

**BIRD CLINIC**

**Software Design Specification**

– HoChiMinh, August 2023 –

**Record of changeS**

| **Date** | **A\* M, D** | **In charge** | **Change Description** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[I. Overview 4](#_heading=h.gjdgxs)

[1. Code Packages 4](#_heading=h.30j0zll)

[2. Database Design 4](#_heading=h.1fob9te)

[a. Database Schema 4](#_heading=h.3znysh7)

[b. Table Description 4](#_heading=h.2et92p0)

[II. Code Designs 5](#_heading=h.tyjcwt)

[1. <Feature/Function Name1> 5](#_heading=h.3dy6vkm)

[a. Class Diagram 5](#_heading=h.1t3h5sf)

[b. Class Specifications 5](#_heading=h.4d34og8)

[c. Sequence Diagram(s) 5](#_heading=h.2s8eyo1)

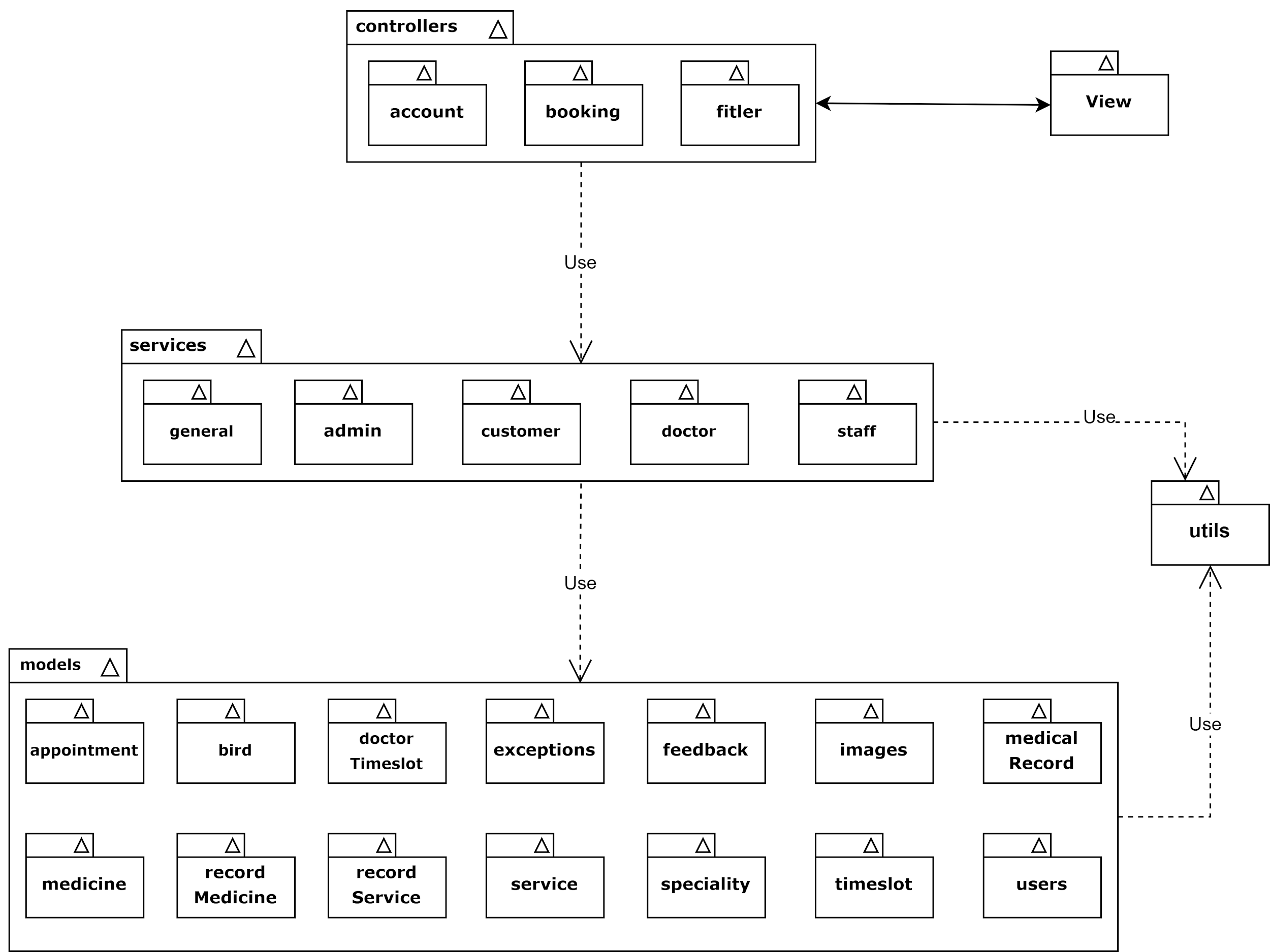
[d. Database queries 6](#_heading=h.17dp8vu)

[2. <Feature/Function Name2> 6](#_heading=h.3rdcrjn)

# I. Overview

## 1. Code Packages

*[Provide the package diagram for each sub-system. The content of this section including the overall package diagram, the explanation, package and class naming conventions in each package. Please see the sample & description table format below (****please note: package names don’t follow Java package naming convention yet****)]*



***Package descriptions***

| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | utils | This package contains utility functions and tools that provide common functionalities or assist in various tasks across the application |
| 02 | models | The models package comprises classes that represent the data entities or objects used in the application. These models typically define the structure, behavior, and relationships of the data |
| 03 | controllers | The controllers package contains the controller classes responsible for handling the incoming requests and coordinating the application's overall flow. These classes often interact with the models, services, and other components to process and respond to user actions |
| 04 | filter | The filters package includes classes or functions that enable filtering and sorting operations on data sets based on specific criteria |
| 05 | services | The services package encapsulates the business logic and operations performed on the data. It provides a layer of abstraction between the controllers and the underlying data models, allowing for modular and reusable code |
| 06 | View | The View package represents the user interface components responsible for displaying the application's content to the users. It often includes templates, layouts, and other elements necessary for rendering the visual presentation |
| 07 | appointment | The appointment package handles the functionalities related to scheduling and managing appointments within the application. It include classes and functions for creating, updating, canceling appointments and related error |
| 08 | bird | The bird package likely represents a specific domain or feature within the application related to birds. It contain classes, functions, or data structures specific to handling bird-related data or operations and related error |
| 09 | doctorTimeslot | The doctorTimeslot package handles the availability and scheduling of doctors' time slots for appointments. It include classes or functions for managing and manipulating the doctor's schedule and related error |
| 10 | exceptions | The exceptions package comprises custom exception classes or error handling mechanisms used throughout the application. It provides a way to handle and communicate exceptional situations or errors that may occur during the execution of the code |
| 11 | feedback | The feedback package deals with collecting and managing user feedback or reviews within the application. It may include classes or functions for capturing feedback, storing it in the database, presenting it to the relevant stakeholders and related error |
| 12 | images | The images package handles the functionalities related to managing and manipulating images within the application. It may include classes or functions for uploading, storing, retrieving, displaying images and related error |
| 13 | medicalRecord | The medicalRecord package likely represents a module or component dedicated to managing and maintaining medical records of patients or users. It may include classes or functions for creating, updating, accessing medical records and related error |
| 14 | medicine | The medicine package handles the functionalities related to managing medications within the application. It include classes or functions for maintaining a catalog of medicines, tracking inventory, managing prescriptions and related error |
| 15 | recordMedicine | The recordMedicine package deals with the association or linkage between medical records and the specific medicines prescribed or administered. It may include classes or functions for recording the prescribed medicines, their dosages in the medical records and related error |
| 16 | recordService | The recordService package likely represents the association or linkage between medical records and the services provided to patients. It include classes or functions for recording the services rendered such as diagnostic tests or treatments, within the medical records and related error |
| 17 | service | The service package handles the functionalities related to providing various services within the application. It include classes or functions for managing service offerings, pricing and related error |
| 18 | speciality | The speciality package represents a classification or categorization system for doctors or healthcare professionals based on their specialized areas of expertise. It include classes or functions for managing, querying specialities and related error |
| 19 | timeslot | The timeslot package likely contains classes or functions for representing and managing time slots or intervals within the application. It include functionalities for creating, modifying, deleting and related error |
| 20 | user | The user package deals with user-related functionalities and operations within the application. It may include classes or functions for user registration, authentication, authorization, profile management and related error. |
| 21 | account |  |
| 22 | admin | The admin package represents the administrative functionalities and operations within the application. It include classes or functions for managing user roles, permissions, and system settings accessible only to administrators |
| 23 | customer | The customer package likely deals with functionalities and operations specific to customer or user interactions within the application. It may include classes or functions for handling customer-related tasks, such as booking appointments, accessing services, and providing feedback |
| 24 | doctor | The doctor package represents the functionalities and operations specific to doctors or healthcare professionals within the application. It may include classes or functions for managing doctor profiles, checking medical record and other related task |
| 25 | staff | The staff package handles the functionalities related to staff members or employees within the application. It may include classes or functions for managing staff profiles, roles, services, blog |
| 26 | booking | The booking package deals with functionalities related to the booking process within the application. It may include classes or functions for managing and coordinating the booking of appointments, services, or other resources |
| 27 | general | The general package encompasses a collection of commonly used services and functionalities that are not specific to any particular domain or feature within the application. It serves as a utility package providing a set of general-purpose tools and resources |

## 2. Database Design

### a. Database Schema

Link to schema:[Database Schema - dbdiagram.io](https://dbdiagram.io/d/64a39beb02bd1c4a5e72c010)



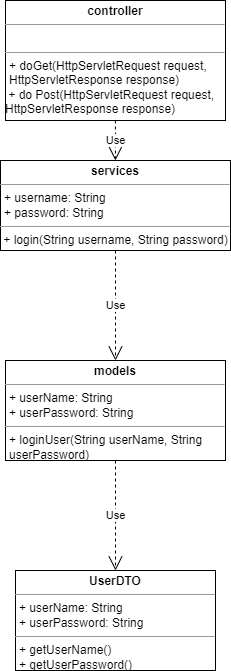
### b. Table Description

| **No** | **Table** | **Description** |
| --- | --- | --- |
| 01 | Images | Store information about images, including their unique ID and URL name |
| 02 | Speciality | Represents different specialities or area of expertise, including a unique ID and the name of the speciality |
| 03 | TimeSlot | Represents available time slots for appointments, including a unique ID and the day of the week |
| 04 | Users | Contains user-related information, such as user ID, username, password, fullname, contact details, gender, email, user role and status. It also references the Images table for the user’s profile picture |
| 05 | Doctor | Contains information about doctors, including their unique ID, age, years of experience, academic title and degree. It also references the Speciality table for the doctor’s associated speciality |
| 06 | Customer | Store details about customer including their unique ID, date of birth, and address |
| 07 | RecordServices | Represents the services provided for a specific medical record, establishing a relationship between the MedicalRecord table and the Service table |
| 08 | Medical Record | Store medical records related to appointments in the Appointment table, including a unique ID, associated appointment, bird, doctor, record time, diagnosis, treatment duration and doctor’s notes |
| 09 | Appointment | Contains details about appointments, including a unique ID, associated bird referenced in the Bird tables, associated doctor referenced in the Doctor table, time slot referenced in the TimeSlot table, associated service referenced in the Service table, appointment time, notes, payment status and appointment status |
| 10 | Service | Represents various services offered, such as treatments or procedures. It includes a unique ID, associated speciality referenced in the Speciality table, service name and price |
| 11 | Bird | Store information about birds, including a unique ID, associated customer, image, fullname, gender, breed, band or chip information, weight, sexing method, medical history, hatching date and feather color |
| 12 | Feedback | Stores feedback received for appointments, including a uniqueID, associated appointment referenced in the Appointment table, feedback content, title, feedback time, and rating |
| 13 | Blog | Contains information about blog posts, including a unique ID, title, upload datetime, category and blog content |
| 14 | Medicine | Stores information about medicines, including a unique ID, medicine name and price |
| 15 | RecordMedicine | Represents the services prescribed for a specific medical record, establishing a relationship between the MedicalRecord table and the Service table |
| 16 | DoctorTimeSlot | Store information about the doctor’s available time slot, establishing a relationship between the Doctor table and the TimeSlot table |

# II. Code Designs

## 1. Common/Login

### a. Class Diagram

**

### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | req.getRequestDispatcher("./home.jsp").forward(req, resp); |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | Call itself |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | loadHomePage() | loads and handles the home page request |
| 2 | login(String userID,String password) | user login |

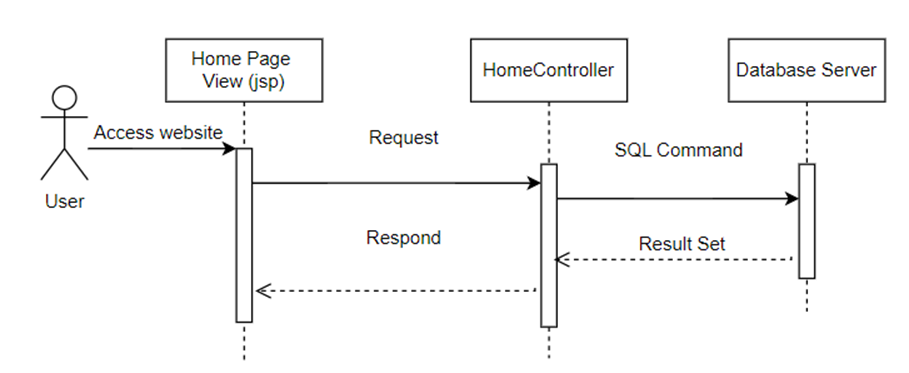
* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | loginUser(String userName, String userPassword) | creates a new user in the database |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |
| 02 | checkPassword(String password) | check the password of the account |

### c. Sequence Diagram(s)



### d. Database Queries

//login

SELECT \*

FROM Users

WHERE userName = '<userName>' AND userPassword = '<userPassword>'

//retrieve the user profile from the database based on the provided userID

SELECT \*

FROM Users

WHERE userID = 'userID\_value';

//creates a new user in the database

INSERT INTO Users (userID, imageID, userName, userPassword, fullName, phoneNumber, gender, email, userRole, status\_)

VALUES ('userID\_value', 'imageID\_value', 'userName\_value', 'userPassword\_value', 'fullName\_value', 'phoneNumber\_value', 'gender\_value', 'email\_value', 'userRole\_value', 'status\_value');

//Update the user’s password in the database

UPDATE Users

SET userPassword = 'new\_userPassword\_value'

WHERE userID = 'userID\_value';

//update the user’s profile in the database

UPDATE Users

SET fullName = 'new\_fullName\_value',

phoneNumber = 'new\_phoneNumber\_value',

gender = 'new\_gender\_value',

email = 'new\_email\_value'

WHERE userID = 'userID\_value';

//retrieve the user profile from the database based on the provided email and username

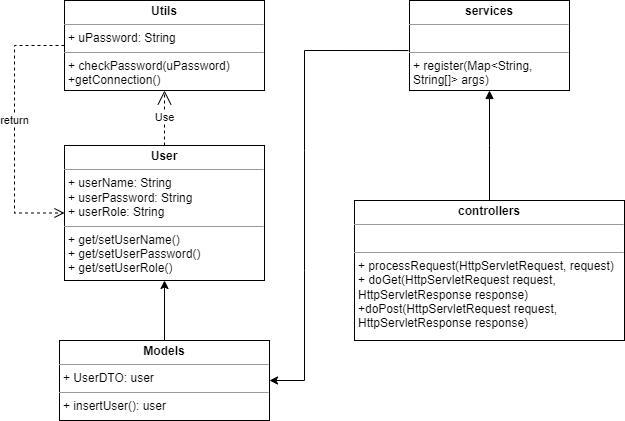
SELECT \*

FROM Users

WHERE email = 'email\_value' AND userName = 'userName\_value';

## 2. Common/Register

### a. Class Diagram



### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | register(Map<String, String[]> args) | Update the user’s password |

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | readUserByEmailUserName(email, username) | Check if the email or username is duplicated or not |
| 02 | insertCustomer(insertCustomer(CustomerDTO customer) | Insert the customer to the database |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | checkPassword(uPassword) | Check the user’s password |
| 02 | getConnection() | Make the connection to the database |

### c. Sequence Diagram(s)

### d. Database Queries

//Check if the email or username is duplicated or not

SELECT \*

FROM Users

WHERE email = 'email\_value';

//Insert the customer to the database

SELECT \*

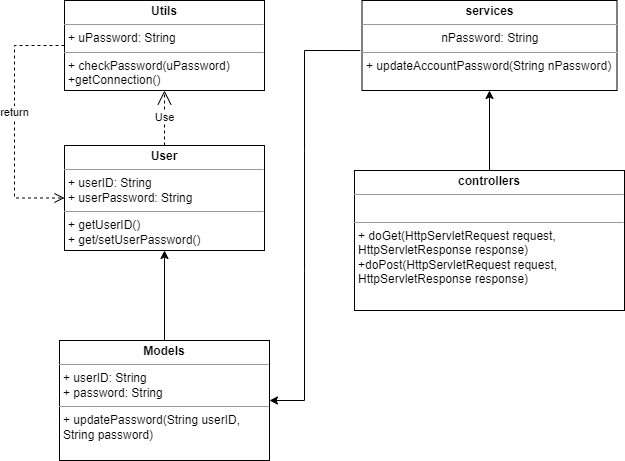
FROM Users

WHERE userName = 'username\_value';

### 

## 3. Common/Change password

### a. Class Diagram



### b. Class Specifications

* Controller

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | redirect to the ChangePassword Page |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | get the current password and the new password |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateAccountPassword(String nPassword) | Updates the user’s password in the Database |
| 02 | getCurrentUser() | Check if this current user exist or not |

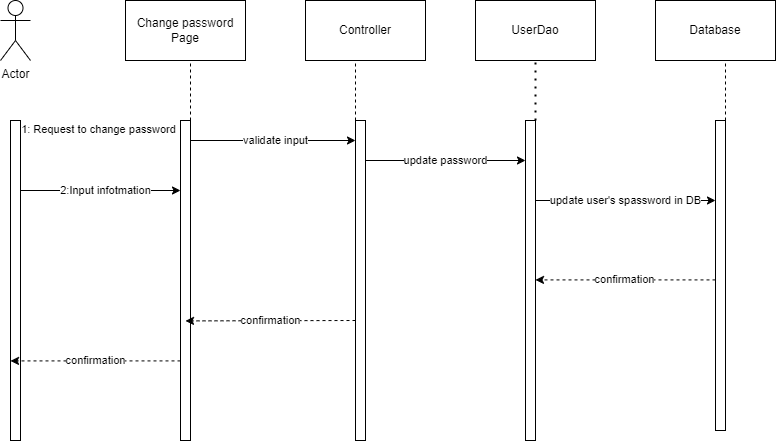
* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updatePassword(String userID, String password) | Updates the user’s password in the Database |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |
| 02 | checkPassword(nPassword) | Check the format of the password, also check if password is duplicate or not |

### c. Sequence Diagram(s)



### d. Database Queries

//update password

“UPDATE password

SET userPassword = 'new\_password'

WHERE userName = 'userName'”

### 

//Check if this current user exist or not

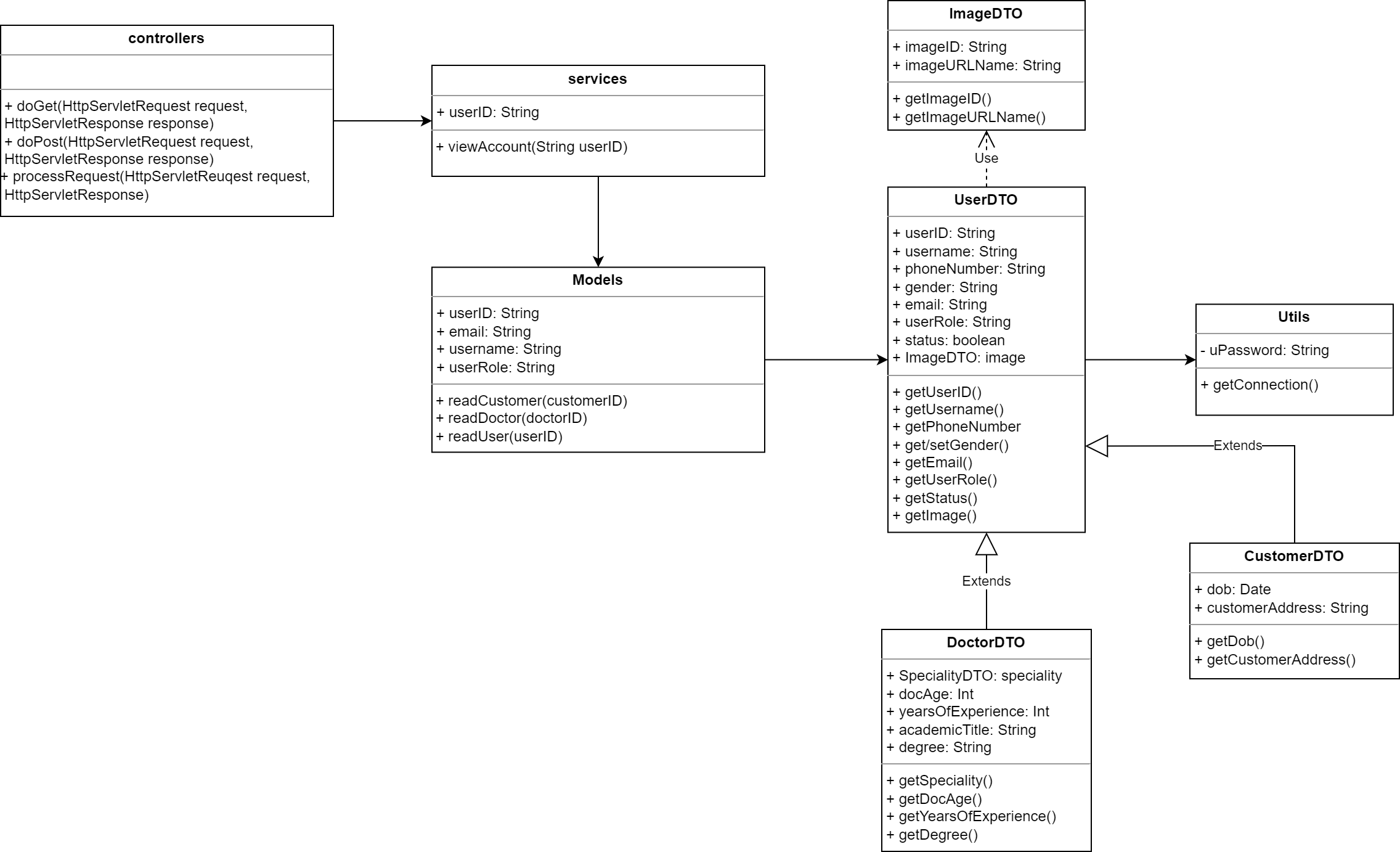
SELECT \*

FROM Users

WHERE userID = 'userID\_value';

### 4. Common/View User Profile

### a. Class Diagram



### b. Class Specifications

* Controller

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | redirect to the ChangePassword Page |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | get the current password and the new password |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | viewAccount(String userID) | View the user’s profile |

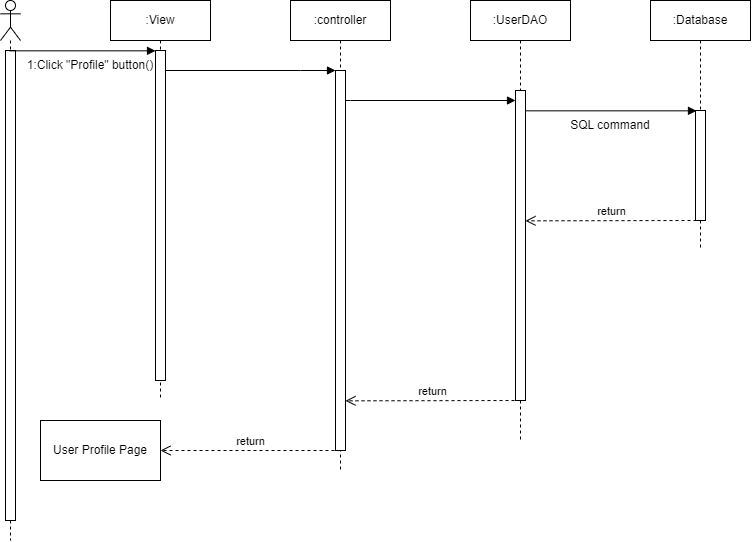
* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | readCustomer(String customerID) | Get the customer’s information based on customerID |
| 02 | readDoctor(String doctorID) | Get the doctor’s information based on doctorID |
| 03 | readUser(String userID) | Get the user’s information based on userID |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |

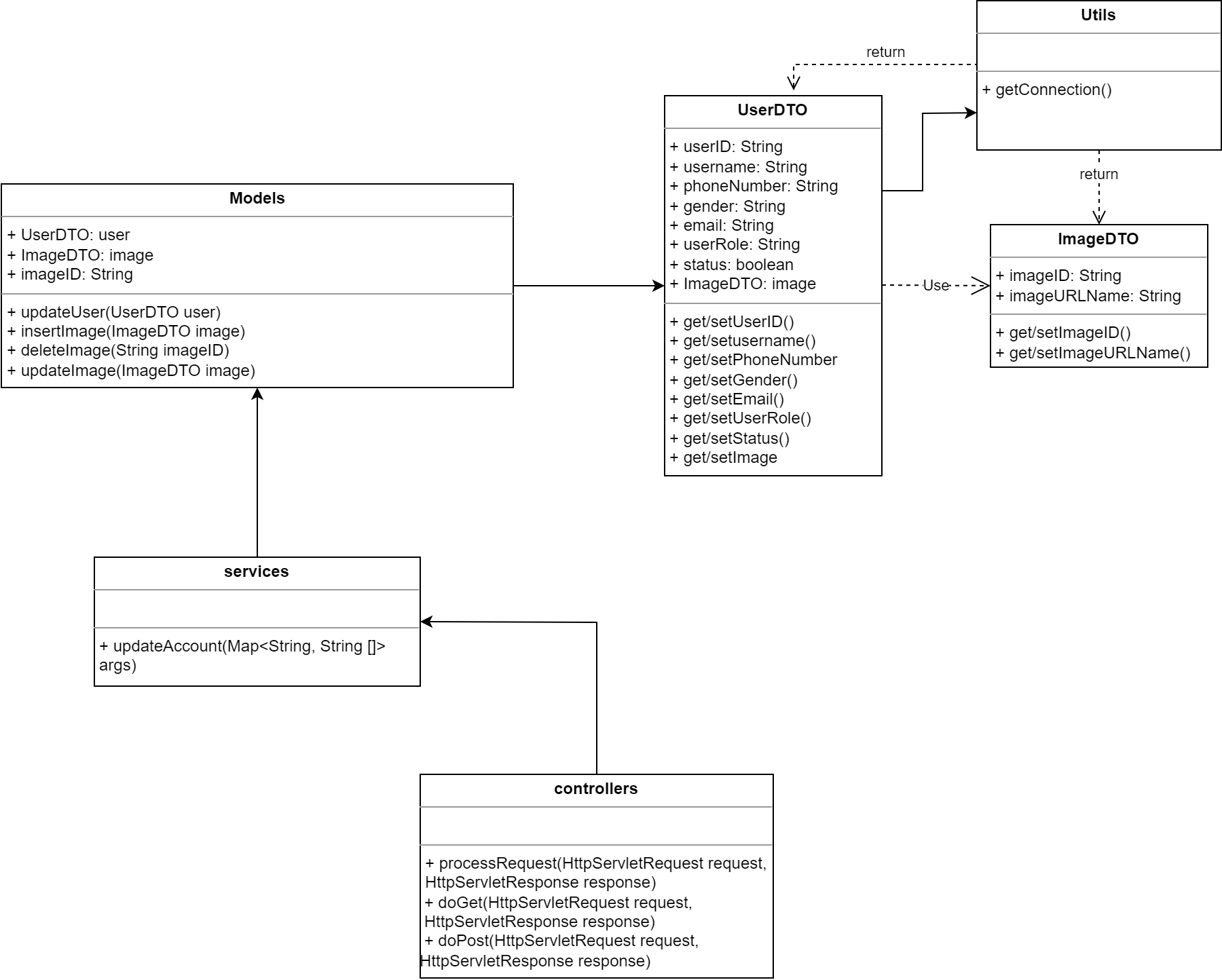
### c. Sequence Diagram(s)



### d. Database Queries

### 5. Common/Manage profile

### a. Class Diagram



### b. Class Specifications

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateUser(UserDTO user) | update the user’s information |
| 02 | insertImage(ImageDTO image) | allow customer/doctor to insert a new image |
| 03 | deleteImage(String imageID) | allow customer/doctor to delete an image |
| 04 | updateImage(ImageDTO image) | allow customer/doctor to update an image |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateAccount(Map<String, String[]> args) | update the information of the account |

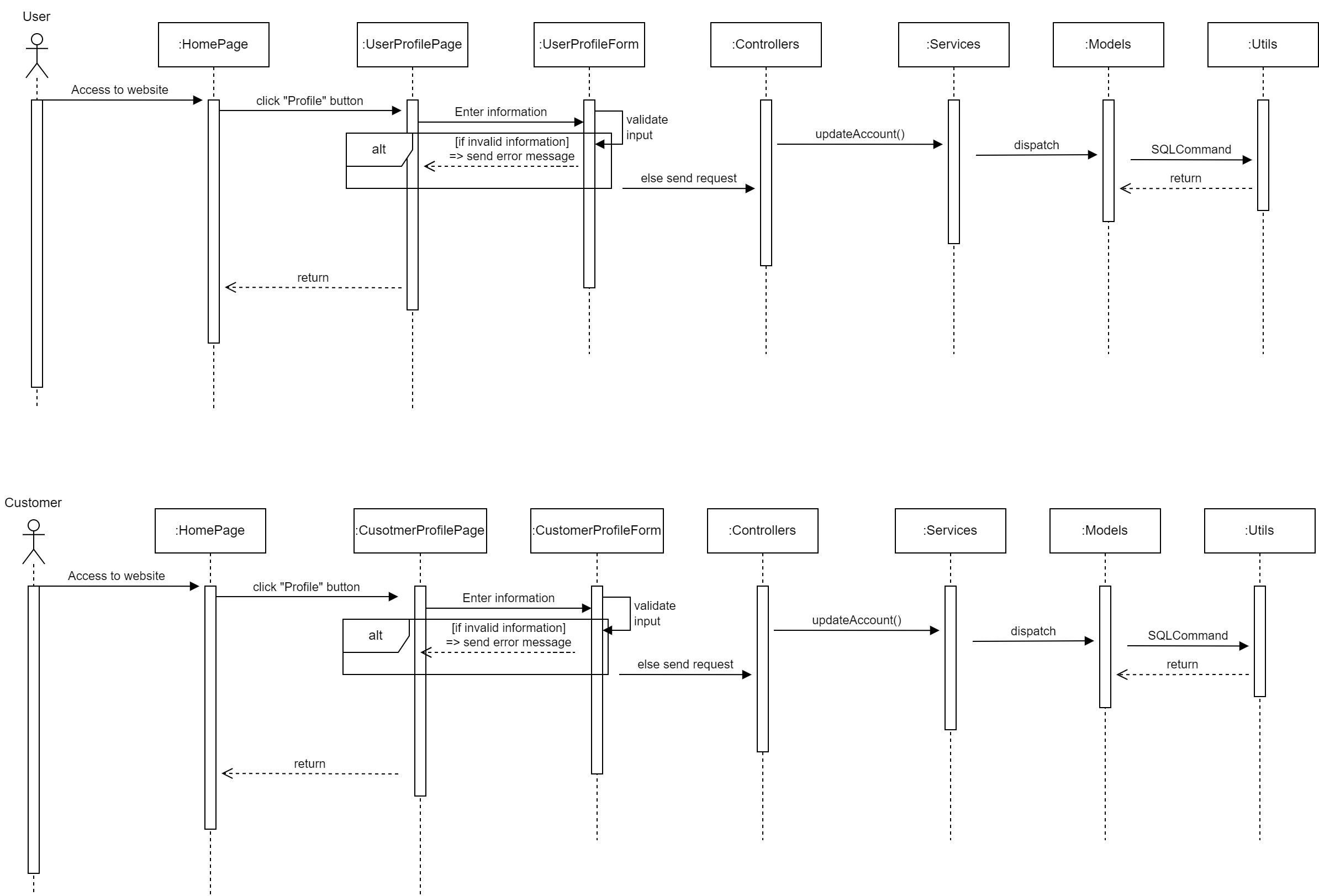
* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |

### c. Sequence Diagram(s)



### d. Database Queries

//check for duplicate emails

“SELECT email, COUNT(\*) AS count

FROM YourTableName

GROUP BY email

HAVING COUNT(\*) > 1”

//check for duplicate usernames:

“SELECT username, COUNT(\*) AS count

FROM YourTableName

GROUP BY username

HAVING COUNT(\*) > 1”

UPDATE Users

“SET fullName = '<newFullName>', phoneNumber = '<newPhoneNumber>', gender = '<newGender>', email = '<newEmail>'

WHERE userID = '<userID>'”

//update the information of the account

UPDATE Users

SET imageID = 'new\_imageID\_value',

userName = 'new\_userName\_value',

userPassword = 'new\_userPassword\_value',

fullName = 'new\_fullName\_value',

phoneNumber = 'new\_phoneNumber\_value',

gender = 'new\_gender\_value',

email = 'new\_email\_value',

userRole = 'new\_userRole\_value',

status\_ = 'new\_status\_value'

WHERE userID = 'userID\_value';

//allow customer/doctor to insert a new image

INSERT INTO Images (imageID, imageURLname)

VALUES ('imageID\_value', 'imageURLname\_value');

//allow customer/doctor to delete an image

DELETE FROM Images

WHERE imageID = 'imageID\_value';

// allow customer/doctor to update an image

UPDATE Images

SET imageURLName = 'new\_imageURLName\_value'

WHERE imageID = 'imageID\_value';

//Get the information of the customer by customerID

SELECT \*

FROM Customer

WHERE customerID = 'customerID\_value';

//Get the user information based on userRole

SELECT \*

FROM Users

WHERE userRole = 'userRole\_value';

//Get the current user information

SELECT \*

FROM Users

WHERE userID = 'current\_userID';

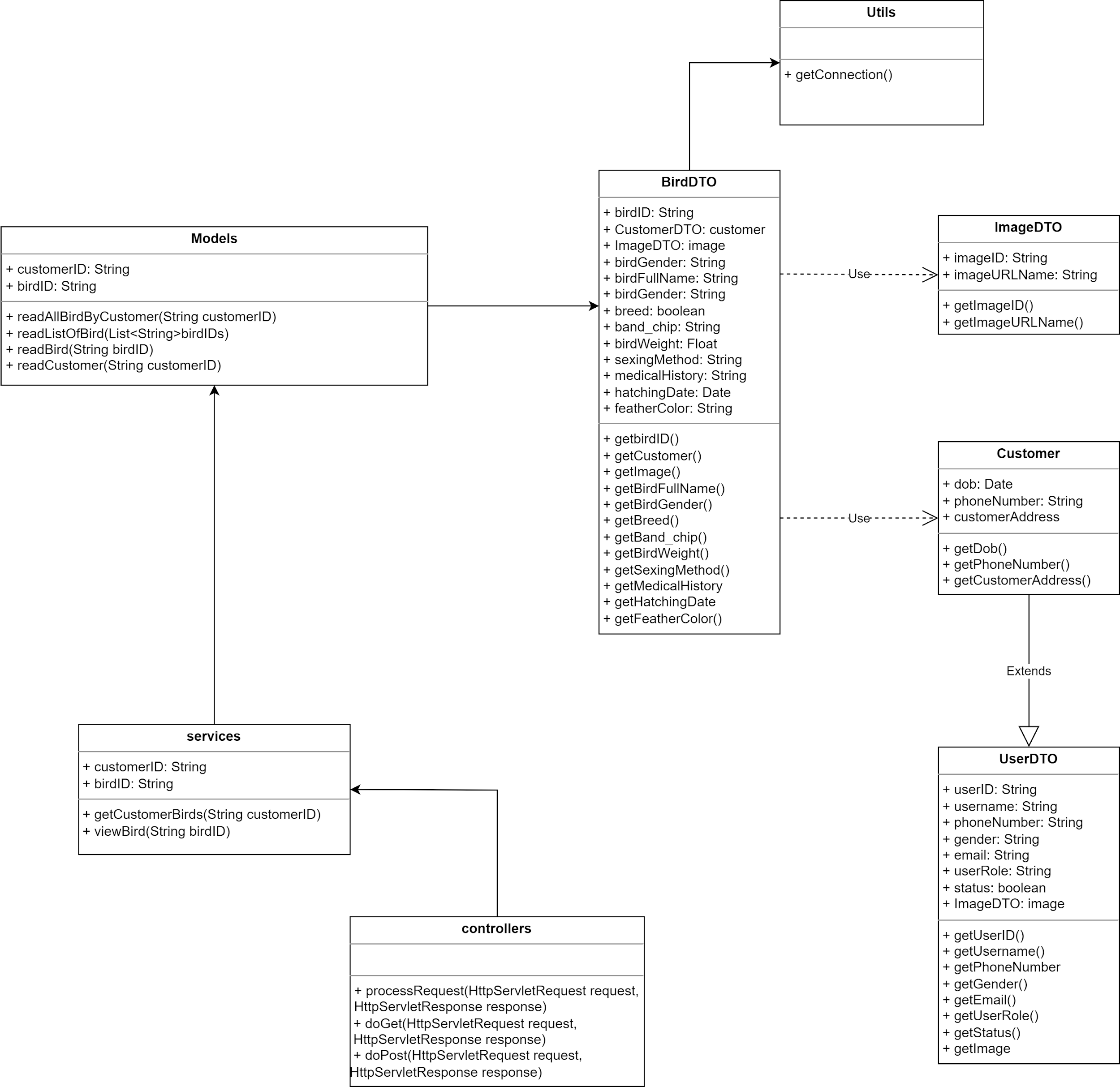
//Check the role for each user

SELECT userID, userRole

FROM Users;

### 6. Customer/View Bird

### a. Class Diagram



### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

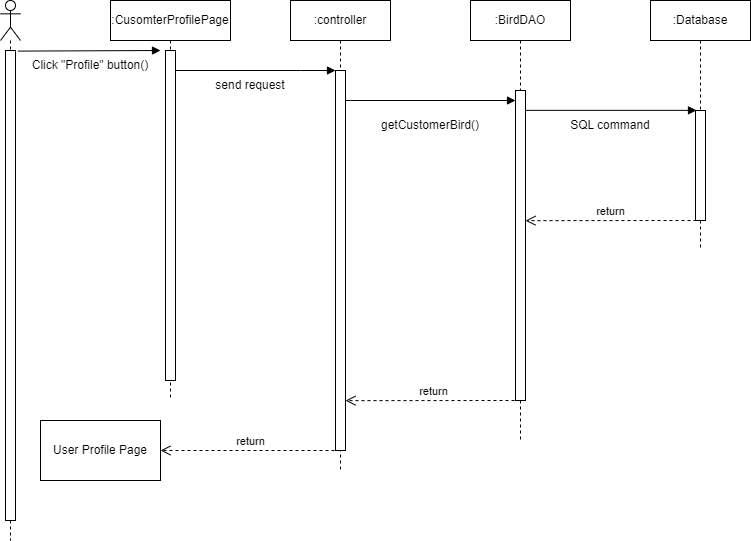
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getCustomerBirds(String customerID) | Get all of the customer’s birds based on customerID |
| 02 | viewBird(String birdID) | View the customer’s bird based on birdID |

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | readAllBirdByCustomer(String customerID) | Get all of the customer’s birds information based on customerID |
| 02 | readBird(String birdID) | Get the bird’s information based on birdID |
| 03 | readCustomer(String customerID) | Get the customer based on customerID |
| 04 | readImage(String imageID) | Get the image based in imageID |

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |

### c. Sequence Diagram(s)



### d. Database Queries

//Get all of the customer’s birds information based on customerID

SELECT \*

FROM Bird

WHERE customerID = 'customerID\_value';

// Get the bird's information based on birdID

SELECT \*

FROM Bird

WHERE birdID = 'your\_birdID';

// Get the customer based on customerID

SELECT \*

FROM Customer

WHERE customerID = 'your\_customerID';

// Get the image based on imageID

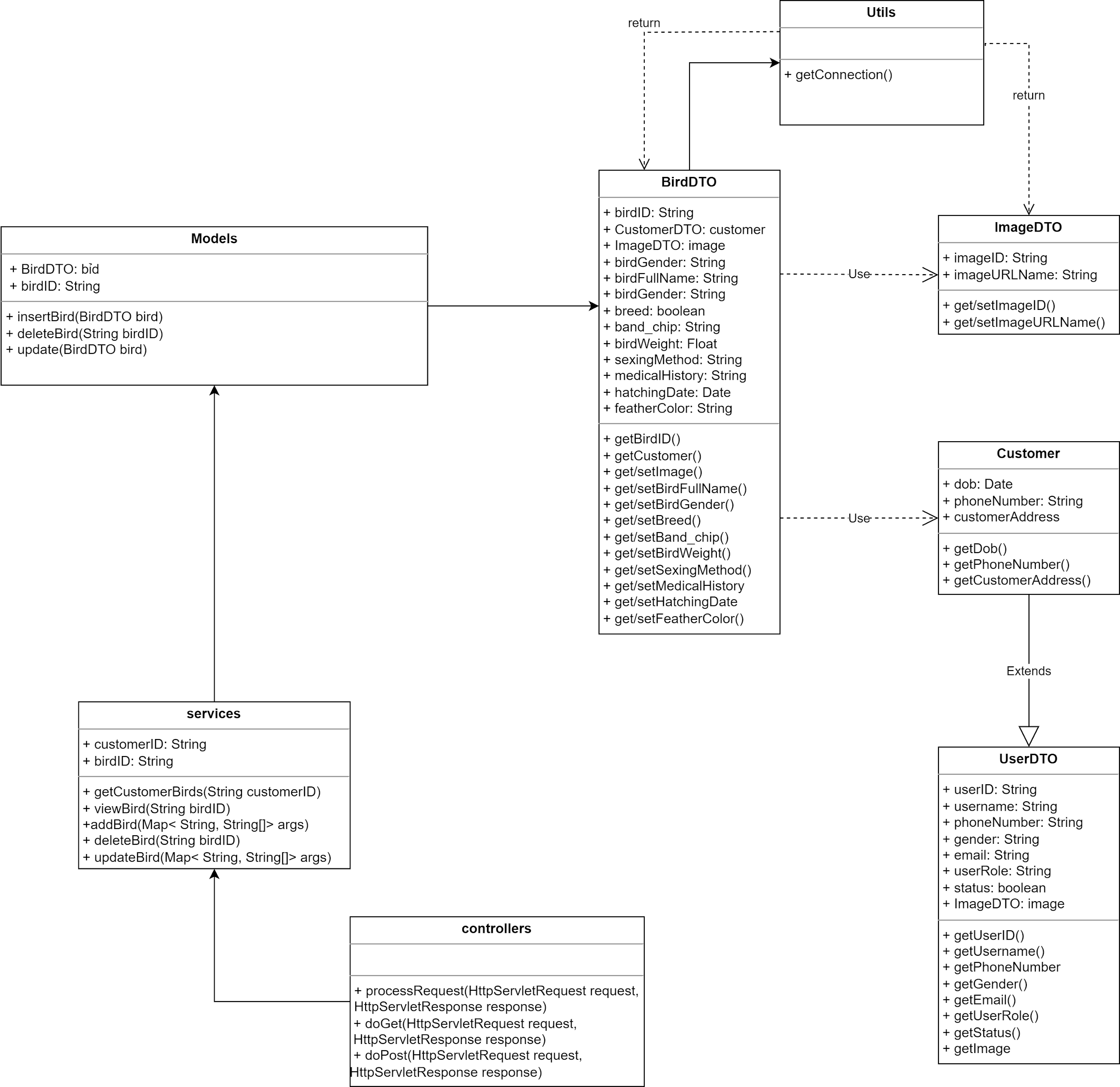
SELECT \*

FROM Images

WHERE imageID = 'your\_imageID';

### 7. Customer/Manage Bird

### a. Class Diagram



### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | addBird(Map<String, String[]> args) | Update a bird to the database |
| 02 | getCustomerBirds(String customerID) | Get all of the customer’s birds based on customerID |
| 03 | deleteBird(String birdID) | Delete a bird based on birdID |
| 04 | updateBird(Map<String, String[]> args) | Update a bird to the database |
| 05 | viewBird(String birdID) | View the customer’s bird based on birdID |

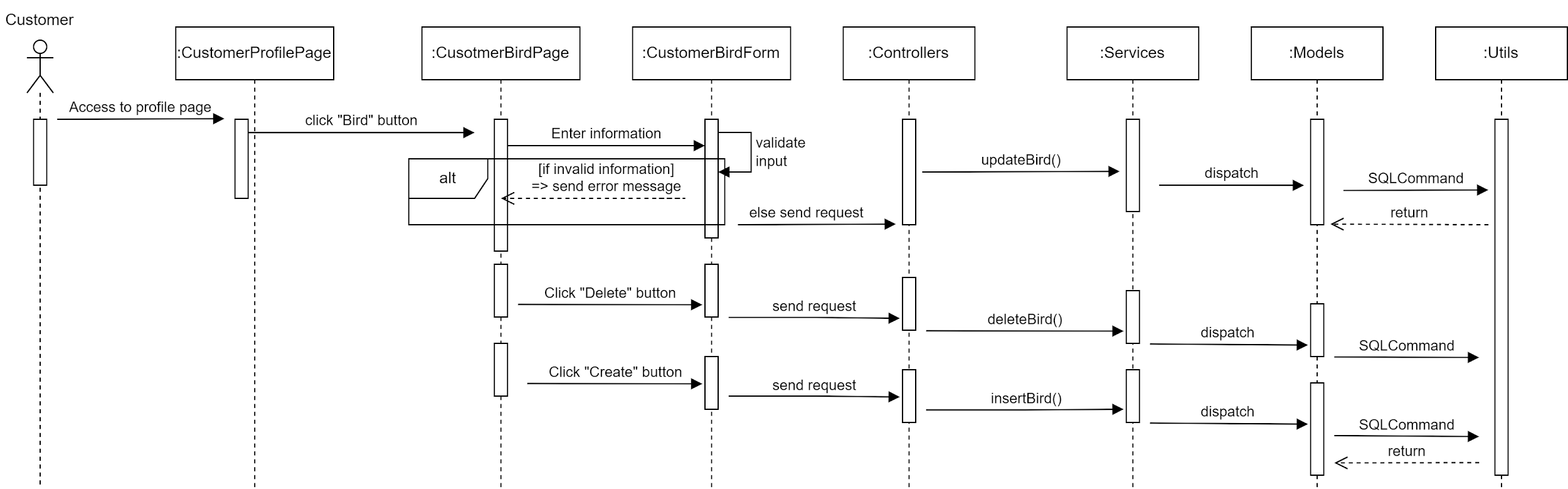
* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 03 | insertBird(BirdDTO bird) | Insert a new bird to the database |
| 04 | deleteBird(String BirdID) | Delete a bird in the database |
| 05 | updateBird(BirdDTO bird) | Update a bird in the database |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |

### c. Sequence Diagram(s)



### d. Database Queries

//Check if a bird exists based on the birdID

SELECT COUNT(\*) AS BirdCount

FROM Bird

WHERE birdID = 'birdID';

//Get the bird based on the birdID

SELECT \*

FROM Bird

WHERE birdID = 'birdId';

//Get a list of birds belongs to the customer

SELECT \*

FROM Bird

WHERE customerID = 'customerId';

//Insert a new bird to the database

“INSERT INTO Bird(birdID, customerID,imageID, medicalHistory, birdGender, breed, band\_chip, birdWeight, sexingMethod, birdFullname, hatchingDate, featherColor)

VALUES

('new\_birdID' , 'new\_customerID' , 'new\_imageID', N'new\_medicalHistory' ,'new\_birdGender','new\_breed', 'new\_band\_chip', new\_birdWeight,N'new\_sexingMethod', N'new\_birdFullname', 'new\_yyyy-mm-dd', N'new\_featherColor')”

//Delete a bird in the database

DELETE FROM Bird

WHERE birdID = 'your\_bird\_id';

//Update a bird in the database

UPDATE Bird

SET birdFullname = N'new\_birdFullname',

birdGender = 'new\_birdGender',

breed = N'new\_breed',

band\_chip = 'new\_band\_chip',

birdWeight = new\_birdWeight,

sexingMethod = 'new\_sexingMethod',

medicalHistory = N'new\_medicalHistory',

hatchingDate = 'new\_hatchingDate',

featherColor = N'new\_featherColor'

WHERE birdID = 'birdID';

//get a bird base on customerId

SELECT \*

FROM Bird

WHERE customerID = 'customerID\_value';

// Get the image based on imageID

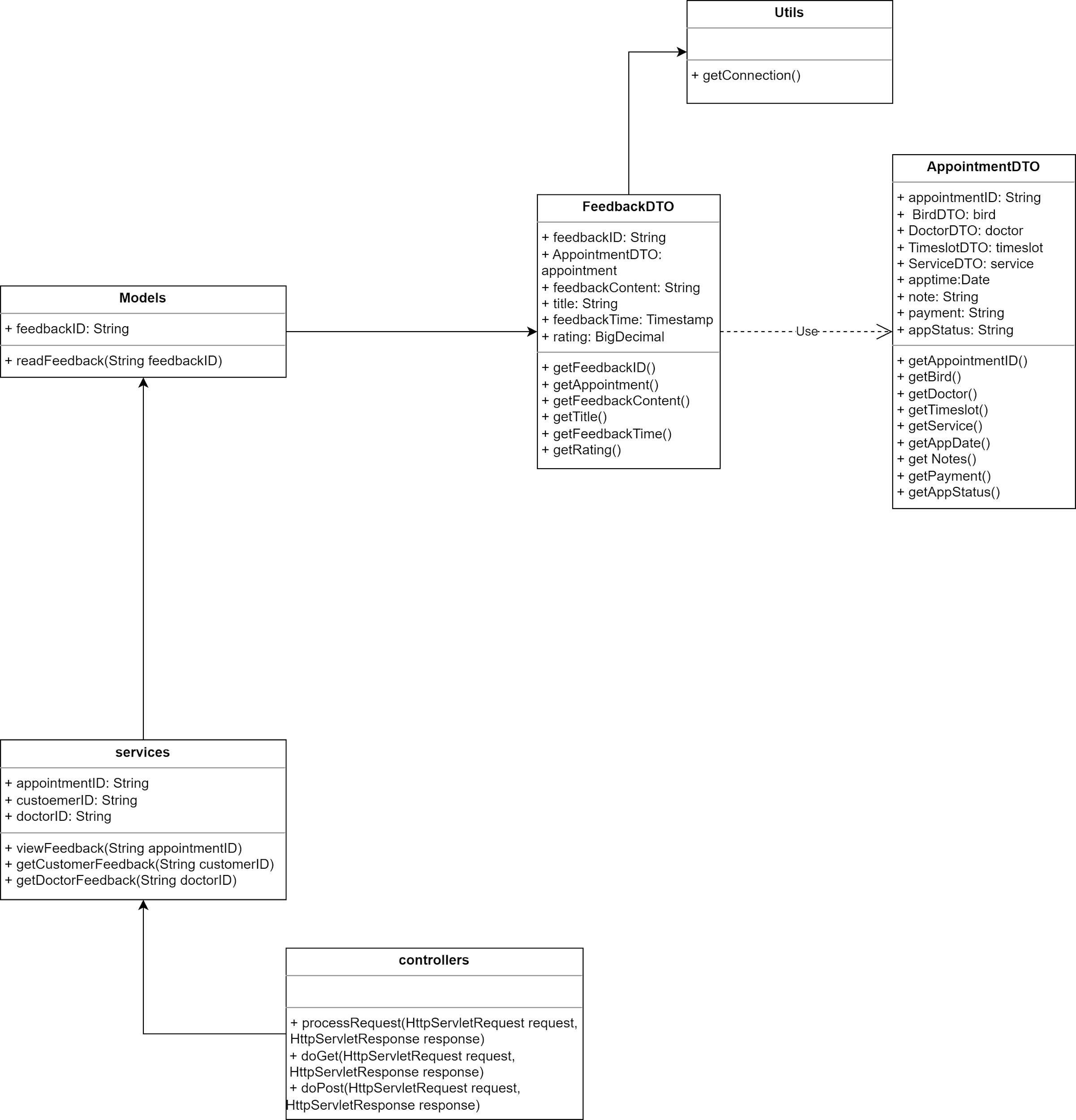
SELECT \*

FROM Images

WHERE imageID = 'your\_imageID';

### 9. Common/Feedback Appointment

### a. Class Diagram



### b. Class Specifications

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | readFeedback(String feedbackID) | read the feedback using feedback id |
| 02 | readAppointment(String appointmentID) | Get the feedback information based on appointmentID |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | viewFeedback(String appointmentID) | view the feedback related to the appointment |
| 02 | getCustomerFeedback(String customerID) | Get the feedback belong to the customer based on customerID |
| 03 | getDoctorFeedback(String doctorID) | Get the feedback belong to the doctor based on doctorID |

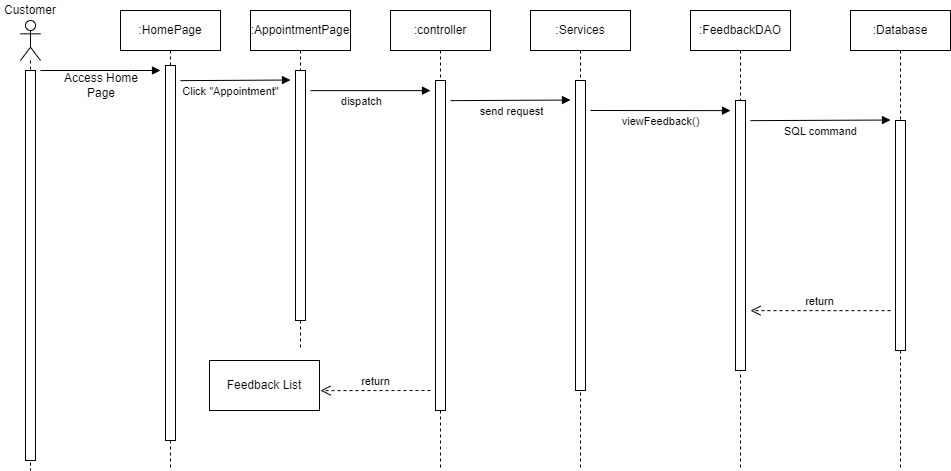
* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Database

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |

### c. Sequence Diagram(s)



### d. Database Queries

// View the feedback related to the appointment

SELECT \*

FROM Feedback

WHERE appointmentID = 'your\_appointmentID';

### 10. Staff/Manage Feedback

### a. Class Diagram



### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | insertFeedback(FeedbackDTO feedback) | Insert the new feedback to the database |
| 02 | updateFeedback(FeedbackDTO feedback) | Update a feedback to the database |
| 03 | deleteFeedback(String feedbackID) | Delete a feedback from the database |

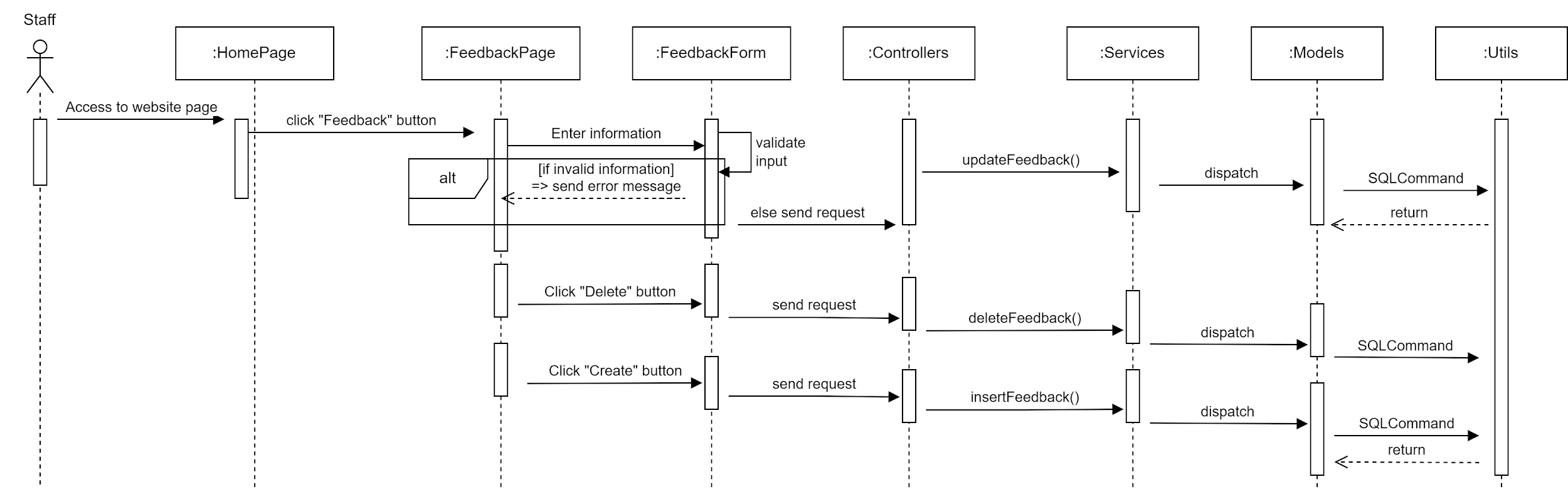
* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | viewFeedback(String appointmentID) | view the feedback related to the appointment |
| 02 | getCustomerFeedback(String customerID) | Get the feedback belong to the customer based on customerID |
| 03 | getDoctorFeedback(String doctorID) | Get the feedback belong to the doctor based on doctorID |

* Database

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |

### c. Sequence Diagram(s)



### d. Database Queries

//insert a new feedback to the database

### “INSERT INTO Feedback (feedbackID, customerID, doctorID, rating, comment)

### VALUES ('feedbackID\_value', 'customerID\_value', 'doctorID\_value', rating\_value, 'comment\_value')”

### 

//delete a feedback from the database

DELETE FROM Feedback

WHERE feedbackID = 'feedbackID\_value';

//update a feedback from the database

UPDATE Feedback

SET rating = new\_rating, comment = 'new\_comment'

WHERE feedbackID = 'feedbackID\_value';

// Get the feedback information based on feedbackID

SELECT \*

FROM Feedback

WHERE feedbackID = 'your\_feedbackID';

// Get the appointment information based on appointmentID

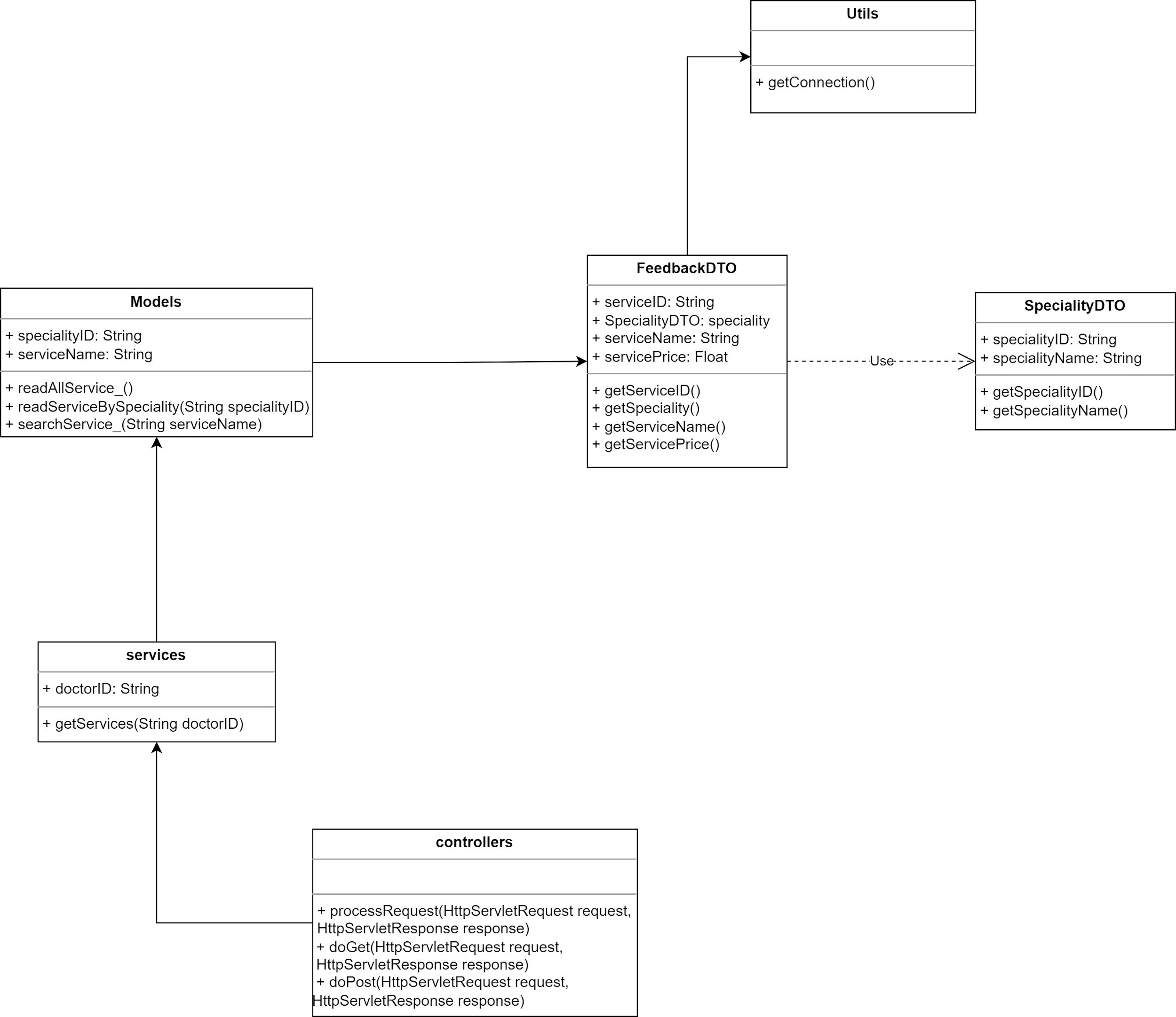
SELECT \*

FROM Appointment

WHERE appointmentID = 'your\_appointmentID';

### 11. Common/Service List

### a. Class Diagram



### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getServices(String doctorID) | Get all the services based on doctorID |

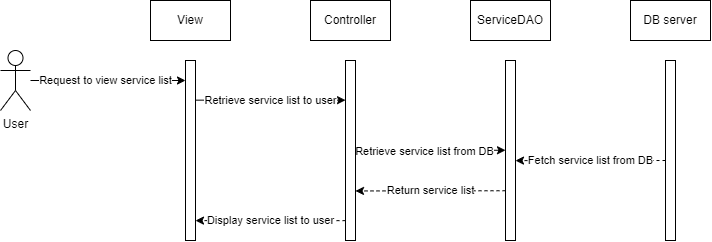
* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | readAllService\_() | Get all the service assigned to the doctor |
| 02 | readServiceBySpeciality(String specialityID) | Get all the service based on specialityID |
| 03 | searchService\_(String serviceName) | Search service in the clinic by serviceName |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | retrieves the service list from Database |

### c. Sequence Diagram(s)



### d. Database Queries

//Get all the services based on doctorID

SELECT \*

FROM Service\_

WHERE doctorID = 'doctorID\_value';

//Get all the service in the clinic

SELECT \*

FROM Service\_

//Search service in the clinic by serviceName

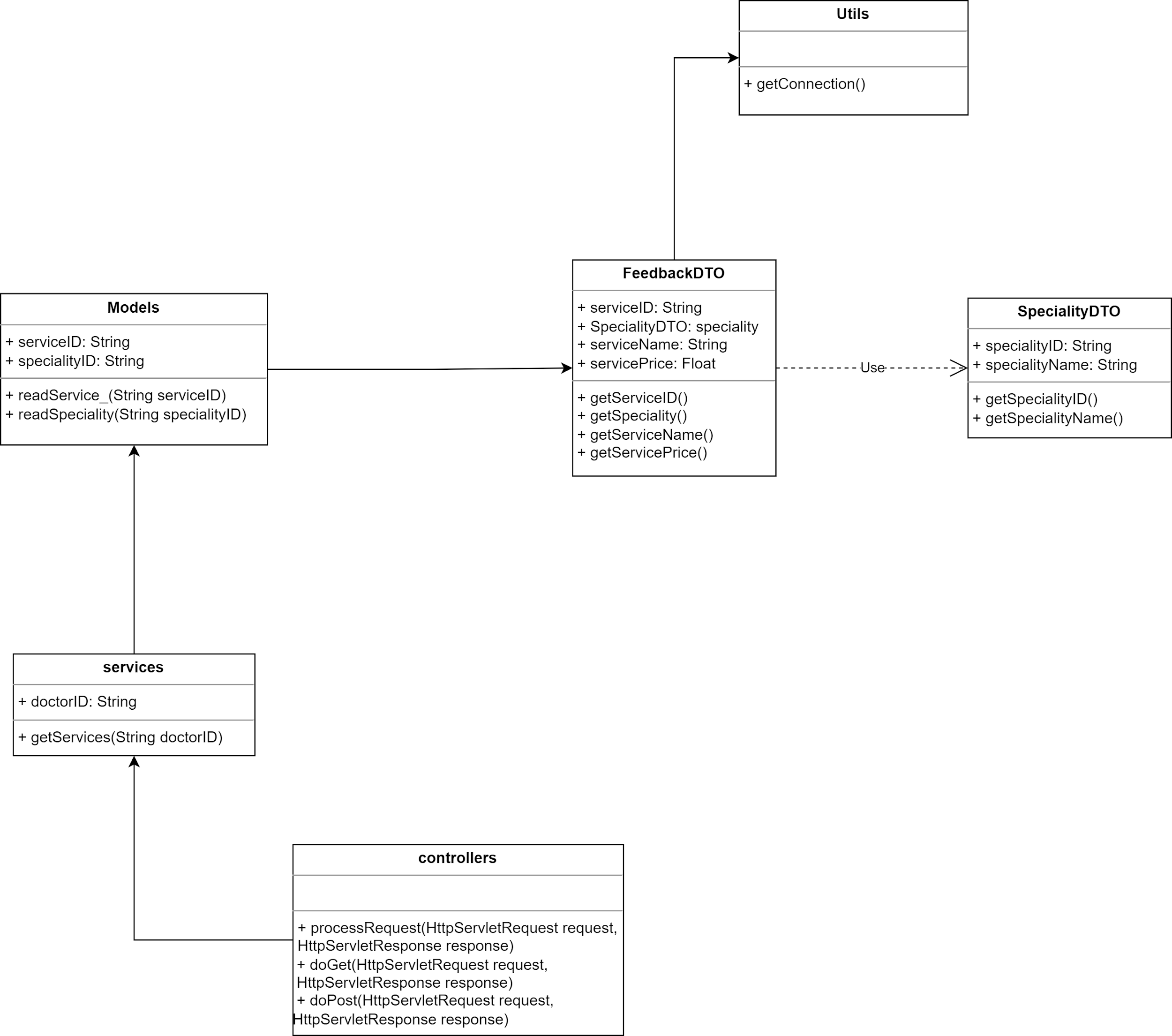
SELECT \*

FROM Service\_

WHERE serviceName LIKE '%service\_name%';

### 12. Common/Service Details

### a. Class Diagram



### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getService(String serviceID) | Get the service based on serviceID |

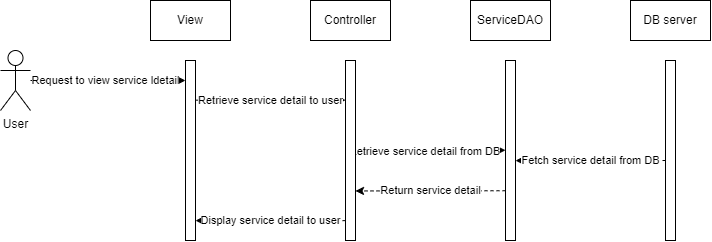
* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | readService\_(String serviceID) | Get the service information by serviceID |
| 02 | readSpeciality(String specialityID) | Get the speciality information based on specialityID |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | retrieves the service list from Database |

### c. Sequence Diagram(s)



### d. Database Queries

//Get all the service information based on serviceID

SELECT serviceName

FROM Service

WHERE serviceID = 'service\_id';

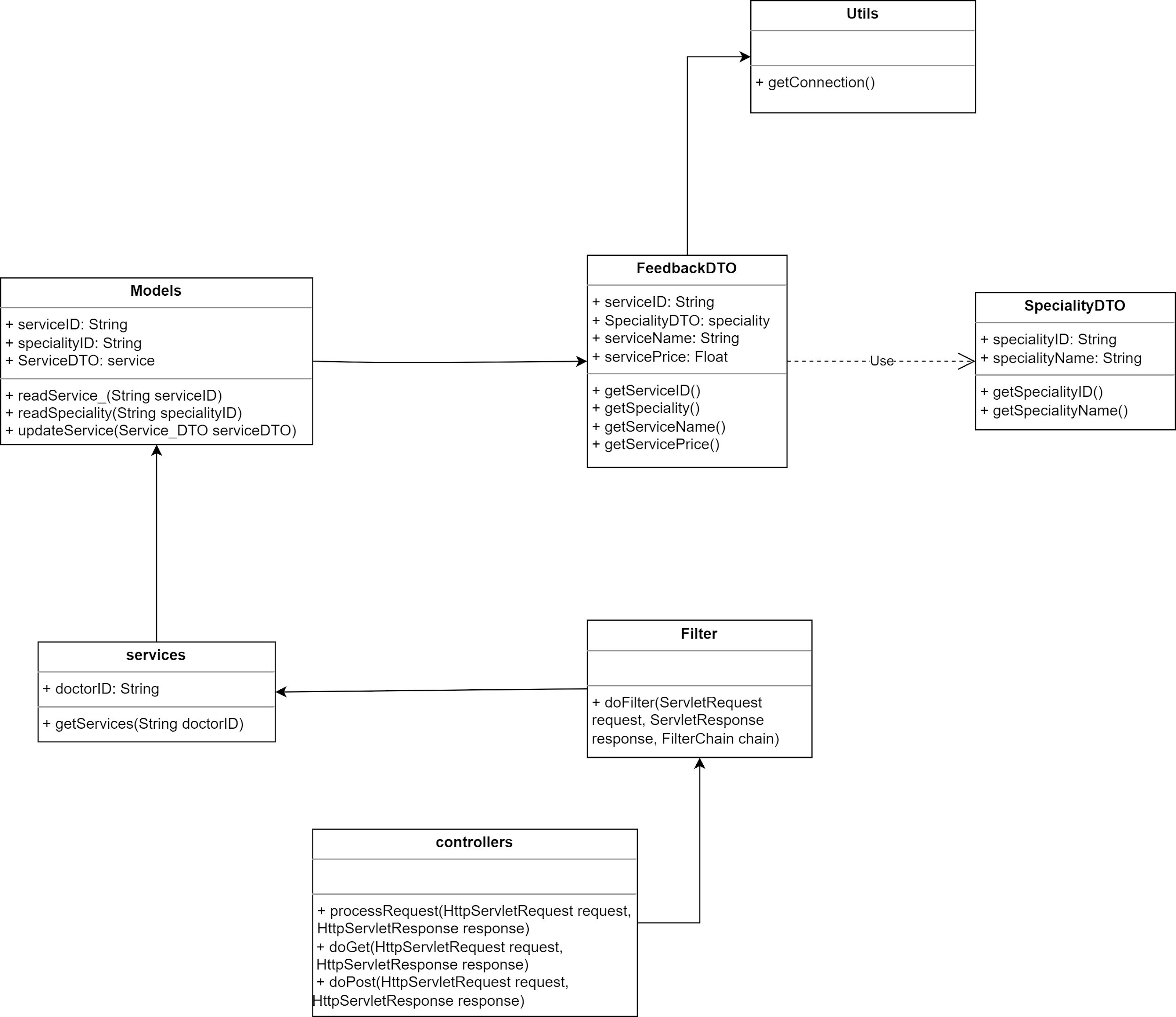
//retrieves the service list from Database

SELECT serviceID, serviceName, serviceDescription, servicePrice

FROM Service;

### 13. Staff/Manage Service

### a. Class Diagram



### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getServices(String doctorID) | Get all the service information by doctor speciality based on doctorID |

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | readService\_(String serviceID) | Get all the service information based on serviceID |
| 02 | updateService(Service\_DTO serviceDTO) | Update a new service to the database |
| 03 | readSpeciality(String specialityID) | Get all the speciality information based on specialityID |

### c. Sequence Diagram(s)

### d. Database Queries

//Get all the service information based on serviceID

SELECT \*

FROM Service

WHERE serviceID = serviceID\_value;

//Update a new service to the database

### UPDATE Service

### SET serviceName = service\_name\_value,

### serviceDescription = service\_description\_value,

### servicePrice = service\_price\_value

### WHERE serviceID = serviceID\_value;

### 

### 14. Common/Blog List

### a. Class Diagram

### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
|  |  |  |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
|  |  |  |

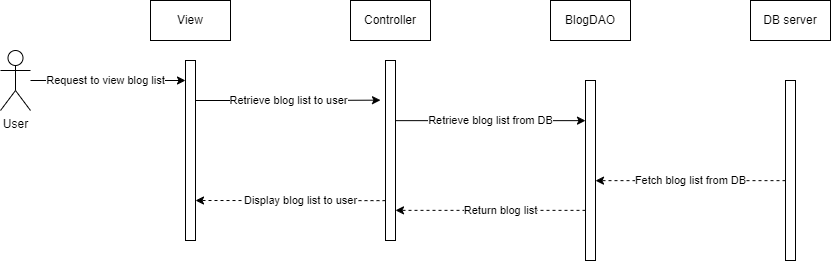
* Database

| **No** | **Method** | **Description** |
| --- | --- | --- |
|  | getConnection() | get the connection to the database |

Blog (Data class)

* title: String
* author: String
* postDate: Date
* comment: String

### c. Sequence Diagram(s)



### d. Database Queries

//retrieves the blog list from the Database

SELECT blogID, title, content, author

FROM Blog;

### 15. Common/Blog Details

### a. Class Diagram

### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

### c. Sequence Diagram(s)

### d. Database Queries

### 16. Staff/Manage Blogs

### a. Class Diagram

### b. Class Specifications

* Controllers

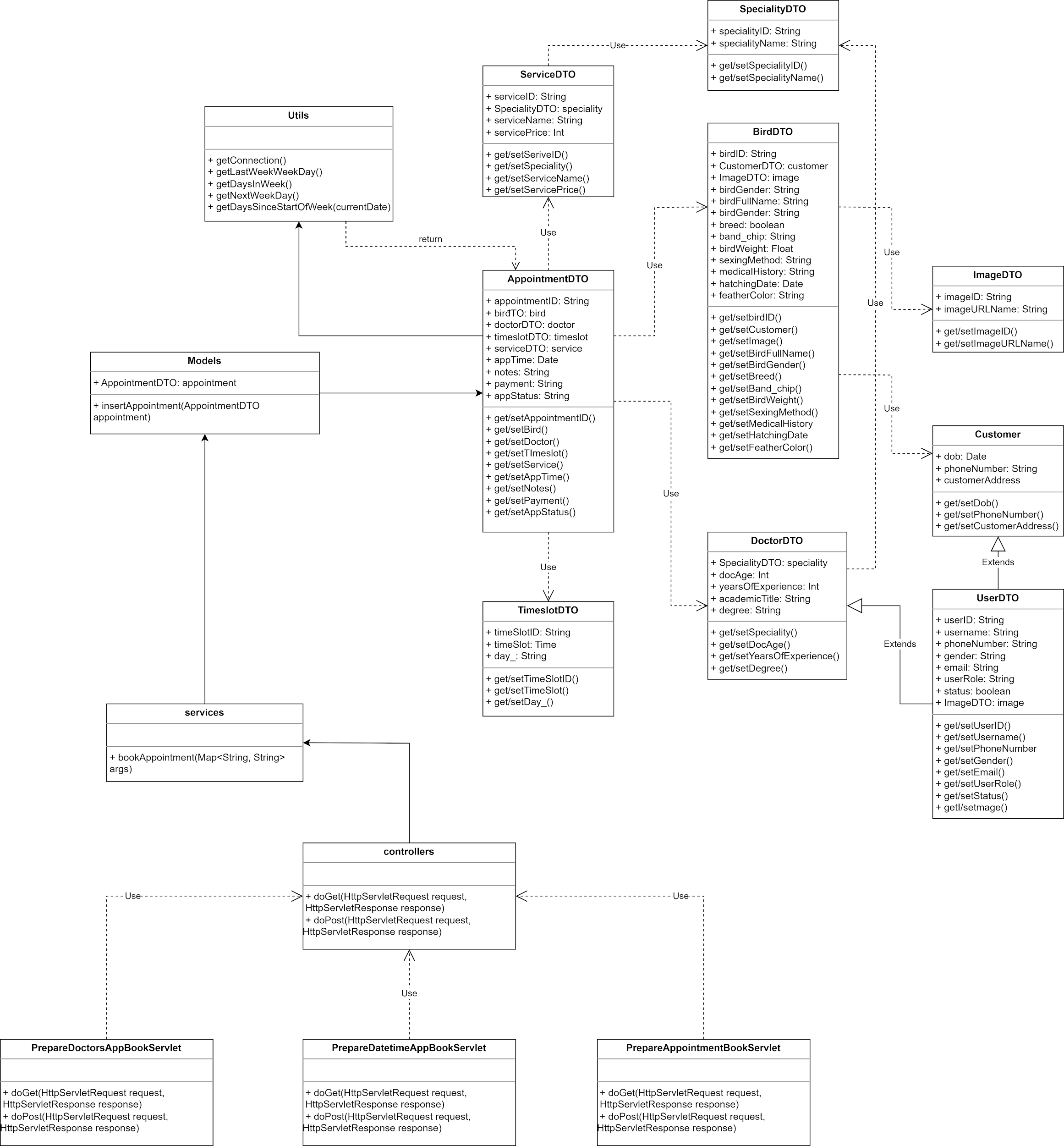
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

### c. Sequence Diagram(s)

### d. Database Queries

## 17. Common/Make an appointment

### a. Class Diagram



### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | Get all the information of the customer, the bird, service, time slot, notes and the date in the appointment. Also redirect to required pages |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | send error, call out all the information in the Appointment |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | bookAppointment(Map<String, String> args) | Book a new appointment |

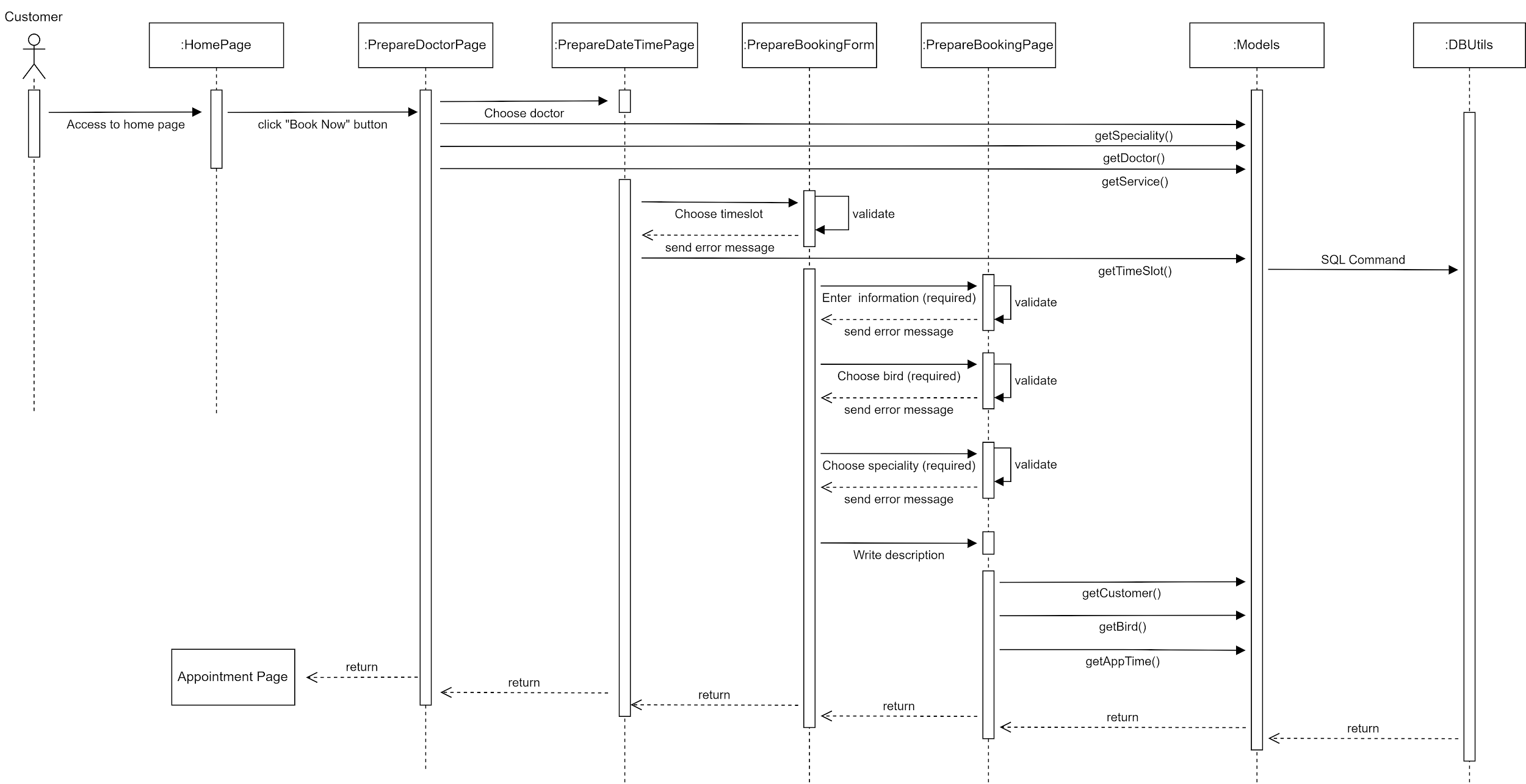
* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | insertAppointment(AppointmentDTO appointment) | Create a new appointment in the database |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |
| 02 | getLastWeekWeekday(currentDate) | get the day in the last week |
| 03 | getDaysSinceStartOfWeek(currentDate) | get the day before today |
| 04 | getDaysInWeek(currentDate) | get the day in the week |
| 05 | getNextWeekWeekday(currentDate) | get the next week day |

### c. Sequence Diagram(s)



### d. Database Queries

//Get all the information of the customer, the bird, service, time slot, notes and the date in the appointment

SELECT c.\*, b.\*, s.\*, t.\*, a.notes, a.appTime

FROM Appointment a

JOIN Customer c ON a.customerID = c.customerID

JOIN Bird b ON a.birdID = b.birdID

JOIN Service s ON a.serviceID = s.serviceID

JOIN TimeSlot t ON a.timeSlotID = t.timeSlotID;

//book appointment

“INSERT INTO Appointment (appointmentID, birdID, doctorID, timeSlotID, serviceID, appTime, notes, payment, appStatus)

VALUES ('[appointmentID]', '[birdID]', '[doctorID]', '[timeSlotID]', '[serviceID]', '[appTime]', '[notes]', '[payment]', '[appStatus]')”

//Get all the available doctor

SELECT d.\*

FROM Doctor d

WHERE d.doctorID NOT IN (SELECT doctorID FROM Appointment);

//Check the status of the doctor based on doctorID

SELECT appStatus

FROM Appointment

WHERE doctorID = 'doctorID\_value';

//Get all the time slot information related to the doctorID

SELECT t.\*

FROM TimeSlot t

JOIN Appointment a ON t.timeSlotID = a.timeSlotID

WHERE a.doctorID = 'doctorID\_value';

//Get all the specialities in the clinic

SELECT DISTINCT specialty

FROM Doctor;

//Create a new appointment in the database

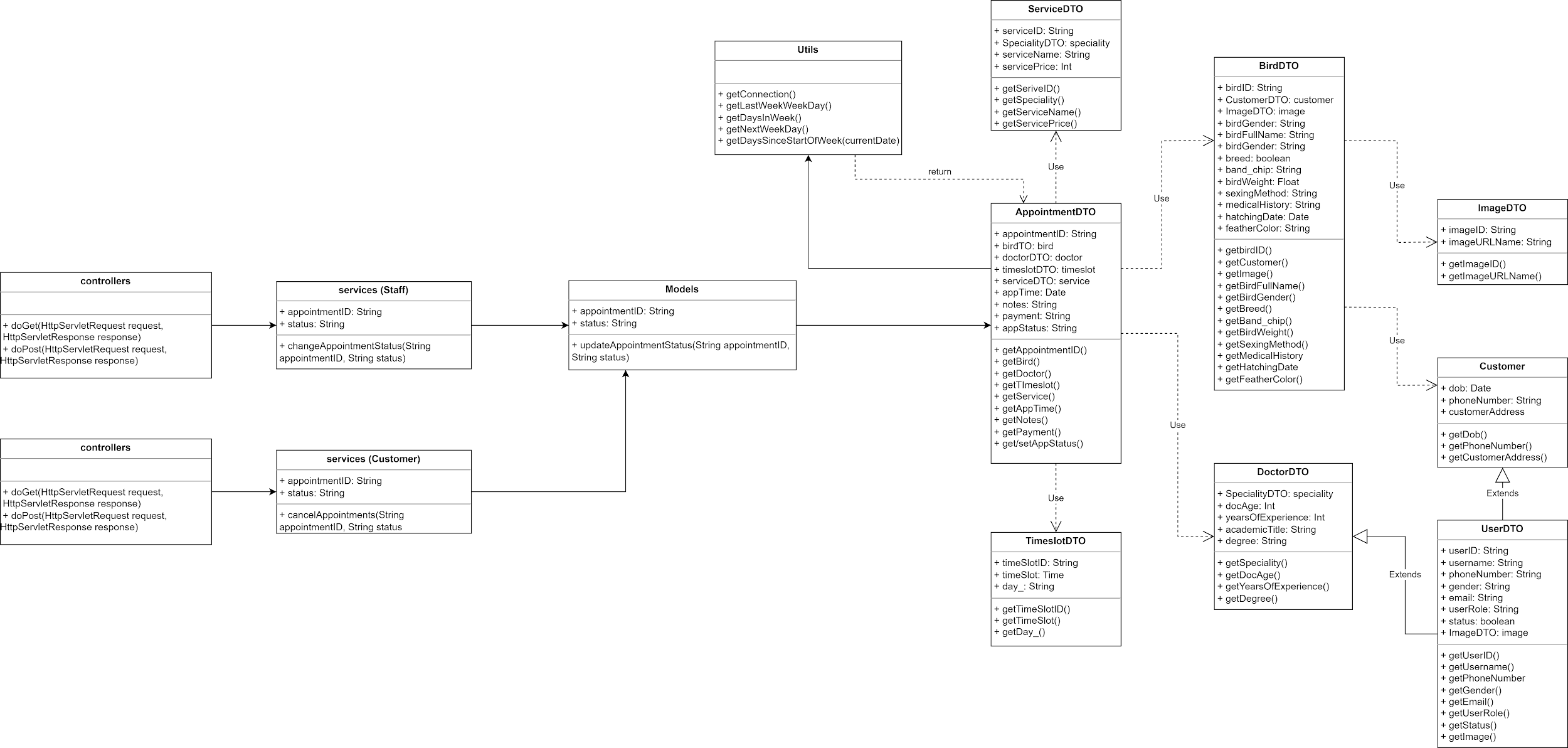
SELECT \*

FROM Appointment

WHERE appointmentID = 'appointmentID\_value';

### 18. Customer,Staff/Manage Appointments

### a. Class Diagram



### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | changeAppointmentStatus(String appointmentID, String status) | Change the appointment’s status based on appointmentID and status |
| 02 | cancelAppointment(String appointmentID, String status) | Set the appointment’s status into “Cancel‘ |

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateAppointmentStatus(String appointmentID, String status) | Update the appointment’s status |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |
| 02 | getLastWeekWeekday(currentDate) | get the day in the last week |
| 03 | getDaysSinceStartOfWeek(currentDate) | get the day before today |
| 04 | getDaysInWeek(currentDate) | get the day in the week |
| 05 | getNextWeekWeekday(currentDate) | get the next week day |

### c. Sequence Diagram(s)

### d. Database Queries

//View the appointment based on appointmentID

SELECT \*

FROM Appointment

WHERE appointmentID = 'appointmentID\_value';

### //View the medical record assigned to the appointment

SELECT medicalRecordID, appointmentID, recordDate, diagnosis, treatment

FROM MedicalRecord

WHERE appointmentID = 'appointmentID\_value';

//View the appointment based on doctorID

SELECT \*

FROM Appointment

WHERE doctorID = 'doctorID\_value';

//View the appointment booked by the customer based on customerID

SELECT \*

FROM Appointment

WHERE customerID = 'customerID\_value';

//View the appointment based on the timeslotID

SELECT \*

FROM Appointment

WHERE timeSlotID = 'timeslotID\_value';

//View the appointment assigned to the doctor’s schedule based on doctorID, timeslotID, appDate

SELECT \*

FROM Appointment

WHERE doctorID = 'doctorID\_value'

AND timeSlotID = 'timeslotID\_value'

AND appDate = 'appDate\_value';

//Delete a reservation in the list

DELETE FROM Appointment

WHERE appointmentID = 'appointmentID\_value';

//Insert a new appointment to the list

INSERT INTO Appointment (appointmentID, doctorID, customerID, timeslotID, appDate)

VALUES ('appointmentID\_value', 'doctorID\_value', 'customerID\_value', 'timeslotID\_value', 'appDate\_value');

//Update a new appointment to the list

UPDATE Appointment

SET doctorID = 'new\_doctorID\_value', customerID = 'new\_customerID\_value', timeslotID = 'new\_timeslotID\_value', appDate = 'new\_appDate\_value'

WHERE appointmentID = 'appointmentID\_value';

//View all the clinic’s appointments in a list

SELECT \*

FROM Appointment;

//View the time slot based on timeslotID

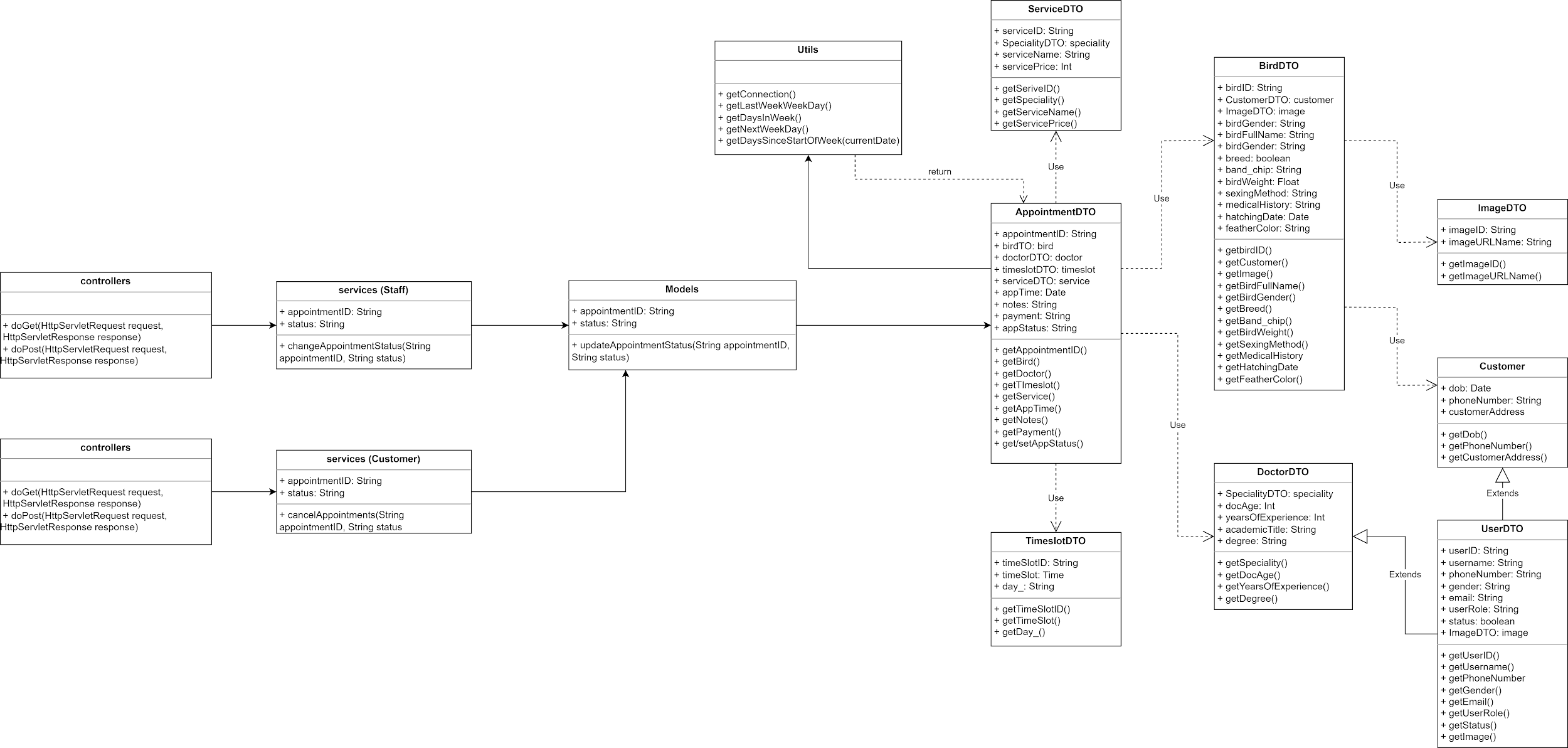
SELECT \*

FROM Timeslot

WHERE timeslotID = 'timeslotID\_value';

### 19. Common/View Appointments

### a. Class Diagram



### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | viewAppointment(String appointmentID) | View the appointment information based on appointmentID |

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | readAppointment(String appointmentID) | Get the appointment information based on appointmentID |
| 02 | readAppointmentByCustomer(String customerID) | Get the appointment information based on appointmentID |
| 03 | readAppointmentByBird(String birdID) | Get the appointment information by birds based on birdID |
| 04 | readAppointmentByDoctor(String doctorID) | Get the appointment information by doctors based on doctorID |
| 05 | readAppointmentByTimeslot(String timeSlotID) | Get the appointment information by time slots based on timeslotID |
| 06 | readAppointmentByDocTime(String doctorID, String timeslotID, Date appDate) | Get the appointment information by doctor’s schedule based on doctorID, timeslotID, appDate |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |
| 02 | getLastWeekWeekday(currentDate) | get the day in the last week |
| 03 | getDaysSinceStartOfWeek(currentDate) | get the day before today |
| 04 | getDaysInWeek(currentDate) | get the day in the week |
| 05 | getNextWeekWeekday(currentDate) | get the next week day |

### c. Sequence Diagram(s)

### d. Database Queries

//Get the appointment information of the current user

SELECT \*

FROM Appointment

WHERE customerID = 'customerID\_value';

//Get the appointment information based on appointmentID

SELECT \*

FROM Appointment

WHERE appointmentID = 'appointmentID\_value';

//Get the appointment information by birds based on birdID

SELECT \*

FROM Appointment

WHERE birdID = 'birdID\_value';

//Get the appointment information by doctors based on doctorID

SELECT \*

FROM Appointment

WHERE doctorID = 'doctorID\_value';

//Get the appointment information by time slots based on timeslotID

SELECT \*

FROM Appointment

WHERE timeslotID = 'timeslotID\_value';

//Get the appointment information by doctor’s schedule based on doctorID, timeslotID, appDate

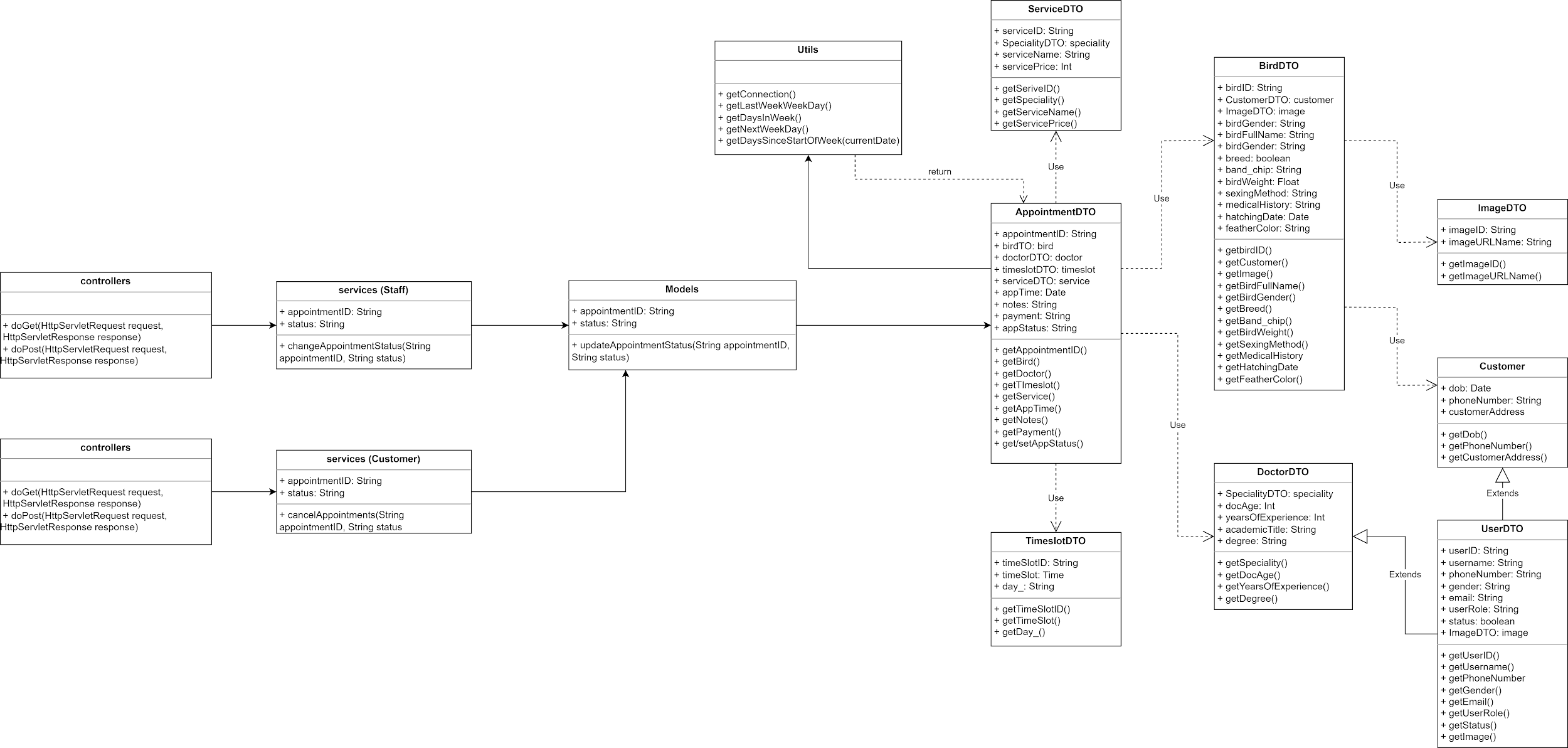
SELECT \*

FROM Appointment

WHERE doctorID = 'doctorID\_value' AND timeslotID = 'timeslotID\_value' AND appDate = 'appDate\_value';

### 20. Customer/View Record

### a. Class Diagram



### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | viewMedicalRecord(String appointmentID) | View the medical record assigned to the appointment based on appointmentID |
| 02 | viewRecordMeds(String medicalRecordID) | View the medical record based on the medicalRecordID |

* Models

| 01 | readMedicalRecord(String medicalRecordID) | Get the medical record information based on medicalRecordID |
| --- | --- | --- |
| 02 | readMedicalRecordByAppointment(String appointmentID) | Get the medical record information assigned to the appointment based on appointmentID |
| 03 | readMedicalRecordByBird(String birdID) | Get the medical record assigned to the bird based on birdID |
| 04 | readMedicalRecordByDoctor(String doctorID) | Get the medical record assigned to the doctor based on doctorID |
| 05 | readMedicineFromRecord(String medicalRecordID) | Get the medicine in the medical record based on medicalRecordID |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |

### c. Sequence Diagram(s)

### d. Database Queries

//Get the medical record information based on medicalRecordID

SELECT \*

FROM MedicalRecord

WHERE medicalRecordID = 'medicalRecordID\_value';

//Get the medical record information assigned to the appointment based on appointmentID

SELECT \*

FROM MedicalRecord

WHERE appointmentID = 'appointmentID\_value';

//Get the medical record assigned to the bird based on birdID

SELECT \*

FROM MedicalRecord

WHERE birdID = 'birdID\_value';

//Get the medical record assigned to the doctor based on doctorID

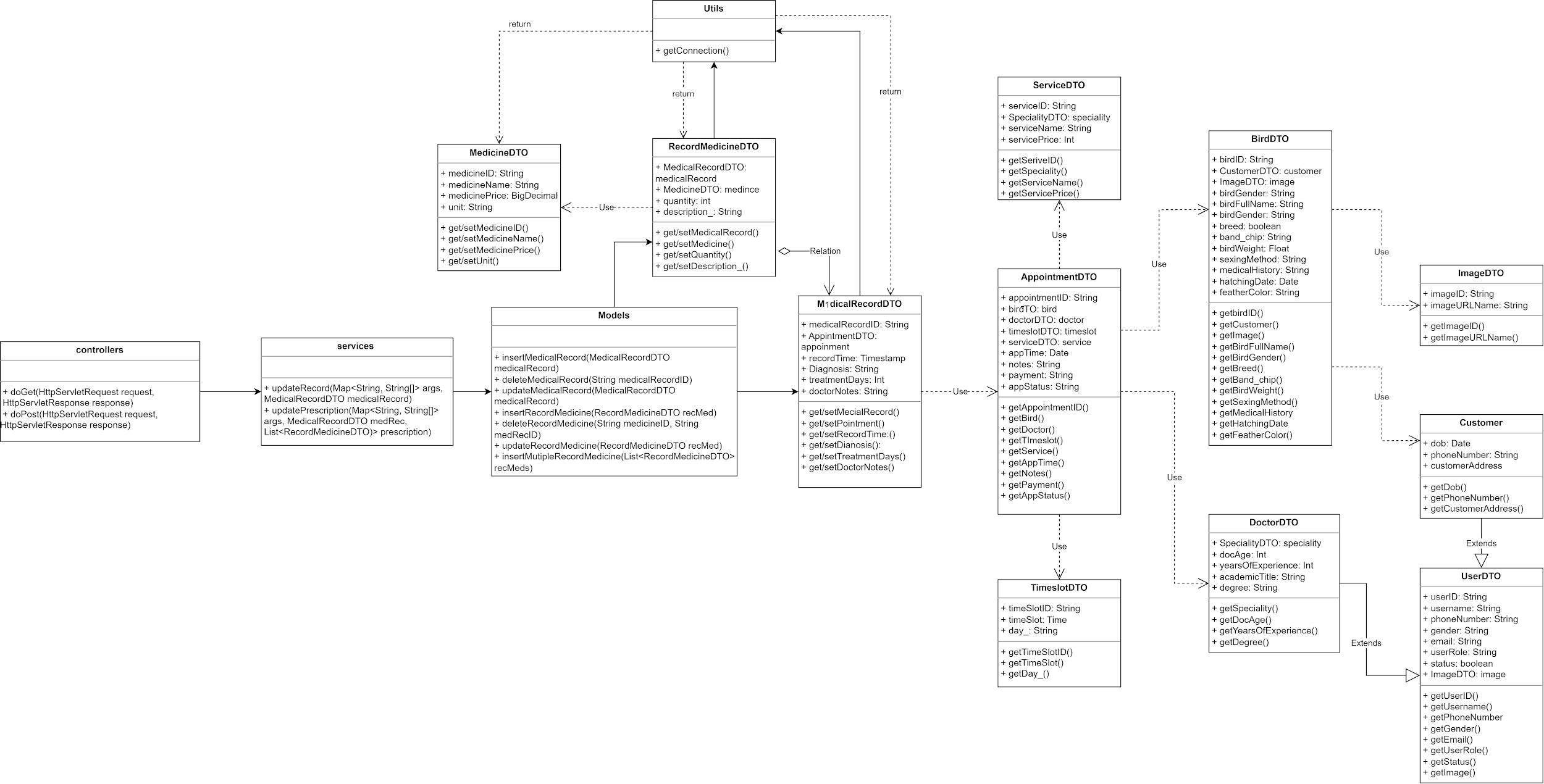
SELECT \*

FROM MedicalRecord

WHERE doctorID = 'doctorID\_value';

### 21. Doctor/Edit Record

### a. Class Diagram



### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateRecord(Map<String, String[]> args, MedicalRecordDTO medicalRecord) | Update a medical record |
| 02 | updatePrescription(Map<String, String[]> args, MedicalRecordDTO medRec, List<RecordMedicineDTO> prescription) | Update medicines included in a medical record |

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | insertMedicalRecord(MedicalRecordDTO medicalRecord) | Insert a new medical record to the database |
| 02 | deleteMedicalRecord(String medicalRecordID) | Delete a medical record in the database |
| 03 | updateMedicalRecord(MedicalRecordDTO medicalRecord) | Update a new medical record to the database |
| 04 | deleteRecordMedicine(String medicineID, String medRecID) | Delete the medicine included in the medical record based on medicineID and mecRecID |
| 05 | insertRecordMedicine(RecordMedicineDTO recMed) | Insert a new medicine to the medical record |
| 06 | updateRecordMedicine(RecordMedicineDTO recMed) | Update a medicine to the medical record |
| 07 | insertMultipleRecordMedicine(List<RecordMedicineDTO> recMeds) | Insert multiple medicines to the medical record |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |

### c. Sequence Diagram(s)

### d. Database Queries

//Get all the medical record information based on medicalRecordID

SELECT \*

FROM MedicalRecord

WHERE medicalRecordID = 'medicalRecordID\_value';

//Insert a new medical record to the database

INSERT INTO MedicalRecord (medicalRecordID, appointmentID, birdID, doctorID, diagnosis, prescription)

VALUES ('medicalRecordID\_value', 'appointmentID\_value', 'birdID\_value', 'doctorID\_value', 'diagnosis\_value', 'prescription\_value');

//Delete a medical record in the database

DELETE FROM MedicalRecord

WHERE medicalRecordID = 'medicalRecordID\_value';

//Update a new medical record to the database

UPDATE MedicalRecord

SET diagnosis = 'new\_diagnosis\_value', treatmentDays = new\_treatmentDays\_value, doctorNotes = 'new\_doctorNotes\_value'

WHERE medicalRecordID = 'medicalRecordID\_value';

//Get the medicine information based on medicineID

SELECT \*

FROM Medicine

WHERE medicineID = 'medicineID\_value';

//Search the medicine based on medicineName

SELECT \*

FROM Medicine

WHERE medicineName LIKE '%search\_term%';

//Get all the medicine information in the database

SELECT \*

FROM Medicine;

//Get all the medicine information organized in list

SELECT medicineName

FROM Medicine;

//Update a new medicine to the database

UPDATE Medicine

SET medicineName = 'new\_medicine\_name', dosage = 'new\_dosage'

WHERE medicineID = 'medicineID\_value';

//Get all the medicine included in the medical record based on medicalRecordID

SELECT Medicine.\*

FROM Medicine

JOIN MedicalRecordMedicine ON Medicine.medicineID = MedicalRecordMedicine.medicineID

WHERE MedicalRecordMedicine.medicalRecordID = 'medicalRecordID\_value';

//Delete the medicine included in the medical record based on medicineID and mecRecID

DELETE FROM MedicalRecordMedicine

WHERE medicineID = 'medicineID\_value' AND medicalRecordID = 'medicalRecordID\_value'; // Replace 'medicineID\_value' and 'medicalRecordID\_value'

//Insert a new medicine to the medical record

INSERT INTO RecordMedicine (medicalRecordID, medicineID, quantity, description\_)

VALUES ('medicalRecordID\_value', 'medicineID\_value', quantity\_value, 'description\_value');

//Update a medicine to the medical record

UPDATE MedicalRecord

SET medicineID = 'medicineID\_value'

WHERE medicalRecordID = 'medicalRecordID\_value';

### 22. Admin/Manage User

### a. Class Diagram



### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createDoctor(Map<String, String[]> args) | Create a new account with the role assigned as “Doctor” |
| 02 | createStaffAdmin(Map<String, String[]> args) | Create a new account with the role assigned as “Staff” or “Admin” |

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | insertUser(UserDTO user) | Insert a new user to the database |
| 02 | deleteUser(String userID) | Delete a user from the database |
| 03 | updateUser(UserDTO user) | Update a new user to the database |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |

### c. Sequence Diagram(s)

### d. Database Queries

//Create a new account with the role assigned as “Doctor"

INSERT INTO Users (userID, imageID, userName, userPassword, fullName, phoneNumber, gender, email, userRole, status\_)

VALUES ('userID\_value', 'imageID\_value', 'userName\_value', 'userPassword\_value', 'fullName\_value', 'phoneNumber\_value', 'gender\_value', 'email\_value', 'Doctor', 'status\_value');

//Create a new account with the role assigned as “Staff” or “Admin”

INSERT INTO Users (userID, imageID, userName, userPassword, fullName, phoneNumber, gender, email, userRole, status\_)

VALUES ('userID\_value', 'imageID\_value', 'userName\_value', 'userPassword\_value', 'fullName\_value', 'phoneNumber\_value', 'gender\_value', 'email\_value', 'Staff', 'status\_value');

//Get the user information based on userID

SELECT \*

FROM Users

WHERE userID = 'userID\_value';

//Get the user information based on email and username

SELECT \*

FROM Users

WHERE email = 'email\_value' AND userName = 'userName\_value';

//Get the user information based on userRole

SELECT \*

FROM Users

WHERE userRole = 'userRole\_value';

//Insert a new user to the database

INSERT INTO Users (userID, imageID, userName, userPassword, fullName, phoneNumber, gender, email, userRole, status\_)

VALUES ('userID\_value', 'imageID\_value', 'userName\_value', 'userPassword\_value', 'fullName\_value', 'phoneNumber\_value', 'gender\_value', 'email\_value', 'userRole\_value', 'status\_value');

//Delete a user from the database

DELETE FROM Users

WHERE userID = 'userID\_value';

//Update a new user to the database

UPDATE Users

SET imageID = 'new\_imageID\_value', userName = 'new\_userName\_value', userPassword = 'new\_userPassword\_value', fullName = 'new\_fullName\_value', phoneNumber = 'new\_phoneNumber\_value', gender = 'new\_gender\_value', email = 'new\_email\_value', userRole = 'new\_userRole\_value', status\_ = 'new\_status\_value'

WHERE userID = 'userID\_value';

### 23. Admin/User List

### a. Class Diagram

### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
|  |  |  |

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | readUser(String userID) | Get the user information based on userID |
| 02 | readUserByEmailUserName(String email, String username) | Get the user information based on email and username |
| 03 | readUserByRole(String userRole) | Get the user information based on userRole |
| 04 | readAllUsers() | Get all of the user information in the database |
| 05 | readListOfUsers(List<String> userID) | Get all of the user information organized in list |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |

### c. Sequence Diagram(s)

### d. Database Queries

// Get user information based on userID

SELECT \*

FROM Users

WHERE userID = 'your\_userID';

// Get user information based on email and username

SELECT \*

FROM Users

WHERE email = 'your\_email' AND userName = 'your\_username';

// Get user information based on userRole

SELECT \*

FROM Users

WHERE userRole = 'your\_userRole';

// Get all user information in the database

SELECT \*

FROM Users;

// Get all user information organized in a list

SELECT userID, userName, fullName, email, userRole

FROM Users;

### 24. Admin/Choose time slot

### a. Class Diagram

### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | changeTimeslot(String doctorID, List<String> timeslotID) | Change time slots for the doctor |

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deleteDoctorTimeslot(String doctorID, String timeSlotID) | Delete a time slot assigned to a doctor based on doctorID, timeSlotID |
| 02 | insertDoctorTimeslot(String doctorID, String timeSlotID) | Insert a new time slot assigned to a doctor based on doctorID, timeSlotID |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |

### c. Sequence Diagram(s)

### d. Database Queries

//choose time slot

“SELECT timeSlotID, timeSlot, day\_

FROM TimeSlot

WHERE timeSlotID NOT IN (

SELECT timeSlotID

FROM Appointment

WHERE appTime = 'yyyy-mm-dd hh:mm:ss'

)”

// Get all time slots in the week based on doctorID

SELECT \*

FROM TimeSlots

WHERE doctorID = 'your\_doctorID';

// Get the time slot based on TimeslotID

SELECT \*

FROM TimeSlots

WHERE timeSlotID = 'your\_timeslotID';

// Get all time slots in the database

SELECT \*

FROM TimeSlots;

// Get all time slots organized in a list

SELECT timeSlotID, doctorID, dayOfWeek, startTime, endTime

FROM TimeSlots;

// Get all days in a week

SELECT dayOfWeek

FROM TimeSlots

GROUP BY dayOfWeek;

// Get all time slots assigned to the doctor based on doctorID

SELECT \*

FROM TimeSlots

WHERE doctorID = 'your\_doctorID';

// Get all doctor's time slots based on timeslotID

SELECT \*

FROM TimeSlots

WHERE timeSlotID = 'your\_timeslotID';

// Delete a time slot assigned to a doctor based on doctorID, timeSlotID

DELETE FROM TimeSlots

WHERE doctorID = 'your\_doctorID' AND timeSlotID = 'your\_timeslotID';

// Insert a new time slot assigned to a doctor based on doctorID, timeSlotID

INSERT INTO TimeSlots (doctorID, timeSlotID, dayOfWeek, startTime, endTime)

VALUES ('your\_doctorID', 'your\_timeslotID', 'your\_dayOfWeek', 'your\_startTime', 'your\_endTime');

// Get a group of time slots assigned to a doctor based on doctorID

SELECT \*

FROM TimeSlots

WHERE doctorID = 'your\_doctorID';

### 25. Admin/Admin Dashboard

### a. Class Diagram

### b. Class Specifications

* Controllers

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest request, HttpServletResponse response) | To handle requests and get data from the database. Also redirect to the jsp |
| 02 | doGet(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doPost(HttpServletRequest request, HttpServletResponse response) | Call processRequest(HttpServletRequest, HttpServletResponse) |

* Services

| **No** | **Method** | **Description** |
| --- | --- | --- |
|  |  |  |
|  |  |  |

* Models

| **No** | **Method** | **Description** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

* Utils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | make the connection to the database |

### c. Sequence Diagram(s)

### d. Database Queries

// Get all the doctor's time slots based on doctorID

SELECT \*

FROM TimeSlots

WHERE doctorID = 'your\_doctorID';

// Get all the doctor's time slots based on timeSlotID

SELECT \*

FROM TimeSlots

WHERE timeSlotID = 'your\_timeSlotID';

// Delete the time slot assigned to a doctor based on doctorID and timeSlotID

DELETE FROM TimeSlots

WHERE doctorID = 'your\_doctorID' AND timeSlotID = 'your\_timeSlotID';

// Insert a new time slot assigned to a doctor based on doctorID and timeSlotID

INSERT INTO TimeSlots (doctorID, timeSlotID, dayOfWeek, startTime, endTime)

VALUES ('your\_doctorID', 'your\_timeSlotID', 'your\_dayOfWeek', 'your\_startTime', 'your\_endTime');

// Get the group of time slots assigned to a doctor based on doctorID

SELECT \*

FROM TimeSlots

WHERE doctorID = 'your\_doctorID';