Mohamed Farook Fazrin

[Email address]

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Student Accommodation Booking System

[Document subtitle]

# Task A

## Purpose

The students of XYZ University are always seeking for properties for accommodation closer to their campus area. To support the student needs the university is planning to host an online Student accommodation system (SAS) to provide details of property-owners, their properties and of students seeking accommodation. As of today, this whole operation is manual.

XYZ University has decided to automate above operations to achieve following objectives

1. Better and faster service to students

2. Better services to property owners

3. Increase the communication between students and property owners

The purpose of this document is to identify the high-end requirements for developing a web application system, to coordinate and monitor students and property owners for XYZ university. This document will explain operations of the student accommodation booking system which need to be monitored and optimized. The proposed system should automate the manual process

Some of the required functionalities are

* The Student ID, First Name, Last name, address email and telephone number of property-owners are kept in SAS. Each property is allocated a unique identifier as well as details such as the address, type of property [ex: Flat, terraced house, detached house...etc.], the maximum number of tenants it is suitable for and the amount of the rent. A fee is charged for each property that is added to the system.
* Students can register with the service providing their Student ID, First Name, Last name, address email and telephone number. The registered students will be provided with a list of available properties. Further the student can request for viewing the properly and system will notify this to the property-owners.
* The property-owners can update the system if their property is available or not available for rent.
* Also, the system must have the facility to provide reports such as available properties, registered students and current occupation details etc.

## List of Stakeholders

This proposed system will be expecting to be used by the following users

1. Students
2. Property Owners
3. Admin

## Project Scope

Once this system is implemented, whole manual processes related to accommodation booking process could be tracked in the system. Therefore, the project scope will have to be covered with following functional areas;

1. Login page for property owners and students

2. Manage property details by property owners

3. Search property details by students

4. Request to view the property through system

5. Accept/Reject the view request for property by respective property owners

6. Make a payment through system

7. View property details and availability of the property

8. View the student’s details

## Requirement Clarity Index

### 1.4.1 Functional Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Requirements | Description | Risk level | RCI |
| 1 | Use Accounts | User authentication with different types of privileges and access levels | Medium | 100% |
| 2 | Authentication | Proper Login system with username and password for each user (admin, student, owners ) | High | 100% |
| 3 | Student registration | Register student details | Medium | 100% |
| 4 | Edit student details | Edit the existing student details | Medium | 100% |
| 5 | Delete student details | Delete the existing student account | Medium | 100% |
| 6 | Owner registration | Register owner details | Medium | 100% |
| 7 | Edit owner details | Edit the existing owner details | Medium | 100% |
| 8 | Delete owner details | Delete the existing owner account | Medium | 100% |
| 9 | Create property details | Create new property details | Medium | 100% |
| 10 | Edit property details | Edit the existing property details | Medium | 100% |
| 11 | Delete property details | Delete the existing property details | Medium | 100% |
| 12 | Change the status of a property | Change property status as available or occupied | Medium | 100% |
| 13 | Search rooms by various criteria | Search room by specific parameters such as price, maximum number of tenants can stay, | Medium | 100% |
| 14 | View Request | Student can request to owners to view the property on specific date | Medium | 100% |
| 15 | Accept/Reject Request | Owners can accept or reject by request to view by student | Medium | 100% |
| 16 | View available properties | Report of all available properties | Medium | 100% |
| 17 | View all students | View all registered students | Medium | 100% |

### 1.4.2 Non-Functional Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Requirements | Description | Risk level | RCI |
| 1 | Web server | A web server to host the web application | High | 100% |
| 2 | Concurrency management | Server should be able to handle multiple concurrent requests by students and owners | High | 100% |

# Task B

## Level 1 Use case diagram

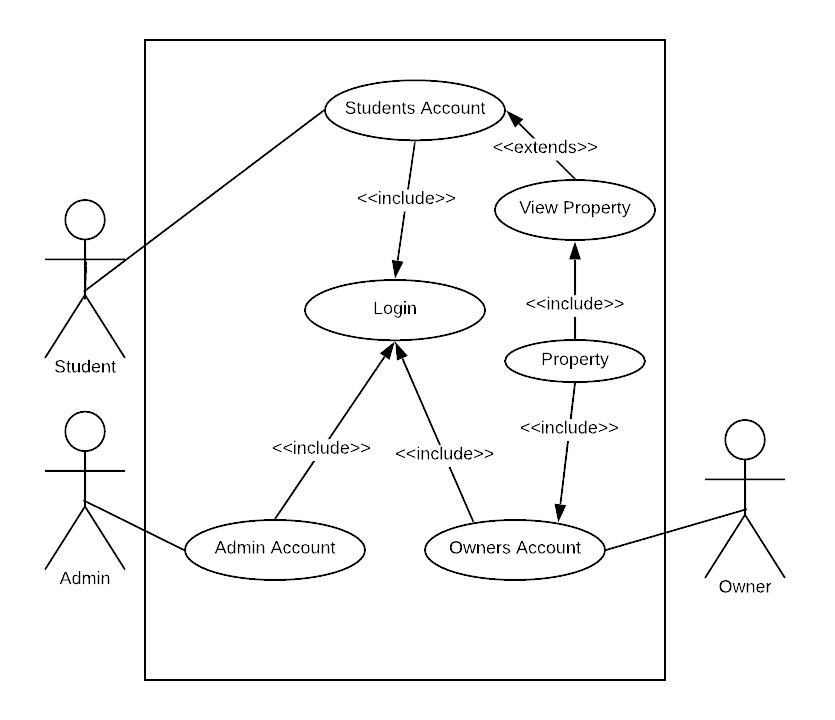


Figure 1 level 1 use case diagram

### Level 2 Use case diagram for student

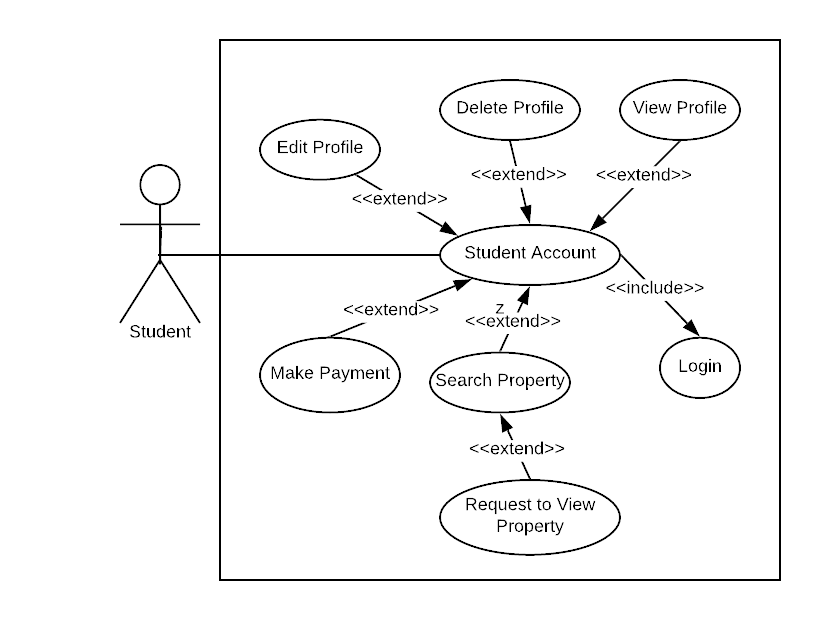


Figure 2 use case diagram for student

### Level 2 Use case diagram for owner

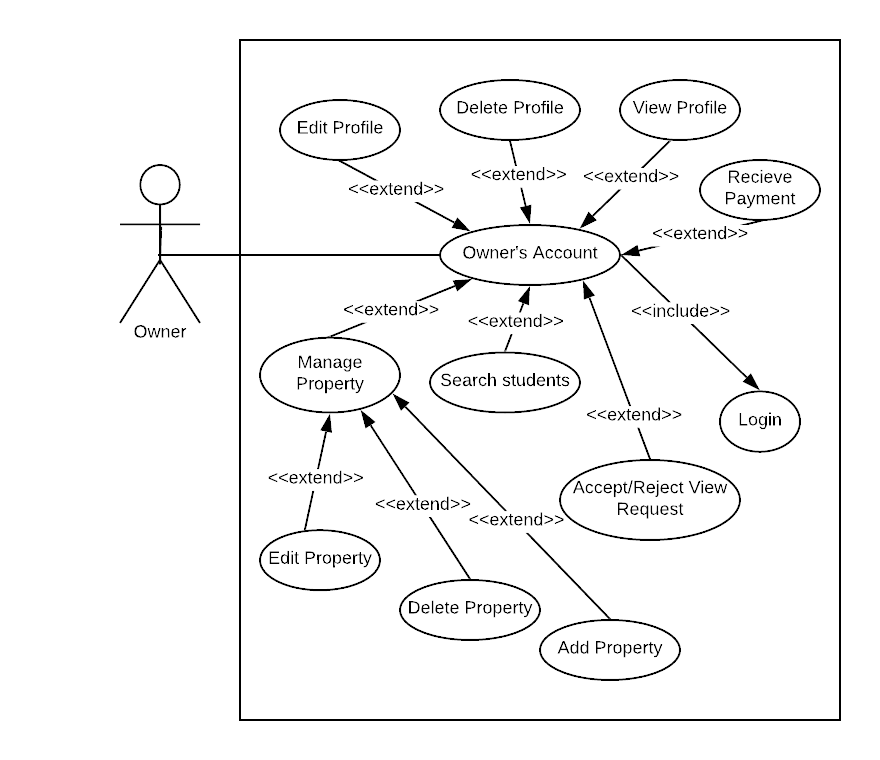


Figure 3 Level 2 Use case diagram for owner

### Level 2 Use case diagram for admin

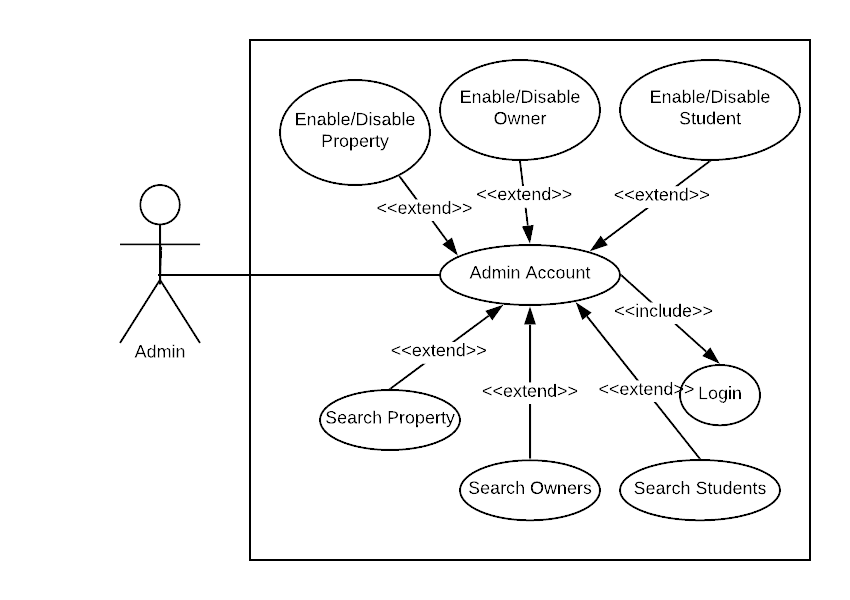


Figure 4 Level 2 Use case diagram for admin

## Class Diagram

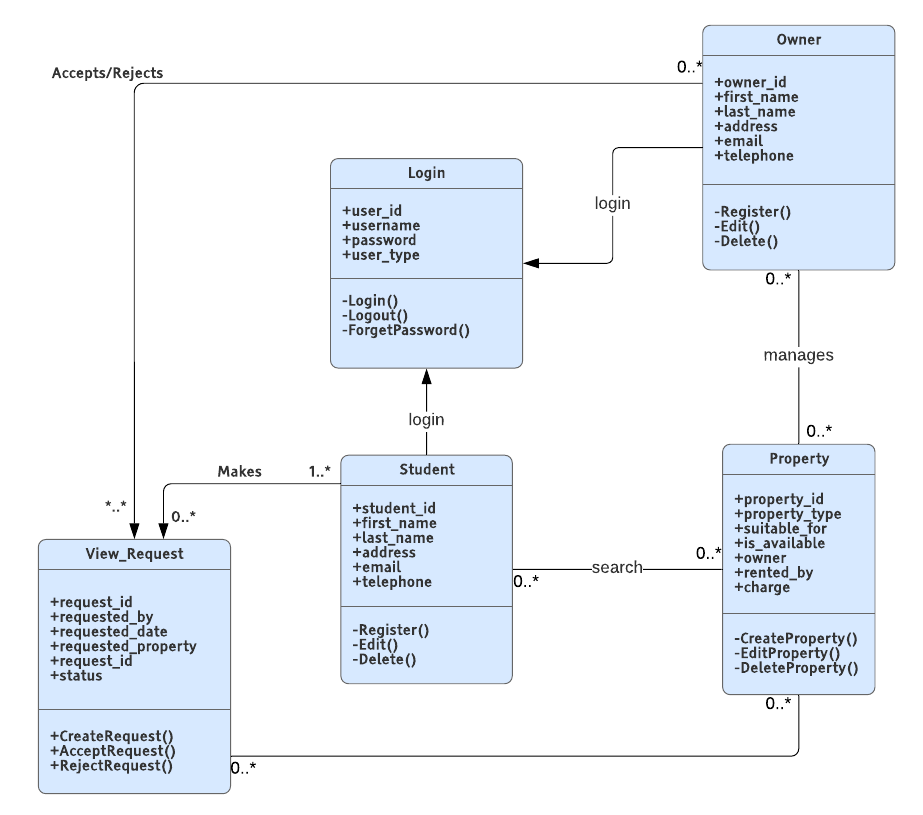


Figure 5 class diagram for student accommodation booking system

### Class diagram explanation

|  |  |
| --- | --- |
| Class Names | Description |
| Login | Login class will handle authentication, authorization and forget password functions for student, owner and admin |
| Owner | Owner class will handle create, edit and delete owner details functions |
| Property | Property class will handle create, edit and delete property details this will including manage the availability of the property |
| Student | Student class will handle create, edit and delete student details |
| View\_Request | View request will handle create, accept and rejecting the view request of the property |

Table 1 Class diagram explanation table

## Sequence Diagram for accommodation booking system

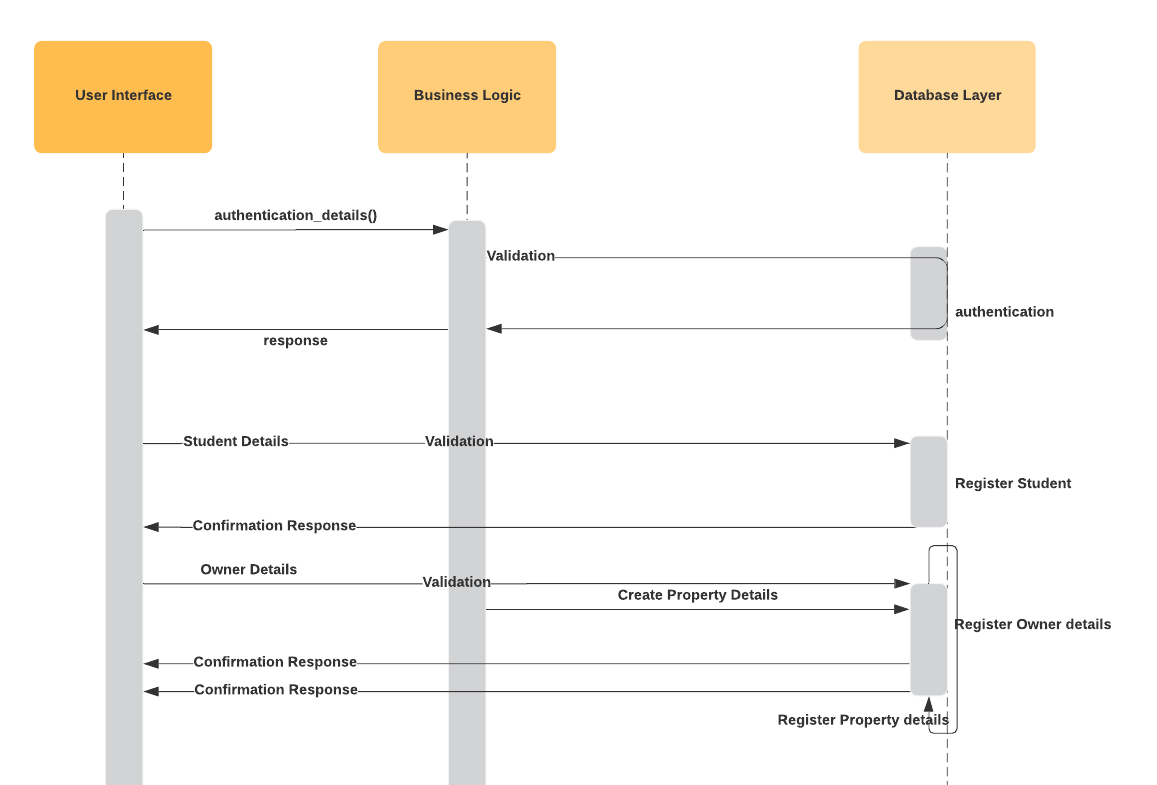


Table 2 Sequence diagram for accommodation booking system

# Task C

Design patterns are conceptual tools for solving complex software problems. These patterns are simple and elegant solutions that have evolved over time and may have become generally accepted as the best way to address certain design (RudraSambyal, 2018),

## Singleton Design Pattern

The Singleton pattern encapsulates a shared resource within a single unique class instance. This instance arbitrates access to the resource and storage-related state information. A class method provides the reference to this instance, so there is no need to pass the reference around. Any object that has access to the Singleton’s class header can use the Singleton (RudraSambyal, 2018).

Following image will explain the overview of the singleton design pattern

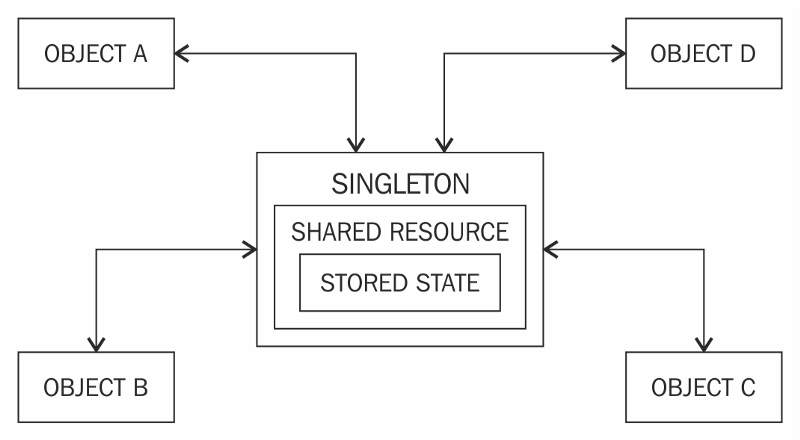


Figure 6 overview of the singleton design pattern

This design pattern defines the structure of a class that can have only one instance. A Singleton encapsulates a unique resource and makes it readily available throughout the application. The resource might be hardware, a network service, a persistent store, or anything else that can be modelled as a unique object or service (RudraSambyal, 2018).

### Implementation

Implementation of the Singleton pattern often typically creates a single object using the factory method, and this instance/object is called a shared instance in most cases. Since the access to the instance is passed on through a class method, the need to create an object is eliminated (RudraSambyal, 2018).

### Singleton design pattern : cons

* Singletons hinder unit testing: A Singleton might cause issues for writing testable code if the object and the methods associated with it are so tightly coupled that it becomes impossible to test without writing a fully-functional class dedicated to the Singleton (RudraSambyal, 2018).
* Singletons create hidden dependencies: As the Singleton is readily available throughout the code base, it can be overused. Moreover, since its reference is not completely transparent while passing to different methods, it becomes difficult to track (RudraSambyal, 2018).

### Singleton design pattern : pros

To avoid these complications, when considering the Singleton pattern, need to make certain that the class is a Singleton. Also, while thinking of designing the Singleton design pattern, keep testing in mind and use dependency injection whenever possible — that is, try to pass the Singleton as a parameter to the initializer whenever possible (RudraSambyal, 2018).

## Factory Method Pattern

A Factory Pattern or Factory Method Pattern says that just define an interface or abstract class for creating an object but let the subclasses decide which class to instantiate. In other words, subclasses are responsible to create the instance of the class (javatpoint, 2019).

### Advantage of Factory Design Pattern

Factory Method Pattern allows the sub-classes to choose the type of objects to create. It promotes the loose-coupling by eliminating the need to bind application-specific classes into the code. That means the code interacts solely with the resultant interface or abstract class, so that it will work with any classes that implement that interface or that extends that abstract class (javatpoint, 2019).

### Usage of Factory Design Pattern

* When a class doesn't know what sub-classes will be required to create
* When a class wants that its sub-classes specify the objects to be created.
* When the parent classes choose the creation of objects to its sub-classes.

(javatpoint, 2019)

## Abstract Factory Pattern

Abstract Factory pattern is almost like Factory Pattern is considered as another layer of abstraction over factory pattern. Abstract Factory patterns work around a super-factory which creates other factories (Kumar, 2015).

Abstract factory pattern implementation provides a framework that allows to create objects that follow a general pattern. So, at runtime, abstract factory is coupled with any desired concrete factory which can create objects of desired type (Kumar, 2015).

Abstract Factory provides interfaces for creating families of related or dependent objects without specifying their concrete classes. Client software creates a concrete implementation of the abstract factory and then uses the generic interfaces to create the concrete objects that are part of the family of objects. The client does not know or care which concrete objects it gets from each of these concrete factories since it uses only the generic interfaces of their products. So, with this idea of Abstract Factory pattern, we will now try to create a design that will facilitate the creation of related objects (Kumar, 2015).

### Advantages of Abstract Factory Pattern

This pattern is particularly useful when the client doesn’t know exactly what type to create.

**Isolation of concrete classes:** The Abstract Factory pattern helps you control the classes of objects that an application creates. Because a factory encapsulates the responsibility and the process of creating product objects, it isolates clients from implementation classes. Clients manipulate instances through their abstract interfaces. Product class names are isolated in the implementation of the concrete factory; they do not appear in client code.

**Exchanging Product Families easily:** The class of a concrete factory appears only once in an application, that is where it’s instantiated. This makes it easy to change the concrete factory an application uses. It can use various product configurations simply by changing the concrete factory. Because an abstract factory creates a complete family of products, the whole product family changes at once (Kumar, 2015).

**Promoting consistency among products:** When product objects in a family are designed to work together, it’s important that an application use objects from only one family at a time. Abstract Factory makes this easy to enforce (Kumar, 2015).

### Disadvantages of Abstract Factory Pattern

Difficult to support new kind of products: Extending abstract factories to produce new kinds of Products isn’t easy. That’s because the Abstract Factory interface fixes the set of products that can be created. Supporting new kinds of products requires extending the factory interface, which involves changing the Abstract Factory class and all its subclasses (Kumar, 2015).