

**Capstone Project Report**

**Report 4 – Software Design Document**

– Hanoi, Sep 2022 –

**Table of Contents**

[I. Record of Changes 3](#_Toc118751909)

[II. Software Design Document 4](#_Toc118751910)

[1. System Design 4](#_Toc118751911)

[1.1 System Architecture 4](#_Toc118751912)

[1.2 Package Diagram 5](#_Toc118751913)

[2. Database Design 6](#_Toc118751914)

[3. Detailed Design 7](#_Toc118751915)

[3.1 Home Page 7](#_Toc118751916)

[3.2 <Feature/Function Name2> 8](#_Toc118751917)

[4. Database Tables 8](#_Toc118751918)

[1. Campuses 8](#_Toc118751919)

[2. UserRoles 8](#_Toc118751920)

[3. Addresses 9](#_Toc118751921)

[4. Users 9](#_Toc118751922)

[5. Houses 10](#_Toc118751923)

[6. Rooms 10](#_Toc118751924)

[7. Rates 11](#_Toc118751925)

[8. ImagesOfHouse 11](#_Toc118751926)

[9. ImagesOfRoom 11](#_Toc118751927)

[10. Districts 12](#_Toc118751928)

[11. Communes 12](#_Toc118751929)

[12. Villages 12](#_Toc118751930)

[13. Statuses 12](#_Toc118751931)

[14. RoomTypes 13](#_Toc118751932)

[15. Reports 13](#_Toc118751933)

[16. RoomHistories 13](#_Toc118751934)

[17. Issues 13](#_Toc118751935)

# I. Record of Changes

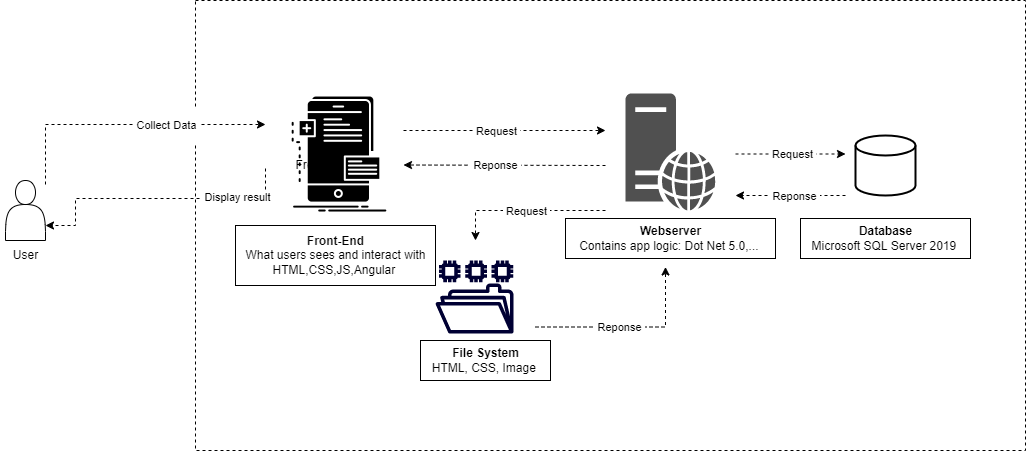
|  |  |  |  |
| --- | --- | --- | --- |
| Date | A\* M, D | In charge | Change Description |
| 08/10/2022 | A | KienNT | Database Tables |
| 09/10/2022 | M | GiangNT | Database Tables |
| 09/10/2022 | A | ThongPQ | System Architecture |
| 12/10/2022 | A | KienNT | Package Diagram, Package Description |
| 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 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

\*A - Added M - Modified D - Deleted

# II. Software Design Document

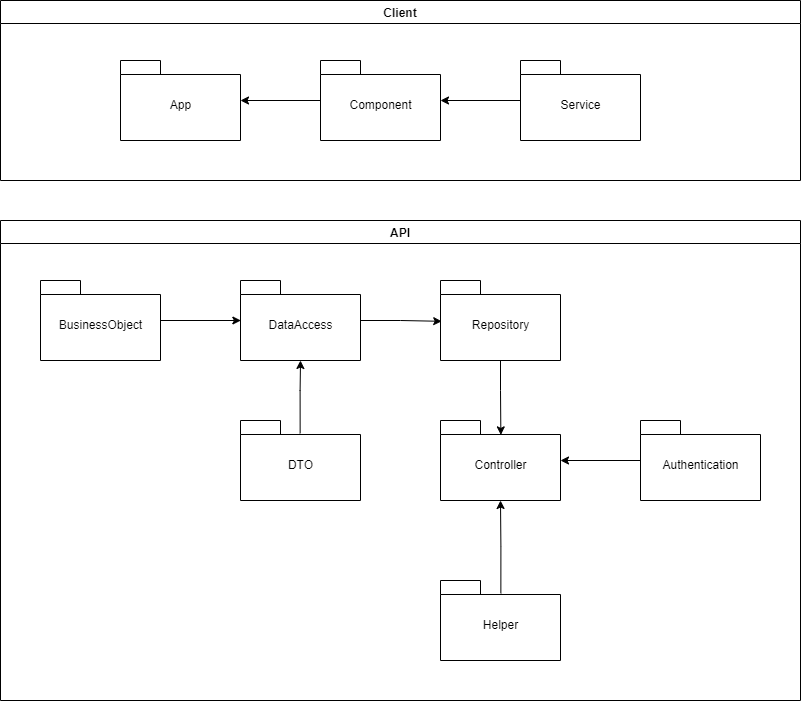
## 1. System Design

### 1.1 System Architecture



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

### 1.2 Package Diagram



***Package Descriptions***

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

## 2. Database Design

*[Provide the files description, database table relationship & table descriptions like example below]*



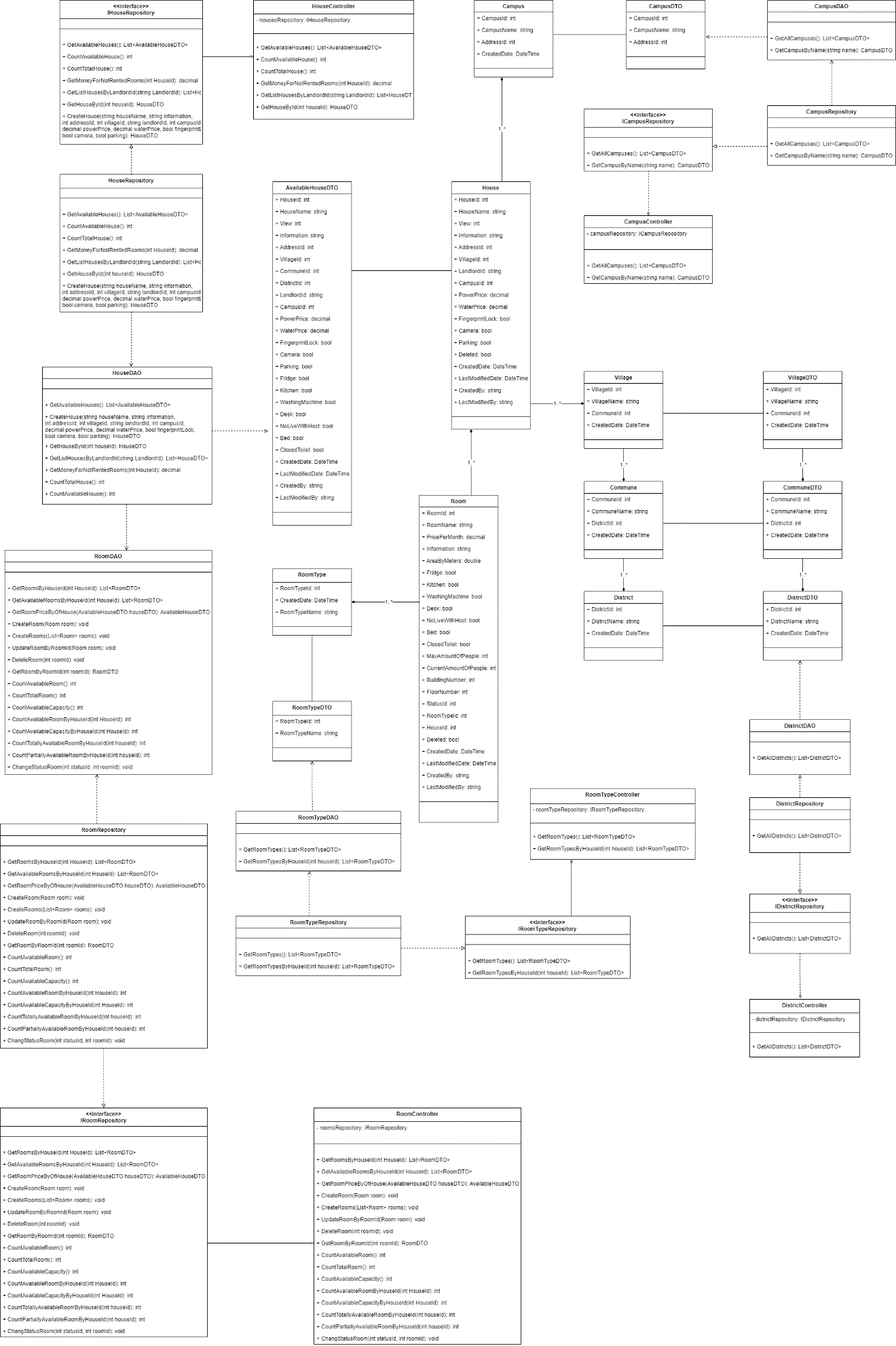
***Table Descriptions***

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| *01* | *<Table name>* | *<Description of the table>*  *- Primary keys: <<list of primary key fields>>*  *- Foreign keys: <<list of foreign key fields>>* |
| *02* | *<Table name2>* | *…* |

## 3. Detailed Design

### 3.1 Home Page

#### 3.1.1 Class Diagram



#### 3.1.2 Class Specifications

##### 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 |

##### HouseController Class

This class 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 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 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 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 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 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 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 |

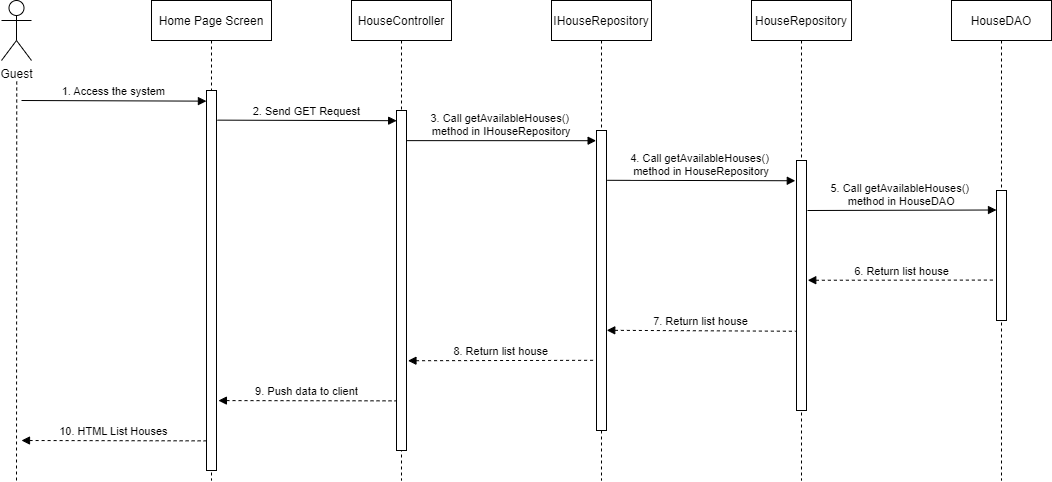
##### RoomDAO Class

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

**Class Methods**

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

#### 3.1.3 Sequence Diagram



#### 3.1.3 <Sequence Diagram Name2>

#### 3.1.4 …

### 3.2 <Feature/Function Name2>

…

## 4. Database Tables

### 1. Campuses

*Store the information of all campus for users*

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

### 2. UserRoles

*Store the information of all roles of users*

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

### 3. Addresses

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

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

### 4. Users

*Store the information of all users*

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

### 5. Houses

*Store the information of all houses*

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

### 6. Rooms

*Store the information of all rooms of houses*

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

### 7. Rates

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

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

### 8. ImagesOfHouse

*Store the information of all pictures of house*

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

### 9. ImagesOfRoom

*Store the information of all pictures of room*

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

### 10. Districts

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

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

### 11. Communes

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

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

### 12. Villages

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

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

### 13. Statuses

*Store the information of all status of a Room*

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

### 14. RoomTypes

*Store the information of all types of a Room*

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

### 15. Reports

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

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

### 16. RoomHistories

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

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

### 17. Issues

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

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