

**<<Cyber Booking Management >>**

**Software Design Document**

**Members:**

**- Trần Lê Minh Hoàng**

**- Huỳnh Thái Phong**

**- Ngô Nguyễn Xuân Linh**

**- Lê Thành Nhân**

– Da Nang, September 2021 –

**Table of Contents**

* [I. Overview 6](#_Toc87251361)

[1. Code Packages/Namespaces 6](#_Toc87251362)

[2. Database Schema 6](#_Toc87251363)

[II. Code Designs 8](#_Toc87251364)

[1. Authorization / Login 8](#_Toc87251365)

[a. Class Diagram 8](#_Toc87251366)

[b. Class Specifications 8](#_Toc87251367)

[c. Sequence Diagram(s) 10](#_Toc87251368)

[d. Database queries 10](#_Toc87251369)

[2. Authorization / Register 11](#_Toc87251370)

[a. Class Diagram 11](#_Toc87251371)

[b. Class Specifications 11](#_Toc87251372)

[c. Sequence Diagram(s) 12](#_Toc87251373)

[d. Database queries 12](#_Toc87251374)

[3. User / Update Profile 13](#_Toc87251375)

[a. Class Diagram 13](#_Toc87251376)

[b. Class Specifications 13](#_Toc87251377)

[c. Sequence Diagram(s) 14](#_Toc87251378)

[d. Database queries 14](#_Toc87251379)

[4. Recharge / Recharge & CreateRechargeReceipt 14](#_Toc87251380)

[a. Class Diagram 14](#_Toc87251381)

[b. Class Specifications 15](#_Toc87251382)

[c. Sequence Diagram(s) 15](#_Toc87251383)

[d. Database queries 15](#_Toc87251384)

[5. Recharge / Paid 16](#_Toc87251385)

[a. Class Diagram 16](#_Toc87251386)

[b. Class Specifications 16](#_Toc87251387)

[c. Sequence Diagram(s) 16](#_Toc87251388)

[d. Database queries 16](#_Toc87251389)

[6. Cyber / View Cyber List 17](#_Toc87251390)

[a. Class Diagram 17](#_Toc87251391)

[b. Class Specifications 17](#_Toc87251392)

[c. Sequence Diagram(s) 17](#_Toc87251393)

[d. Database queries 18](#_Toc87251394)

[7. Cyber / View Cyber Detail 18](#_Toc87251395)

[a. Class Diagram 18](#_Toc87251396)

[b. Class Specifications 18](#_Toc87251397)

[c. Sequence Diagram(s) 18](#_Toc87251398)

[d. Database queries 19](#_Toc87251399)

[8. Cyber/ Create Cyber 19](#_Toc87251400)

[a. Class Diagram 19](#_Toc87251401)

[b. Class Specifications 19](#_Toc87251402)

[c. Sequence Diagram(s) 20](#_Toc87251403)

[d. Database queries 20](#_Toc87251404)

[9. Cyber/ Update Cyber 21](#_Toc87251405)

[a. Class Diagram 21](#_Toc87251406)

[b. Class Specifications 21](#_Toc87251407)

[c. Sequence Diagram(s) 21](#_Toc87251408)

[d. Database queries 22](#_Toc87251409)

[10. Service / View Service Detail 22](#_Toc87251410)

[a. Class Diagram 22](#_Toc87251411)

[b. Class Specifications 22](#_Toc87251412)

[c. Sequence Diagram(s) 23](#_Toc87251413)

[d. Database queries 23](#_Toc87251414)

[11. Service/ Create Service 23](#_Toc87251415)

[a. Class Diagram 23](#_Toc87251416)

[b. Class Specifications 24](#_Toc87251417)

[c. Sequence Diagram(s) 24](#_Toc87251418)

[d. Database queries 25](#_Toc87251419)

[12. Service/ Update Service 25](#_Toc87251420)

[a. Class Diagram 25](#_Toc87251421)

[b. Class Specifications 25](#_Toc87251422)

[c. Sequence Diagram(s) 26](#_Toc87251423)

[d. Database queries 26](#_Toc87251424)

[13. Booking/ Book (Create BookingReceipt) 26](#_Toc87251425)

[a. Class Diagram 26](#_Toc87251426)

[b. Class Specifications 26](#_Toc87251427)

[c. Sequence Diagram(s) 27](#_Toc87251428)

[d. Database queries 27](#_Toc87251429)

[14. Search/ SearchCyber 28](#_Toc87251430)

[a. Class Diagram 28](#_Toc87251431)

[b. Class Specifications 28](#_Toc87251432)

[c. Sequence Diagram(s) 29](#_Toc87251433)

[d. Database queries 29](#_Toc87251434)

[15. Search/ SearchBookingReceipt 30](#_Toc87251435)

[a. Class Diagram 30](#_Toc87251436)

[b. Class Specifications 30](#_Toc87251437)

[c. Sequence Diagram(s) 31](#_Toc87251438)

[d. Database queries 31](#_Toc87251439)

[16. Receipt/ ViewReceipt 32](#_Toc87251440)

[a. Class Diagram 32](#_Toc87251441)

[b. Class Specifications 32](#_Toc87251442)

[c. Sequence Diagram(s) 33](#_Toc87251443)

[d. Database queries 33](#_Toc87251444)

[17. Delete User & Manager 33](#_Toc87251445)

[a. Class Diagram 33](#_Toc87251446)

[b. Class Specifications 33](#_Toc87251447)

[c. Sequence Diagram(s) 34](#_Toc87251448)

[d. Database queries 34](#_Toc87251449)

[18. Service/Delete Service 34](#_Toc87251450)

[a. Class Diagram 34](#_Toc87251451)

[b. Class Specifications 34](#_Toc87251452)

[c. Sequence Diagram(s) 35](#_Toc87251453)

[d. Database queries 35](#_Toc87251454)

[19. Cyber/Delete Cyber 35](#_Toc87251455)

[a. Class Diagram 35](#_Toc87251456)

[b. Class Specifications 36](#_Toc87251457)

[c. Sequence Diagram(s) 36](#_Toc87251458)

[d. Database queries 36](#_Toc87251459)

[20. Authorization / Remember account 36](#_Toc87251460)

[a. Class Diagram 36](#_Toc87251461)

[b. Class Specifications 36](#_Toc87251462)

[c. Sequence Diagram(s) 37](#_Toc87251463)

[d. Database queries 38](#_Toc87251464)

[21. Filter / Filter 38](#_Toc87251465)

[a. Class Diagram 38](#_Toc87251466)

[b. Class Specifications 38](#_Toc87251467)

[c. Sequence Diagram(s) 38](#_Toc87251468)

[d. Database queries 38](#_Toc87251469)

[22. Admin / Web’s statistic 39](#_Toc87251470)

[a. Class Diagram 39](#_Toc87251471)

[b. Class Specifications 39](#_Toc87251472)

[c. Sequence Diagram(s) 39](#_Toc87251473)

[d. Database queries 39](#_Toc87251474)

[III. Database Tables 40](#_Toc87251475)

[1. User 40](#_Toc87251476)

[2. Manager 40](#_Toc87251477)

[3. Cyber 40](#_Toc87251478)

[4. Service 41](#_Toc87251479)

[5. Wallet 41](#_Toc87251480)

[6. Payment 41](#_Toc87251481)

[7. Recharge\_Receipt 41](#_Toc87251482)

[8. Booking\_Receipt 41](#_Toc87251483)

[9. User\_Favorite 42](#_Toc87251484)

[10. City 42](#_Toc87251485)

[11. Admin\_Notification 42](#_Toc87251486)

# I. Overview

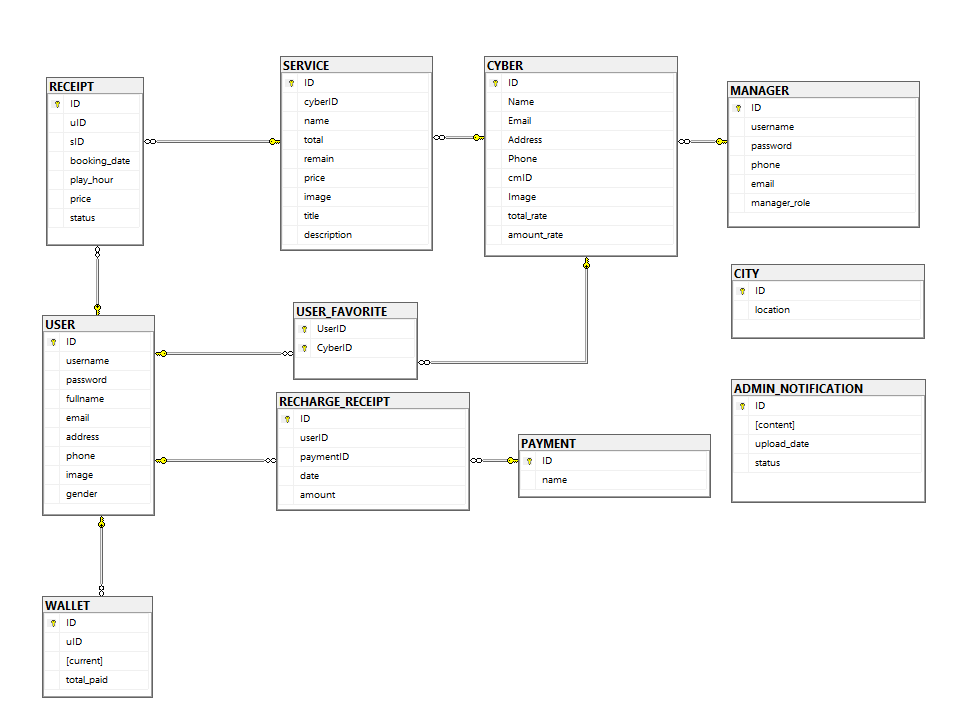
## 1. Code Packages/Namespaces



***Package descriptions & package class naming conventions***

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| 01 | DBContext | Connect database |
| 02 | Models | Contain entities |
| 03 | Controller | Contain Controllers |
| 04 | DAO | Forward data from database to Controller.  Handle information. |
| 05 | Filter | Forward page |
| 06 | View | Contain html file, css file, js file, … |

## 2. Database Schema



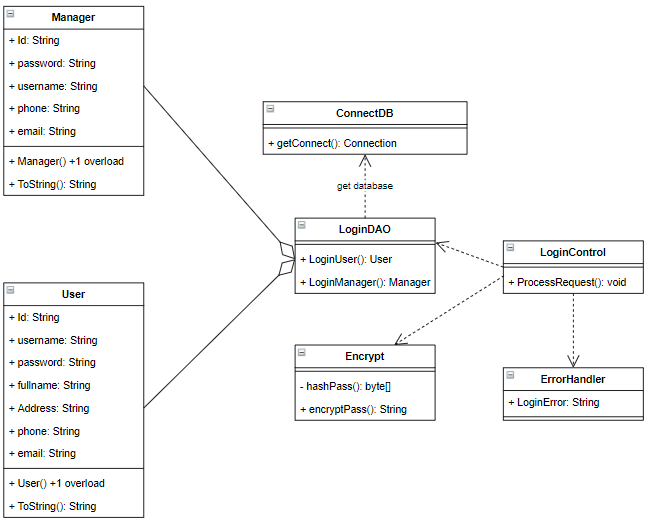
***Table descriptions & package class naming conventions are as below***

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| 01 | USER | - Contain information of Users  - Primary keys: [Id] |
| 02 | MANAGER | - Contain information of Manager, be distinguish by [Role]  - Primary keys: [Id] |
| 03 | WALLET | - Contain wallet information of each User  - Primary keys: [Id]  - Foreign keys: [uID] = [USER.Id] |
| 04 | CYBER | - Contain Cybers Information  - Primary keys: [Id]  - Foreign keys: [cmID] = [MANAGER].[Id] on delete cascade |
| 05 | SERVICE | - Contain Service Information  - Primary keys: [Id]  - Foreign keys: [cyberId] = [CYBER].[ID] on delete cascade |
| 06 | RECEIPT | - Contain receipt information after user books a service  - Primary keys: [Id]  - Foreign keys: [userId] = [USER].[Id] on delete cascade |
| 07 | USER\_FAVORITE | - Contain all Cyber that is user likes  - Keys: [uId, cID]  - Foreign keys: [cId] = [CYBER].[Id],  [uId] = [USER].[Id] |
| 08 | PAYMENT | - Contain payment method  - Primary keys: [Id] |
| 09 | RECHARGE\_RECEIPT | - Contain receipt information after User recharge  - Primary keys: [Id]  - Foreign keys: [userId] = [USER].[Id] on delete cascade |
| 10 | CITY | - Contain city  - Primary keys: [Id] |
| 11 | ADMIN\_NOTIFICATION | - Contain admin notification  - Primary keys: [Id] |

# II. Code Designs

## 1. Authorization / Login

### a. Class Diagram



### b. Class Specifications

#### User

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | String ToString() | -Print attribute of User in console. |

#### Manager

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | String ToString() | Print attribute of Manager in console. |

#### ConnectDB

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void getConnect() | Connect to database |

#### LoginDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | User LoginUser() | - Check if User existed in database. If existed, return that User.  - Input: Username : String,  Password : String.  - Output: new User(Id, username, password, fullname, address, phone, email); |
| 02 | Manager LoginManager() | - Check if Manager existed in database. If existed, return that User.  - Input: Username : String,  Password : String.  - Output: new Manager (Id, username, password, address, phone, email); |

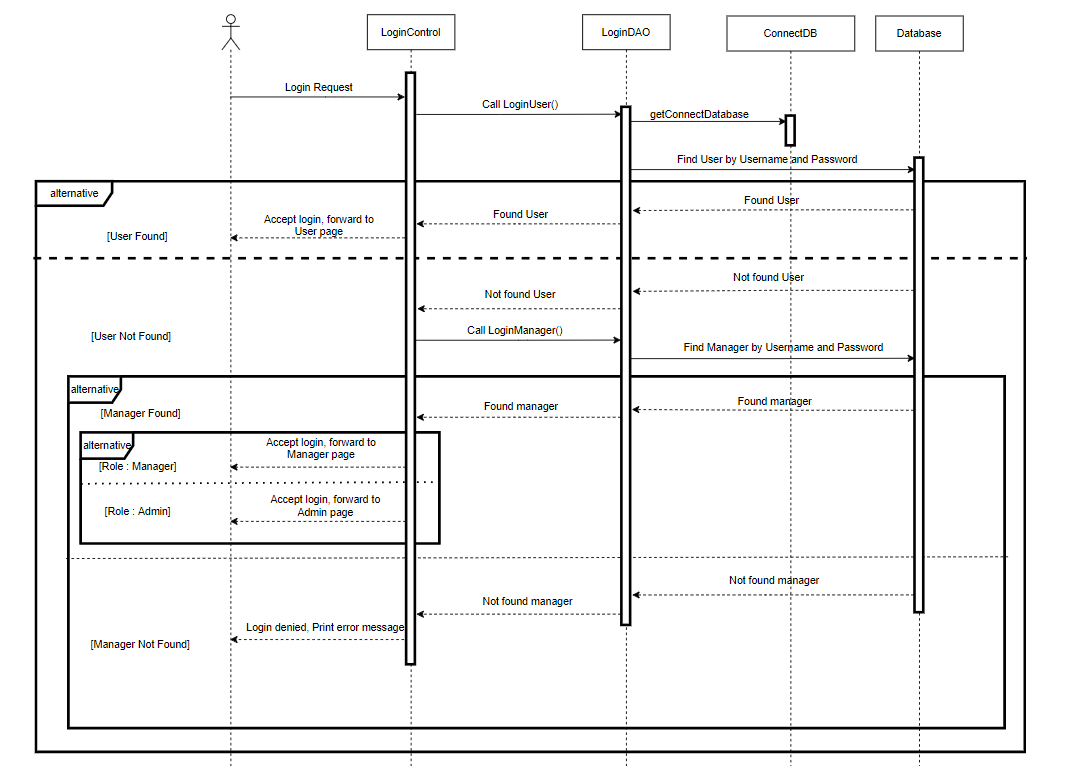
#### Encrypt

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | byte[] Hashpass() | - Hash password to char array  - Input: password : char[], salt : byte, iteration : int, keyLength : int.  - Output: array of char after hash password. |
| 02 | String EncryptPass() | - Encrypt password  - Input : password: String();  - Output: String of encrypted password. |

#### LoginControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - Get username and password from jsp page and check if this user is existed in database, if existed forward to home page, if not, display error from Errorhandler class.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

### c. Sequence Diagram(s)

**

### d. Database queries

* Get User by Username and password:

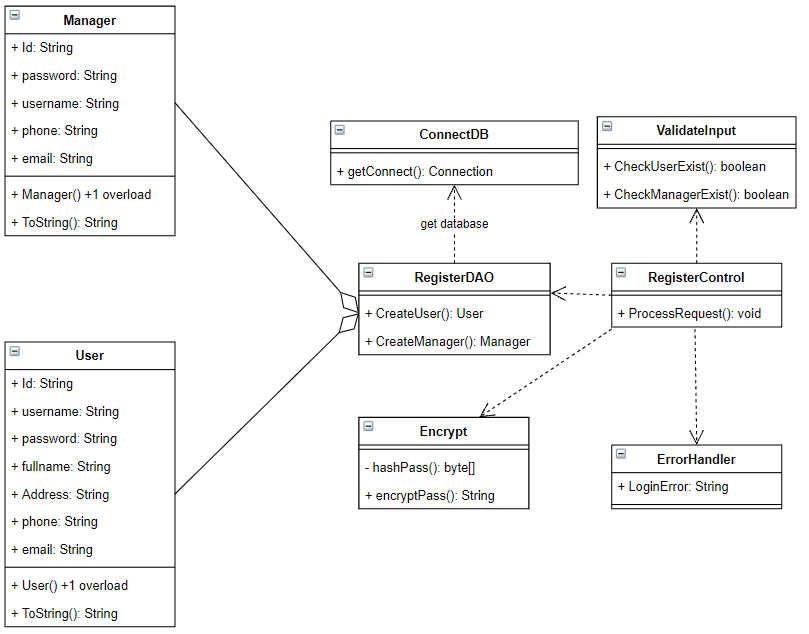


* Get Manager by Username and password:



## 2. Authorization / Register

### a. Class Diagram



### b. Class Specifications

#### RegisterDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void CreateUser() | - Create new User in database.  - Input: fullname : String, username : String, password : String |
| 02 | void CreateManager() | - Create new Manager in database.  - Input: fullname : String, username : String, password : String |

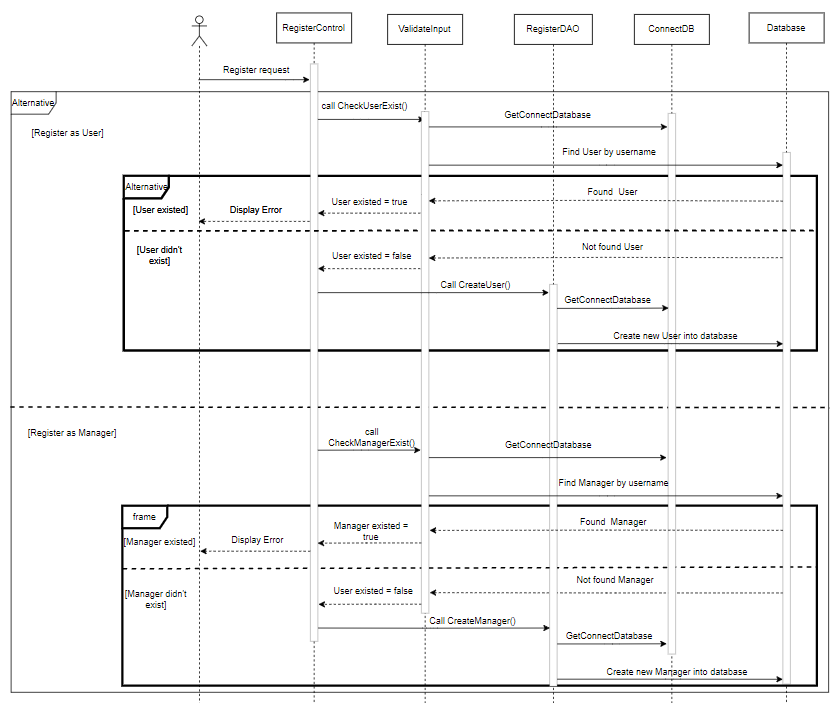
#### RegisterControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - Get information from jsp page and call CreateUser() or CreateManager() to create new user or manager in database.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

#### ValidateInput

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Bool checkUserExist() | - Check if there is any user in database has username like input username  - Input: username : String;  - Output: if not exist -> true, else -> false |
| 01 | Bool checkManagerExist() | - Check if there is any manager in database has username like input username  - Input: username : String;  - Output: if not exist -> true, else -> false |

### c. Sequence Diagram(s)



### d. Database queries

* Create new User

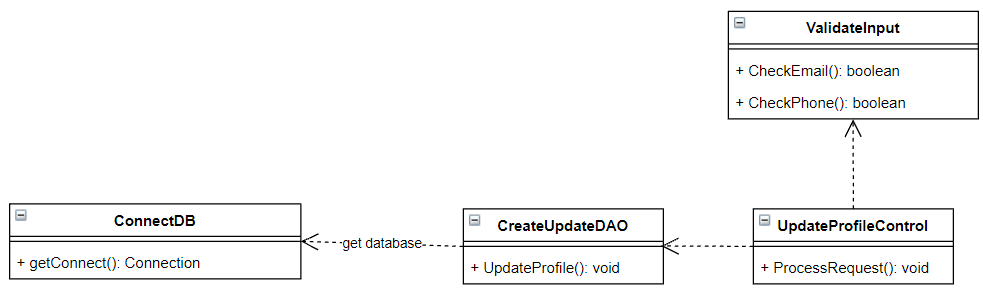


* Create new Manager



## 3. User / Update Profile

### a. Class Diagram



### b. Class Specifications

#### CreateUpdateDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void UpdateProfile() | - Update user information and save it into database  - Input: String id,  String fullname,  String email,  String phone,  String address,  String image,  String gender. |

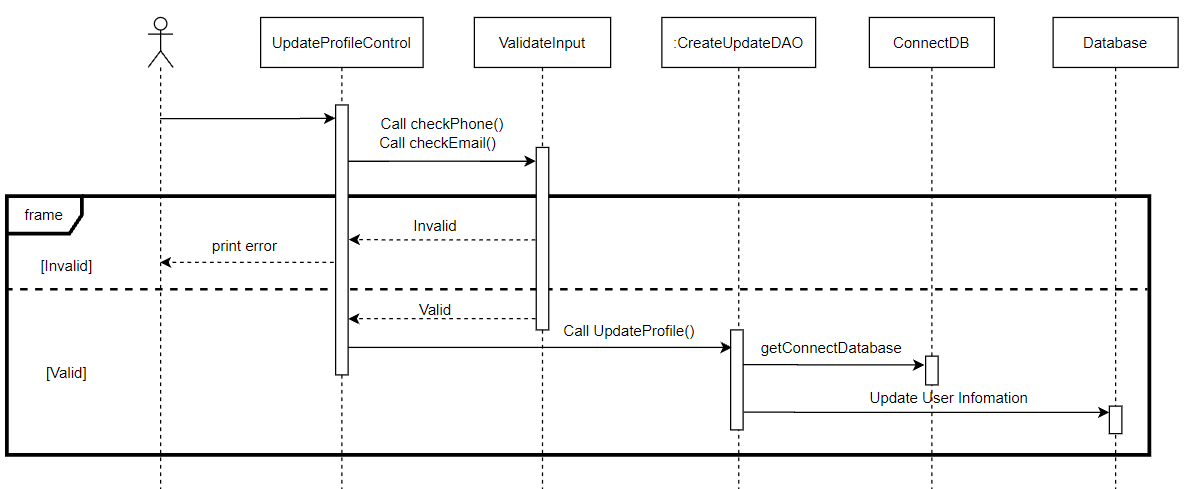
#### UpdateProfileControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - Get and validate information from jsp page, then call UpdateProfile() to update user information.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

#### ValidateInput

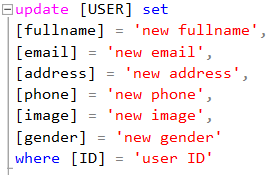
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Bool checkEmail() | - Check if email matches the correct format.  - Input: email : String;  - Output: if match -> true, else -> false |
| 01 | Bool checkPhone() | - Check if phone number matches correct format.  - Input: phone: String;  - Output: if match -> true, else -> false |

### c. Sequence Diagram(s)



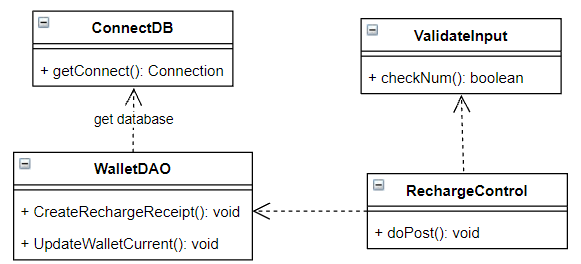
### d. Database queries

* Update User Information



## 4. Recharge / Recharge & CreateRechargeReceipt

### a. Class Diagram



### b. Class Specifications

#### WalletDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void CreateRechargeReceipt() | - Create recharge receipt after user recharge.  - Input: String userID,  String paymentID,  String date,  double amount |
| 02 | Void UpdateWalletCurrent() | - Update the current money in user wallet.  - Input: String walletID, double moneyChange |

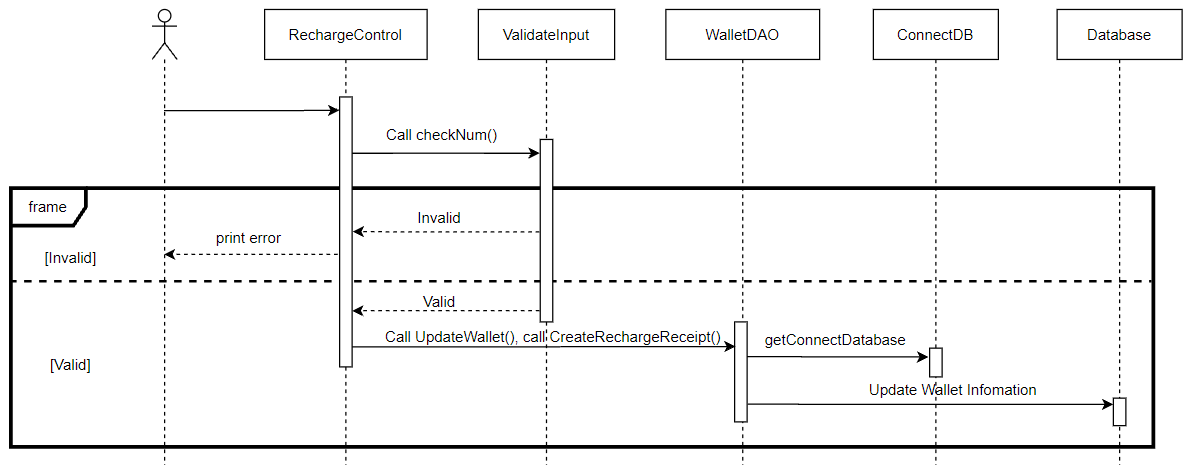
#### RechargeControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - Get data form jsp page, check valid the input amount, then start update the user wallet and create receipt.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

#### ValidateInput

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Bool checkNum () | - Check if input amount is number and larger than 0  - Input: amount : String;  - Output: true / false |

### c. Sequence Diagram(s)



### d. Database queries

* Update Wallet current:

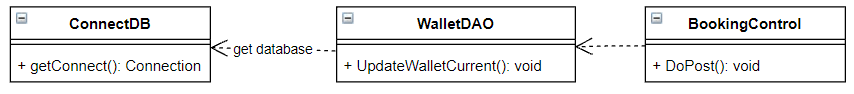


* Create new recharge receipt:



## 5. Recharge / Paid

### a. Class Diagram



### b. Class Specifications

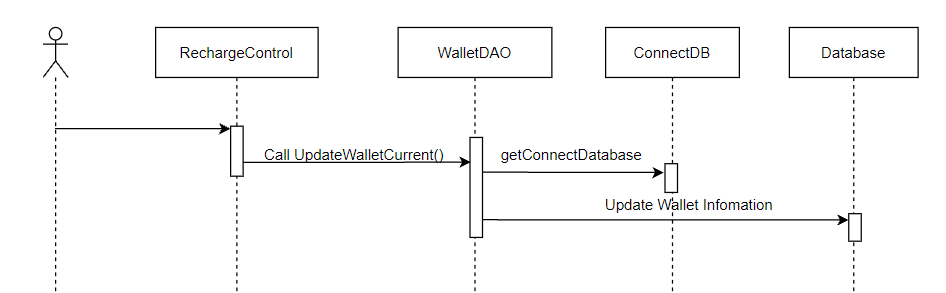
#### WalletDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 02 | Void UpdateWalletCurrent() | - Update the current money in user wallet.  - Input: String walletID, double moneyChange |

#### BookingControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void DoPost() | - After user booking a service, system will paid for that service by using user’s wallet.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

### c. Sequence Diagram(s)



### d. Database queries

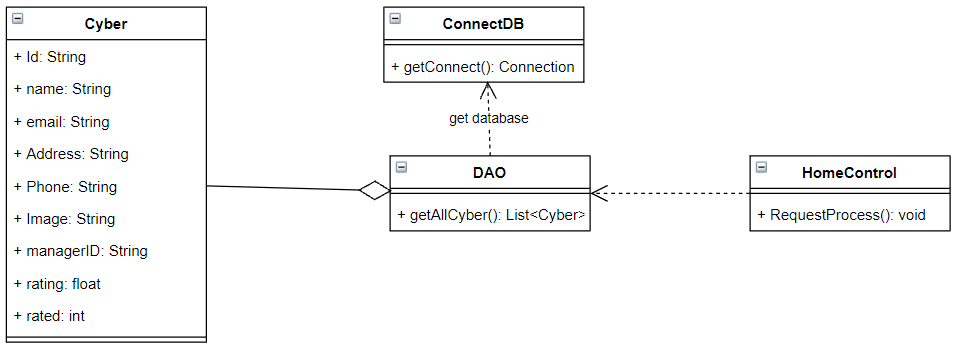
* Update Wallet current:



## 6. Cyber / View Cyber List

View Cyber List on the Home page of User.

### a. Class Diagram



### b. Class Specifications

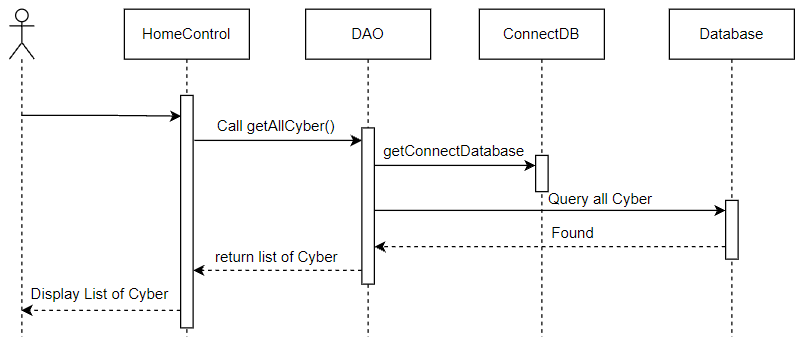
#### DAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | List<Cyber> getAllCyber() | - Get all Cyber in the database |

#### HomeControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - Get list of cybers form the database and push that list on the jsp page.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

### c. Sequence Diagram(s)



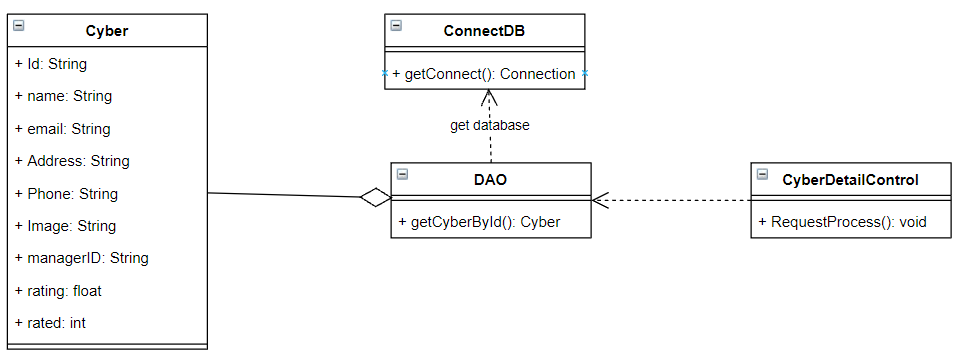
### d. Database queries

* Get all Cyber



## 7. Cyber / View Cyber Detail

### a. Class Diagram



### b. Class Specifications

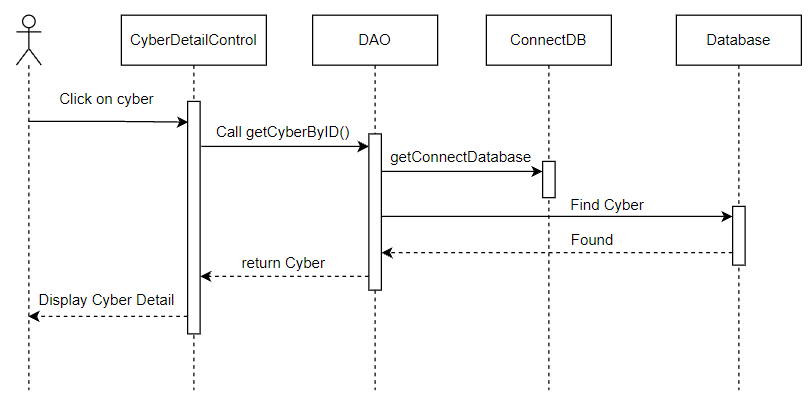
#### DAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Cyber getCyberById() | - Get Cyber by cyberID from the database |

#### HomeControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - Get cybers from the database by input UserID and push that cyber to the jsp page.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

### c. Sequence Diagram(s)



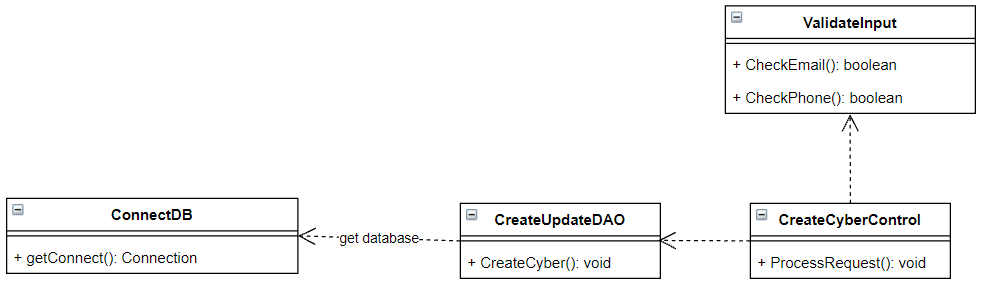
### d. Database queries

* Get all Cyber



## 8. Cyber/ Create Cyber

### a. Class Diagram



### b. Class Specifications

#### CreateUpdateDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | CreateCyber() | - Add new Cyber into database  - Input: String name,  String address,  String email,  String phone,  String managerID,  String image |

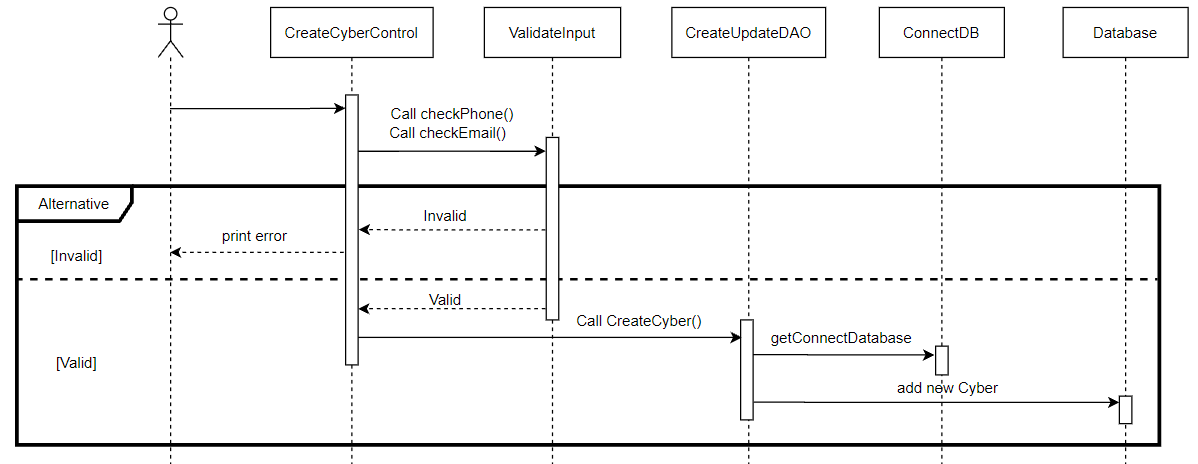
#### UpdateCyberControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - Get and validate information from jsp page, then call CreateCyber() to add new Cyber.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

#### ValidateInput

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Bool checkEmail() | - Check if email matches the correct format.  - Input: email : String;  - Output: if match -> true, else -> false |
| 01 | Bool checkPhone() | - Check if phone number matches correct format.  - Input: phone: String;  - Output: if match -> true, else -> false |

### c. Sequence Diagram(s)



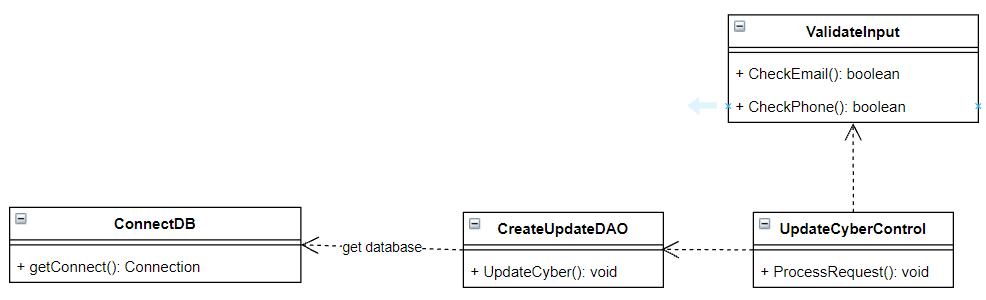
### d. Database queries

-Create new Cyber:



## 9. Cyber/ Update Cyber

### a. Class Diagram



### b. Class Specifications

#### CreateUpdateDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | UpdateCyber() | - Update Cyber information in the database  - Input: String ID,  String name,  String address,  String email,  String phone |

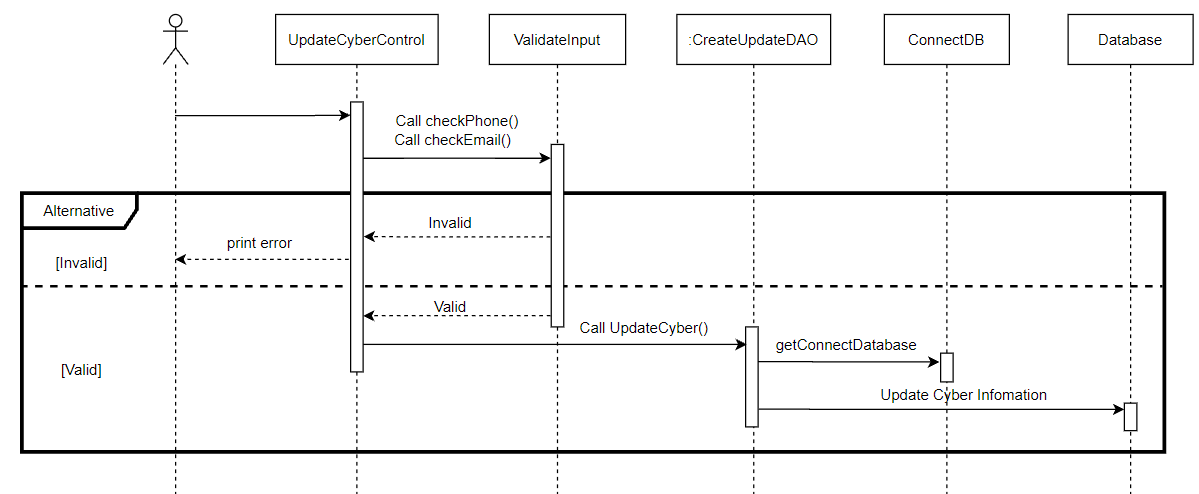
#### UpdateCyberControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - Get and validate information from jsp page, then call UpdateCyber() to update Cyber information.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

#### ValidateInput

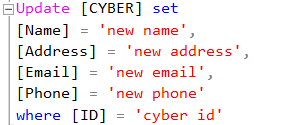
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Bool checkEmail() | - Check if email matches the correct format.  - Input: email : String;  - Output: if match -> true, else -> false |
| 01 | Bool checkPhone() | - Check if phone number matches correct format.  - Input: phone: String;  - Output: if match -> true, else -> false |

### c. Sequence Diagram(s)



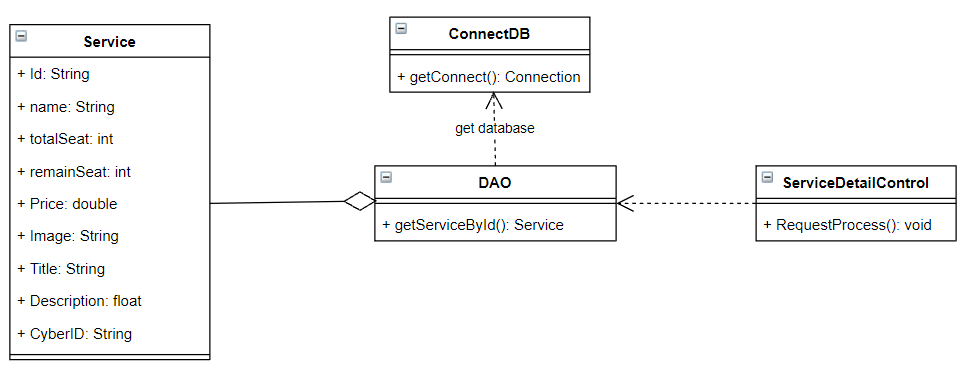
### d. Database queries

-Update Cyber by cyber ID:



## 10. Service / View Service Detail

### a. Class Diagram



### b. Class Specifications

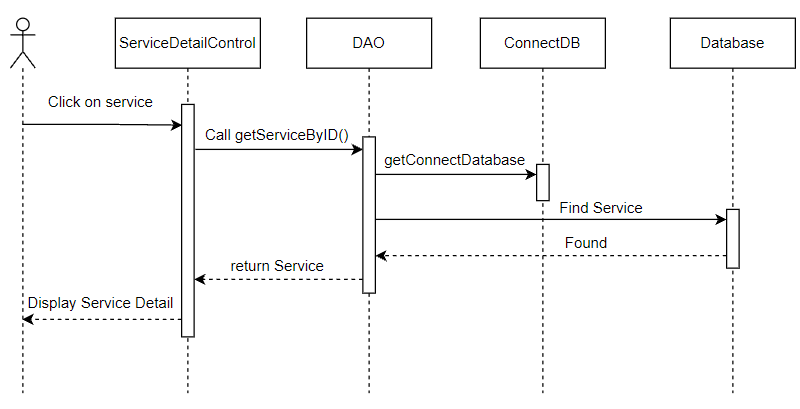
#### DAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Cyber getServiceById() | - Get Service by ServiceID from the database |

#### HomeControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - Get Service from the database by input ServiceID and push that Service to the jsp page.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

### c. Sequence Diagram(s)



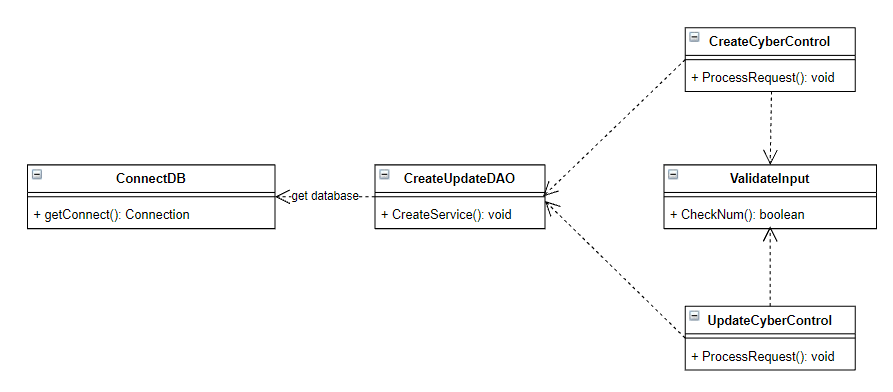
### d. Database queries

-Get Service by service ID:



## 11. Service/ Create Service

### a. Class Diagram



### b. Class Specifications

#### CreateUpdateDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | CreateService() | - Create new Service of certain Cyber.  - Input: String CyberID,  String name,  double price,  int TotalSeat |

#### UpdateCyberControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - Get and validate information from jsp page, add new service when update Cyber information.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

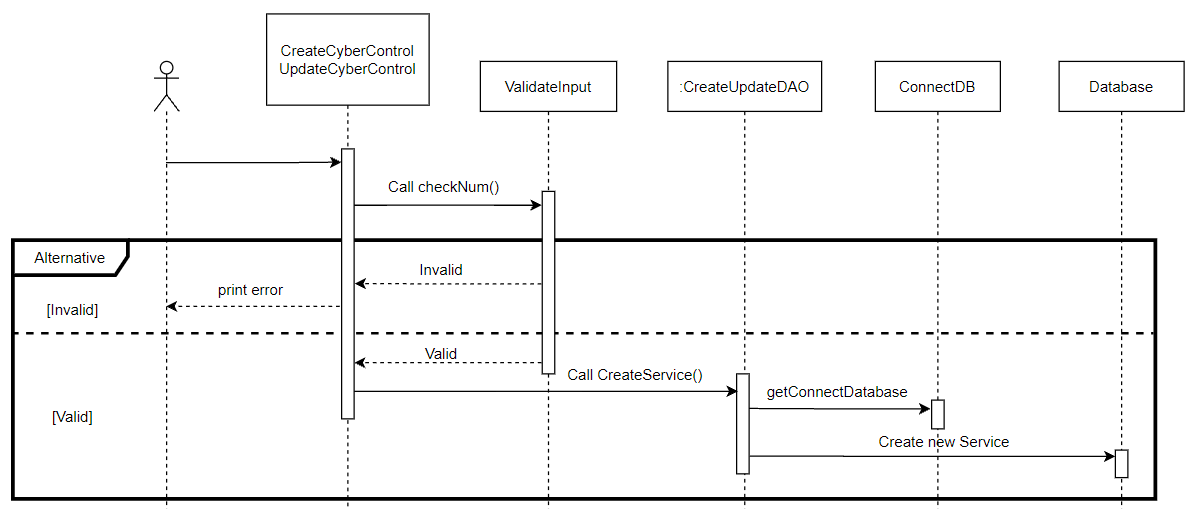
#### CreateCyberControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - Get and validate information from jsp page, add new service when create new Cyber.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

#### ValidateInput

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Bool checkNum() | - Check the number of service’s seats and the price of the service is valid or not. (All of them are larger than 0 and must be number)  - Input: email : String;  - Output: if is valid -> true, else -> false |

### c. Sequence Diagram(s)



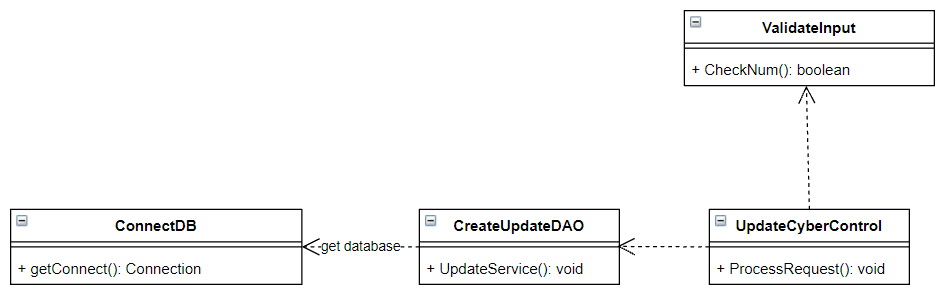
### d. Database queries

-Update Service by service ID:



## 12. Service/ Update Service

### a. Class Diagram



### b. Class Specifications

#### CreateUpdateDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | UpdateService() | - Update Service information in the database  - Input: String ID,  String name,  double price,  int newTotalSeat,  int newRemain |

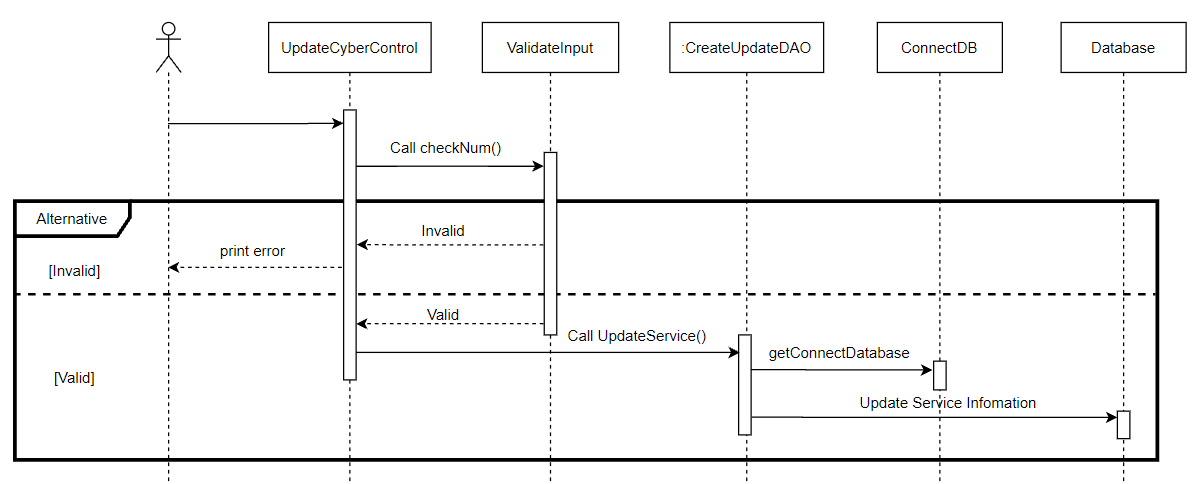
#### UpdateCyberControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - Get and validate information from jsp page, then call UpdateService() to update Service information.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

#### ValidateInput

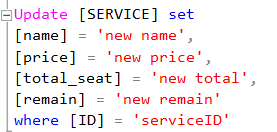
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Bool checkNum() | - Check the number of service’s seats and the price of the service is valid or not. (All of them are larger than 0 and must be number)  - Input: email : String;  - Output: if is valid -> true, else -> false |

### c. Sequence Diagram(s)



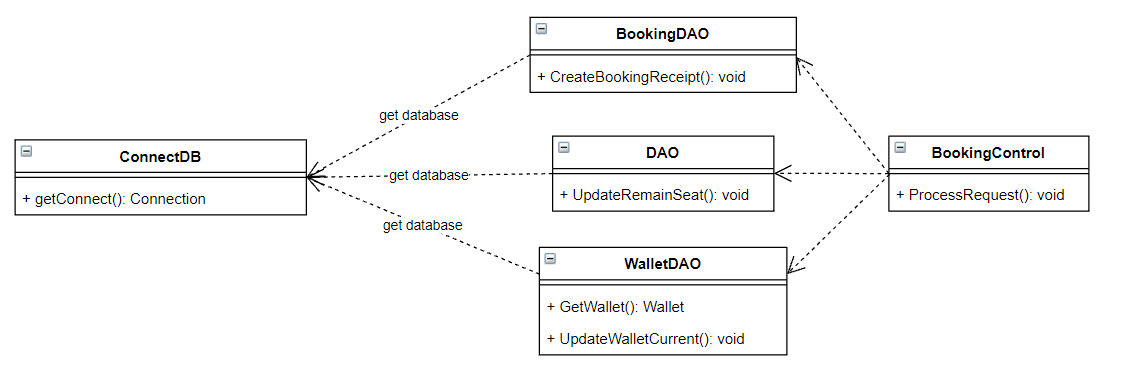
### d. Database queries

-Update Service by service ID:



## 13. Booking/ Book (Create BookingReceipt)

### a. Class Diagram



### b. Class Specifications

#### DAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | UpdateRemainSeat() | - Update remain seats of the booked service  - Input: String ServiceID, int changeAmount (-1) |

#### BookingDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | CreateBookingReceipt() | - Create booking receipt  - Input: String userID,  String serviceID,  int playHour,  double totalPaid |

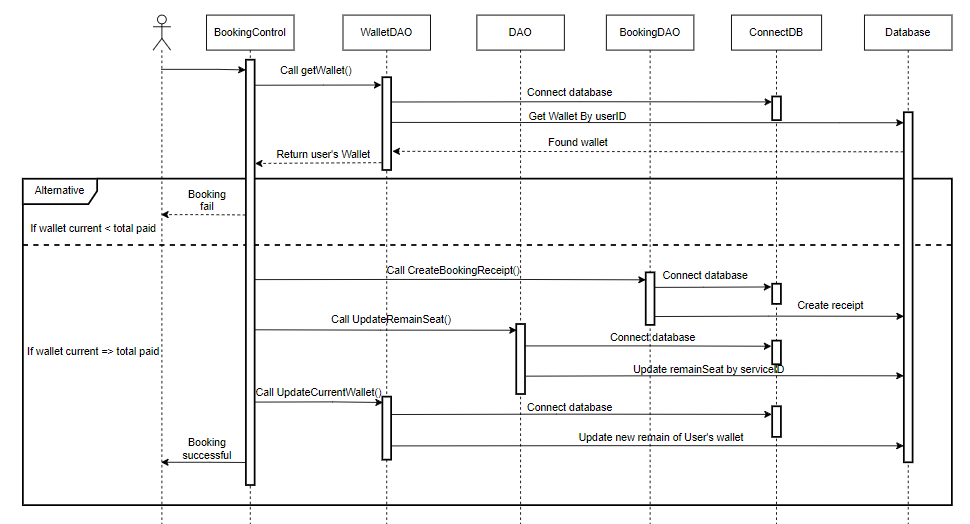
#### WalletDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | GetWallet() | - Get User’s wallet  - Input: String userID  -Output: Wallet userWallet |
| 02 | UpdateCurrentWallet() | - Detail at II.5 Recharge/Paid. |

#### BookingControl

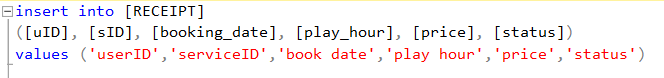
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - get request from jsp page and do execute all function above to complete booking task.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

### c. Sequence Diagram(s)



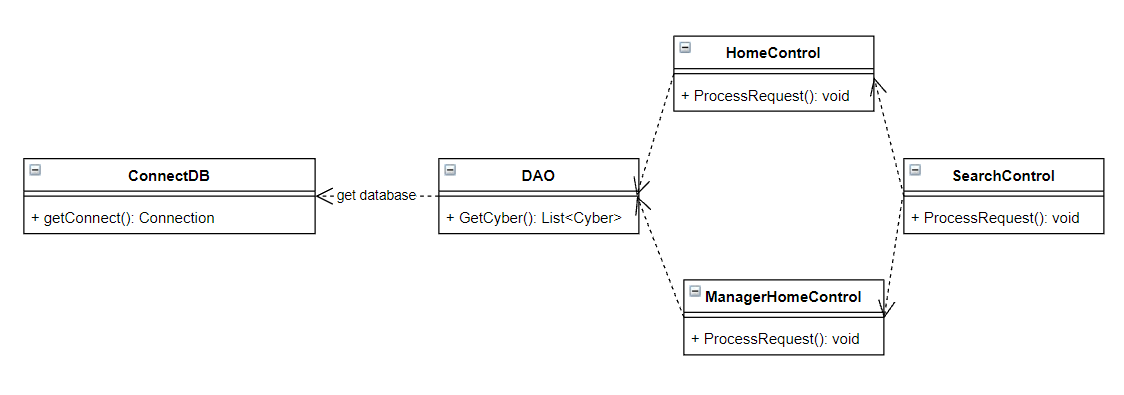
### d. Database queries

-Create Booking receipt:



## 14. Search/ SearchCyber

### a. Class Diagram



### b. Class Specifications

#### DAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | List<Cyber> GetCyber() | - Get cyber contain search string in name  - Input: String searchString. |

#### SearchControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - get search string from jsp page and check if it is User or Manager to forward to the correct controller  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

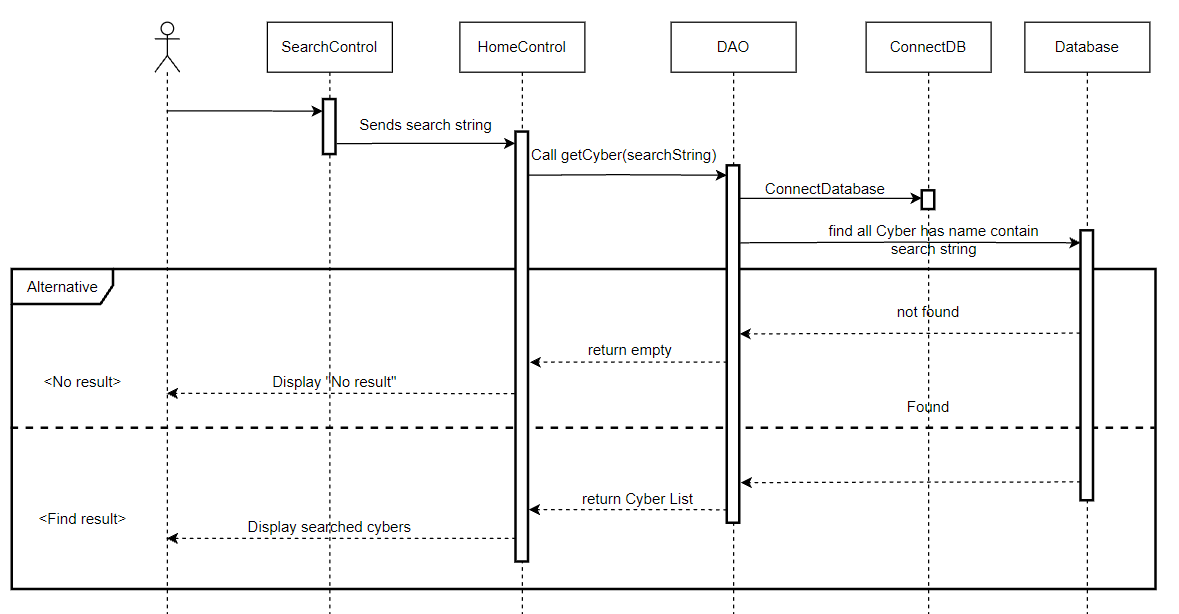
#### HomeControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - SearchControl will forward to this if User is searching Cyber  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

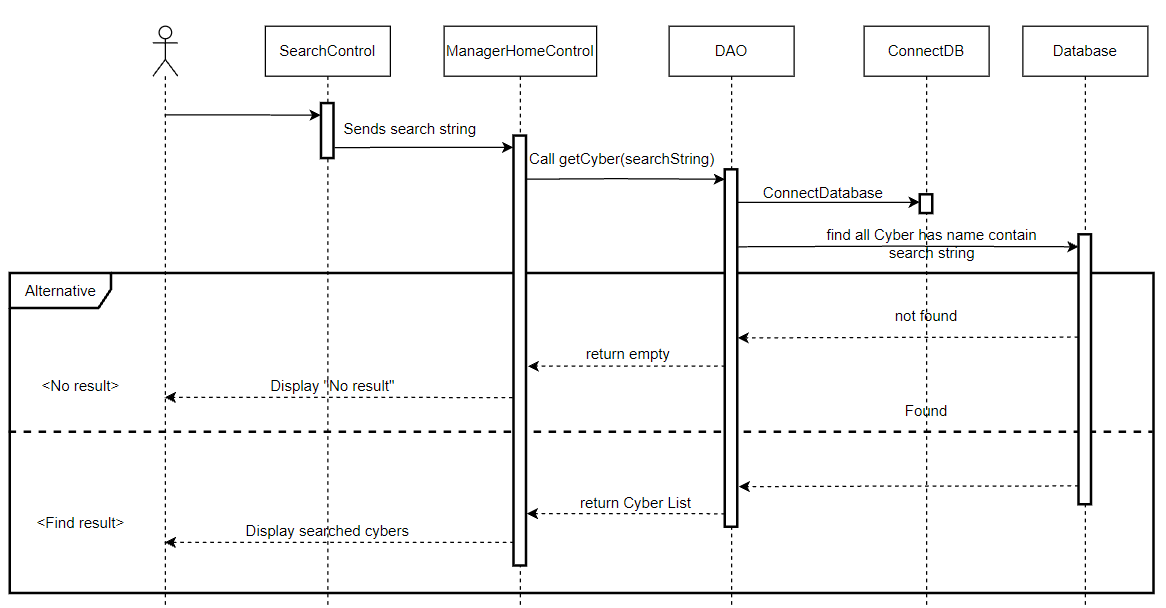
#### ManagerHomeControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - SearchControl will forward to this if Manager is searching Cyber  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

### c. Sequence Diagram(s)

**

SearchCyber sequence diagram for User

**

SearchCyber sequence diagram for Cyber Manager

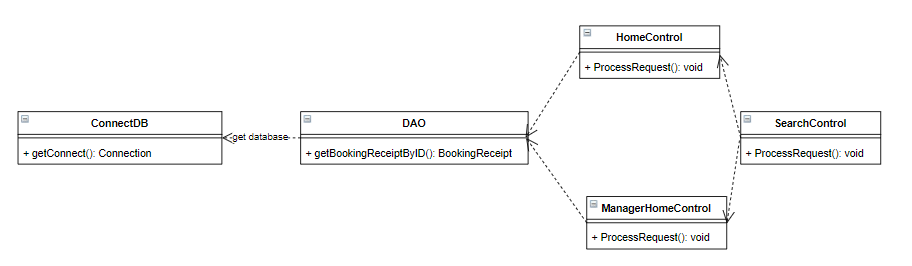
### d. Database queries

-Get Cyber has name contain search string:



## 15. Search/ SearchBookingReceipt

### a. Class Diagram



### b. Class Specifications

#### DAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | BookingReceipt getBookingReceiptByID () | - Get Booking receipt by receipt ID  - Input: String searchString. |

#### SearchControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - get search string from jsp page and check if it is User or Manager to forward to the correct controller  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

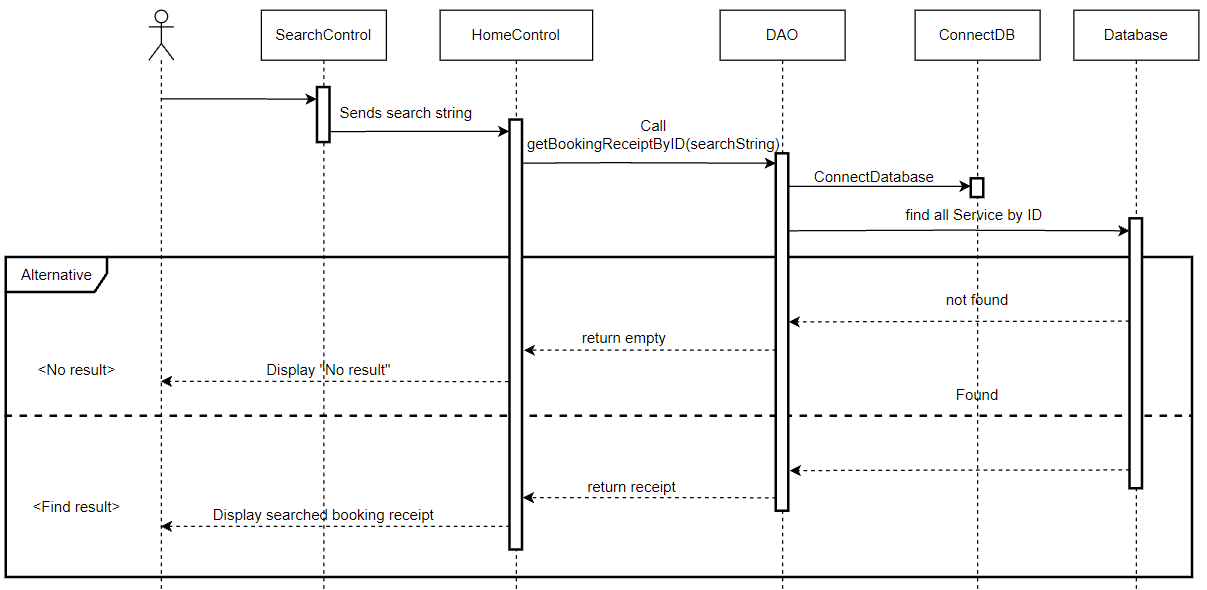
#### HomeControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - SearchControl will forward to this if User is searching Receipt  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

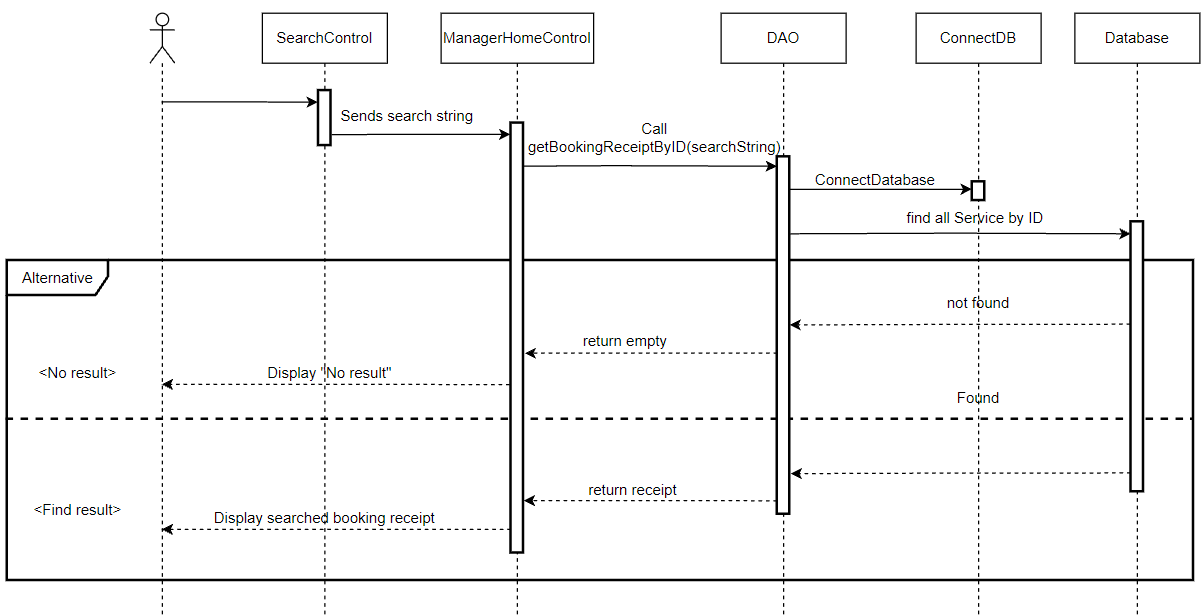
#### ManagerHomeControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - SearchControl will forward to this if Manager is searching Receipt  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

### c. Sequence Diagram(s)

**

Search Booking Receipt sequence diagram for User

**

Search Booking Receipt sequence diagram for Cyber Manager

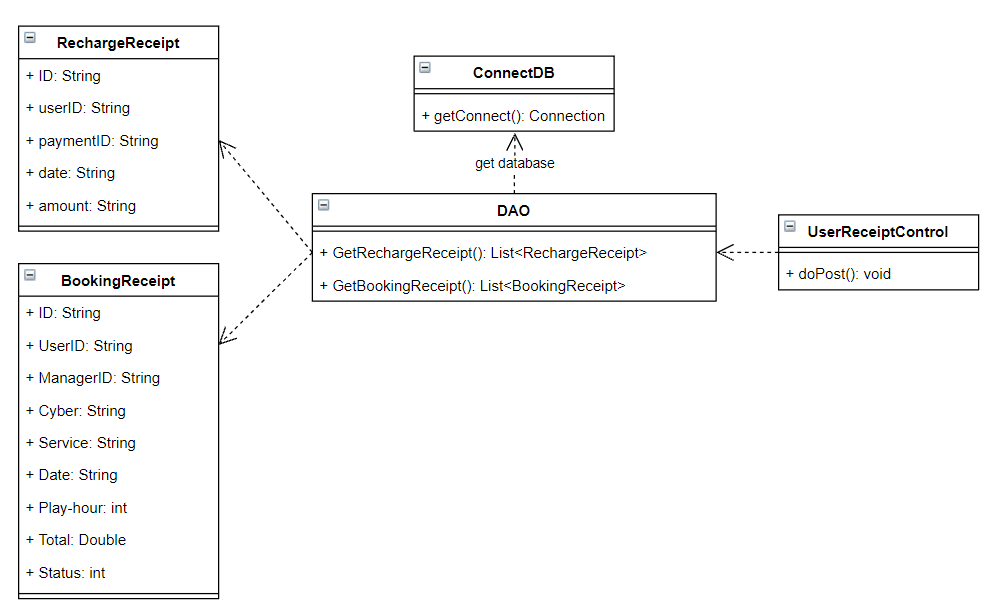
### d. Database queries

-Get Booking receipt by ID:



## 16. Receipt/ ViewReceipt

### a. Class Diagram



### b. Class Specifications

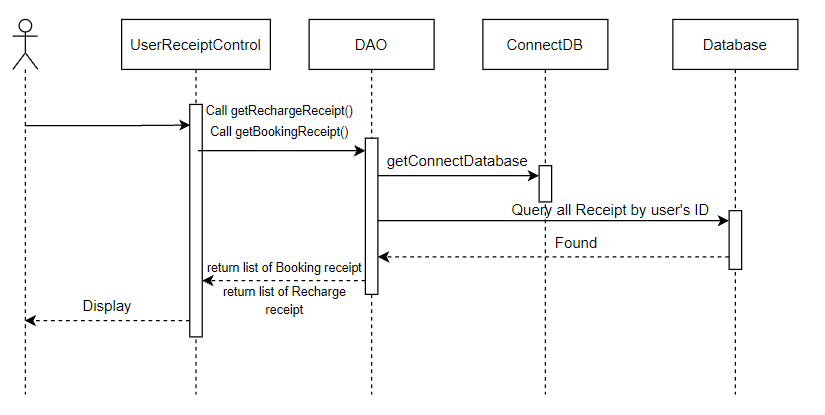
#### DAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | List<BookingReceipt> getBookingReceipt() | - Get all user Booking receipts by user’s id  - Input: String userID.  -Output: List of user’s Booking receipts |
| 02 | List<RechargeReceipt> getRechargeReceipt() | - Get all user Recharge receipts by user’s id  - Input: String userID.  -Output: List of user’s Recharge receipts |

#### DeleteControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void doPost() | - Gets all User’s receipts and pushes them to user interface.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

### c. Sequence Diagram(s)

**

View Receipt sequence diagram for User

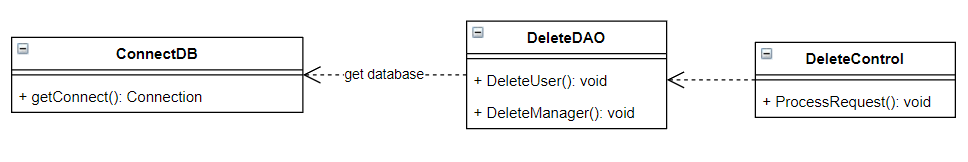
### d. Database queries

- Get Booking receipt by user ID: 

- Get Recharge receipt by user ID: 

## 17. Delete User & Manager

### a. Class Diagram



### b. Class Specifications

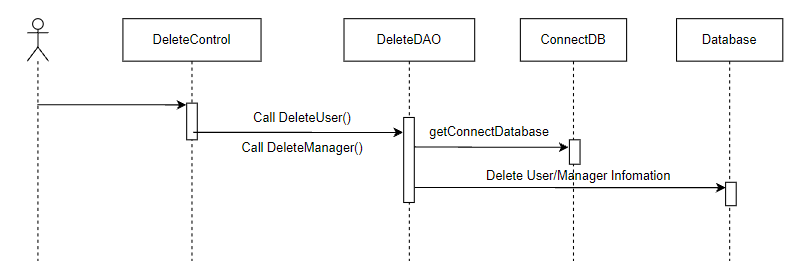
#### DeleteDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Void DeleteUser() | - Remove User from database  - Input: String userID. |
| 02 | Void DeleteManager() | - Remove Manager from database  - Input: String mangerID. |

#### DeleteControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void doPost() | - Get delete request and delete a User or Manager  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

### c. Sequence Diagram(s)

**

Delete User/Manager sequence diagram

### d. Database queries

- Delete User

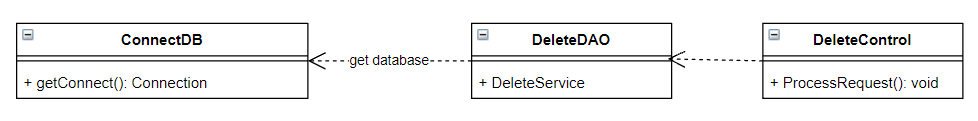


- Delete manager



## 18. Service/Delete Service

### a. Class Diagram



### b. Class Specifications

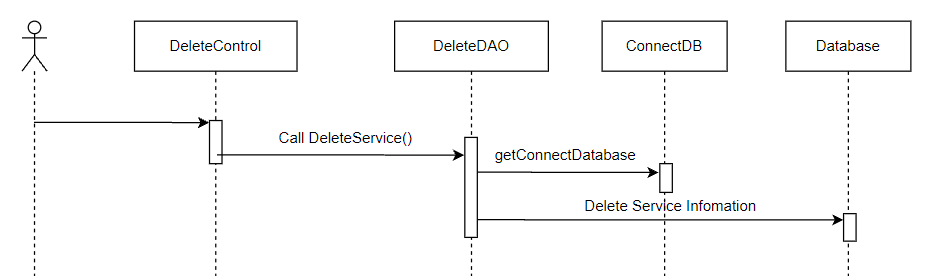
#### DeleteDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Void DeleteService() | - Remove Service from database  - Input: String serviceID. |

#### DeleteControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void doPost() | - Get delete request and delete a User or Manager  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

### c. Sequence Diagram(s)

**

Delete Service sequence diagram

### d. Database queries

- Delete Service



-Trigger when delete a service, all booking receipts which haven’t used will refund to user

create trigger [dbo].[del\_Service] on [dbo].[SERVICE] for DELETE as

Begin

update [BOOKING\_RECEIPT]

Set

[status] = -1

where [status] = 0 and [BOOKING\_RECEIPT].[Sevice] in (Select Cast([ID] as nvarchar(3))+'-'+[name] from deleted)

end

create trigger updateWallet\_after\_CancelReceipt on [BOOKING\_RECEIPT] after UPDATE as

Begin

declare @uID int;

Select @uID = [uID] from inserted

update [WALLET]

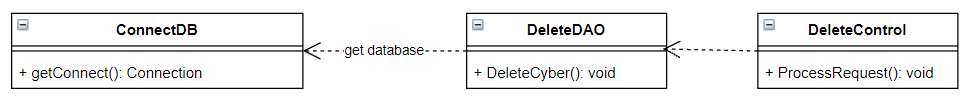
set [current] = [current] + (select [price] from inserted where [status] = -1)

where [uID] = @uID

end

## 19. Cyber/Delete Cyber

### a. Class Diagram



### b. Class Specifications

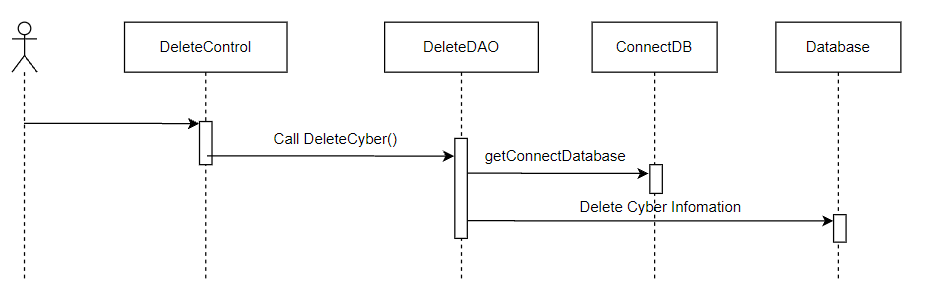
#### DeleteDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Void DeleteCyber() | - Remove Cyber from database  - Input: String CyberID. |

#### DeleteControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void doPost() | - Get delete request and delete a User or Manager  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

### c. Sequence Diagram(s)

**

Delete Cyber sequence diagram

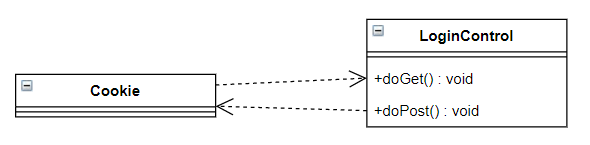
### d. Database queries

- Delete Service



## 20. Authorization / Remember account

### a. Class Diagram

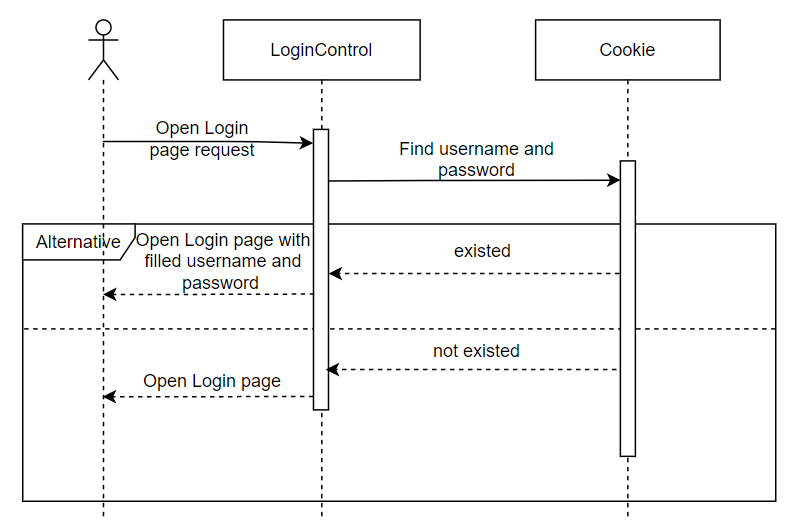


### b. Class Specifications

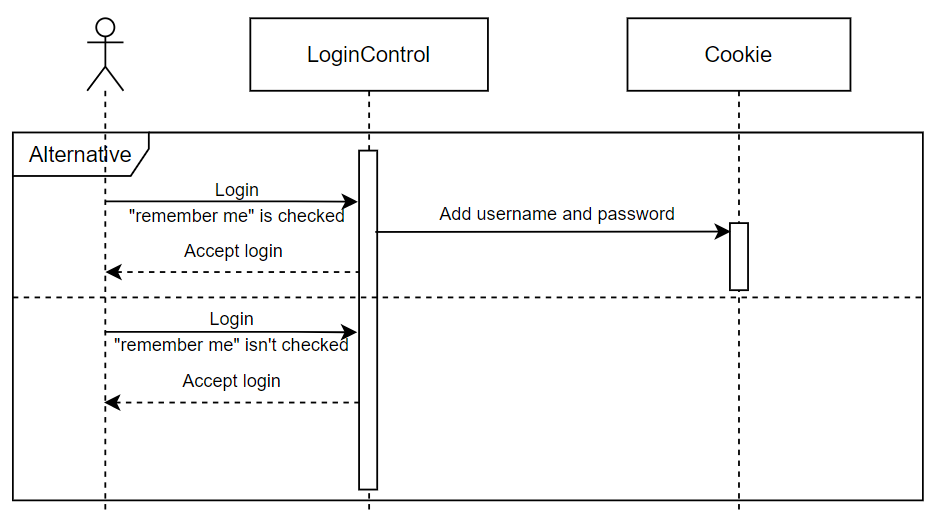
#### LoginControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void doGet() | - If username and password are existed in Cookie, gets them add auto fill in username and password textbox on login page.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |
| 02 | void doPost() | - Login account, if “Remember me” box is checked, add that username and password and save them in Cookie.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

### c. Sequence Diagram(s)

**

Sequence diagram when open a login page with a remembered account



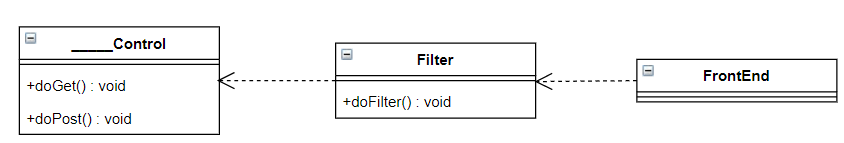
Sequence diagram when check “remember me” check box during log in

### d. Database queries

<<None>>

## 21. Filter / Filter

### a. Class Diagram

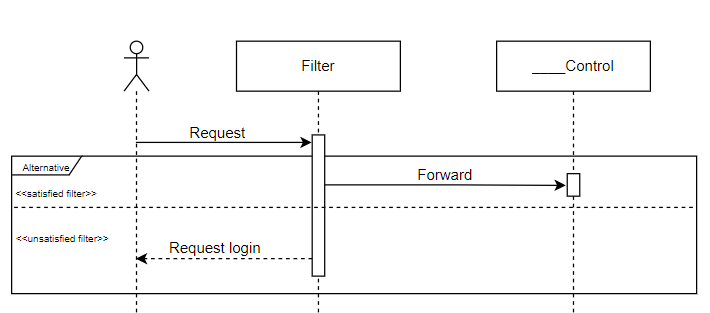


### b. Class Specifications

#### Filter

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void doFilter() | - check if session has already had a account, allows access to the page. If not, send back the request to login account |

### c. Sequence Diagram(s)

**

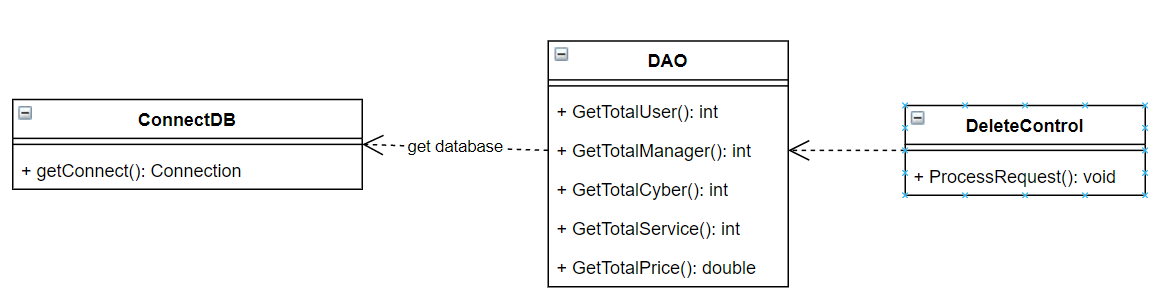
Sequence diagram when open a login page with a remembered account

### d. Database queries

<<None>>

## 22. Admin / Web’s statistic

### a. Class Diagram



### b. Class Specifications

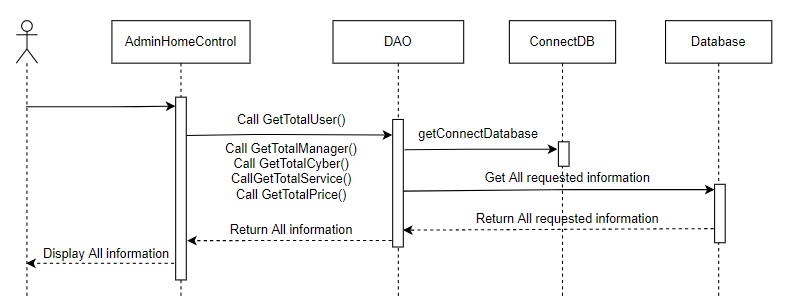
#### AdminHomeControl

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void ProcessRequest() | - Get information from database and push them to admin page.  - Input: request : HttpServletRequest,  response : HttpServletResponse. |

#### DAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Int GetTotalUser() | - Get number of Users who sign up to use the website. |
| 02 | Int GetTotalManager() | - Get number of Manager |
| 03 | Int GetTotalCyber() | - Get number of Cyber that connect with the website. |
| 04 | Int GetTotalService() | - Get number of Service |
| 05 | Double GetTotalPrice() | - Get total price of used booking-receipt |

### c. Sequence Diagram(s)

**

Sequence diagram of loading Web’s statistic page

### d. Database queries

- Get number of Users: select count(ID) from [USER]

- Get number of Managers: select count(ID) from [MANAGER]

- Get number of Cybers: select count(ID) from [CYBER]

- Get number of Service: select count(ID) from [SERVICE]

- Get total price of used Booking-receipt:

select sum(price) from [BOOKING\_RECEIPT] where [status] = 1

# III. Database Tables

## 1. User

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | Int |  |  | ✓ | PK |  |
| 2 | Username | Varchar | 255 |  | ✓ |  |  |
| 3 | Password | Varchar | 255 |  | ✓ |  | Encrypted password |
| 4 | Fullname | Nvarchar | 255 |  | ✓ |  |  |
| 5 | Email | Varchar | 50 |  |  |  |  |
| 6 | address | Nvarchar | 255 |  |  |  |  |
| 7 | Phone | Varchar | 15 |  |  |  |  |
| 8 | Image | Varchar | Max |  |  |  |  |
| 9 | Gender | Varchar | 10 |  |  |  |  |

## 2. Manager

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | Int |  |  | ✓ | PK |  |
| 2 | Username | Varchar | 255 |  | ✓ |  |  |
| 3 | Password | Varchar | 255 |  | ✓ |  | Encrypted password |
| 5 | Email | Varchar | 50 |  |  |  |  |
| 7 | Phone | Varchar | 15 |  |  |  |  |
| 8 | Role | int |  |  |  |  | 0 is admin, 1 is Cyber Manager |

## 3. Cyber

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | Int |  |  | ✓ | PK |  |
| 2 | Name | Varchar | 255 |  | ✓ |  |  |
| 3 | Email | Varchar | 50 |  |  |  |  |
| 4 | Address | Nvarchar | 255 |  | ✓ |  |  |
| 5 | Phone | Varchar | 15 |  |  |  |  |
| 6 | Image | Varchar | Max |  |  |  |  |
| 7 | Total\_rate | Float |  |  |  |  | Total rating score |
| 8 | Amount\_rated | Int |  |  |  |  | Amount of user rated the cyber |
| 9 | cmID | Int |  |  | ✓ | FK | cmID = [Manager].[ID]  on delete cascade |

## 4. Service

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | Int |  |  | ✓ | PK |  |
| 2 | Name | Varchar | 255 |  |  |  |  |
| 3 | Total | Int |  |  |  |  | Total seats |
| 4 | Remain | Int |  |  |  |  | Amount of seats after users booked |
| 5 | Price | Money |  |  |  |  | Price/h |
| 6 | Image | Varchar | Max |  |  |  |  |
| 7 | title | nvarchar | 225 |  |  |  |  |
| 8 | Description | Nvarchar | Max |  |  |  |  |
| 9 | cyberID | Int |  |  | ✓ | FK | cyberID = [Cyber].[ID]  on delete cascade |

## 5. Wallet

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | Int |  |  | ✓ | PK |  |
| 2 | uID | Int |  |  | ✓ | FK | uID = [User].[ID]  on delete cascade |
| 3 | Current | Money |  |  |  |  | Current money in user wallet |
| 4 | Total\_paid | Money |  |  |  |  |  |

## 6. Payment

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | Int |  |  | ✓ | PK |  |
| 2 | Name | Varchar | 255 |  |  |  |  |

## 7. Recharge\_Receipt

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | Int |  |  | ✓ | PK |  |
| 2 | UserID | int |  |  | ✓ | FK | UserID = [User].[ID]  on delete cascade |
| 3 | PaymentID | nvarchar | 50 |  | ✓ |  |  |
| 4 | date | varchar | 15 |  | ✓ |  |  |
| 5 | amount | Money |  |  | ✓ |  | Amount of recharged money |

## 8. Booking\_Receipt

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | Int |  |  | ✓ | PK |  |
| 2 | UserID | Int |  |  | ✓ | FK | UserID = [User].[ID]  on delete cascade |
| 3 | ManagerID | Nvarchar(30) |  |  | ✓ |  |  |
| 4 | Cyber | Nvarchar(30) |  |  | ✓ |  |  |
| 5 | Service | Int |  |  | ✓ |  |  |
| 6 | date | varchar | 15 |  | ✓ |  |  |
| 7 | Play\_hours | Int |  |  | ✓ |  |  |
| 8 | Price | Int |  |  | ✓ |  |  |
| 9 | Status | Int |  |  | ✓ |  | 0 is available, 1 is unavailable, -1 is canceled |

## 9. User\_Favorite

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | UserID | Int |  |  | ✓ | PK, FK | UserID = [User].[ID]  on delete cascade |
| 2 | CyberID | int |  |  | ✓ | PK, FK | cyberID = [Cyber].[ID] |

## 10. City

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | Int |  |  | ✓ | PK |  |
| 2 | Name | Varchar | 50 |  | ✓ |  |  |

## 11. Admin\_Notification

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | Int |  |  | ✓ | PK |  |
| 2 | Content | Nvarchar | Max |  | ✓ |  |  |
| 3 | Upload\_date | Varchar | 15 |  | ✓ |  |  |
| 4 | Status | Int |  |  | ✓ |  | 1 is displaying, 0 is used |