

## **ICT602 - Software Engineering**

# **SHAMS Developer Guide: Code Mapping**

**Complex Module:** Appointment Scheduling and Notifications

### **Group Members:**

**Shraddha Neel Diyora - 65430**

**Yashvi Viradiya - 65422**

**Sneh Vijaybhai Bhikadiya - 66186**

**Hiren Kanubhai - 65564**

# SHAMS Developer Guide: Code Mapping

This short developer guide accompanies Assessment 3 - Part A of the ICT602 submission, it is the immediate mapping of the conceptual design (Layered Architecture) and the Complex Module classes to their physical location in the codebase, which satisfies the demand to have documentation of code quality.

Layer	Key Classes / Files
Presentation Layer	ShamsGui.java, GuiApp.java
Service Layer	SchedulingService.java, NotificationService.java
Domain Layer	Appointment.java, Doctor.java, TimeSlot.java
Repository Layer	AppointmentRepository.java, InMemoryAppointmentRepository.java

## 1. Mapping Architecture (Layered Architecture)

The project is strictly based on the Layered (N-Tier) Architecture pattern.

Architecture Layer	Core Class Responsibility	File Path
<b>Presentation Layer</b>	Processes all user interaction, input sanitization and display (Swing GUI). In charge of making calls on the Service Layer.	src/main/java/au/vit/shams/gui/ShamsGui.java
<b>Application Entry</b>	May be called to initialize all the dependencies (Services, Repositories) and inject them into the Presentation Layer in accordance with Dependency Inversion.	src/main/java/au/vit/shams/GuiApp.java
<b>Service Layer</b>	All business logic, validation (conflict checking, availability) and intricate module orchestration.	src/main/java/au/vit/shams/service/SchedulingService.java

<b>Service Layer (Complex)</b>	Handles time based logic of appointment reminders and dispatch queues.	src/main/java/au/vit/shams/service/NotificationService.java
<b>Data Access Layer (DAL)</b>	Supplies Domain object persistence functions (CRUD), data storage. At the moment it is an in-memory implementation (Repository Pattern).	src/main/java/au/vit/shams/repository/InMemoryAppointmentRepository.java
<b>Domain Objects</b>	Basic POJOs (Plain Old Java Objects) of basic entities.	src/main/java/au/vit/shams/domain/* (e.g., Appointment.java, Doctor.java, TimeSlot.java)

## 2. Complex Module (Appointment Scheduling & Notifications)

The main functionality of the complex module is spread in three major components which are major in separation of concerns:

### A. SchedulingService (Business Logic)

This is the main group of appointment management and validation logic.

Functionality	Method in SchedulingService.java	Description
<b>Booking/Conflict Check</b>	book(AppointmentRequest request)	1. Verifies whether the time of appointment is in conflict with the schedule of the doctor. 2. Makes sure that it is in the working hours of the doctor. 3. Casts failure.
<b>Cancellation</b>	cancel(String appointmentId)	Removes and retrieves the appointment out of the repository.
<b>Listing</b>	listByDoctorAndDate(String doctorId, LocalDate date)	Gets all the appointments of a particular doctor on a particular date.

## B. NotificationService (Time based Logic)

The complicated logic of timing and dispatching reminders is dealt with in this class.

Functionality	Method in <code>NotificationService.java</code>	Description
<b>Reminder Queuing</b>	<code>onAppointmentBooked(Appointment appointment)</code>	Contacted on the spot of a successful booking. Enhances and stores(24 hours and 2 hours before) two Reminder objects in the internal queue.
<b>Dispatch Logic</b>	<code>dispatchDue()</code>	The essence of it is as follows: it goes through the internal queue, checking the reminder time against the current Clock time, "sends" (logs) reminders due, and removes them off of the queue.

## C. ShamsGui (Presentation Logic)

The user initiation of the functions of the complex module is done by the GUI.

Functionality	Method in <code>ShamsGui.java</code>	Action
<b>Book Appointment</b>	<code>book()</code>	Parses text fields and calls <code>schedulingService.book()</code> . On success, it calls <code>notificationService.onAppointmentBooked()</code> to queue reminders.
<b>Dispatch Reminders</b>	<code>dispatch()</code>	Calls <code>notificationService.dispatchDue()</code> . This is manually triggered in the GUI to demonstrate the background process.

## 3. Sample Data for Testing

The `InMemoryDoctorRepository.java` file contains sample data that is pre-loaded into the system. Test this data using the following required steps:

Doctor ID	Name	Working Hours
D1	Dr. G. Alex (Cardiologist)	09:00 to 17:00
D2	Dr. K. Smitha (Pediatrician)	10:00 to 18:00

## **Complex Module Test Scenario (Steps of Marker)**

### **1. Book Appointment:**

- Patient: Jane Doe
- Doctor ID: D1
- Start Time: Start time should be future time, e.g. [Current Date]T17:00.
- End Time: e.g. [Current Date]T17:30
- Click **Book**.
- The success message should be shown in the log window and the generated ID should be shown.

### **2. Validate Conflict Check:**

- Repeat the same information and select Book once again.
- The log is supposed to show an Error message that there is a conflict in appointment time.

### **3. Validate Out-of-Hours Booking:**

- Patient: Jane Doe
- Doctor ID: D1
- Start Time: Start time should be after the doctor's available time, e.g. after 17:00
- End Time: End time should also be after the doctor's available time, e.g. 17:30
- Click **Book**.
- The log is supposed to show an Error message that the time is OUT\_OF\_HOURS.

### **4. Validate Reschedule Appointment:**

- Use the appointment ID from the first booking the long auto generated ID.
- Enter appointment ID and other details under Reschedule/Cancel section as mentioned below:
  - Appointment ID: [e.g, 8969d16b-6761-4dba-af91-49eb77851f11 ]
  - Start Time: Start time should be future time, e.g. [Current Date]T17:00.
  - End Time: e.g. [Current Date]T17:30
- Click **Reschedule**.

- The log is supposed to show a success message that the appointment has been rescheduled to the new time.

## **5. Validate Cancel Appointment:**

- Use the appointment ID from the first booking.
- Enter appointment ID and other details under Reschedule/Cancel section as mentioned below:
  - Appointment ID: [e.g, 8969d16b-6761-4dba-af91-49eb77851f11 ]
- Click **Cancel By ID**.
- The log is supposed to show a message that the appointment has been **cancelled**.

## **6. Validate List Tomorrow's Appointments:**

- Enter Doctor ID in booking section: D1
- Click **List Tomorrow by Doctor**.
- The log is supposed to show all appointments for D1 on **next days on which days of you are checking**, or say **no appointments** if none exist.

## **7. Check Queuing of Notification - Complex module :**

**Preparation (OS Time):** In order to test the notification module, you will need to make an appointment in the near future (e.g. 5 minutes after the time when the test is taking place). The notification will be scheduled 2 hours before the appointment.

- Please follow the step 1 **Book Appointment** and book appointment for needed time.
- Note the current time. In case you have booked 2025-11-12T17:00, the reminder of 2 hours will be received at 2025-11-12T15:00.
- Wait until the present OS time surpasses the reminder time. (e.g. wait till after the 2 hour reminder time).
- Select Dispatch Reminders button.
- This should be showing in the log: Reminders sent: 1 (or 2, depending on the distance between the date at which a booking was made), which should confirm that the logic behind the notification is working.