**[**Student Property Rental System**] Requirements Specification**

**Version** 3**.0**

March 8th**, 202**5

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**](#_2et92p0)

[**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](#_35nkun2)

[**3.2.1 Product Requirements 5**](#)

[3.2.1.1 Usability Requirements 5](#_q8yo01f084zc)

[3.2.1.2 Performance Requirements 5](#_ddqqf5pzexcw)

[3.2.1.3 Availability 6](#_vshy0pmxh14t)

[3.2.1.4 Security 6](#_8i9hvv5zz8n4)

[3.2.2 Organizational Requirements 6](#_sas9qa6j3xo9)

[3.2.3 External Requirements 6](#_3as4poj)

[**4. User Scenarios/Use Cases 6**](#)

[**5. Diagrams 7**](#_41mghml)

[**6. Design Patterns 7**](#_xr1wgl8t7616)

[**7. Appendix. 7**](#_uucczfw9bw0v)

[**Organizing the Requirements 7**](#_5l1q87u4mqs7)

# 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 | If a student wants to sign up, they have to provide a valid “.edu” email. | 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 his 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. 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 | If the student fills up the form correctly and decides to go through with their order by pressing the necessary button, then they will be provided with the contract that they have to sign via e-signature. | The contract needs to be signed, we will be using Adobe Sign for the e-Signature. | 1 | 10/03/2025 | Eden Pajo |
| FR\_ST\_06 | If the student signs the contract they are legally obliged to pay. Now they have to choose their payment method. If they choose an online payment method like banking card or paypal the system will provide them with the necessary steps. | In case they choose cash then they will have to pay in person. | 1 | 10/03/2025 | Eden Pajo |
| FR\_ST\_07 | At the end of their stay the system will provide the student with the capability to leave a review. They will be provided to leave a score of 1 to 5 and a comment regarding their stay. | This is entirely optional. | 1 | 10/03/2025 | Aleksander Murati |
| FR\_LL\_01 | The landlord will need to sign up with KYC regulations so basically using their id card besides their email. | 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 | Aleksander Murati |
| 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. |  | 1 | 10/03/2025 | Klajdi Murataj |

## ***Non-Functional Requirements***

### 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

Provide a summary of the major functions that the product will perform. Organize the functions to be understandable to the customer or a first time reader. Include use cases and business scenarios, or provide a link to a separate document (or documents). A business scenario:

* Describes a significant business need
* Identifies, documents, and ranks the problem that is driving the scenario
* Describes the business and technical environment that will resolve the problem
* States the desired objectives
* Shows the “Actors” and where they fit in the business model
* Is specific, and measurable, and uses clear metrics for success

Use cases are associated with a particular Functional Requirement. Assuming you have the first functional requirement named BR\_01, you will map it into the Use Case called UC\_01 and user scenario US\_01. Please keep this naming convention throughout all your use cases and diagrams.

# 5. Diagrams

# 

In this section you are going to place all of the diagrams that you build throughout to the course, in following with the slides presented throughout the weeks.

5.1 ER Diagram

Standard ERD for your project. Not much but the skills gained in the DBMS course are required.  
  
5.2 Use Case Diagram (general)

Use Case Diagram (only one, with all the use cases).

5.3 Activity Diagram

Each Activity Diagram should be associated with an use case, associated with a particular requirement which is further associated with a particular use-case. E.g BR\_01 which becomes UC\_01 which becomes AC\_01.

5.4. Class diagram.

One class diagram (general) for all the classes. Edit it afterwards with the design pattern implemented in it.

5.5 State diagram

Place all the relevant state diagrams here.

5.6 Sequence diagram.

All sequence diagrams are associated with an Activity Diagram. A Sequence Diagram is built based on an activity diagram. If the activity diagram is named AC\_07, the Sequence Diagram will be named SC\_07.

5.7. Collaboration diagram

All collaboration diagrams directly relate to a sequence diagram. If a sequence diagram is named SC\_07, then the collaboration diagram is named CC\_07

# 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.