

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

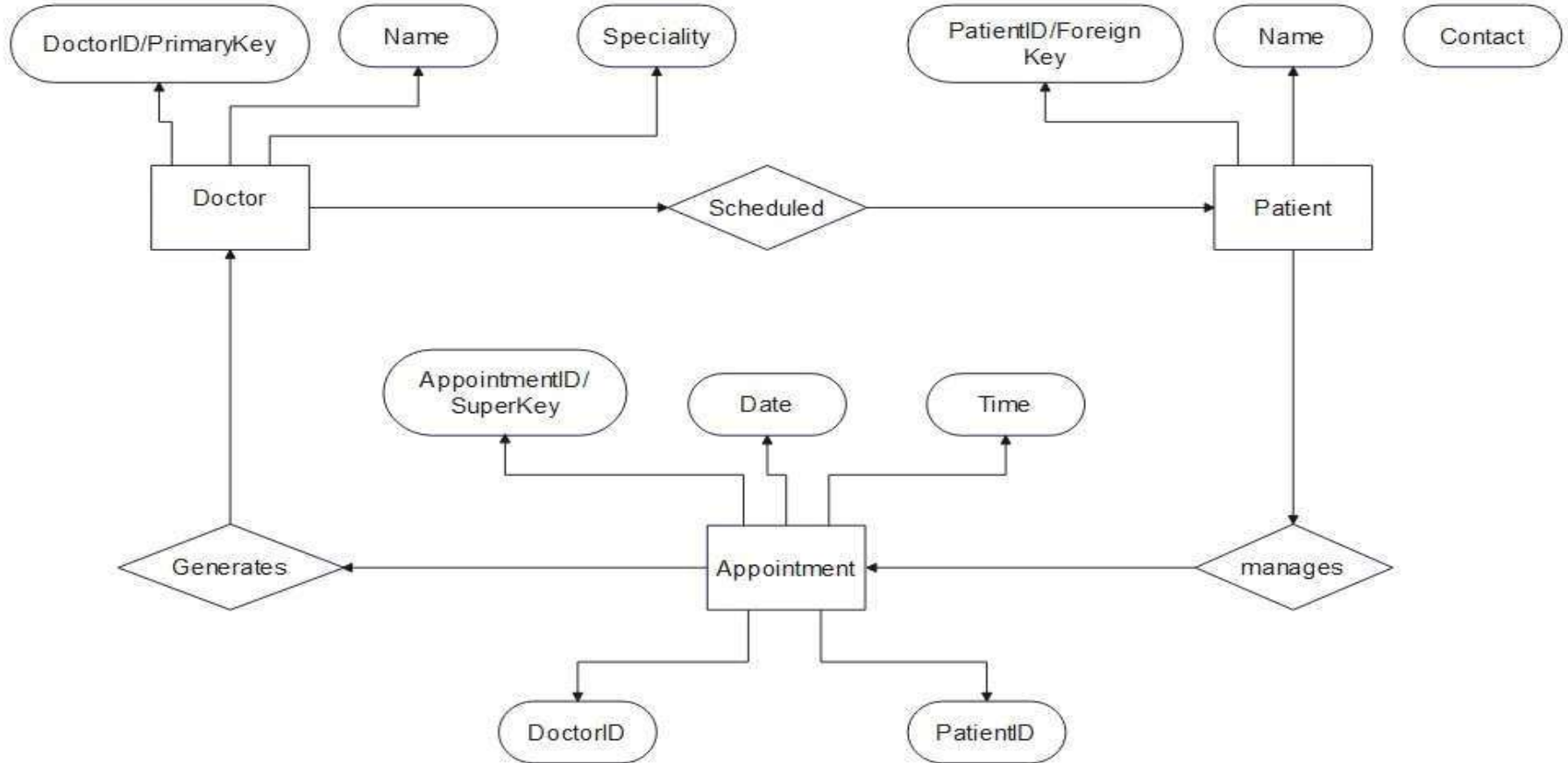
RAM: Minimum 4 GB

Hard Disk: 10 GB free space

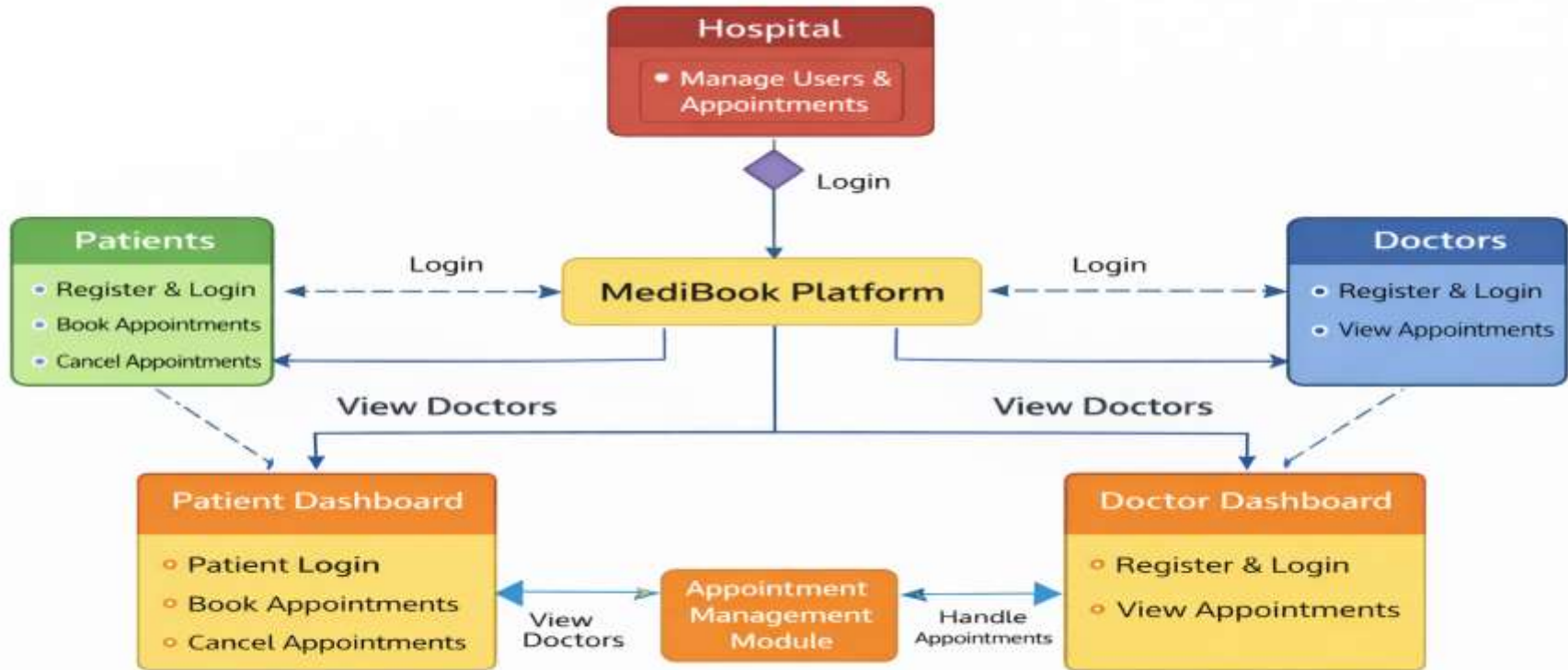
PROJECT MODULE

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

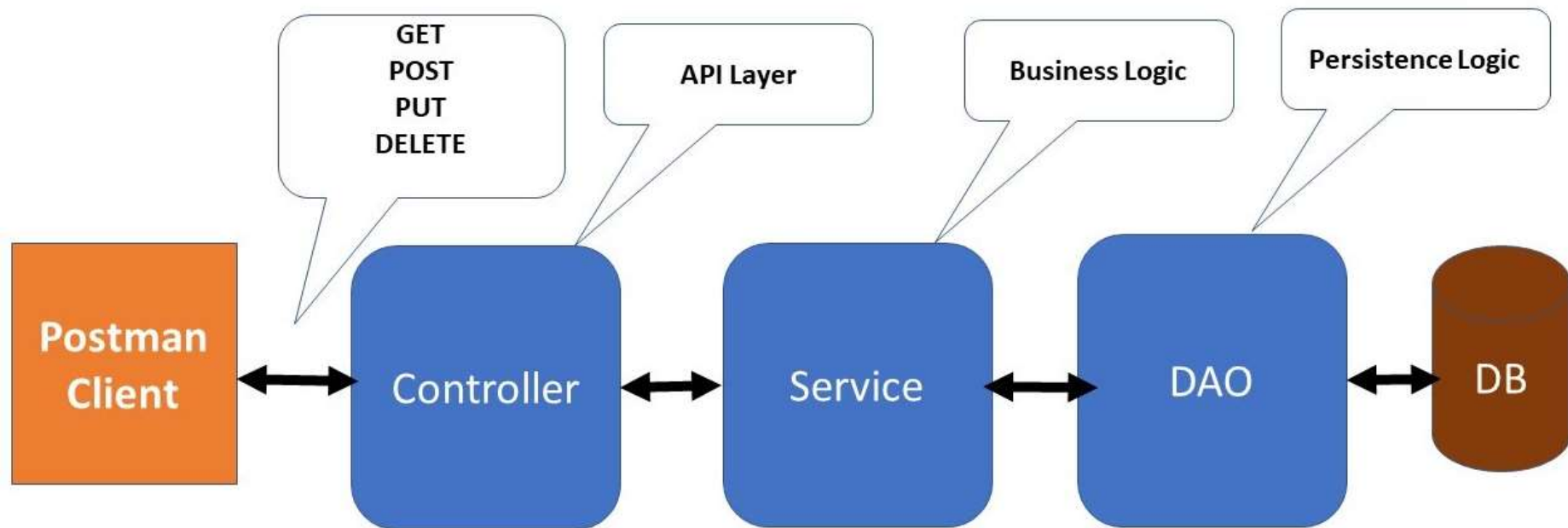
ER DIAGRAM



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	http://localhost:8080/api/doctors	View doctors
POST	http://localhost:8080/api/appointments/book	Book appointment
GET	http://localhost:8080/api/appointments/patient	View patient appointments
GET	http://localhost:8080/api/appointments/doctor/id	View doctor appointments
DELETE	http://localhost:8080/api/appointments/id	Cancel appointment

DATA DICTIONARY

TABLES OF DATABASE

MySQL Workbench

Local instance MySQL90 - W... x

File Edit View Query Database Server Tools Scripting Help

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

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

```
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
```

Limit to 1000 rows

Result Grid Filter Rows: Export: Wrap Cell Contents:

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
- 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* SQL File 11*

Limit to 1000 rows

```
1 use user_management;
2 SELECT * FROM patients;
3
```

Result Grid

	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	satish@gmail.com	Satish	sat2987	9578643201
5	5	roshini@gmail.com	Roshini	roshini@987	9854702654
6	6	divya@gmail.com	Diviya	divya@987	9765207639
7	7	keerthi@gmail.com	Keerthi	keerthi@987	8974702463
8	8				

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

Administration Schemas Information

DOCTOR 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* SQL File 11*

Limit to 1000 rows

```
1 * use user_management;
2 * SELECT * FROM patients;
3 * SELECT * FROM doctors;
4
```

Result Grid

	id	email	name	phone	specialization	password
▶	1	raj@gmail.com	Dr. Raj	9876057432	Dermatologist	09876
	2	amit@gmail.com	Dr. Amit	8976543210	Cardiologist	12345
	3	devi@gmail.com	Dr. Devi	9745372014	Gynecologist	98765
	4	maran@gmail.com	Dr. Maran	9567731843	Physiotherapist	54321
	5	priya@gmail.com	Dr. Priya	9182736450	Dermatologist	76543
*	NULL	NULL	NULL	NULL	NULL	NULL

SQLAdditions

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

Administration Schemas

Information

APPOINTMENT DATABASE

MySQL Workbench

Local instance MySQL80 - W... x

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

Limit to 1000 rows

```
1 use user_management;
2 SELECT
3 SELECT FROM doctors;
4 select * from appointments;
5
```

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

Edits 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
	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
•	total	total	total	total	total	total	total

Administration Schemas

Information

Result Grid

Form Editor

Field Types

Query Status

Appointments for a Specific Doctor

The screenshot displays the MySQL Workbench interface. The left sidebar shows a 'SCHEMAS' list with various databases, including 'amazon', 'companydb', 'crud', 'hospitalmanagement', 'medibook_db', 'rec_company', 'saturday', 'sis', 'student', 'studentd', 'studentdata', 'studentdatabase', 'sys', 'test', 'test29', and 'tests'. The 'amazon' database is selected. The main editor window shows a SQL query in 'SQL File 11':

```
1 use user_management;
2 SELECT * FROM patients;
3 SELECT * FROM doctors;
4 select * from appointments;
5
6 SELECT *
7 FROM appointments
8 WHERE doctor_id = 4;
9
```

The query results are displayed in the 'Result Grid' at the bottom. The grid shows columns: id, appointment_date, appointment_time, doctor_id, patient_email, patient_phone, and patient_id. The results are as follows:

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	5

On the right side of the interface, a message states: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

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

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
```

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: Exports: Wrap Cell Contents: T

	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

Administration Schemas Information

Form Editor

Field Types

Query State

GET METHOD FOR VIEWING DOCTORS

The screenshot displays the Postman interface for a GET request to `http://localhost:8080/api/doctors`. The request is successful, returning a 200 OK status with a response time of 30 ms and a body size of 792 B.

Query Params

Key	Value	Description
Key	Value	Description

Body

JSON

```
[{"id": 1, "name": "Dr. Raj", "specialization": "Dermatologist", "email": "raj@gmail.com", "phone": "987654321", "password": "99N76"}, {"id": 2, "name": "Dr. Amit", "specialization": "Cardiologist", "email": "amit@gmail.com", "phone": "8976543218", "password": "12345"}, {"id": 3, "name": "Dr. Devi", "specialization": "Neurologist", "email": "devi@gmail.com", "phone": "765432189", "password": "56789"}]
```


POST METHOD FOR BOOKING APPOINTMENT

The screenshot displays the Postman interface for a REST client. The top bar shows the 'API Network' tab with a search bar and a 'Send' button. The main workspace is set to 'POST' for the endpoint `http://localhost:8080/api/appointments/book`. The 'Body' tab is selected, showing a raw JSON payload:

```
1 {  
2   "doctorId": 1,  
3   "appointmentDate": "2025-01-30",  
4   "appointmentTime": "10:10",  
5   "patientEmail": "akiran@gmail.com"  
6 }  
7
```

The bottom status bar indicates a successful response: **200 OK** with a response time of 38 ms and a body size of 168 B. The 'Body' tab is also selected in the bottom panel, showing the response content: `{ "true" }`.

GET METHOD FOR VIEWING PATIENT APPOINTMENTS

The screenshot shows the Postman interface with a GET request configured to retrieve patient appointments. The URL is `http://localhost:8080/api/appointments/patient?email=ram@gmail.com`. The Headers tab is active, showing 7 hidden headers. The Body tab is also active, displaying a JSON array of two appointment objects. The status bar indicates a 200 OK response with a 20 ms response time and 761 B of data.

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

Method: GET

Headers (7 hidden):

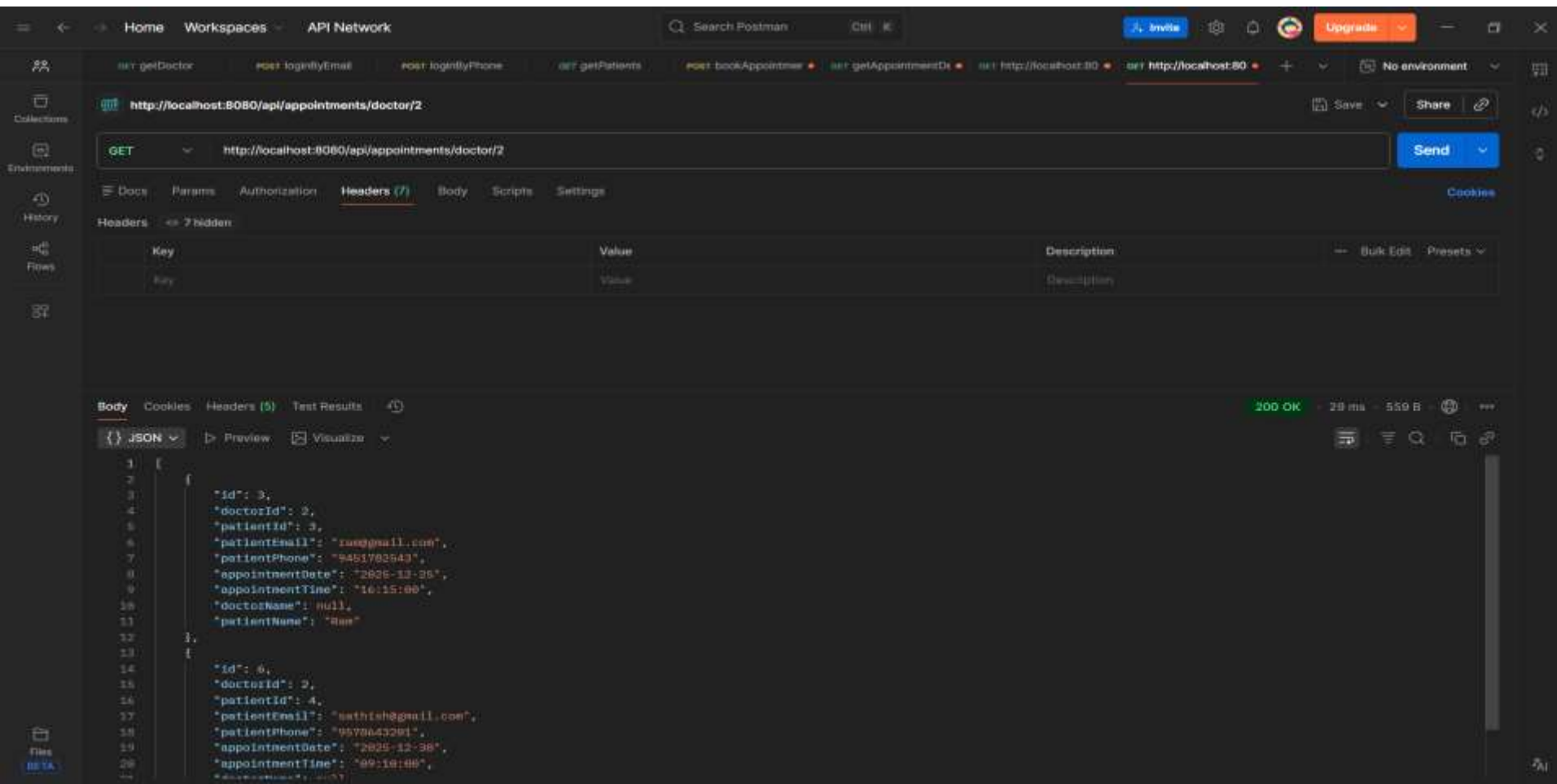
Key	Value	Description
Key	Value	Description

Body (JSON):

```
1 {
2   {
3     "id": 3,
4     "doctorId": 2,
5     "patientId": 3,
6     "patientEmail": "ram@gmail.com",
7     "patientPhone": "9451782543",
8     "appointmentDate": "2025-12-25",
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": "2025-12-26",
20    "appointmentTime": "10:30:00",
21    "doctorName": "Dr. Amit",
22    "patientName": null
23  }
24 }
```

Status: 200 OK - 20 ms - 761 B

GET METHOD FOR VIEWING DOCTOR APPOINTMENTS



The screenshot displays the Postman interface for testing an API endpoint. The request is a GET method to `http://localhost:8080/api/appointments/doctor/2`. The response is a 200 OK status with a 29 ms response time and 559 B of data. The response body is a JSON array containing two appointment objects.

Request:

- Method: GET
- URL: `http://localhost:8080/api/appointments/doctor/2`

Response:

```
200 OK - 29 ms - 559 B
```

```
{
  "id": 3,
  "doctorId": 2,
  "patientId": 3,
  "patientEmail": "raag@gmail.com",
  "patientPhone": "9451782543",
  "appointmentDate": "2025-12-26",
  "appointmentTime": "16:15:00",
  "doctorName": null,
  "patientName": "Rag"
},
{
  "id": 6,
  "doctorId": 2,
  "patientId": 4,
  "patientEmail": "sathish@gmail.com",
  "patientPhone": "9578643281",
  "appointmentDate": "2025-12-26",
  "appointmentTime": "09:10:00",
  "doctorName": null,
  "patientName": "Sathish"
}
```

DELETE METHOD FOR CANCELLING APPOINTMENT

The screenshot displays the Postman interface for a DELETE request. The URL bar shows `http://localhost:8080/api/appointments/4`. The method is set to **DELETE**. The response status is **200 OK** with a response time of 17 ms and a content length of 198 B. The response body, viewed in raw text, contains the message: `1. Appointment cancelled successfully`.

Headers (7)

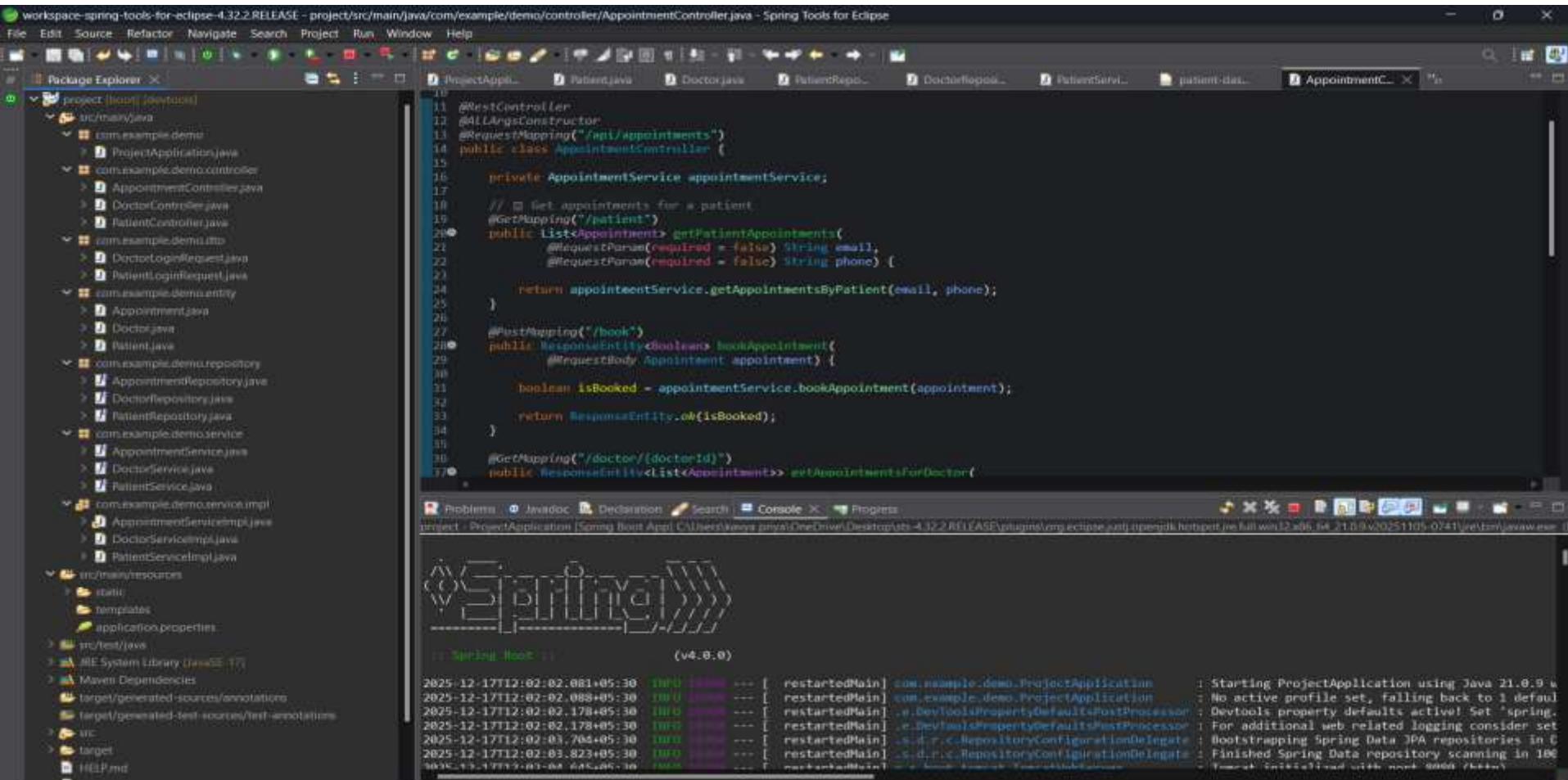
Key	Value	Description
Key	value	Description

Body

Raw

```
1. Appointment cancelled successfully
```

WORKING OF SPRING TOOL



WEBSITE DRIVE LINK

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