

**Capstone Project Report**

**Report 4 – Software Design Document**

– Hanoi, Sep 2022 –

**Table of Contents**

[I. Record of Changes 4](#_Toc119390811)

[II. Software Design Document 5](#_Toc119390812)

[1. System Design 5](#_Toc119390813)

[1.1 System Architecture 5](#_Toc119390814)

[1.2 Package Diagram 6](#_Toc119390815)

[2. Database Design 8](#_Toc119390816)

[3. Detailed Design 11](#_Toc119390817)

[3.1 Home Page 11](#_Toc119390818)

[3.2 House Detail 17](#_Toc119390819)

[3.3 Room Detail 22](#_Toc119390820)

[3.4 Landlord – List Room 26](#_Toc119390821)

[3.5 Landlord – Create Room 32](#_Toc119390822)

[3.6 Landlord – Update Room 37](#_Toc119390823)

[3.7 Landlord – Delete Room 42](#_Toc119390824)

[3.8 Login 45](#_Toc119390825)

[3.9 Landlord – Upload House Information 48](#_Toc119390826)

[4. Database Tables 51](#_Toc119390827)

[1. Campuses 51](#_Toc119390828)

[2. UserRoles 51](#_Toc119390829)

[3. Addresses 51](#_Toc119390830)

[4. Users 52](#_Toc119390831)

[5. Houses 52](#_Toc119390832)

[6. Rooms 53](#_Toc119390833)

[7. Rates 54](#_Toc119390834)

[8. ImagesOfHouse 54](#_Toc119390835)

[9. ImagesOfRoom 54](#_Toc119390836)

[10. Districts 55](#_Toc119390837)

[11. Communes 55](#_Toc119390838)

[12. Villages 55](#_Toc119390839)

[13. RoomStatuses 55](#_Toc119390840)

[14. RoomTypes 55](#_Toc119390841)

[15. Reports 56](#_Toc119390842)

[16. RoomHistories 56](#_Toc119390843)

[17. Issues 56](#_Toc119390844)

[18. Orders 56](#_Toc119390845)

[19. OrderStatuses 57](#_Toc119390846)

# I. Record of Changes

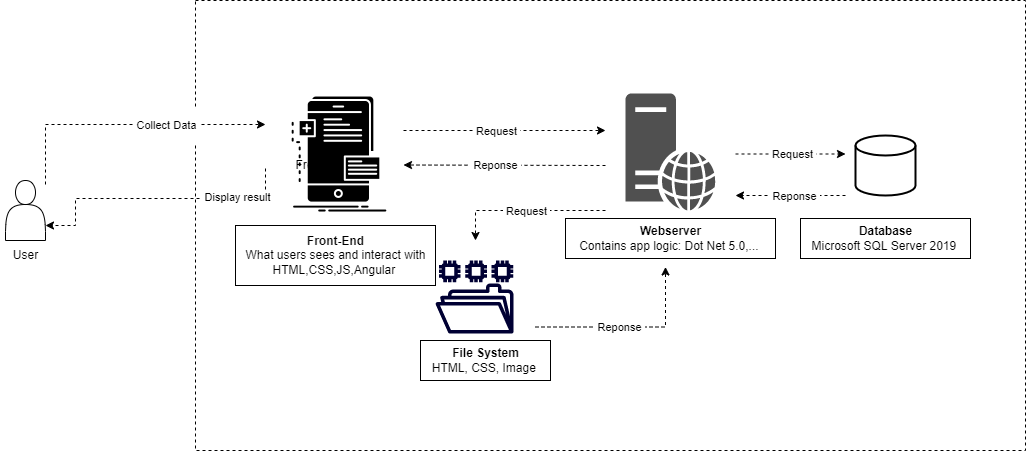
|  |  |  |  |
| --- | --- | --- | --- |
| Date | A\* M, D | In charge | Change Description |
| 08/10/2022 | A | KienNT | Database Tables |
| 09/10/2022 | M | GiangNT | Database Tables |
| 09/10/2022 | A | ThongPQ | System Architecture |
| 12/10/2022 | A | KienNT | Package Diagram, Package Description |
| 13/10/2022 | M | GiangNT | Package Diagram, Package Description |
| 17/10/2022 | M | ThongPQ | System Architecture |
| 25/10/2022 | M | GiangNT | Package Diagram, Package Description |
| 02/11/2022 | M | HuyenBN | Database Tables |
| 13/11/2022 | A | GiangNT | Detail Design: Home Page |
| 14/11/2022 | A | KienNT | Detail Design: House Detail, Room Detail |
| 14/11/2022 | A | ThongPQ | Detail Design: Landlord – Manage Rooms |
| 14/11/2022 | A | AnNT | Detail Design: Login, Landlord – Upload Info |
| 14/11/2022 | A | HuyenBN | Database Design; Database Tables |
| 14/11/2022 | M | GiangNT | All Document |

\*A - Added M - Modified D - Deleted

# II. Software Design Document

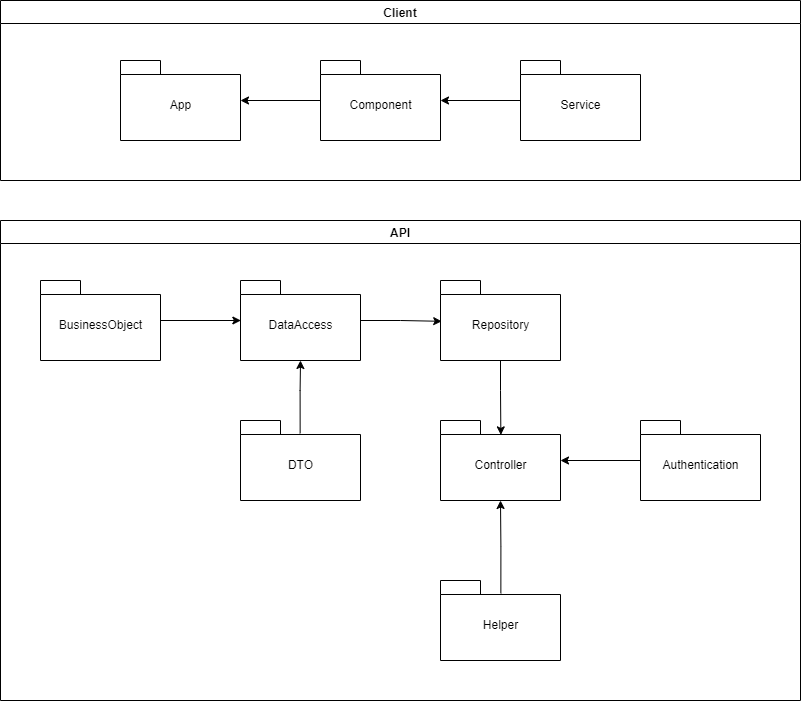
## 1. System Design

### 1.1 System Architecture



|  |  |  |
| --- | --- | --- |
| **No** | **Component** | **Description** |
| 1 | User | As a user, impact on the site |
| 2 | Front-End | What users sees and interact with HTML, CSS, JS, Angular |
| 3 | Webserver | Contains app logic: Dot Net 5.0, ... |
| 4 | File System | Contains HTML, CSS, Image in client. |
| 5 | Database | Microsoft SQL Server 2019, this contains all data of website. |

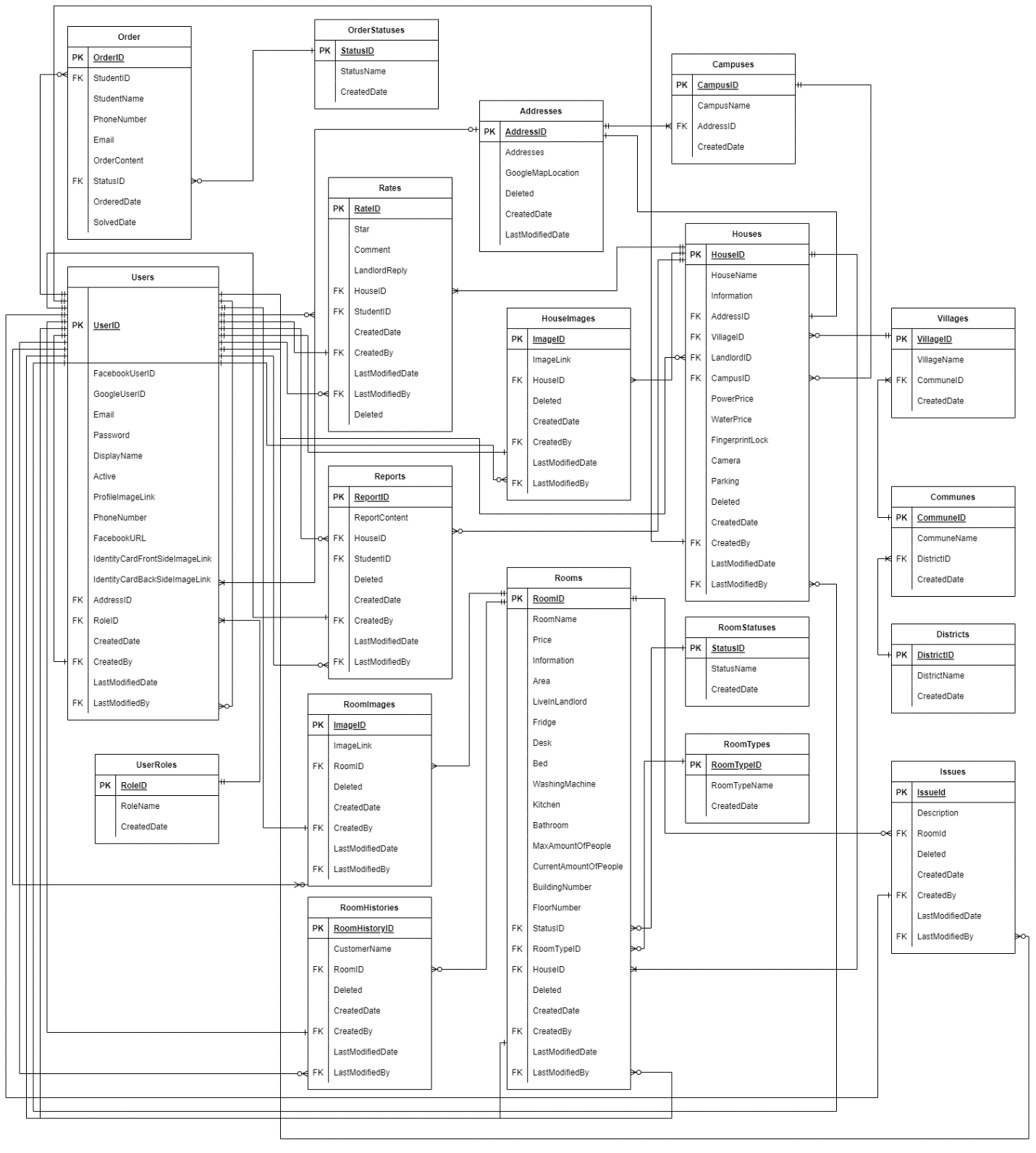
### 1.2 Package Diagram



***Package Descriptions***

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| 01 | BusinessObject | Define the logic of data extracted from Database |
| 02 | DataAccess | Provide the connection to the Database; Extract data for the system |
| 03 | Repository | Create an abstraction layer between the data access layer and the business logic layer of an application |
| 04 | DTO | Model with predefined validation in place for HTTP responses and requests |
| 05 | Authentication | Provide JWT Authentication methods for using in Controller |
| 06 | Helper | Provide helping methods for using in Controller |
| 05 | Controller | The controller takes the result of the model's processing (if any) and returns either the proper view and its associated view data or the result of the API call |
| 06 | App | Root component |
| 07 | Component | Components are the main building block for Angular applications. Each component consists of: An HTML template that declares what renders on the page. A TypeScript class that defines behavior |
| 08 | Service | Define code or functionalities that are then accessible and reusable in many other components in Angular project |

## 2. Database Design



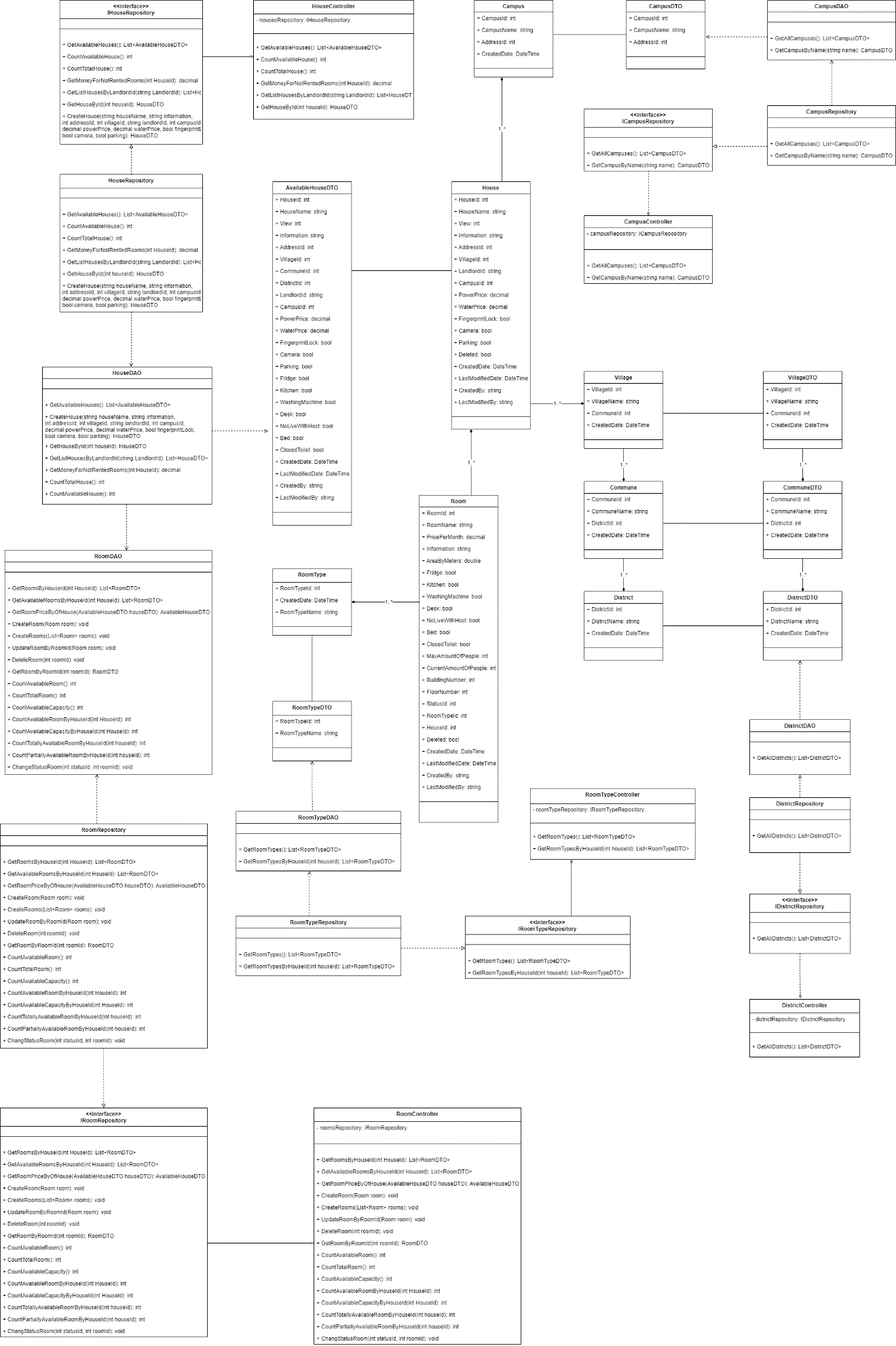
***Table Descriptions***

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| *01* | *Addresses* | *Store the information of addresses for Users (Landlords) and Houses*   * *Primary key: AddressId* |
| *02* | *Campuses* | *Store the information of all campus for users*   * *Primary key: CampusId* * *Foreign keys: AddressId* |
| *03* | *Communes* | *Store the information of all communes (for identifying house address)*   * *Primary key: CommuneId* * *Foreign keys: DistrictId* |
| *04* | *Districts* | *Store the information of all districts (for identifying house address)*   * *Primary key: DistrictId* * *Foreign keys: CampusId* |
| *05* | *Houses* | *Store the information of all houses*   * *Primary key: HouseId* * *Foreign keys: AddressId, LandlordId, VillageId, CampusId, CreatedBy, LastModifiedBy* |
| *06* | *ImagesOfHouse* | *Store the information of all pictures of house*   * *Primary key: ImageId* * *Foreign keys: HouseId, CreatedBy, LastModifiedBy* |
| *07* | *ImagesOfRoom* | *Store the information of all pictures of room*   * *Primary key: ImageId* * *Foreign keys: RoomId, CreatedBy, LastModifiedBy* |
| *08* | *Issues* | *Store the information of all records of Issues usage for Landlords to manage if they want to*   * *Primary key: IssueId* * *Foreign keys: RoomId, CreatedBy, LastModifiedBy* |
| *09* | *Order* | *Store the information of all records of all Orders*   * *Primary key: OrderId* * *Foreign keys: StudentId, StatusId* |
| *10* | *OrderStatuses* | *Store the information of all status of an Order*   * *Primary key: StatusId* |
| *11* | *Rates* | *Store the information of all rates and comments of users*   * *Primary key: RateId* * *Foreign keys: HouseId, StudentId, CreatedBy, LastModifiedBy* |
| *12* | *Reports* | *Store the information of all Reports of Students to Houses*   * *Primary key: ReportId* * *Foreign keys: HouseId, StudentId, CreatedBy, LastModifiedBy* |
| *13* | *RoomHistories* | *Store the information of all records of histories of a room*   * *Primary key: RoomHistoryId* * *Foreign keys: RoomId, CreatedBy, LastModifiedBy* |
| *14* | *Rooms* | *Store the information of all rooms of houses*   * *Primary key: RoomId* * *Foreign keys: StatusId, RoomTypeId, HouseId, CreatedBy, LastModifiedBy* |
| *15* | *RoomStatuses* | *Store the information of all status of a Room*   * *Primary key: StatusId* |
| *16* | *RoomTypes* | *Store the information of all types of a Room*   * *Primary key: RoomTypeId* |
| *17* | *UserRoles* | *Store the information of all roles of users*   * *Primary key: RoleId* |
| *18* | *Users* | *Store the information of all users*   * *Primary key: UserId* * *Foreign keys: RoleId, AddressId, CreatedBy, LastModifiedBy* |
| *19* | *Villages* | *Store the information of all villages (for identifying house address)*   * *Primary key: VillageId* * *Foreign keys: CommuneId* |

## 3. Detailed Design

### 3.1 Home Page

#### 3.1.1 Class Diagram



#### 3.1.2 Class Specifications

##### HouseDAO Class

This class performs action to data (CRUD) in House Table in Database

**Class Methods**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | GetAvailableHouses() | Return list of all available Houses in the system (Houses that have available Rooms) with additional information to support filtering |
| 02 | CreateHouse() | Add a new House into the Database |
| 03 | GetHouseById() | Find detail information of a House by its Id |
| 04 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |
| 05 | GetMoneyForNotRentedRooms() | Get total amount of money of rooms that has not been rented (of 1 House) |
| 06 | CountTotalHouse() | Count total number of Houses in the system |
| 07 | CountAvailableHouse() | Count number of available Houses in the system |

##### HouseController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | GetAvailableHouses() | Return list of all available Houses in the system (Houses that have available Rooms) with additional information to support filtering |
| 02 | CreateHouse() | Add a new House into the Database |
| 03 | GetHouseById() | Find detail information of a House by its Id |
| 04 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |
| 05 | GetMoneyForNotRentedRooms() | Get total amount of money of rooms that has not been rented (of 1 House) |
| 06 | CountTotalHouse() | Count total number of Houses in the system |
| 07 | CountAvailableHouse() | Count number of available Houses in the system |

##### CampusDAO Class

This class performs action to data (CRUD) in Campus Table in Database

**Class Methods**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | GetAllCampuses() | Return list of all Campuses in the Database |
| 02 | GetCampusByName() | Find a Campus by its Name |

##### CampusController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | GetAllCampuses() | Return list of all Campuses in the Database |
| 02 | GetCampusByName() | Find a Campus by its Name |

##### DistrictDAO Class

This class performs action to data (CRUD) in District Table in Database

**Class Methods**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | GetAllDistricts() | Return list of all Districts in the Database |

##### DistrictController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | GetAllDistricts() | Return list of all Districts in the Database |

##### RoomTypeDAO Class

This class performs action to data (CRUD) in RoomType Table in Database

**Class Methods**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | GetRoomTypes() | Return list of all RoomTypes in the Database |
| 02 | GetRoomTypesByHouseId() | Get all roomTypes that this house has |

##### RoomTypeController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | GetRoomTypes() | Return list of all RoomTypes in the Database |
| 02 | GetRoomTypesByHouseId() | Get all roomTypes that this house has |

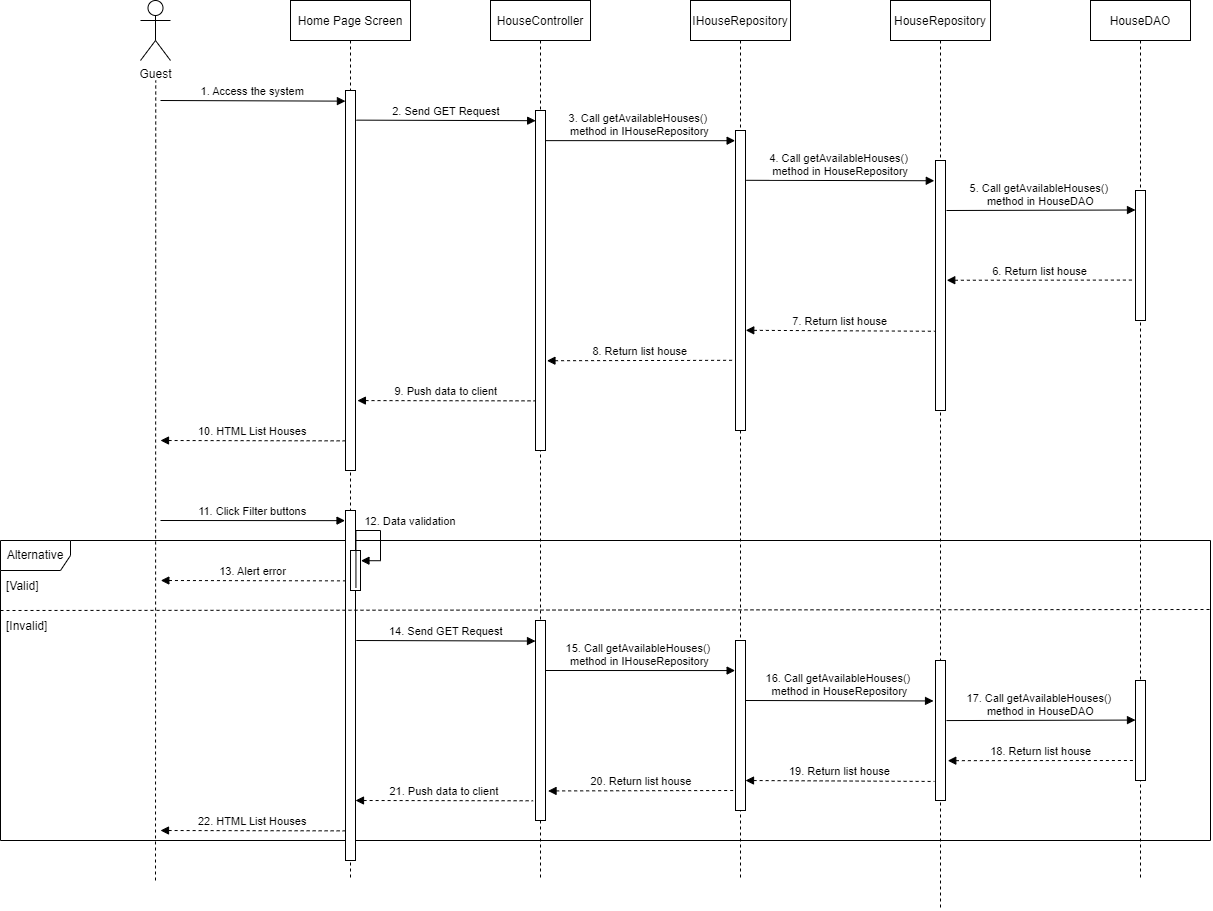
##### RoomDAO Class

This class performs action to data (CRUD) in Room Table in Database

**Class Methods**

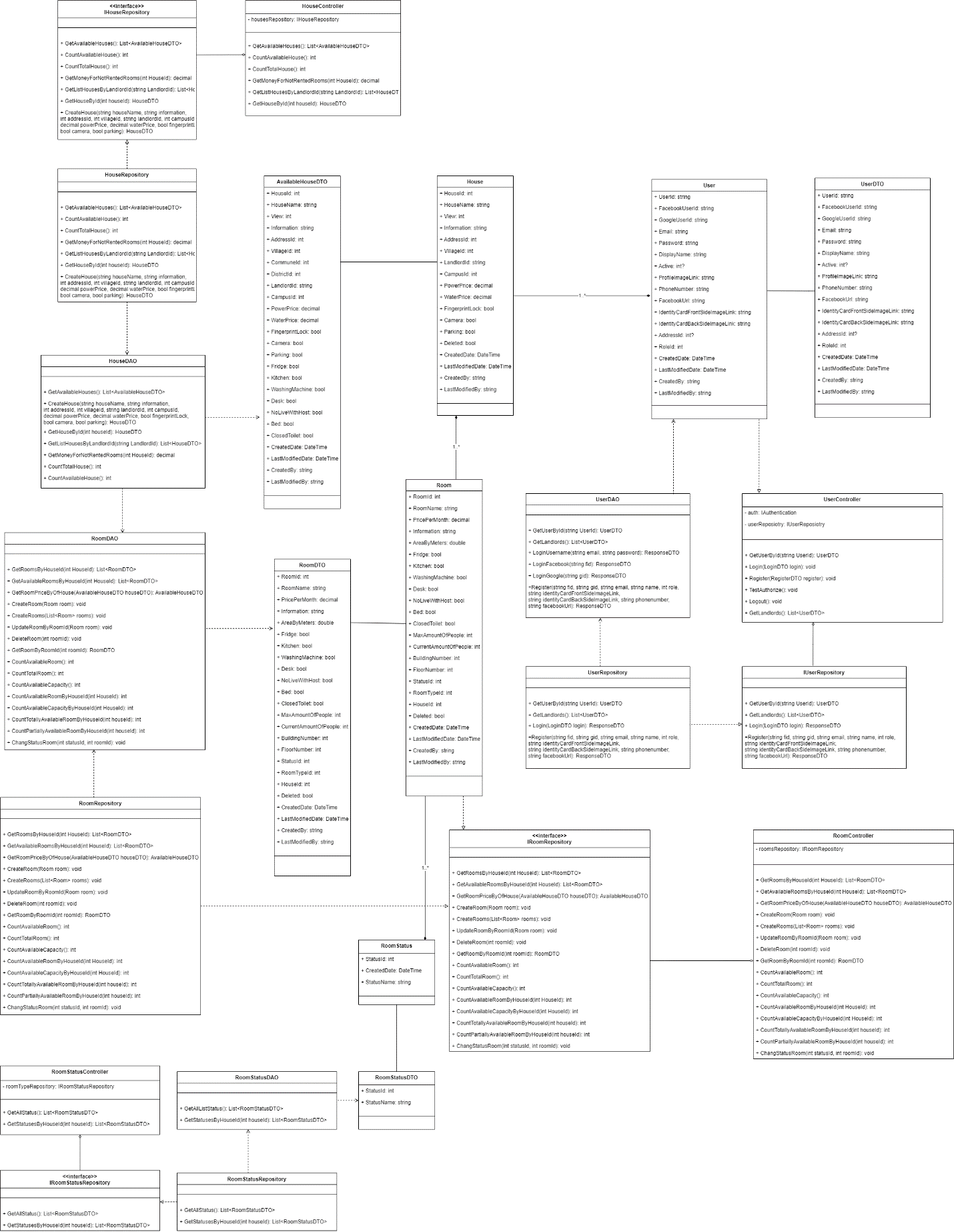
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price for of 1 House (used in HouseDAO) |
| 04 | CreateRoom() | Add a new Room to the Database |
| 05 | CreateRooms() | Add a list of Rooms of 1 House to the Database |
| 06 | UpdateRoomByRoomId() | Update detail information of 1 Room |
| 07 | DeleteRoom() | Delete a Room from a system without removing its data from the Database |
| 08 | GetRoomByRoomId() | Find a Room by its Id |
| 09 | CountAvailableRoom() | Count the sum of all available Rooms (partially or totally) of all Houses in the system |
| 10 | CountTotalRoom() | Count the sum of all Rooms (available or not) of all Houses in the system |
| 11 | CountAvailableCapacity() | Count the sum of Capacity of all Rooms of all Houses in the system |
| 12 | CountAvailableRoomByHouseId() | Count the sum of all available Rooms (partially or totally) of 1 House |
| 13 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 14 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 15 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |
| 16 | ChangeStatusRoom() | Change Status (Available or Occupied) of 1 Room |

#### 3.1.3 Sequence Diagram



### 3.2 House Detail

#### 3.2.1 Class Diagram



#### 3.2.2 Class Specifications

##### HouseDAO Class

This class performs action to data (CRUD) in House Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAvailableHouses() | Return list of all available Houses in the system (Houses that have available Rooms) with additional information to support filtering |
| 02 | CreateHouse() | Add a new House into the Database |
| 03 | GetHouseById() | Find detail information of a House by its Id |
| 04 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |
| 05 | GetMoneyForNotRentedRooms() | Get total amount of money of rooms that has not been rented (of 1 House) |
| 06 | CountTotalHouse() | Count total number of Houses in the system |
| 07 | CountAvailableHouse() | Count number of available Houses in the system |

##### HouseController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAvailableHouses() | Return list of all available Houses in the system (Houses that have available Rooms) with additional information to support filtering |
| 02 | CreateHouse() | Add a new House into the Database |
| 03 | GetHouseById() | Find detail information of a House by its Id |
| 04 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |
| 05 | GetMoneyForNotRentedRooms() | Get total amount of money of rooms that has not been rented (of 1 House) |
| 06 | CountTotalHouse() | Count total number of Houses in the system |
| 07 | CountAvailableHouse() | Count number of available Houses in the system |

##### RoomStatusDAO Class

This class performs action to data (CRUD) in RoomType Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAllListStatus() | Return list of all RoomStatus in the Database |
| 02 | GetStatusesByHouseId() | Get all RoomStatus that this house has |

##### RoomStatusController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAllStatus() | Return list of all RoomStatus in the Database |
| 02 | GetStatusesByHouseId() | Get all RoomStatus that this house has |

##### RoomDAO Class

This class performs action to data (CRUD) in Room Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price for of 1 House (used in HouseDAO) |
| 04 | CreateRoom() | Add a new Room to the Database |
| 05 | CreateRooms() | Add a list of Rooms of 1 House to the Database |
| 06 | UpdateRoomByRoomId() | Update detail information of 1 Room |
| 07 | DeleteRoom() | Delete a Room from a system without removing its data from the Database |
| 08 | GetRoomByRoomId() | Find a Room by its Id |
| 09 | CountAvailableRoom() | Count the sum of all available Rooms (partially or totally) of all Houses in the system |
| 10 | CountTotalRoom() | Count the sum of all Rooms (available or not) of all Houses in the system |
| 11 | CountAvailableCapacity() | Count the sum of Capacity of all Rooms of all Houses in the system |
| 12 | CountAvailableRoomByHouseId() | Count the sum of all available Rooms (partially or totally) of 1 House |
| 13 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 14 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 15 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |
| 16 | ChangeStatusRoom() | Change Status (Available or Occupied) of 1 Room |

##### RoomController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price for of 1 House (used in HouseDAO) |
| 04 | CreateRoom() | Add a new Room to the Database |
| 05 | CreateRooms() | Add a list of Rooms of 1 House to the Database |
| 06 | UpdateRoomByRoomId() | Update detail information of 1 Room |
| 07 | DeleteRoom() | Delete a Room from a system without removing its data from the Database |
| 08 | GetRoomByRoomId() | Find a Room by its Id |
| 09 | CountAvailableRoom() | Count the sum of all available Rooms (partially or totally) of all Houses in the system |
| 10 | CountTotalRoom() | Count the sum of all Rooms (available or not) of all Houses in the system |
| 11 | CountAvailableCapacity() | Count the sum of Capacity of all Rooms of all Houses in the system |
| 12 | CountAvailableRoomByHouseId() | Count the sum of all available Rooms (partially or totally) of 1 House |
| 13 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 14 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 15 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |
| 16 | ChangeStatusRoom() | Change Status (Available or Occupied) of 1 Room |

##### UserDAO Class

This class performs action to data (CRUD) in User Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetUserById () | Find detail information of User by its Id |
| 02 | GetLandlords() | Get list of Landlords in the database |
| 03 | LoginUsername() | User login in the system |
| 04 | LoginFacebook() | User login with facebook in the system |
| 05 | LoginGoogle () | User login with google in the system |
| 06 | Register() | User create an account |

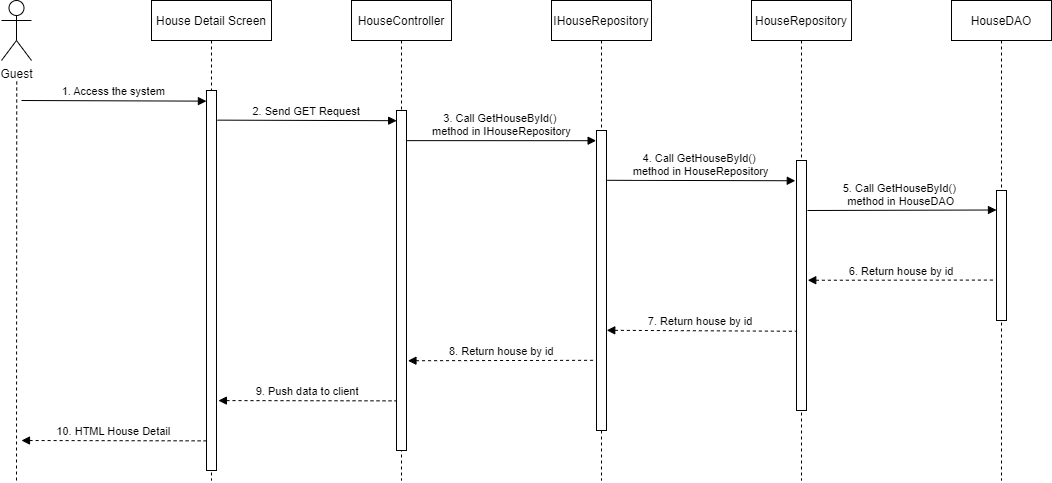
##### UserController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

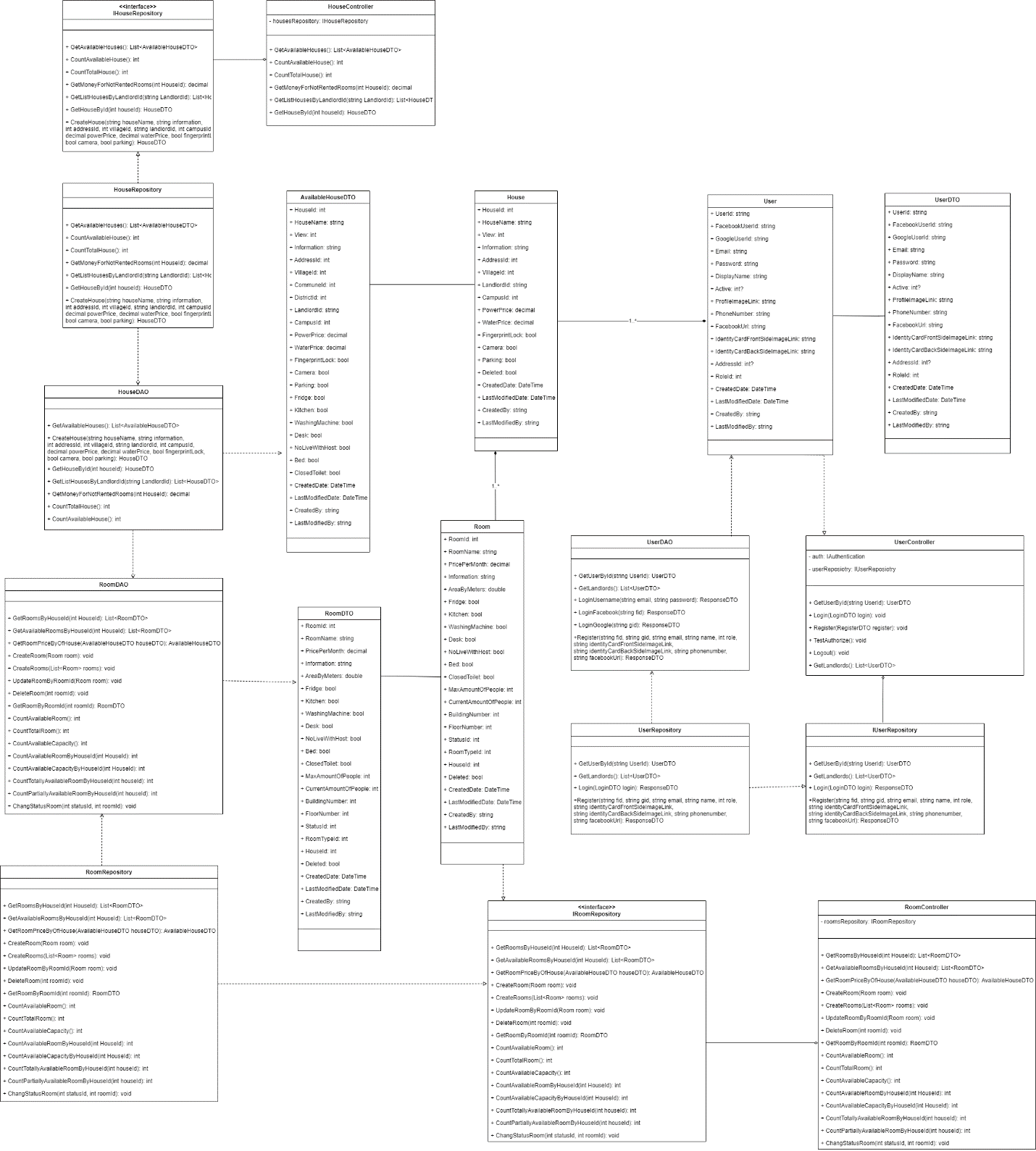
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetUserById () | Find detail information of User by its Id |
| 02 | GetLandlords() | Get list of Landlords in the database |
| 03 | LoginUsername() | User login in the system |
| 04 | LoginFacebook() | User login with facebook in the system |
| 05 | LoginGoogle () | User login with google in the system |
| 06 | Register() | User create an account |

#### 3.2.3 Sequence Diagram



### 3.3 Room Detail

#### 3.3.1 Class Diagram



#### 3.3.2 Class Specifications

##### HouseDAO Class

This class performs action to data (CRUD) in House Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAvailableHouses() | Return list of all available Houses in the system (Houses that have available Rooms) with additional information to support filtering |
| 02 | CreateHouse() | Add a new House into the Database |
| 03 | GetHouseById() | Find detail information of a House by its Id |
| 04 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |
| 05 | GetMoneyForNotRentedRooms() | Get total amount of money of rooms that has not been rented (of 1 House) |
| 06 | CountTotalHouse() | Count total number of Houses in the system |
| 07 | CountAvailableHouse() | Count number of available Houses in the system |

##### HouseController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAvailableHouses() | Return list of all available Houses in the system (Houses that have available Rooms) with additional information to support filtering |
| 02 | CreateHouse() | Add a new House into the Database |
| 03 | GetHouseById() | Find detail information of a House by its Id |
| 04 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |
| 05 | GetMoneyForNotRentedRooms() | Get total amount of money of rooms that has not been rented (of 1 House) |
| 06 | CountTotalHouse() | Count total number of Houses in the system |
| 07 | CountAvailableHouse() | Count number of available Houses in the system |

##### RoomDAO Class

This class performs action to data (CRUD) in Room Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price for of 1 House (used in HouseDAO) |
| 04 | CreateRoom() | Add a new Room to the Database |
| 05 | CreateRooms() | Add a list of Rooms of 1 House to the Database |
| 06 | UpdateRoomByRoomId() | Update detail information of 1 Room |
| 07 | DeleteRoom() | Delete a Room from a system without removing its data from the Database |
| 08 | GetRoomByRoomId() | Find a Room by its Id |
| 09 | CountAvailableRoom() | Count the sum of all available Rooms (partially or totally) of all Houses in the system |
| 10 | CountTotalRoom() | Count the sum of all Rooms (available or not) of all Houses in the system |
| 11 | CountAvailableCapacity() | Count the sum of Capacity of all Rooms of all Houses in the system |
| 12 | CountAvailableRoomByHouseId() | Count the sum of all available Rooms (partially or totally) of 1 House |
| 13 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 14 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 15 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |
| 16 | ChangeStatusRoom() | Change Status (Available or Occupied) of 1 Room |

##### RoomController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomsByHouseId() | Find detail information of list Rooms by house Id |
| 02 | GetAvailableRoomsByHouseId () | Find detail information of list available Rooms by house Id |
| 03 | GetRoomsByRoomId () | Find detail information of list Room by its Id |
| 04 | CreateRoom () | Add a new Room into the Database |
| 05 | UpdateRoomByRoomId () | Update a Room into the Database |
| 06 | DeleteRoom () | Delete a Room |
| 07 | CountAvailableRoom () | Count number of available Rooms in the system |
| 08 | CountTotalRoom() | Count number of total Rooms in the system |
| 09 | CountAvailableCapacity() | Count number of capacity of available Rooms in the system |
| 10 | CountTotalCapacity() | Count number of capacity of Rooms in the system |
| 11 | CountTotallyAvailableCapacity() | Count number of total capacity of available Rooms in the system |
| 12 | CountTotallyAvailableRoomByHouseId() | Count number of total capacity of available Rooms in the system by house id |
| 13 | CountPartiallyAvailableRoomByHouseId() | Count number of total partially of available Rooms in the system by house id |
| 14 | CountAvailableCapacityByHouseId() | Count number of total available capacity of Rooms in the system by house id |
| 15 | ChangeStatusRoom() | Update a Room status into the Database |

##### UserDAO Class

This class performs action to data (CRUD) in User Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetUserById () | Find detail information of User by its Id |
| 02 | GetLandlords() | Get list of Landlords in the database |
| 03 | LoginUsername() | User login in the system |
| 04 | LoginFacebook() | User login with facebook in the system |
| 05 | LoginGoogle () | User login with google in the system |
| 06 | Register() | User create an account |

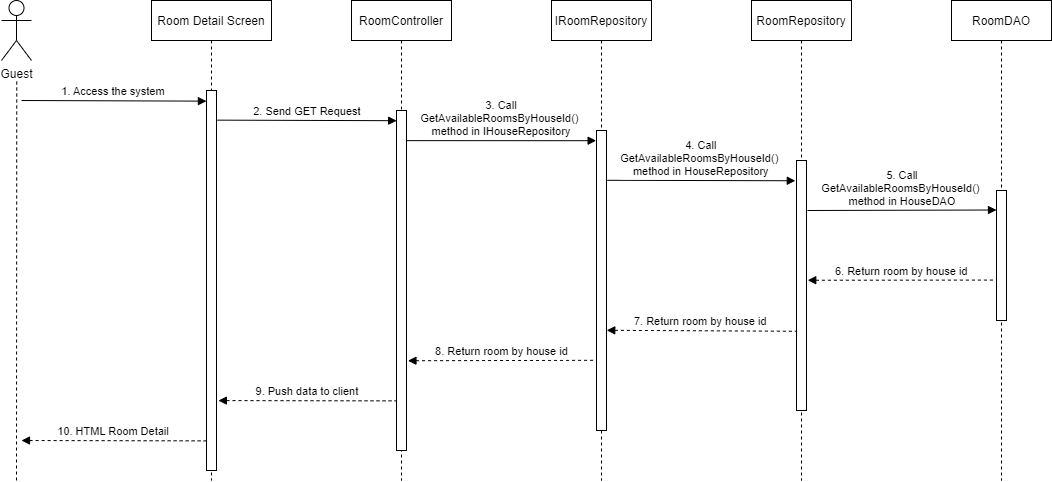
##### UserController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

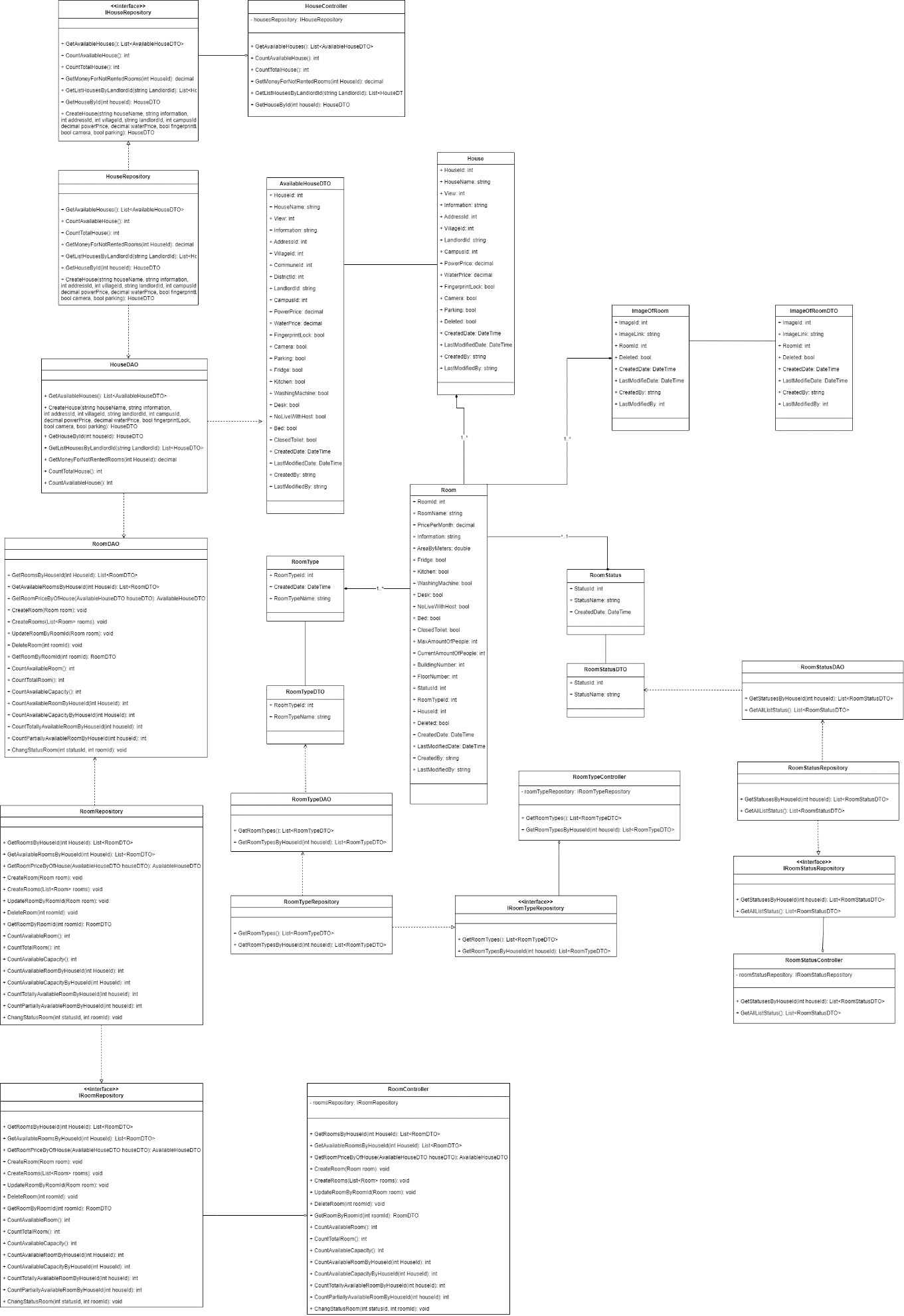
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetUserById () | Find detail information of User by its Id |
| 02 | GetLandlords() | Get list of Landlords in the database |
| 03 | LoginUsername() | User login in the system |
| 04 | LoginFacebook() | User login with facebook in the system |
| 05 | LoginGoogle () | User login with google in the system |
| 06 | Register() | User create an account |

#### 3.3.3 Sequence Diagram



### 3.4 Landlord – List Room

#### 3.4.1 Class Diagram



#### 3.4.2 Class Specifications

##### HouseDAO Class

This class performs action to data (CRUD) in House Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAvailableHouses() | Return list of all available Houses in the system (Houses that have available Rooms) with additional information to support filtering |
| 02 | CreateHouse() | Add a new House into the Database |
| 03 | GetHouseById() | Find detail information of a House by its Id |
| 04 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |
| 05 | GetMoneyForNotRentedRooms() | Get total amount of money of rooms that has not been rented (of 1 House) |
| 06 | CountTotalHouse() | Count total number of Houses in the system |
| 07 | CountAvailableHouse() | Count number of available Houses in the system |

##### HouseController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAvailableHouses() | Return list of all available Houses in the system (Houses that have available Rooms) with additional information to support filtering |
| 02 | CreateHouse() | Add a new House into the Database |
| 03 | GetHouseById() | Find detail information of a House by its Id |
| 04 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |
| 05 | GetMoneyForNotRentedRooms() | Get total amount of money of rooms that has not been rented (of 1 House) |
| 06 | CountTotalHouse() | Count total number of Houses in the system |
| 07 | CountAvailableHouse() | Count number of available Houses in the system |

##### RoomDAO Class

This class performs action to data (CRUD) in Room Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price for of 1 House (used in HouseDAO) |
| 04 | CreateRoom() | Add a new Room to the Database |
| 05 | CreateRooms() | Add a list of Rooms of 1 House to the Database |
| 06 | UpdateRoomByRoomId() | Update detail information of 1 Room |
| 07 | DeleteRoom() | Delete a Room from a system without removing its data from the Database |
| 08 | GetRoomByRoomId() | Find a Room by its Id |
| 09 | CountAvailableRoom() | Count the sum of all available Rooms (partially or totally) of all Houses in the system |
| 10 | CountTotalRoom() | Count the sum of all Rooms (available or not) of all Houses in the system |
| 11 | CountAvailableCapacity() | Count the sum of Capacity of all Rooms of all Houses in the system |
| 12 | CountAvailableRoomByHouseId() | Count the sum of all available Rooms (partially or totally) of 1 House |
| 13 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 14 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 15 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |
| 16 | ChangeStatusRoom() | Change Status (Available or Occupied) of 1 Room |

##### RoomController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price for of 1 House (used in HouseDAO) |
| 04 | CreateRoom() | Add a new Room to the Database |
| 05 | CreateRooms() | Add a list of Rooms of 1 House to the Database |
| 06 | UpdateRoomByRoomId() | Update detail information of 1 Room |
| 07 | DeleteRoom() | Delete a Room from a system without removing its data from the Database |
| 08 | GetRoomByRoomId() | Find a Room by its Id |
| 09 | CountAvailableRoom() | Count the sum of all available Rooms (partially or totally) of all Houses in the system |
| 10 | CountTotalRoom() | Count the sum of all Rooms (available or not) of all Houses in the system |
| 11 | CountAvailableCapacity() | Count the sum of Capacity of all Rooms of all Houses in the system |
| 12 | CountAvailableRoomByHouseId() | Count the sum of all available Rooms (partially or totally) of 1 House |
| 13 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 14 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 15 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |
| 16 | ChangeStatusRoom() | Change Status (Available or Occupied) of 1 Room |

##### RoomStatusDAO Class

This class performs action to data (CRUD) in RoomType Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAllListStatus() | Return list of all RoomStatus in the Database |
| 02 | GetStatusesByHouseId() | Get all RoomStatus that this house has |

##### RoomStatusController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAllStatus() | Return list of all RoomStatus in the Database |
| 02 | GetStatusesByHouseId() | Get all RoomStatus that this house has |

##### RoomTypeDAO Class

This class performs action to data (CRUD) in RoomType Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomTypes() | Return list of all RoomTypes in the Database |
| 02 | GetRoomTypesByHouseId() | Get all roomTypes that this house has |

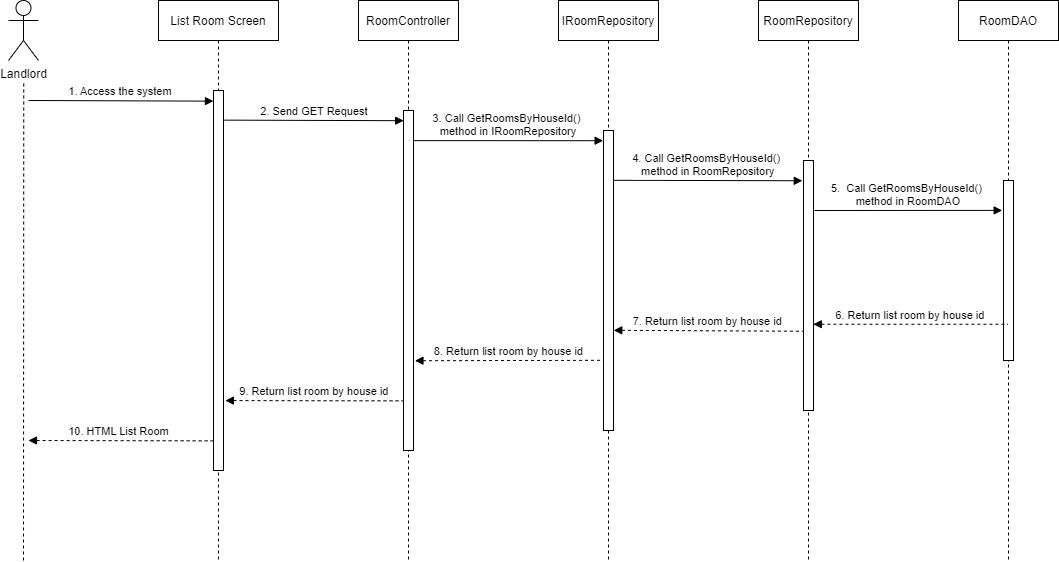
##### RoomTypeController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

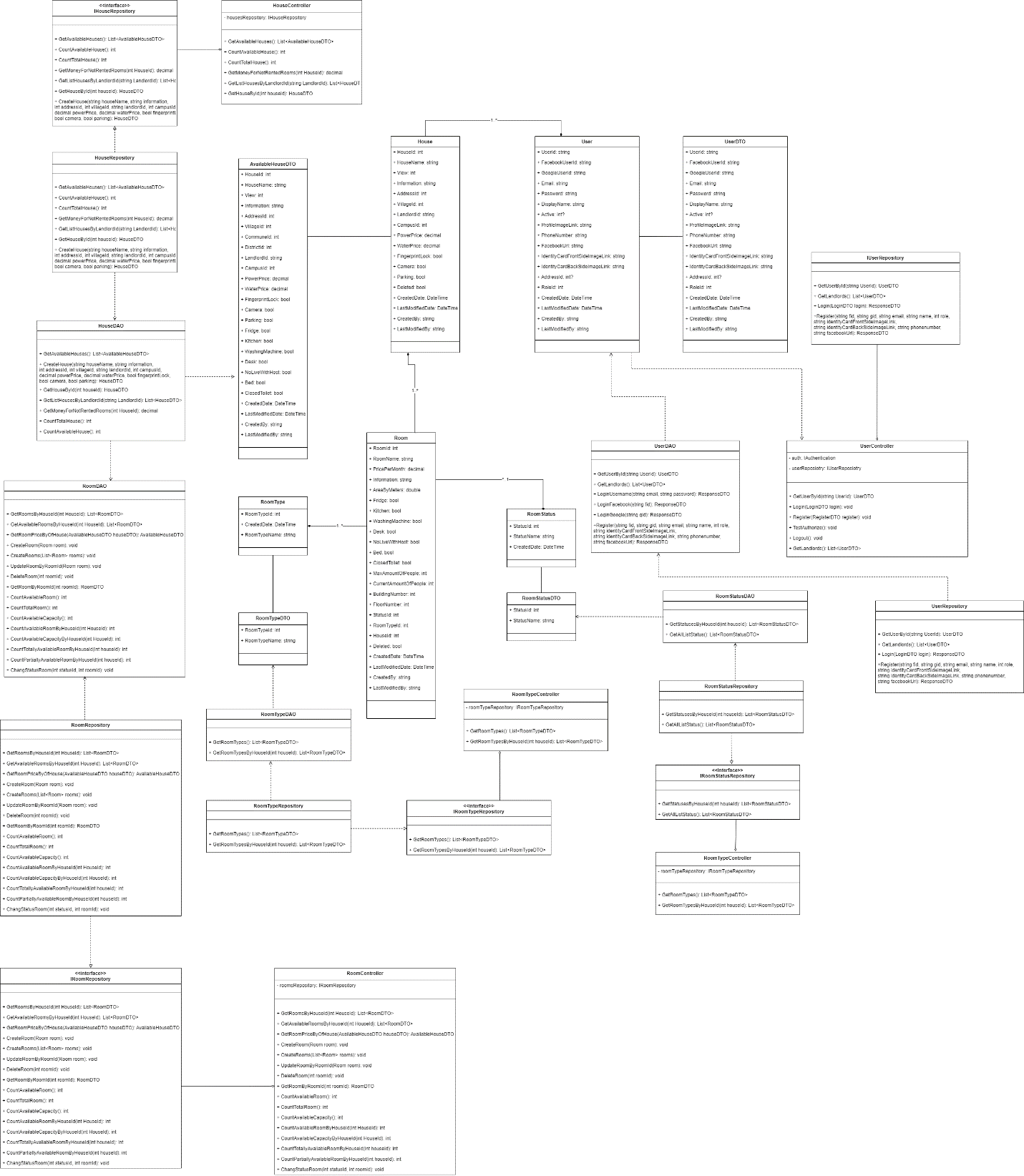
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomTypes() | Return list of all RoomTypes in the Database |
| 02 | GetRoomTypesByHouseId() | Get all roomTypes that this house has |

#### 3.4.3 Sequence Diagram



### 3.5 Landlord – Create Room

#### 3.5.1 Class Diagram



#### 3.5.2 Class Specifications

##### HouseDAO Class

This class performs action to data (CRUD) in House Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAvailableHouses() | Return list of all available Houses in the system (Houses that have available Rooms) with additional information to support filtering |
| 02 | CreateHouse() | Add a new House into the Database |
| 03 | GetHouseById() | Find detail information of a House by its Id |
| 04 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |
| 05 | GetMoneyForNotRentedRooms() | Get total amount of money of rooms that has not been rented (of 1 House) |
| 06 | CountTotalHouse() | Count total number of Houses in the system |
| 07 | CountAvailableHouse() | Count number of available Houses in the system |

##### HouseController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAvailableHouses() | Return list of all available Houses in the system (Houses that have available Rooms) with additional information to support filtering |
| 02 | CreateHouse() | Add a new House into the Database |
| 03 | GetHouseById() | Find detail information of a House by its Id |
| 04 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |
| 05 | GetMoneyForNotRentedRooms() | Get total amount of money of rooms that has not been rented (of 1 House) |
| 06 | CountTotalHouse() | Count total number of Houses in the system |
| 07 | CountAvailableHouse() | Count number of available Houses in the system |

##### RoomDAO Class

This class performs action to data (CRUD) in Room Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price for of 1 House (used in HouseDAO) |
| 04 | CreateRoom() | Add a new Room to the Database |
| 05 | CreateRooms() | Add a list of Rooms of 1 House to the Database |
| 06 | UpdateRoomByRoomId() | Update detail information of 1 Room |
| 07 | DeleteRoom() | Delete a Room from a system without removing its data from the Database |
| 08 | GetRoomByRoomId() | Find a Room by its Id |
| 09 | CountAvailableRoom() | Count the sum of all available Rooms (partially or totally) of all Houses in the system |
| 10 | CountTotalRoom() | Count the sum of all Rooms (available or not) of all Houses in the system |
| 11 | CountAvailableCapacity() | Count the sum of Capacity of all Rooms of all Houses in the system |
| 12 | CountAvailableRoomByHouseId() | Count the sum of all available Rooms (partially or totally) of 1 House |
| 13 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 14 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 15 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |
| 16 | ChangeStatusRoom() | Change Status (Available or Occupied) of 1 Room |

##### RoomController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price for of 1 House (used in HouseDAO) |
| 04 | CreateRoom() | Add a new Room to the Database |
| 05 | CreateRooms() | Add a list of Rooms of 1 House to the Database |
| 06 | UpdateRoomByRoomId() | Update detail information of 1 Room |
| 07 | DeleteRoom() | Delete a Room from a system without removing its data from the Database |
| 08 | GetRoomByRoomId() | Find a Room by its Id |
| 09 | CountAvailableRoom() | Count the sum of all available Rooms (partially or totally) of all Houses in the system |
| 10 | CountTotalRoom() | Count the sum of all Rooms (available or not) of all Houses in the system |
| 11 | CountAvailableCapacity() | Count the sum of Capacity of all Rooms of all Houses in the system |
| 12 | CountAvailableRoomByHouseId() | Count the sum of all available Rooms (partially or totally) of 1 House |
| 13 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 14 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 15 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |
| 16 | ChangeStatusRoom() | Change Status (Available or Occupied) of 1 Room |

##### UserDAO Class

This class performs action to data (CRUD) in User Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetUserById () | Find detail information of User by its Id |
| 02 | GetLandlords() | Get list of Landlords in the database |
| 03 | LoginUsername() | User login in the system |
| 04 | LoginFacebook() | User login with facebook in the system |
| 05 | LoginGoogle () | User login with google in the system |
| 06 | Register() | User create an account |

##### UserController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetUserById () | Find detail information of User by its Id |
| 02 | GetLandlords() | Get list of Landlords in the database |
| 03 | LoginUsername() | User login in the system |
| 04 | LoginFacebook() | User login with facebook in the system |
| 05 | LoginGoogle () | User login with google in the system |
| 06 | Register() | User create an account |

##### RoomStatusDAO Class

This class performs action to data (CRUD) in RoomType Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAllListStatus() | Return list of all RoomStatus in the Database |
| 02 | GetStatusesByHouseId() | Get all RoomStatus that this house has |

##### RoomStatusController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAllStatus() | Return list of all RoomStatus in the Database |
| 02 | GetStatusesByHouseId() | Get all RoomStatus that this house has |

##### RoomTypeDAO Class

This class performs action to data (CRUD) in RoomType Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomTypes() | Return list of all RoomTypes in the Database |
| 02 | GetRoomTypesByHouseId() | Get all roomTypes that this house has |

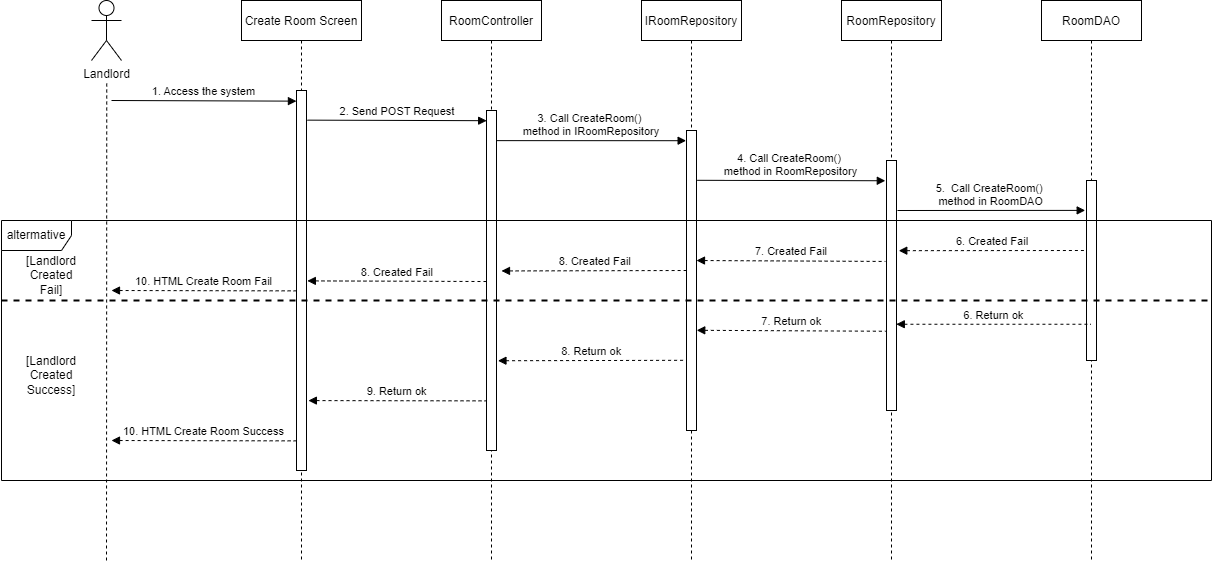
##### RoomTypeController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

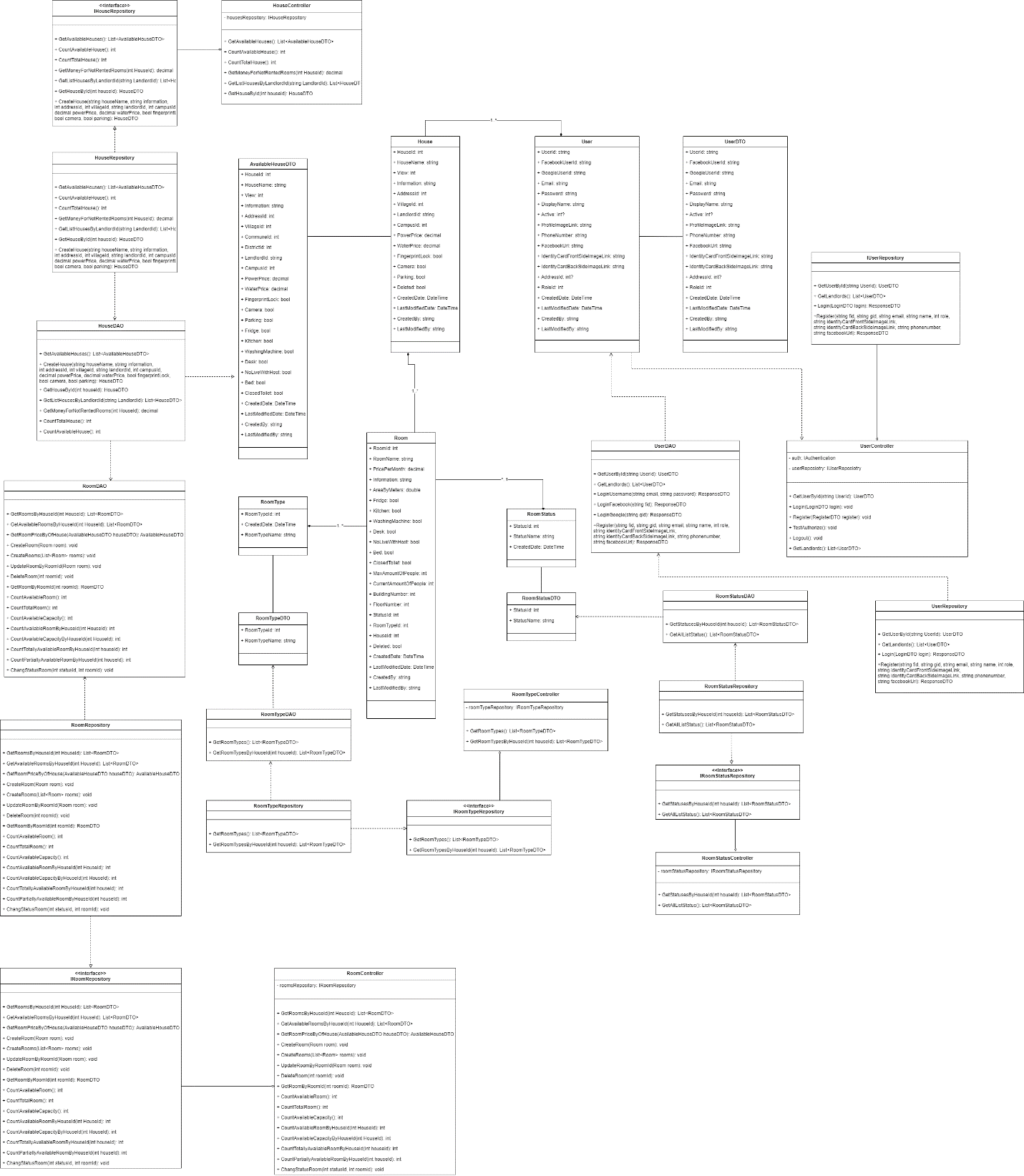
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomTypes() | Return list of all RoomTypes in the Database |
| 02 | GetRoomTypesByHouseId() | Get all roomTypes that this house has |

#### 3.5.3 Sequence Diagram



### 3.6 Landlord – Update Room

#### 3.6.1 Class Diagram



#### 3.6.2 Class Specifications

##### HouseDAO Class

This class performs action to data (CRUD) in House Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAvailableHouses() | Return list of all available Houses in the system (Houses that have available Rooms) with additional information to support filtering |
| 02 | CreateHouse() | Add a new House into the Database |
| 03 | GetHouseById() | Find detail information of a House by its Id |
| 04 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |
| 05 | GetMoneyForNotRentedRooms() | Get total amount of money of rooms that has not been rented (of 1 House) |
| 06 | CountTotalHouse() | Count total number of Houses in the system |
| 07 | CountAvailableHouse() | Count number of available Houses in the system |

##### HouseController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAvailableHouses() | Return list of all available Houses in the system (Houses that have available Rooms) with additional information to support filtering |
| 02 | CreateHouse() | Add a new House into the Database |
| 03 | GetHouseById() | Find detail information of a House by its Id |
| 04 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |
| 05 | GetMoneyForNotRentedRooms() | Get total amount of money of rooms that has not been rented (of 1 House) |
| 06 | CountTotalHouse() | Count total number of Houses in the system |
| 07 | CountAvailableHouse() | Count number of available Houses in the system |

##### RoomDAO Class

This class performs action to data (CRUD) in Room Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price for of 1 House (used in HouseDAO) |
| 04 | CreateRoom() | Add a new Room to the Database |
| 05 | CreateRooms() | Add a list of Rooms of 1 House to the Database |
| 06 | UpdateRoomByRoomId() | Update detail information of 1 Room |
| 07 | DeleteRoom() | Delete a Room from a system without removing its data from the Database |
| 08 | GetRoomByRoomId() | Find a Room by its Id |
| 09 | CountAvailableRoom() | Count the sum of all available Rooms (partially or totally) of all Houses in the system |
| 10 | CountTotalRoom() | Count the sum of all Rooms (available or not) of all Houses in the system |
| 11 | CountAvailableCapacity() | Count the sum of Capacity of all Rooms of all Houses in the system |
| 12 | CountAvailableRoomByHouseId() | Count the sum of all available Rooms (partially or totally) of 1 House |
| 13 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 14 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 15 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |
| 16 | ChangeStatusRoom() | Change Status (Available or Occupied) of 1 Room |

##### RoomController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price for of 1 House (used in HouseDAO) |
| 04 | CreateRoom() | Add a new Room to the Database |
| 05 | CreateRooms() | Add a list of Rooms of 1 House to the Database |
| 06 | UpdateRoomByRoomId() | Update detail information of 1 Room |
| 07 | DeleteRoom() | Delete a Room from a system without removing its data from the Database |
| 08 | GetRoomByRoomId() | Find a Room by its Id |
| 09 | CountAvailableRoom() | Count the sum of all available Rooms (partially or totally) of all Houses in the system |
| 10 | CountTotalRoom() | Count the sum of all Rooms (available or not) of all Houses in the system |
| 11 | CountAvailableCapacity() | Count the sum of Capacity of all Rooms of all Houses in the system |
| 12 | CountAvailableRoomByHouseId() | Count the sum of all available Rooms (partially or totally) of 1 House |
| 13 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 14 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 15 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |
| 16 | ChangeStatusRoom() | Change Status (Available or Occupied) of 1 Room |

##### UserDAO Class

This class performs action to data (CRUD) in User Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetUserById () | Find detail information of User by its Id |
| 02 | GetLandlords() | Get list of Landlords in the database |
| 03 | LoginUsername() | User login in the system |
| 04 | LoginFacebook() | User login with facebook in the system |
| 05 | LoginGoogle () | User login with google in the system |
| 06 | Register() | User create an account |

##### UserController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetUserById () | Find detail information of User by its Id |
| 02 | GetLandlords() | Get list of Landlords in the database |
| 03 | LoginUsername() | User login in the system |
| 04 | LoginFacebook() | User login with facebook in the system |
| 05 | LoginGoogle () | User login with google in the system |
| 06 | Register() | User create an account |

##### RoomStatusDAO Class

This class performs action to data (CRUD) in RoomType Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAllListStatus() | Return list of all RoomStatus in the Database |
| 02 | GetStatusesByHouseId() | Get all RoomStatus that this house has |

##### RoomStatusController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAllStatus() | Return list of all RoomStatus in the Database |
| 02 | GetStatusesByHouseId() | Get all RoomStatus that this house has |

##### RoomTypeDAO Class

This class performs action to data (CRUD) in RoomType Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomTypes() | Return list of all RoomTypes in the Database |
| 02 | GetRoomTypesByHouseId() | Get all roomTypes that this house has |

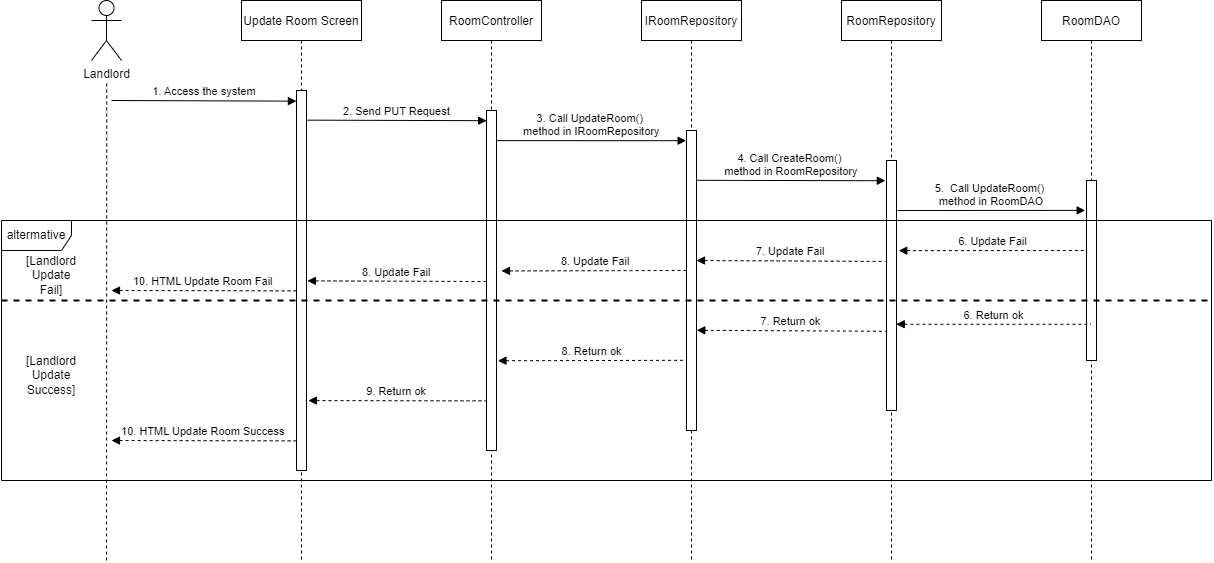
##### RoomTypeController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

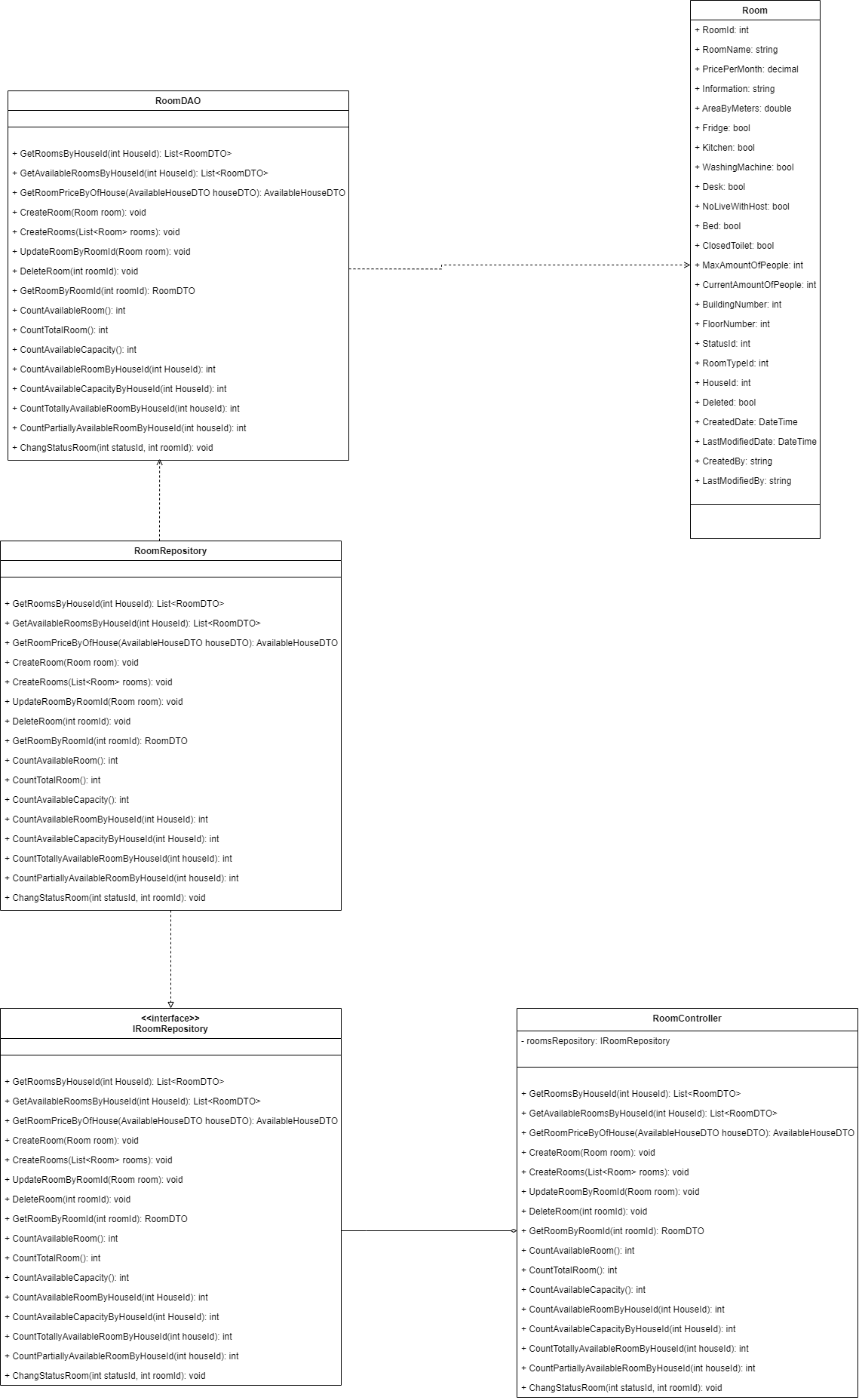
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomTypes() | Return list of all RoomTypes in the Database |
| 02 | GetRoomTypesByHouseId() | Get all roomTypes that this house has |

#### 3.6.3 Sequence Diagram



### 3.7 Landlord – Delete Room

#### 3.7.1 Class Diagram



#### 3.7.2 Class Specifications

##### RoomDAO Class

This class performs action to data (CRUD) in Room Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price for of 1 House (used in HouseDAO) |
| 04 | CreateRoom() | Add a new Room to the Database |
| 05 | CreateRooms() | Add a list of Rooms of 1 House to the Database |
| 06 | UpdateRoomByRoomId() | Update detail information of 1 Room |
| 07 | DeleteRoom() | Delete a Room from a system without removing its data from the Database |
| 08 | GetRoomByRoomId() | Find a Room by its Id |
| 09 | CountAvailableRoom() | Count the sum of all available Rooms (partially or totally) of all Houses in the system |
| 10 | CountTotalRoom() | Count the sum of all Rooms (available or not) of all Houses in the system |
| 11 | CountAvailableCapacity() | Count the sum of Capacity of all Rooms of all Houses in the system |
| 12 | CountAvailableRoomByHouseId() | Count the sum of all available Rooms (partially or totally) of 1 House |
| 13 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 14 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 15 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |
| 16 | ChangeStatusRoom() | Change Status (Available or Occupied) of 1 Room |

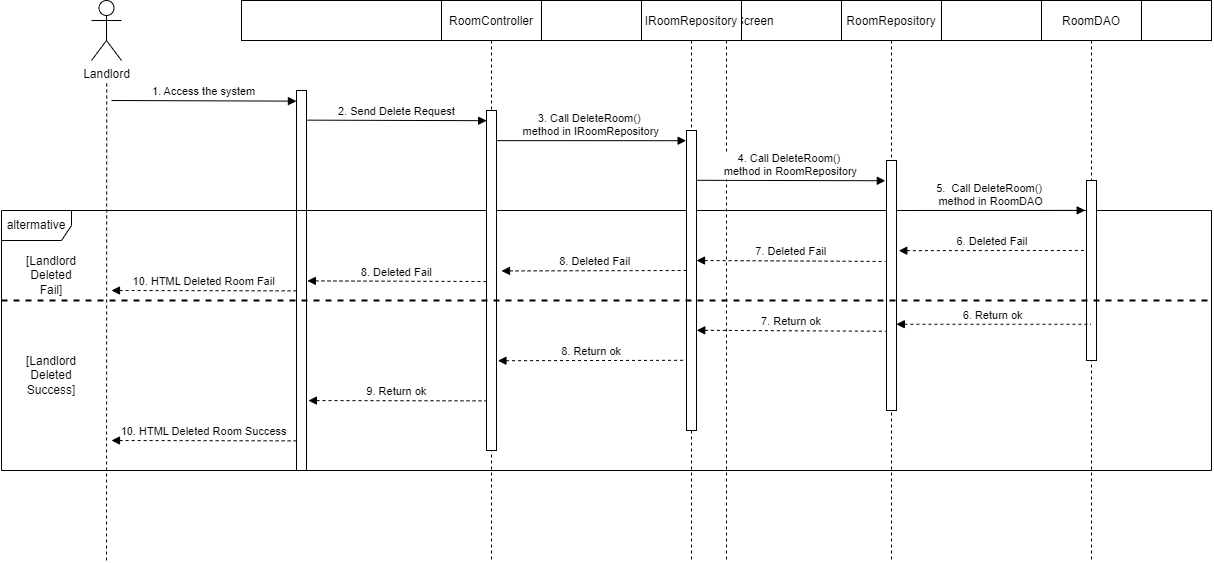
##### RoomController Class

This class takes data from DAO (Repository) and push it to API

**Class Methods**

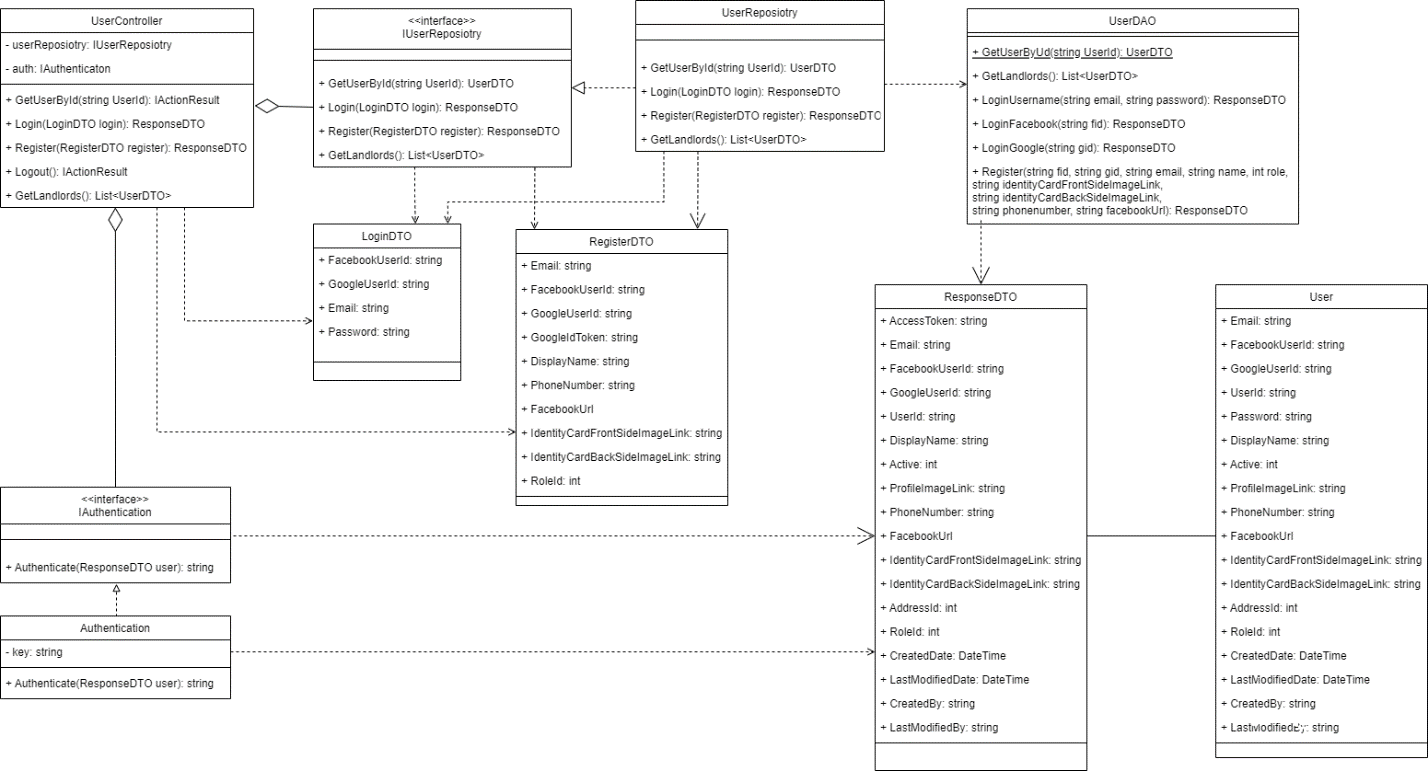
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price for of 1 House (used in HouseDAO) |
| 04 | CreateRoom() | Add a new Room to the Database |
| 05 | CreateRooms() | Add a list of Rooms of 1 House to the Database |
| 06 | UpdateRoomByRoomId() | Update detail information of 1 Room |
| 07 | DeleteRoom() | Delete a Room from a system without removing its data from the Database |
| 08 | GetRoomByRoomId() | Find a Room by its Id |
| 09 | CountAvailableRoom() | Count the sum of all available Rooms (partially or totally) of all Houses in the system |
| 10 | CountTotalRoom() | Count the sum of all Rooms (available or not) of all Houses in the system |
| 11 | CountAvailableCapacity() | Count the sum of Capacity of all Rooms of all Houses in the system |
| 12 | CountAvailableRoomByHouseId() | Count the sum of all available Rooms (partially or totally) of 1 House |
| 13 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 14 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 15 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |
| 16 | ChangeStatusRoom() | Change Status (Available or Occupied) of 1 Room |

#### 3.7.3 Sequence Diagram



### 3.8 Login

#### 3.8.1 Class Diagram



#### 3.8.2 Class Specifications

##### UserController Class

This class takes request, get data from Repository and return Response

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetUserById () | Find detail information of User by its Id |
| 02 | GetLandlords() | Get list of Landlords in the database |
| 03 | LoginUsername() | User login in the system |
| 04 | LoginFacebook() | User login with facebook in the system |
| 05 | LoginGoogle () | User login with google in the system |
| 06 | Register() | User create an account |

##### UserDAO Class

This class performs action to data (CRUD) in User Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetUserById () | Find detail information of User by its Id |
| 02 | GetLandlords() | Get list of Landlords in the database |
| 03 | LoginUsername() | User login in the system |
| 04 | LoginFacebook() | User login with facebook in the system |
| 05 | LoginGoogle () | User login with google in the system |
| 06 | Register() | User create an account |

##### UserRepository Class

This class performs logic and get data from DAO to data (CRUD) in User Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetUserById () | Find detail information of User by its Id |
| 02 | Login() | Get user with login data provided |
| 06 | Register() | User create an account |

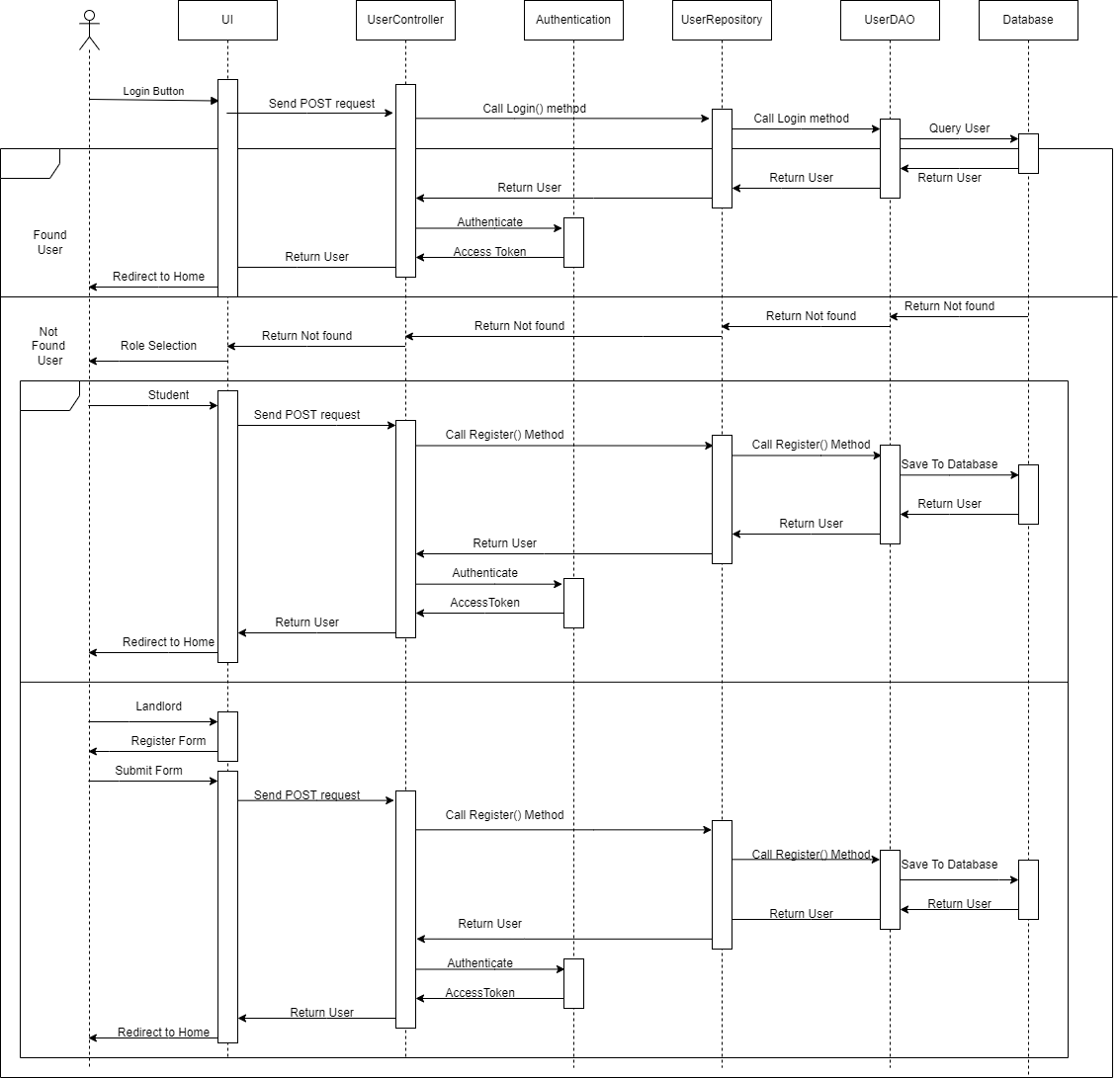
##### Authentication Class

This class create AccessToken for User

**Class Methods**

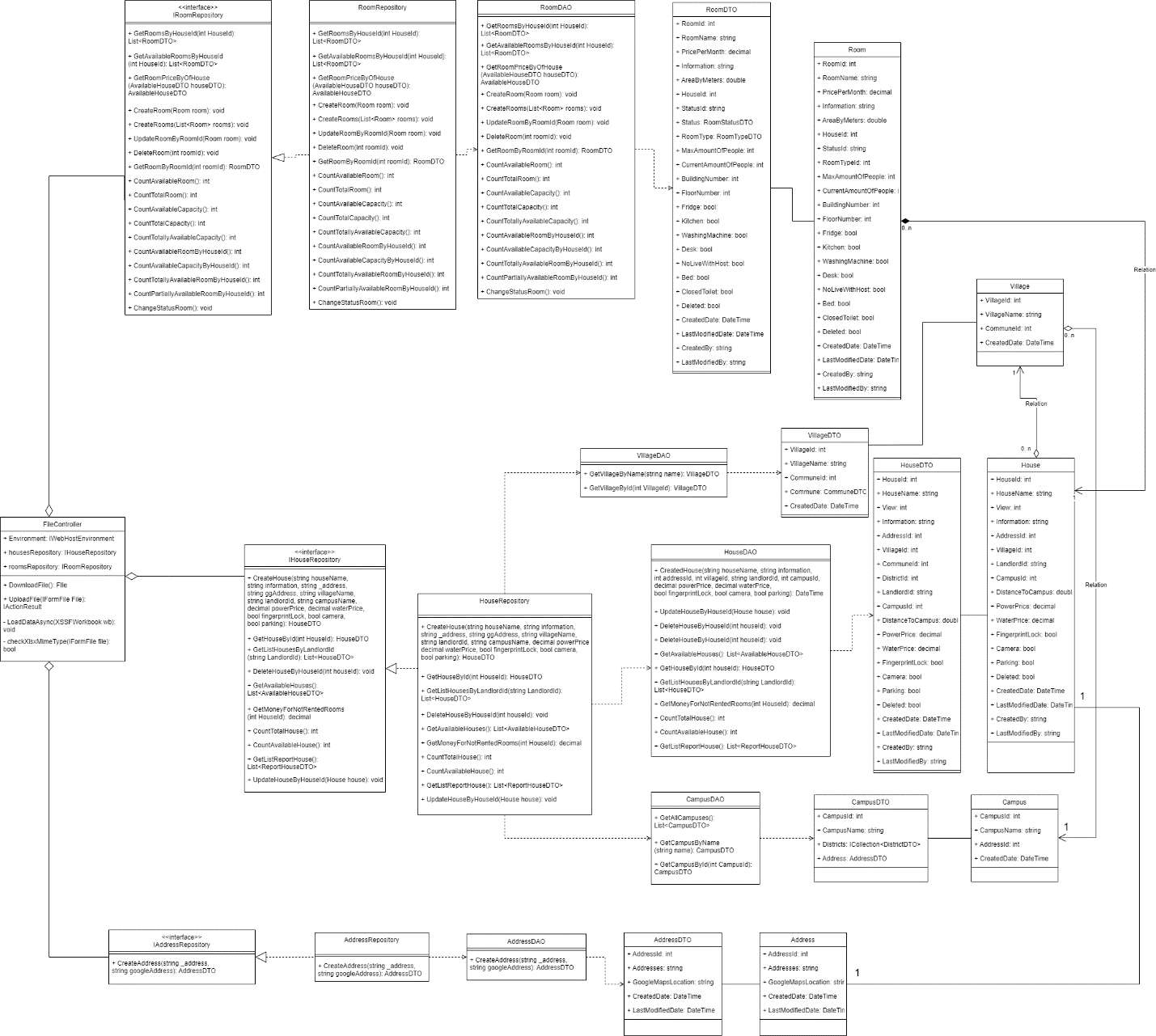
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Authenticate() | Find detail information of User by its Id |

#### 3.8.3 Sequence Diagram



### 3.9 Landlord – Upload House Information

#### 3.9.1 Class Diagram



#### 3.9.2 Class Specifications

##### FileController Class

This class takes request, get data from Repository and return Response

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DownloadFile() | Download Template file to User device |
| 02 | UploadFile() | Get data File from request |
| 03 | LoadDataAsync() | Create house, room data with Data from file |
| 04 | checkXlsxMimeType() | Check upload file mime type |

##### RoomDAO Class

This class class performs action to data (CRUD) in Room Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price for of 1 House (used in HouseDAO) |
| 04 | CreateRoom() | Add a new Room to the Database |
| 05 | CreateRooms() | Add a list of Rooms of 1 House to the Database |
| 06 | UpdateRoomByRoomId() | Update detail information of 1 Room |
| 07 | DeleteRoom() | Delete a Room from a system without removing its data from the Database |
| 08 | GetRoomByRoomId() | Find a Room by its Id |
| 09 | CountAvailableRoom() | Count the sum of all available Rooms (partially or totally) of all Houses in the system |
| 10 | CountTotalRoom() | Count the sum of all Rooms (available or not) of all Houses in the system |
| 11 | CountAvailableCapacity() | Count the sum of Capacity of all Rooms of all Houses in the system |
| 12 | CountAvailableRoomByHouseId() | Count the sum of all available Rooms (partially or totally) of 1 House |
| 13 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 14 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 15 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |
| 16 | ChangeStatusRoom() | Change Status (Available or Occupied) of 1 Room |

##### HouseDAO Class

This class class performs action to data (CRUD) in House Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAvailableHouses() | Return list of all available Houses in the system (Houses that have available Rooms) with additional information to support filtering |
| 02 | CreateHouse() | Add a new House into the Database |
| 03 | GetHouseById() | Find detail information of a House by its Id |
| 04 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |
| 05 | GetMoneyForNotRentedRooms() | Get total amount of money of rooms that has not been rented (of 1 House) |
| 06 | CountTotalHouse() | Count total number of Houses in the system |
| 07 | CountAvailableHouse() | Count number of available Houses in the system |

##### AddressDAO Class

This class performs action to data (CRUD) in Address Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | CreateAddress() | Create Address record and save to database |

##### CampusDAO Class

This class performs action to data (Read) in Campus Table in Database

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAllCampuses() | Get List of All Campuses |
| 02 | GetCampusByName() | Get Campus by Campus Name |
| 03 | GetCampusById() | Get Campus by Campus Id |

##### VillageDAO Class

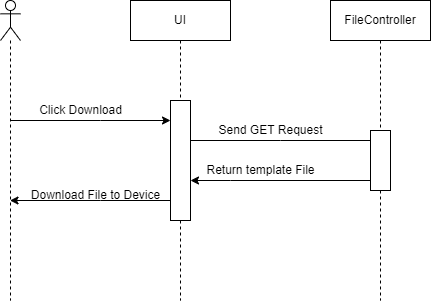
This class performs action to data (Read) in Village Table in Database

**Class Methods**

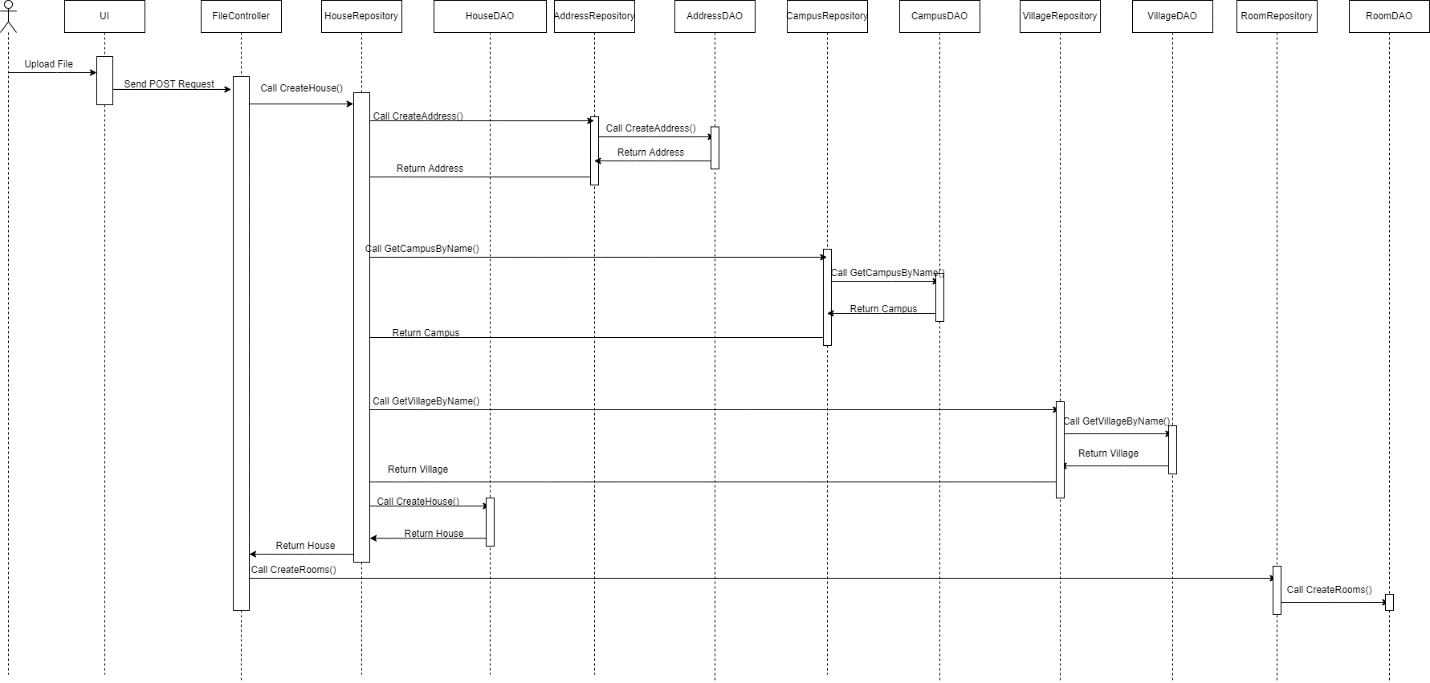
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetVillageByName() | Get Village by Village Name |
| 02 | GetVillageById() | Get Village by Village Id |

#### 3.9.3 Sequence Diagram

###### Download Template



###### Upload File



## 4. Database Tables

### 1. Campuses

*Store the information of all campus for users*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | CampusId | int |  |  | Yes | PK |  |
| 2 | CampusName | nvarchar | 100 |  | Yes |  |  |
| 3 | AddressId | int |  |  | Yes |  | To Table *Addresses* |
| 4 | CreatedDate | datetime |  |  | Yes |  |  |

### 2. UserRoles

*Store the information of all roles of users*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | RoleId | int |  |  | Yes | PK |  |
| 2 | RoleName | nvarchar | 100 |  | Yes |  |  |
| 3 | CreatedDate | datetime |  |  | Yes |  |  |

### 3. Addresses

*Store the information of addresses for Users (Landlords) and Houses*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | AddressId | int |  |  | Yes | PK |  |
| 2 | Addresses | nvarchar | 1000 |  | Yes |  |  |
| 3 | GoogleMapLocation | nvarchar | MAX |  |  |  | Location on Google Map |
| 4 | Deleted | bit |  |  | Yes |  |  |
| 5 | CreatedDate | datetime |  |  | Yes |  |  |
| 6 | LastModifiedDate | datetime |  |  |  |  |  |

### 4. Users

*Store the information of all users*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | UserId | nchar | 30 |  | Yes | PK | Include student id, host id and admission id;  Has specific convention to make |
| 2 | FacebookUserId | nchar | 300 |  |  |  | For people login with Facebook |
| 3 | GoogleUserId | nchar | 300 |  |  |  | For people login with Google |
| 4 | Email | nvarchar | 100 |  |  |  |  |
| 5 | Password | nvarchar | 100 |  |  |  | Hashed with PBKDF2 with SHA-512 |
| 6 | DisplayName | nvarchar | 500 |  |  |  | For Landlord and Staff |
| 7 | Active | int |  |  | Yes |  |  |
| 9 | ProfileImageLink | nvarchar | 500 |  |  |  | For Landlord and Staff |
| 10 | PhoneNumber | nvarchar | 50 |  |  |  | Only for Landlord |
| 11 | FacebookURL | nvarchar | 300 |  |  |  | Only for Landlord |
| 12 | IdentityCardFrontSideImageLink | nvarchar | 500 |  |  |  | Only for Landlord |
| 13 | IdentityCardBackSideImageLink | nvarchar | 500 |  |  |  | Only for Landlord |
| 14 | AddressId | int |  |  |  |  | Only for Landlord;  To Table *Addresses* |
| 15 | RoleId | int |  |  | Yes | FK | To Table *Roles* |
| 16 | CreatedDate | datetime |  |  | Yes |  |  |
| 17 | LastModifiedDate | datetime |  |  |  |  |  |
| 18 | CreatedBy | nchar | 30 |  | Yes | FK | To Table *Users* |
| 19 | LastModifiedBy | nchar | 30 |  |  | FK | To Table *Users* |

### 5. Houses

*Store the information of all houses*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | HouseId | int |  |  | Yes | PK |  |
| 2 | HouseName | nvarchar | 100 |  | Yes |  |  |
| 3 | Information | nvarchar | MAX |  |  |  |  |
| 4 | View | int |  |  |  |  |  |
| 5 | AddressId | int |  |  | Yes |  | To Table *Addresses* |
| 6 | VillageId | int |  |  | Yes | FK | To Table *Villages* |
| 7 | LandlordId | nchar | 30 |  | Yes | FK | To Table *Users* |
| 8 | CampusId |  |  |  |  |  |  |
| 9 | PowerPrice | money |  |  | Yes |  |  |
| 10 | WaterPrice | money |  |  | Yes |  |  |
| 11 | FingerprintLock | bit |  |  |  |  |  |
| 12 | Camera | bit |  |  |  |  |  |
| 13 | Parking | bit |  |  |  |  |  |
| 14 | Deleted | bit |  |  | Yes |  |  |
| 15 | CreatedDate | datetime |  |  | Yes |  |  |
| 16 | LastModifiedDate | datetime |  |  |  |  |  |
| 17 | CreatedBy | nchar | 30 |  | Yes | FK | To Table *Users* |
| 18 | LastModifiedBy | nchar | 30 |  |  | FK | To Table *Users* |

### 6. Rooms

*Store the information of all rooms of houses*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | RoomId | int |  |  | Yes | PK |  |
| 2 | RoomName | nvarchar | 50 |  |  |  |  |
| 3 | PricePerMonth | money |  |  | Yes |  |  |
| 6 | Information | nvarchar | MAX |  |  |  |  |
| 7 | AreaByMeters | float |  |  |  |  |  |
| 8 | Fridge | bit |  |  |  |  |  |
| 9 | Bed | bit |  |  |  |  |  |
| 10 | WashingMachine | bit |  |  |  |  |  |
| 11 | Desk | bit |  |  |  |  |  |
| 12 | Kitchen | bit |  |  |  |  |  |
| 13 | ClosedToilet | bit |  |  |  |  |  |
| 14 | NoLiveWithLandlord | bit |  |  |  |  |  |
| 15 | MaxAmountOfPeople | int |  |  |  |  |  |
| 16 | CurrentAmountOfPeople | int |  |  |  |  |  |
| 17 | BuildingNumber | int |  |  |  |  |  |
| 18 | FloorNumber | int |  |  |  |  |  |
| 19 | StatusId | int |  |  | Yes | FK | To Table *Statuses* |
| 20 | RoomTypeId | int |  |  | Yes | FK | To Table *RoomTypes* |
| 21 | HouseId | int |  |  | Yes | FK | To Table *Houses* |
| 22 | CampusId | int |  |  | Yes | FK | To Table *Campuses* |
| 23 | CreatedDate | datetime |  |  | Yes |  |  |
| 24 | LastModifiedDate | datetime |  |  |  |  |  |
| 25 | CreatedBy | nchar | 30 |  | Yes | FK | To Table *Users* |
| 26 | LastModifiedBy | nchar | 30 |  |  | FK | To Table *Users* |

### 7. Rates

*Store the information of all rates and comments of users*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | RateId | int |  |  | Yes | PK |  |
| 2 | Star | int |  |  | Yes |  |  |
| 3 | Comment | nvarchar | MAX |  |  |  |  |
| 4 | LandlordReply | nvarchar | MAX |  |  |  |  |
| 5 | HouseId | int |  |  | Yes | FK | To Table *Houses* |
| 6 | StudentId | nchar | 30 |  | Yes | FK | To Table *Users* |
| 7 | CreatedDate | datetime |  |  | Yes |  |  |
| 8 | LastModifiedDate | datetime |  |  |  |  |  |
| 9 | CreatedBy | nchar | 30 |  | Yes | FK | To Table *Users* |
| 10 | LastModifiedBy | nchar | 30 |  |  | FK | To Table *Users* |
| 11 | Deleted | bit |  |  | Yes |  |  |

### 8. ImagesOfHouse

*Store the information of all pictures of house*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ImageId | int |  |  | Yes | PK |  |
| 2 | ImageLink | nvarchar | 500 |  | Yes |  |  |
| 3 | HouseId | int |  |  | Yes | FK | To Table *Houses* |
| 4 | CreatedDate | datetime |  |  | Yes |  |  |
| 5 | LastModifiedDate | datetime |  |  |  |  |  |
| 6 | CreatedBy | nchar | 30 |  | Yes | FK | To Table *Users* |
| 7 | LastModifiedBy | nchar | 30 |  |  | FK | To Table *Users* |
| 8 | Deleted | bit |  |  | Yes |  |  |

### 9. ImagesOfRoom

*Store the information of all pictures of room*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | int |  |  | Yes | PK |  |
| 2 | ImageLink | nvarchar | 500 |  | Yes |  |  |
| 3 | RoomId | int |  |  | Yes | FK | To Table *Rooms* |
| 4 | CreatedDate | datetime |  |  | Yes |  |  |
| 5 | LastModifiedDate | datetime |  |  |  |  |  |
| 6 | CreatedBy | nchar | 30 |  | Yes | FK | To Table *Users* |
| 7 | LastModifiedBy | nchar | 30 |  |  | FK | To Table *Users* |
| 8 | Deleted | bit |  |  | Yes |  |  |

### 10. Districts

*Store the information of all districts (for identifying house address)*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | DistrictId | int |  |  | Yes | PK |  |
| 2 | DistrictName | nvarchar | 100 |  | Yes |  |  |
| 3 | CreatedDate | datetime |  |  | Yes |  |  |

### 11. Communes

*Store the information of all communes (for identifying house address)*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | CommuneId | int |  |  | Yes | PK |  |
| 2 | CommunetName | nvarchar | 100 |  | Yes |  |  |
| 3 | DistrictId | int |  |  | Yes | FK | To Table *Districts* |
| 4 | CreatedDate | datetime |  |  | Yes |  |  |

### 12. Villages

*Store the information of all villages (for identifying house address)*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | VillageId | int |  |  | Yes | PK |  |
| 2 | VillageName | nvarchar | 100 |  | Yes |  |  |
| 3 | CommuneId | int |  |  | Yes | FK | To Table *Communes* |
| 4 | CreatedDate | datetime |  |  | Yes |  |  |

### 13. RoomStatuses

*Store the information of all status of a Room*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | StatusId | int |  |  | Yes | PK |  |
| 2 | StatusName | nvarchar | 300 |  | Yes |  |  |
| 3 | CreatedDate | datetime |  |  | Yes |  |  |

### 14. RoomTypes

*Store the information of all types of a Room*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | RoomTypeId | int |  |  | Yes | PK |  |
| 2 | RoomTypeName | nvarchar | 300 |  | Yes |  |  |
| 3 | CreatedDate | datetime |  |  | Yes |  |  |

### 15. Reports

*Store the information of all Reports of Students to Houses*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ReportId | int |  |  | Yes | PK |  |
| 2 | ReportContent | nvarchar | MAX |  | Yes |  |  |
| 3 | StudentId | nchar | 30 |  | Yes | FK | To Table *Users* |
| 4 | HouseId | int |  |  | Yes | FK | To Table *Houses* |
| 5 | CreatedDate | datetime |  |  | Yes |  |  |
| 6 | LastModifiedDate | datetime |  |  |  |  |  |
| 7 | CreatedBy | nchar | 30 |  | Yes | FK | To Table *Users* |
| 8 | LastModifiedBy | nchar | 30 |  |  | FK | To Table *Users* |
| 9 | Deleted | bit |  |  | Yes |  |  |

### 16. RoomHistories

*Store the information of all records of Room usage for Landlords to manage if they want to*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | RoomHistoryId | int |  |  | Yes | PK |  |
| 2 | CustomerName | nvarchar | 800 |  | Yes |  | Name of the person stayed in the Room |
| 3 | RoomId | int |  |  | Yes | FK | To Table *Rooms* |
| 4 | CreatedDate | datetime |  |  | Yes |  |  |
| 5 | LastModifiedDate | datetime |  |  |  |  |  |
| 6 | CreatedBy | nchar | 30 |  | Yes | FK | To Table *Users* |
| 7 | LastModifiedBy | nchar | 30 |  |  | FK | To Table *Users* |
| 8 | Deleted | bit |  |  | Yes |  |  |

### 17. Issues

*Store the information of all records of Room usage for Landlords to manage if they want to*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | IssueId | int |  |  | Yes | PK |  |
| 2 | Description | nvarchar | 800 |  | Yes |  |  |
| 3 | RoomId | int |  |  | Yes | FK | To Table *Rooms* |
| 4 | Deleted | bit |  |  | Yes |  |  |
| 5 | CreatedDate | datetime |  |  | Yes |  |  |
| 6 | LastModifiedDate | datetime |  |  |  |  |  |
| 7 | CreatedBy | nchar | 30 |  | Yes | FK | To Table *Users* |
| 8 | LastModifiedBy | nchar | 30 |  |  | FK | To Table *Users* |

### 18. Orders

*Store the information of all records of Order usage for Staffs to manage if they want to*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | OrderId | int |  |  | Yes | PK |  |
| 2 | StudentId | nchar | 30 |  | Yes | FK | To Table *Users* |
| 3 | StudentName | nvarchar | 100 |  | Yes |  |  |
| 4 | PhoneNumber | nvarchar | 50 |  | Yes |  |  |
| 5 | Email | nvarchar | 100 |  | Yes |  |  |
| 6 | OrderContent | nvarchar |  |  |  |  |  |
| 7 | StatusId | int |  |  | Yes | FK | To Table *OrderStatuses* |
| 8 | OrderedDate | datetime |  |  |  |  |  |
| 9 | SolvedDate | datetime |  |  |  |  |  |

### 19. OrderStatuses

*Store the information of all status of an Order*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | StatusId | int |  |  | Yes | PK |  |
| 2 | StatusName | nvarchar | 300 |  | Yes |  |  |
| 3 | CreatedDate | datetime |  |  | Yes |  |  |