-A) and press F9.

**Table of Contents**

[**1. Executive Summary 2**](#_heading=h.tcba24wckjfe)

**1.1 Project Overview 3**

**2. Product/Service Description 3**

**2.1 Product Context 3**

**2.2 User Characteristics 3**

**2.3 Assumptions 3**

**2.4 Constraints Dependencies 3**

**3. Requirements 3**

**3.1 Functional Requirements 4**

[3.2 Non-Functional Requirements 5](#_heading=h.jhhup9emwygu)

**3.2.1 Product Requirements 5**

[3.2.1.1 Usability Requirements 5](#_heading=h.mt5jnhc4pg9t)

[3.2.1.2 Performance Requirements 5](#_heading=h.9ya7za4kkz65)

[3.2.1.3 Availability 6](#_heading=h.uo6rfh7fd0oo)

[3.2.1.4 Security 6](#_heading=h.wxk3jphf8vs)

[3.2.2 Organizational Requirements 6](#_heading=h.6hqsfmx3lh7l)

[3.2.3 External Requirements 6](#_heading=h.y3bcnef30nrz)

**4. User Scenarios/Use Cases 6**

[**5. Diagrams 7**](#_heading=h.3i8jekmg4yzf)

[**6. Design Patterns 7**](#_heading=h.6lf64n1yrkv1)

[**7. Appendix. 7**](#_heading=h.oulty3m9lfpr)

[**Organizing the Requirements 7**](#_heading=h.olbo735dli7o)

# Executive Summary

## Project Overview

Being a student away from home has its hardships: the main one being finding a suitable accommodation. The purpose of this software is to make it easier for students to find accommodation during their academic year. The target users of this software are: students, landlords, and universities.

**Intended audience**

* *Students*: Individuals in need of housing. They can apply for rent, list their current place to share with other students, provide feedback, i.e. a rating of their experience during their stay. They will need to sign a lease agreement and pay the rent using cash or card (however they find it fitting).
* *Landlords*: They are property owners, who will have the option to list such properties for renting. They can also hire a real estate agency to do the listing and connect with the students for them.
* *Real estate agents*: they work for the landlords who hire them and they list and manage their properties, and connect with the students on the landlords behalf.
* *Universities:* A representative of a university will use this software to list free spaces in their dormitories.
* *System Administrator*: Manages user accounts, handles technical issues, and ensures smooth operation.

# Product/Service Description

This software is a web and mobile-based platform that allows students to pick their preferred accommodation from any of the available listings. The actors that will offer the accommodation are landlords and university dormitories. The students will utilize this software to connect with these actors.  
It is also connected to google maps API to verify available properties. Adobe Sign will be necessary for the e-signature for the contract.

## Product Context

This product is dependent on payment gateways such as PayPal and direct bank communications, i.e. card payments. Another dependency is email & SMS notification services for logging in and communication between actors. The last dependency is the SheerID API which confirms the existence of a student email for the student users, preventing other random people from using this service.

## User Characteristics

### 1. Student (Tenant)

* Role: Primary user renting housing.
* Experience: Likely first-time renters, may not be familiar with lease agreements.
* Technical Expertise: Moderate – comfortable with mobile apps/web platforms.
* General Characteristics:
  + Seeks affordable and convenient housing close to university.
  + May need roommate-matching options.
  + Prefers easy online payment and contract signing.
  + May require flexible lease terms (semester-based).

### 

### 

### 

### 

### 2. Landlord (Property Owner)

* Role: Lists and manages student properties.
* Experience: Varies – could be individual landlords or professional property managers.
* Technical Expertise: Low to moderate – some may struggle with online platforms.
* General Characteristics:
  + Wants an easy way to list and manage properties.
  + Prefers automated rent collection and contract management.
  + Needs tenant verification to avoid unreliable renters.
  + Interested in communication tools for tenant interactions.

### 3. University Administrator (Optional)

* Role: Verifies student status and may oversee university-affiliated housing.
* Experience: Administrative and database management experience.
* Technical Expertise: Moderate to high – familiar with student records systems.
* General Characteristics:
  + Ensures housing availability meets student needs.
  + Prefers seamless integration with university databases.
  + Needs tools for reporting and monitoring housing trends.

### 4. Real Estate Agent (Optional)

* Role: Manages property listings on behalf of landlords.
* Experience: Professional property management experience.
* Technical Expertise: Moderate to high – familiar with CRM and real estate platforms.
* General Characteristics:
  + Wants a dashboard to manage multiple properties.
  + Needs marketing tools to attract student renters.
  + Prefers analytics for pricing and rental demand insights.

### 5. System Administrator

* Role: Manages the technical infrastructure of the platform.
* Experience: IT or software engineering background.
* Technical Expertise: High – proficient in database, server management, and security.
* General Characteristics:
  + Monitors and maintains system performance.
  + Handles security, user authentication, and bug fixes.
  + Ensures uptime and smooth operation of integrations (payments, email, etc.).

## Assumptions

### General Assumptions:

1. The system will primarily be used by students looking for short-term or long-term housing near universities.
2. Landlords or property owners are willing to rent exclusively to students.
3. Users will access the system via a web platform and/or mobile application.
4. The system will support multiple universities but may require manual verification for new institutions.
5. Only verified students (with a university email or other validation) can book a rental.

### User Assumptions:

1. Students may not have prior renting experience, so the system should be easy to use.
2. Landlords may have varying levels of technical expertise, requiring a user-friendly property management interface.
3. University administrators (if involved) will have limited interaction, mainly for verification purposes.

### Functional Assumptions:

1. The system will support direct messaging between landlords and tenants.
2. Payments will be handled through a third-party payment gateway (e.g., PayPal, Stripe).
3. Lease agreements will be managed digitally, with e-signature support.
4. Users will be able to filter listings based on price, location, amenities, and availability.
5. Reviews and ratings will be implemented to ensure trust between landlords and students.

### Technical Assumptions:

1. The platform will store data securely in a cloud-based database.
2. The system will be scalable to support multiple users and properties.
3. User authentication will be implemented using email/password login and possibly social logins (Google, university email, etc.).

### Legal & Business Assumptions:

1. The system will comply with data privacy laws (e.g., GDPR, if applicable).
2. The platform will not be responsible for disputes between landlords and students but will provide tools for reporting issues.
3. Landlords are responsible for the accuracy of their property listings.

## Constraints and Dependencies

### 1. Business Constraints

* The system must cater primarily to students, meaning rental agreements, pricing, and features must be designed with student needs in mind.
* Only verified students will be allowed to book properties, limiting the system’s user base.
* Landlords must comply with local rental regulations, but enforcement is beyond the system’s scope.
* The system does not handle legal disputes between landlords and tenants—only facilitates reporting issues.
* Revenue model constraints (e.g., subscription-based for landlords, commission on bookings, or a free model with optional premium features).

### 2. Technical Constraints

* The system must support both desktop and mobile interfaces, but initial development may focus on web-first implementation.
* A third-party payment gateway (e.g., Stripe, PayPal) will be used instead of an in-house payment system.
* Cloud-based infrastructure will be used for scalability, but high-traffic scenarios need to be planned for.
* Integration with university databases for student verification is dependent on university cooperation and API availability.
* The system must ensure data privacy and comply with GDPR or other applicable regulations.
* User authentication should support email verification and optionally social logins (Google, university email).

### 3. Operational Constraints

* The system should be available 24/7, requiring regular maintenance schedules to minimize downtime.
* Customer support availability may be limited, requiring an automated help system or chatbot for basic inquiries.
* The number of simultaneous users (students searching, booking, landlords managing listings) must be optimized to avoid performance issues.

### 4. Security Constraints

* Users must be authenticated using secure login mechanisms (OAuth, two-factor authentication optional).
* Personal and payment information must be encrypted and securely stored.
* Fraud prevention measures must be in place to prevent fake listings or fraudulent users.

# Requirements

* Describe all system requirements in enough detail for designers to design a system satisfying the requirements and testers to verify that the system satisfies requirements.
* Organize these requirements in a way that works best for your project. See Appendix DAppendix D, Organizing the Requirements for different ways to organize these requirements.
* Describe every input into the system, every output from the system, and every function performed by the system in response to an input or in support of an output. (Specify what functions are to be performed on what data to produce what results at what location for whom.)
* Each requirement should be numbered (or uniquely identifiable) and prioritized.

See the sample requirements in Functional Requirements, and System Interface/Integration, as well as these example priority definitions:

**Priority Definitions**

The following definitions are intended as a guideline to prioritize requirements.

* Priority 1 – The requirement is a “must have” as outlined by policy/law
* Priority 2 – The requirement is needed for improved processing, and the fulfillment of the requirement will create immediate benefits
* Priority 3 – The requirement is a “nice to have” which may include new functionality

It may be helpful to phrase the requirement in terms of its priority, e.g., "The value of the employee status sent to DIS **must be** either A or I" or "It **would be nice** if the application warned the user that the expiration date was 3 business days away". Another approach would be to group requirements by priority category.

* A good requirement is:
* Correct
* Unambiguous (all statements have exactly one interpretation)
* Complete (where TBDs are absolutely necessary, document why the information is unknown, who is responsible for resolution, and the deadline)
* Consistent
* Ranked for importance and/or stability
* Verifiable (avoid soft descriptions like “works well”, “is user friendly”; use concrete terms and specify measurable quantities)
* Modifiable (evolve the Requirements Specification only via a formal change process, preserving a complete audit trail of changes)
* Does not specify any particular design
* Traceable (cross-reference with source documents and spawned documents).

## Functional Requirements

| **Req#** | **Requirement** | **Comments** | **Priority** | **Date Rvwd** | **SME Reviewed / Approved** |
| --- | --- | --- | --- | --- | --- |
| FR\_ST\_01 | Student Sign-Up & Verification **1. Account Creation:**  Students must register using their full name, university-issued “.edu” email, phone number, and password.  **2. Email Verification:**  The system will send a verification link to the provided “.edu” email. The student must click the link to confirm their email before proceeding.  **3. Access Approval:** Only students with a verified “.edu” email can access the platform’s rental services. If the email is invalid or unverified, the system will deny access and prompt the student to reattempt verification. | This will ensure only real students may rent property as that is the intention of our app. | 1 | 10/03/2025 | Moel Plasa |
| FR\_ST\_02 | Once a student has been registered, they may now search for property, the system will provide them with the capability to search for property in their desired area, via google maps. |  | 1 | 10/03/2025 | Moel Plasa |
| FR\_ST\_03 | Initially the system will provide the student with properties within a 2km radius of their university. | The student can use the functionality defined in the previous requirement if they wish | 1 | 10/03/2025 | Thomas Kroj |
| FR\_ST\_04 | The system will provide the students with the capability to rent a property.Initially they have to press rent.After that they will be provided with a form to fill up. | The form will require info regarding the time period of when the student needs this property. | 1 | 10/03/2025 | Thomas Kroj |
| FR\_ST\_05 | The system will provide the students with the capability to apply for listed dorm rooms of their respective university. |  | 1 | 10/03/2025 | Eden Pajo |
| FR\_ST\_06 | **The system shall guide students through the contract signing and payment process as follows:**   1. The student fills out the rental application form and confirms their order. 2. The system generates a digital lease contract and presents it for e-signature. 3. Once the student signs, they are legally obliged to pay the agreed amount. 4. The system prompts the student to select a payment method (PayPal, bank card, or cash). 5. If an online payment method is chosen, the system provides the necessary steps to complete the transaction. 6. Once payment is confirmed, the system marks the contract as active and notifies both parties. 7. The system creates a chatbox between the student and the landlord, which:  * Remains open for the duration of the contract. * Closes automatically when the contract ends to prevent further communication.  1. After staying at the property, the student can submit a review and rating if they choose to. | The contract needs to be signed, we will be using Adobe Sign for the e-Signature. | 1 | 10/03/2025 | Moel Plasa |
| FR\_ST\_07 | In case the student has a property, and they want to put this property up for listing so they can cohabitate with another student, the system will provide the student with the capability to do so. Of course they will need to provide the necessary details such as: photos, location, availability. | If another student wants to rent such a property the process is entirely the same as mentioned above. | 1 | 10/03/2025 | Aleksander Murati |
| FR\_LL\_01 | **LandLord sign-up process**  **1.Sign-Up & Initial Verification:**  The landlord must create an account with their email, phone number, and password. The system will send a verification email and an OTP to confirm their contact details.  **2.KYC Document Submission:**  To list properties, landlords must upload: A valid government-issued ID (passport, national ID, or driver’s license). Proof of property ownership (deed, lease agreement, or certificate).  **3.Review & Approval Process:**  The system performs automated checks, with manual review if needed. If approved, the landlord gains full access to the platform. If rejected, they receive a notification with the reason and can resubmit. The system enforces fraud detection measures to verify authenticity. Periodic KYC revalidation may be required. | We are taking these measures to ensure that no bots sign up and also no landlord has 2 accounts. We are trying to take considerable measures for safety, in case of the students the email was already verified by the universities. | 1 | 10/03/2025 | Nadire Lazri |
| FR\_LL\_02 | The system will provide the landlord with the capability to put their property up for rent and specify the price for their property and also decide if they only accept online payment or also cash. The system will also allow the landlord to make changes to their existing listings or entirely delete them. | Of course the landlord will need to provide pictures, a description and available dates. | 1 | 10/03/2025 | Klajdi Murataj |
| FR\_LL\_03 | The system will notify the landlord in case any students have applied for their property and of course the system will provide the landlord with the capability to reject or accept this application. |  | 1 | 17/03/2025 | Klajdi Murataj |
| FR\_LL\_04 | The system shall allow landlords to generate digital lease agreements using predefined templates. And also the system will allow the landlords to send these contracts to the tenants for e-signature.  This is the process:   * Student applies for rent and fills the form * Landlord rejects or accepts the application * Landlord sends the lease agreement he has chosen * The rest of the steps from “FR\_ST\_06” need to be followed by the student * The system will notify the landlords when a payment is sent their way by a student | The templates are provided in the app, of course the landlord has to choose one of them prior to sending contracts to students. | 1 | 17/03/2025 | Moel Plasa |
| FR\_LL\_05 | The system will allow landlords to store existing signed contracts for safety reasons. They may choose to delete them after the contract time period is over, but not prior. |  | 1 | 17/03/2025 | Thomas Kroj |
| FR\_LL\_06 | The system will provide the landlord with the capability to leave reviews for the students who stayed in their property to help other landlords understand if they should accept applications from these students. |  | 1 | 17/03/2025 | Eden Pajo |
| FR\_LL\_07 | The system will allow the landlord to hire real estate agents and also leave a review regarding their performance  **Inviting an Agent:**   * The landlord can invite a real estate agent to manage one or more of their properties. * To do this, the landlord must: * Select which properties they want the agent to manage. * Specify the commission percentage the agent will receive from the rent. * View a list of available agents provided by the system. * Once an agent is selected, the system shall send an invitation request to the agent for approval. * The agent may accept or decline the request.   **Reviewing the Agent:**   * Once an agent has managed a property, the landlord will have the option to submit a review of the agent’s performance. * The review may include:    + A rating system (e.g., 1–5 stars).   + Written feedback describing the agent’s performance. * The system shall ensure that only landlords who have worked with the agent can leave a review. * Reviews will be visible to other landlords when selecting an agent |  | 1 | 17/03/2025 | Moel Plasa |
| FR\_UA\_01 | **This is the process for a university administrator to create an account**  **1. Account Creation & Verification:**   * University administrators must sign up with their name, official university email, phone number, and password. * The system will send a verification link to their email and an OTP to their phone.   **2. University Affiliation Validation:**   * Administrators must upload a university-issued ID or authorization letter. * The system will verify the email domain and documents through automated checks and manual review if needed.   **3. Approval & Access:**   * If approved, the administrator gains full access to list, manage, and remove dormitory properties. * If rejected, they receive a notification with the reason and can resubmit documents. | We made sure to take all the necessary safety precautions, so only real university personnel may add listings, it will be impossible for anyone else to take this position. | 1 | 17/03/2025 | Eden Pajo |
| FR\_UA\_02 | The system will allow university administrators to list available dorm rooms along with their specifications, such as: is the room single or shared, what is the exact location and price. |  | 1 | 17/03/2025 | Aleksander Murati |
| FR\_UA\_03 | The system will allow university administrators to accept or reject student applications for dorm rooms. Once a student application is accepted by the administrator the student is assigned to that room and the listing is deleted from the system. |  | 1 | 17/03/2025 | Aleksander Murati |
| FR\_REA\_01 | KYC Verification for Real Estate Agents **1. Application Submission:** Real estate agents must sign up using their email, phone number, and government-issued ID.  **2. License Verification:** Agents must upload a scanned copy of their real estate agent license for validation.  **3. Review & Approval: -** The system will verify the submitted documents through automated checks and may require manual review. - If approved, the agent gains access to the platform. - If rejected, they will receive a notification with the reason and an option to reapply. | Again we require an id card, and this time also a certified real estate agent license. Again these measures are taken for maximum safety, so only proper individuals may have access to the system. | 1 | 17/03/2025 | Klajdi Murataj |
| FR\_REA\_02 | The agents cannot directly apply for jobs, the system will notify them if a landlord requires their services, and also the system will provide the details that the landlord requires. Of course the agent may choose to reject. |  | 1 | 17/03/2025 | Klajdi Murataj |
| FR\_REA\_03 | If the agent agrees they will receive a contract which they will have to sign via e-signature, in the contract the payment method will be specified and what percentage of the rent the agent will receive, if the contract specifies online payment then the agent will be required to enter card info after signing so they may receive payments. |  | 1 | 17/03/2025 | Nadire Lazri |
| FR\_REA\_04 | The system will create a chatbox between the landlord and the agent if the agent accepts the job, so they may chat directly. |  | 1 | 17/03/2025 | Nadire Lazri |
| FR\_SA\_01 | The system shall allow the System Administrator to manage user accounts, including approving or rejecting new landlord, agent, and university administrator registrations.If needed, the administrator can also suspend or deactivate an account later. |  | 1 | 17/03/2025 | Moel Plasa |
| FR\_SA\_02 | The system shall allow the System Administrator to manage access control, ensuring users only have permissions relevant to their roles.The System Administrator assigns role-based permissions to users (landlords, agents, students, university admins). The system restricts users from accessing unauthorized sections. If necessary, the administrator can modify or revoke permissions. |  | 1 | 17/03/2025 | Thomas Kroj |
| FR\_SA\_03 | The system shall allow the System Administrator to monitor and manage online payment transactions, including resolving disputes.If a payment dispute occurs, the System Administrator can:   * Review transaction logs. * Approve or deny refund requests. |  | 1 | 17/03/2025 | Thomas Kroj |
| FR\_SA\_04 | The system shall automatically generate backups of critical data and allow the System Administrator to restore lost data when needed. |  | 1 | 17/03/2025 | Eden Pajo |
| FR\_SA\_05 | The system shall allow the System Administrator to review and resolve disputes between users (students, landlords, agents, university admins). If a user files a complaint through the system then the admin can send a warrant to the reported user. |  | 1 | 17/03/2025 | Klajdi Murataj |

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 3.2 Non-Functional Requirements

| **Req#** | **Requirement** | **Comment** | **Priority** | **Date Rvwd** | **Sme**  **Reviewed/ Approved** |
| --- | --- | --- | --- | --- | --- |
| NFR\_01 | The application should :   * Run on:mobile,desktop or laptop devices * Ron on:Windows,Mac Os and Linux |  | 2 | 16/03/2025 | Klajdi Murataj |
| NFR\_02 | The application must have an UI which is:user-friendly,intuitive,easy to use,concise and error-tolerant. |  | 3 | 16/03/2025 | Klajdi Murataj |
| NFR\_03 | The application shouldn’t take more than 10 seconds to bootstart.    The application shouldn’t take more than 1 minute to perform a task or to transition pages after an action is performed. |  | 3 | 16/03/2025 | Moel Plasa |
| NFR\_04 | The application should be operational during peak network traffic.  The application shouldn’t delay a process or a page transition for more than 5 minutes during peak network traffic. |  | 3 | 16/03/2025 | Moel Plasa |
| NFR\_05 | The application should handle a minimum of 10,000 connections with optimizations for more.  The application should keep a session open for 3 hours from the connection start and should change the status of the connection as ‘expired’.  When the user opens the application and its connection status is ‘expired’,the application should reload and prompt the user to authenticate again. |  | 3 | 16/03/2025 | Aleksander Murati |
| NFR\_06 | The application should be descriptive during the user experience.  When the user performs an action which can be successful or a failure, the application should display a responsive alert or an informative pop-up to indicate the success or failure result. |  | 2 | 16/03/2025 | Aleksander Murati |
| NFR\_07 | The application should be helpful in case of a mistake by the user,an error or user confussion.  In case the user enters wrong credentials or performs an action which results in an ‘failure’ alert,the application should provide information about the ‘failure’ cause and additional information or steps to resolve the ‘failure’.  When there is an ‘error’ status of the system,the application should provide information about the cause and possibly the timeframe of the error and it should also provide additional steps in order to troubleshoot the error when possible.  In the case when the user is confused about the workings of the application,there should be a dedicated FAQ’s or live support section,providing the necessary steps and the cause of the confusion in order to resolve the problem. |  | 3 | 16/03/2025 | Thomas Kroj |
| NFR\_08 | The user should be able to log in through: username and password and biometric authentication.  For the user data and biometric authentication,the application must properly manage the data in order to secure the credentials from misuse or data leaks. |  | 3 | 16/03/2025 | Thomas Kroj |
| NFR\_09 | Users should be able to register and delete their account.  Users can register an account only through their university email.  In case the user demands it’s account to be deleted,the application should carry it out within 30 days and the user should be informed through their university email for the status of the process once the process is completed. |  | 3 | 16/03/2025 | Eden Pajo |
| NFR\_10 | Once logged in, the users are displayed on the homepage and depending on their device ,users are shown the UI accordingly. |  | 2 | 16/03/2025 | Eden Pajo |
| NFR\_11 | Users are allowed to change settings like :Allow position sharing and Allow access to contacts,camera and photos.  When the user prompts and action which requires access to the before-mentioned settings and that setting’s status is ‘disabled’ or ‘not allowed’,the application should present a pop-up which gives the user the opportunity to change the relevant setting/s in order for that functionality to work.  If the user doesn’t want to allow the setting/s in the upper scenario,the user is given the chance to withdraw from that functionality . |  | 2 | 16/03/2025 | Nadire Lazri |
| NFR\_12 | The user should be able to view and edit its profile information such:phone number and address.  The user won’t be able to change profile information like:email,name,surname,date of birth as they will be taken from the school email they are logged in. |  | 3 | 16/03/2025 | Nadire Lazri |
| NFR\_13 | The user is transported to the homepage when the login is successful,displaying the rental deals in a mixed order of the filterable elements. |  | 2 | 16/03/2025 | Klajdi Murataj |
| NFR\_14 | The user should be able to filter the deals based on :location,price,rental owner,the housing capacity and shared individual specifics like:gender and age. |  | 2 | 16/03/2025 | Klajdi Murataj |
| NFR\_15 | The application should present the user with the rental deal specifications on click,like:Rental owner,address,price,and if a rental deal sharer is included it should display information like :name,surname,age,gender,university,program of study and optionally a description of some requirements demanded from the deal sharer or partner |  | 3 | 16/03/2025 | Moel Plasa |
| NFR\_16 | The user should be able to pay in cash or with card.  The application should provide a payment id along with a payment tracking which tracks the status of the payment.  The application should proceed with the next phase of the payment process once the cash payment tracking results in ‘approved’ status. |  | 3 | 16/03/2025 | Moel Plasa |
| NFR\_17 | The application should accept payment with card.  The user should be able to input the necessary card details for the transaction.  The application should track the transactions and if a transaction fails,it should make sure that the transactions are either completed or not carried out. |  | 3 | 16/03/2025 | Thomas Kroj |
| NFR\_18 | The application should link the user and the rental owner in a chat in case of a successful rental payment or if the user selects this functionality. |  | 3 | 16/03/2025 | Eden Pajo |
| NFR\_19 | The application should allow the user to navigate to the map page and display a magnified map with pinned locations of rental deals nearby.  The user should be able to select the magnification index of the map . |  | 2 | 16/03/2025 | Aleksander Murati |
| NFR\_20 | The user should be able to filter the deals based on the selection criterias on the homepage. |  | 2 | 16/03/2025 | Nadire Lazri |

### Product Requirements

Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.

#### 3.2.1.1 Usability Requirements

Include any specific usability requirements, for example,

* Learnability
* The user documentation and help should be complete
* The system should be easy to learn

#### 3.2.1.2 Performance Requirements

Specify static and dynamic numerical requirements placed on the system or on human interaction with the system:

* Static numerical requirements may include the number of terminals to be supported, the number of simultaneous users to be supported, and the amount and type of information to be handled.
* Dynamic numerical requirements may include the number of transactions and tasks and the amount of data to be processed within a certain time period for both normal and peak workload conditions.

All of these requirements should be stated in measurable form. For example, "95% of the transactions shall be processed in less than 1 second" rather than “an operator shall not have to wait for the transaction to complete”.

#### 3.2.1.3 Availability

Include specific and measurable requirements for:

* Level of availability required
* Coverage for geographic areas
* Impact of downtime on users and business operations
* Impact of scheduled and unscheduled maintenance on uptime and maintenance communications procedures
* Reliability (e.g., acceptable mean time between failures (MTBF), or the maximum permitted number of failures per hour).

#### 3.2.1.4 Security

Specify the factors that will protect the system from malicious or accidental access, modification, disclosure, destruction, or misuse. For example:

* encryption
* activity logging, historical data sets
* restrictions on intermodule communications
* data integrity checks

### Organizational Requirements

Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements.

### External Requirements

* + Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

# User Scenarios/Use Cases

* FR\_ST\_01

| UC Name | UC\_ST\_1 |
| --- | --- |
| Summary | The Use case depicts the email format validity |
| Dependency |  |
| Actors | **Student**,System |
| Preconditions | The system should be up and running.  User should select the user mode of use. |
| Description of the main sequence | 1. User opens the system 2. User enters the login interface 3. User enters an email and a password 4. User presses the login button 5. The system checks whether the email format is correct or not 6. .a .1 The system returns an authentication error 7. .2 User re-enters the credentials or continues to navigate the system. 8. .b .1 The system confirms the authentication 9. .2 The user is transported to the homepage |
| Description of the alternative sequence |  |
| Non-functional Requirement | The system should be able to accept user input through forms and process it to produce a result. |
| Postconditions | The system always returns a result regarding the authentication attempts. |

* FR\_ST\_02

| UC Name | UC\_ST\_2 |
| --- | --- |
| Summary | The Use case lets the user to search for properties via the map function |
| Dependency | FR\_ST\_1 |
| Actors | **Student**,System |
| Preconditions | The system should be up and running.  User should select the user mode of use.  User should be logged in. |
| Description of the main sequence | 1. User successfully logs in 2. User selects the map function 3. The system displays the map UI 4. User may filter the parameters or navigate unfiltered 5. User navigates through the map |
| Description of the alternative sequence |  |
| Non-functional Requirement | The system should be responsive of user commands and should be operational. |
| Postconditions | The user is interacting with the map UI and is logged in. |

* FR\_ST\_03

| UC Name | UC\_ST\_3 |
| --- | --- |
| Summary | The Use case lets the user to search for properties in a 2km radius by default |
| Dependency | FR\_ST\_2 |
| Actors | **System,**Student |
| Preconditions | The system should be up and running.  User should select the user mode of use.  User should be logged in. |
| Description of the main sequence | 1. User successfully logs in 2. User selects the map function 3. The system accesses the current location of the user 4. The system displays the properties within 2km radius in the map 5. User may edit the radius or continue navigating |
| Description of the alternative sequence |  |
| Non-functional Requirement | The system should be able to adjust the search radius based on user command or by default and should be operational. |
| Postconditions | The user is displayed a desired scope of the map function |

* FR\_ST\_04

| UC Name | UC\_ST\_4 |
| --- | --- |
| Summary | The Use case lets the user to rent a property through filling a form. |
| Dependency | FR\_ST\_1,FR\_ST\_2 |
| Actors | **Student**,System |
| Preconditions | The system should be up and running.  User should be logged in.  The system should display properties for rent. |
| Description of the main sequence | 1. User navigates the property searching. 2. User selects a property. 3. The system displays the property info. 4. A .1 User doesn’t slect rent and continues navigating 5. B .1 User selects rent 6. .2 The system provides the user with a form to fill 7. .3 User fills the form 8. .4 The system accepts the form 9. .5 The system displays an affirmative message 10. .6 The system processes the form |
| Description of the alternative sequence |  |
| Non-functional Requirement | The system should be able to accept and process user input through the form and should be operational. |
| Postconditions | The form application is processed. |

* FR\_ST\_05

| UC Name | UC\_ST\_5 |
| --- | --- |
| Summary | The Use case lets the user to apply for shared dorms in their respective university. |
| Dependency | FR\_ST\_1 |
| Actors | **Student**,System |
| Preconditions | The system should be up and running.  User should have a valid email.  User should be logged in. |
| Description of the main sequence | 1. User selects the shared dormitory function 2. The system accesses the student’s university info 3. The system checks for validity or for availability 4. A.1 The system returns an error and a alert with the description of the alert. 5. .2 User may try again or continue navigating the system 6. .B.1 The system returns an affirmative label 7. .2 The system sends an application to the university 8. .3 The system returns a responsive label |
| Description of the alternative sequence |  |
| Non-functional Requirement | The system should be operational and it should be able to process user’s data and it should have a stable connection with the university’s servers |
| Postconditions | The user ‘s relevant university data is processed. |

* FR\_ST\_06

| UC Name | UC\_ST\_6 |
| --- | --- |
| Summary | The Use case lets the user make a complete process of renting. |
| Dependency | FR\_ST\_2,FR\_ST\_4 |
| Actors | **System,**Student |
| Preconditions | The system should be up and running.  User should select the user mode of use.  User should be logged in. |
| Description of the main sequence | 1. User selects rent 2. The system displays a form application 3. User completes the form 4. User submits the form 5. The system processes the form 6. The system generates a lease contract available for e-signing 7. A.1 user reads and declines the contract and chooses cancel 8. B.1 user reads the contract and signs 9. The system displays a list for payment methods 10. A.1 User chooses cash 11. .2 The system generates an online bill format with a transaction id as reference to pay physically ata bank 12. .3 User makes the payment 13. .4 The system processes the transaction and returns an answer 14. A.1 The system prompts an error and displays an error with info about it 15. .2 The system automatically terminates the contract and the process is reset 16. B.1 The system returns an affirmative answer 17. .2 The contract is deemed ‘finalized’ and it’s content takes effect . 18. .3 The system notifies both parties involved that the contract has been finalized 19. B.1 User chooses PayPal or card as a method of payment 20. .2 The system prompts a form for the card credentials or PayPal account 21. .3 User completes the form and submits it 22. .4 The system processes the form and checks the transaction and returns an answer 23. .A.1 The system returns an error and displays information regarding it 24. .2 User can re-enter the data or cancel the contract 25. .B.1 The system returns an affirmative answer and an informative label 26. .2 The system informs both parties involved that the transaction has been made and the contract is ‘active’ 27. The system displays an interface prompting the user to leave a review and the rating 28. A.1 User closes the interface 29. B.1 User fills the data fields of the interface 30. .2 The system accepts and processes the data 31. .3 The system dispalys a ‘Thank you!’ banner 32. User completes the contract 33. The system creates a new chat and adds the landlord and the user 34. The system notifies the landlord and the user that the contract has been successful and highlights the chat between them 35. The student may leave a review |
| Description of the alternative sequence |  |
| Non-functional Requirement | The system should be operational and should be able to accept user input and process it through forms and furthermore it should be able to generate recipts,contract in a .pdf format and form a stable connection with bank servers in order to communicate transactions and recive feedback. |
| Postconditions | The user completes the application form |

* FR\_ST\_07

| UC Name | UC\_ST\_7 |
| --- | --- |
| Summary | The Use case lets the user to list his property as a shared property |
| Dependency |  |
| Actors | **Student**,System |
| Preconditions | The system should be up and running.  User should be logged in. |
| Description of the main sequence | 1. User successfully logs in 2. User selects to add the property as a property for share 3. The system displays an application form 4. A.1 User cancels the process 5. B.1 Use fills the form and submit it 6. .2 The system accepts and processes the form 7. .3 The system displays an informative label |
| Description of the alternative sequence |  |
| Non-functional Requirement | The system should have a list of valid properties for their users in case they want to add their property |
| Postconditions | The user has moved to the form phase of the application |

* FR\_LL\_01

| UC Name | UC\_LL\_1 |
| --- | --- |
| Summary | The Use case lets the landlord to create a profile |
| Dependency |  |
| Actors | **Landlord**,System |
| Preconditions | The system should be up and running. |
| Description of the main sequence | 1. Landlord select signup 2. The system displays a form asking for his phone number,password and email 3. A.1 Landlord cancels the process 4. B.1 Landlord fills the form 5. .2 The system accepts and processes the form and returns an answer 6. .A.1 The system returns an error and displays info regarding it 7. .B.1 The system returns an affirmative answer 8. .2 The system displays a new form that prompts the landlord to enter his identy card id and the document which proves the ownership of the property 9. .A.1 The system processes the results and returns an error and and an info panel regarding it 10. .2 Landlord can resubmit or continue navigating in the system 11. .B.1 The system processes the results and returns an affirmative label 12. .2 The system creates the profile and sends a notice to the landlord |
| Description of the alternative sequence |  |
| Non-functional Requirement | The system should be able to accept and process user input through forms |
| Postconditions | The user is intercating with the sign up interface |

* FR\_LL\_02

| UC Name | UC\_LL\_2 |
| --- | --- |
| Summary | The Use case lets the Landlord customise and put his property notice for rent |
| Dependency | FR\_LL\_1 |
| Actors | **System,**Landlord |
| Preconditions | The system should be up and running.  Landlord must be logged in. |
| Description of the main sequence | 1. Landlord successfully logs in 2. Landlord selects ‘add new’ 3. The system displays a form with multiple fields prompting the landlord to enter specifications like: price,payment method etc. 4. Landlord edits the data fields 5. Landlord confirms the add 6. The system accepts and processes the form 7. The system returns an informative label 8. Furthermore ,Landlord selects one of his rent notices 9. The system displays information about the notice and also gives two options: edit,delete 10. A.1 Landlord selects ‘edit’ 11. .2 The system makes the form editable 12. .3 Landlord edits the form 13. .4 Landlord submits the edited form 14. .5 The system accepts and processes the form 15. .6 The system updates the rent notice 16. B.1 Landlord selects ‘delete’ 17. .2 The system displays a confirmation form 18. .A.1 Landlord selects ‘yes’ 19. .2 The system deletes the notice 20. .3 The system returns a notice 21. .B.1 Landlord selects ‘no’ 22. .2 The system canels the process |
| Description of the alternative sequence |  |
| Non-functional Requirement | The system should be abke to accept and process user input through forms and keep the system operational. |
| Postconditions | The landlord adds a property for rent |

* FR\_LL\_03

| UC Name | UC\_LL\_3 |
| --- | --- |
| Summary | The Use case lets the Landlord accept or refuse an application for rent by a student |
| Dependency | FR\_LL\_1 |
| Actors | **System,**Landlord |
| Preconditions | The system should be up and running.  Landlord must be logged in. |
| Description of the main sequence | 1. Landlord recives a notice if a student has applied for one of his rents 2. The system displays a form with two options: refuse,accept 3. A.1 Landlord selects ‘refuse’ 4. .2 The system closes the display and a notice is sent to the student 5. B.1 Landlord selects ‘accept’ 6. .2 The system closes the display and a notice is sent to the student |
| Description of the alternative sequence |  |
| Non-functional Requirement | The system should be able to accept and process user input through forms and keep the system operational. |
| Postconditions | The landlord recives notices. |

* FR\_LL\_04

| UC Name | UC\_LL\_04 |
| --- | --- |
| Summary | The system shall allow landlords to generate and send digital lease agreements using templates. The process includes tenant application, landlord approval, contract selection, and student actions from FR\_ST\_06. Landlords are notified upon student payment. |
| Dependency | * Student Application Submission * Landlord Approval * Predefined Lease Agreement Templates * E-Signature Functionality * Payment Processing System * Notification System |
| Actors | Student, Landlord, System Administrator, Payment Gateway (External System) |
| Preconditions | Before the lease agreement and payment process can take place, both the student and landlord must have registered and verified accounts, with the landlord complying with KYC regulations. The landlord must have at least one active property listing, and the student must submit a rental application for a listed property. The landlord must review and approve the application before proceeding with the lease agreement. The system must provide predefined contract templates and support e-signature functionality to facilitate digital signing. Additionally, the student must have access to a valid payment method, and the system should be able to notify landlords when payments are made. |
| Description of the main sequence | 1. Student applies for rent by filling out the required form. 2. Landlord reviews the application and either accepts or rejects it.    * If rejected, the process ends.    * If accepted, the process continues. 3. Landlord selects a lease agreement from predefined templates. 4. Landlord sends the lease agreement to the student for e-signature. 5. Student reviews and signs the contract, legally committing to the rental terms. 6. Student selects a payment method (PayPal, bank card, or cash).   If online payment is chosen, the system:   * + Processes the transaction.   + Notifies the landlord of the received payment.   If cash payment is chosen, the landlord is informed that payment will be made in person.   1. Once payment is confirmed, the rental process is finalized, and the student gains access to the property. |
| Non functional requirements | Security – Complies with KYC regulations and encrypts user data.  Performance – Processes lease agreements and payments within 5 seconds.  Scalability – Supports 10,000+ simultaneous users without lag.  Availability – Ensures 99.9% uptime for system reliability.  Notifications – Sends real-time updates for applications, payments, and agreements.  Cross-Platform – Accessible via desktop and mobile, supporting major browsers.  Legal Compliance – Supports legally binding e-signatures and follows contract laws. |
| Postconditions | Once the lease is signed, it becomes legally binding. If an online payment method is chosen, the transaction is processed, and the landlord is notified. The system marks the property as occupied and removes it from listings. A chatbox opens for student-landlord communication during the lease. If an agent is involved, they are notified. The student gains access to the property and can leave a review. All transaction and contract details are securely logged. |

* FR\_LL\_05

| UC Name | UC\_LL\_5 |
| --- | --- |
| Summary | The Use case lets the Landlord store signed contracts after the contract time period is over, as during the time period the system stores them by default |
| Dependency | FR\_LL\_04 |
| Actors | **System,**Landlord |
| Preconditions | There should be an existing contract which is accomplished by following the steps in FR\_LL\_04 |
| Description of the main sequence | 1. The landlord accesses the contract storage section in the system. 2. The system displays a list of existing signed contracts. 3. The landlord can view contract details for reference. 4. The system securely stores the contracts for the duration of the lease. 5. Once the contract period ends, the landlord may choose to delete the contract. 6. If deleted, the system removes the contract from storage. |
| Description of the alternative sequence |  |
| Non-functional Requirement | The system shall provide a reliable and accessible interface for landlords to manage stored contracts.  Data integrity shall be maintained, preventing accidental modification or corruption of signed contracts.  The system shall ensure fast retrieval of stored contracts for reference when needed. |
| Postconditions | After storing a signed contract, the system ensures it remains securely accessible to the landlord. The contract cannot be deleted until the lease period ends. If the landlord chooses to delete it after this period, the system permanently removes it from storage. |

* FR\_LL\_06

| UC Name | UC\_LL\_6 |
| --- | --- |
| Summary | The Use case lets the Landlord leave a review for the client that stayed in their property, so other landlords may value if this is a worthy client. |
| Dependency | FR\_LL\_04 |
| Actors | **System,**Landlord |
| Preconditions | The system should be up and running.  Landlord must be logged in.  There should be an existing contract which is accomplished by following the steps in FR\_LL\_04 |
| Description of the main sequence | 1. The landlord accesses the review section in the system. 2. The system displays a list of past tenants eligible for review. 3. The landlord selects a student to review. 4. The system provides a review form with rating options and a comment section. 5. The landlord submits the review. 6. The system stores the review and makes it accessible to other landlords when evaluating future applications. |
| Description of the alternative sequence |  |
| Non-functional Requirement | The system shall ensure that reviews are securely stored and cannot be altered after submission.  The review interface shall be user-friendly and responsive across different devices.  Reviews shall be visible only to landlords to maintain privacy. |
| Postconditions | The review is up for other landlords to see. |

* FR\_LL\_07

| UC Name | UC\_LL\_7 |
| --- | --- |
| Summary | The Use case lets the Landlord to hire real estate agents by selecting properties, setting commission rates, and sending invitations. Agents can accept or decline. After working with an agent, landlords can leave a review with ratings and feedback, visible to other landlords for future selection.. |
| Dependency |  |
| Actors | **System,**Landlord, Real estate agent |
| Preconditions | The landlord must have an active, verified account and be logged into the system.  The system must have verified real estate agents available for selection.  The landlord must have at least one listed property to assign to an agent.  The real estate agent must have completed the KYC verification process.  The review feature is only accessible after the agent has managed a property. |
| Description of the main sequence | 1. The landlord logs into the system and navigates to the agent management section. 2. The system displays a list of verified real estate agents. 3. The landlord selects one or more properties to assign to an agent. 4. The landlord specifies the commission percentage the agent will receive. 5. The system sends an invitation request to the selected agent. 6. The agent reviews the request and either accepts or declines. 7. If accepted, the system finalizes the assignment and updates property management details. 8. After the agent has managed the property, the landlord can leave a review. 9. The system stores the review and makes it visible to other landlords. |
| Description of the alternative sequence |  |
| Non-functional Requirement | Reviews shall be protected from unauthorized edits or deletions.  The system shall notify landlords and agents of any status changes via email or in-app notifications.  The agent selection and review process shall be intuitive and easy to use. |
| Postconditions | If the agent accepts the invitation, they are assigned to the selected properties, and the system updates the management details.  Once an agent has managed a property, the landlord can submit a review.  The review is stored in the system and made visible to other landlords. |

* FR\_UA\_01

| UC name | UC\_UA\_01 |
| --- | --- |
| Summary | This UC describes the creation of a University administrator account |
| Dependency | SheerID API |
| Actors | University Administrator |
| Preconditions | The user should not have an existing account, this use case is valid only for the creation of one. |
| Description of the main sequence | 1. The user opens the software 2. The login form contains a button that says: “Don’t have an account? REGISTER HERE”. The User presses it. 3. The user fills up the form that shows up. This form contains these fields: type (a dropdown which contains the types of users, University admin should be selected by the user), university email (which then is validated by SheerID), Name, Surname, Phone Number, Birth date, Password, Confirm Password. 4. The form validates if the last two fields are the same. 5. If everything is okay then an account is successfully created and stored in the database. 6. This is a one time process and the user can log in next time only using the email and password. |
| Description of the alternative sequence | If the form is not completed in a valid way a panel is displayed on the screen with the reason for rejection, although a resubmission is still allowed |
| Non-functional requirement |  |
| Postconditions | The user is logged in and can access what he wishes within the available options of the university administrator interface |

* FR\_UA\_02

| UC name | UC\_UA\_02 |
| --- | --- |
| Summary | This UC describes the listing of university dorm rooms and their specifics |
| Dependency | Google Maps API |
| Actors | University Administrator |
| Preconditions | The user should be logged in |
| Description of the main sequence | 1. The user hits the List new room button on top of the list of the already listed dorms in the primary screen. 2. A form is displayed, containing property name, description capacity, size. 3. A map is also displayed, where the user specifies where the property is located 4. The user must also add a minimum of 5 pictures of the property 5. In the end a pdf of the contract should be added 6. User presses the CONFIRM button |
| Description of the alternative sequence |  |
| Non-functional requirement | EVERY field should be filled, nothing is optional |
| Postconditions | The user goes back to the main screen, where the new property now shows up in the listings |

* FR\_UA\_03

| UC name | UC\_UA\_03 |
| --- | --- |
| Summary | This UC describes the users approval of a students application |
| Dependency |  |
| Actors | University Administrator, Student |
| Preconditions | The user should be logged in and should have at least one request to aprove |
| Description of the main sequence | 1. The user hits the Requests button in the home screen 2. The page that shows up displays each request in a card, containing the name of the property and the name of the student who has requested the dorm. These cards will be sorted by latest. 3. Upon clicking one of the cards, the contract and the students e-signature show up. Below them there is the contact student option (since the phone number and email are in the database), the approve, and the refuse option. 4. a- after approval the dorm is not publicly displayed anymore and is moved to the “Booked” tab of the university admins profile. Other requests on the same dorm are automatically denied and a notification is sent out that the dorm has been booked by someone else. 5. b- after denial that student gets a message saying that he has been refused. The property remains public. |
| Description of the alternative sequence |  |
| Non-functional requirement |  |
| Postconditions | After approval or denial the user is returned to the previous page. |

* FR\_REA\_01

| UC name | UC\_REA\_01 |
| --- | --- |
| Summary | This UC describes the creation of a real estate agent account |
| Dependency | Jumio API |
| Actors | Real estate agent |
| Preconditions |  |
| Description of the main sequence | 1. The user opens the software 2. The login form contains a button that says: “Don’t have an account? REGISTER HERE”. The User presses it. 3. The user fills up the form that shows up. This form contains these fields: type (a dropdown which contains the types of users, Real estate agent should be selected by the user), company email and real estate license (which then is validated by Jumio), Name, Surname, Phone Number, Birth date, Password, Confirm Password. 4. The form validates if the last two fields are the same. 5. If everything is okay then an account is successfully created and stored in the database. 6. This is a one time process and the user can log in next time only using the email and password. |
| Description of the alternative sequence |  |
| Non-functional requirement | EVERY field should be filled, nothing is optional |
| Postconditions | The user is logged in and can access what he wishes within the available options of the real estate agent interface |

* FR\_REA\_02

| UC name | UC\_REA\_02 |
| --- | --- |
| Summary | This UC describes the job offer part of the agent account |
| Dependency |  |
| Actors | Real estate agent, landlord |
| Preconditions |  |
| Description of the main sequence | 1. The user hits the Requests button in the home screen 2. The page that shows up displays each request in a card, containing the name of the property and the name of the landlord who wants to put the property up for rent. These cards will be sorted by latest. 3. Upon clicking one of the cards, the property and its basic details show up. Below them there is the contact landlord option (since the phone number and email are in the database), the approve, and the refuse option. 4. a- after approval the agent can create the page for that property (same interface as other users with listing privileges) and is responsible for selling it. Other requests on the same property are automatically denied and a notification is sent out that the property has been moved to the responsibility of another agent. 5. b- after denial that landlord gets a message saying that he has been refused. The requests to other agents remains valid. |
| Description of the alternative sequence |  |
| Non-functional requirement |  |
| Postconditions | After approval or denial the user is returned to the previous page. |

* FR\_REA\_03

| UC name | UC\_REA\_03 |
| --- | --- |
| Summary | This requirement specifies the contract signing procedure. |
| Dependency | Adobe Sign, Jumio |
| Actors | Real estate agent, landlord |
| Preconditions |  |
| Description of the main sequence | 1. User Initiates Verification: Real estate agent begins the KYC process. 2. Document Capture: Agent uploads documents (ID, license). 3. Send to Jumio API: Send documents and data to Jumio for verification. 4. Jumio Verification: Jumio processes the documents and checks for validity. 5. Receive Response: Handle Jumio’s response (success, failure, pending). 6. Notify the Agent: Notify the agent of the verification result. 7. Store Data: Store the verification results securely for auditing purposes. 8. Ongoing Monitoring (optional): Monitor license validity periodically. |
| Description of the alternative sequence |  |
| Non-functional requirement | The signature is mandatory for approval, as is the payment in the end. |
| Postconditions | After the contract is sealed, the real estate agent has permission to lease the property. |

* FR\_SA\_01

| UC name | UC\_SA\_01 |
| --- | --- |
| Summary | This requirement states that the System Administrator deals with the management of all the user accounts and their specifications |
| Dependency |  |
| Actors | System Administrator, landlord, Real Estate Agent, University Administrator |
| Preconditions | 1. User Registration System: Users must be able to register and provide necessary details. 2. Role Management System: Proper role-based permissions must be in place (landlord, agent, admin). 3. Administrator Authentication: Secure authentication for system administrators. 4. Account Suspension/Deactivation: Features to suspend or deactivate user accounts. 5. Notification System: Notify users about status changes (approval, rejection, suspension). 6. Audit Logging: Track and log administrator actions for accountability. 7. Admin UI: An interface for administrators to manage user accounts. 8. Reactivation Workflow (Optional): Ability to reactivate suspended accounts. |
| Description of the main sequence | 1. User registers for an account (user submits registration details). 2. Admin logs in and accesses the admin dashboard (admin interface is authenticated). 3. Admin reviews pending registrations (admin verifies user information). 4. Admin approves or rejects the user registration (status changes to active or rejected). 5. Admin suspends or deactivates an existing account (admin takes action based on user behavior). 6. System updates user status and logs action (audit logs maintain transparency). 7. Admin notifies the user of the decision (user is informed via notifications). 8. Admin reviews and potentially reactivates suspended accounts (ongoing account management). 9. Admin continues monitoring user accounts and activity (admin ensures security and proper user management). |
| Description of the alternative sequence |  |
| Non-functional requirement |  |
| Postconditions |  |

* FR\_SA\_02

| UC name | UC\_SA\_02 |
| --- | --- |
| Summary | This requirement states that the System Administrator is allowed to revoke or modify user permissions for different user levels. |
| Dependency |  |
| Actors | System Administrator, landlord, Real Estate Agent, University Administrator, Student |
| Preconditions | 1. The System Administrator is logged in 2. The target user exists in the system (is registered in the database). 3. The System Administrator has permission to manage access control    * The admin’s role must include permissions like edit\_user\_roles or manage\_permissions. 4. Roles and permissions are predefined    * The system must have a predefined list of roles (e.g., Landlord, Student) and associated permissions. 5. No conflicting system maintenance    * The system is not undergoing updates or maintenance that would disrupt permission changes. |
| Description of the main sequence | 1. Admin navigates to User Management. 2. System displays list of users 3. Admin selects target user 4. System loads user’s current permissions    * The system fetches and displays:      + The user’s current role (e.g., "Landlord").      + Assigned permissions (e.g., "Add Property," "View Leases"). 5. Admin modifies permissions    * The Administrator:      + Changes the user’s role (e.g., from "Landlord" to "Agent") OR      + Manually checks/unchecks specific permissions (e.g., grants "Approve Payments"). 6. Admin confirms changes. 7. System validates and updates permissions    * The system verifies that:      + The new role/permissions are valid.      + The Admin has authority to make these changes.    * If valid, the system updates the user’s permissions in the database. 8. System notifies Admin of success |
| Description of the alternative sequence |  |
| Non-functional requirement | * Security: The system will ensure only authorized admins can make such changes. * Reliability: The system shall maintain 99.9% availability for permission management functions to ensure consistent access control. * Performance: The system shall process all permission checks in under 100 milliseconds to maintain system responsiveness. |
| Postconditions |  |

* FR\_SA\_03

| UC name | UC\_SA\_03 |
| --- | --- |
| Summary | This requirement states that the System Administrator is allowed to manage the online transactions and also resolve disputes revolving around these transactions. |
| Dependency | VISA Direct, PayPal API |
| Actors | System Administrator, landlord, Real Estate Agent, |
| Preconditions | 1. The System Administrator must be successfully logged in with valid credentials. 2. The online payment system must be properly connected and operational. 3. The disputed transaction must be recorded in the system logs with complete details (amount, timestamp, parties involved). 4. A payment dispute must have been formally raised by either the payer or payee. 5. The logged-in administrator must have "payment\_dispute\_management" privileges. 6. The payment management module must be online and not under maintenance. |
| Description of the main sequence | 1. Admin logs into the dashboard and accesses payment transaction management. 2. Admin reviews transaction logs and identifies any disputes. 3. Admin reviews the dispute case to understand the situation. 4. Admin approves or denies the refund request based on the information provided. 5. If approved, the admin processes the refund via the payment gateway API (e.g., PayPal, VISA). 6. If denied, the admin documents the denial and informs the user. 7. System updates the transaction and dispute status based on the action taken. 8. The admin monitors ongoing or new disputes to ensure timely resolution. 9. The system logs all actions taken by the admin for auditing purposes. |
| Description of the alternative sequence |  |
| Non-functional requirement | * Security: The system will ensure only authorized admins can access the admin dashboard to resolve the disputes * Performance: The system shall load transaction logs and dispute cases in under 2 seconds to maintain operational efficiency. |
| Postconditions |  |

* FR\_SA\_04

| UC name | UC\_SA\_04 |
| --- | --- |
| Summary | This requirement allows the System Administrator to restore lost data. |
| Dependency | Database API |
| Actors | System Administrator |
| Preconditions | 1. The System Administrator must be logged in with valid credentials and have "backup\_management" permissions. 2. A secure backup storage location (e.g., cloud storage, external server) must be pre-configured and accessible. 3. The system must have predefined what constitutes "critical data" (e.g., user accounts, transaction logs, permission settings). 4. Automated backup schedules (e.g., daily/weekly) must be enabled in the system settings. 5. For restorations, at least one valid backup file must exist for the data being recovered. |
| Description of the main sequence | 1. Scheduled Backup – System automatically creates backups at predefined intervals. 2. Storage – Backups are securely stored in a database or cloud storage. 3. Monitoring – System logs backup status and alerts the admin on failures. 4. Restore Request – Admin initiates data restoration if needed. 5. Data Recovery – System retrieves and restores the latest backup. 6. Verification – System confirms successful restoration and logs the action. |
| Description of the alternative sequence |  |
| Non-functional requirement | * Security: The system shall encrypt all backup files using AES-256 encryption both during transmission and storage.   The system shall require multi-factor authentication for all data restoration requests.   * Reliability: The system shall maintain a 99.9% success rate for scheduled backup operations.The system shall automatically retry failed backup attempts up to 3 times before alerting administrators. |
| Postconditions |  |

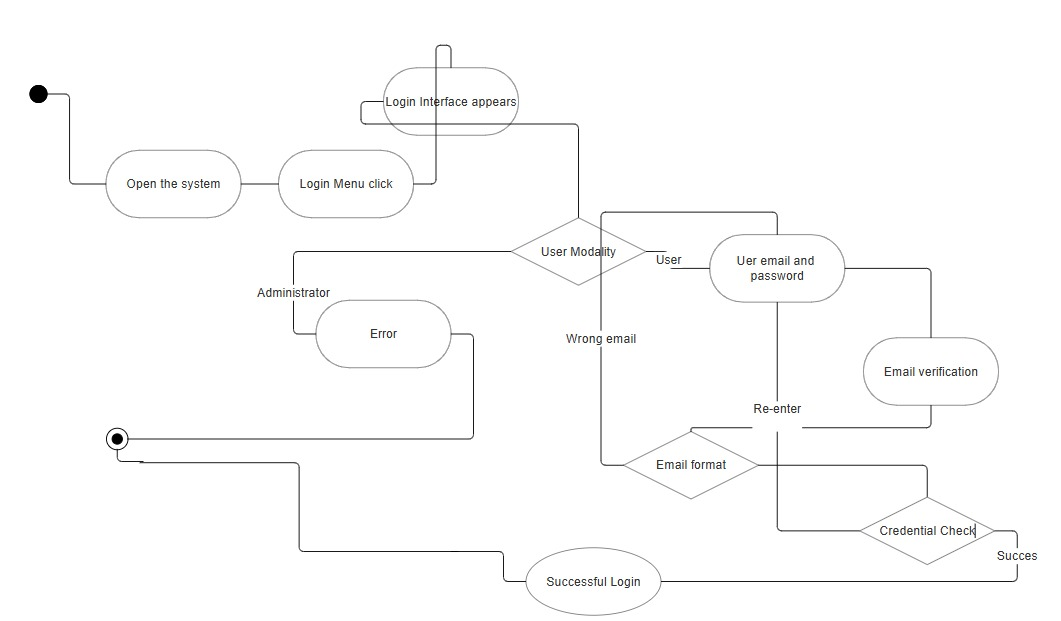
* FR\_SA\_05

| UC name | UC\_SA\_05 |
| --- | --- |
| Summary | This requirement allows the System Administrator to review and resolve disputes between users |
| Dependency |  |
| Actors | System Administrator, Student, Landlord |
| Preconditions | 1. Admin Access – System Administrator must be authenticated with dispute management permissions. 2. Payment System Active – Payment gateway integrated and operational. 3. Valid Transaction – Disputed payment exists in logs with complete details. 4. Dispute Registered – Formal dispute raised by payer, payee, or fraud system. 5. Evidence Available – Supporting documentation (logs, delivery proof) and recent chat history should be accessible for evidence. |
| Description of the main sequence | 1. System Administrator logs into the system using secure credentials. 2. The system authenticates the System Administrator and loads the Admin Dashboard. 3. The System Administrator navigates to the “Dispute Management” or “Complaints” section. 4. The system displays a list of user-filed complaints, including details such as date, involved parties, and status (e.g., “Pending”, “Under Review”). 5. The System Administrator selects a complaint to review. 6. The system retrieves and displays the complaint details, including:  * Reporting user info (e.g., student, landlord) * Reported user info * Complaint description * Attached evidence (if any) * Timestamp and complaint history  1. The System Administrator reviews both parties' profiles and any additional history or communication related to the complaint. 2. Based on the evidence and system logs, the System Administrator chooses to resolve the complaint. 3. The System Administrator may choose to:  * Dismiss the complaint (if unjustified) * Send a warning or warrant to the reported user  1. If the admin sends a warrant, the system:  * Records the action with date/time and reason * Updates the complaint status to “Resolved – Warrant Issued” * Sends a notification to the reported user containing the warrant and a message |
| Description of the alternative sequence |  |
| Non-functional requirement | Performance: Enable fast loading of complaint details and user profiles (≤ 2 seconds).  Usability: Provide a user-friendly interface for filtering, viewing, and resolving complaints.  Accessibility: Ensure the complaint management module is accessible to admins with disabilities.  Reliability: Maintain accurate records of all filed complaints and admin actions taken.  Scalability: Handle an increasing number of dispute cases without affecting performance.  Security: Protect the privacy of both parties involved in the dispute and log all admin decisions securely. |
| Postconditions |  |

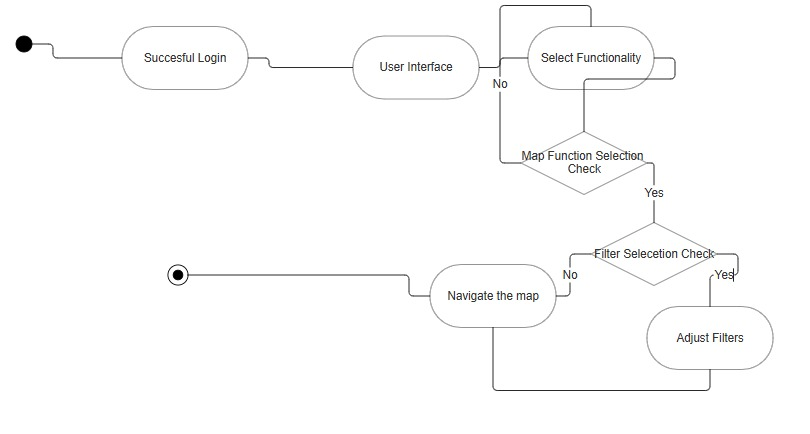
# 5. Diagrams

5.1 Activity Diagrams

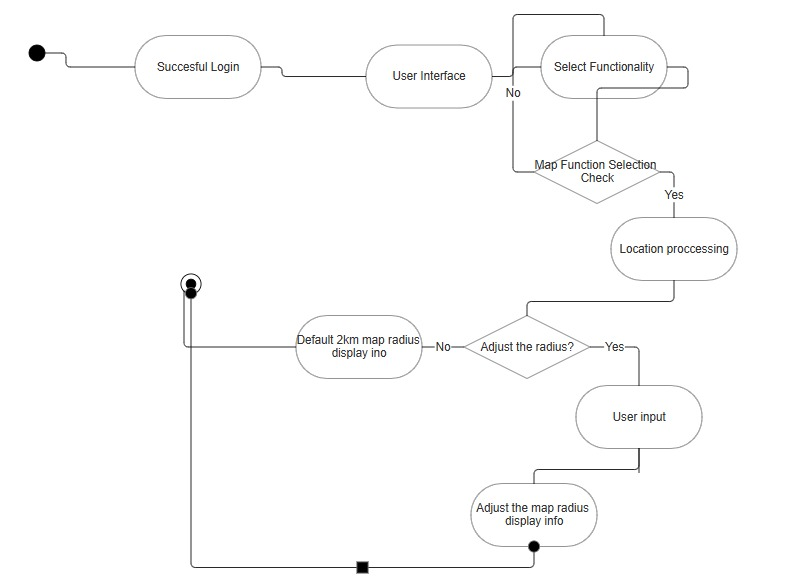
* UC\_ST\_01



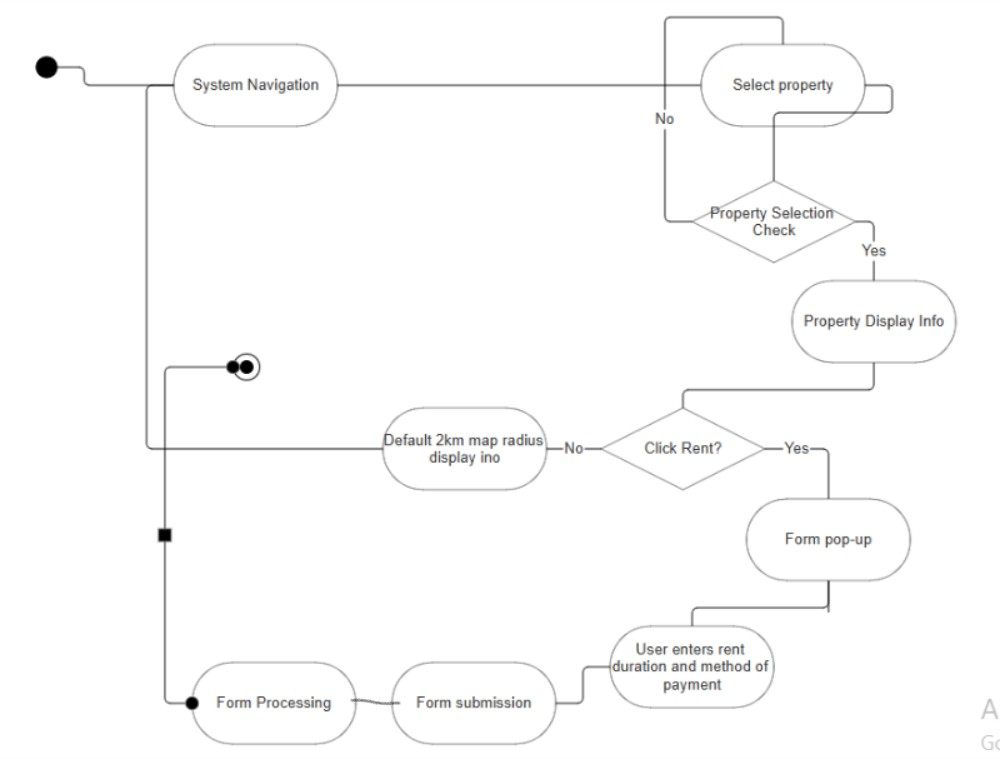
* UC\_ST\_02



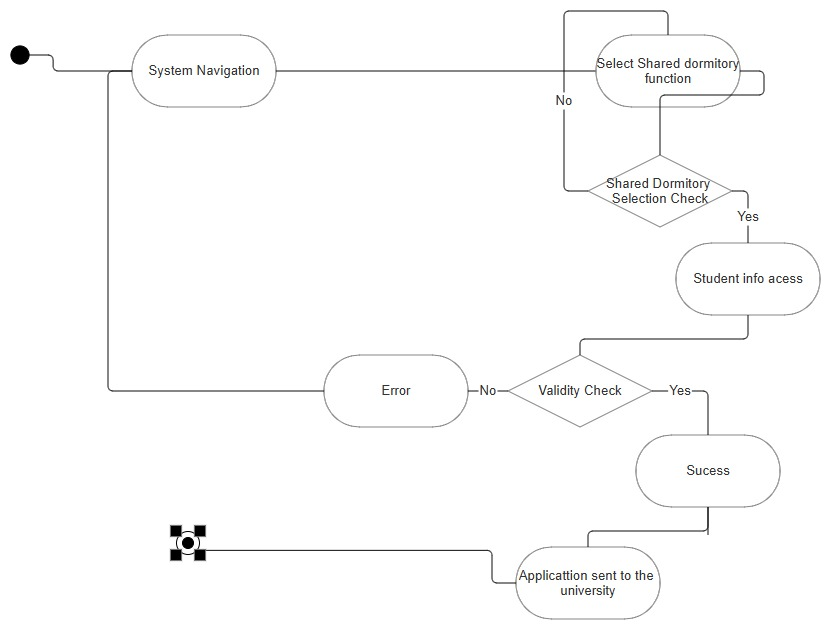
* UC\_ST\_03



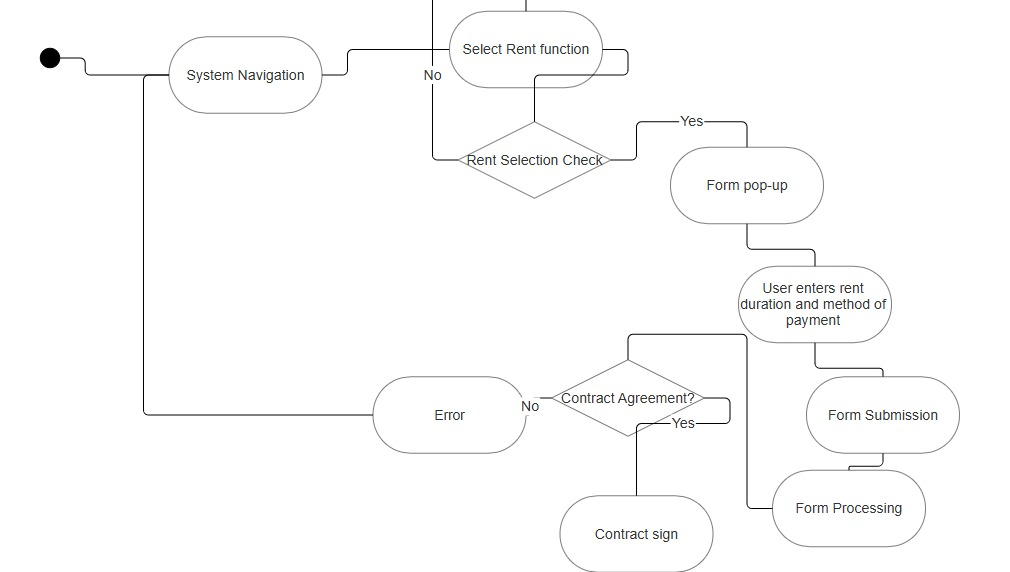
* UC\_ST\_04

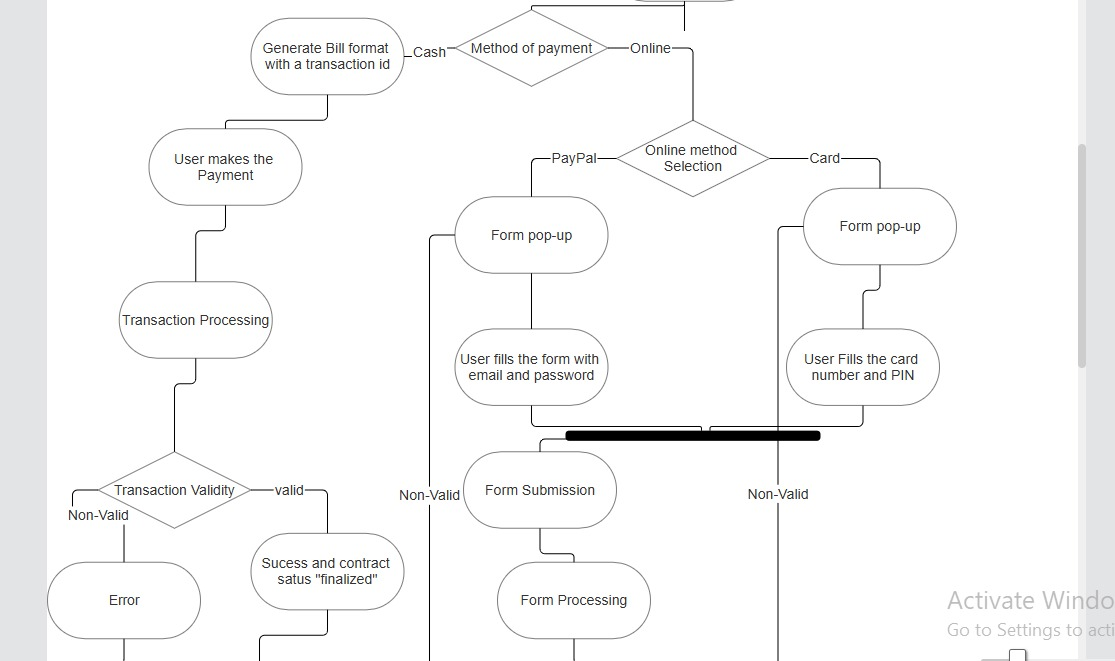


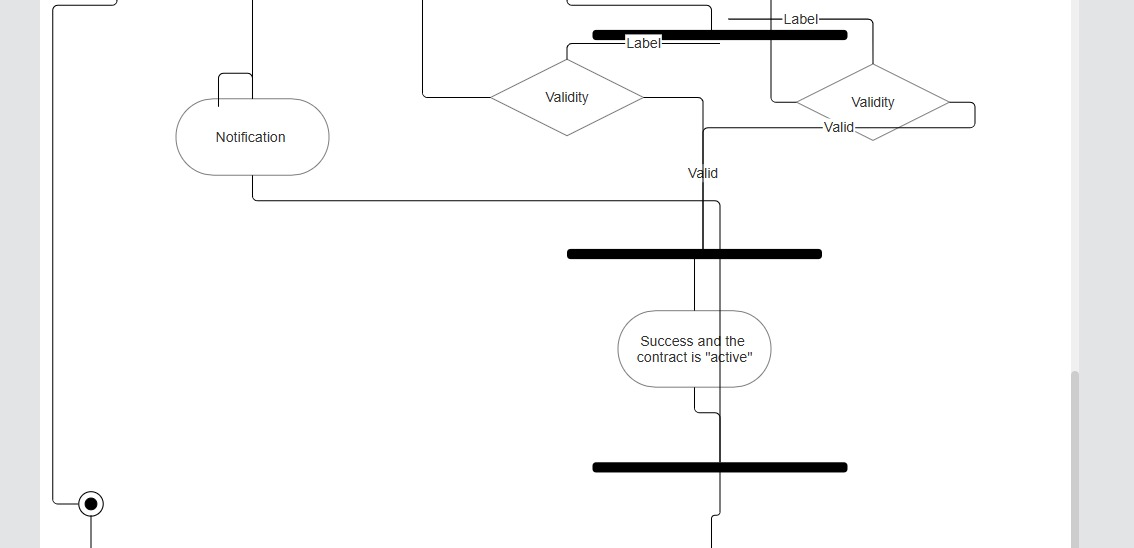
* UC\_ST\_05

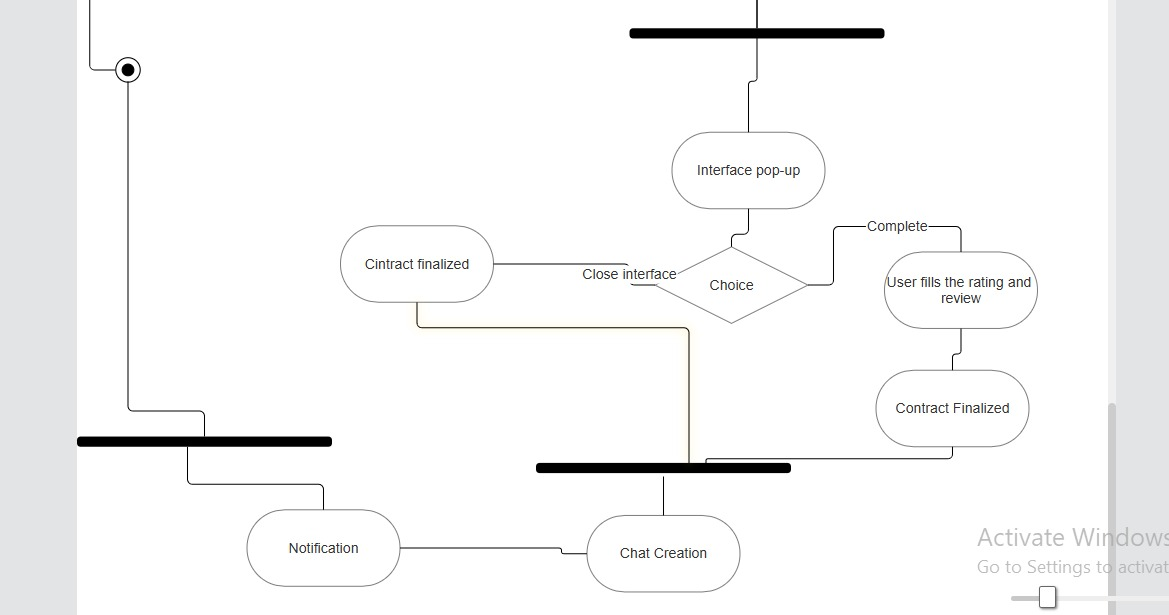


* UC\_ST\_06

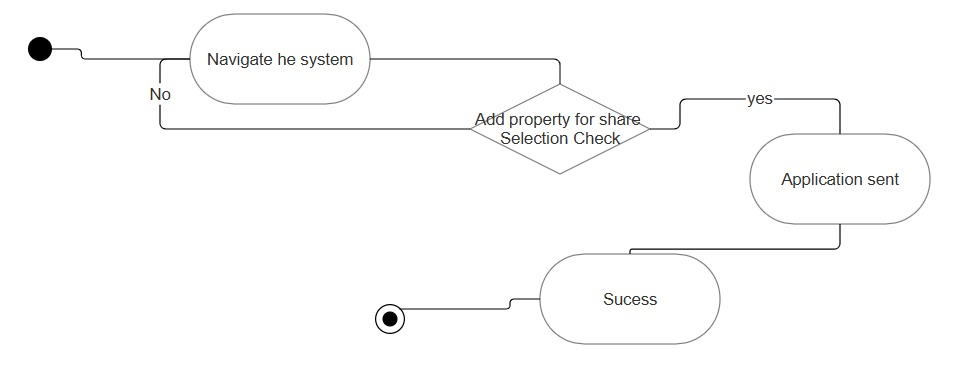




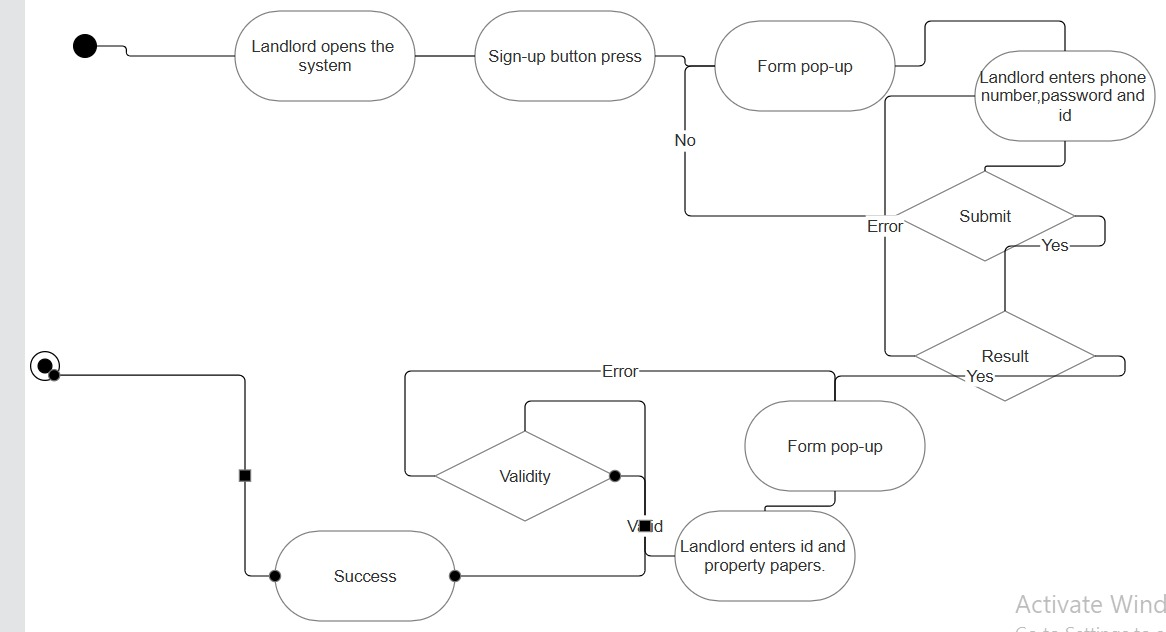




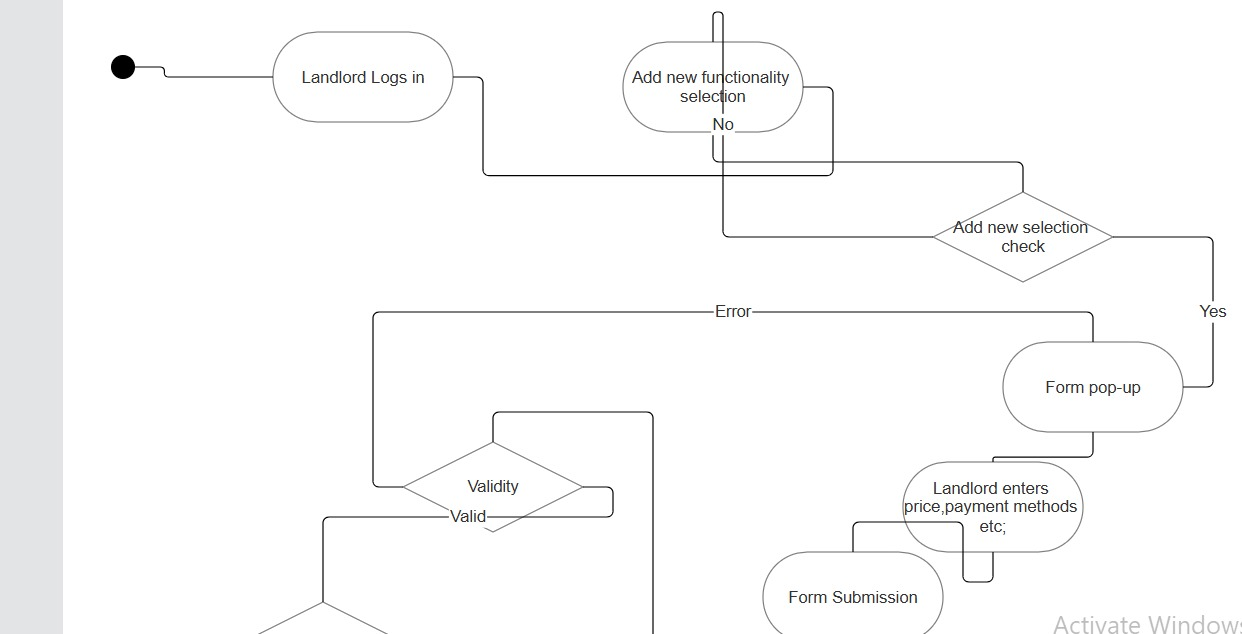
* UC\_ST\_07

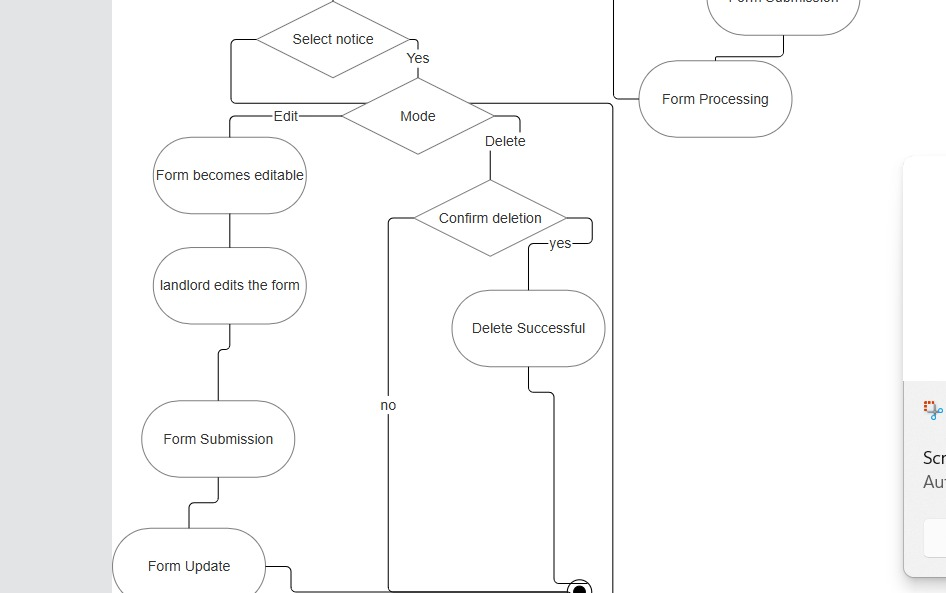


* UC\_LL\_01

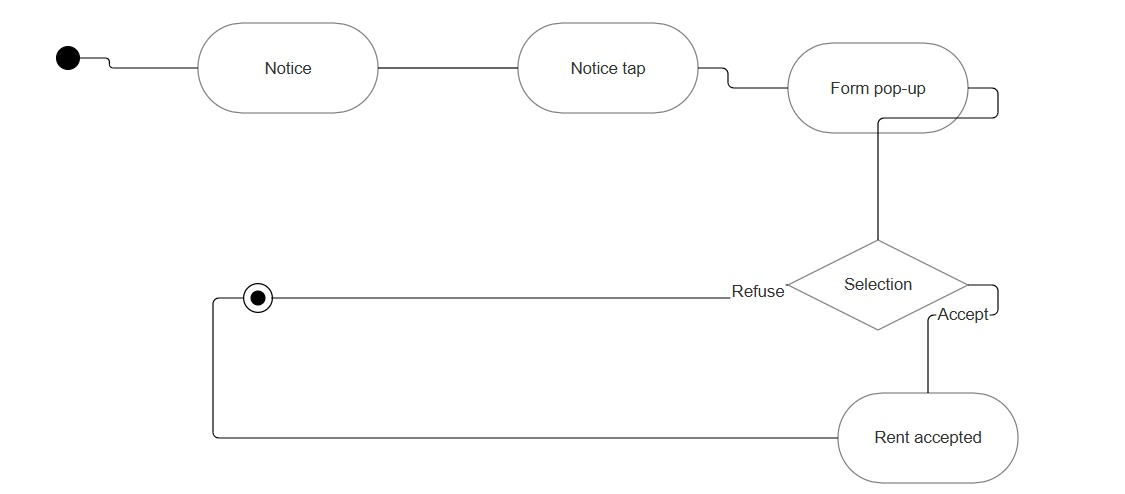


* UC\_LL\_02

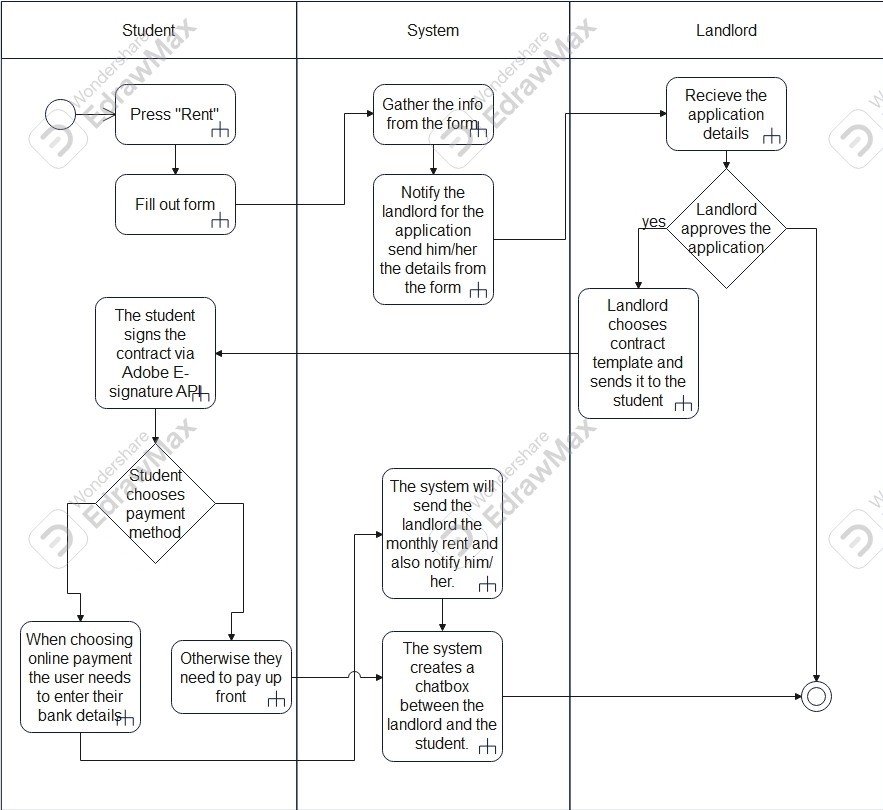




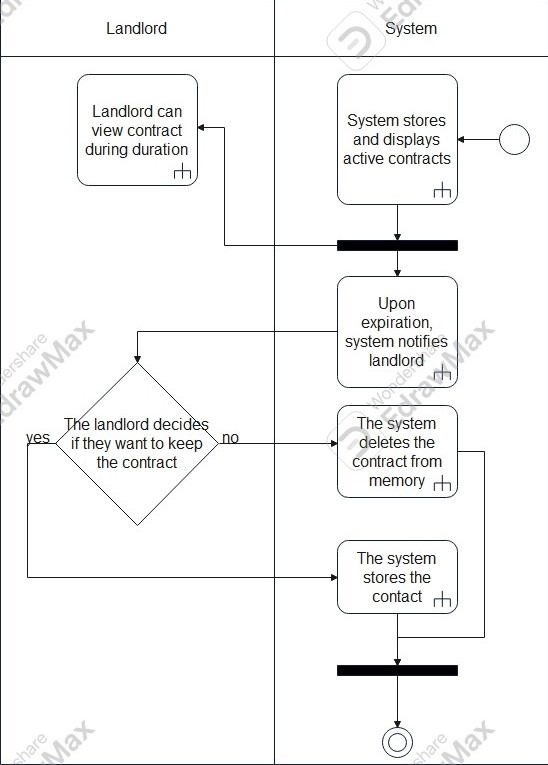
* UC\_LL\_03



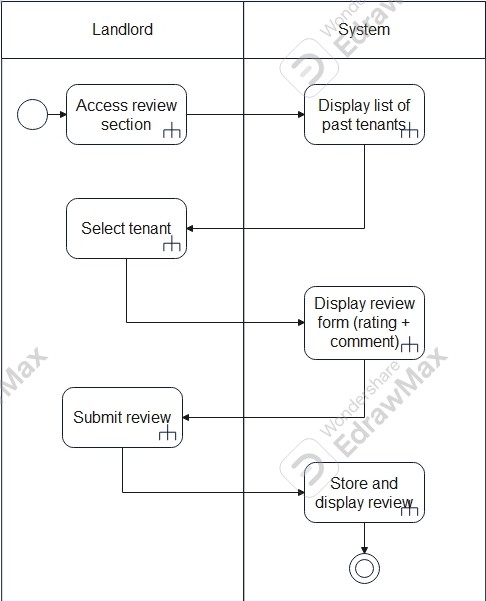
* UC\_LL\_04



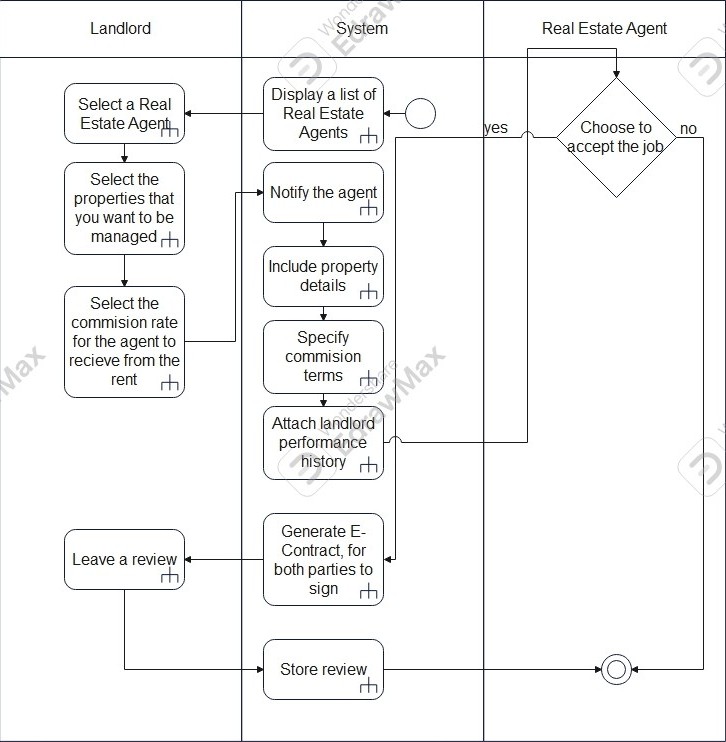
* UC\_LL\_05



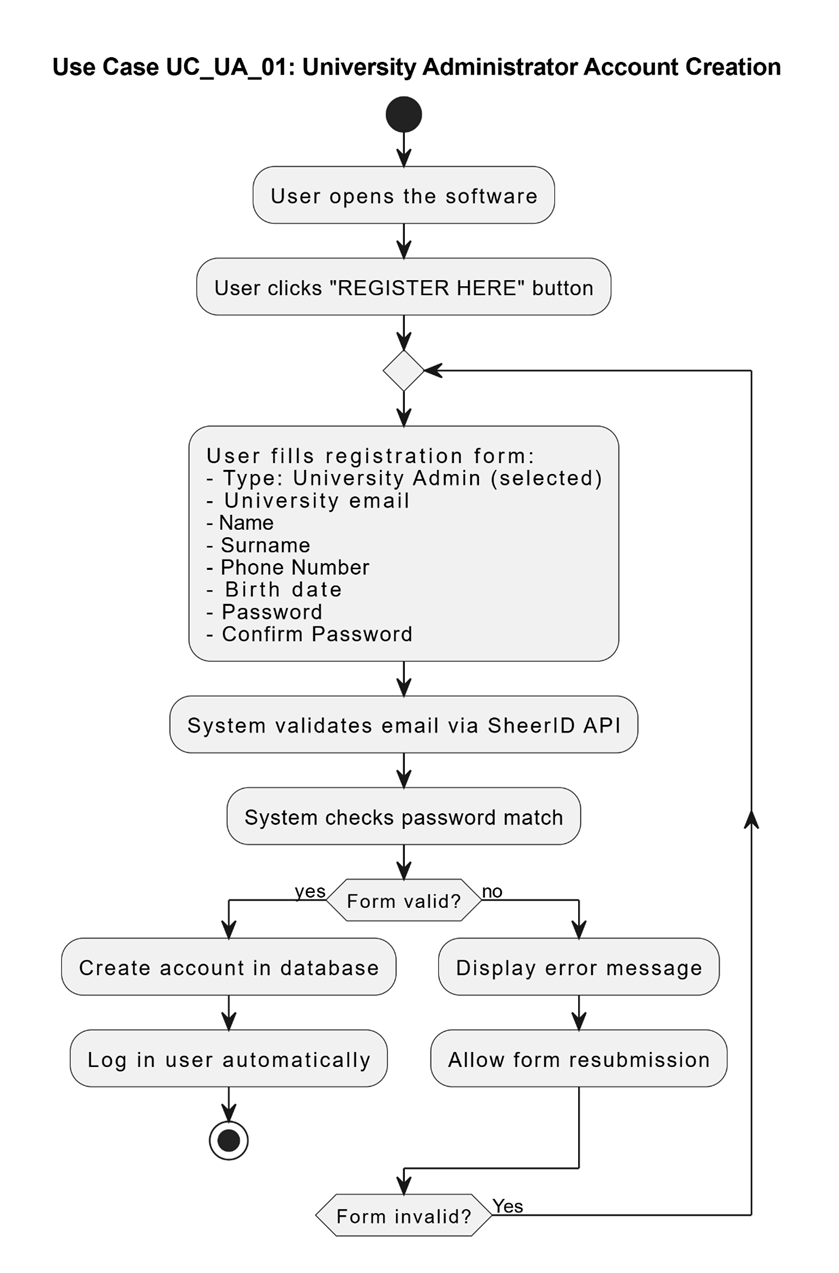
* UC\_LL\_06



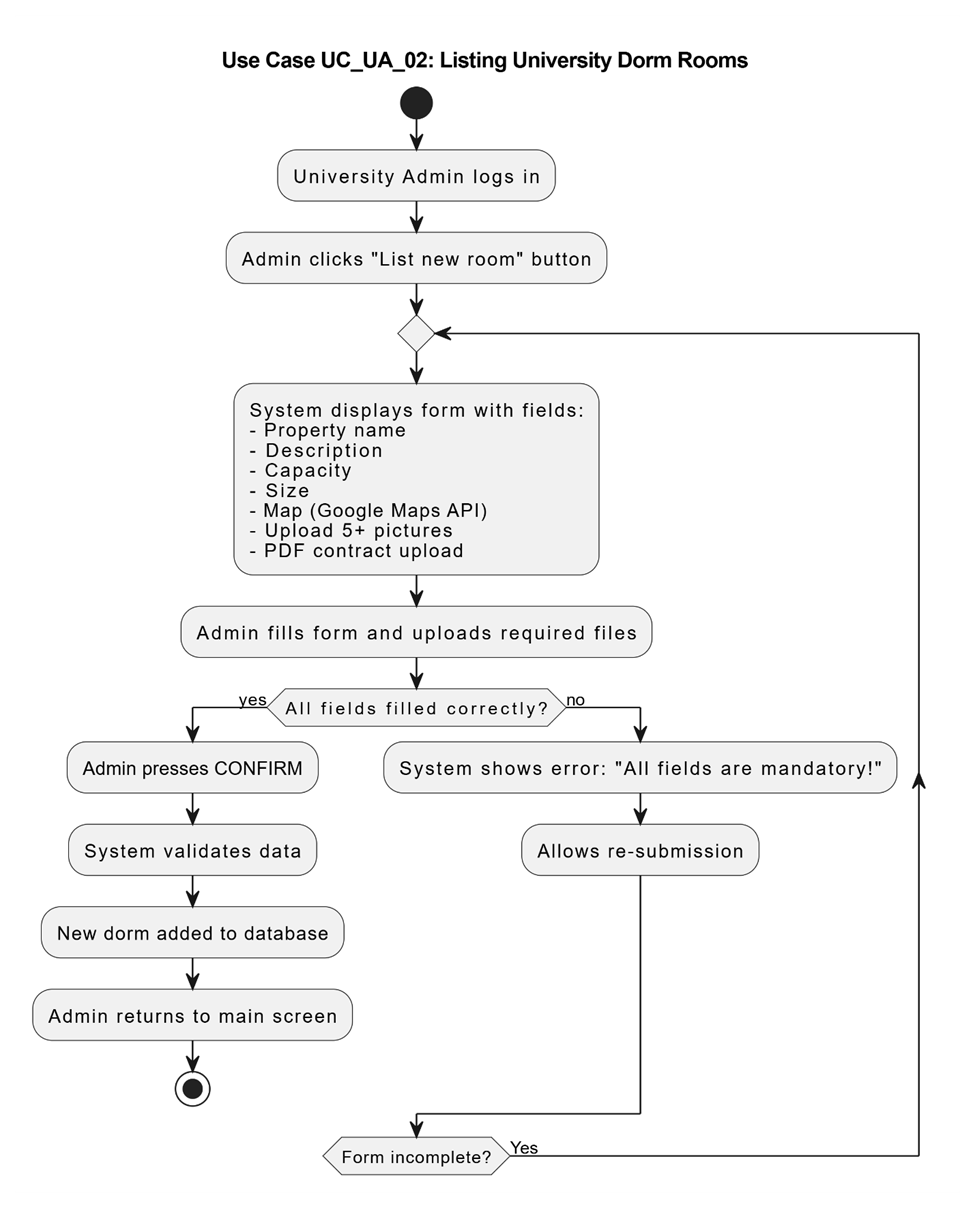
* UC\_LL\_07



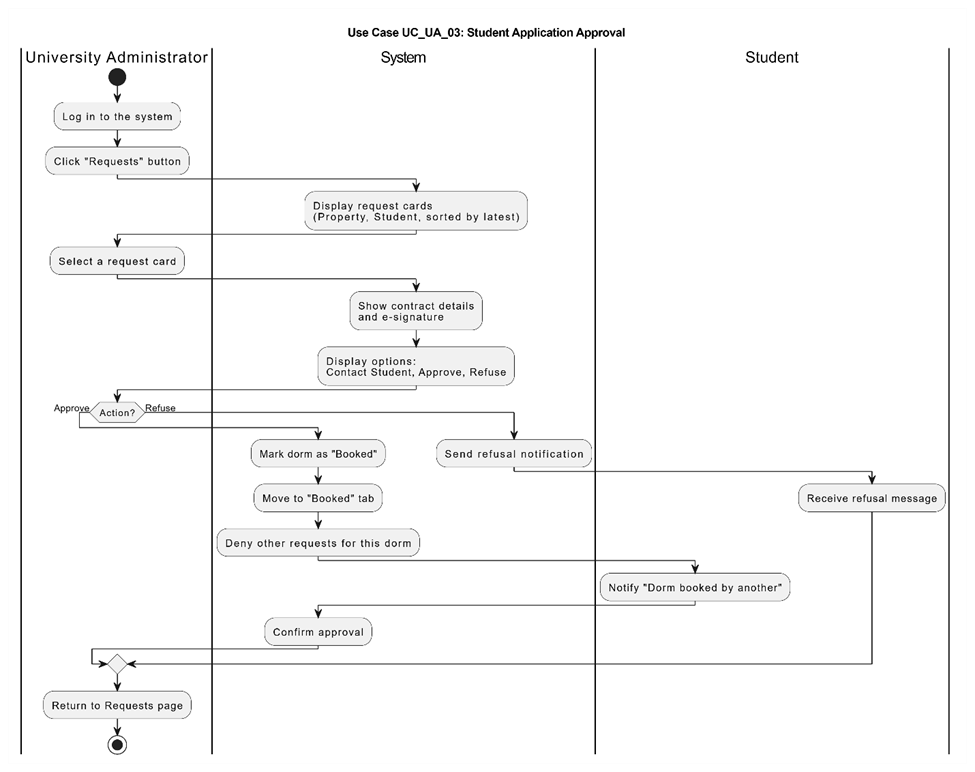
* UC\_UA\_01



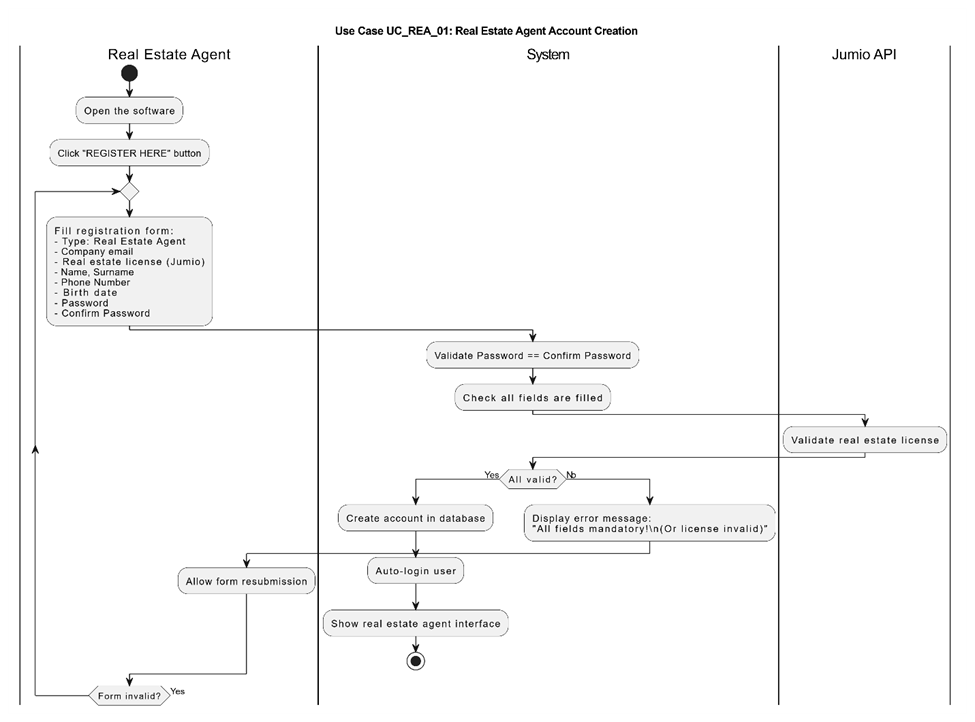
* UC\_UA\_02



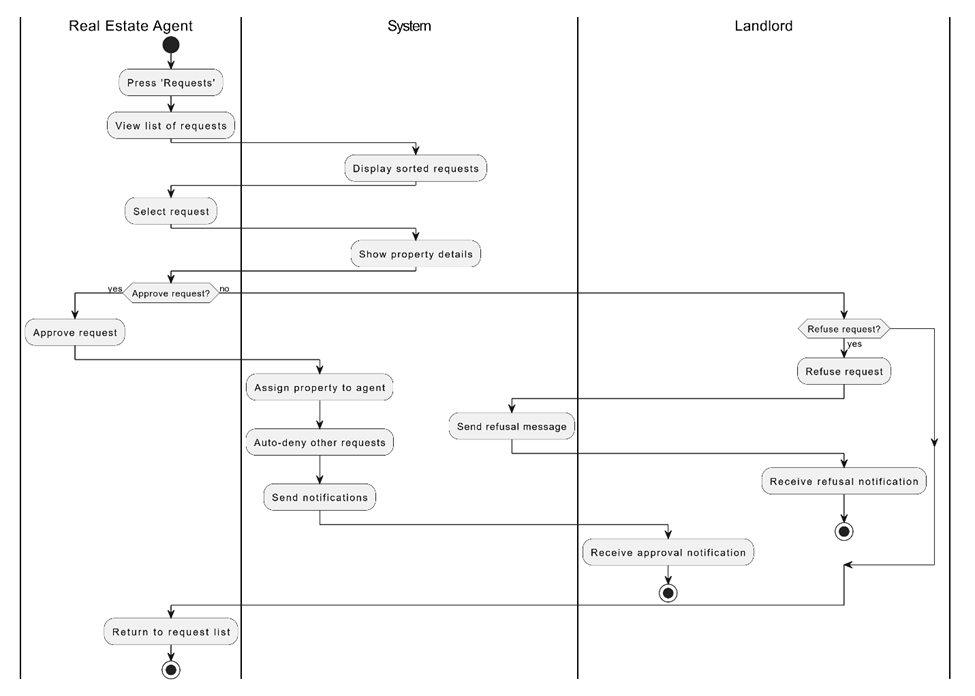
* UC\_UA\_03



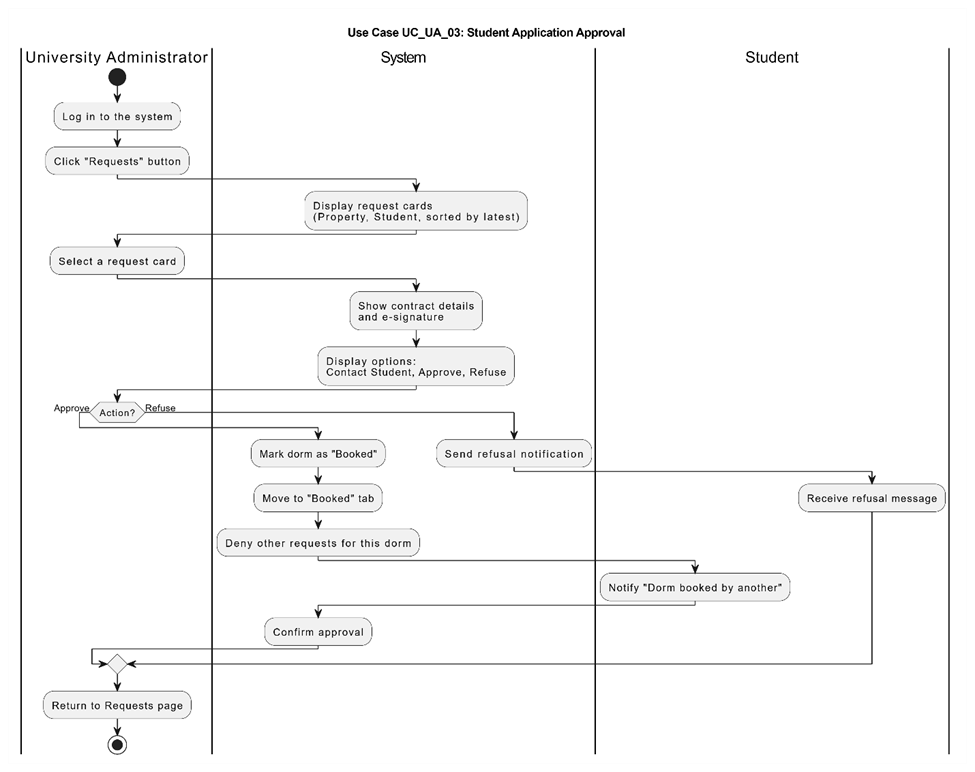
* UC\_REA\_01



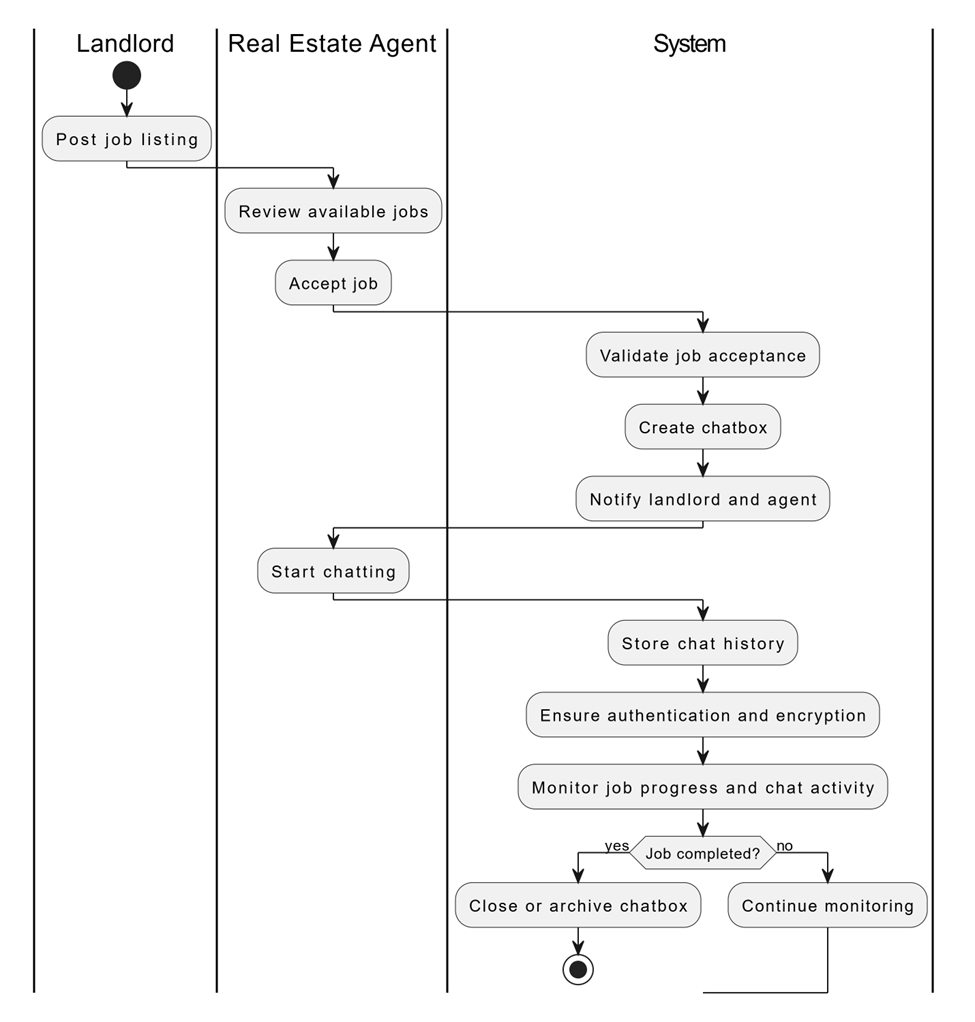
* UC\_REA\_02



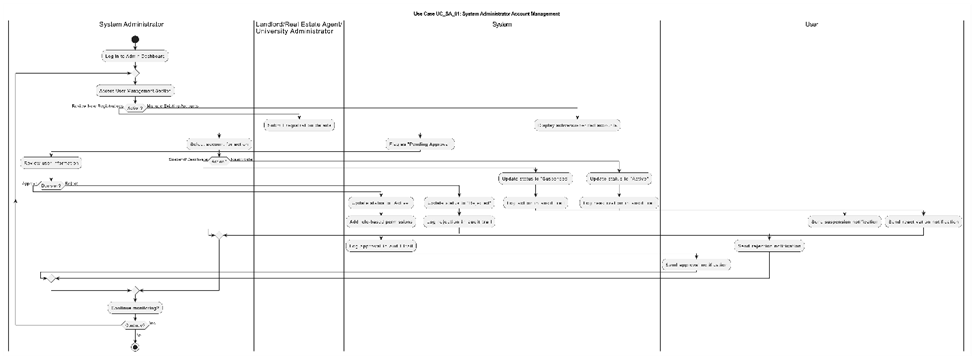
* UC\_REA\_03



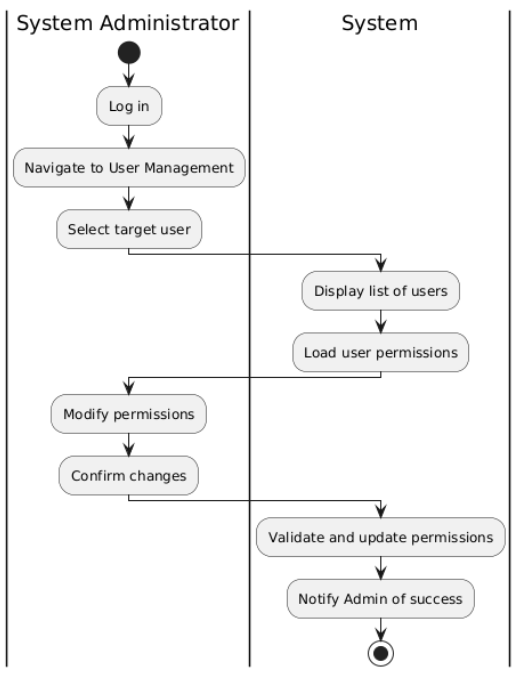
* UC\_REA\_04



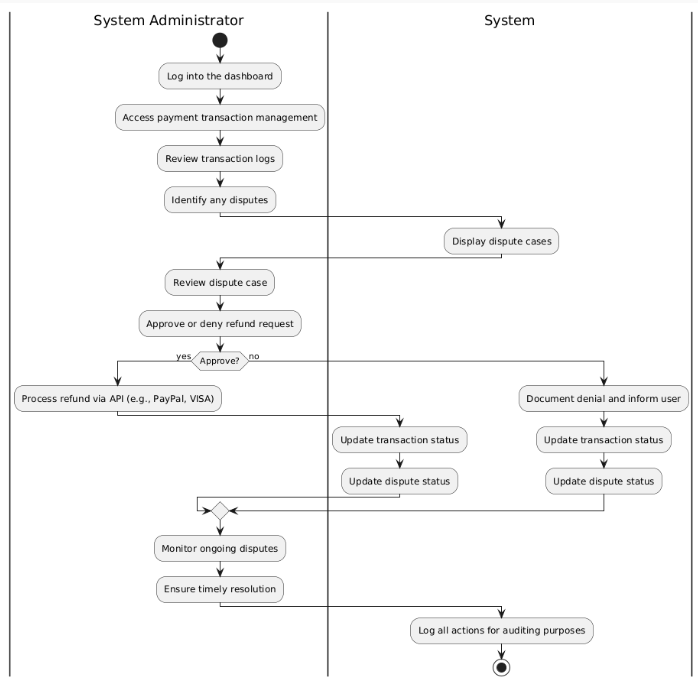
* UC\_SA\_01



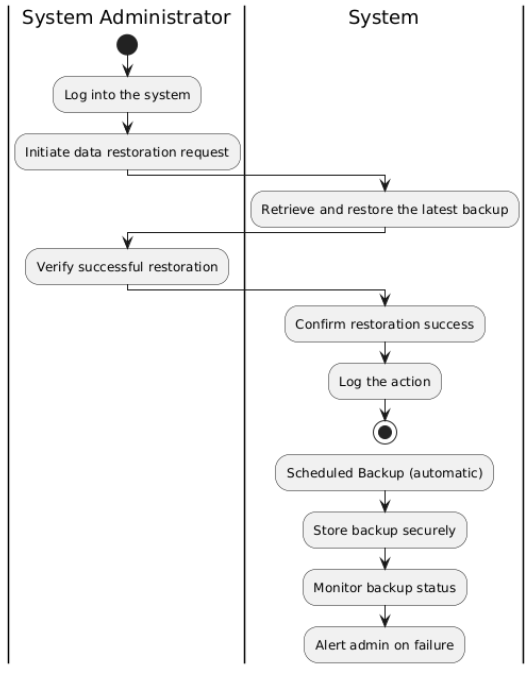
* UC\_SA\_02



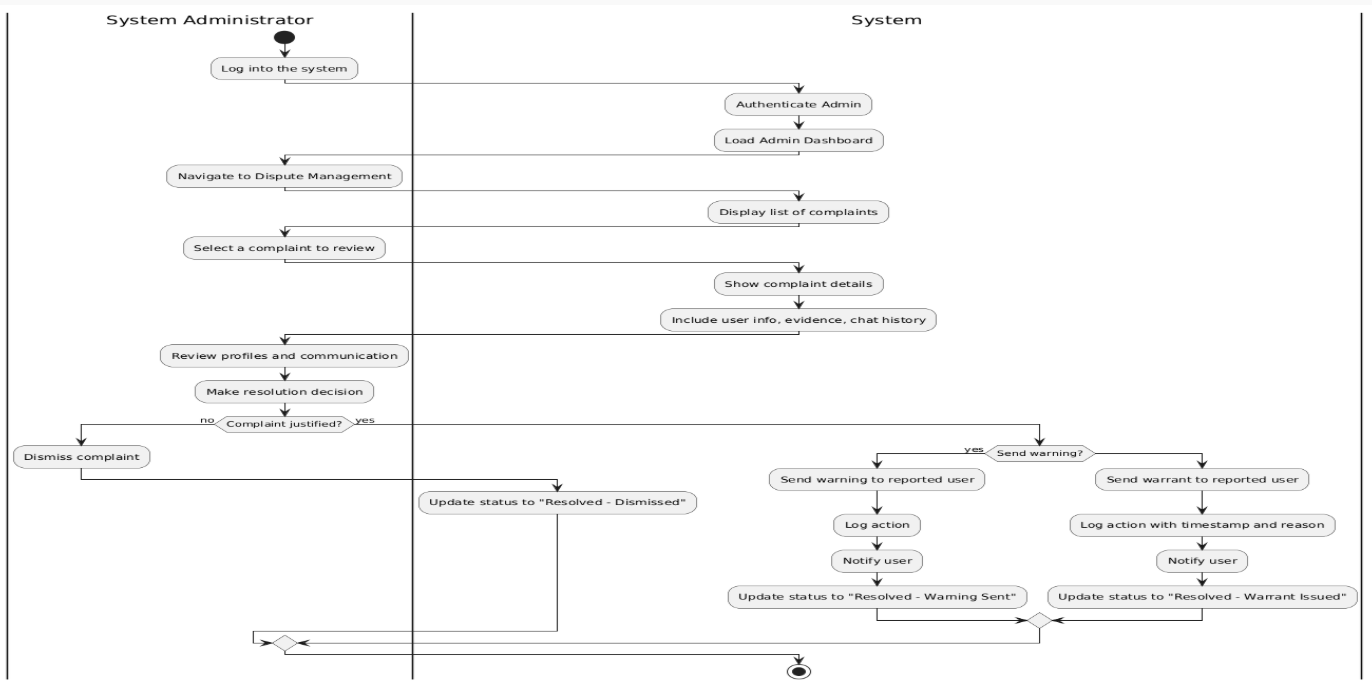
* UC\_SA\_03



* UC\_SA\_04

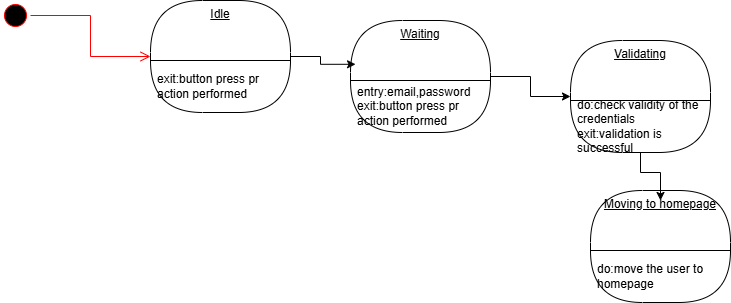


* UC\_SA\_05

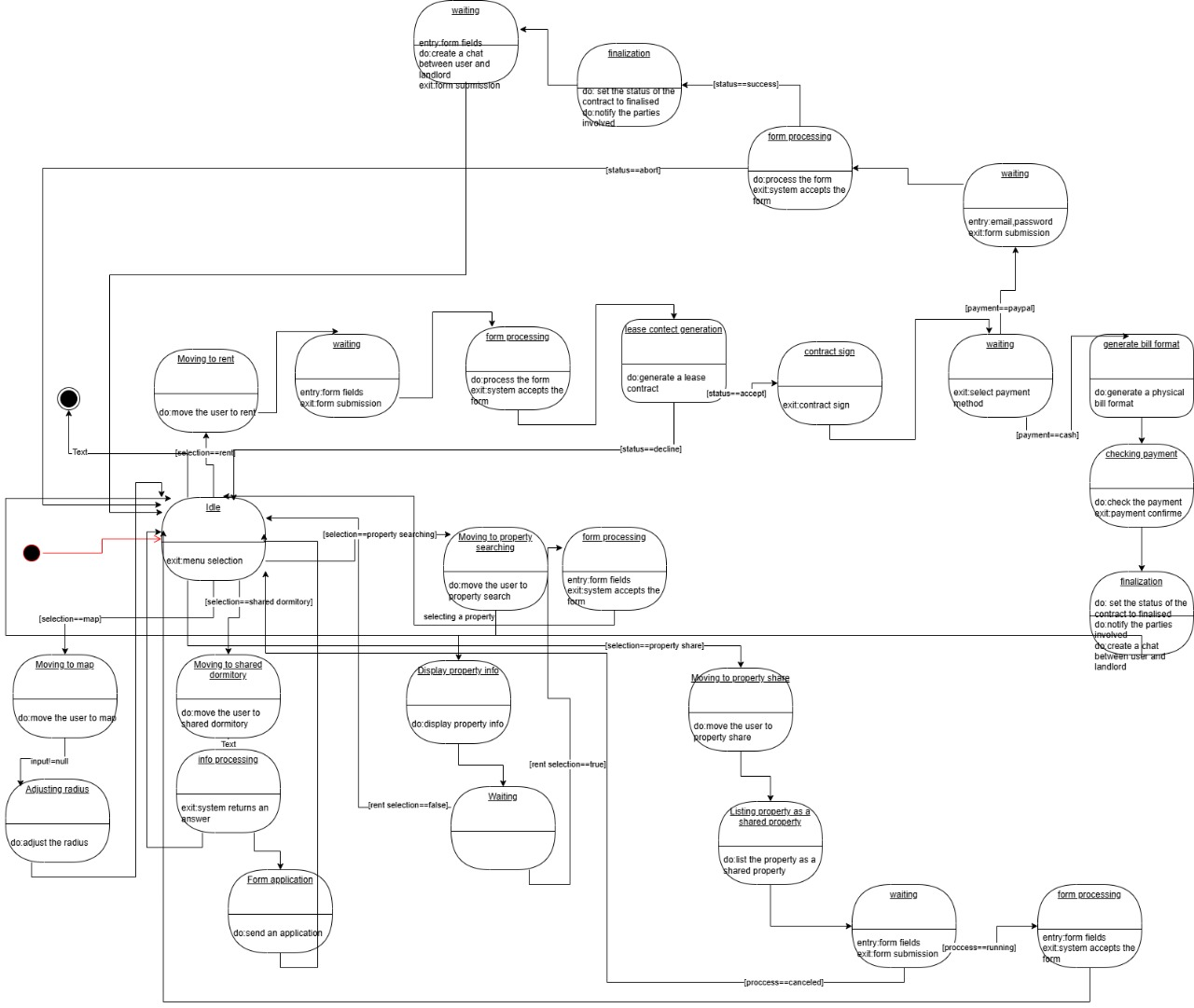


5.2 State Diagrams

* UC\_ST\_01



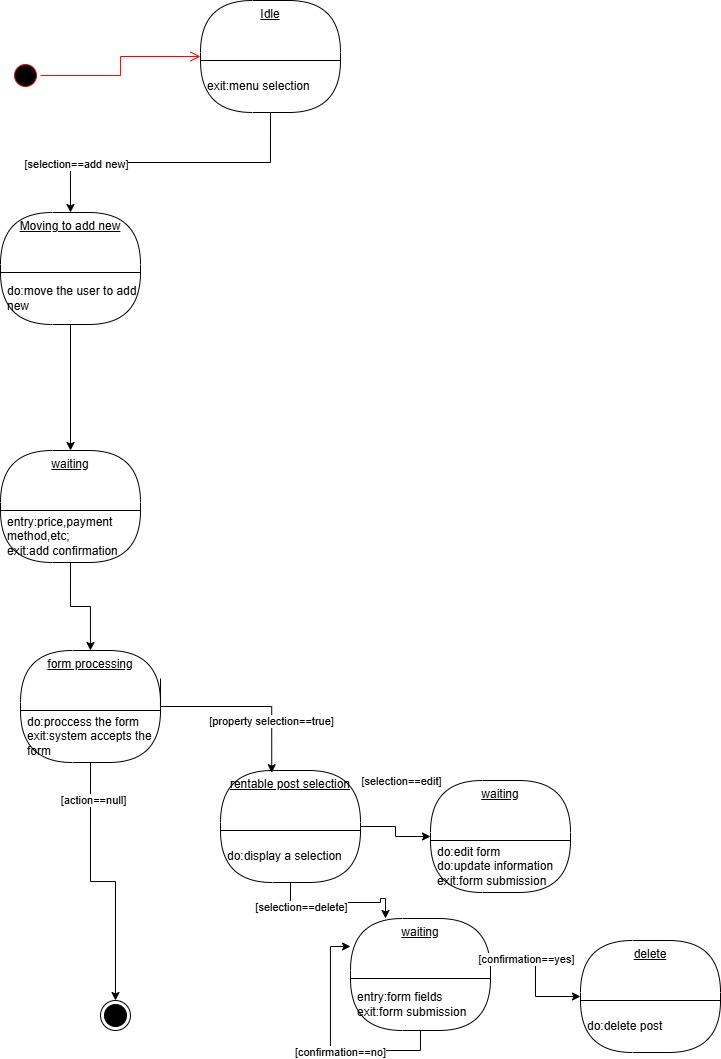
* UC\_ST\_02 to UC\_ST\_07



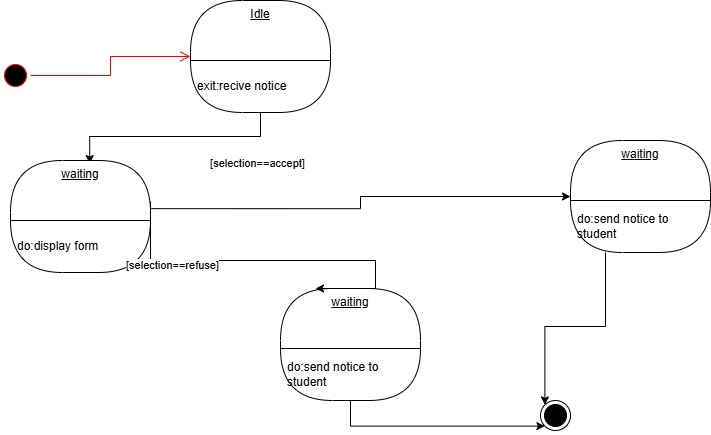
* UC\_LL\_01



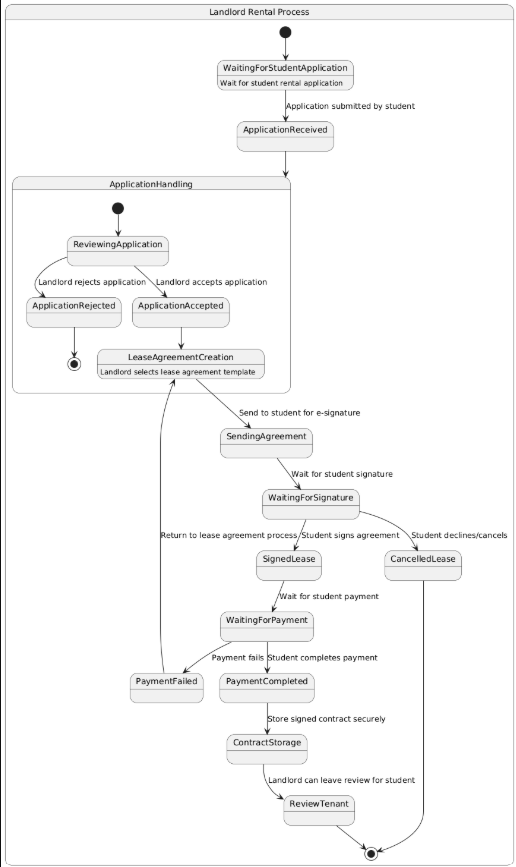
* UC\_LL\_02



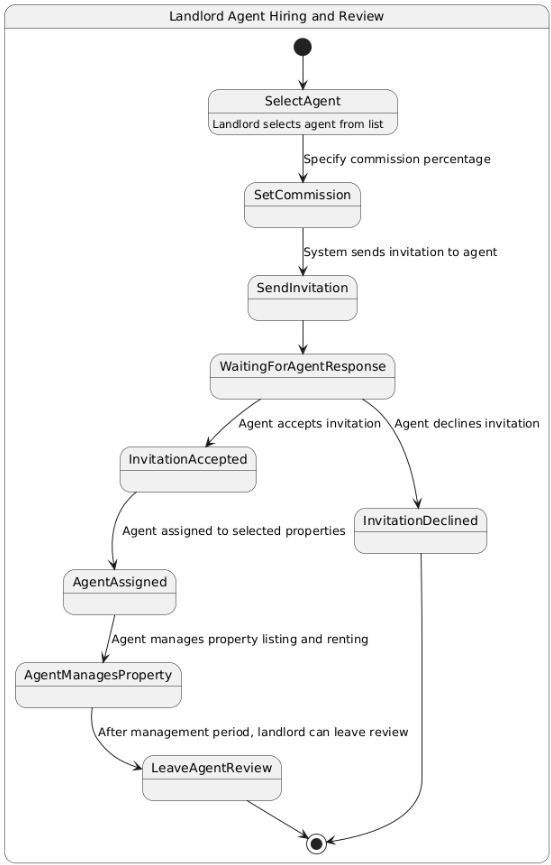
* UC\_LL\_03



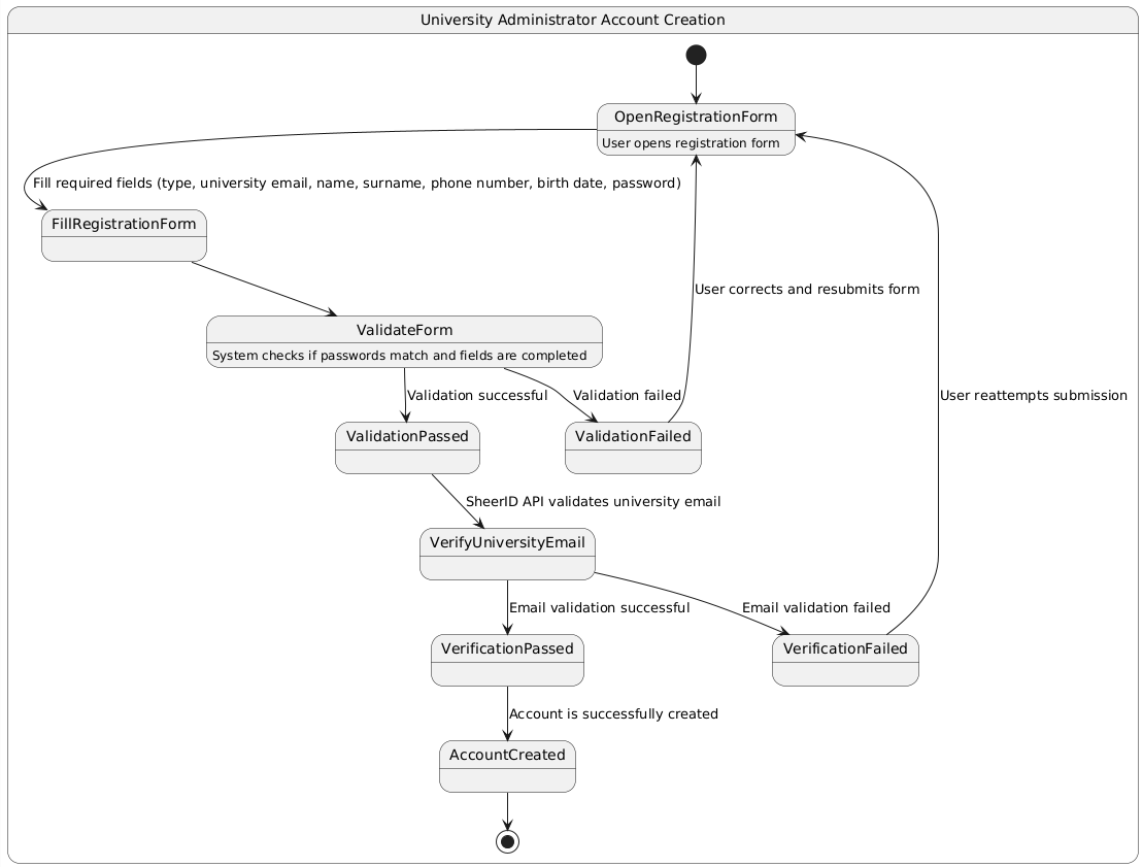
* UC\_LL\_04 to UC\_LL\_06



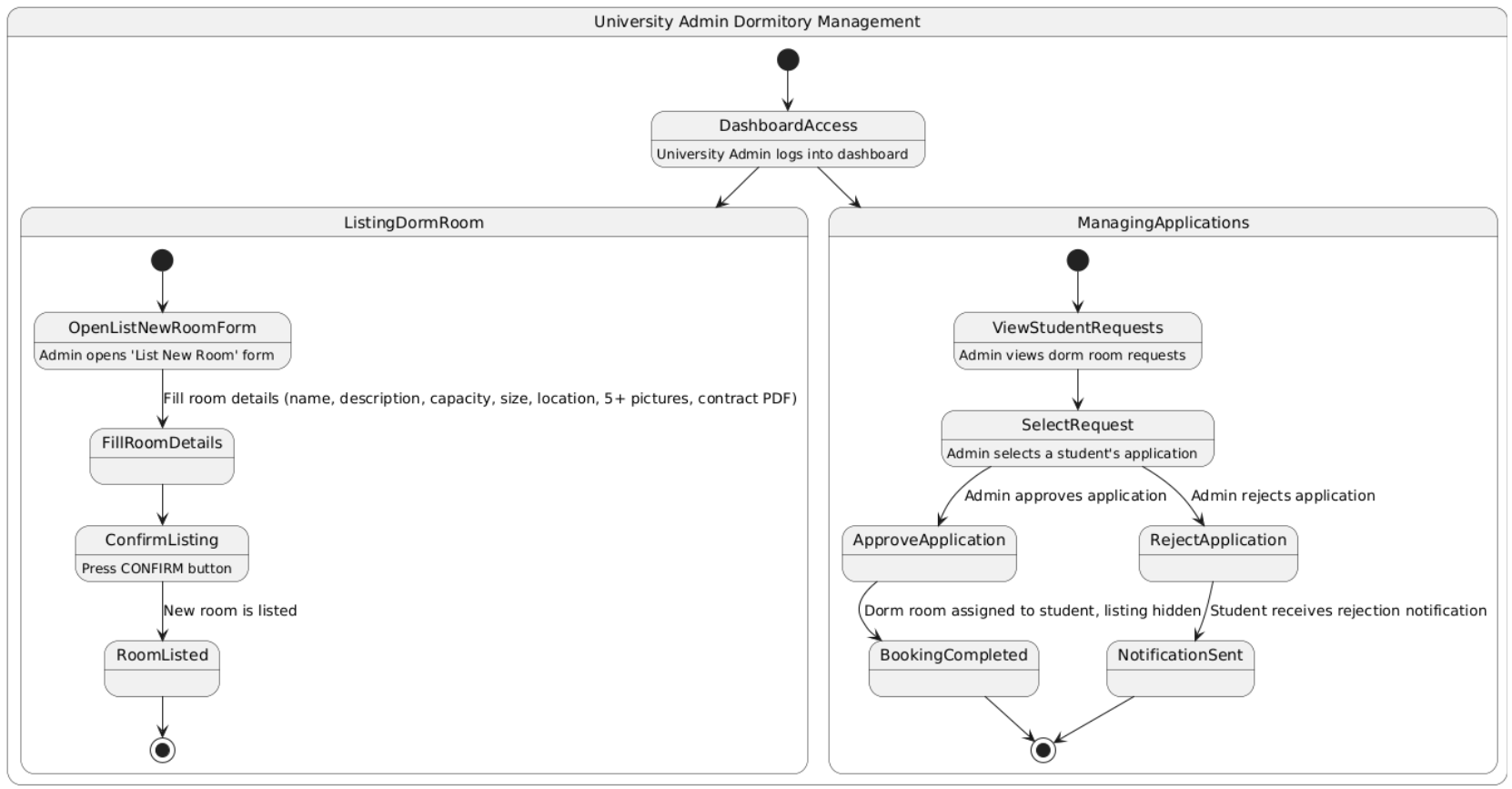
* UC\_LL\_07



* UC\_UA\_01



* UC\_UA\_02 to UC\_UA\_03



# 6. Design Patterns

Choose the relevant design patterns for your project. For each, give a reasoning and the associated class and sequence diagram. These are NOT part of the above diagrams, and need not carry the following naming scheme.

# 7. Appendix.

# **Organizing the Requirements**

This section is for information only as an aid in preparing the requirements document.

Detailed requirements tend to be extensive. Give careful consideration to your organization scheme. Some examples of organization schemes are described below:

**By System Mode**

Some systems behave quite differently depending on the mode of operation. For example, a control system may have different sets of functions depending on its mode: training, normal, or emergency.

**By User Class**

Some systems provide different sets of functions to different classes of users. For example, an elevator control system presents different capabilities to passengers, maintenance workers, and fire fighters.

**By Objects**

Objects are real-world entities that have a counterpart within the system. For example, in a patient monitoring system, objects include patients, sensors, nurses, rooms, physicians, medicines, etc. Associated with each object is a set of attributes (of that object) and functions (performed by that object). These functions are also called services, methods, or processes. Note that sets of objects may share attributes and services. These are grouped together as classes.

**By Feature**

A feature is an externally desired service by the system that may require a sequence of inputs to affect the desired result. For example, in a telephone system, features include local call, call forwarding, and conference call. Each feature is generally described in a sequence of stimulus-response pairs, and may include validity checks on inputs, exact sequencing of operations, responses to abnormal situations, including error handling and recovery, effects of parameters, relationships of inputs to outputs, including input/output sequences and formulas for input to output.

**By Stimulus**

Some systems can be best organized by describing their functions in terms of stimuli. For example, the functions of an automatic aircraft landing system may be organized into sections for loss of power, wind shear, sudden change in roll, vertical velocity excessive, etc.

**By Response**

Some systems can be best organized by describing all the functions in support of the generation of a response. For example, the functions of a personnel system may be organized into sections corresponding to all functions associated with generating paychecks, all functions associated with generating a current list of employees, etc.

**By Functional Hierarchy**

When none of the above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by common inputs, common outputs, or common internal data access. Data flow diagrams and data dictionaries can be used to show the relationships between and among the functions and data.

**Additional Comments**

Whenever a new Requirements Specification is contemplated, more than one of the organizational techniques given above may be appropriate. In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification.

There are many notations, methods, and automated support tools available to aid in the documentation of requirements. For the most part, their usefulness is a function of organization. For example, when organizing by mode, finite state machines or state charts may prove helpful; when organizing by object, object-oriented analysis may prove helpful; when organizing by feature, stimulus-response sequences may prove helpful; and when organizing by functional hierarchy, data flow diagrams and data dictionaries may prove helpful.