

**CLINIC MANAGEMENT SYSTEMS**

**Software Design Document**

– Hanoi, March 2022 –

**Table of Contents**

[I. Overview 3](#_Toc99186212)

[1. Code Packages/Namespaces 3](#_Toc99186213)

[II. Code Designs 4](#_Toc99186214)

[1. User Management 4](#_Toc99186215)

[a. Class Diagram 4](#_Toc99186216)

[b. Class Specifications 4](#_Toc99186217)

[UserController 4](#_Toc99186218)

[UserServiceImpl 4](#_Toc99186219)

[UserDAOImpl 5](#_Toc99186220)

[c. Sequence Diagram(s) 5](#_Toc99186221)

[d. Database queries 7](#_Toc99186222)

[2. Health news management 8](#_Toc99186223)

[a. Class Diagram 8](#_Toc99186224)

[b. Class Specifications 8](#_Toc99186225)

[HealthNewController 8](#_Toc99186226)

[HealthNewServiceImpl 8](#_Toc99186227)

[HealthNewDAOImpl 9](#_Toc99186228)

[c. Sequence Diagram(s) 9](#_Toc99186229)

[3. Appointment management 11](#_Toc99186230)

[a. Class Diagram 11](#_Toc99186231)

[b. Class Specifications 11](#_Toc99186232)

[MedicalScheduleController 11](#_Toc99186233)

[MedicalScheduleServiceImpl 11](#_Toc99186234)

[MedicalScheduleDAOImpl 12](#_Toc99186235)

[c. Sequece Diagram(s) 13](#_Toc99186236)

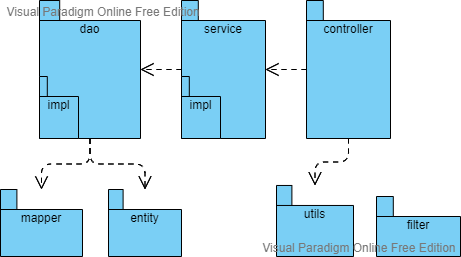
[III. Database Design 14](#_Toc99186237)

[1. Database Schema 14](#_Toc99186238)

[2. Table Description 15](#_Toc99186239)

# I. Overview

## 1. Code Packages/Namespaces



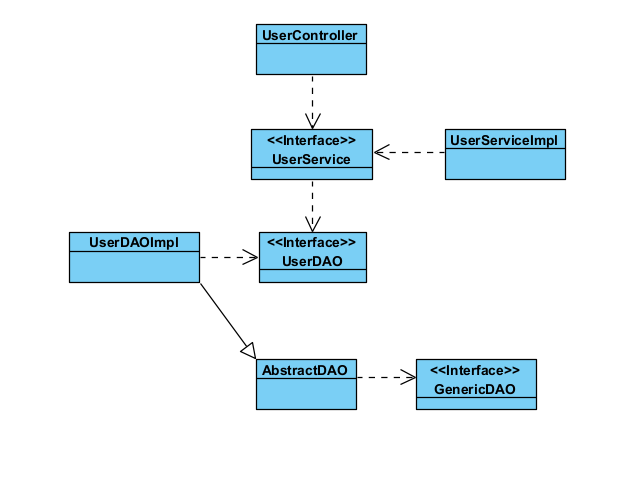
***Package descriptions & package class naming conventions***

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| *1* | *Controller* | *Handle request from client* |
| *2* | *service* | *Contain business logic* |
| *3* | *dao* | *Data access object* |
| *4* | *entity* | *model of entity* |
| *5* | *mapper* | *mapping entity’ attribute with field columns in database* |
| *6* | *utils* | *has utilities function* |
| *7* | *filter* | *web filter* |

# II. Code Designs

## 1. User Management

### a. Class Diagram

**

### b. Class Specifications

#### UserController

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doGet() | handle request from client with method GET |
| 2 | doPost | handle request from client with method POST |

#### UserServiceImpl

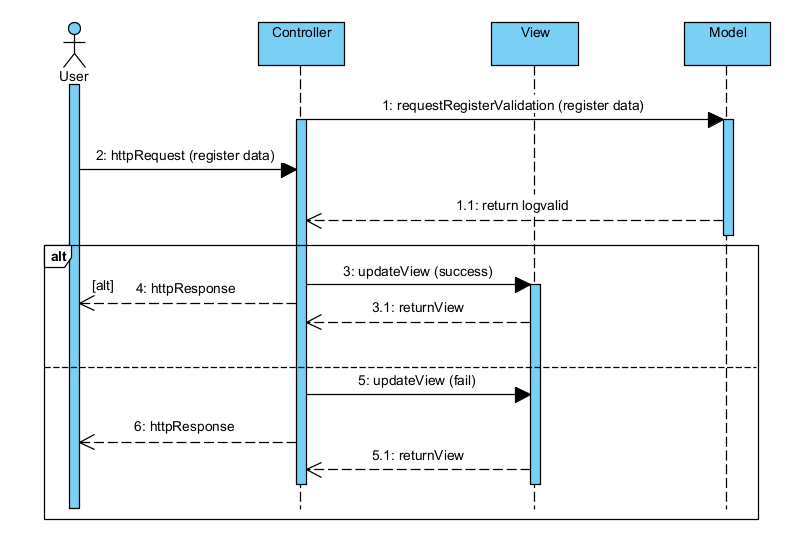
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | signIn(userName, password) | Sign in to the system |
| 2 | changePassword(userId, oldPassword, newPassword) | Update user password |
| 3 | updateInfo(userApp) | Update user information |
| 4 | signUp(userApp) | Create new account to the system |
| 5 | getUserInfoFromRequest | Return user information from request |
| 6 | findById(); | Get user information by id |
| 7 | getUserMember(type) | Get user by role |

#### UserDAOImpl

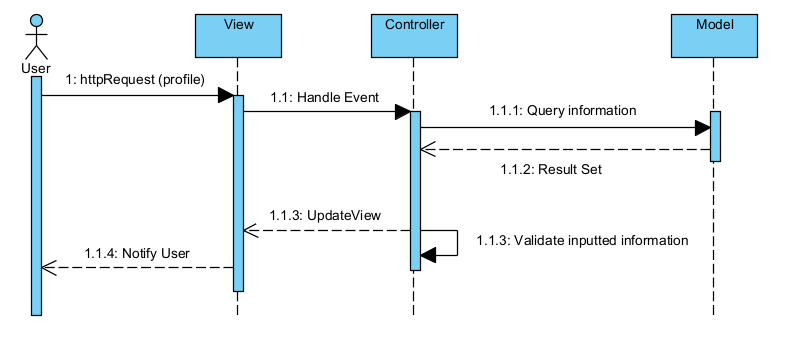
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | findByNameAndPassword(userName, password) | Get user information by username and password |
| 2 | FindByUserName | Get user information by username |
| 3 | save(userApp) | Insert new user to database |
| 4 | update(userApp) | Update user information |
| 5 | findById(id) | Get user by id |
| 6 | findByRoleId(roleID) | Get user by role id |
| 7 | save(roleApp) | Insert new role to database |
| 8 | changePassword(id, password) | Update user password |
| 9 | getAllUserMember() | Get all user |
| 10 | findUserByRoleName(role) | Get all user by role name |

### c. Sequence Diagram(s)

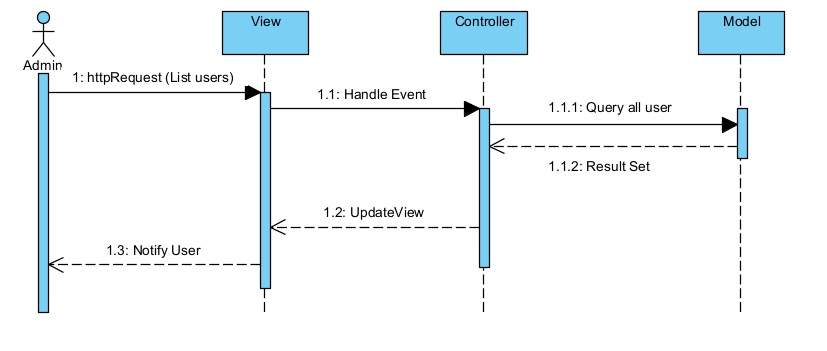
Register:



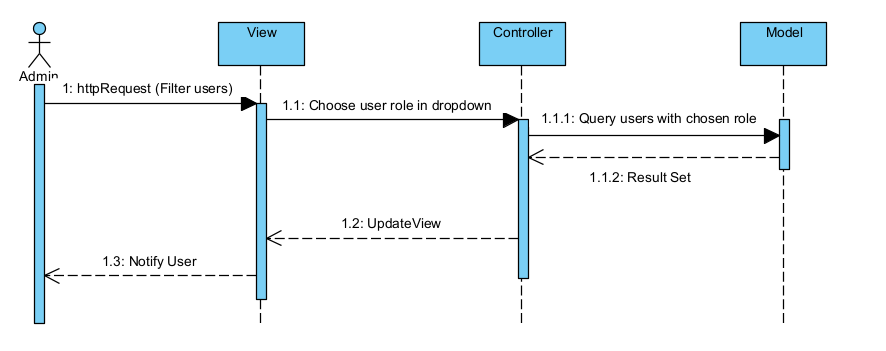
User profile:



User list:



Filter user:

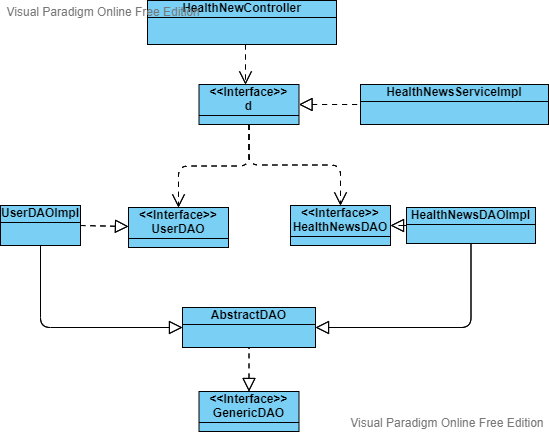


### d. Database queries

* *Find by username and password:* SELECT \* FROM user u join role r ON u.role\_id=r.id WHERE username =? AND password =?
* *Find by username:* SELECT \* FROM user WHERE username =?
* *Save user:* INSERT INTO user (role\_id, first\_name, last\_name, card\_id, gender, dob, phone\_number, avatar, email, username, password, status, created\_stamp) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
* *Update user:* UPDATE user SET first\_name=?, last\_name=?, card\_id=?, gender=?, dob=?, phone\_number=?, email=?, avatar=?, modified\_stamp=? WHERE id =?
* *Find by id:* SELECT \* FROM user WHERE id= ?
* *Find by role id:* SELECT \* FROM role WHERE id = ?
* *Find by role name:* SELECT \* FROM role WHERE name = ?
* *Save role:* INSERT INTO role (name) VALUES (?)
* *Change password:* UPDATE user SET password = ? WHERE id =?
* *Get all user:* SELECT \* FROM user u JOIN role r ON u.role\_id=r.id and r.name != 'ADMIN'
* *Find user by role:* SELECT \* FROM user u JOIN role r ON u.role\_id=r.id and r.name = ?

## 2. Health news management

### a. Class Diagram



### b. Class Specifications

#### HealthNewController

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doGet() | handle request from client with method GET |
| 2 | doPost | handle request from client with method POST |

#### HealthNewServiceImpl

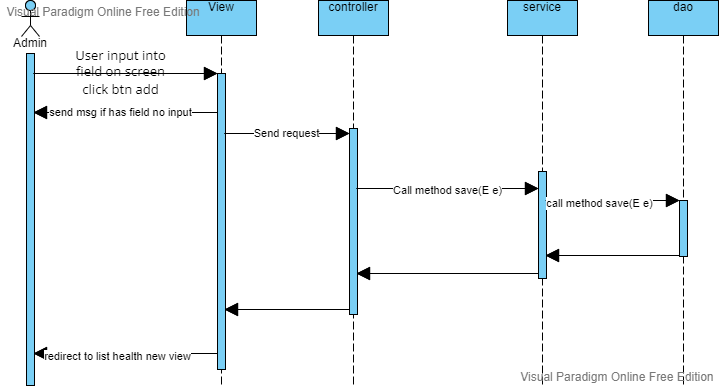
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | save(HealthNews healthNews); | add new record |
| 2 | update(HealthNews healthNews); | update record |
| 3 | findById(Long id); | get record by id |
| 4 | delete(Long id); | delete record by id |
| 5 | buildData(HttpServletRequest request) | create an instance of HealthNew |
| 6 | findAll(); | get all record |
| 7 | findByCreatedId(Long createdId); | get data of user who create the new has {id} |
| 8 | findAll(Pageable pageable); | pagination |
| 9 | getTotalItem(); | get total record |

#### HealthNewDAOImpl

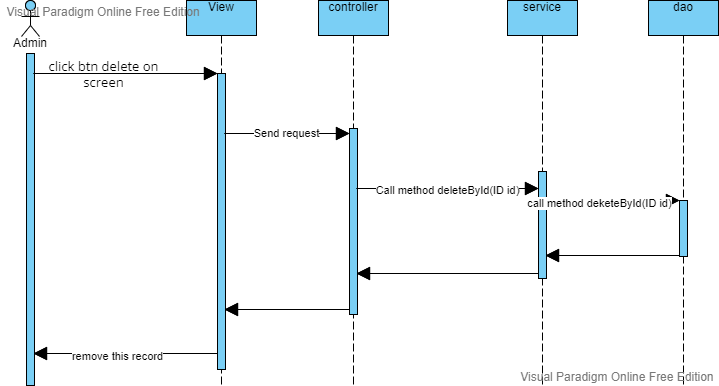
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | save(HealthNews healthNews); | add new record |
| 2 | update(HealthNews healthNews); | update record |
| 3 | findById(Long id); | get record by id |
| 4 | delete(Long id); | delete record by id |
| 5 | buildData(HttpServletRequest request) | create an instance of HealthNew |
| 6 | findAll(); | get all record |
| 7 | findAll(Pageable pageable); | pagination |
| 8 | getTotalItem(); | get total record |

### c. Sequence Diagram(s)

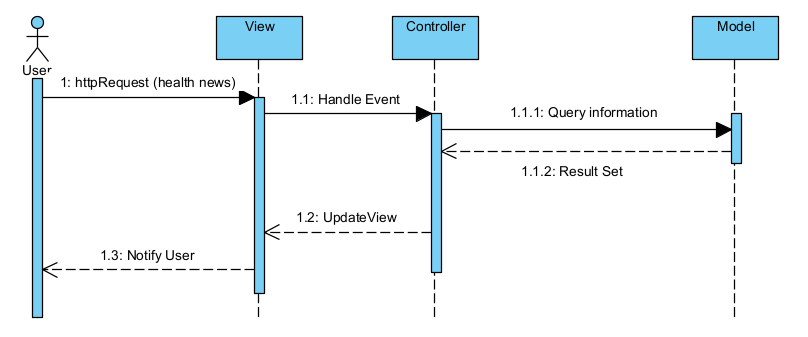
1. Add/ Update new Health news



1. Delete Health news



1. Get list health news



### **d. Database queries**

 "INSERT INTO health\_news (created\_id, title, thumbnail, short\_description, content, created\_stamp) VALUES (?, ?, ?, ?, ?, ?)"

"UPDATE health\_news SET title=?, thumbnail = ?, short\_description = ?, content= ?,modified\_stamp=? WHERE id =?"

"SELECT \* FROM health\_news WHERE id= ?"

"DELETE FROM health\_news WHERE id = ?";

"SELECT \* FROM health\_news";

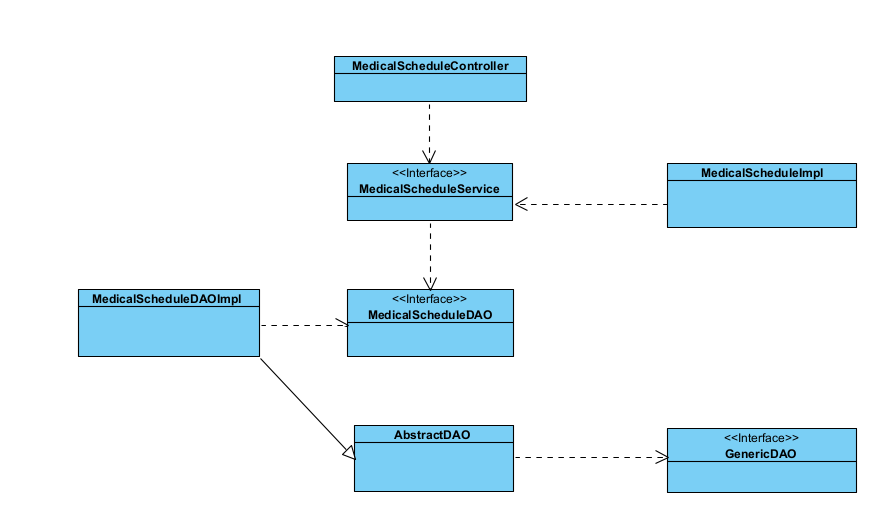
"SELECT \* FROM health\_news WHERE created\_id= ?";

SELECT \* FROM health\_news");

SELECT count(\*) FROM health\_news";

## 3. Appointment management

### a. Class Diagram



### b. Class Specifications

#### MedicalScheduleController

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doGet() | handle request from client with method GET |
| 2 | doPost() | handle request from client with method POST |

#### MedicalScheduleServiceImpl

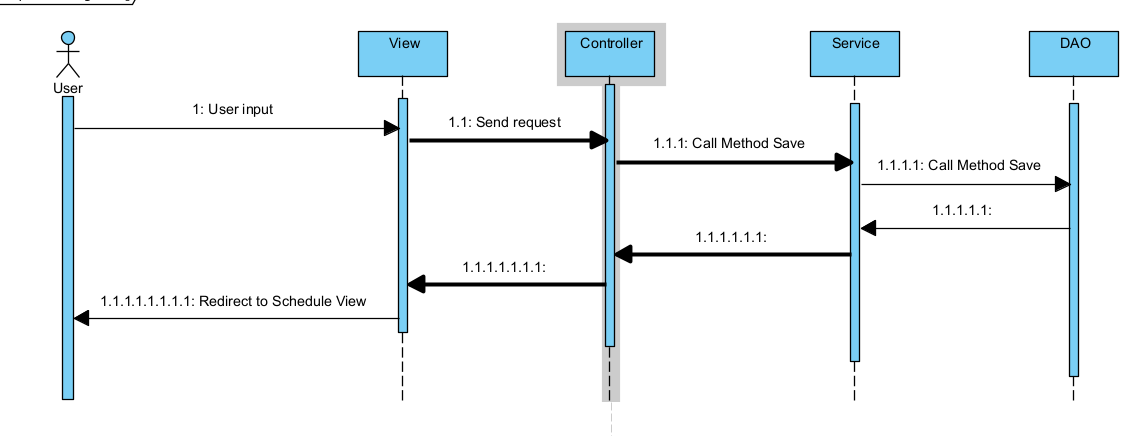
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | save(MedicalSchedule medicalSchedule); | add new record |
| 2 | update(MedicalSchedule medicalSchedule); | update record |
| 3 | findById(Long id); | get record by id |
| 4 | delete(Long id); | delete record by id |
| 5 | buildData(HttpServletRequest request) | create an instance of MedicalSchedule |
| 6 | findAll(); | get all record |
| 7 | findByCreatedId(Long createdId); | get data of user who create the new has {id} |
| 8 | updateStatus(); | update status of appointment |

#### MedicalScheduleDAOImpl

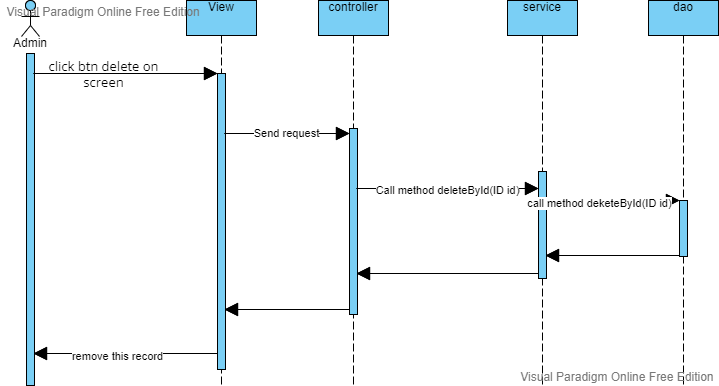
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | save(MedicalSchedule medicalSchedule); | add new record |
| 2 | update(MedicalSchedule medicalSchedule); | update record |
| 3 | findById(Long id); | get record by id |
| 4 | delete(Long id); | delete record by id |
| 5 | findAll(); | get all record |
| 6 | findByCreatedId(Long createdId); | get data of user who create the new has {id} |
| 7 | updateStatus(); | update status of appointment |

### c. Sequece Diagram(s)

1.Add/ Update Apointment:



2.Delete Apointment:

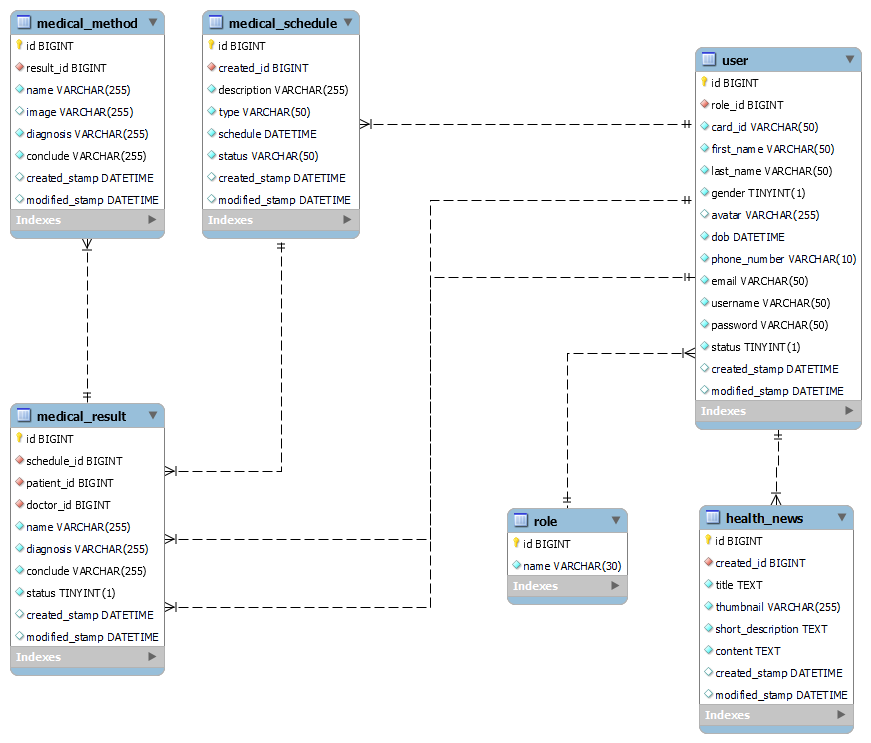


### d. Database queries

* Create new schedule:INSERT INTO medical\_schedule (created\_id, description, schedule, type, status, created\_stamp) " +"VALUES (?, ?, ?, ?, ?, ?)
* Update a schedule :UPDATE medical\_schedule SET description=?, schedule=?, type=?, modified\_stamp=? WHERE id =?
* Find schedule by id:SELECT \* FROM medical\_schedule WHERE id= ?
* Delete schedule:DELETE FROM medical\_schedule WHERE id = ?
* Find schedule by createdId:SELECT \* FROM medical\_schedule WHERE created\_id= ?
* Find all schedule:SELECT \* FROM medical\_schedule
* Update Status of schedule:UPDATE medical\_schedule SET status = ? WHERE id = ?

# III. Database Design

## 1. Database Schema



## 2. Table Description

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| 01 | User | Store information about all of user include admin, doctor, receptionist, patient  CREATE TABLE IF NOT EXISTS `user` (  `id` BIGINT NOT NULL PRIMARY KEY AUTO\_INCREMENT,  `role\_id` BIGINT NOT NULL,  `card\_id` VARCHAR(50) NOT NULL,  `first\_name` NVARCHAR(50) NOT NULL,  `last\_name` NVARCHAR(50) NOT NULL,  `gender` BOOLEAN NOT NULL,  `avatar` NVARCHAR(255) NULL,  `dob` DATETIME NOT NULL,  `phone\_number` VARCHAR(10) NOT NULL,  `email` VARCHAR(50) NOT NULL,  `username` VARCHAR(50) NOT NULL,  `password` VARCHAR(50) NOT NULL,  `status` BOOLEAN NOT NULL,  `created\_stamp` DATETIME,  `modified\_stamp` DATETIME  ); |
| 02 | Role | CREATE TABLE IF NOT EXISTS `role` (  `id` BIGINT NOT NULL PRIMARY KEY AUTO\_INCREMENT,  `name` VARCHAR (30) NOT NULL  ); |
|  | Medical\_schedule | CREATE TABLE IF NOT EXISTS `medical\_schedule` (  `id` BIGINT NOT NULL PRIMARY KEY AUTO\_INCREMENT,  `created\_id` BIGINT NOT NULL,  `description` NVARCHAR(255) NOT NULL,  `type` VARCHAR(50) NOT NULL,  `schedule` DATETIME NOT NULL,  `status` VARCHAR(50) NOT NULL,  `created\_stamp` DATETIME,  `modified\_stamp` DATETIME  ); |
|  | Medical\_result | CREATE TABLE IF NOT EXISTS `medical\_result` (  `id` BIGINT NOT NULL PRIMARY KEY AUTO\_INCREMENT,  `schedule\_id` BIGINT NOT NULL,  `patient\_id` BIGINT NOT NULL,  `doctor\_id` BIGINT NOT NULL,  `name` NVARCHAR(255) NOT NULL,  `diagnosis` NVARCHAR(255) NOT NULL,  `conclude` NVARCHAR(255) NOT NULL,  `status` BOOLEAN NOT NULL,  `created\_stamp` DATETIME,  `modified\_stamp` DATETIME  ); |
|  | Medical\_method | CREATE TABLE IF NOT EXISTS `medical\_method` (  `id` BIGINT NOT NULL PRIMARY KEY AUTO\_INCREMENT,  `result\_id` BIGINT NOT NULL,  `name` NVARCHAR(255) NOT NULL,  `image` NVARCHAR(255) NULL,  `diagnosis` NVARCHAR(255) NOT NULL,  `conclude` NVARCHAR(255) NOT NULL,  `created\_stamp` DATETIME,  `modified\_stamp` DATETIME  ); |
|  | health\_news | CREATE TABLE IF NOT EXISTS `health\_news` (  `id` BIGINT NOT NULL PRIMARY KEY AUTO\_INCREMENT,  `created\_id` BIGINT NOT NULL,  `title` TEXT NOT NULL,  `thumbnail` NVARCHAR(255) NOT NULL,  `short\_description` TEXT NOT NULL,  `content` TEXT NOT NULL,  `created\_stamp` DATETIME,  `modified\_stamp` DATETIME  ); |