



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF ENGINEERING
SCHOOL OF COMPUTING
SCSJ3303 INTERNET PROGRAMMING
GROUP PROJECT - SECTION 01

Final Report

Lecturer : Dr Nor Azizah Binti Sa'adon

Prepared by:

Group 7: BlueZone

NO.	NAME	MATRICS NUMBER
1.	NIK ADAM DANISH BIN JOHARRIS	A17CS0292
2.	MUHAMMAD HUZAIFAH BIN AZMAN	A17CS0120
3.	NUR HUSNA BINTI AZMI	A17CS0158
4.	NURHAFIZAH BINTI SHUKRI	A17CS0164

Contents

1. INTRODUCTION.....	3
2. SYSTEM MODULE	4
2.1 Patient	5
2.2 Doctor	5
2.3 Pharmacist	6
2.4 Admin	7
3. USE CASE.....	8
4. FLOW OF PROJECT	9
5. PROJECT ELEMENTS	10
5.1 Register	10
5.2 Session/page Validation.....	12
5.3 View/edit Profile.....	15
5.4 Consistent GUI	20
5.5 Data management - CRUD + activation	21
5.6 User transaction and history	27
5.7 Admin manage user transaction	33
5.8 Reporting.....	39
5. REFLECTION	45

1. INTRODUCTION

Our web application is known as the Clinic Management System helps to keep track, organize, and automate the process in the clinic. In addition, this application will also assist patients to book their appointments before entering the clinic. This ensures that the patients do not have to wait an excessive period to receive their diagnosis

To book an appointment, patients will have to log in to their account and if they do not have an account they have to register first. Then, they can fill the form generated by the application. They have to fill up their booking date and time, and any remarks/queries they may have. Once the form is sent, the admin forwards the request to the doctor, who has the option to approve or decline the booking. If the booking is approved, the system will display a status “Approved” at the ‘View Appointment’ page. The patients also have an option to cancel the appointment by clicking ‘Cancel Appointment’ then it will be automated deleted at both Doctor and Patients side.

The doctor as aforementioned can approve and reject the medical appointments. Other than that, he can issue prescription forms too. Once the doctor is done with diagnosing his patient, he will have to fill the prescription forms online. The forms are then sent to the pharmacist to give the patient the appropriate medication. Besides that, dr can view the history of the previous appointments that have been done and also the appointments for the current date.

The pharmacist was the one that can view the prescription that doctors have prescribed to the patients. After they prepare the medication based on the prescription, they will update the status of the medicine that has been given to the patients. Therefore, the inventory of the medicines will be updated.

The admin is in charge of managing staff (doctors and pharmacists) and medicine inventory. Admin can also keep track of the history of patients and appointments. Medicine can only be given to the patient after a prescription form has been issued by the doctor. The admin is allowed to edit, add, delete also activate or deactivate the medicine inventory. Besides that, the admin can view monthly and each patients’ appointment counts.

2. SYSTEM MODULE

Administrator	<ul style="list-style-type: none">● An administrator can view monthly appointment counts● An administrator can view patient appointment history● An administrator can activate/deactivate inventory● An administrator can manage inventory● An administrator can manage staff
Doctor	<ul style="list-style-type: none">● A doctor can view today's medical appointment● A doctor can approve/reject appointments● A doctor can view the history of previous appointments● A doctor can issue prescriptions form● A doctor can view and update profile
Pharmacist	<ul style="list-style-type: none">● A pharmacist can view the patient's prescription● A pharmacist can update the status of inventory● A pharmacist can view completed prescription● A pharmacist can view and update profile
Patient	<ul style="list-style-type: none">● A patient can log in or register an account● A patient can view and update profile● A patient can book medical appointments● A patient can cancel medical appointments● A patient can view the prescribed prescription

2.1 Patient

Book Appointment

The patient is able to book an appointment. The patient is able to choose the date they want their appointment to be, the time of the appointment, as well the doctor of their choice. Once a booking has been made, the booking will be stored in the database.

View booking

The patient is able to view the bookings they have made. The booking has four states: Rejected, Approved, Pending, and Complete. For this module, the patient can only view the first three states. The states will be displayed in a table to indicate whether the doctor has validated their booking. If the doctor has rejected the booking, the patient can view the doctor's remarks to see why he/she has rejected the appointment.

View prescription:

After the doctor has approved the appointment, the patient is called into the clinic to receive his/her diagnosis. Once the diagnosis is done, the patient is able to view their own prescription in their account

2.2 Doctor

Validate appointment

The doctor can choose to accept or reject patient appointment. Once the doctor accepts or rejects the appointment, the system will update the booking status to "Approved" or "Rejected" respectively. If the doctor chooses to reject an appointment, the doctor must give a remark stating why they cannot accept the appointment on that particular day.

Today's appointment

The doctor can view the appointments he/she has on the day. The appointments displayed are only the ones that have “Approved” as their status. Once the appointment is done, the doctor can prescribe the medicine to the patient. The medicine prescribed is connected with the current inventory. If the medicine prescribed by the doctor is more than what is in the current inventory, the system displays an error message. Once the medicine is prescribed to the patient, the system will set the appointment status to “Completed”.

View upcoming appointments

The doctor can view the upcoming appointments he/she may have. The appointments displayed are only the ones that have “Approved” as their status. The doctor can also filter the appointments by their dates.

Appointment History

The doctor can view the appointment history. The appointments displayed are only the ones that have “Completed” as their status.

2.3 Pharmacist

View prescription

The pharmacist can view the prescribed medicine the doctor has set for the patient. The prescription displayed is only available after the doctor has prescribed medication for the patient. The pharmacist can also view the details of the prescribed medication in order to hand it out to the patient over the counter.

Completed prescription

Once the prescriptions are handed to the patient, the pharmacist will click on the “Task Complete”. This will set the prescription status to “Completed”, meaning that the patient's visit is now over.

2.4 Admin

View Patient History

The admin can view the lists of patients that have undergone appointments. From the lists, admin can view the number of appointments of the particular patient and also provided an option to view the list of appointments for the specific patient.

View Appointment History

The admin can view the monthly appointment amount. From the amount, admin has the option to view the lists of appointments for that particular month.

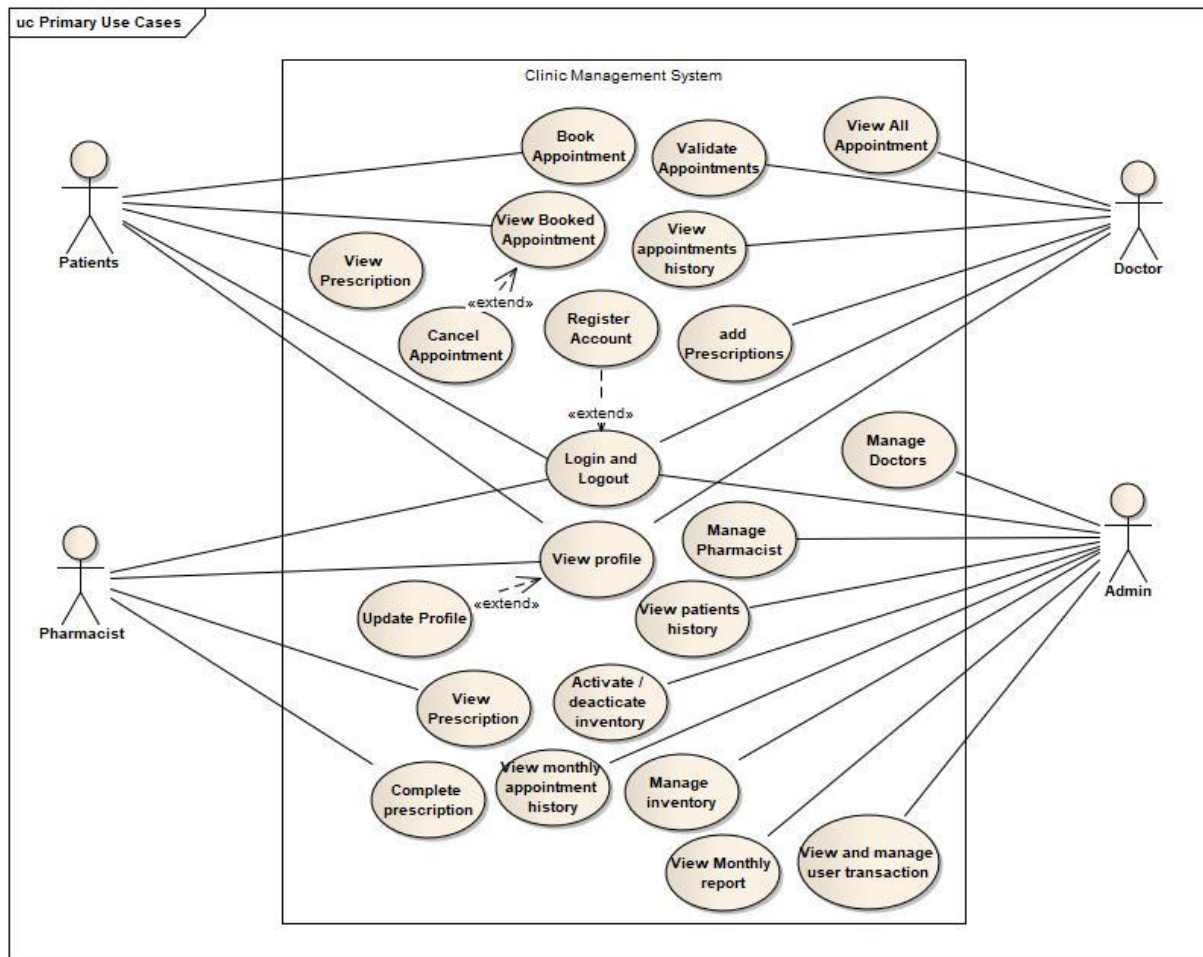
Manage Staff

The admin can manage the staffs' account, which is for a doctor or pharmacist. Admin can view all the staffs' information in a list. From the list, admin has the option to add new staff, update staff's information and delete staffs' accounts.

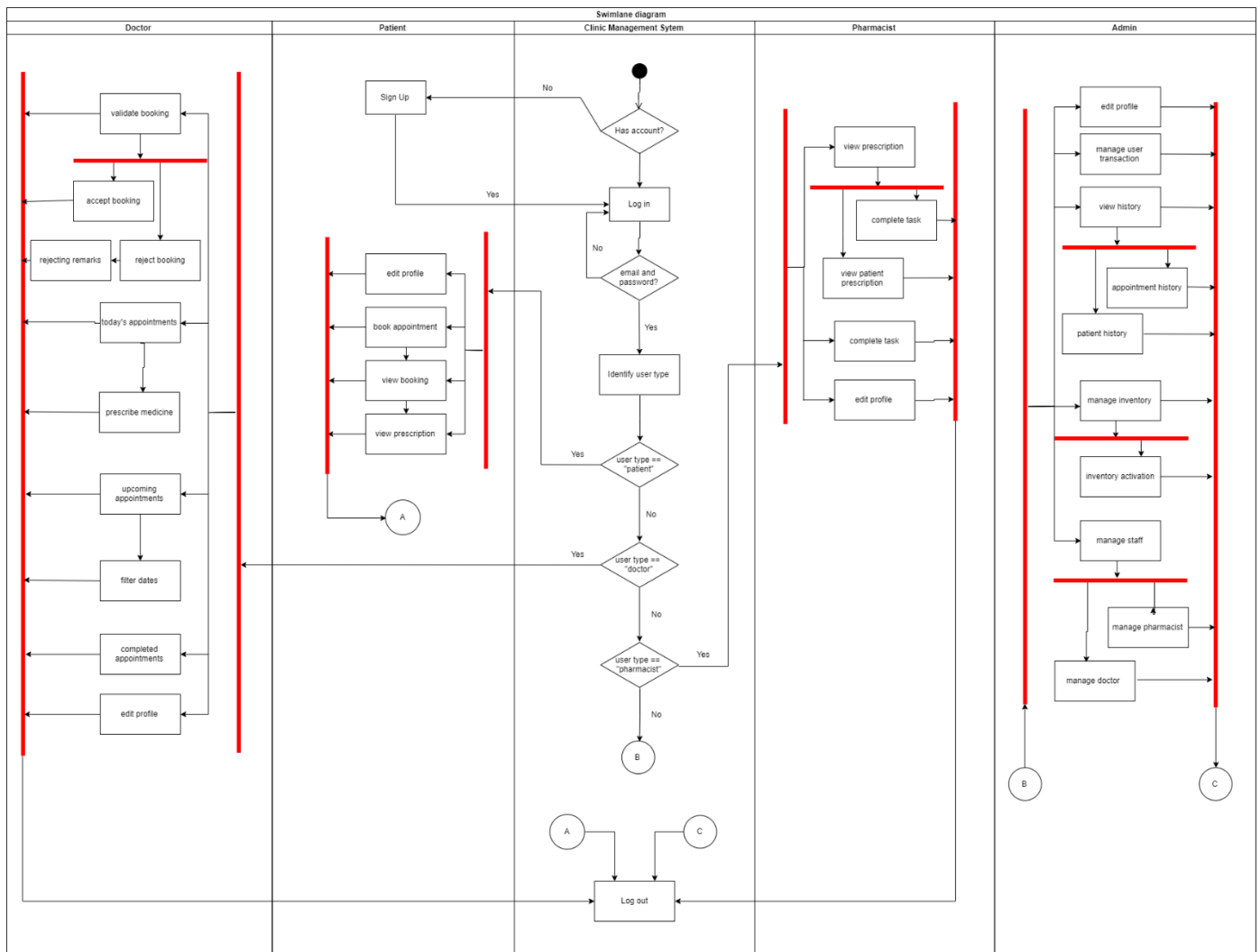
Manage Inventory

The admin can view the lists of inventory for the clinic. From the list, admin can add new inventory, update inventories' information and delete any inventory. Also, the admin can activate or deactivate the status for any specific inventory so that the doctor can only prescribe the available medicine for the patients.

3. USE CASE



4. FLOW OF PROJECT



****For a clearer view, proceed to the last page for image attachment.**

PatientController.java

```
private void registerPatient(HttpServletRequest request, HttpServletResponse response) throws ClassNotFoundException,
    SQLException, ServletException, IOException {

    //get parameters
    String name = request.getParameter("name");
    String ic = request.getParameter("ic");
    String dob = request.getParameter("dob");
    String phone = request.getParameter("phone");
    String gender = request.getParameter("gender");
    String email = request.getParameter("email");
    String password = request.getParameter("password");

    //set parameter to class
    user.setName(name);
    user.setIc(ic);
    user.setDOB(dob);
    user.setPhone(phone);
    user.setGender(gender);
    user.setEmail(email);
    user.setPassword(password);
    user.setRole("patient");

    //inset data to database
    userDao.registerUser(user);

    loginUser(request, response);
}
```

UserDAO.java

```
public void registerUser(User user) throws ClassNotFoundException, SQLException{
    //get connection
    Class.forName(driver);
    Connection con = DriverManager.getConnection(url, userName, password);

    //set query
    String query = "insert into user"
        + "(name, email, password, phone, DOB, ic, gender, role) "
        + "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

    PreparedStatement ps = con.prepareStatement(query);

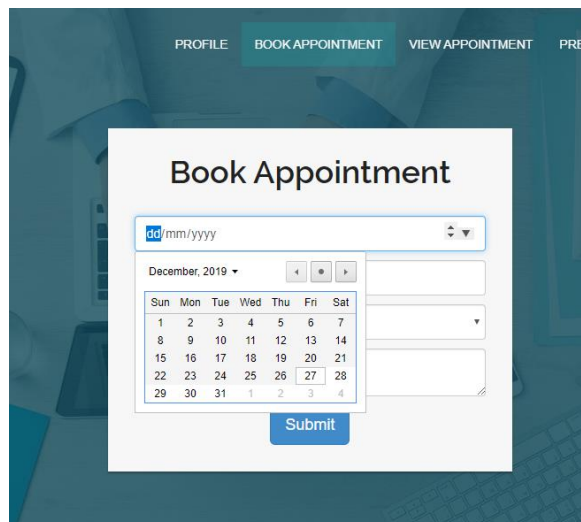
    ps.setString(1, user.getName());
    ps.setString(2, user.getEmail());
    ps.setString(3, user.getPassword());
    ps.setString(4, user.getPhone());
    ps.setString(5, user.getDOB());
    ps.setString(6, user.getIc());
    ps.setString(7, user.getGender());
    ps.setString(8, user.getRole());

    //execute query
    ps.execute();
    con.close();
}
```

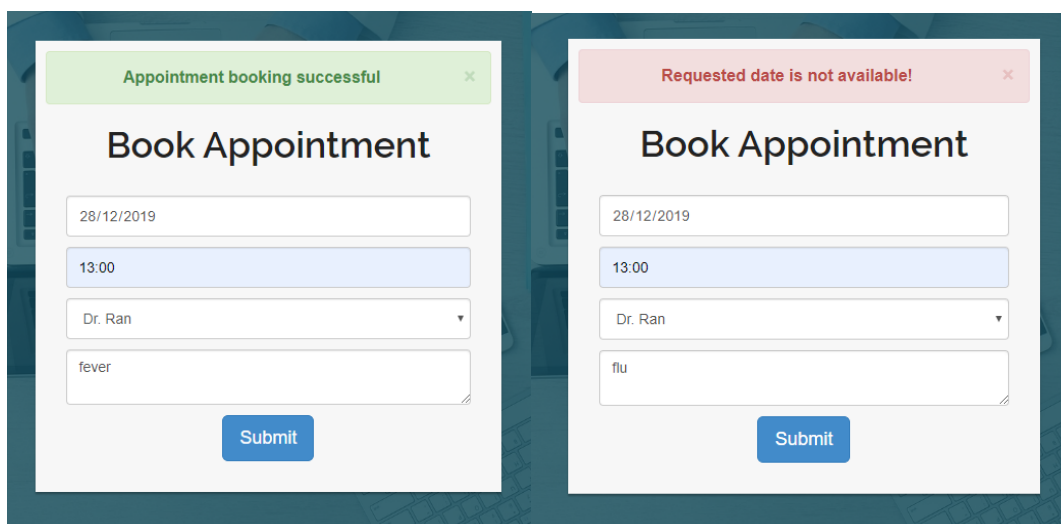
5.2 Session/page Validation

We implement validation in a few of our session and page. In our site, patients can only booking their appointment when the doctor is available for that day or the date is available. For example, at the booking page, the calendar will show dark grey boxes for the dates that are not available or valid. So the patients can choose any other dates that are available. Besides that, each day, each doctor will have their own limit for appointments. Therefore, the patients cannot book their appointment when the doctors' appointment limit is reached. Furthermore, we implement this validation on our add prescription page where the doctor cannot exceed a limit when prescribe medications to patients.

Interface



The screenshot shows the 'Book Appointment' form. At the top, there are navigation links: PROFILE, BOOK APPOINTMENT (highlighted), VIEW APPOINTMENT, and PRES. The form has a title 'Book Appointment'. Below the title is a date input field with a placeholder 'dd/mm/yyyy'. Below that is a calendar for December 2019. The calendar shows days from 1 to 31. Some days are highlighted in dark grey, indicating they are not available. To the right of the calendar are two empty input fields and a dropdown menu. At the bottom of the form is a blue 'Submit' button.



The image shows two side-by-side screenshots of the 'Book Appointment' form. The left screenshot shows a successful booking with a green success message at the top: 'Appointment booking successful'. The form fields are filled with: Date: 28/12/2019, Time: 13:00, Doctor: Dr. Ran, and Reason: fever. The right screenshot shows an error message at the top: 'Requested date is not available!'. The form fields are filled with: Date: 28/12/2019, Time: 13:00, Doctor: Dr. Ran, and Reason: flu. Both forms have a blue 'Submit' button at the bottom.

patientBooking.jsp

```
<div class="row">
  <div class="col-sm-4 col-md-5 col-md-offset-4">
    <div class="sign-up">
      <c:if test="${not empty success}">
        <div class="alert alert-dismissable alert-success">
          <button type="button" class="close" data-dismiss="alert">&times;</button>
          <strong>${success}</strong>
        </div>
      </c:if>
      <c:if test="${not empty fail}">
        <div class="alert alert-dismissable alert-danger">
          <button type="button" class="close" data-dismiss="alert">&times;</button>
          <strong>${fail}</strong>
        </div>
      </c:if>
      <h1>Book Appointment</h1><br>
      <form action="PatientController" method="post">
        <input type="hidden" name="function" value="bookAppointment">
        <div class="form-groups">
          <input type="date" class="form-control" name="date" required autofocus>
        </div>
        <div class="form-groups">
          <input type="text" class="form-control" name="time" placeholder="Time">
        </div>
        <div class="form-groups">
          <select class="form-control" name="doctor">
            <option>Dr. Ran</option>
            <option>Dr. Dom</option>
          </select>
        </div>
      </form>
    </div>
  </div>
</div>
```

BookindDAO.java

```
//patient book appointment
public void bookAppointment(Booking book) throws ClassNotFoundException, SQLException{

    //get connection
    Class.forName(driver);
    Connection con = DriverManager.getConnection(url, userName, password);

    //define query
    String query = "INSERT INTO booking"
        + "(user_id, date, time, symptoms, doctor, status, prescription_status, rejecting_remarks)"
        + "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

    //prepared Statment
    PreparedStatement ps = con.prepareStatement(query);

    java.sql.Date bookDate = new java.sql.Date(book.getDate().getTime());

    //insert values
    ps.setInt(1, book.getUser_id());
    ps.setDate(2, bookDate);
    ps.setString(3, book.getTime());
    ps.setString(4, book.getSymptoms());
    ps.setInt(5, book.getDoctor());
    ps.setString(6, book.getStatus());
    ps.setString(7, "Pending");
    ps.setString(8, "");

    //execute query
    ps.execute();
}
```

patientController.java

```
private void bookAppointment(HttpServletRequest request, HttpServletResponse response) throws
    HttpSession session = request.getSession(false);
    User currentUser = (User) session.getAttribute("user");

    //set parameters
    String stringDate = request.getParameter("date");

    //parse string to date
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    Date date = sdf.parse(stringDate);

    String time = request.getParameter("time");
    String symptom = request.getParameter("symptom");
    String doctor = request.getParameter("doctor");

    //get doctor id
    int doctorID = userDao.getDoctorID(doctor);

    //set parameters to the class
    booking.setUser_id(currentUser.getUserID());
    booking.setDate(date);
    booking.setTime(time);
    booking.setSymptoms(symptom);
    booking.setDoctor(doctorID);
    booking.setStatus("Pending");

    //get the number of bookings on a specific date
    List<Integer> bookedDates = bookingDAO.getDate(booking);

    if(bookedDates.size() >= 3){
        request.setAttribute("fail", "Requested date is not available!");
        patientPage(request, response);
    }else{
        //pass values to BookingDAO
        bookingDAO.bookAppointment(booking);
        request.setAttribute("success", "Appointment booking successful");
        patientPage(request, response);
    }
}
```

5.3 View/edit Profile

Profile is available for all actors. Their personal information will be displayed on their profile. Furthermore, they can update their information anytime if there are any changes by clicking on profile at the navigation bar.

Interface view profile

NAME	Nurhafizah Shukri
E-MAIL	fizah@gmail.com
PHONE NUMBER	0142294824
DATE OF BIRTH	13/11/1998
I/C NUMBER	981113105056

Update

Interface edit profile

Personal Details

Nurhafizah Shukri

fizah@gmail.com

0142294824

13/11/1998 981113105056

.....

Submit

patientViewProfile.jsp

```

<form action="PatientController" method="post">
    <input type="hidden" name = "function" value="updateUser">
    <table align="center" border="1">
        <tr>
            <th bgcolor="#008A8A">NAME</th>
            <td><c:out value = "${patientInfo.name}" /></td>
        </tr>
        <tr>
            <th bgcolor="#008A8A">E-MAIL</th>
            <td><c:out value = "${patientInfo.email}" /></td>
        </tr>
        <tr>
            <th bgcolor="#008A8A">PHONE NUMBER</th>
            <td><c:out value = "${patientInfo.phone}" /></td>
        </tr>
        <tr>
            <th bgcolor="#008A8A">DATE OF BIRTH</th>
            <td><c:out value = "${patientInfo.DOB}" /></td>
        </tr>
        <tr>
            <th bgcolor="#008A8A">I/C NUMBER</th>
            <td><c:out value = "${patientInfo.ic}" /></td>
        </tr>
    </table>
    <br><br>
    <p align="center"><button class="btn btn-primary btn-lg" type="submit">Update</button></p>
</form>

```

patientUpdateProfile.jsp

```

<form action="PatientController" method="post">
    <input type="hidden" name = "function" value="updateProfile">
    <div class="form-groups">
        <input type="text" class="form-control" name="name" value="<c:out value = "${patientInfo.name}" />" placeholder="Full Name">
    </div>
    <div class="form-groups">
        <input type="text" class="form-control" name="email" value="<c:out value = "${patientInfo.email}" />" placeholder="Email Address">
    </div>
    <div class="form-groups">
        <input type="text" class="form-control" name="phone" value="<c:out value = "${patientInfo.phone}" />" placeholder="Phone Number">
    </div>
    <div class="form-inline">
        <div class="form-group">
            <input type="text" class="form-control" name="DOB" value="<c:out value = "${patientInfo.DOB}" />" placeholder="DOB (e.g. DD/MM/YYYY)">
        </div>
        <div class="form-group">
            <input type="text" class="form-control" name="Ic" value="<c:out value = "${patientInfo.ic}" />" placeholder="IC (e.g. 99053103)">
        </div>
    </div>
    <div class="form-groups">
        <input type="password" class="form-control" name="password" value="<c:out value = "${patientInfo.password}" />" placeholder="Password">
    </div>
    <p align="center"><button class="btn btn-primary btn-lg" type="submit">Submit</button></p>
</form>

```

PatientController.java


```

private void displayUser (HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException, ClassNotFoundException, SQLException {

    HttpSession session = request.getSession(false);

    User currentUser = (User) session.getAttribute("user");

    user.setUserID(currentUser.getUserID());

    User name = userDAO.viewUser(user);

    request.setAttribute("patientInfo", user);
    RequestDispatcher dispatcher;
    dispatcher = request.getRequestDispatcher("/patient/patientViewProfile.jsp");
    dispatcher.forward(request, response);
}

```

```

private void updateUser (HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException, ClassNotFoundException, SQLException {

    HttpSession session = request.getSession(false);

    User currentUser = (User) session.getAttribute("user");

    user.setUserID(currentUser.getUserID());

    User name = userDAO.viewUser(user);

    request.setAttribute("patientInfo", user);
    RequestDispatcher dispatcher;
    dispatcher = request.getRequestDispatcher("/patient/patientUpdateProfile.jsp");
    dispatcher.forward(request, response);
}

```

```

private void updateProfile(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException, SQLException, ClassNotFoundException {

    HttpSession session = request.getSession(false);

    User currentUser = (User) session.getAttribute("user");

    String name = request.getParameter("name");
    String email = request.getParameter("email");
    String phone = request.getParameter("phone");
    String DOB = request.getParameter("DOB");
    String ic = request.getParameter("ic");
    String password = request.getParameter("password");

    user.setUserID(currentUser.getUserID());
    user.setName(name);
    user.setEmail(email);
    user.setPhone(phone);
    user.setDOB(DOB);
    user.setIc(ic);
    user.setPassword(password);

    //inset data to database
    userDAO.updateUserProfile(user);
    displayUser(request, response);
}

```

UserDAO.java

```

public User viewUser(User user) throws ClassNotFoundException, SQLException {
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    try{
        //get connection
        Class.forName(driver);
        con = DriverManager.getConnection(url, userName, password);
        //set query
        String query = "SELECT * FROM user "
            + "WHERE ID=?";
        ps = con.prepareStatement(query);
        ps.setInt(1, user.getUserID());
        rs = ps.executeQuery();
        while(rs.next()){

            String name = rs.getString("name");
            String email = rs.getString("email");
            String phone = rs.getString("phone");
            String DOB = rs.getString("DOB");
            String ic = rs.getString("ic");
            String password = rs.getString("password");

            user.setName(name);
            user.setEmail(email);
            user.setPhone(phone);
            user.setDOB(DOB);
            user.setIc(ic);
            user.setPassword(password);
        }
    }finally{
        con.close();
    }
    return user;
}

```

```

public User updateUserProfile(User user) throws ClassNotFoundException, SQLException{
    //User user = new User();
    Connection con = null;
    PreparedStatement ps = null;
    //ResultSet rs = null;

    try{
        //get connection
        Class.forName(driver);
        con = DriverManager.getConnection(url, userName, password);

        //set query
        String query = "UPDATE user "
            + "SET name = ?, email = ?, phone = ?, DOB = ?, ic = ?, password = ?"
            + "WHERE ID= ?";

        ps = con.prepareStatement(query);

        ps.setString(1, user.getName());
        ps.setString(2, user.getEmail());
        ps.setString(3, user.getPhone());
        ps.setString(4, user.getDOB());
        ps.setString(5, user.getIc());
        ps.setString(6, user.getPassword());
        ps.setInt(7, user.getUserID());
        ps.execute();

    }finally{
        con.close();
    }

    return user;
}

```

5.4 Consistent GUI

For the consistency of GUI, we implement UI elements that are commonly used such as menu bars, scroll bars, drop-down list and radio button. Besides that, we also reserve commonly used locations for various graphical elements such as having the logo on the top left and login/logout on the top right. Furthermore, we make sure that our website visual elements are consistent whereby the background and colours are harmony and consistent. The language that we used on our site shows familiarity and consistency to anyone who has used any other website in the past.

Interfaces

The first screenshot shows the Medilab home page with a navigation bar containing links: HOME, SERVICES, ABOUT, TESTIMONIAL, CONTACT, and LOGIN. The main content area features the Medilab logo, the slogan "HEALTHCARE AT YOUR DESK!!", a placeholder text, and a "Make an Appointment" button.

The second screenshot shows the "Book Appointment" form. The navigation bar now includes PROFILE, BOOK APPOINTMENT, VIEW APPOINTMENT, PRESCRIPTION, and LOGOUT. The form fields are: a date picker (31/mm/yyyy), a "Time" input field, a dropdown menu for "Dr. Ran", a "Symptoms (100 Characters max)" text area, and a "Submit" button.

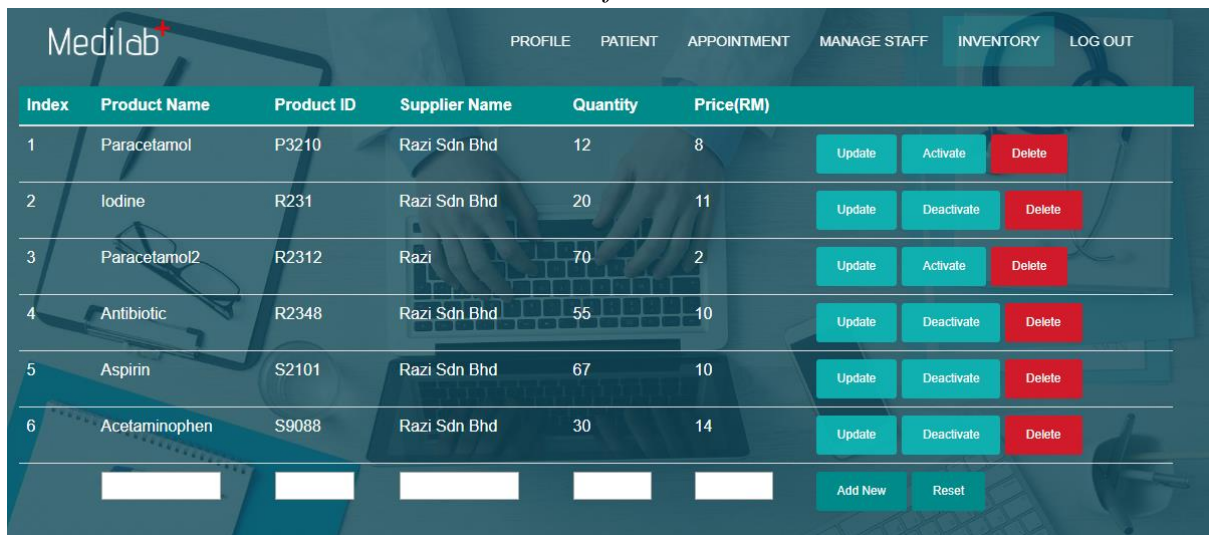
The third screenshot shows the "PATIENT" management interface. The navigation bar includes PROFILE, PATIENT, APPOINTMENT, MANAGE STAFF, INVENTORY, and LOG OUT. Below the navigation bar is a table with patient data and a "View Patient" button for each entry.

Index	User ID	Name	Email	Appointment Counts	
1	2	bellarina chew	bellarinachew1118@gmail.com	8	View Patient
2	8	Nurhafizah Shukri	fizah@gmail.com	1	View Patient

5.5 Data management - CRUD + activation

For this element, we implement it inside the inventory part where the admin can activate any item when the item is available or deactivate it when the item is out of stock. Everytime, pharmacist prescribe a medicine to a patient the amount of stock will be decreased so the admin can keep the inventory updated. Besides that, admin can add or delete any product to make sure the pharmacist and doctor does not prescribe any out of stock items.

Interfaces



The screenshot displays the 'Medilab' application interface, specifically the 'INVENTORY' section. The top navigation bar includes links for PROFILE, PATIENT, APPOINTMENT, MANAGE STAFF, INVENTORY (highlighted), and LOG OUT. The main content area features a table with columns: Index, Product Name, Product ID, Supplier Name, Quantity, and Price(RM). Below the table, there are five input fields and two buttons: 'Add New' and 'Reset'.

Index	Product Name	Product ID	Supplier Name	Quantity	Price(RM)			
1	Paracetamol	P3210	Razi Sdn Bhd	12	8	Update	Activate	Delete
2	Iodine	R231	Razi Sdn Bhd	20	11	Update	Deactivate	Delete
3	Paracetamol2	R2312	Razi	70	2	Update	Activate	Delete
4	Antibiotic	R2348	Razi Sdn Bhd	55	10	Update	Deactivate	Delete
5	Aspirin	S2101	Razi Sdn Bhd	67	10	Update	Deactivate	Delete
6	Acetaminophen	S9088	Razi Sdn Bhd	30	14	Update	Deactivate	Delete

Below the table, there are five input fields and two buttons: Add New, Reset.

adminInventory.jsp

```
<c:forEach items="${inventory}" var="currentinventory" varStatus="loop">

    <c:choose>

    <c:when test = "${currentinventory.product_id.equals(updateID)}">
        <tr>
            <form class="form-inv" action="AdminController" method="post">
                <input type="hidden" name="function" value="updateInventory">
                <td style = "color: white;" scope="col"><c:out value="${loop.index + 1}" /></td>
                <td style = "color: white;" scope="col"><p><input size="10" maxlength="20" type="text" name="product_name" id="product_name" value="${currentinventory.product_name}" required/></p></td>
                <td style = "color: white;" scope="col"><p><input size="5" maxlength="5" type="text" name="product_id" id="product_id" value="${currentinventory.product_id}" required/></p></td>
                <td style = "color: white;" scope="col"><p><input size="10" maxlength="25" type="text" name="supplier_name" id="supplier_name" value="${currentinventory.supplier_name}" required/></p></td>
                <td style = "color: white;" scope="col"><p><input min="1" max="100" type="number" name="stock" id="stock" value="${currentinventory.stock}" required/></p></td>
                <td style = "color: white;" scope="col"><p><input min="1" max="100" type="number" name="price" id="price" value="${currentinventory.price}" required/></p></td>
                <td style = "color: white;" scope="col">
                    <input type="hidden" name="status" value="${currentinventory.status}">
                    <input class="btn btn-inv" type="submit" name="button" id="button" value='Update'>
                </td>
            </form>
        </td>
    </tr>
    </c:when>

```

```
</c:choose>

<c:otherwise>
    <tr>
        <td style = "color: white;" scope="col"><c:out value="${loop.index + 1}" /></td>
        <td style = "color: white;" scope="col"><c:out value="${currentinventory.product_name}" /></td>
        <td style = "color: white;" scope="col"><c:out value="${currentinventory.product_id}" /></td>
        <td style = "color: white;" scope="col"><c:out value="${currentinventory.supplier_name}" /></td>
        <td style = "color: white;" scope="col"><c:out value="${currentinventory.stock}" /></td>
        <td style = "color: white;" scope="col"><c:out value="${currentinventory.price}" /></td>
        <td style = "color: white;" scope="col">
            <form class="form-inv" action="AdminController" method="post">
                <input type="hidden" name="function" value="updateFormInventory">
                <input type="hidden" name="product_id" value="${currentinventory.product_id}">
                <input class="btn btn-inv" type="submit" name="button" id="button" value='Update'>
            </form>

            <c:if test = "${currentinventory.status == 'inactive'}">
                <form class="form-inv" action="AdminController" method="post">
                    <input type="hidden" name="function" value="activationInventory">
                    <input type="hidden" name="product_id" value="${currentinventory.product_id}">
                    <input class="btn btn-inv" type="submit" name="button" id="button" value='Activate'>
                </form>
            </c:if>

            <c:if test = "${currentinventory.status == 'active'}">
                <form class="form-inv" action="AdminController" method="post">
                    <input type="hidden" name="function" value="deactivationInventory">
                    <input type="hidden" name="product_id" value="${currentinventory.product_id}">
                    <input class="btn btn-inv" type="submit" name="button" id="button" value='Deactivate'>
                </form>
            </c:if>

            <c:if>
                <form class="form-inv" action="AdminController" method="post">
                    <input type="hidden" name="function" value="deleteInventory">
                    <input type="hidden" name="product_id" value="${currentinventory.product_id}">
                    <input class="btn btn-inv" style="background-color: #d11a2a" type="submit" name="button" id="button" value='Delete'>
                </form>
            </c:if>
        </td>
    </tr>
</c:otherwise>

```

```
</c:choose>

</c:forEach>

<tr>
    <form action="AdminController" method="post">
        <input type="hidden" name="function" value="addInventory">
        <input type="hidden" name="status" value="active">

        <td style = "color: white;" scope="col"></td>
        <td style = "color: white;" scope="col"><p><input size="10" maxlength="20" type="text" name="product_name" required/></p></td>
        <td style = "color: white;" scope="col"><p><input size="5" maxlength="5" type="text" name="product_id" required/></p></td>
        <td style = "color: white;" scope="col"><p><input size="10" maxlength="25" type="text" name="supplier_name" required/></p></td>
        <td style = "color: white;" scope="col"><p><input min="1" max="100" type="number" name="stock" required/></p></td>
        <td style = "color: white;" scope="col"><p><input min="1" max="100" type="number" name="price" required/></p></td>
        <td style = "color: white;" scope="col">
            <input class="btn btn-inv" style="background-color: darkcyan" type="submit" name="button" id="button" value='Add New'>
            <input class="btn btn-inv" style="background-color: darkcyan" type="reset" name="button" id="button" value='Reset'>
        </td>
    </tr>
</tbody>
</table>

```

AdminController.java

```
private void deactivationInventory(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String product_id = request.getParameter("product_id");

    inventoryDAO.deactivationInventory(product_id);

    inventoryPage(request, response);
}

private void activationInventory(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String product_id = request.getParameter("product_id");

    inventoryDAO.activationInventory(product_id);

    inventoryPage(request, response);
}
```

```
private void addInventory(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String product_name = request.getParameter("product_name");
    String product_id = request.getParameter("product_id");
    String supplier_name = request.getParameter("supplier_name");
    int stock = Integer.parseInt(request.getParameter("stock"));
    int price = Integer.parseInt(request.getParameter("price"));
    String status = request.getParameter("status");

    inventory.setProduct_name(product_name);
    inventory.setProduct_id(product_id);
    inventory.setSupplier_name(supplier_name);
    inventory.setStock(stock);
    inventory.setPrice(price);
    inventory.setStatus(status);

    inventoryDAO.addInventory(inventory);

    inventoryPage(request, response);
}

private void deleteInventory(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String product_id = request.getParameter("product_id");

    inventoryDAO.deleteInventory(product_id);

    inventoryPage(request, response);
}
```

```
private void updateFormInventory(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    HttpSession session = request.getSession(false);

    String product_id = request.getParameter("product_id");

    session.setAttribute("updateID", product_id);

    inventoryPage(request, response);
}

private void updateInventory(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    HttpSession session = request.getSession(false);

    String product_name = request.getParameter("product_name");
    String product_id = request.getParameter("product_id");
    String supplier_name = request.getParameter("supplier_name");
    int stock = Integer.parseInt(request.getParameter("stock"));
    int price = Integer.parseInt(request.getParameter("price"));
    String status = request.getParameter("status");

    inventory.setProduct_name(product_name);
    inventory.setProduct_id(product_id);
    inventory.setSupplier_name(supplier_name);
    inventory.setStock(stock);
    inventory.setPrice(price);
    inventory.setStatus(status);

    inventoryDAO.updateInventory(inventory);

    session.setAttribute("updateID", "");
    inventoryPage(request, response);
}
```

inventoryDAO.java

```
public void activationInventory(String product_id) throws ClassNotFoundException, SQLException{
    //get connection
    Class.forName(driver);
    Connection con = DriverManager.getConnection(url, userName, password);

    //define query
    String query = "UPDATE inventory SET status = 'active' WHERE product_id = ?";

    //prepared Statment
    PreparedStatement ps = con.prepareStatement(query);

    //insert values
    ps.setString(1, product_id);

    //execute query
    ps.execute();

    //close connection
    con.close();
}
```

```
public void deactivationInventory(String product_id) throws ClassNotFoundException, SQLException{

    //get connection
    Class.forName(driver);
    Connection con = DriverManager.getConnection(url, userName, password);

    //define query
    String query = "UPDATE inventory SET status = 'inactive' WHERE product_id = ?";

    //prepared Statment
    PreparedStatement ps = con.prepareStatement(query);

    //insert values
    ps.setString(1, product_id);

    //execute query
    ps.execute();

    //close connection
    con.close();
}
```

```
public void deleteInventory(String product_id) throws ClassNotFoundException, SQLException{

    //get connection
    Class.forName(driver);
    Connection con = DriverManager.getConnection(url, userName, password);

    //define query
    String query = "DELETE FROM inventory WHERE product_id = ?";

    //prepared Statment
    PreparedStatement ps = con.prepareStatement(query);

    //insert values
    ps.setString(1, product_id);

    //execute query
    ps.execute();

    //close connection
    con.close();
}
```



```

public void addInventory(Inventory inventory) throws ClassNotFoundException, SQLException{

    //get connection
    Class.forName(driver);
    Connection con = DriverManager.getConnection(url, userName, password);

    //define query
    String query = "INSERT INTO inventory(product_name, product_id, supplier_name, stock, p

    //prepared Statment
    PreparedStatement ps = con.prepareStatement(query);

    //insert values
    ps.setString(1, inventory.getProduct_name());
    ps.setString(2, inventory.getProduct_id());
    ps.setString(3, inventory.getSupplier_name());
    ps.setInt(4, inventory.getStock());
    ps.setInt(5, inventory.getPrice());
    ps.setString(6, inventory.getStatus());

    //execute query
    ps.execute();

    //close connection
    con.close();
}

```

```

public void updateInventory(Inventory inventory) throws ClassNotFoundException, SQLException{

    //get connection
    Class.forName(driver);
    Connection con = DriverManager.getConnection(url, userName, password);

    //define query
    String query = "UPDATE inventory SET product_name = ?, supplier_name = ?, stock = ?, price

    //prepared Statment
    PreparedStatement ps = con.prepareStatement(query);

    //insert values
    ps.setString(1, inventory.getProduct_name());
    ps.setString(2, inventory.getSupplier_name());
    ps.setInt(3, inventory.getStock());
    ps.setInt(4, inventory.getPrice());
    ps.setString(5, inventory.getStatus());
    ps.setString(6, inventory.getProduct_id());

    //execute query
    ps.execute();

    //close connection
    con.close();
}

```

```

public List<Inventory> viewInventory() throws SQLException, ClassNotFoundException{

    List<Inventory> inventory = new ArrayList<>();
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    Statement stmt = null;

    try{

        //Connect to database
        Class.forName(driver);
        con = DriverManager.getConnection(url, userName, password);

        //define query
        String query = "SELECT * FROM inventory";

        //execute query
        ps = con.prepareStatement(query);

        rs = ps.executeQuery();

        while(rs.next()){
            //get data from database
            String product_name = rs.getString("product_name");
            String product_id = rs.getString("product_id");
            String supplier_name = rs.getString("supplier_name");
            int stock = rs.getInt("stock");
            int price = rs.getInt("price");
            String status = rs.getString("status");

            Inventory inv = new Inventory(product_name, product_id, supplier_name, stock, price, status);

            inventory.add(inv);
        }

    }finally{
        con.close();
    }

    return inventory;
}

```

5.6 User transaction and history

For this element, we implement it inside booking appointment transaction and history. Everytime an appointment is booked, the patient, doctor that is choosed and admin will be able to display the transaction. All changes in the status will be displayed on all three users. For the history part, Admin have the authority to view all the user transaction and history. Also, admin can view the amount of appointment for each patients and monthly appointment counts.

Interfaces

The image displays two screenshots of the Medilab web application interface. The top screenshot shows the 'PATIENT' management screen, and the bottom screenshot shows the 'APPOINTMENT' management screen.

Top Screenshot: PATIENT Management

Navigation: PROFILE, PATIENT, APPOINTMENT, MANAGE STAFF, INVENTORY, LOG OUT

Index	User ID	Name	Email	Total Appointment
1	2	bellarina chew	bellarinachew1118@gmail.com	7
2	8	hafizah	hafizah@gmail.com	4

Buttons: View Patient Appointments

Index	Date	Time	Symptoms	Doctor	Status
1	2019-12-31	13:00	fever	Dr. Ran	Completed
2	2019-12-31	13:00	flu	Dr. Ran	Rejected
3	2019-12-31	11:00	headache	Dr. Ran	Approved
4	2019-12-31	11:00	stomachache	Dr. Dom	Pending

Buttons: Delete

Bottom Screenshot: APPOINTMENT Management

Navigation: PROFILE, PATIENT, APPOINTMENT, MANAGE STAFF, INVENTORY, LOG OUT

Index	Year	Month	Appointment Counts
1	2019	DECEMBER	10
2	2020	JANUARY	1

Buttons: View Appointments, Report

Index	Date	Time	Symptoms	Doctor	Status
1	2020-01-08	13:00	back pain	Dr. Ran	Approved

adminPatientHistory.jsp

```

<table class = "table text-left">
  <thead>
    <tr>
      <th style = "color: white;" >Index</th>
      <th style = "color: white;" scope="col">User ID</th>
      <th style = "color: white;" scope="col">Name</th>
      <th style = "color: white;" scope="col">Email</th>
      <th style = "color: white;" scope="col">Total Appointment</th>
      <th style = "color: white;" scope="col"></th>
    </tr>
  </thead>

  <c:forEach items="${patientHistory}" var="currentpatient" varStatus="loop">
    <tr>
      <th style = "color: white;" ><c:out value="${loop.index + 1}" /></th>
      <td style = "color: white;" >${currentpatient.userID}</td>
      <td style = "color: white;" >${currentpatient.name}</td>
      <td style = "color: white;" >${currentpatient.email}</td>
      <td style = "color: white;" >${currentpatient.totalAppointment}</td>
      <td style = "color: white;" >
        <form class="form-in" action="AdminController" method="post">
          <input type="hidden" name="function" value="patientHistory">
          <input type="hidden" name="id" value="${currentpatient.userID}">
          <input class="btn btn-in" type="submit" name="button" id="button" value='View Patient Appointments'>
        </form>
      </td>
    </tr>
  </c:forEach>
</table>

```

```

<c:if test = "${displayPatientHistory == 'yes'}">
  <table class = "table text-left">
    <thead>
      <tr background-color="">
        <th style = "color: white;" >Index</th>
        <th style = "color: white;" scope="col">Date</th>
        <th style = "color: white;" scope="col">Time</th>
        <th style = "color: white;" scope="col">Symptoms</th>
        <th style = "color: white;" scope="col">Doctor</th>
        <th style = "color: white;" scope="col">Status</th>
        <th style = "color: white;" scope="col"></th>
      </tr>
    </thead>

```

```

    <c:forEach var="tempBook" items="${aPatientHistory}" varStatus="loop">
      <tr>
        <th style = "color: white;" ><c:out value="${loop.index + 1}" /></th>
        <td style = "color: white;" >${tempBook.date}</td>
        <td style = "color: white;" >${tempBook.time}</td>
        <td style = "color: white;" >${tempBook.symptoms}</td>
        <td style = "color: white;" >${tempBook.doctorName}</td>
        <td style = "color: white;" >${tempBook.status}</td>
        <td style = "color: white;" >
          <form class="form-in" action="AdminController" method="post">
            <input type="hidden" name="function" value="deleteAppointment">
            <input type="hidden" name="booking_id" value="${tempBook.booking_id}">
            <input class="btn btn-in" style="background-color: #d1a2a2" type="submit" name="button" id="button" value='Delete'>
          </form>
        </td>
      </tr>
    </c:forEach>
  </table>
</c:if>

```

AdminController.java

```
private void adminPage(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    HttpSession session = request.getSession(false);

    String admin_email = (String) session.getAttribute("admin");

    user = userDAO.getUser(admin_email);

    session.setAttribute("user", user);

    List<User> getPatientHistory = userDAO.getPatientHistory();

    session.setAttribute("patientHistory", getPatientHistory);
    session.setAttribute("displayPatientHistory", "");

    RequestDispatcher r = request.getRequestDispatcher("/admin/adminPatientHistory.jsp");
    r.forward(request, response);
}

private void patientHistory(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    HttpSession session = request.getSession(false);

    int id = Integer.parseInt(request.getParameter("id"));

    List<Booking> getPatientHistory = bookingDAO.getAPatientHistory(id);

    session.setAttribute("aPatientHistory", getPatientHistory);

    session.setAttribute("displayPatientHistory", "yes");

    RequestDispatcher r = request.getRequestDispatcher("/admin/adminPatientHistory.jsp");
    r.forward(request, response);
}

private void deleteAppointment(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    int booking_id = Integer.parseInt(request.getParameter("booking_id"));

    prescriptionDAO.deletePrescriptionBooking(booking_id);
    bookingDAO.cancelAppointment(booking_id);

    RequestDispatcher r = request.getRequestDispatcher("/admin/adminPatientHistory.jsp");
    r.forward(request, response);
}
```

UserDAO.java

```
public List<User> getPatientHistory() throws ClassNotFoundException, SQLException{
    List<User> userList = new ArrayList<>();
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;

    try{
        //get connection
        Class.forName(driver);
        con = DriverManager.getConnection(url, userName, password);

        //set query
        String query = "SELECT user.ID, user.name, user.email, COUNT(booking.booking_id) AS totalAppointment "
            + "FROM user "
            + "JOIN booking ON user.ID=booking.user_id "
            + "WHERE booking.user_id=user.ID "
            + "GROUP BY user.ID";

        ps = con.prepareStatement(query);

        rs = ps.executeQuery();

        while(rs.next()){
            //get parameters
            int userID = rs.getInt("user.ID");
            String name = rs.getString("user.name");
            String email = rs.getString("user.email");
            int totalAppointment = rs.getInt("totalAppointment");

            //set parameter to class
            User user = new User(userID, name, email, totalAppointment);
            userList.add(user);
        }

    }finally{
        con.close();
    }

    return userList;
}
```

BookingDAO.java

```
public List<Booking> getAPatientHistory(int id) throws ClassNotFoundException, SQLException{

    List<Booking> appointment = new ArrayList<>();
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    Statement smst = null;

    try{

        Class.forName(driver);
        con = DriverManager.getConnection(url, userName, password);

        String query = "SELECT booking.*, user.* "
            + "FROM booking "
            + "JOIN user ON booking.doctor = user.ID "
            + "WHERE user_id=?";

        //execute query
        ps = con.prepareStatement(query);

        //set user_id
        ps.setInt(1, id);

        rs = ps.executeQuery();

        while(rs.next()){
            //get data from database
            Date date = rs.getDate("booking.date");
            String time = rs.getString("booking.time");
            String symptoms = rs.getString("booking.symptoms");
            int doctor = rs.getInt("booking.doctor");
            String doctorName = rs.getString("user.name");
            String status = rs.getString("booking.status");
            int booking_id = rs.getInt("booking_id");

            Booking book = new Booking( date, symptoms, time, status, doctorName, doctor, booking_id);
            appointment.add(book);
        }

    }finally{
        con.close();
    }

    return appointment;
}
```

```
//patient cancel appointment
public void cancelAppointment(int book_id) throws ClassNotFoundException, SQLException{

    Class.forName(driver);
    Connection con = DriverManager.getConnection(url, userName, password);

    String query = "DELETE FROM booking "
        + "WHERE booking_id=? ";

    PreparedStatement ps = con.prepareStatement(query);

    ps.setInt(1, book_id);

    ps.execute();

    con.close();

}
```

PrescriptionDAO.java

```
public void deletePrescriptionBooking(int book_id) throws ClassNotFoundException, SQLException{

    Class.forName(driver);
    Connection con = DriverManager.getConnection(url, userName, password);

    String query = "DELETE FROM prescription "
        + "WHERE booking_id=? ";

    PreparedStatement ps = con.prepareStatement(query);

    ps.setInt(1, book_id);

    ps.execute();

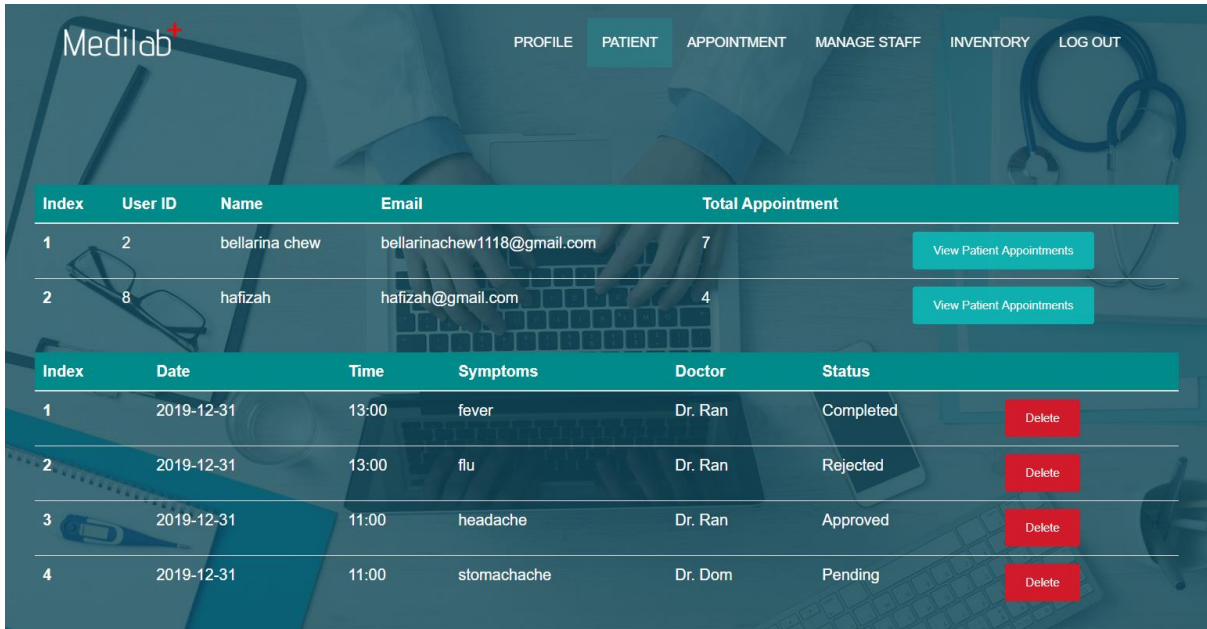
    con.close();

}
```


5.7 Admin manage user transaction

As we know, Admin had the authority to view all user transactions. So, for the patients' appointment history, Admin can delete the transaction if there happen to be human error done by the patients and doctors.

Interface



Index	User ID	Name	Email	Total Appointment
1	2	bellarina chew	bellarinachew1118@gmail.com	7
2	8	hafizah	hafizah@gmail.com	4

Index	Date	Time	Symptoms	Doctor	Status
1	2019-12-31	13:00	fever	Dr. Ran	Completed
2	2019-12-31	13:00	flu	Dr. Ran	Rejected
3	2019-12-31	11:00	headache	Dr. Ran	Approved
4	2019-12-31	11:00	stomachache	Dr. Dom	Pending

adminPatientHistory.jsp

```

<table class="table text-left">
  <thead>
    <tr>
      <th style="color: white;">Index</th>
      <th style="color: white;" scope="col">User ID</th>
      <th style="color: white;" scope="col">Name</th>
      <th style="color: white;" scope="col">Email</th>
      <th style="color: white;" scope="col">Total Appointment</th>
      <th style="color: white;" scope="col"></th>
    </tr>
  </thead>

  <c:forEach items="${patientHistory}" var="currentpatient" varStatus="loop">
    <tr>
      <th style="color: white;"><c:out value="${loop.index + 1}" /></th>
      <td style="color: white;">${currentpatient.userID}</td>
      <td style="color: white;">${currentpatient.name}</td>
      <td style="color: white;">${currentpatient.email}</td>
      <td style="color: white;">${currentpatient.totalAppointment}</td>
      <td style="color: white;">
        <form class="form-inline" action="AdminController" method="post">
          <input type="hidden" name="function" value="patientHistory">
          <input type="hidden" name="id" value="${currentpatient.userID}">
          <input class="btn btn-inverse" type="submit" name="button" id="button" value='View Patient Appointments'>
        </form>
      </td>
    </tr>
  </c:forEach>
</table>

```

```

<c:if test="${displayPatientHistory == 'yes'}">
  <table class="table text-left">
    <thead>
      <tr background-color="">
        <th style="color: white;">Index</th>
        <th style="color: white;" scope="col">Date</th>
        <th style="color: white;" scope="col">Time</th>
        <th style="color: white;" scope="col">Symptoms</th>
        <th style="color: white;" scope="col">Doctor</th>
        <th style="color: white;" scope="col">Status</th>
        <th style="color: white;" scope="col"></th>
      </tr>
    </thead>

```

```

    <c:forEach var="tempBook" items="${aPatientHistory}" varStatus="loop">
      <tr>
        <th style="color: white;"><c:out value="${loop.index + 1}" /></th>
        <td style="color: white;">${tempBook.date}</td>
        <td style="color: white;">${tempBook.time}</td>
        <td style="color: white;">${tempBook.symptoms}</td>
        <td style="color: white;">${tempBook.doctorName}</td>
        <td style="color: white;">${tempBook.status}</td>
        <td style="color: white;">
          <form class="form-inline" action="AdminController" method="post">
            <input type="hidden" name="function" value="deleteAppointment">
            <input type="hidden" name="booking_id" value="${tempBook.booking_id}">
            <input class="btn btn-inverse" style="background-color: #d11a2a" type="submit" name="button" id="button" value='Delete'>
          </form>
        </td>
      </tr>
    </c:forEach>
  </table>
</c:if>

```

AdminController.java

```
private void adminPage(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    HttpSession session = request.getSession(false);

    String admin_email = (String) session.getAttribute("admin");

    user = userDao.getUser(admin_email);

    session.setAttribute("user", user);

    List<User> getPatientHistory = userDao.getPatientHistory();

    session.setAttribute("patientHistory", getPatientHistory);
    session.setAttribute("displayPatientHistory", "");

    RequestDispatcher r = request.getRequestDispatcher("/admin/adminPatientHistory.jsp");
    r.forward(request, response);
}
```

```
private void patientHistory(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    HttpSession session = request.getSession(false);

    int id = Integer.parseInt(request.getParameter("id"));

    List<Booking> getPatientHistory = bookingDAO.getAPatientHistory(id);

    session.setAttribute("aPatientHistory", getPatientHistory);

    session.setAttribute("displayPatientHistory", "yes");

    RequestDispatcher r = request.getRequestDispatcher("/admin/adminPatientHistory.jsp");
    r.forward(request, response);
}
```

```
private void deleteAppointment(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    int booking_id = Integer.parseInt(request.getParameter("booking_id"));

    prescriptionDAO.deletePrescriptionBooking(booking_id);
    bookingDAO.cancelAppointment(booking_id);

    RequestDispatcher r = request.getRequestDispatcher("/admin/adminPatientHistory.jsp");
    r.forward(request, response);
}
```

UserDAO.java

```
public List<User> getPatientHistory() throws ClassNotFoundException, SQLException{
    List<User> userList = new ArrayList<>();
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;

    try{
        //get connection
        Class.forName(driver);
        con = DriverManager.getConnection(url, userName, password);

        //set query
        String query = "SELECT user.ID, user.name, user.email, COUNT(booking.booking_id) AS totalAppointment "
            + "FROM user "
            + "JOIN booking ON user.ID=booking.user_id "
            + "WHERE booking.user_id=user.ID "
            + "GROUP BY user.ID";

        ps = con.prepareStatement(query);

        rs = ps.executeQuery();

        while(rs.next()){
            //get parameters
            int userID = rs.getInt("user.ID");
            String name = rs.getString("user.name");
            String email = rs.getString("user.email");
            int totalAppointment = rs.getInt("totalAppointment");

            //set parameter to class
            User user = new User(userID, name, email, totalAppointment);
            userList.add(user);
        }

    }finally{
        con.close();
    }

    return userList;
}
```

BookingDAO.java

```
public List<Booking> getAPatientHistory(int id) throws ClassNotFoundException, SQLException{

    List<Booking> appointment = new ArrayList<>();
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    Statement smst = null;

    try{

        Class.forName(driver);
        con = DriverManager.getConnection(url, userName, password);

        String query = "SELECT booking.*, user.* "
            + "FROM booking "
            + "JOIN user ON booking.doctor = user.ID "
            + "WHERE user_id=?";

        //execute query
        ps = con.prepareStatement(query);

        //set user_id
        ps.setInt(1, id);

        rs = ps.executeQuery();

        while(rs.next()){
            //get data from database
            Date date = rs.getDate("booking.date");
            String time = rs.getString("booking.time");
            String symptoms = rs.getString("booking.symptoms");
            int doctor = rs.getInt("booking.doctor");
            String doctorName = rs.getString("user.name");
            String status = rs.getString("booking.status");
            int booking_id = rs.getInt("booking_id");

            Booking book = new Booking( date, symptoms, time, status, doctorName, doctor, booking_id);
            appointment.add(book);
        }

    }finally{
        con.close();
    }

    return appointment;
}
```

```
//patient cancel appointment
public void cancelAppointment(int book_id) throws ClassNotFoundException, SQLException{

    Class.forName(driver);
    Connection con = DriverManager.getConnection(url, userName, password);

    String query = "DELETE FROM booking "
        + "WHERE booking_id=? ";

    PreparedStatement ps = con.prepareStatement(query);

    ps.setInt(1, book_id);

    ps.execute();

    con.close();

}
```

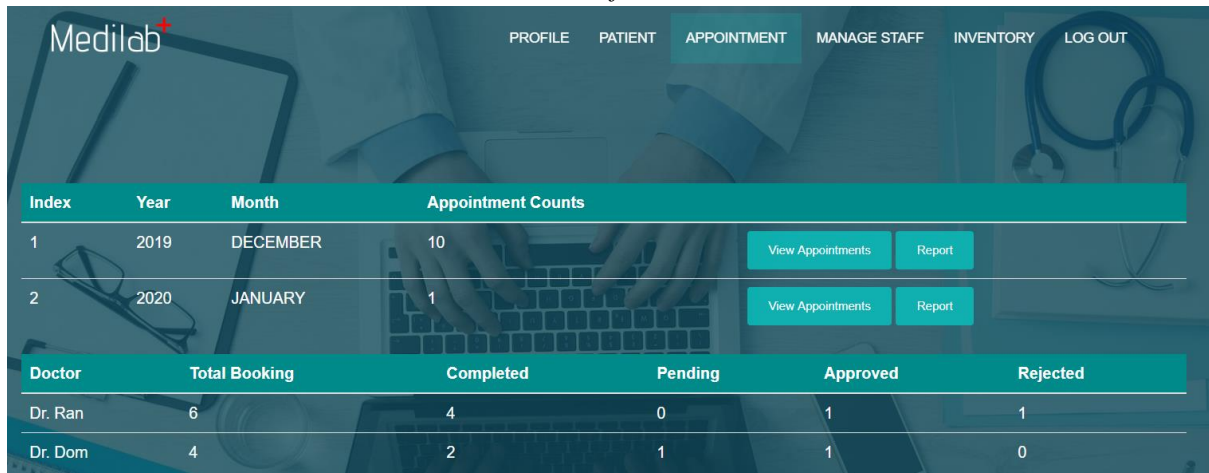
PrescriptionDAO.java

```
public void deletePrescriptionBooking(int book_id) throws ClassNotFoundException, SQLException{  
    Class.forName(driver);  
    Connection con = DriverManager.getConnection(url, userName, password);  
  
    String query = "DELETE FROM prescription "  
        + "WHERE booking_id=? ";  
  
    PreparedStatement ps = con.prepareStatement(query);  
  
    ps.setInt(1, book_id);  
  
    ps.execute();  
  
    con.close();  
}
```

5.8 Reporting

Reporting is where Admin can view the monthly report of the doctors' appointments. From the report, Admin can know which doctor completed higher appointment counts and who rejected more.

Interface



The screenshot shows the Medilab web application interface. At the top, there is a navigation bar with links: PROFILE, PATIENT, APPOINTMENT (highlighted), MANAGE STAFF, INVENTORY, and LOG OUT. Below the navigation bar, there is a table with the following data:

Index	Year	Month	Appointment Counts
1	2019	DECEMBER	10
2	2020	JANUARY	1

Below this table, there is another table showing appointment counts for two doctors:

Doctor	Total Booking	Completed	Pending	Approved	Rejected
Dr. Ran	6	4	0	1	1
Dr. Dom	4	2	1	1	0

adminAppointment.jsp

```
<div class="col-10">
  <table class="table text-left">
    <thead>
      <tr>
        <th style = "color: white;" scope="col">Index</th>
        <th style = "color: white;" scope="col">Year</th>
        <th style = "color: white;" scope="col">Month</th>
        <th style = "color: white;" scope="col">Appointment Counts</th>
        <th style = "color: white;" scope="col"></th>
      </tr>
    </thead>
    <tbody>
      <c:forEach items="${monthlyAptHistory}" var="currentappointment" varStatus="loop">
        <tr>
          <td style = "color: white;" ><c:out value="${loop.index + 1}" /></td>
          <td style = "color: white;" >${currentappointment.year}</td>
          <td style = "color: white;" >${currentappointment.getMonthString()}</td>
          <td style = "color: white;" >${currentappointment.appointmentCount}</td>
          <td style = "color: white;" >
            <form class="form-inline" action="AdminController" method="post">
              <input type="hidden" name="function" value="appointmentHistory">
              <input type="hidden" name="year" value="${currentappointment.year}">
              <input type="hidden" name="month" value="${currentappointment.month}">
              <input class="btn btn-inverse" type="submit" name="button" id="button" value="View Appointments">
            </form>
            <form class="form-inline" action="AdminController" method="post">
              <input type="hidden" name="function" value="appointmentReport">
              <input type="hidden" name="year" value="${currentappointment.year}">
              <input type="hidden" name="month" value="${currentappointment.month}">
              <input class="btn btn-inverse" type="submit" name="button" id="button" value="Report">
            </form>
          </td>
        </tr>
      </c:forEach>
    </tbody>
  </table>
```

```

<c:if test = "${displayAppointmentHistory == 'yes'}">
  <table class = "table text-left">
    <thead>
      <tr background-color="">
        <th style = "color: white;" scope="col">Index</th>
        <th style = "color: white;" scope="col">Date</th>
        <th style = "color: white;" scope="col">Time</th>
        <th style = "color: white;" scope="col">Symptoms</th>
        <th style = "color: white;" scope="col">Doctor</th>
        <th style = "color: white;" scope="col">Status</th>
      </tr>
    </thead>

    <c:forEach var="tempBook" items="${monthlyHistory}" varStatus="loop">
      <tr>
        <th style = "color: white;" ><c:out value="${loop.index + 1}" /></th>
        <td style = "color: white;" >${tempBook.date}</td>
        <td style = "color: white;" >${tempBook.time}</td>
        <td style = "color: white;" >${tempBook.symptoms}</td>
        <td style = "color: white;" >${tempBook.doctorName}</td>
        <td style = "color: white;" >${tempBook.status}</td>
      </tr>
    </c:forEach>

  </table>
</c:if>

```

```

<c:if test = "${displayAppointmentReport == 'yes'}">
  <table class = "table text-left">
    <thead>
      <tr background-color="">
        <th style = "color: white;" scope="col">Doctor</th>
        <th style = "color: white;" scope="col">Total Booking</th>
        <th style = "color: white;" scope="col">Completed</th>
        <th style = "color: white;" scope="col">Pending</th>
        <th style = "color: white;" scope="col">Approved</th>
        <th style = "color: white;" scope="col">Rejected</th>
      </tr>
    </thead>

    <c:forEach var="temp" items="${monthlyHistoryReport}" varStatus="loop">
      <tr>
        <td style = "color: white;" >${temp.doctorName}</td>
        <td style = "color: white;" >${temp.appointmentCount}</td>
        <td style = "color: white;" >${temp.completedCount}</td>
        <td style = "color: white;" >${temp.pendingCount}</td>
        <td style = "color: white;" >${temp.approvedCount}</td>
        <td style = "color: white;" >${temp.rejectedCount}</td>
      </tr>
    </c:forEach>
  </table>
</c:if>
</div>

```


AdminController.java

```
private void appointmentPage(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    HttpSession session = request.getSession(false);

    List<AppointmentHistory> getMonthlyAptHistory = bookingDAO.allMonthlyAppointment();

    session.setAttribute("monthlyAptHistory", getMonthlyAptHistory);

    session.setAttribute("displayAppointmentHistory", "");
    session.setAttribute("displayAppointmentReport", "");

    RequestDispatcher r = request.getRequestDispatcher("/admin/adminAppointment.jsp");
    r.forward(request, response);
}

private void appointmentHistory(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    HttpSession session = request.getSession(false);

    //set paramaters
    int year = Integer.parseInt(request.getParameter("year"));
    int month = Integer.parseInt(request.getParameter("month"));

    List<Booking> getMonthlyApt = bookingDAO.monthlyAppointment(year, month);

    session.setAttribute("monthlyHistory", getMonthlyApt);

    session.setAttribute("displayAppointmentHistory", "yes");
    session.setAttribute("displayAppointmentReport", "");

    RequestDispatcher r = request.getRequestDispatcher("/admin/adminAppointment.jsp");
    r.forward(request, response);
}

private void appointmentReport(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    HttpSession session = request.getSession(false);

    //set paramaters
    int year = Integer.parseInt(request.getParameter("year"));
    int month = Integer.parseInt(request.getParameter("month"));

    // number list of appoint for specific month
    List<AppointmentHistory> getMonthlyAptReport = bookingDAO.allMonthlyAppointmentReport(year, month);

    session.setAttribute("monthlyHistoryReport", getMonthlyAptReport);

    session.setAttribute("displayAppointmentReport", "yes");
    session.setAttribute("displayAppointmentHistory", "");

    RequestDispatcher r = request.getRequestDispatcher("/admin/adminAppointment.jsp");
    r.forward(request, response);
}
```

BookingDAO.java

```
//patient view appointment
public List<AppointmentHistory> allMonthlyAppointment() throws ClassNotFoundException,

    List<AppointmentHistory> aptHistory = new ArrayList<>();
    AppointmentHistory apt = new AppointmentHistory();
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    Statement smst = null;

    try{
        Class.forName(driver);
        con = DriverManager.getConnection(url, userName, password);

        String query = "SELECT doctor, YEAR(date) AS Year, MONTH(date) AS Month, "
            + "COUNT(date) AS AppointmentCount "
            + "FROM booking "
            + "GROUP BY YEAR(date), MONTH(date)";

        //execute query
        ps = con.prepareStatement(query);

        rs = ps.executeQuery();

        while(rs.next()){
            int year = rs.getInt("Year");
            int month = rs.getInt("Month");
            int aptMonth = rs.getInt("AppointmentCount");

            apt = new AppointmentHistory(year, month, aptMonth);
            aptHistory.add(apt);
        }

    }finally{
        con.close();
    }

    return aptHistory;
```

```

public List<Booking> monthlyAppointment(int year, int month) throws ClassNotFoundException, SQLException{

    List<Booking> aptHistory = new ArrayList<>();
    AppointmentHistory apt = new AppointmentHistory();
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    Statement smst = null;

    try{
        Class.forName(driver);
        con = DriverManager.getConnection(url, userName, password);

        String query = "SELECT booking.*, user.* "
            + "FROM booking JOIN user ON booking.doctor = user.ID "
            + "WHERE YEAR(date) = ? AND MONTH(date) = ?";

        //execute query
        ps = con.prepareStatement(query);

        ps.setInt(1, year);
        ps.setInt(2, month);

        rs = ps.executeQuery();

        while(rs.next()){
            //get data from database
            Date date = rs.getDate("booking.date");
            String time = rs.getString("booking.time");
            String symptoms = rs.getString("booking.symptoms");
            int doctor = rs.getInt("booking.doctor");
            String doctorName = rs.getString("user.name");
            String status = rs.getString("booking.status");
            int booking_id = rs.getInt("booking_id");

            Booking book = new Booking( date, symptoms, time, status, doctorName, doctor, booking_id);
            aptHistory.add(book);
        }

    }finally{
        con.close();
    }

    return aptHistory;
}

```

```

//patient view appointment
public List<AppointmentHistory> allMonthlyAppointmentReport(int year, int month) throws ClassNotFoundException, SQLException{

    List<AppointmentHistory> aptHistory = new ArrayList<>();
    AppointmentHistory apt = new AppointmentHistory();
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    Statement smst = null;
    UserDao userDao = new UserDao();

    try{
        Class.forName(driver);
        con = DriverManager.getConnection(url, userName, password);

        String query = "SELECT doctor, YEAR(date) AS Year, MONTH(date) AS Month, COUNT(date) AS AppointmentCount FROM booking "
            + "WHERE YEAR(date) = ? "
            + "AND MONTH(date) = ? "
            + "GROUP BY doctor";

        //execute query
        ps = con.prepareStatement(query);

        ps.setInt(1, year);
        ps.setInt(2, month);

        rs = ps.executeQuery();

        while(rs.next()){
            String doctorName = userDao.getName(rs.getInt("doctor"));
            int aptMonth = rs.getInt("AppointmentCount");

            int rejectedCount = this.monthlyAppointmentStatusCount(rs.getInt("doctor"), year, month, "Rejected");
            int approvedCount = this.monthlyAppointmentStatusCount(rs.getInt("doctor"), year, month, "Approved");
            int completedCount = this.monthlyAppointmentStatusCount(rs.getInt("doctor"), year, month, "Completed");
            int pendingCount = this.monthlyAppointmentStatusCount(rs.getInt("doctor"), year, month, "Pending");

            apt = new AppointmentHistory(doctorName, year, month, aptMonth, rejectedCount, approvedCount, completedCount, pendingCount);
            aptHistory.add(apt);
        }

    }finally{
        con.close();
    }

    return aptHistory;
}

```

5. REFLECTION

Our reflection for this project is we managed to implement most of the topics that we have learned in class which are from servlet until JSTL. Firstly, we implement the MVC design pattern into our project. MVC has 3 layers which are Model, Controller, and View. After using MVC, we can say that the separation between view logic (View) and business logic (Controller) actually helps each of us to understand the codes better. We can easily change or redesign the codes when we want to.

We also used JDBC MySQL to connect to our database. Besides, JDBC also managed to send queries and update statements to the database and retrieve and process the results received from the database in answer to the query. Since it is capable of reading any database and the only requirement for it to do so is the proper installation of all the drivers, it is easy for us to apply it in our project.

Furthermore, we used JSTL and EL in our JSP which allows us to get the data from the entity or beans. We can easily display the data without using too many scriptlets or out method. Besides, we use HttpSession for session management which can store any kind of object into a session such as a text or a database. Besides, it is secure since all data will operate on the server-side.

In conclusion, we managed to complete this project even though we had some difficulty throughout the project. With the help of others, we not only can complete our project, but we gained knowledge that somehow will be useful in the future.