# Health Care Appointment & Patient Visit Management System

Java with Spring Boot MVC Framework

(https://github.com/Nyein-Ko-Lat/HealthCareSystem)

Prepared by

NYEIN KO LAT (**GH1043541**)

MEng Computer Science

**GISMA UNIVERSITY OF APPLIED SCIENCES**

M604: Advanced Programming

Prof. Dr. Mazhar Hameed

## Table of Contents

# 1. Design and Implementation of a Health Care Appointment & Patient Visit Management System Using Spring Boot, MySQL and Thymeleaf

## Project Overview

The Healthcare Appointment & Patient Visit Management System is a web-based application developed using Spring Boot, MySQL, and Thymeleaf.

The system is designed to manage core healthcare workflows including patient registration, doctor scheduling, appointment booking, and patient visit completion.

The project intentionally demonstrates a hybrid database interaction approach using:

JPA/Hibernate for entity mapping and user related operation.

All other operations with JDBC Template process Stored Procedures and reporting.

# 2. System Architecture and Design

The application follows a 3-Tier MVC Architecture.

## Presentation Layer (Frontend)

- Thymeleaf + AdminLTE template
- HTML, JS, Forms

## Application Layer (Backend)

- Spring Boot
    - Config
        - Request http and security/ user login configuration
    - Controllers & REST Controller
        - Controller for all Operation
        - One REST controller reporting JS loading
    - Entities
        - Entity relationship for each tables
    - Repositories

- Database operation JDBC connect with Storeprocedures
- Services

## 3. Key Technologies

### 3.1. Backend

- Java 17
- Spring Boot
- Spring MVC
- Spring Data JPA
- JDBC Template
- Spring Security (Authentication only)

### 3.2. Frontend

- Thymeleaf
- Thymeleaf Layout Dialect
- AdminLTE 4.0.0 (Free HTML template)
- Flatpickr (Date/Time Picker)

### 3.3. Database

- MySQL
- Stored Procedures
- Views
- Foreign Key Constraints
- Triggers

## 4. Implementation

### 4.1. Hybrid Data Access Strategy

| Technology | Usage | Note |
|---|---|---|
| JPA | Entity mapping, and User controls | Only user related operation as I use Spring security |
| JDBC Template | Store procedures, Views | All operation include CRUD and reporting use SP and views |

### 4.2. Controller Example (Appointment)

```
package edu.gisma.gh1043541.healthcaresystem.controller;
```

```
import edu.gisma.gh1043541.healthcaresystem.entity.Appointment;
import edu.gisma.gh1043541.healthcaresystem.entity.Doctor;
import edu.gisma.gh1043541.healthcaresystem.service.*;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.time.LocalDateTime;
import java.util.List;

@Controller
@RequestMapping("/appointments")
public class AppointmentController {

    private final AppointmentService service;
    private final PatientService patientService;
    private final DoctorService doctorService;
    private final UserService userService;

    public AppointmentController(AppointmentService service,
                                 PatientService patientService,
                                 DoctorService doctorService,
                                 UserService userService) {
        this.service = service;
        this.patientService = patientService;
        this.doctorService = doctorService;
        this.userService = userService;
    }

    @GetMapping
    public String list(Model model) {
        model.addAttribute("appointments", service.findAll());
        return "appointment/index";
    }

    @GetMapping("/form")
    public String form(@RequestParam(required = false) Long id, Model
model) {
        model.addAttribute("drSpecialist",
StaticDataService.getDrSpecialist());
        Appointment a = id != null ? service.findById(id) : new
Appointment();
        model.addAttribute("appointment", a);
        model.addAttribute("patients", patientService.findAll());
        model.addAttribute("doctors", doctorService.findAll());
        return "appointment/form";
    }

    @PostMapping("/save")
    public String save(@ModelAttribute Appointment appointment, Model
model) {
        boolean availableSlot =
service.findAvailableSlot(appointment.getDoctor().getDoctorID(),appointment
.getAppointmentDate());
        //Check whether available time slot and book before 1 hours
        if(!availableSlot ||
appointment.getAppointmentDate().isBefore(LocalDateTime.now().plusHours(1L)
)) {
            model.addAttribute("error", "Selected time slot is not
available!");
```

```
            model.addAttribute("appointment", appointment);
            model.addAttribute("patients", patientService.findAll());
            model.addAttribute("doctors", doctorService.findAll());
            return "appointment/form"; // Back to form
        }
        String username = SecurityContextHolder
                .getContext()
                .getAuthentication()
                .getName();
        Long userId = userService.findByUsername(username).getId();
        appointment.setUpdatedBy(userId);
        appointment.setCreatedBy(userId);

        service.save(appointment);
        return "redirect:/appointments";
    }

    @GetMapping("/delete")
    public String delete(@RequestParam Long id) {
        service.delete(id);
        return "redirect:/appointments";
    }

    // ---------- Load doctors by speciality (AJAX) ----------
    @GetMapping("/get_doctors_by_speciality/{speciality}")
    @ResponseBody
    public List<Doctor> loadDoctors(@PathVariable String speciality) {
        return doctorService.findBySpecialityId(speciality);
    }
}
```

## 4.3. Repository Example (Appointment)

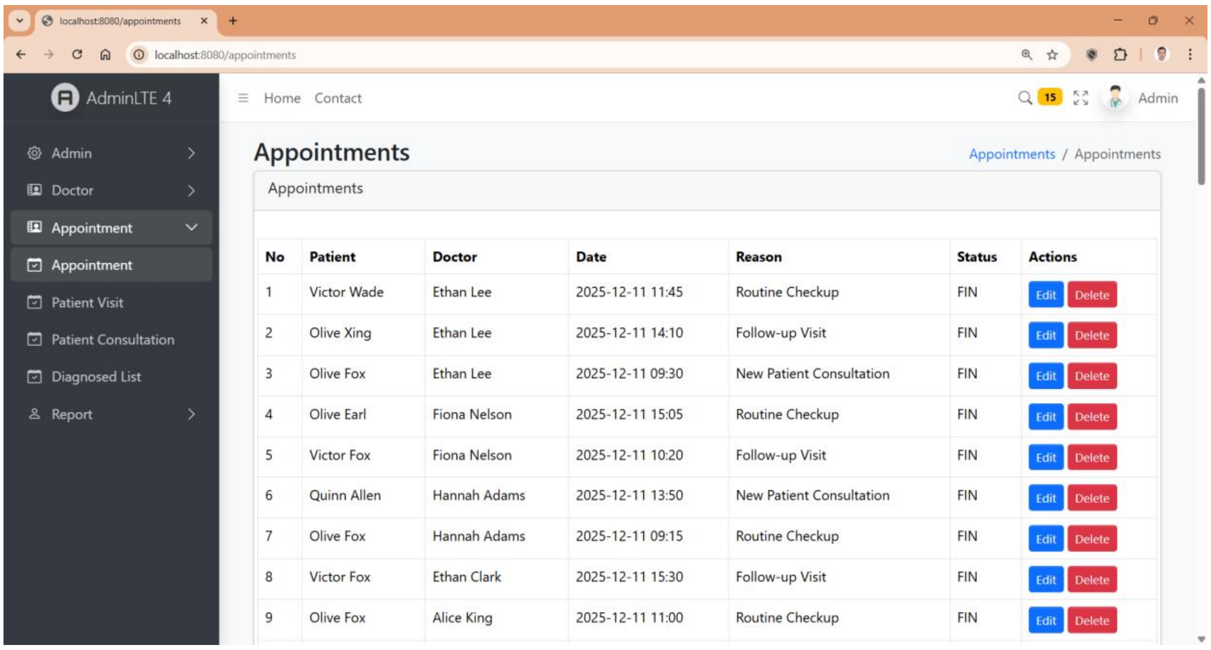
```
package edu.gisma.gh1043541.healthcaresystem.repository;

import edu.gisma.gh1043541.healthcaresystem.entity.Appointment;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.time.LocalDate;
import java.util.List;

@Repository
public class AppointmentRepository implements IBaseRepository<Appointment,
Long> {

    private final JdbcTemplate jdbc;

    public AppointmentRepository(JdbcTemplate jdbc) {
        this.jdbc = jdbc;
    }

    @Override
    public Appointment save(Appointment appointment) {
        String sql = "CALL sp_save_appointment(?, ?, ?, ?, ?, ?, ?, ?)";
        jdbc.update(sql,
                appointment.getAppointmentID() != null ?
appointment.getAppointmentID() : 0,
                appointment.getPatient().getPatientID(),
                appointment.getDoctor().getDoctorID(),
                appointment.getAppointmentDate(),
                appointment.getReason(),
                appointment.getStatusCode(),
                appointment.getCreatedBy(),
                appointment.getUpdatedBy()
        );
        return appointment;
    }

    @Override
    public Appointment findById(Long appointmentID) {
        String sql = "CALL sp_get_appointment_by_id(?)"; // create SP if
not exists
        return jdbc.queryForObject(sql, new Object[]{appointmentID}, (rs,
rowNum) -> fillAppointment(rs));
    }

    @Override
    public void delete(Long appointmentID) {
        String sql = "CALL sp_delete_appointment(?, ?)";
        jdbc.update(sql, appointmentID, 1); // 1 = system admin
    }

    @Override
    public void delete(Long appointmentID, Long updatedBy) {
        String sql = "CALL sp_delete_appointment(?, ?)";
        jdbc.update(sql, appointmentID, updatedBy); // assume 1 = system
user, replace with proper user
    }

    @Override
    public List<Appointment> findAll() {
        String sql = "CALL sp_list_appointments()";
        return jdbc.query(sql, (rs, rowNum) -> fillAppointment(rs));
    }

    private Appointment fillAppointment(ResultSet rs) throws SQLException {
        Appointment appointment = new Appointment();

        appointment.setAppointmentID(rs.getLong("Appointment ID"));
        appointment.setPatient(new
PatientRepository(jdbc).findById(rs.getLong("Patient ID")));
        appointment.setDoctor(new
DoctorRepository(jdbc).findById(rs.getLong("Doctor ID")));
        appointment.setAppointmentDate(rs.getTimestamp("Appointment
Date").toLocalDateTime());
        appointment.setReason(rs.getString("Reason"));
        appointment.setStatusCode(rs.getString("Status"));
        appointment.setCreatedBy(rs.getLong("Created By"));
        appointment.setUpdatedBy(rs.getLong("Updated By"));
        appointment.setCreatedAt(rs.getTimestamp("Created
At").toLocalDateTime());
        appointment.setUpdatedAt(rs.getTimestamp("Updated
At").toLocalDateTime());

        return appointment;
    }

    public List<Appointment> findApprovedAppointmentsByDate(LocalDate
appointmentDate) {
        String sql = "CALL sp_get_approved_appointment_by_date(?)";
        return jdbc.query(sql,new Object[]{appointmentDate} ,(rs, rowNum) -
> fillAppointment(rs));
    }
}
```

## 5. Results

### 5.1. Application Features Demonstrated

- Doctor schedules with weekly availability
- Appointment booking with validation
- Approved appointments auto-listed as bookings
- Patient visits creation
- All patient visits listed to patient diagnosed
- Complete diagnosed list (read only view)

### 5.2. Sample UI Output



Appointments

Appointments

| No | Patient | Doctor | Date | Reason | Status | Actions |
|---|---|---|---|---|---|---|
| 1 | Victor Wade | Ethan Lee | 2025-12-11 11:45 | Routine Checkup | FIN | Edit Delete |
| 2 | Olive Xing | Ethan Lee | 2025-12-11 14:10 | Follow-up Visit | FIN | Edit Delete |
| 3 | Olive Fox | Ethan Lee | 2025-12-11 09:30 | New Patient Consultation | FIN | Edit Delete |
| 4 | Olive Earl | Fiona Nelson | 2025-12-11 15:05 | Routine Checkup | FIN | Edit Delete |
| 5 | Victor Fox | Fiona Nelson | 2025-12-11 10:20 | Follow-up Visit | FIN | Edit Delete |
| 6 | Quinn Allen | Hannah Adams | 2025-12-11 13:50 | New Patient Consultation | FIN | Edit Delete |
| 7 | Olive Fox | Hannah Adams | 2025-12-11 09:15 | Routine Checkup | FIN | Edit Delete |
| 8 | Victor Fox | Ethan Clark | 2025-12-11 15:30 | Follow-up Visit | FIN | Edit Delete |
| 9 | Olive Fox | Alice King | 2025-12-11 11:00 | Routine Checkup | FIN | Edit Delete |

## 6. Challenges and Solutions

| Challenge | Solution | Note |
|---|---|---|
| Time Zone drift | Unified Berlin Time-zone | Difference between MySQL & Java setup drifted appointment time |
| SQL syntax, parameter error | Careful SP parameter matching | Using store procedure with JDBC call always creates miss match parameter while adding columns or |

| | | |
|---|---|---|
| | | remove from tables and SP |
| Entity & JPA auto create DDL additional columns | Change app config spring.jpa.hibernate.ddl-auto=none | As JPA auto create columns to column name with "_", however I create columns with Capital Letters. |

## 7. Conclusion and Future Work

### 7.1. Conclusion

- This project successfully demonstrates:
- Strong backend architecture
- Advanced SQL usage
- Realistic healthcare workflows
- Clean MVC-based UI integration

### 7.2. Future Enhancements

- Role-based access control
- Analytics dashboard
- Chat or messaging system
- Notification system (Email/SMS)
- Multi-clinic support

---

Git Repo - https://github.com/Nyein-Ko-Lat/HealthCareSystem

Video recording - GH1043541-Personal Finance Tracker.mp4