POLISH-JAPANESE ACADEMY
OF INFORMATION TECHNOLOGY

Final Project Documentation

Valerii Trembovetsky
s20970

Hotel Management System

1. Story

The system should store clients and employees, for both we want to remember first name, last name, date of birth, phone number and be able to get an age. Client may leave an email if he wishes.

Rental company apartments are described by their unique number, area, tags and type. Every apartment should have at least one mature bed. Apartment has four types: regular, VIP, family and penthouse. VIP apartments have free drinks, we want to know how many. Family apartments differ with children beds, and the penthouse may have a roof entrance. The apartments have the price for a day of rent. The price differs depending on the type of apartment. Apartments may have multiple types.

Clients can rent multiple apartments. Unique apartment assigned to a client with its number. For rent the system should remember the start date, end date, total price and be able to calculate rent days.

Apartment consists of rooms. Rooms are described by the name and description with no more than 300 words. Rooms have facilities for which the system should remember name, price, weight, manufacturer and serial number if the facility has it.

In company work Porter, maiden, chef and manager. For each employee we want to remember PESEL, Date of hiring, how many days the person is employed and the date of living if the person is not working anymore. Every employee gets a salary according to their position and has a password and work email to log in into the system. The system should keep the information about the porter’s english level, in what cuisine the chef is specialised, minimal apartments which the maiden should clean per day and in cleaning what apartment is specialised the maiden. For all managers we want to keep the minimal amount of subordinates and for a single manager amount of his subordinates.

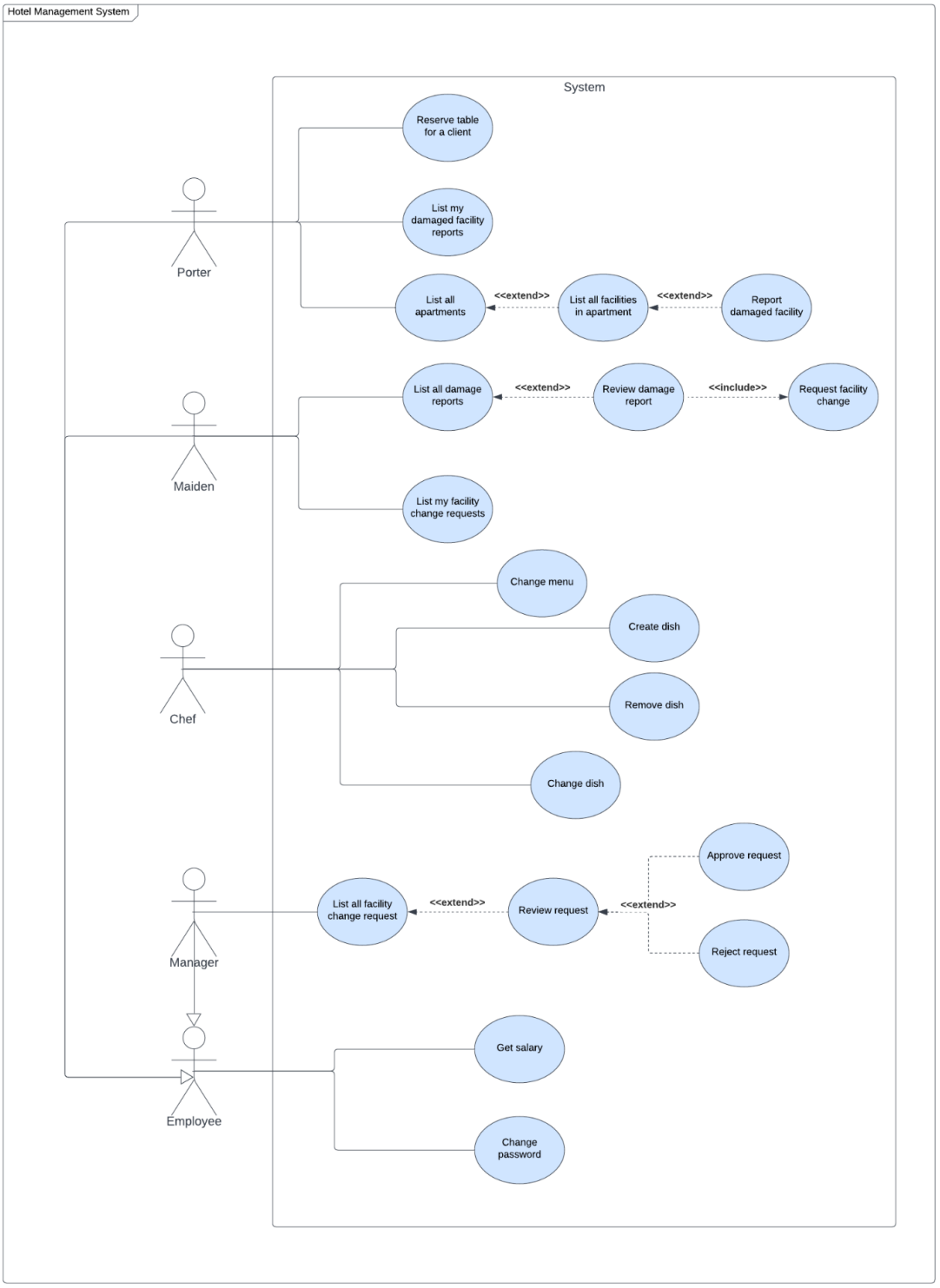
Employed chefs rule the restaurant. There is a single restaurant in the hotel. The restaurant is described by its name and amount of tables. The restaurant consists of at least one table. Every table has its unique number and number of seats. Tables might be located inside the restaurant and outside depending on the seasons. Porter can reserve a table for a client on a given date. Clients can book outdoor tables only in summer. For indoor tables we want to

know if they are in a smoking area and for outdoor tables if they are under an umbrella.

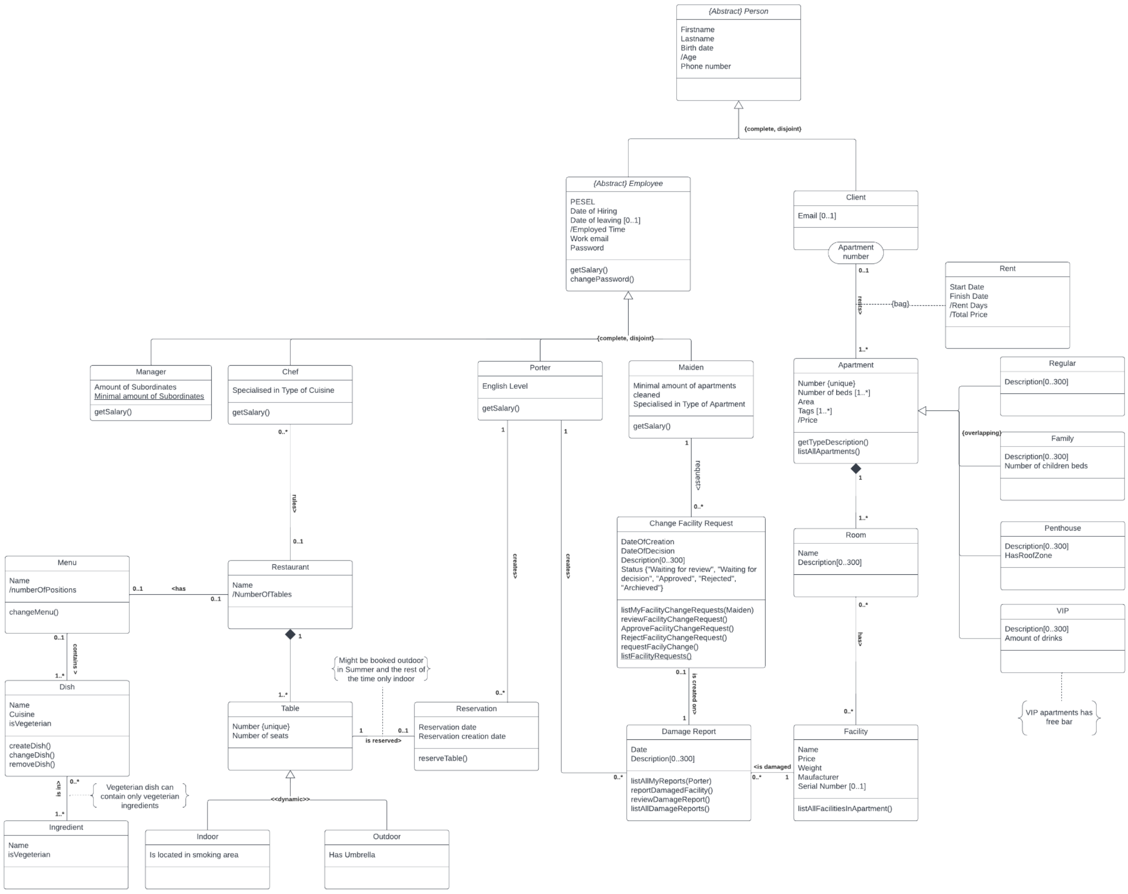
The restaurant has only one many described by the name and number of positions. Menu consists of dishes. Dish has its name, cuisine and we want to know if the dish is vegetarian. The system should keep the information about the name of the ingredient. Only vegetarian ingredients can be in vegetarian dishes. The shef can change the dishes in the menu, create new dishes, change or remove them.

The porter can report damaged facilities and later list all of his facility reports. Damage report described by the date and description in a maximum of 300 words. Maiden can review the damage reports and request facility change if needed. Maiden has access to the list of its own facility change requests. Facility change request is described by date of creation, description and decision date. Managers can list all facility change reports, review them and approve or reject.

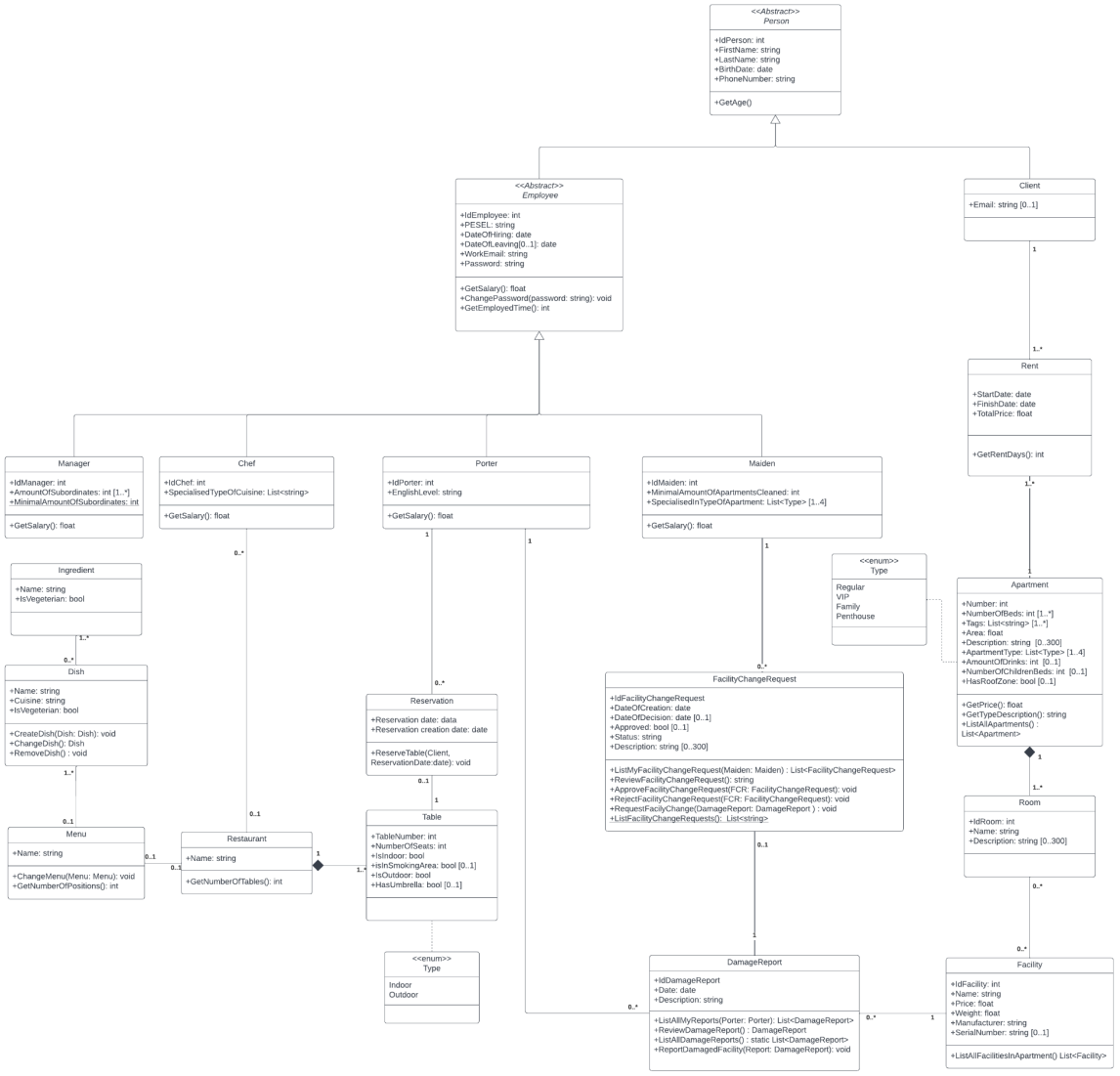
2. Use Case Diagram



3. Analytical Class Diagram



4. Design Class Diagram



5. Use Case Scenario

Name : Report damage facility
Actor : Porter

Pre condition : Actor has access to the system and is logged in to the system as a porter. At least one facility located in an apartment is stored in the system.

Basic Path:

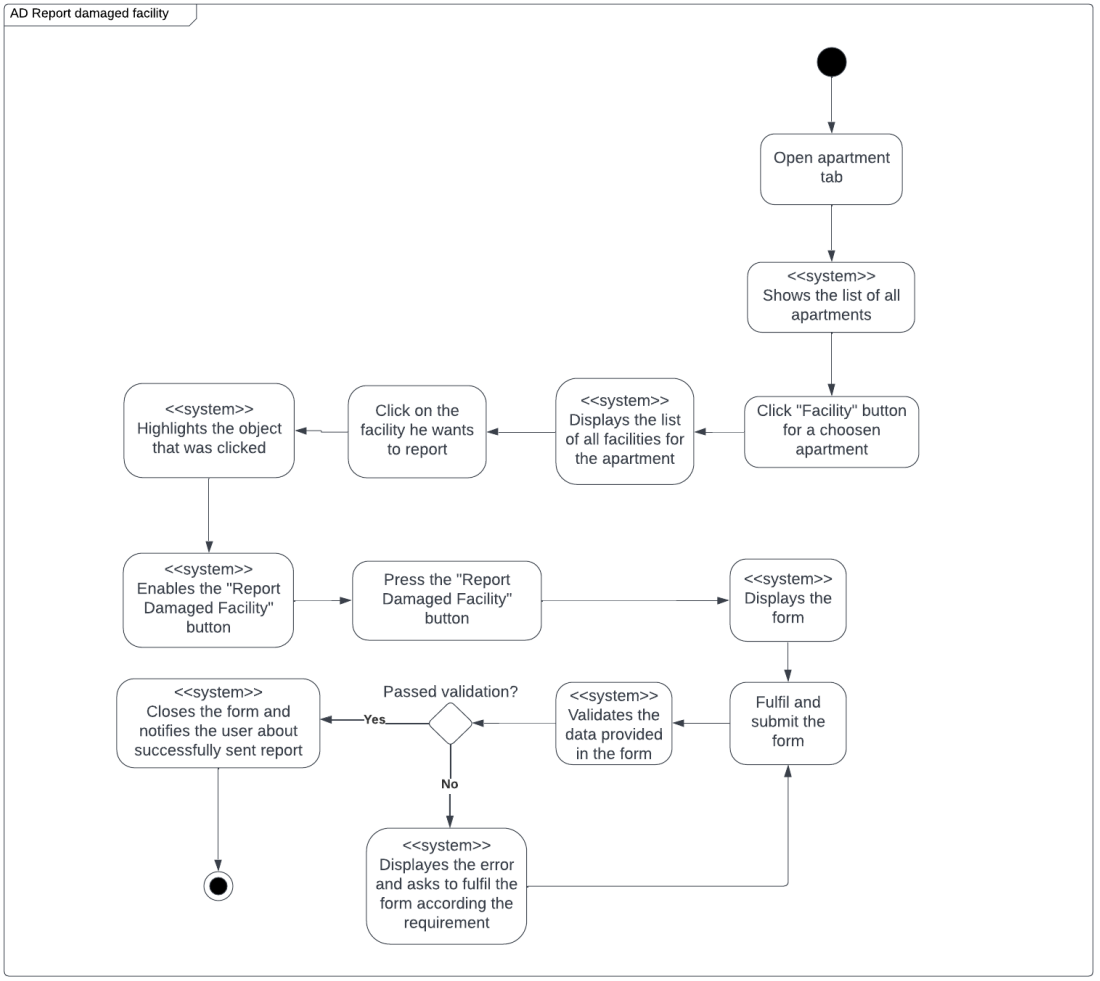
1. The actor navigates to the apartments tab.
2. The system shows the list of all apartments registered in the hotel database.
3. The actor chooses the apartment with damaged facilities.
4. The system displays the list of all facilities in this apartment.
5. The actor chooses the facility he wants to report and presses the button “Report Damaged Facility”.
6. The system shows the form for the report.
7. The actor fulfils the form and saves the report to the system by pressing the “Send report” button.
8. The system validates the data provided in the form.
9. The system shows the confirmation of a successfully sent report.

Alternative path:

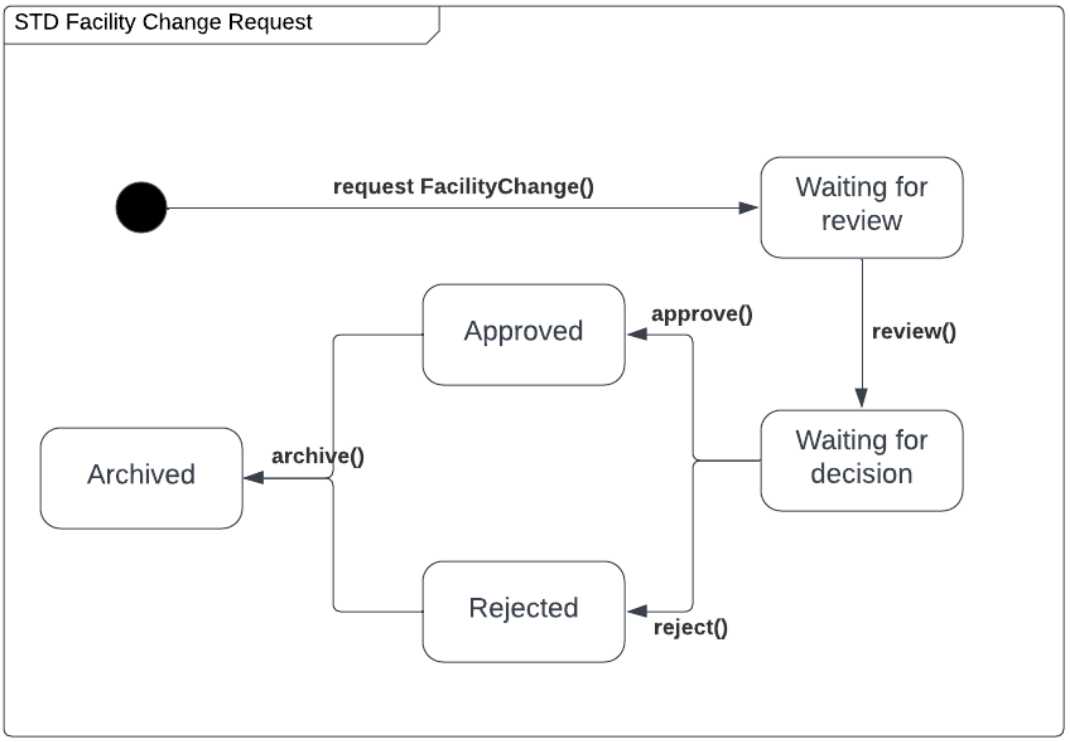
9a. The system shows error and asks to fulfil the form again.
9aa. The actor fulfils the form according to all requirements and saves the report to the system by pressing the “Send report” button.

Post condition : The system successfully saved the facility damage report in the system.

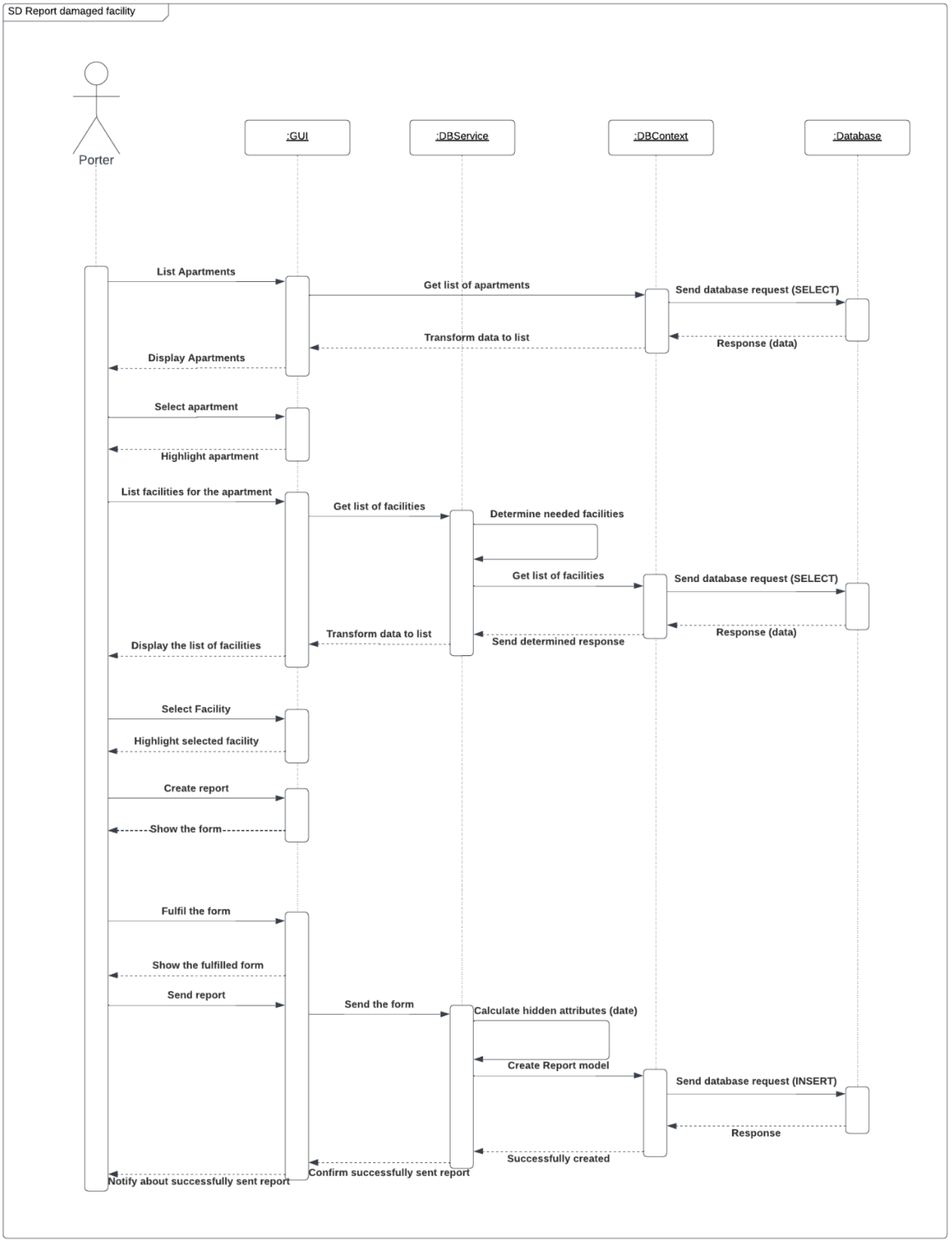
6. Activity Diagram



7. State Diagram

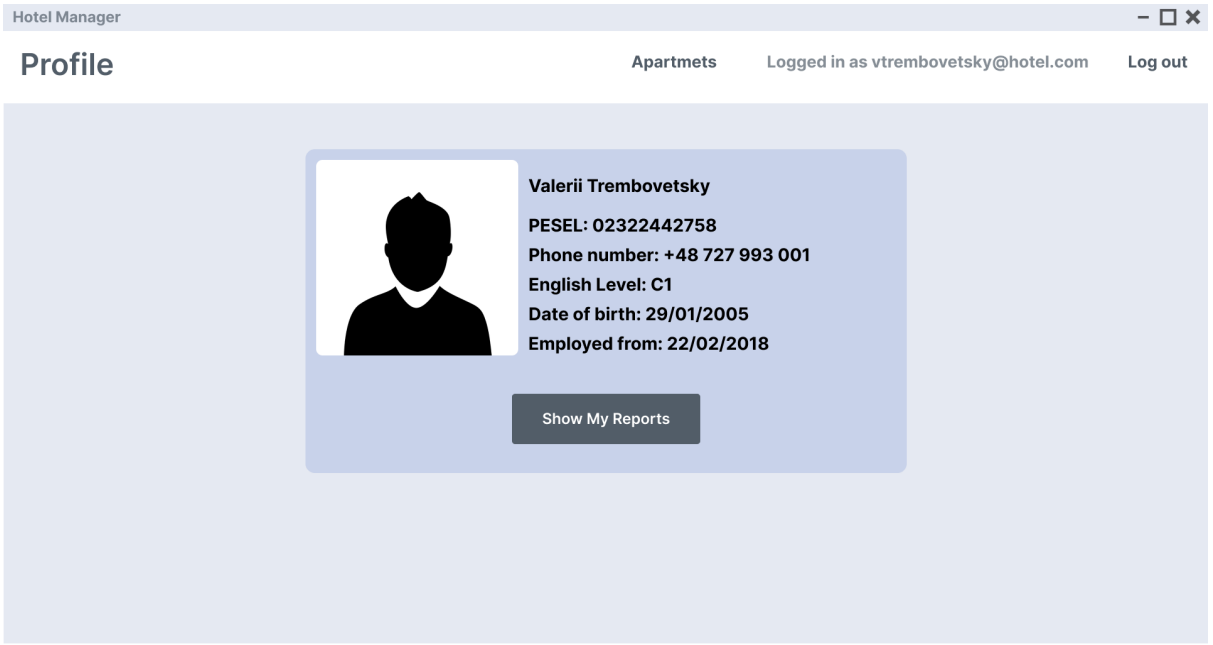


8. Sequence Diagram

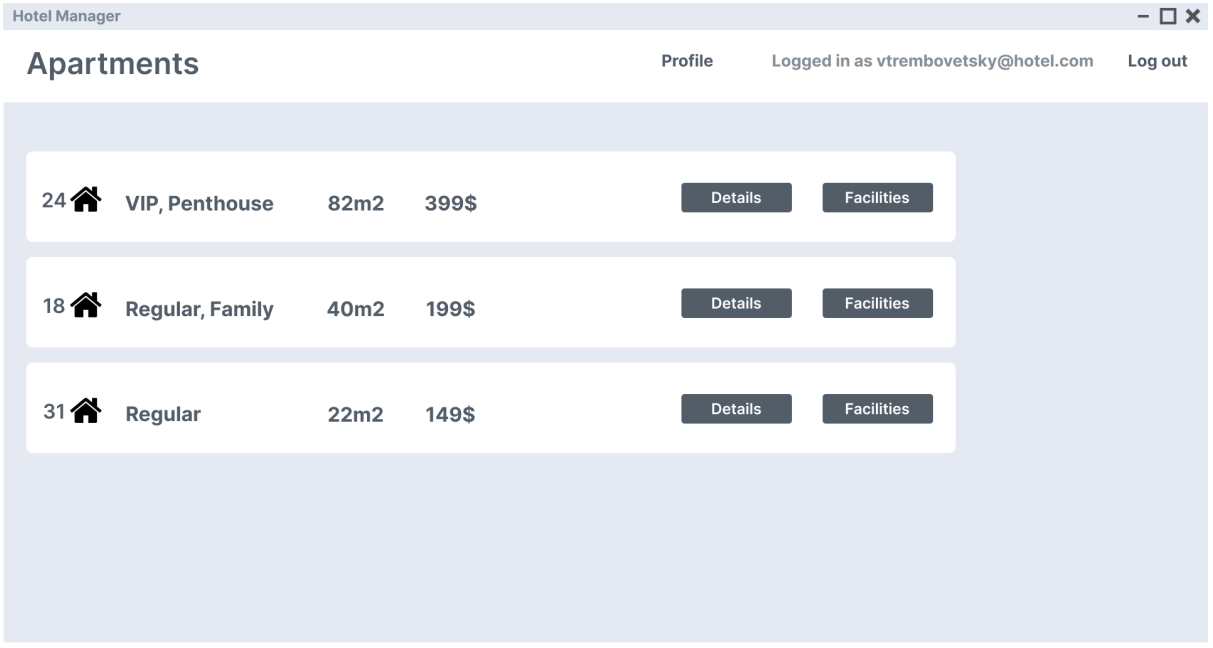


9. GUI Design

Profile tab. The place where the use case starts. Here is the porter's profile and information about him. Click the "Apartments" button to see the list of apartments.

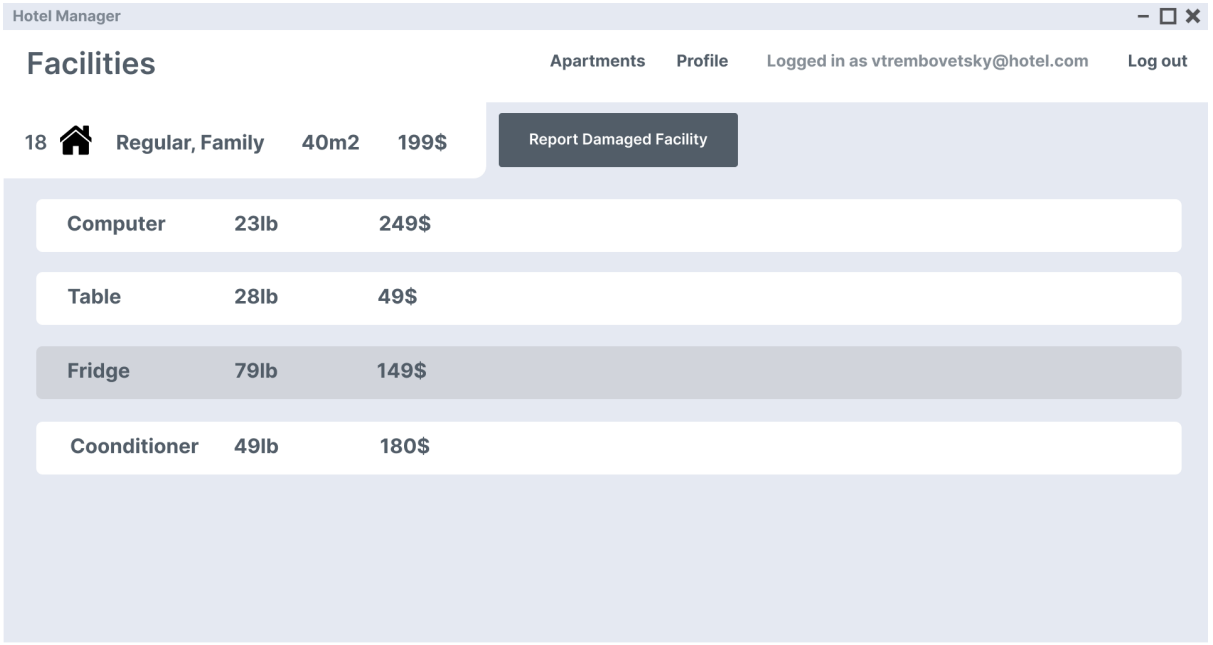


After pressing the button "Apartment" the list of apartments is displayed. Here the actor chooses facilities to display the list of facilities of the particular apartment.

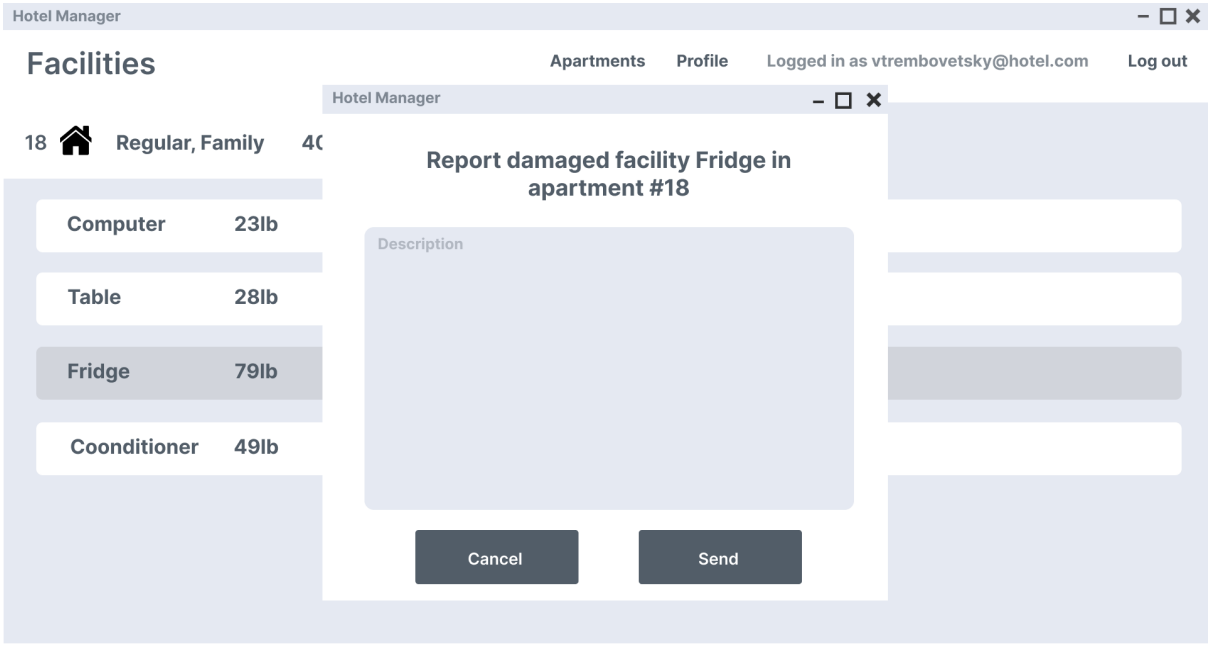


On the next screen we can see the list of facilities of the current apartment. The actor clicks on a particular facility to select it.

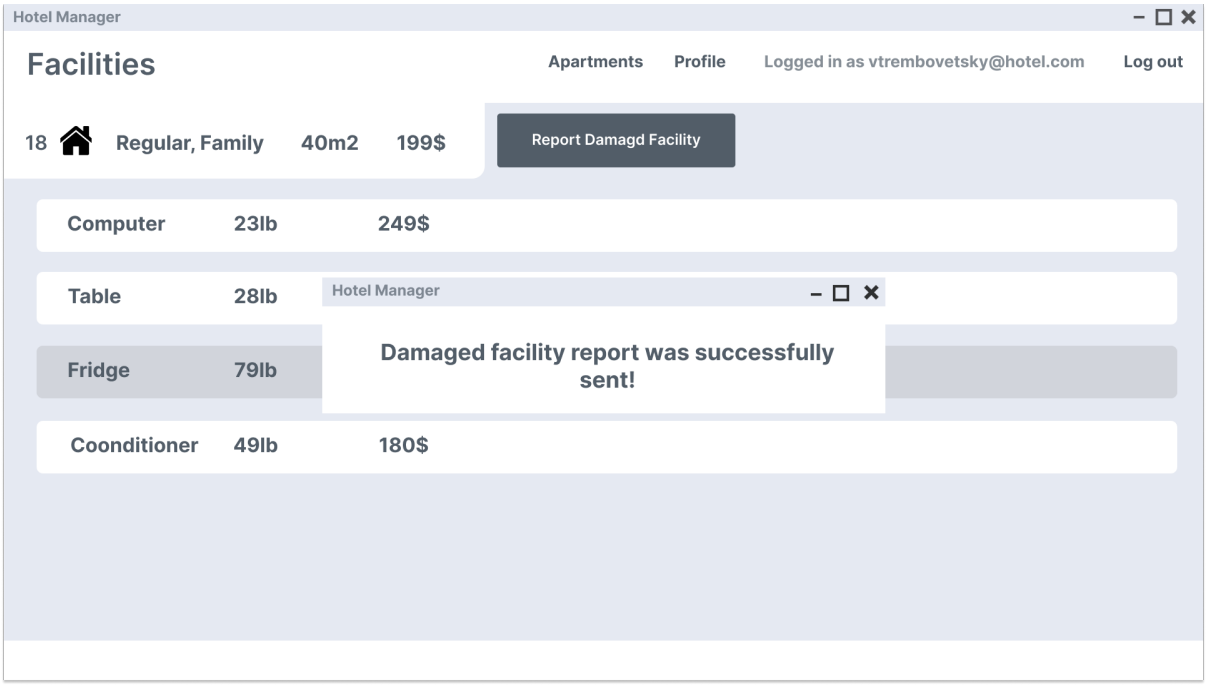
As we can see, the facility the actor clicked on changed its colour that means he successfully selected it. When the facility is selected the next step is to press the button “Report Damaged Facility”.



The new window popped up with the report form. The actor fulfils this form and presses the button “Send” to send the report to the system.



After sending the report pops up a new window notifying the actor about the successfully sent damaged facility report.



10. Design decisions and dynamic analysis

The final project will be developed in .NET Core 5 using C# 9.0. For this desktop application I am going to use the Model-View-ViewModel (MVVM) design pattern that is structured to separate program logic and user interface controls.

Apartments and their types were decided to design with flattered inheritance using enums. The dynamic inheritance from table to indoor and outdoor classes was designed with multiple optional attributes decisions. Derived attributes turned into methods.The bag class Rent is going to be implemented without any unique ID that will help to store the history of all rents in the hotel. The usual disjoint inheritance from Person > Employee and other workers was decided to implement as a table-per-type mapping pattern, where all the types are mapped to individual tables. Properties that belong solely to a base type or derived type are stored in a table that maps to that type. Tables that map to derived types alose store a foreign key that joins the derived table with the base table.

The qualified association between Client and Apartment classes was designed as association with attributes. The associations are designed to be done with foreign keys. Those foreign keys might be shadowed to make the implementation smoother and not to load the models with unnecessary fields. ICollection is designed to be used for associations with more than one instance. Moreover, there is intendence to use the Fluent API to configure whether the relationships and attributes are required or optional.

During the dynamic analysis I came to the decision of creating a new class “FacilityChangeRequest” to complete the idea of the damaged facility. It inspired me to add more functionalities for Manager and Maiden. After those modifications Maiden can review the damaged facility request and request the change. And the manager in his turn can approve or reject the request.

One of the most important influences of dynamic analysis was the reminder about alternative scenarios. In my opinion a developer should take more alternative scenarios into consideration to lower the amount of bugs in an application to improve the user experience.