

# **Case Study On**

## **MediBook – Doctor Appointment Booking System**

# INTRODUCTION

- ❖ MediBook is a web-based doctor appointment booking system that automates appointment scheduling.
- ❖ It allows patients to book, view, and cancel appointments online.
- ❖ The system reduces long waiting times and manual hospital processes.
- ❖ It minimizes administrative workload and improves appointment management.
- ❖ MediBook provides an efficient and user-friendly solution for patients and doctors.

# ABSTRAT

The **MediBook – Doctor Appointment Booking System** is developed to provide a seamless and user-friendly platform for managing doctor appointments online. Patients can register, log in, view available doctors, book appointments, view their booked appointments, and cancel appointments when required. Doctors can register, log in, and view the list of patients who have booked appointments with them, including appointment date and time. The application is developed using Spring Boot, Spring Data JPA (Hibernate), MySQL, Postman, and HTML/CSS/JavaScript for frontend interaction. This system ensures secure, role-based access and real-time appointment handling using REST APIs.

# CLIENT REQUIREMENT

- A web-based Doctor Appointment Booking System named **MediBook**.
- A system that allows patients to register, log in, and manage their appointments online.
- A doctor module where doctors can register, log in, and view appointments booked by patients.
- An appointment management module that enables patients to book, view, and cancel appointments.
- A role-based dashboard system for patients and doctors with separate functionalities.
- A platform that displays available doctors along with their specialization details.
- A scheduling mechanism to avoid appointment conflicts and overlapping bookings.

# TECHNICAL FEATURES

- RESTful APIs using Spring Boot
- Layered Architecture (Controller, Service, Repository)
- Session-based login handling (Session Storage)
- JSON-based data exchange
- CRUD operations
- Exception handling
- Role-based dashboard navigation

# TECHNOLOGIES AND TOOLS USED

## Backend

Java 17  
Spring Boot 3  
Spring Data JPA (Hibernate)  
Maven  
REST APIs  
Apache Tomcat (Embedded)

## Database

MySQL 8.0

## Frontend

HTML  
CSS  
JavaScript

## Tools

Spring Tool Suite (STS) 4  
Postman  
MySQL Workbench  
Google Chrome

# SYSTEM REQUIREMENTS

## Software Requirements

Operating System: Windows 10+

Java JDK 17

MySQL Server 8.0

Spring Tool Suite (STS)

Web Browser (Chrome)

## Hardware Requirements

Processor: Intel i3 or above

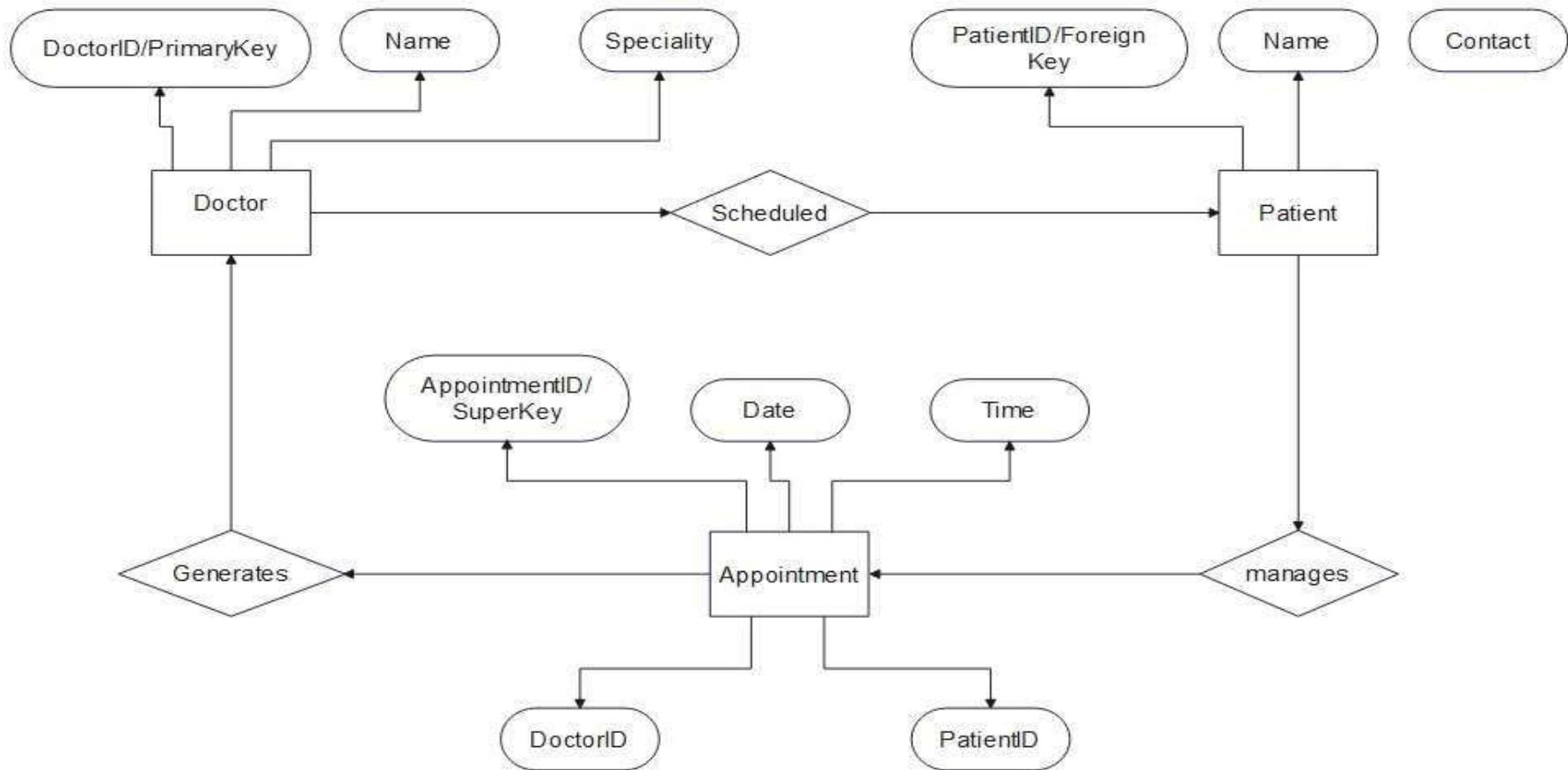
RAM: Minimum 4 GB

Hard Disk: 10 GB free space

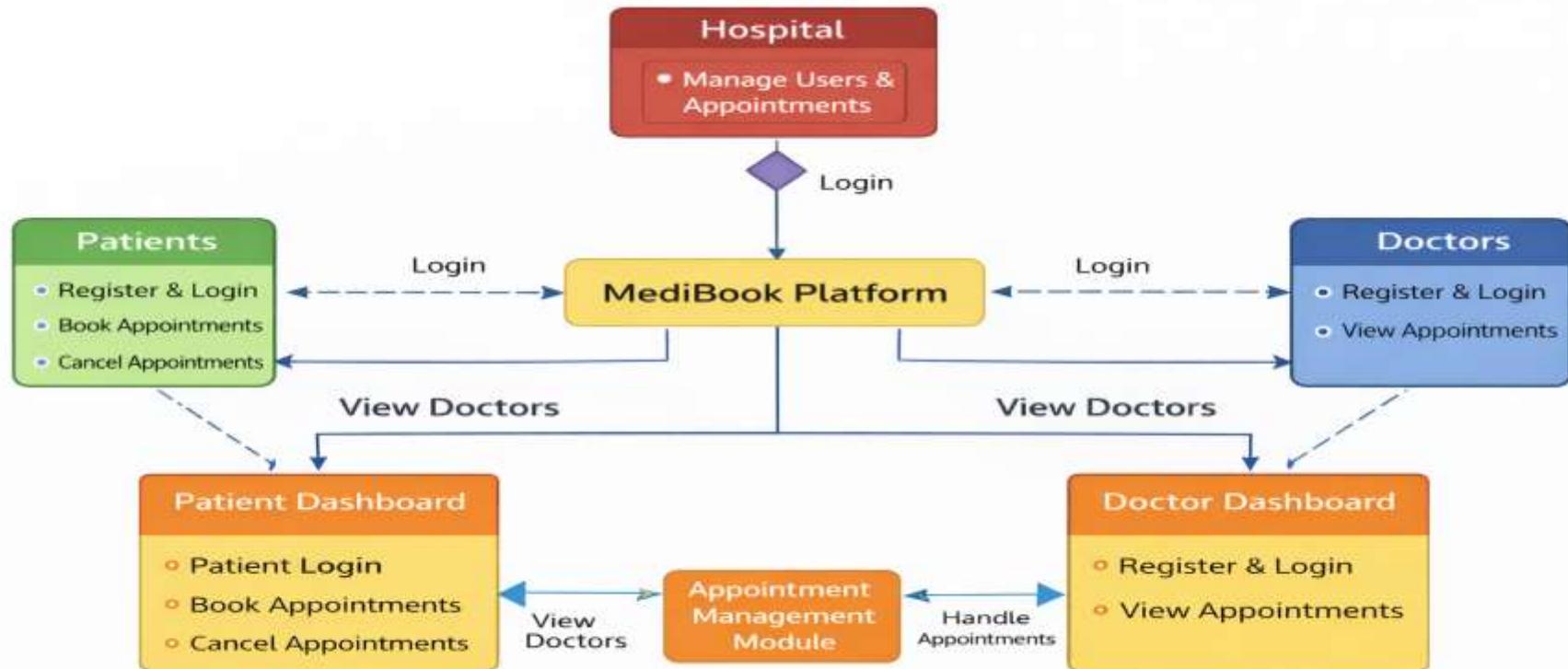
# PROJECT MODULE

- Patient Module
- Doctor Module
- Appointment Module
- Home / Navigation Module

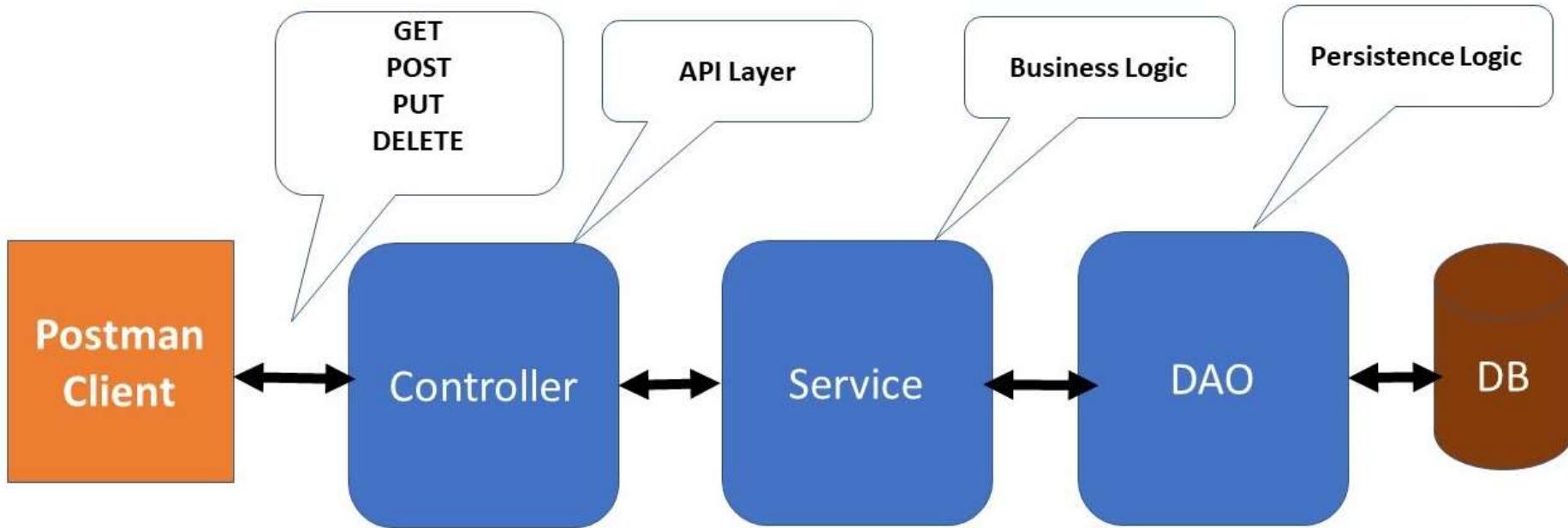
# ER DIAGRAM



# Client – Server Architecture



# Spring Boot APP



# HOME MODULE

Acts as the entry point of the **MediBook** system.

Provides navigation options for:

- [Patient](#)
- [Doctor](#)
- [Hospital](#)

Redirects users based on login status.

Improves usability by guiding users to the correct dashboard.

Ensures easy access to different system roles.

# HOSPITAL MODULE

- Allows hospital staff to view registered doctors.
- Displays doctor details such as name and specialization.
- Helps in managing overall appointment flow.
- Acts as a supervisory interface for hospital operations.
- Improves coordination between doctors and patients

# DOCTOR MODULE

- Allows doctors to register and log in to the system.
- Displays a personalized Doctor Dashboard.
- Enables doctors to view:
  - List of patients who booked appointments
  - Appointment date and time
- Helps doctors manage their daily schedules efficiently.
- Reduces manual tracking of patient appointments.

# PATIENT MODULE

- Allows patients to register and log in securely.
- Displays available doctors with specialization details.
- Enables patients to:
- Book appointments
- View booked appointments
- Cancel appointments if needed
- Prevents overlapping appointments.
- Provides a user-friendly interface for appointment management.

# Http Request Methods

HTTP request methods are used to perform different operations such as fetching data, creating records, and deleting records in the **MediBook** system.

GET	<a href="http://localhost:8080/api/doctors">http://localhost:8080/api/doctors</a>	View doctors
POST	<a href="http://localhost:8080/api/appointments/book">http://localhost:8080/api/appointments/book</a>	Book appointment
GET	<a href="http://localhost:8080/api/appointments/patient">http://localhost:8080/api/appointments/patient</a>	View patient appointments
GET	<a href="http://localhost:8080/api/appointments/doctor/id">http://localhost:8080/api/appointments/doctor/id</a>	View doctor appointments
DELETE	<a href="http://localhost:8080/api/appointments/id">http://localhost:8080/api/appointments/id</a>	Cancel appointment

# **DATA DICTIONARY**

# TABLES OF DATABASE

MySQL Workbench

Local instance MySQL90 - W...

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- amazon
- companydb
- crud
- hospitalmanagement
- medibook\_db
- rec\_company
- saturday
- sist
- student
- studentd
- studentdata
- studentdatabase
- sys
- test
- test29
- tests
- user\_management**
- Tables
- Views
- Stored Procedures
- Functions
- workerdata

amazon\* elipse crud SQL File 6\* SQL File 6\* SQL File 7\* SQL File 8\* SQL File 9\* SQL File 10\* ×

Limit to 1000 rows

```
4 • select * from appointments;
5 • SELECT *
6   FROM appointments
7   WHERE doctor_id = 1;
8
9 • SELECT d.name, d.specialization, a.appointment_date, a.appointment_time
10  FROM doctors d
11    JOIN appointments a
12      ON d.id = a.doctor_id;
13
14 • SHOW tables;
15
```

Result Grid | Filter Rows: Export: Wrap Cell Content: □

Tables_in_user_management
appointments
doctors
patients

Result Grid

# PATIENT DATABASE

MySQL Workbench

Local instance MySQL90 - W...

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- amazon
- companydb
- crud
- hospitalmanagement
- medibook\_db
- rec\_company
- saturday
- sit
- student
- studentd
- studentdata
- studentdatabase
- sys
- test
- test29
- tests
- user\_management**
- Tables
- Views
- Stored Procedures
- Functions
- workerdata

amazons\* elipses crud SQL File 6\* SQL File 7\* SQL File 8\* SQL File 9\* SQL File 10\* SQL File 11\* SQL Additions

Limit to 1000 rows

```
1 • use user_management;
2 • SELECT * FROM patients;
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

	id	email	name	password	phone
1	1	vij@gmail.com	Vij	vij@123	9876543210
2	2	kiran@gmail.com	Kiran	102938	9102378465
3	3	ram@gmail.com	Ram	ram@987	9451782543
4	4	sathish@gmail.com	Sathish	sat2987	9578643201
5	5	roshini@gmail.com	Roshini	roshini@987	9854702654
6	6	divya@gmail.com	Divya	divya@987	9765207639
7	7	keerthi@gmail.com	Keerthi	keerthi@987	8974702463

Result Grid

Form Editor

Field Types

Query Stats

Administration Schemas Information

# DOCTOR DATABASE

MySQL Workbench

Local instance MySQL80 - W...

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- amazon
- companydb
- crud
- hospitalmanagement
- medibook\_db
- rec\_company
- saturday
- sist
- student
- studentd
- studentdata
- studentdatabase
- sys
- test
- test29
- tests
- user\_management**

Tables

Views

Stored Procedures

Functions

workerdata

amazon\* ellipse crud SQL File 6\* SQL File 6\* SQL File 7\* SQL File 8\* SQL File 9\* SQL File 10\* SQL File 11\* x

Unit to 1000 rows

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid | Filter Rows: | Edit | Export/Import: | Wrap Cell Contents: |

	id	email	name	phone	specialization	password
1	1	raj@gmail.com	Dr. Raj	9876543210	Dermatologist	09876
2	2	amit@gmail.com	Dr. Amit	8976543210	Cardiologist	12345
3	3	devi@gmail.com	Dr. Devi	9745321014	Gynecologist	98765
4	4	maran@gmail.com	Dr. Maran	9567731843	Physiotherapist	54321
5	5	priya@gmail.com	Dr. Priya	9182736450	Dermatologist	76543

Rank Grid

Form Editor

Field Types

Query Stats

Administration Schemas Information

# APPOINTMENT DATABASE

MySQL Workbench

Local instance MySQL90 - W...

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- amazon\*
- elipse
- crud
- SQL File 6\*
- SQL File 7\*
- SQL File 8\*
- SQL File 9\*
- SQL File 10\*
- SQL File 11\*

SQLAdditions

Limit to 1000 rows

1 • use user\_management;

2 • SELECT Execute the statement under the keyboard cursor

3 • SELECT \* FROM doctors;

4 • select \* from appointments;

5

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid

Filter Rows

Edit

Export/Import

Wrap Cell Content

ID	appointment_date	appointment_time	doctor_id	patient_email	patient_phone	patient_id
2	2025-12-23	17:00:00	4	kiran@gmail.com	9102379465	2
3	2025-12-25	16:15:00	2	ram@gmail.com	9451782543	3
4	2025-12-26	10:20:00	5	ram@gmail.com	9451782543	3
5	2025-12-27	11:20:00	4	ram@gmail.com	9451782543	3
6	2025-12-30	09:10:00	2	sathish@gmail.com	9578643201	4
7	2026-01-03	20:05:00	3	roshini@gmail.com	9854702654	5
8	2026-01-06	19:35:00	1	roshini@gmail.com	9854702654	5
9	2025-12-20	12:45:00	4	divya@gmail.com	9765207639	6
10	2025-12-20	19:00:00	5	divya@gmail.com	9765207639	6
11	2025-12-30	18:30:00	3	keerthi@gmail.com	8974702463	7
12	2025-12-27	19:15:00	5	keerthi@gmail.com	8974702463	7

Result Grid

Form Editor

Field Types

Query Stats

Administration Schemas Information

# Appointments for a Specific Doctor

MySQL Workbench

Local Instance MySQL90 - W...

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- amazon
- companydb
- crud
- hospitalmanagement
- medibook\_db
- rec\_company
- saturday
- sit
- student
- studentd
- studentdata
- studentdatabase
- sys
- test
- test29
- tests
- user\_management**

Tables

Views

Stored Procedures

Functions

workerdata

SQL File 6\*

SQL File 7\*

SQL File 8\*

SQL File 9\*

SQL File 10\*

SQL File 11\*

Limit to 1000 rows

SQLAdditions

Jump to

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
1 • use user_management;
2 • SELECT * FROM patients;
3 • SELECT * FROM doctors;
4 • select * from appointments;
5
6 • SELECT *
  FROM appointments
  WHERE doctor_id = 4;
```

Result Grid | Filter Rows | Edit | Export/Import | Wrap Cell Contents

ID	Appointment Date	Appointment Time	Doctor ID	Patient Email	Patient Phone	Patient ID
2	2025-12-23	17:00:00	4	kiran@gmail.com	9102378465	2
5	2025-12-27	11:20:00	4	ram@gmail.com	9451782543	3
9	2025-12-20	12:45:00	4	divya@gmail.com	9765207639	6

Result Grid

Form Editor

Field Types

Query Stats

Administration Schemas Information

# HOSPITAL DATABASE

MySQL Workbench

Local instance MySQL90 - W...

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- amazon
- companydb
- crud
- hospitalmanagement
- medbook\_db
- rec\_company
- saturday
- sist
- student
- studentd
- studentdata
- studentdatabase
- sys
- test
- test29
- tests
- user\_management**
  - Tables
  - Views
  - Stored Procedures
  - Functions
- workerdata

amazon\* elipsa crud SQL File 5\* SQL File 6\* SQL File 7\* SQL File 8\* SQL File 9\* SQL File 10\* SQL File 11\* SQLAdditions

Limit to 1000 rows

```
3 * SELECT * FROM doctors;
4 * select * from appointments;
5 *
6 * SELECT *
7   FROM appointments
8   WHERE doctor_id = 4;
9 *
10 * SELECT d.name, d.specialization, a.appointment_date, a.appointment_time
11   FROM doctors d
12   JOIN appointments a
13   ON d.id = a.doctor_id;
14 *
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid Filter Rows Export Wrap Cell Content

name	specialization	appointment_date	appointment_time
Dr. Maran	Physiotherapist	2025-12-23	17:00:00
Dr. Amit	Cardiologist	2025-12-25	16:15:00
Dr. Priya	Dermatologist	2025-12-26	10:20:00
Dr. Maran	Physiotherapist	2025-12-27	11:20:00
Dr. Amit	Cardiologist	2025-12-30	09:10:00
Dr. Devi	Gynecologist	2026-01-03	20:05:00
Dr. Raj	Dermatologist	2026-01-06	19:35:00
Dr. Maran	Physiotherapist	2025-12-20	12:45:00
Dr. Priya	Dermatologist	2025-12-20	19:00:00
Dr. Devi	Gynecologist	2025-12-30	18:30:00
Dr. Priya	Dermatologist	2025-12-27	19:15:00

Result Grid Form Editor Field Types Query Data

Administration Schemas Information

# GET METHOD FOR VIEWING DOCTORS

The screenshot shows the Postman application interface. At the top, there are tabs for Home, Workspaces, and API Network. A search bar says "Search Postman". On the right, there are buttons for Invite, Upgrade, and a collection of tabs including "HTTP getDoctor", "POST loginEmail", "POST loginPhone", "GET getPatients", "POST bookAppointment", "GET getAppointmentDetails", "GET http://localhost:3001", "GET http://localhost:80", and "No environment". Below the tabs, a URL field shows "http://localhost:8080/api/doctors". The main area shows a GET request to "http://localhost:8080/api/doctors". The "Params" tab is selected, showing an empty table for query parameters. Other tabs include "Docs", "Authorization", "Headers (7)", "Body", "Scripts", and "Settings". The "Body" tab is expanded, showing a JSON response with three doctor objects. The response status is 200 OK, with 30 ms response time and 792 B size. The JSON data is as follows:

```
1. [
2.   {
3.     "id": 1,
4.     "name": "Dr. Raj",
5.     "specialization": "Dermatologist",
6.     "email": "raj@gmail.com",
7.     "phone": "9876543210",
8.     "password": "12345"
9.   },
10.  {
11.    "id": 2,
12.    "name": "Dr. Amit",
13.    "specialization": "Cardiologist",
14.    "email": "amit@gmail.com",
15.    "phone": "8976543218",
16.    "password": "12345"
17.  },
18.  {
19.    "id": 3,
20.    "name": "Dr. Devi",
21.    "specialization": "Orthopedist",
22.    "email": "devi@gmail.com",
23.    "phone": "9876543219",
24.    "password": "12345"
25.  }
]
```

# POST METHOD FOR BOOKING APPOINTMENT

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Home', 'Workspaces', and 'API Network'. A search bar says 'Search Postman' and there are 'Invite', 'Upgrade', and 'Logout' buttons. Below the header, a sidebar on the left lists collections, environments, history, and files. The main workspace shows a collection named 'http://localhost:8080/api/appointments/book' with several requests listed: 'getDoctor', 'post\_loginByEmail', 'post\_loginByPhone', 'getPatients', 'post\_bookAppointment' (which is highlighted in yellow), 'getAppointmentDetails', 'get http://localhost:8080', and 'POST http://localhost:8080'. The 'post\_bookAppointment' request has its URL set to 'http://localhost:8080/api/appointments/book'. The 'Body' tab is selected, showing a raw JSON payload:

```
1 {  
2   "doctorId": 1,  
3   "appointmentDate": "2025-01-08",  
4   "appointmentTime": "10:10",  
5   "patientEmail": "Eliza@gmail.com"  
6 }  
7
```

Below the body, the 'Body' tab is active, showing the response: '200 OK' with a response time of '38 ms' and a size of '168 B'. The response content is a single character 'E'.

# GET METHOD FOR VIEWING PATIENT APPOINTMENTS

The screenshot shows the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', 'API Network', 'Search Postman' (with a placeholder 'Ctrl + K'), and various status indicators like 'Invite', 'Upgrade', and 'No environment'. The left sidebar features sections for 'Collections', 'Environments', 'History', and 'Flows'. The main workspace displays a collection named 'http://localhost:8080/api/appointments/patient'. A specific GET request is selected, with the URL being 'http://localhost:8080/api/appointments/patient?email=ram@gmail.com'. The 'Headers' tab is active, showing an empty key-value pair. Below it, the 'Body' tab displays a JSON response with two appointment objects. The first appointment has an ID of 3, a doctor ID of 2, and a patient ID of 3. The second appointment has an ID of 4, a doctor ID of 5, and a patient ID of 3. Both entries include patient email ('ram@gmail.com' and 'ram@gmail.com'), patient phone ('98981782543' and '9451782543'), appointment date ('2028-12-26' and '2028-12-26'), appointment time ('16:15:00' and '18:30:00'), doctor name ('Dr. Amit' and 'Dr. Amit'), and patient name ('null' and 'null'). The bottom right corner shows the response status as '200 OK' with a duration of '20 ms' and a size of '781 B'.

HTTP Method: GET  
URL: http://localhost:8080/api/appointments/patient?email=ram@gmail.com

Headers:

Key	Value	Description
Key	Value	Description

Body:

```
1 [  
2   {  
3     "id": 3,  
4     "doctorId": 2,  
5     "patientId": 3,  
6     "patientEmail": "ram@gmail.com",  
7     "patientPhone": "98981782543",  
8     "appointmentDate": "2028-12-26",  
9     "appointmentTime": "16:15:00",  
10    "doctorName": "Dr. Amit",  
11    "patientName": null  
12  },  
13  {  
14    "id": 4,  
15    "doctorId": 5,  
16    "patientId": 3,  
17    "patientEmail": "ram@gmail.com",  
18    "patientPhone": "9451782543",  
19    "appointmentDate": "2028-12-26",  
20    "appointmentTime": "18:30:00",  
21  }]
```

# GET METHOD FOR VIEWING DOCTOR APPOINTMENTS

The screenshot shows the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', 'API Network', 'Search Postman', 'Invite', 'Upgrade', and tabs for various collections like 'getDoctor', 'postLoginByEmail', 'postLoginByPhone', 'getPatients', 'postBookAppointment', 'getAppointmentDetails', 'http://localhost:80', and 'http://localhost:80'. The left sidebar has sections for 'Collections', 'Environments', 'History', and 'Favorites'. The main workspace shows a GET request for 'http://localhost:8080/api/appointments/doctor/2'. The 'Headers' tab is selected, showing two key-value pairs: 'Key' and 'Value'. The 'Body' tab shows a JSON response with two appointment objects. The response status is '200 OK' with a duration of '29 ms' and a size of '559 B'. The JSON response is:

```
1  [
2    {
3      "id": 3,
4      "doctorId": 2,
5      "patientId": 3,
6      "patientEmail": "rao@gmail.com",
7      "patientPhone": "9451782943",
8      "appointmentDate": "2026-12-26",
9      "appointmentTime": "16:15:00",
10     "doctorName": null,
11     "patientName": "Rao"
12   },
13   {
14     "id": 6,
15     "doctorId": 2,
16     "patientId": 4,
17     "patientEmail": "nathi@gmail.com",
18     "patientPhone": "9578943291",
19     "appointmentDate": "2026-12-26",
20     "appointmentTime": "09:10:00",
21     "doctorName": null
22   }
]
```

# DELETE METHOD FOR CANCELLING APPOINTMENT

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Home', 'Workspaces', 'API Network', a search bar 'Search Postman', and various icons for 'Invite', 'Upgrade', and environment management. Below the navigation is a list of API endpoints: 'getDoctor', 'POST logInByEmail', 'POST logInByPhone', 'getPatients', 'POST bookAppointment' (which is highlighted in orange), 'getAppointmentDetails', 'HTTP://localhost:80', and 'HTTP://localhost:80'. A note 'No environment' is visible next to the environments icon.

The main area shows a single request: a 'DELETE' operation to 'http://localhost:8080/api/appointments/4'. The 'Headers' tab is selected, showing a table with one row: 'Key' (Content-Type) and 'Value' (application/json). Other tabs like 'Body', 'Cookies', and 'Test Results' are also present. The 'Send' button is at the top right of the request details.

In the 'Test Results' section, the status is '200 OK' with a response time of '12 ms' and a size of '198 B'. The response body contains the message: 'Appointment cancelled successfully'.

On the left side of the interface, there are several sidebar panels: 'Collections', 'Environments', 'History', 'Flows', and 'Filters BETA'. At the bottom, there are links for 'Cloud View', 'Find and replace', 'Console', 'Terminal', and 'Runner', along with icons for 'Start Proxy', 'Cookies', 'Vault', and 'Trash'.

# WORKING OF SPRING TOOL

workspace-spring-tools-for-eclipse-4.32.2.RELEASE - project/src/main/java/com/example/demo/controller/AppointmentController.java - Spring Tools for Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer    ProjectAppl...    Patient.java    Doctor.java    DoctorRepo...    PatientServ...    patient-datas...    AppointmentC...   

```
11  @RestController
12  @AllArgsConstructor
13  @RequestMapping("/api/appointments")
14  public class AppointmentController {
15
16      private AppointmentService appointmentService;
17
18      // To get appointments for a patient
19      @GetMapping("/patient")
20      public List<Appointment> getPatientAppointments(
21          @RequestParam(required = false) String email,
22          @RequestParam(required = false) String phone) {
23
24          return appointmentService.getAppointmentsByPatient(email, phone);
25      }
26
27      @PostMapping("/book")
28      public ResponseEntity<Boolean> bookAppointment(
29          @RequestBody Appointment appointment) {
30
31          boolean isBooked = appointmentService.bookAppointment(appointment);
32
33          return ResponseEntity.ok(isBooked);
34      }
35
36      @GetMapping("/doctor/{doctorId}")
37      public ResponseEntity<List<Appointment>> getAppointmentsForDoctor(
38          @PathVariable Long doctorId) {
39
40          return ResponseEntity.ok(appointmentService.getAppointmentsForDoctor(doctorId));
41      }
42  }
```

Problems    Javadoc    Declaration    Search    Console    Progress

Project - ProjectApplication [Spring Boot App] C:\Users\kavya.priya\OneDrive\Desktop\graph-4.32.2.RELEASE\STS\org.eclipse.jdt.ls.core匮乏\http://127.0.0.1:21000/v2/251105-0741\src\main\java\com\example\demo\controller\AppointmentController.java [v4.6.0]

Spring Root (v4.6.0)

```
2025-12-17T12:02:02.001+05:30 INFO [restartedMain] com.example.demo.ProjectApplication : Starting ProjectApplication using Java 21.0.9
2025-12-17T12:02:02.008+05:30 INFO [restartedMain] com.example.demo.ProjectApplication : No active profile set, falling back to 1 default
2025-12-17T12:02:02.178+05:30 INFO [restartedMain] o.DevtoolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring-boot-devtools.property.defaults.active=true' for additional web related logging consider set
2025-12-17T12:02:02.178+05:30 INFO [restartedMain] o.DevtoolsPropertyDefaultsPostProcessor : Bootstrapping Spring Data JPA repositories in 0 ms
2025-12-17T12:02:03.704+05:30 INFO [restartedMain] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 106 ms
2025-12-17T12:02:03.823+05:30 INFO [restartedMain] s.d.r.c.RepositoryConfigurationDelegate :
```

# WEBSITE DRIVE LINK

Link: <https://drive.google.com/file/d/1wN6ZVIBeZeLfZAu-ZThr-UuoN0sXMshQ/view?usp=sharing>