

**Capstone Project Report**

**Report 4 – Software Design Document**

– Hanoi, December 2022 –

**Table of Contents**

[I. Record of Changes 4](#_Toc121991981)

[II. Software Design Document 5](#_Toc121991982)

[1. System Design 5](#_Toc121991983)

[1.1 System Architecture 5](#_Toc121991984)

[1.2 System Architecture Explanation 6](#_Toc121991985)

[1.2 Package Diagram 8](#_Toc121991986)

[2. Database Design 10](#_Toc121991987)

[3. Database Tables 12](#_Toc121991988)

[3.1. Campuses 12](#_Toc121991989)

[3.2. Districts 12](#_Toc121991990)

[3.3. Communes 12](#_Toc121991991)

[3.4. Villages 12](#_Toc121991992)

[3.5. Addresses 13](#_Toc121991993)

[3.6. UserRoles 13](#_Toc121991994)

[3.7. UserStatuses 13](#_Toc121991995)

[3.8. Users 13](#_Toc121991996)

[3.9. Houses 14](#_Toc121991997)

[3.10. ImagesOfHouse 15](#_Toc121991998)

[3.11. Issues 15](#_Toc121991999)

[3.12. Rates 15](#_Toc121992000)

[3.13. RoomStatuses 16](#_Toc121992001)

[3.14. RoomTypes 16](#_Toc121992002)

[3.15. Rooms 16](#_Toc121992003)

[3.16. ImagesOfRoom 17](#_Toc121992004)

[3.17. RoomHistories 17](#_Toc121992005)

[3.18. ReportStatuses 17](#_Toc121992006)

[3.19. Reports 18](#_Toc121992007)

[3.20. OrderStatuses 18](#_Toc121992008)

[3.21. Orders 18](#_Toc121992009)

[4. Detailed Design 19](#_Toc121992010)

[4.1 Home Page 19](#_Toc121992011)

[4.2 House Detail 25](#_Toc121992012)

[4.3 Room Detail 30](#_Toc121992013)

[4.5 Landlord – List Room 34](#_Toc121992014)

[4.6 Landlord – Create Room 40](#_Toc121992015)

[4.7 Landlord – Update Room 45](#_Toc121992016)

[4.8 Landlord – Delete Room 50](#_Toc121992017)

[4.9 Login 53](#_Toc121992018)

[4.10 Landlord – Upload House Information 56](#_Toc121992019)

[4.11 Staff – Dashboard 60](#_Toc121992020)

[4.12 Staff – List Landlord 66](#_Toc121992021)

[4.13 Staff - Landlord Detail 69](#_Toc121992022)

[4.14 Staff - House Detail 74](#_Toc121992023)

[4.15 Staff - Room Detail 78](#_Toc121992024)

[4.16 Staff – List House 82](#_Toc121992025)

[4.17 Staff – List Report 84](#_Toc121992026)

[4.18 Staff – List Order 87](#_Toc121992027)

# 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 |
| 08/12/2022 | M | ThongPQ | System Architecture |
| 11/12/2022 | M | HuyenBN | Database Tables |
| 12/12/2022 | M | HuyenBN | Database Design |
| 14/12/2022 | M | ThongPQ | System Architecture |
| 14/12/2022 | M | HuyenBN | System Architecture Diagram |
| 14/12/2022 | M | GiangNT | System Architecture |
| 16/12/2022 | M | HuyenBN | System Architecture Diagram |

\*A - Added M - Modified D - Deleted

# II. Software Design Document

## 1. System Design

### 1.1 System Architecture

#### 1.1.1 Diagram



#### 1.1.2 Explanation

(1), (2) FHF Front-end sends a request to FHF Back-end through API Gateway.

(3) FHF Back-end sends a request to Database.

(4) FHF Front-end sends a request to Facebook API to perform authentication.

(5), (6) FHF Front-end sends a request to Google Cloud API to perform authentication, FHF Back-end verifies authentication token. FHF Front-end also sends request to use Google Map APIs (Distance Matrix API and Maps JavaScript API)

(7) FHF Front-end sends a request to Image Server to retrieve images

(8) FHF Back-end sends a reqeust to Image Server to store images and get image URL

### 1.2 System Architecture Explanation

The FHF Back-end system will be deployed on Amazon EC2 Linux Virtual Machine while FHF Front-end will be deployed on Vercel. We aim to deliver a secured and responsive system with high usability. In the following section, we will explain the function and mechanism of each unit in the system architecture design.

#### 1.2.1 Angular



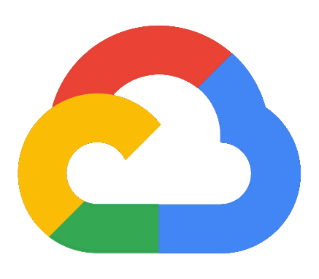
**Angular** is a platform and framework for building single-page client applications using HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your applications. **We use it to build entire website view instead of plain HTML/JavaScript.**

#### 1.2.2 Facebook API



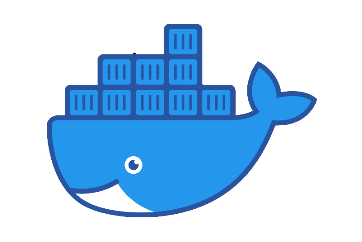
**Facebook API** is a bundle of solutions used as a primary way to get data in and out of the platform. It enables developers and app users to access the functionality of this network: user information, photos and videos, messages and more. **We use it to perform authentication by Facebook**.

#### 1.2.3 Google Cloud API



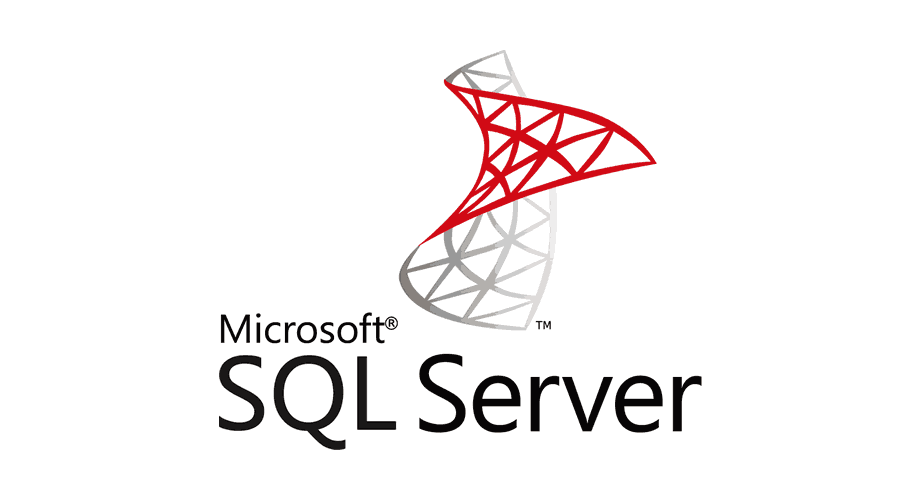
**Google Cloud APIs** are programmatic interfaces to Google Cloud Platform services. They are a key part of Google Cloud Platform, allowing you to easily add the power of everything from computing to networking to storage to machine-learning-based data analysis to us applications. **We use it perform authentication by Google and to use Google Map APIs.**

#### 1.2.4 Docker



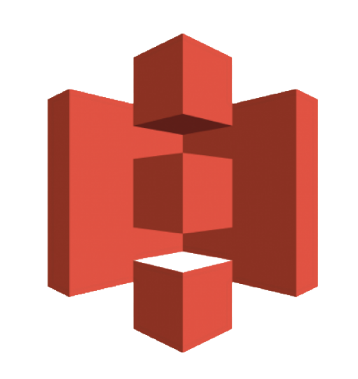
**Docker** is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production. **We use it to containerize our FHF Back-end System so that it can run on multiple platforms and be independent of Operating System.**

#### 1.2.5 SQL Server



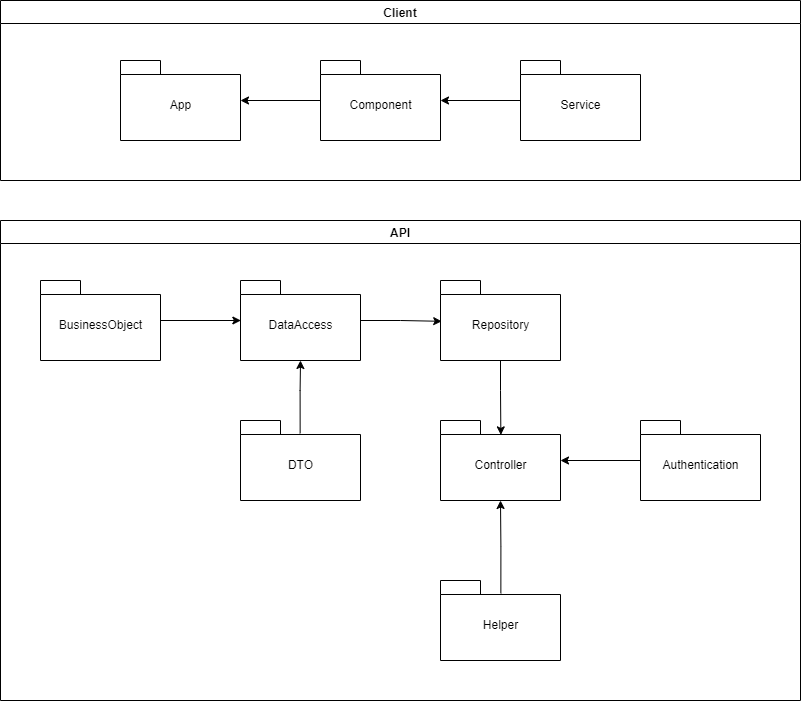
**Microsoft SQL Server** is a relational database management system developed by Microsoft. **We use it to manage FHF operational data.**

#### 1.2.6 Amazon S3



**Amazon S3** is cloud object storage with industry-leading scalability, data availability, security, and performance. **We use it as a Server for Image Storage.**

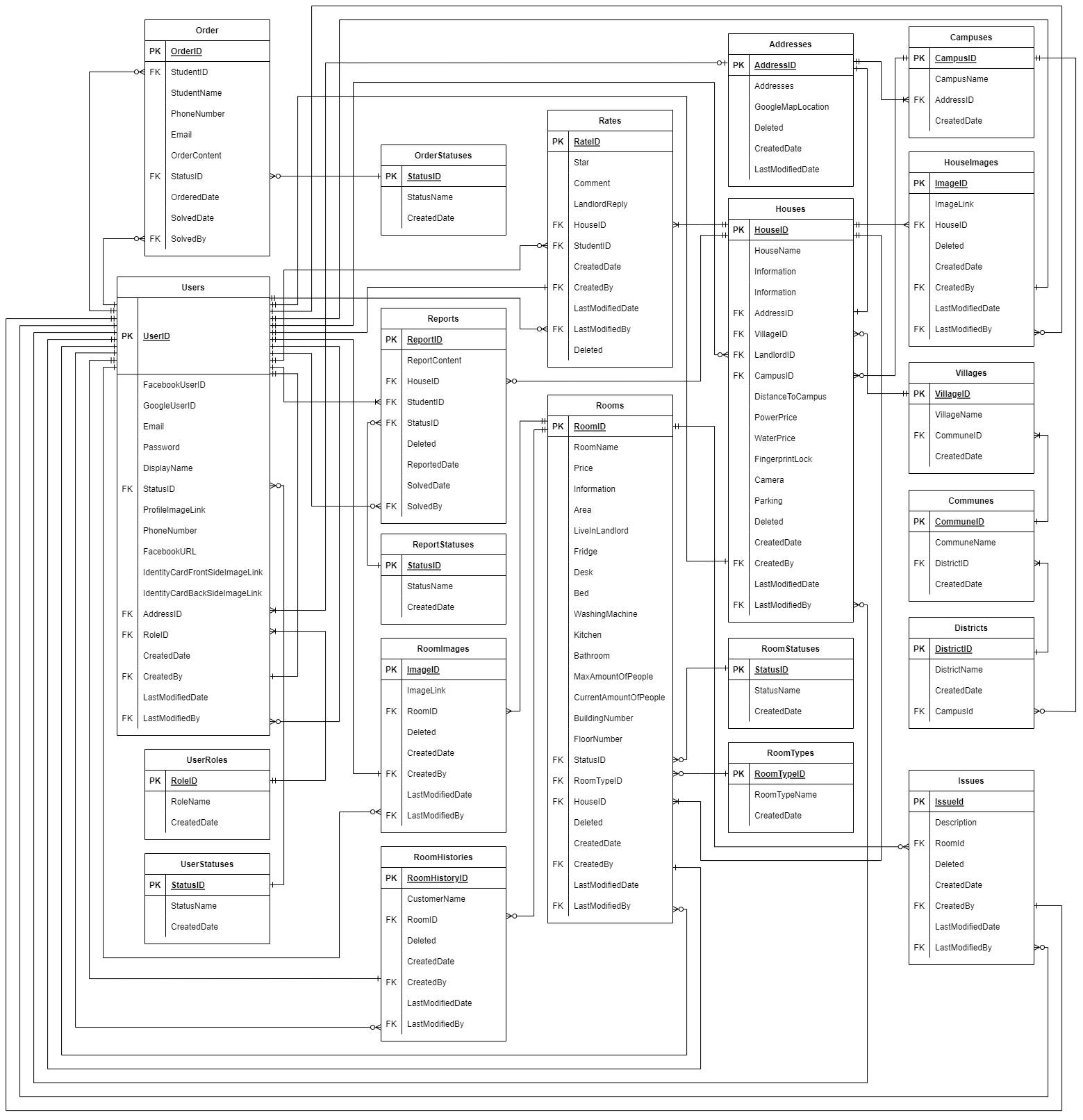
### 1.2 Package Diagram



***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* | *ReportStatuses* | *Store the information of all status of a Report*   * *Primary key: StatusId* |
| *13* | *Reports* | *Store the information of all Reports of Students to Houses*   * *Primary key: ReportId* * *Foreign keys: HouseId, StudentId, StatusId, SolvedBy* |
| *14* | *RoomHistories* | *Store the information of all records of histories of a room*   * *Primary key: RoomHistoryId* * *Foreign keys: RoomId, CreatedBy, LastModifiedBy* |
| *15* | *Rooms* | *Store the information of all rooms of houses*   * *Primary key: RoomId* * *Foreign keys: StatusId, RoomTypeId, HouseId, CreatedBy, LastModifiedBy* |
| *16* | *RoomStatuses* | *Store the information of all status of a Room*   * *Primary key: StatusId* |
| *17* | *RoomTypes* | *Store the information of all types of a Room*   * *Primary key: RoomTypeId* |
| *18* | *UserRoles* | *Store the information of all roles of users*   * *Primary key: RoleId* |
| *19* | *UserStatuses* | *Store the information of all status of an User*  *Primary key: StatusId* |
| *20* | *Users* | *Store the information of all users*   * *Primary key: UserId* * *Foreign keys: RoleId, StatusId, AddressId, CreatedBy, LastModifiedBy* |
| *21* | *Villages* | *Store the information of all villages (for identifying house address)*   * *Primary key: VillageId* * *Foreign keys: CommuneId* |

## 3. Database Tables

### 3.1. 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 |  |  |

### 3.2. 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 | CampusId | int |  |  | Yes | FK | To Table *Campuses* |
| 4 | CreatedDate | datetime |  |  | Yes |  |  |

### 3.3. 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 |  |  |

### 3.4. 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 |  |  |

### 3.5. 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 |  |  |  |  |  |

### 3.6. 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 |  |  |

### 3.7. UserStatuses

*Store the information of all status of an User*

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

### 3.8. 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 | StatusId | int |  |  | Yes | FK | To Table *UserStatuses* |
| 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* |

### 3.9. 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 | int |  |  | Yes | FK | To Table *Campuses* |
| 9 | DistanceToCampus | float |  |  | Yes |  |  |
| 10 | PowerPrice | money |  |  | Yes |  |  |
| 11 | WaterPrice | money |  |  | Yes |  |  |
| 12 | FingerprintLock | bit |  |  |  |  |  |
| 13 | Camera | bit |  |  |  |  |  |
| 14 | Parking | bit |  |  |  |  |  |
| 15 | Deleted | bit |  |  | Yes |  |  |
| 16 | CreatedDate | datetime |  |  | Yes |  |  |
| 17 | LastModifiedDate | datetime |  |  |  |  |  |
| 18 | CreatedBy | nchar | 30 |  | Yes | FK | To Table *Users* |
| 19 | LastModifiedBy | nchar | 30 |  |  | FK | To Table *Users* |

### 3.10. 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 |  |  |

### 3.11. 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* |

### 3.12. 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 |  |  |

### 3.13. 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 |  |  |

### 3.14. 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 |  |  |

### 3.15. 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* |
| 27 | Deleted | bit |  |  | Yes |  |  |

### 3.16. 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 |  |  |

### 3.17. 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 |  |  |

### 3.18. ReportStatuses

*Store the information of all status of a Report*

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

### 3.19. 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 | ReportedDate | datetime |  |  | Yes |  |  |
| 6 | SolvedDate | datetime |  |  |  |  |  |
| 7 | SolvedBy | nchar | 30 |  |  | FK | To Table *Users* |
| 8 | Deleted | bit |  |  | Yes |  |  |

### 3.20. 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 |  |  |

### 3.21. Orders

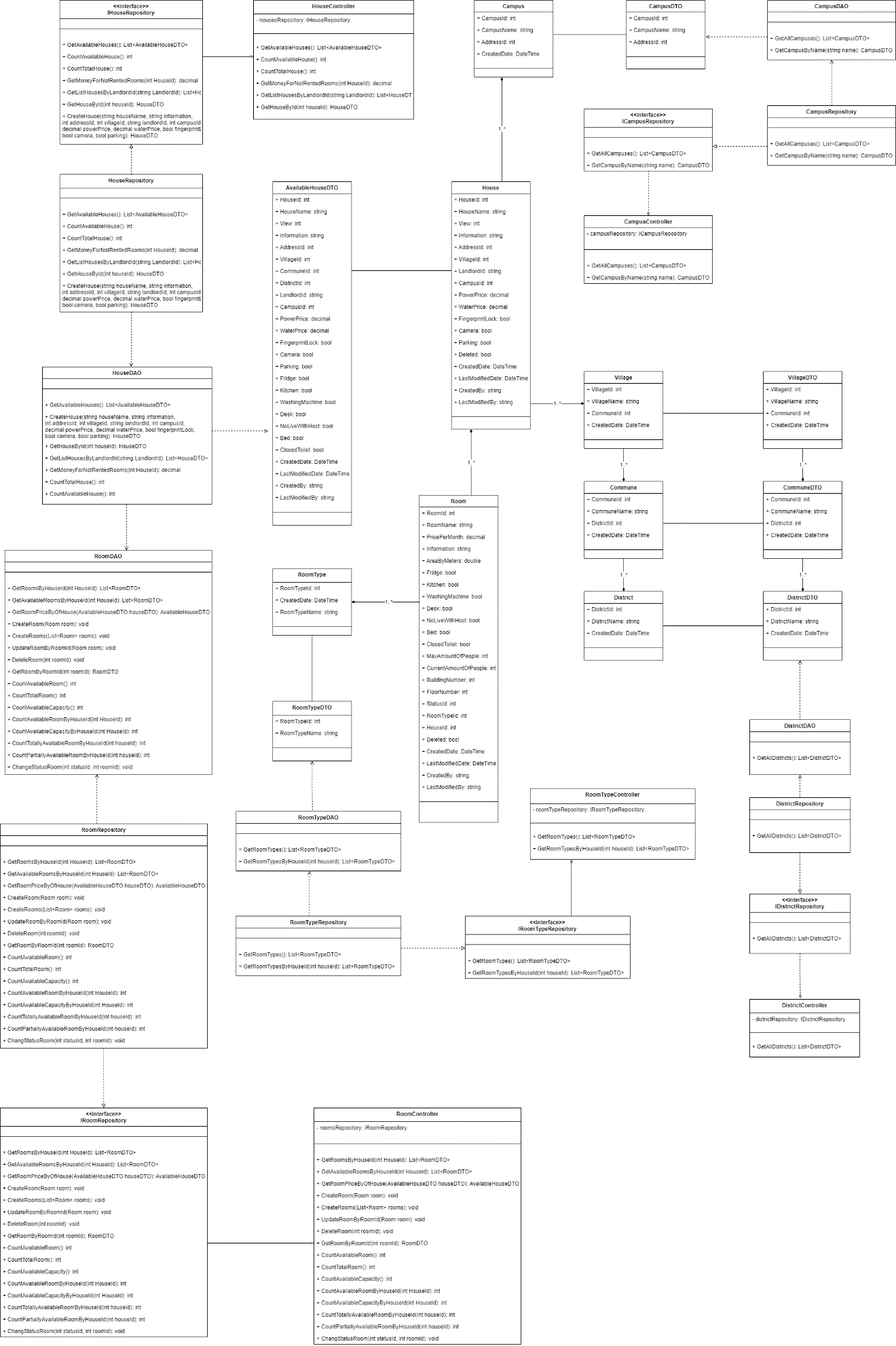
*Store the information of all records of Order usage for Staffs to manage if they want to*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **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 |  |  |  |  |  |
| 10 | SolvedBy | nchar | 30 |  |  | FK | To Table *Users* |

## 4. Detailed Design

### 4.1 Home Page

#### 4.1.1 Class Diagram



#### 4.1.2 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 |

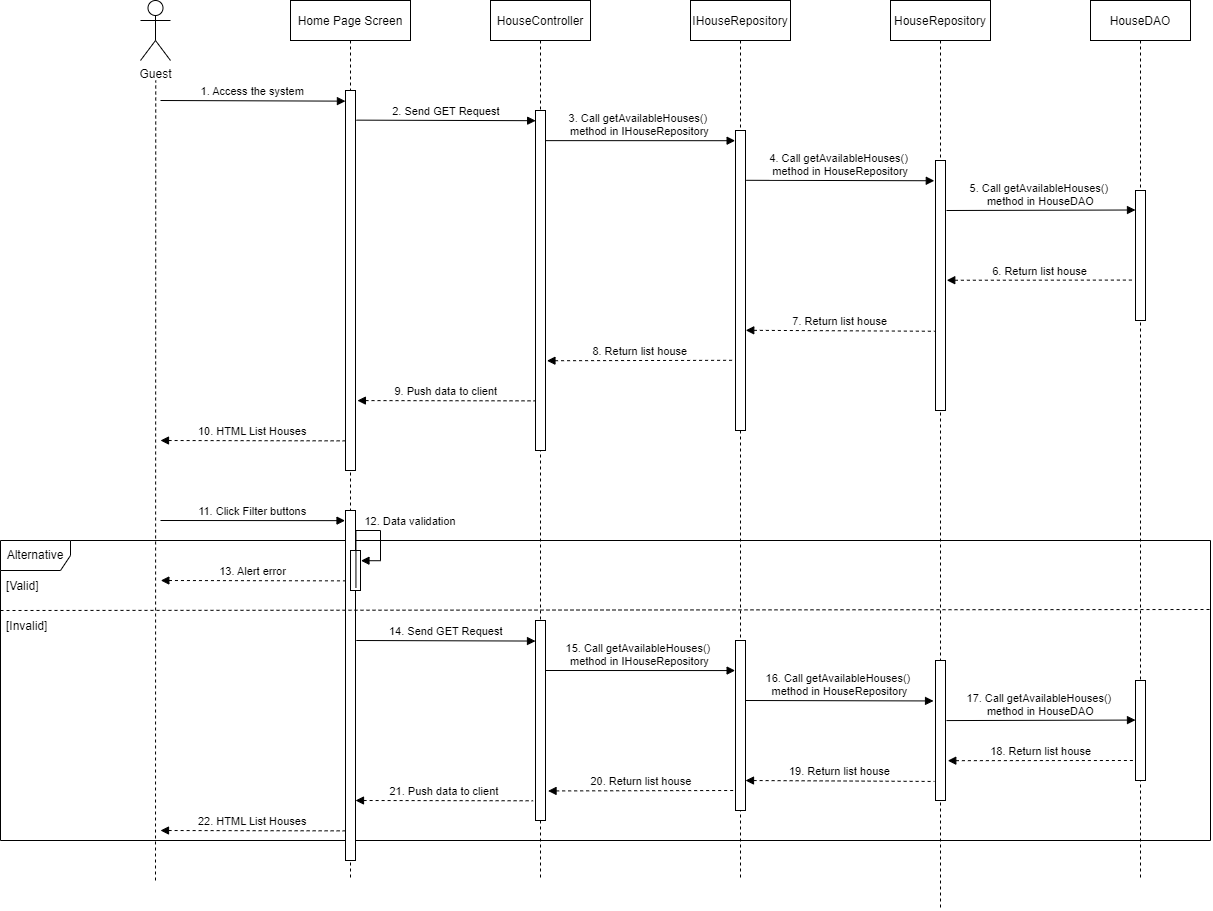
##### RoomDAO Class

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

**Class Methods**

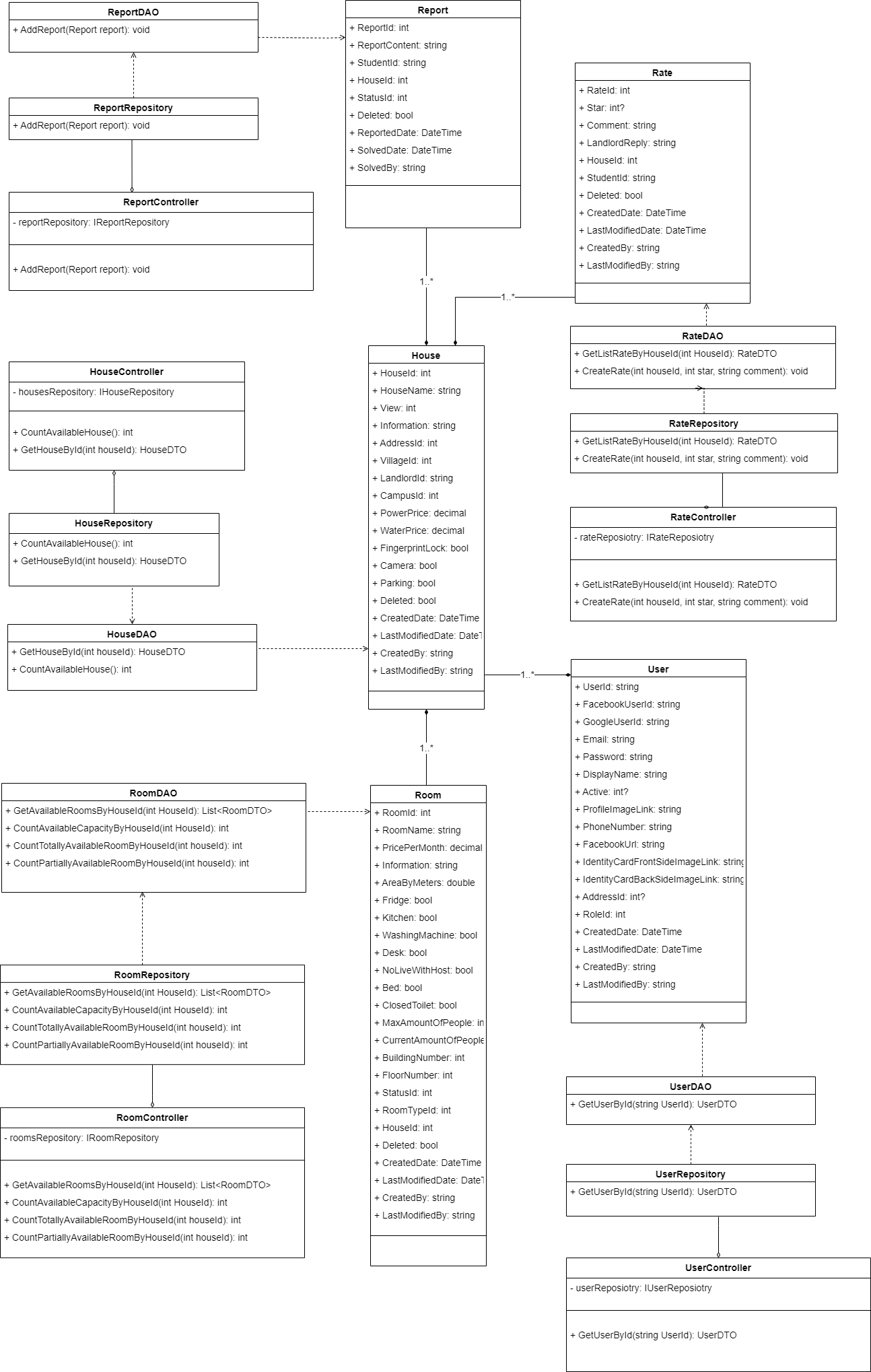
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | GetRoomsByHouseId() | Return list of Rooms of 1 House |
| 02 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 03 | GetRoomPriceByOfHouse() | Get Lowest room price and Highest room price 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 |

#### 4.1.3 Sequence Diagram



### 4.2 House Detail

#### 4.2.1 Class Diagram



#### 4.2.2 Class Specifications

##### HouseDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetHouseById() | Find detail information of a House by its Id |
| 02 | 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 | GetHouseById() | Find detail information of a House by its Id |
| 02 | 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 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 02 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 03 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 04 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |

##### RoomController Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAvailableRoomsByHouseId() | Get list of available Rooms (partially or totally) of 1 House |
| 02 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 03 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 04 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |

##### 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 |

##### 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 |

#### 4.2.3 Sequence Diagram



### 4.3 Room Detail

#### 4.3.1 Class Diagram



#### 4.3.2 Class Specifications

##### HouseDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetHouseById() | Find detail information of a House by its Id |

##### HouseController Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetHouseById() | Find detail information of a House by its Id |

##### RoomDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomByRoomId() | Find a Room by its Id |

##### RoomController Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomByRoomId() | Find a Room by its Id |

##### 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 |

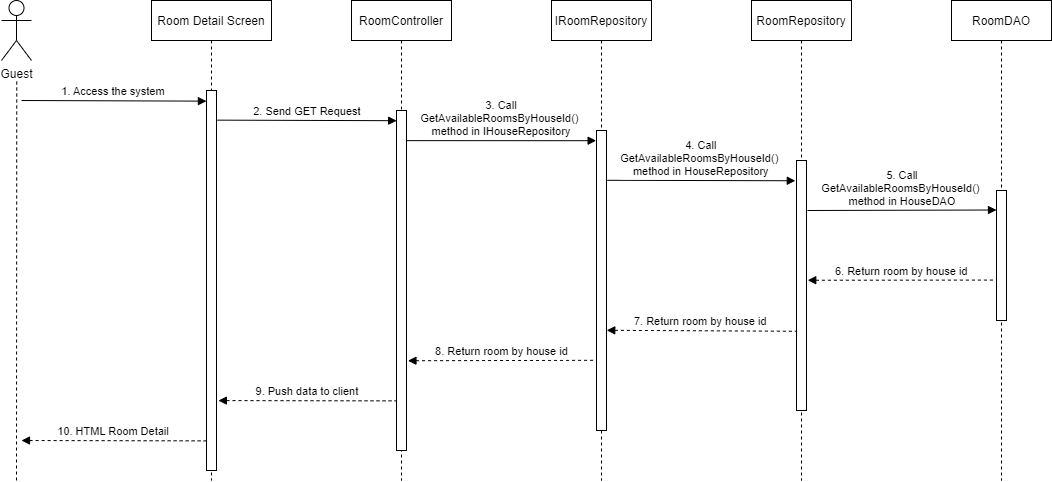
##### UserController Class

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 |

#### 4.3.3 Sequence Diagram



### 4.4 Landlord – Create House

#### 4.4.1 Class Diagram

Diagram

Description automatically generated

#### 4.4.2 Class Specification

##### 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 |

##### 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 |

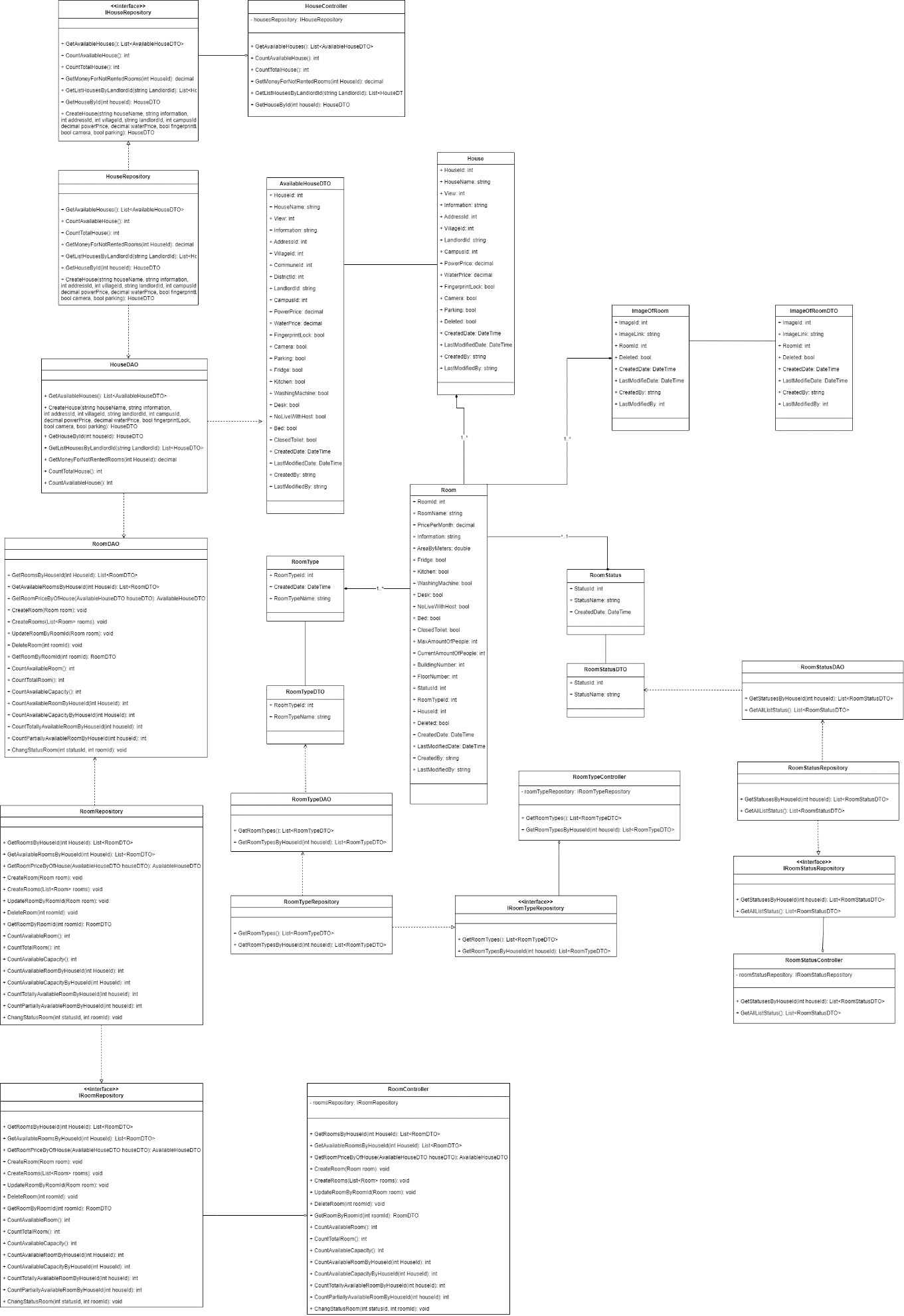
#### 4.4.3 Sequence Diagram

Diagram

Description automatically generated

### 4.5 Landlord – List Room

#### 4.5.1 Class Diagram



#### 4.5.2 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 |

##### 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 |

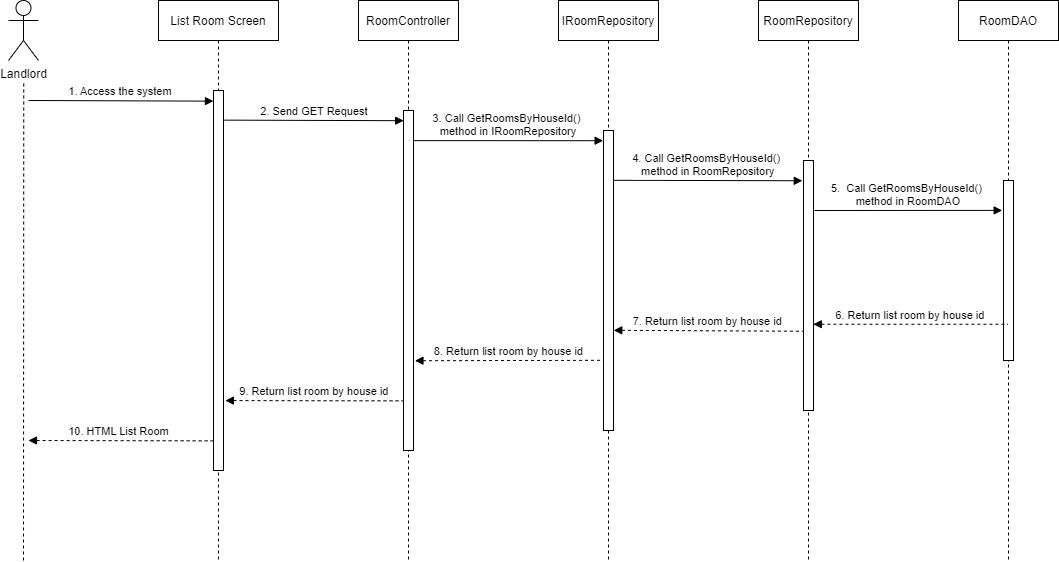
##### RoomTypeController Class

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

**Class Methods**

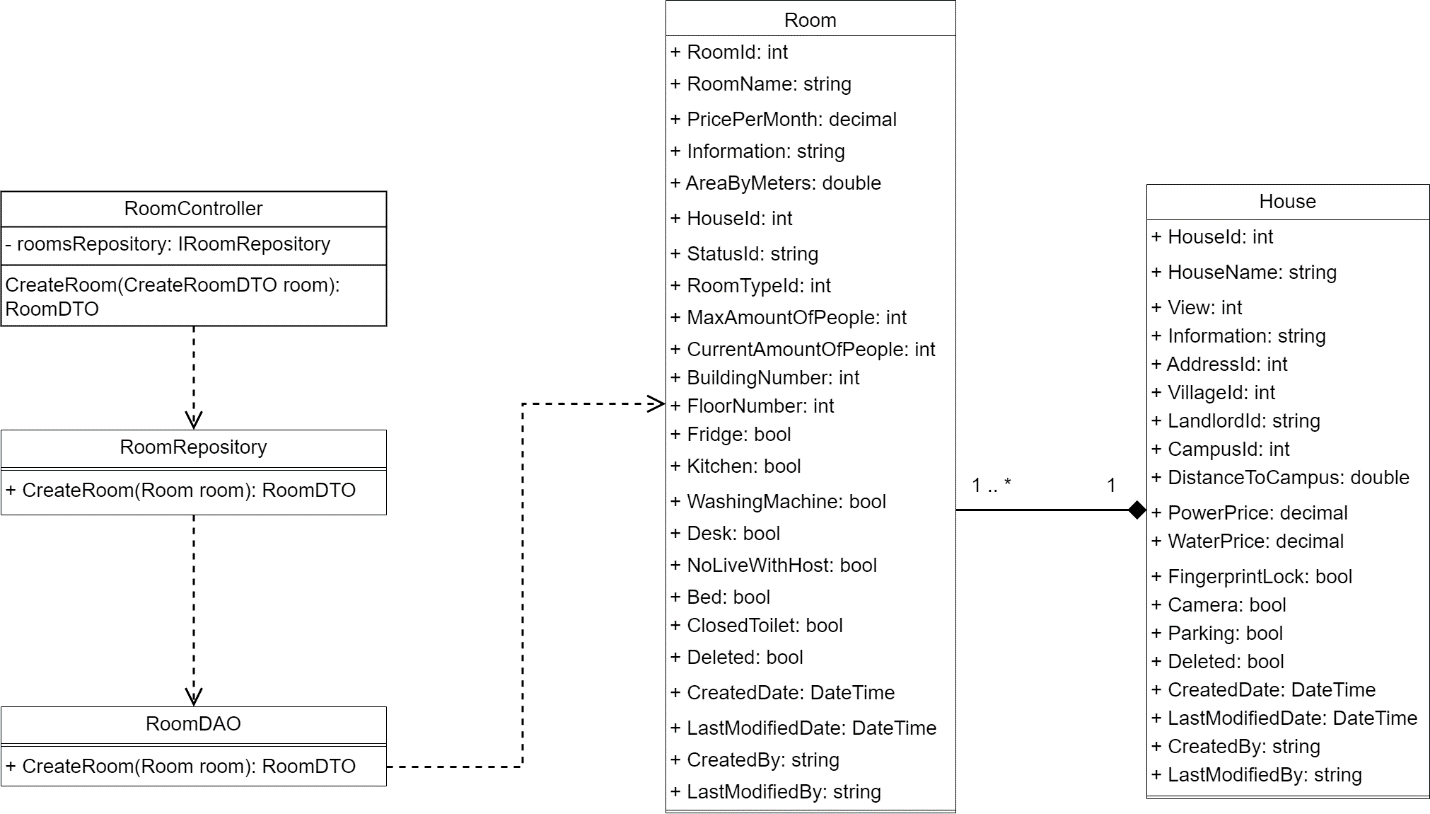
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomTypes() | Return list of all RoomTypes in the Database |
| 02 | GetRoomTypesByHouseId() | Get all roomTypes that this house has |

#### 4.5.3 Sequence Diagram



### 4.6 Landlord – Create Room

#### 4.6.1 Class Diagram



#### 4.6.2 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 |

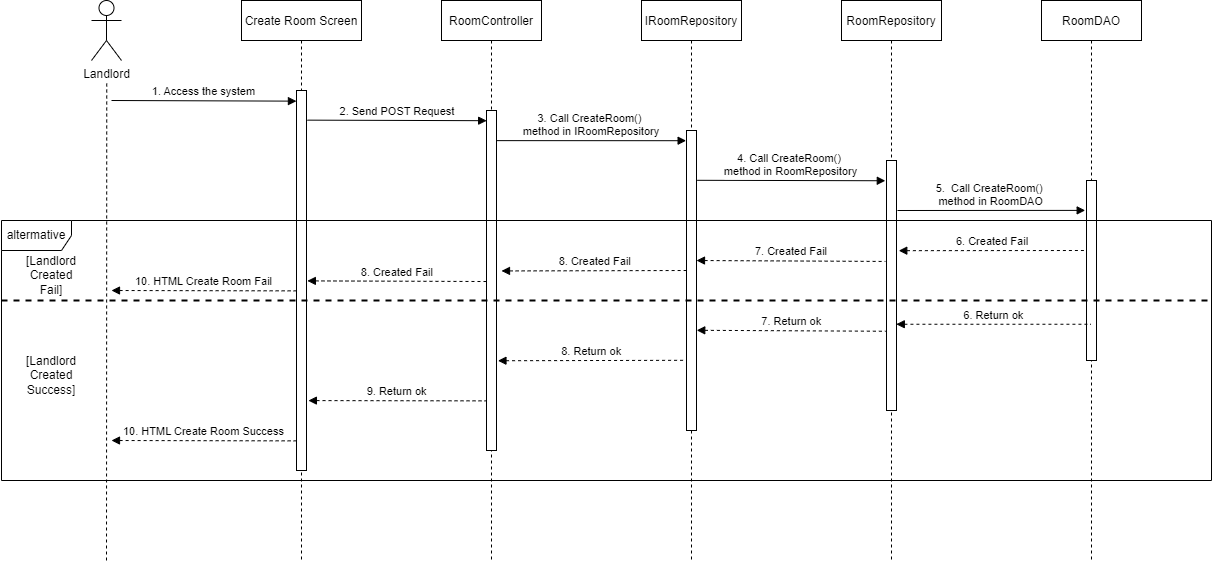
##### RoomTypeController Class

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

**Class Methods**

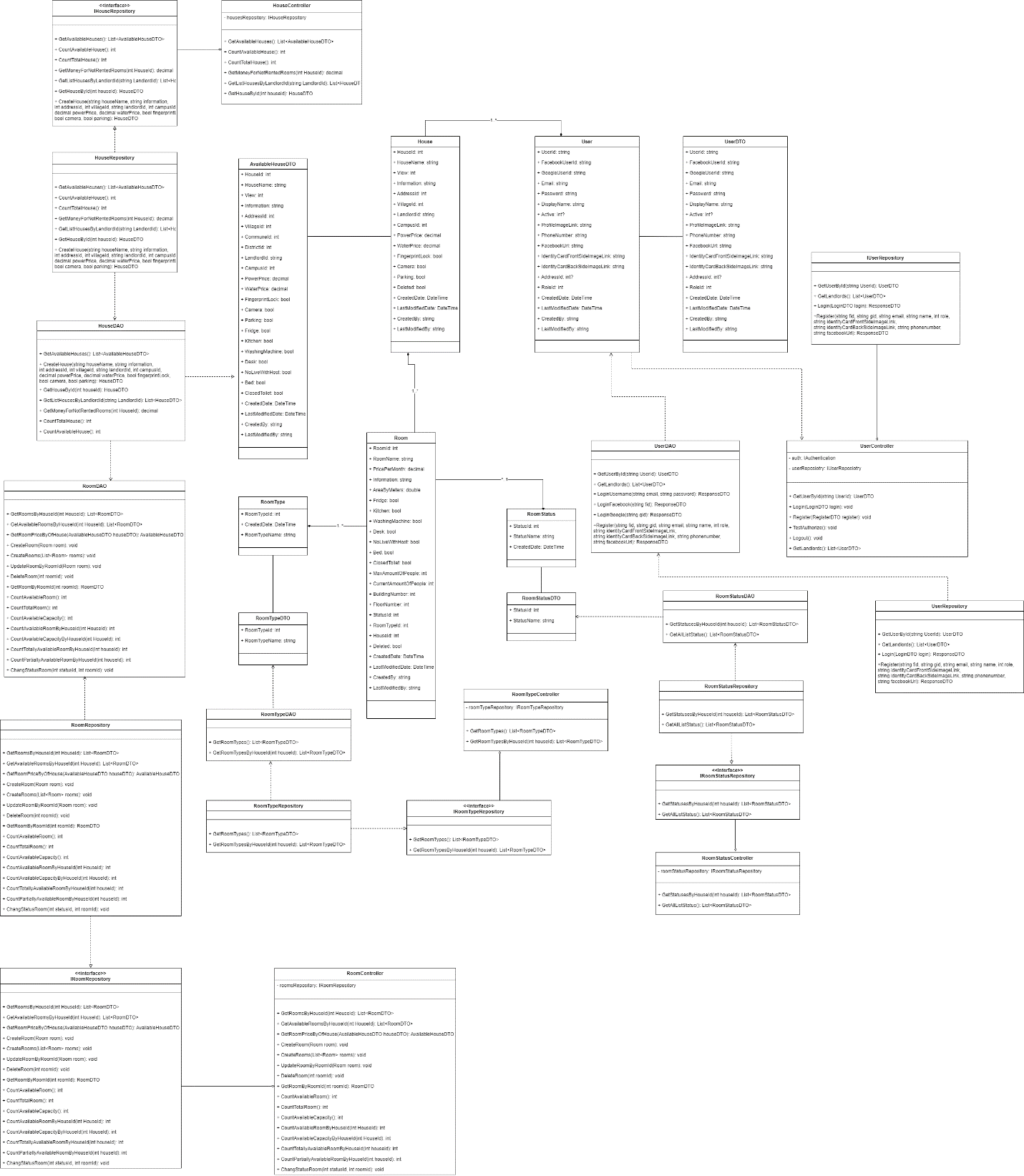
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomTypes() | Return list of all RoomTypes in the Database |
| 02 | GetRoomTypesByHouseId() | Get all roomTypes that this house has |

#### 4.6.3 Sequence Diagram



### 4.7 Landlord – Update Room

#### 4.7.1 Class Diagram



#### 4.7.2 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 |

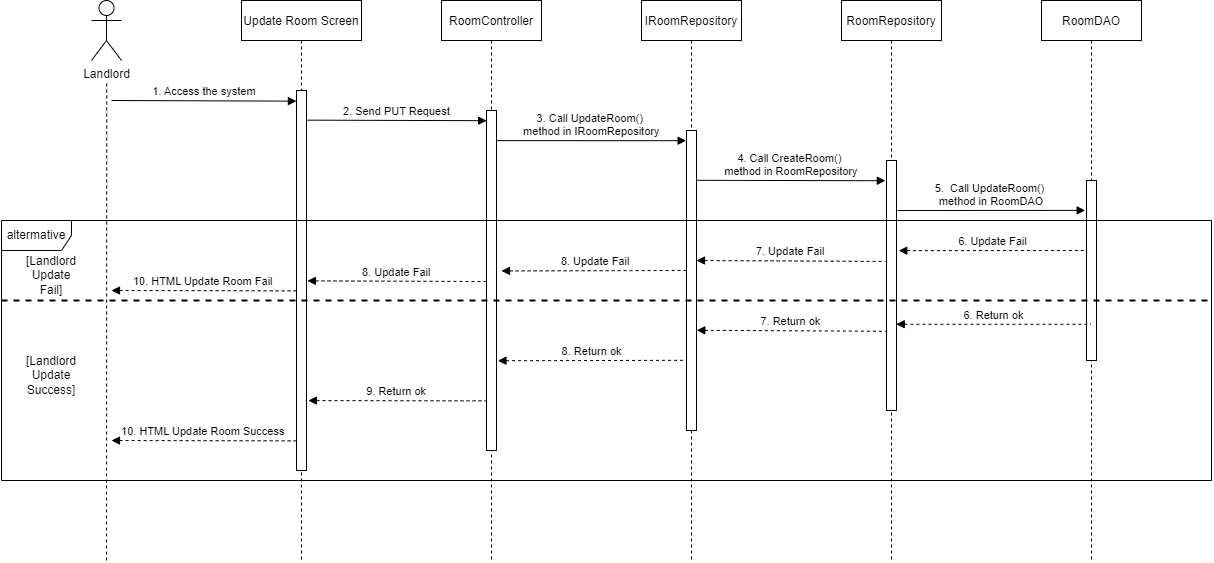
##### RoomTypeController Class

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

**Class Methods**

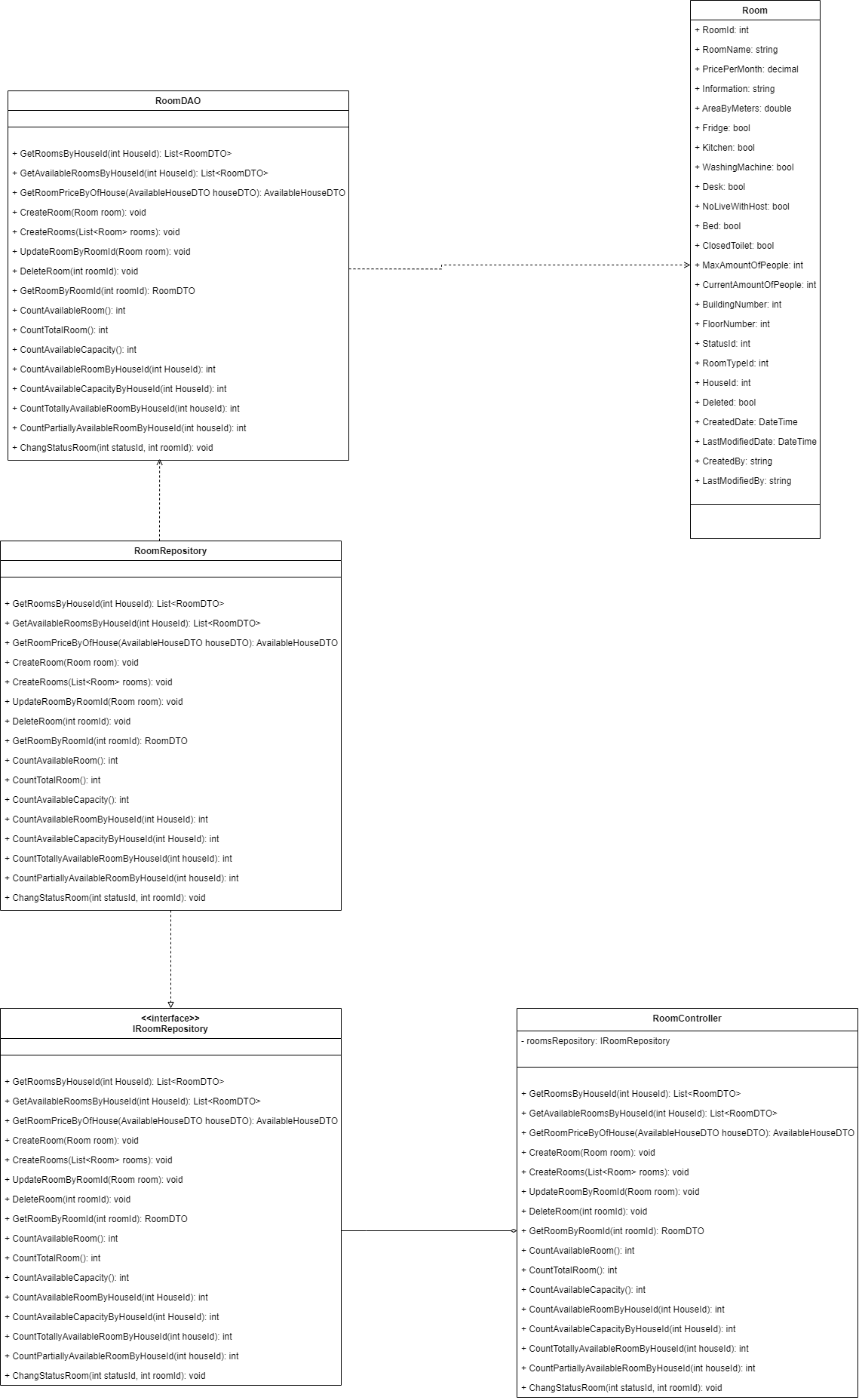
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomTypes() | Return list of all RoomTypes in the Database |
| 02 | GetRoomTypesByHouseId() | Get all roomTypes that this house has |

#### 4.7.3 Sequence Diagram



### 4.8 Landlord – Delete Room

#### 4.8.1 Class Diagram



#### 4.8.2 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 |

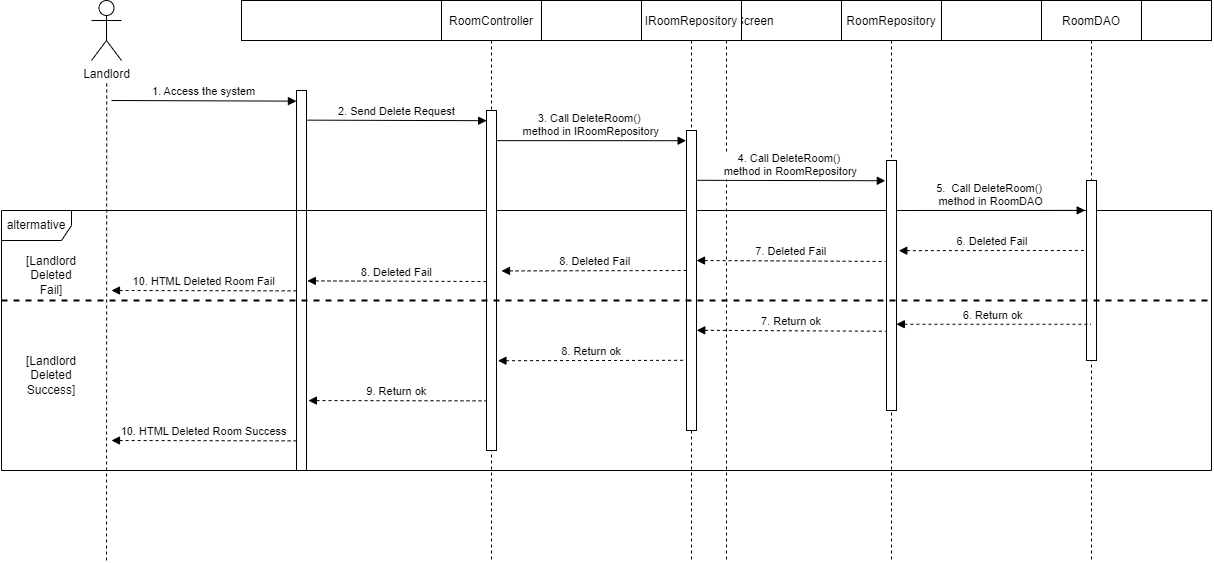
##### RoomController Class

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 |

#### 4.8.3 Sequence Diagram



### 4.9 Login

#### 4.9.1 Class Diagram

A picture containing graphical user interface

Description automatically generated

#### 4.9.2 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 |

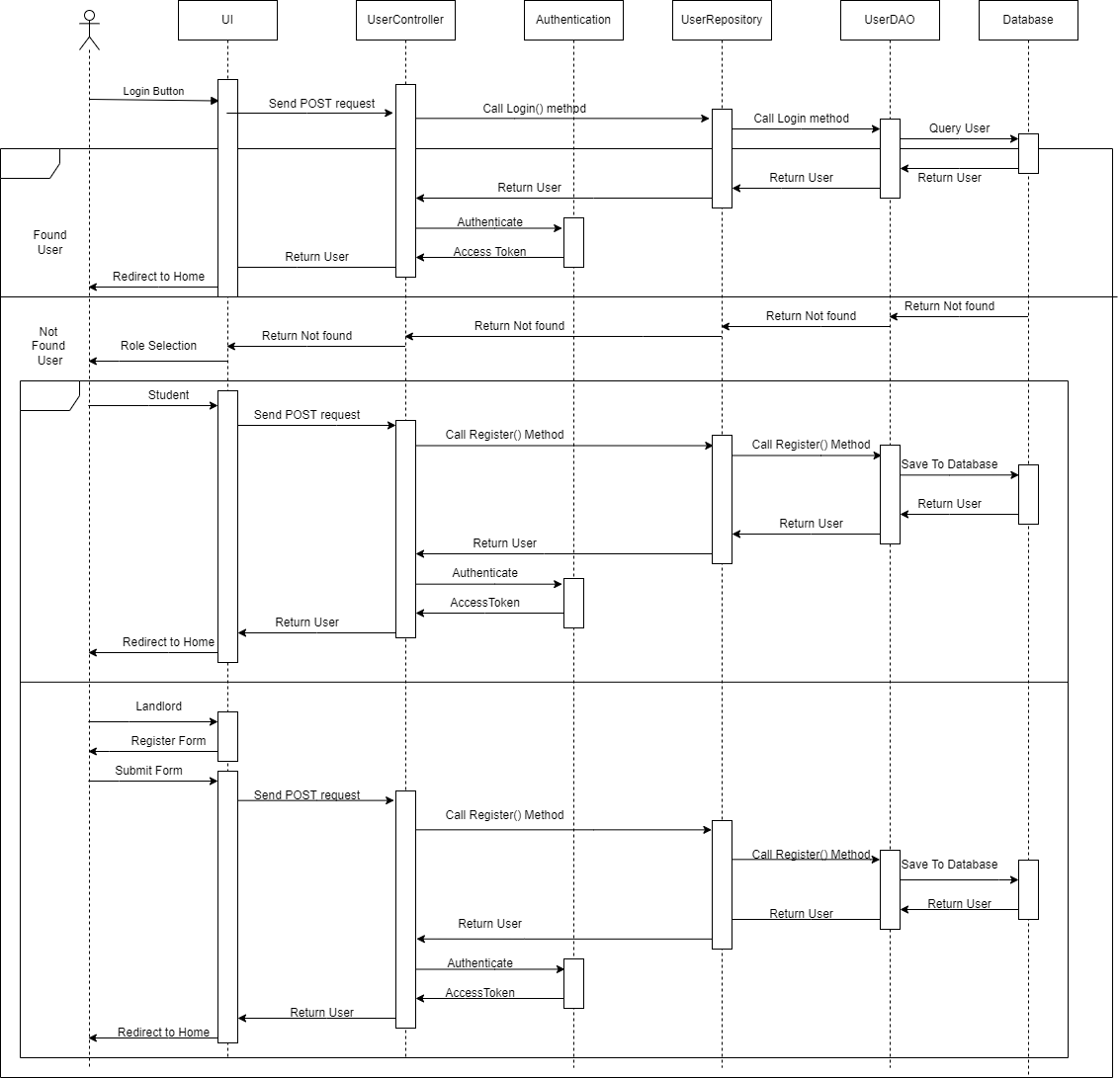
##### Authentication Class

This class create AccessToken for User

**Class Methods**

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

#### 4.9.3 Sequence Diagram



### 4.10 Landlord – Upload House Information

#### 4.10.1 Class Diagram

Graphical user interface, table

Description automatically generated

#### 4.10.2 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

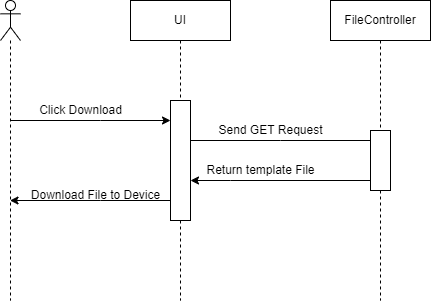
This class performs action to data (Read) in Village Table in Database

**Class Methods**

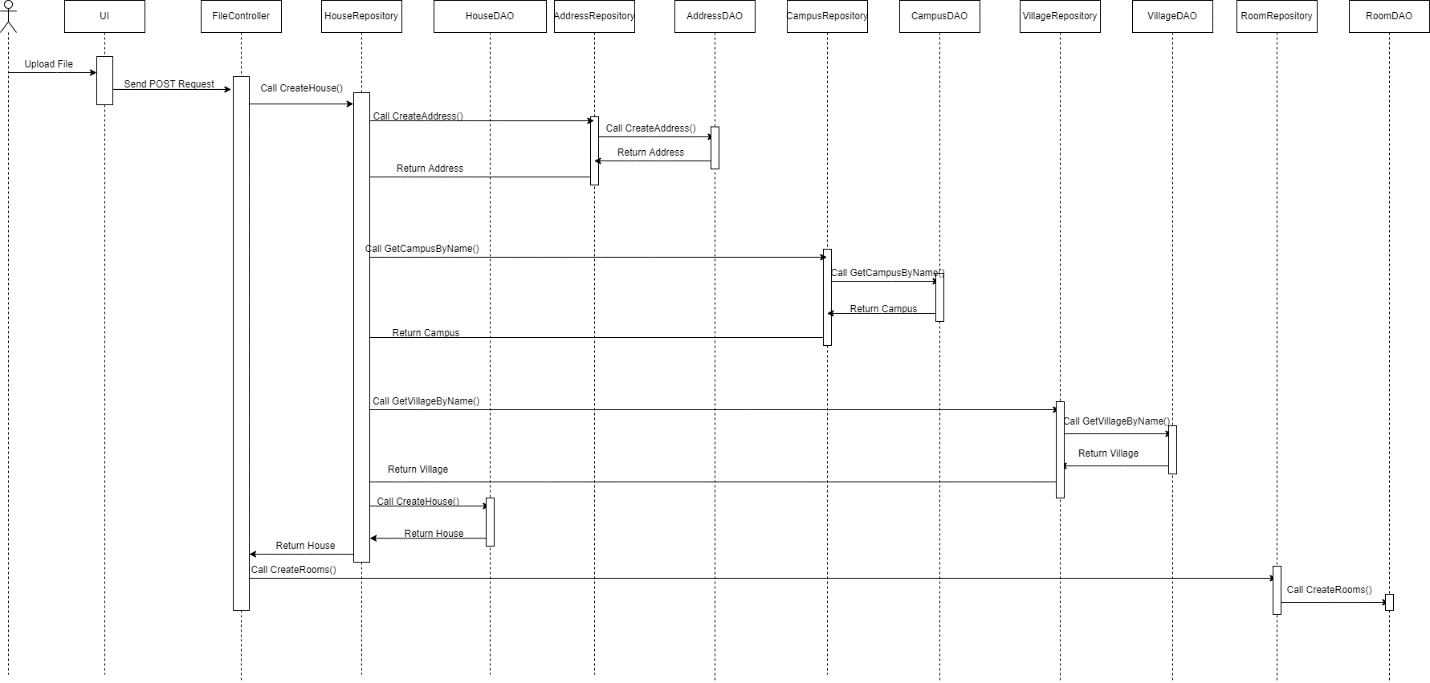
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetVillageByName() | Get Village by Village Name |
| 02 | GetVillageById() | Get Village by Village Id |

#### 4.10.3 Sequence Diagram

###### Download Template

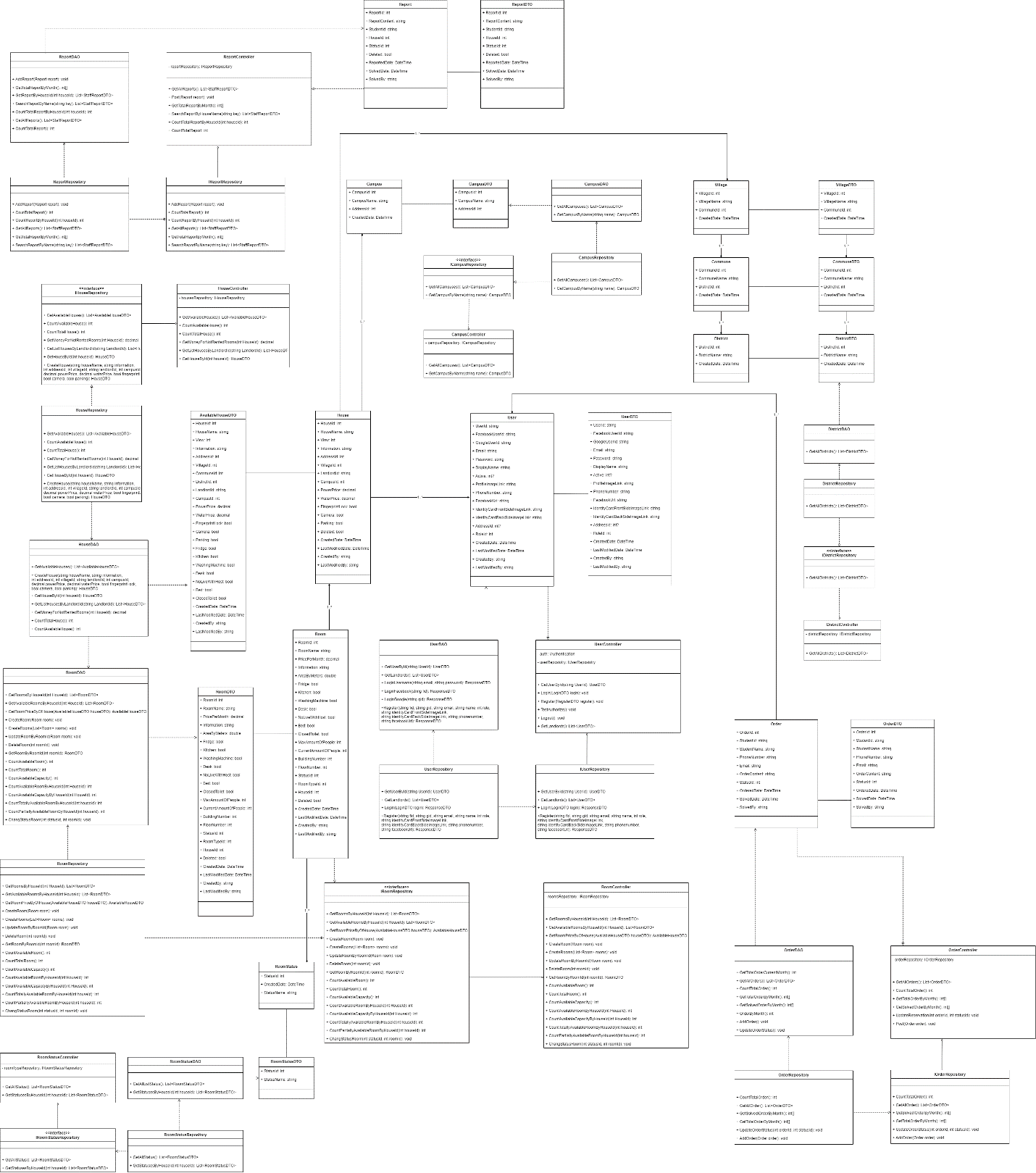


###### Upload File



### 4.11 Staff – Dashboard

#### 4.11.1 Class Diagram



#### 4.11.2 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 |

##### 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 |

##### OrderDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetTotalOrderCurrentMonth () | Get total order of the current month |
| 02 | GetAllOrders() | Get list of Orders in the database |
| 03 | CountTotalOrder() | Count total Orders |
| 04 | GetTotalOrderByMonth() | Get total Orders of the month |
| 05 | GetSolvedOrderByMonth() | Calculate number of solved Order by month |
| 06 | OrderByMonth() | Count Orders by month |

##### OrderController Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetTotalOrderCurrentMonth () | Get total order of the current month |
| 02 | GetAllOrders() | Get list of Orders in the database |
| 03 | CountTotalOrder() | Count total Orders |
| 04 | GetTotalOrderByMonth() | Get total Orders of the month |
| 05 | GetSolvedOrderByMonth() | Calculate number of solved Order by month |
| 06 | OrderByMonth() | Count Orders by month |

##### ReportDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | AddReport() | Add new Report to the database |
| 02 | GetTotalReportByMonth () | Count the total of Report by month |
| 03 | GetReportByHouseId() | Get list of Reports by house id |
| 04 | SearchReportByName() | Search Report by house's name |
| 05 | CountTotalReportByHouseId () | Count total Report by house id |
| 06 | GetAllReports() | Get all Reports |
| 07 | CountTotalReport() | Count total Report |

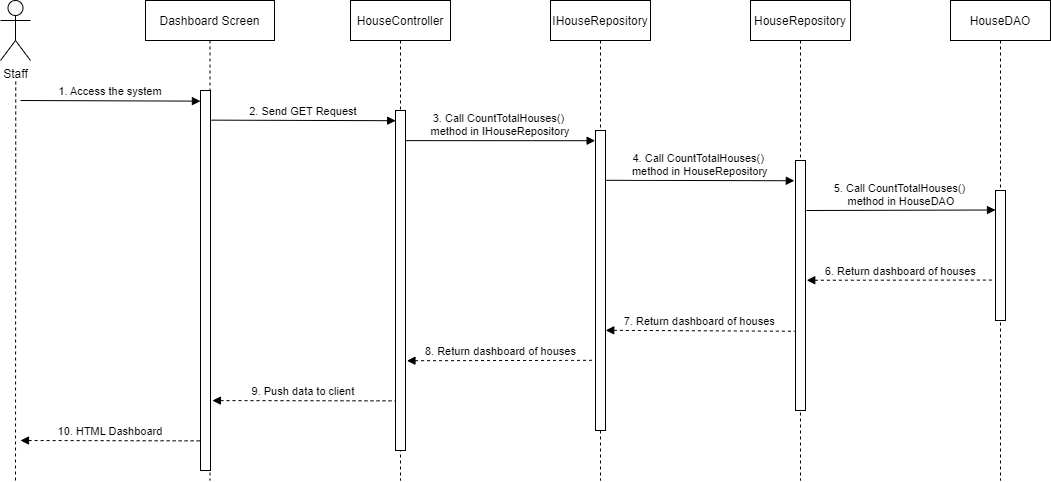
##### ReportController Class

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

**Class Methods**

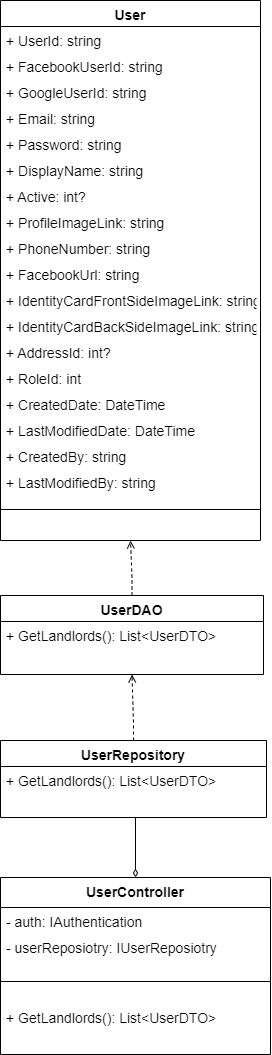
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | AddReport() | Add new Report to the database |
| 02 | GetTotalReportByMonth () | Count the total of Report by month |
| 03 | GetReportByHouseId() | Get list of Reports by house id |
| 04 | SearchReportByName() | Search Report by house's name |
| 05 | CountTotalReportByHouseId () | Count total Report by house id |
| 06 | GetAllReports() | Get all Reports |
| 07 | CountTotalReport() | Count total Report |

#### 4.11.3 Sequence Diagram



### 4.12 Staff – List Landlord

#### 4.12.1 Class Diagram



#### 4.12.2 Class Specifications

##### UserDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetLandlords() | Get list of Landlords in the database |

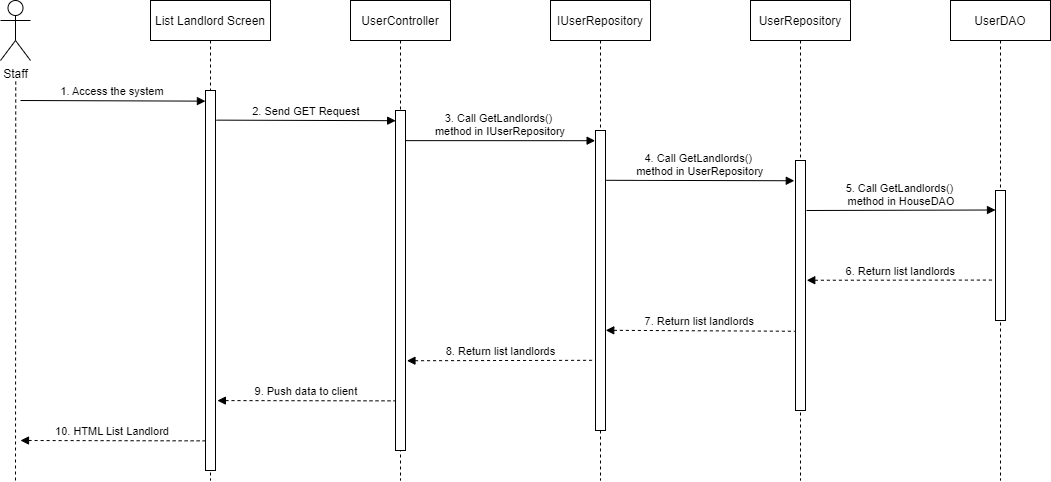
##### UserController Class

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

**Class Methods**

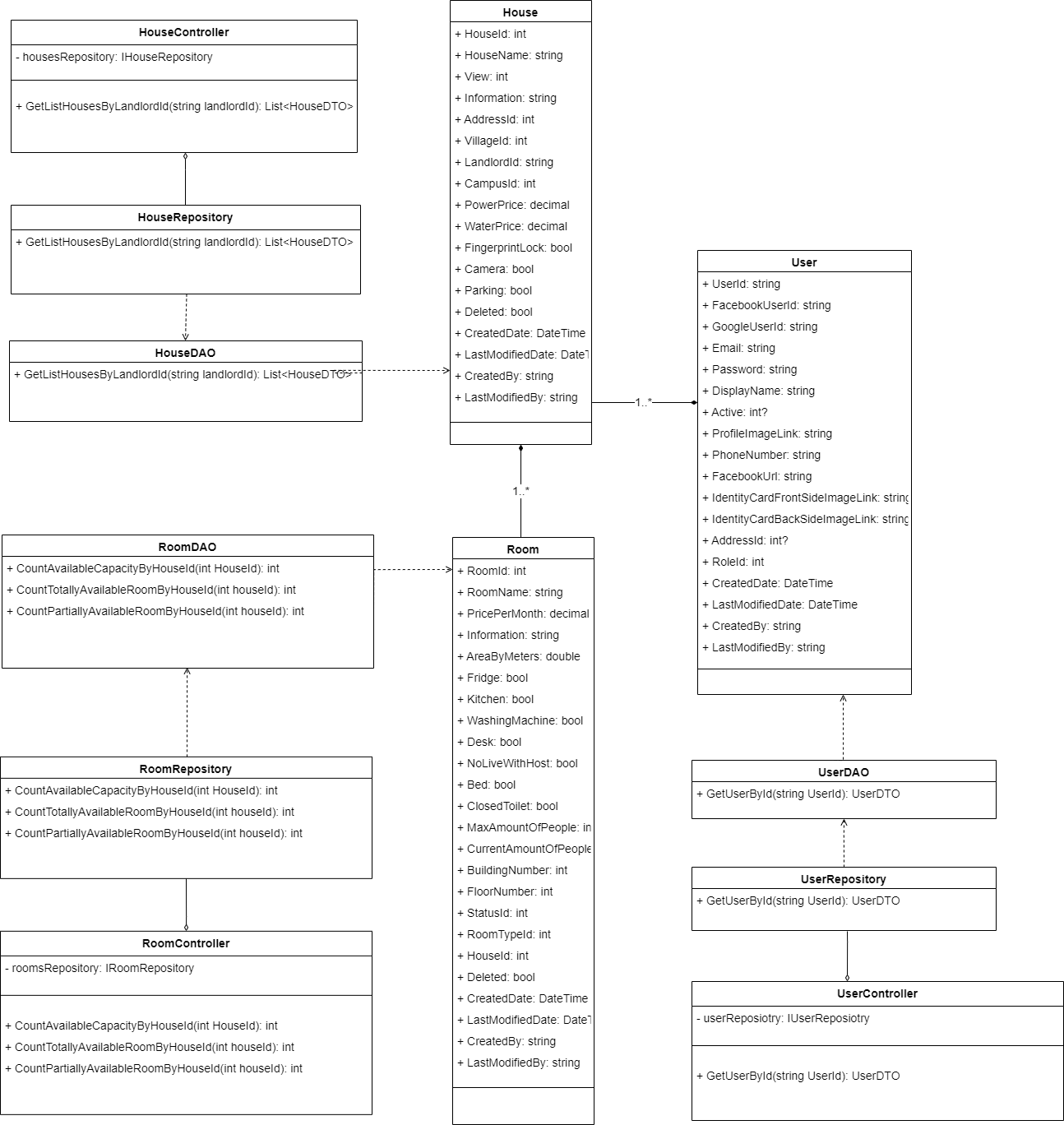
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetLandlords() | Get list of Landlords in the database |

#### 4.12.3 Sequence Diagram



### 4.13 Staff - Landlord Detail

#### 4.13.1 Class Diagram



#### 4.13.2 Class Specifications

##### HouseDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |

##### HouseController Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetListHousesByLandlordId() | Get list of Houses of 1 Landlord for him to manage |

##### RoomDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 2 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 3 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |

##### RoomController Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | CountAvailableCapacityByHouseId() | Count the sum of Capacity of all Rooms of 1 House |
| 2 | CountTotallyAvailableRoomByHouseId() | Count the sum of all available Rooms (totally only) of 1 House |
| 3 | CountPartiallyAvailableRoomByHouseId() | Count the sum of all available Rooms (partially only) of 1 House |

##### 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 |

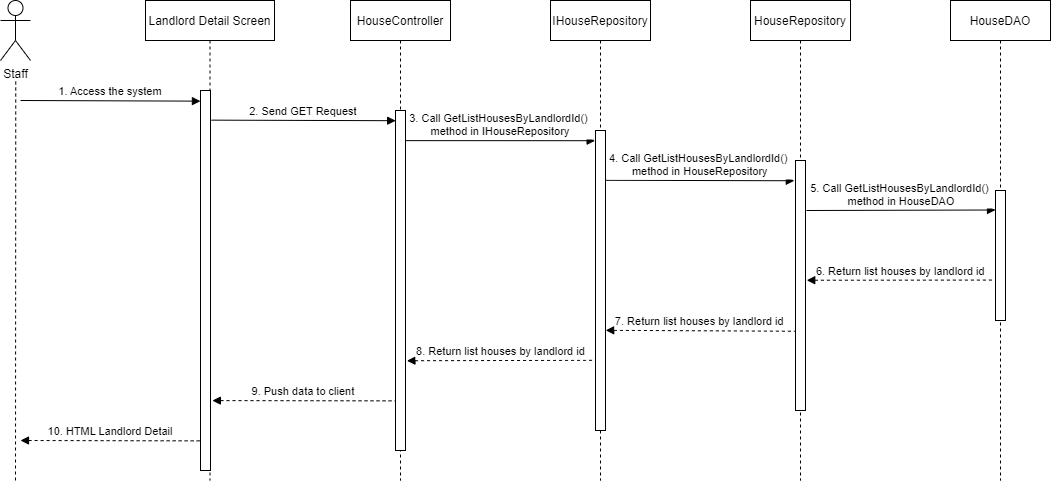
##### UserController Class

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

**Class Methods**

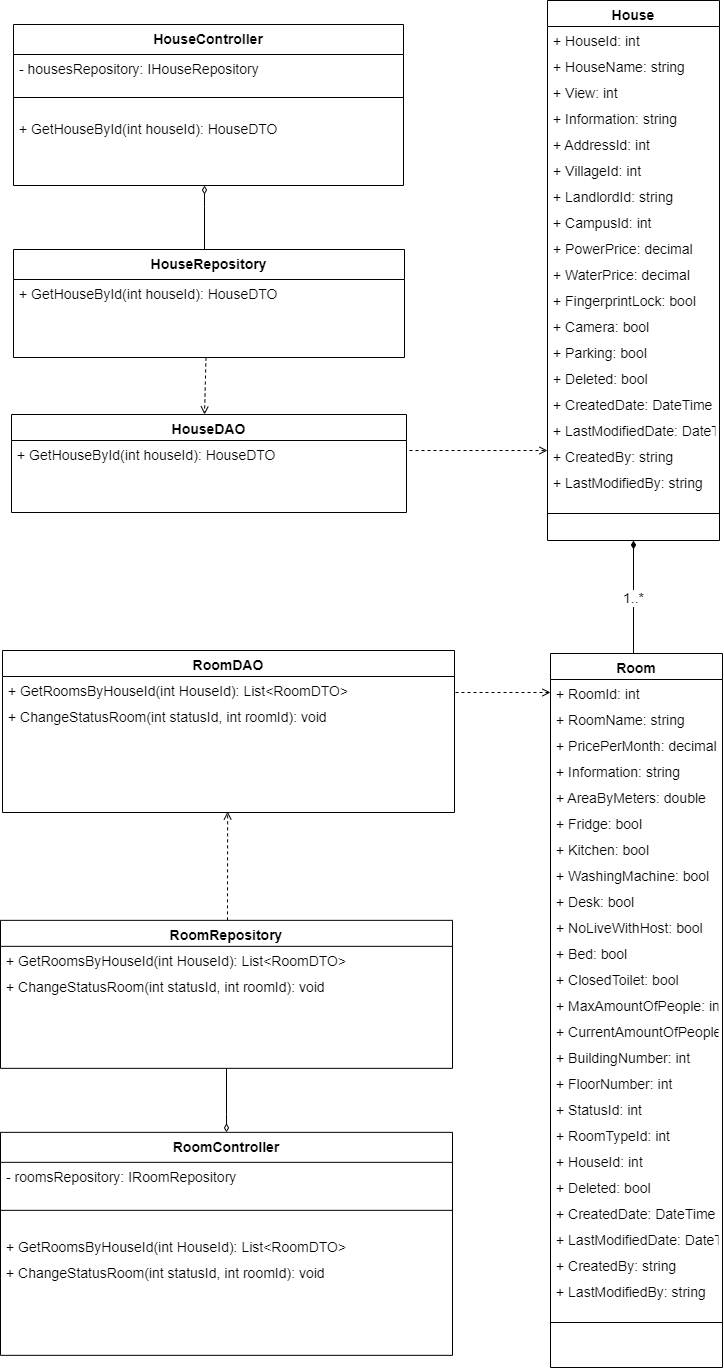
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetUserById () | Find detail information of User by its Id |

#### 4.13.3 Sequence Diagram



### 4.14 Staff - House Detail

#### 4.14.1 Class Diagram



#### 4.14.2 Class Specifications

##### HouseDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetHouseById() | Find detail information of a House by its Id |

##### HouseController Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetHouseById() | Find detail information of a House by its Id |

##### 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 | 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 | 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 |

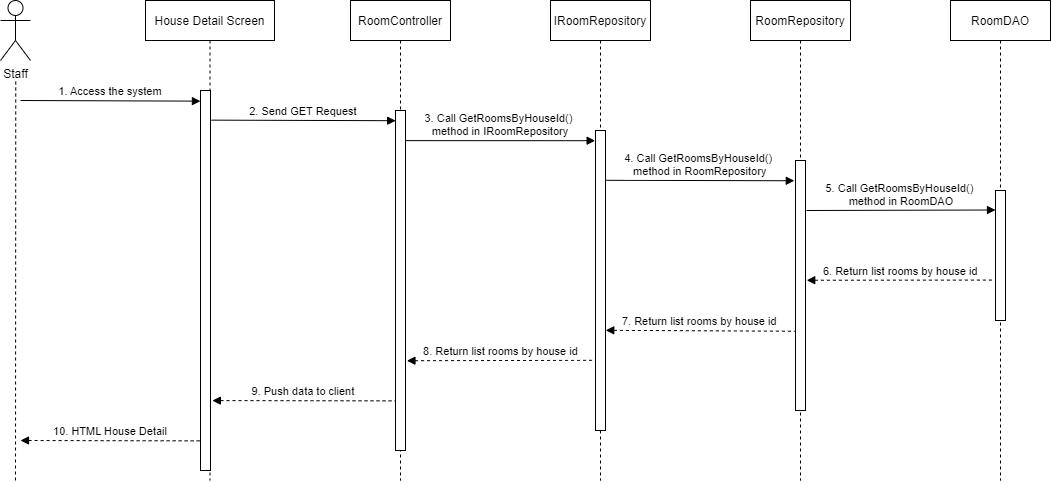
##### UserController Class

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 |

#### 4.14.3 Sequence Diagram



### 4.15 Staff - Room Detail

#### 4.15.1 Class Diagram



#### 4.15.2 Class Specifications

##### HouseDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetHouseById() | Find detail information of a House by its Id |

##### HouseController Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetHouseById() | Find detail information of a House by its Id |

##### RoomDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomByRoomId() | Find a Room by its Id |

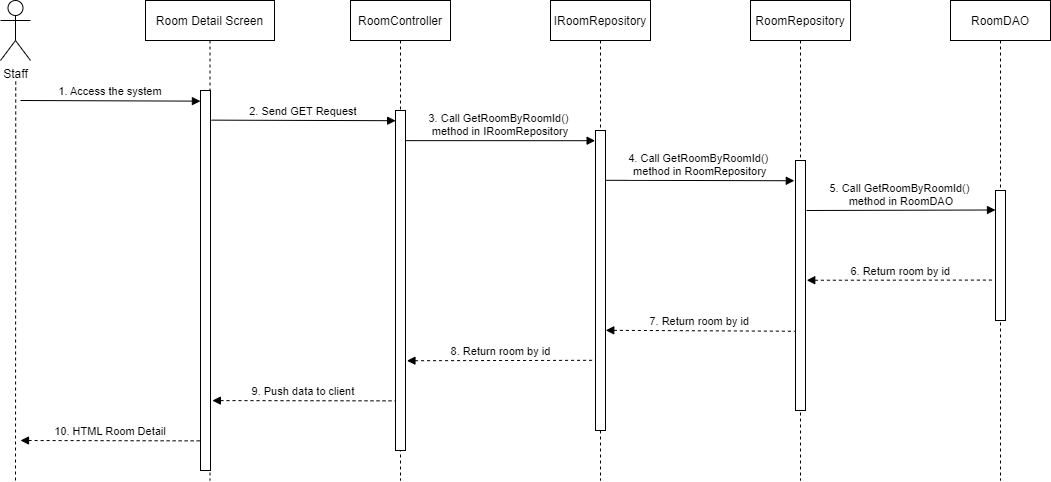
##### RoomController Class

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

**Class Methods**

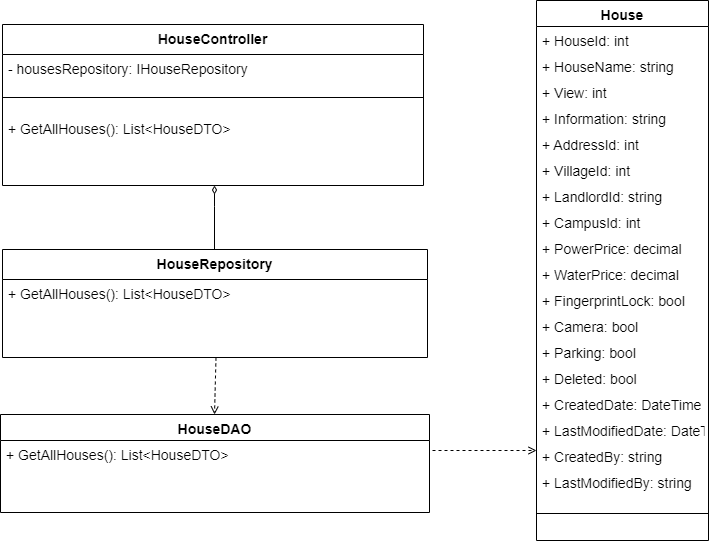
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetRoomByRoomId() | Find a Room by its Id |

#### 4.15.3 Sequence Diagram



### 4.16 Staff – List House

#### 4.16.1 Class Diagram



#### 4.16.2 Class Specifications

##### HouseDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAllHouses() | Return list of all Houses in the system |

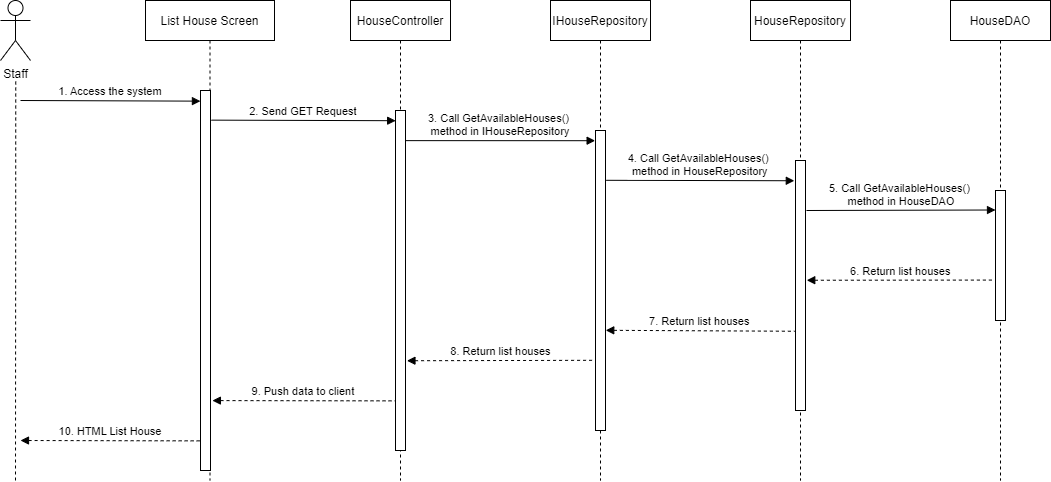
##### HouseController Class

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

**Class Methods**

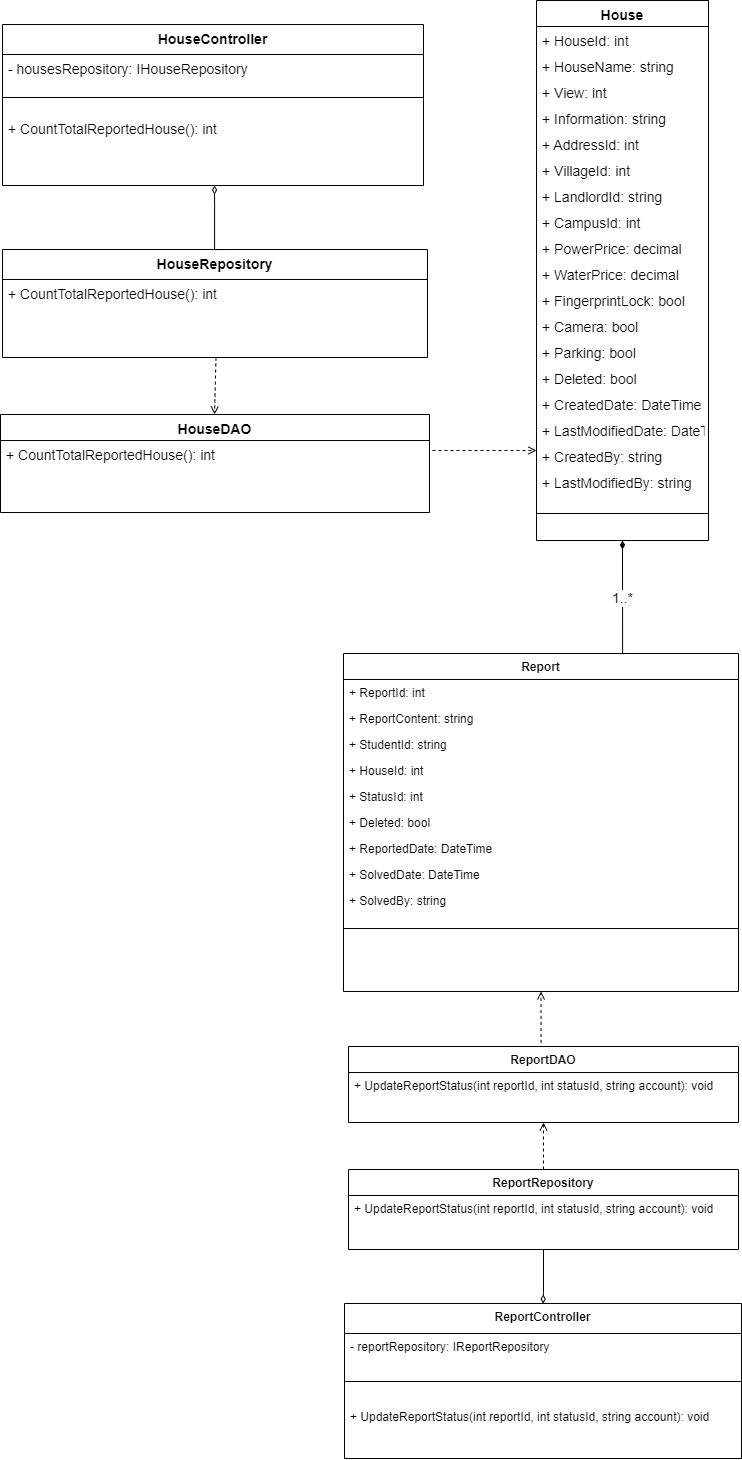
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetAllHouses() | Return list of all Houses in the system |

#### 4.16.3 Sequence Diagram



### 4.17 Staff – List Report

#### 4.17.1 Class Diagram



#### 4.17.2 Class Specifications

##### HouseDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | CountTotalReportedHouse() | Count total number of reported Houses in the system |

##### HouseController Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | CountTotalReportedHouse() | Count total number of reported Houses in the system |

##### ReportDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | UpdateReportStatus() | Update status of Report to the database |

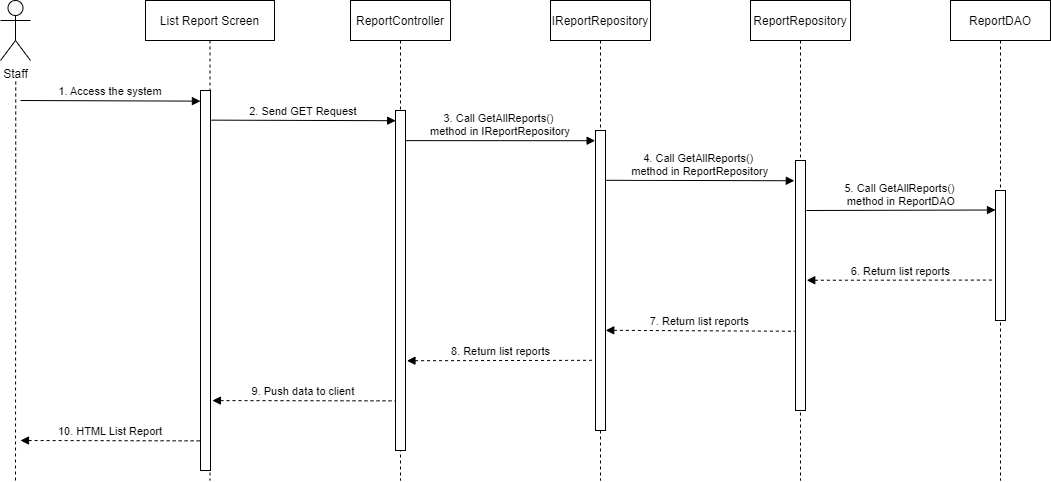
##### ReportController Class

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

**Class Methods**

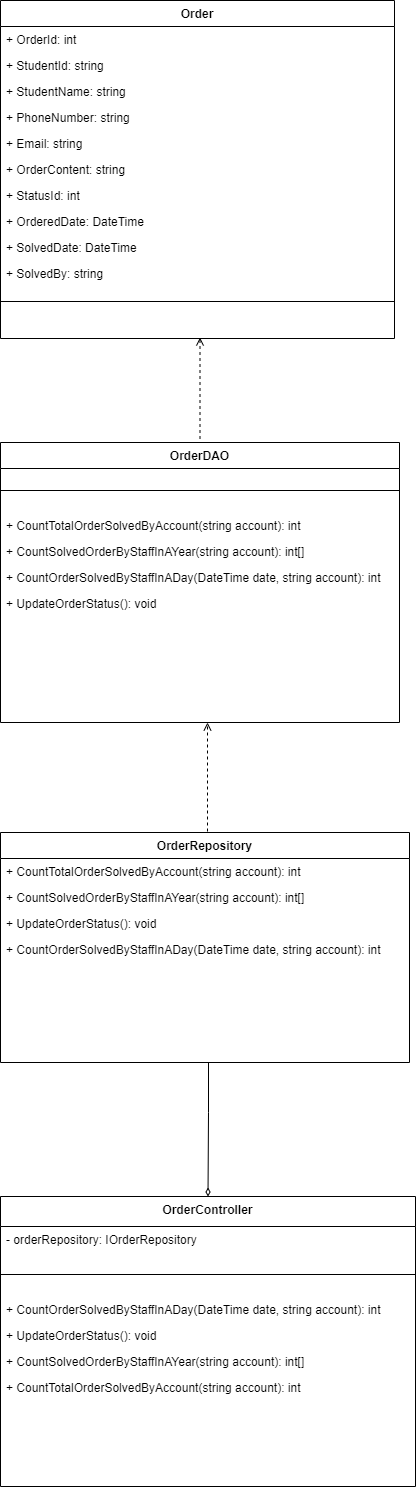
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | UpdateReportStatus() | Update status of Report to the database |

#### 4.17.3 Sequence Diagram



### 4.18 Staff – List Order

#### 4.18.1 Class Diagram



#### 4.18.2 Class Specifications

##### OrderDAO Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | CountOrderSolvedByStaffInADay () | Count total Orders have solved by staff in a day in the database |
| 02 | CountSolvedOrderByStaffInAYear () | Count total Orders have solved by staff in a year in the database |
| 03 | CountTotalOrderSolvedByAccount () | Count total Orders have solved by account in the database |
| 04 | UpdateOrderStatus() | Update status of Order to the database |

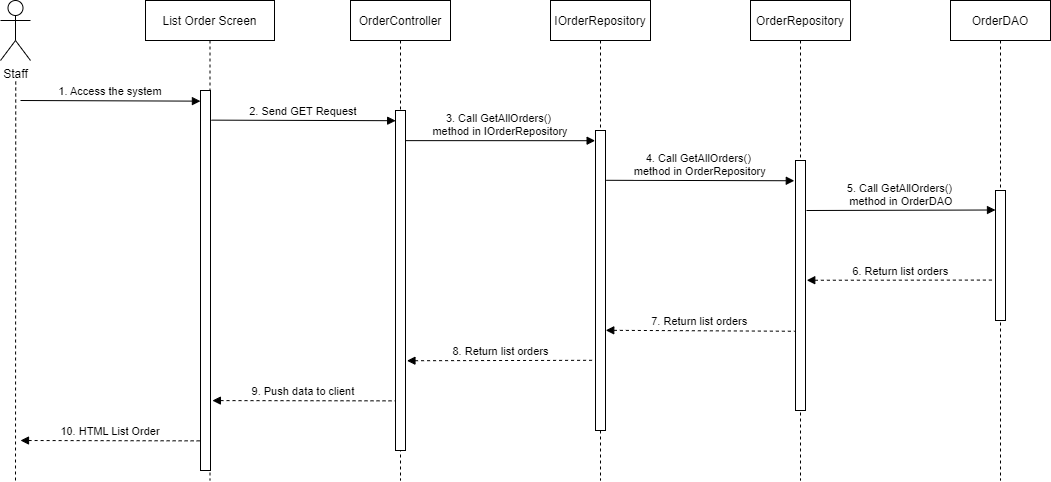
##### OrderController Class

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

**Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | CountOrderSolvedByStaffInADay () | Count total Orders have solved by staff in a day in the database |
| 02 | CountSolvedOrderByStaffInAYear () | Count total Orders have solved by staff in a year in the database |
| 03 | CountTotalOrderSolvedByAccount () | Count total Orders have solved by account in the database |
| 04 | UpdateOrderStatus() | Update status of Order to the database |

#### 4.18.3 Sequence Diagram



### 4.19 Staff – Change Password

#### 4.19.1 Class Diagram

Diagram

Description automatically generated

#### 4.19.2 Class Specification

##### 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 |
| 07 | ChangePassword() | Staff change password |

##### 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 |
| 07 | ChangePassword() | Staff change password |

##### 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 |
| 07 | ChangePassword() | Staff change password |

#### 4.19.3 Sequence Diagram

Diagram

Description automatically generated