

# **APPOINTMED**

**A PROJECT REPORT**

*Submitted by*

**MOHAMED MAHDI**

**230282601068**

**Under the guidance of**

**Dr. A. K. ASHFAUK AHAMED**

*in partial fulfillment for the award of the degree of*

**MASTER OF COMPUTER APPLICATIONS**

**in**

**BRANCH OF STUDY**



**DEC 2024**



## **BONAFIDE CERTIFICATE**

Certified that this project report “**APPOINTMED**” is the bonafide work of **Mr. MOHAMED MAHDI (RRN: 230282601068)** who carried out the thesis work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

### ***SIGNATURE***

**Dr. A. K. ASHFAUK AHAMED**

**Supervisor**

**Assistant Professor(Sr. Gr.)**

Department of Computer Applications

B.S.Abdur Rahman Crescent Institute

of Science and Technology,

Vandalur, Chennai-600048

### ***SIGNATURE***

**Dr. M. Syed Masood**

**Head of the Department,**

Department of Computer Applications

B.S.Abdur Rahman Crescent Institute

of Science and Technology,

Vandalur, Chennai-600048



## VIVA-VOCE EXAMINATION

The viva-voce examination of the mini project work titled “**APPOINTMED**” submitted by **Mr. MOHAMED MAHDI, RRN: 230282601068**, is held on

---

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

I thank the **Almighty** for showering His blessings upon me in completing the project. I submit this project with a deep sense of gratitude and reverence for my beloved parents for their moral support and encouragements.

I sincerely express my heartfelt gratitude to **Dr. T. Murugesan, Vice Chancellor**, B.S. Abdur Rahman Crescent Institute of Science and Technology and **Dr. N. Raja Hussain, Registrar** for furnishing every essential facility for doing my project.

I owe my sincere gratitude to **Dr. Sharmila Sankar, Professor and Dean**, School of Computer, Information and Mathematical Sciences. and I also thank **Dr. M. Syed Masood, Associate Professor and Head**, Department of Computer Applications for providing strong oversight of vision, strategic direction, encouragement and valuable suggestions in completing my project work.

I convey my earnest thanks to my project guide **Dr. A.K. Ashfauk Ahamed, Assistant Professor (Sr. Gr.)**, Department of Computer Applications, for his valuable guidance and support throughout the project.

I extend my sincere thanks to all my faculty members for their valuable suggestions, timely advice and support to complete the project.

**-MOHAMED MAHDI**

## ABSTRACT

The **AppointMed Application** is an appointment management system tailored to streamline the scheduling and management of appointments between doctors and patients. With its user-friendly interface, the application simplifies appointment booking for both patients and healthcare providers, promoting efficiency and transparency in the healthcare setting. Users register and authenticate via role-based access, with patients gaining the ability to view available doctors by city and schedule appointments, while doctors can manage their appointment requests and maintain an organized schedule. The system offers a well-defined flow, allowing patients to select a doctor, choose an appointment date, view available timeslots, and confirm their booking. Doctors can view all appointments scheduled with them and have the option to cancel appointments if necessary, updating patients on any changes.

The **AppointMed** application operates on a robust three-tier architecture, that ensures efficiency and scalability. The **Client Tier** is developed in React.js, providing a responsive, dynamic interface for patient and doctor interactions, with real-time updates achieved through API calls to the server. In the **Application Tier**, built using Spring Boot, the **Controller Layer** handles incoming HTTP requests, routing them to the **Service Layer**, which contains the business logic for managing appointment details, timeslot availability, and status changes. . The **Repository Layer** enables smooth interactions with the database, allowing CRUD operations for appointments and user data , ensuring data integrity and consistency.

In the **Data Tier**, a MySQL database securely stores user profiles, appointment information, timeslot schedules, and logs, guaranteeing reliable

access and quick retrieval. This three-layered structure enhances system maintainability and scalability, making it adaptable to future improvements and new features. The **AppointMed** system reduces administrative workload, enhances communication between patients and doctors, and ensures an organized, transparent approach to appointment scheduling. Its modular architecture facilitates a clear separation of concerns, supporting easy maintenance and seamless updates for continued efficiency and functionality in the healthcare domain.

## TABLE OF CONTENT

S.NO.	DESCRIPTION	PAGE NO
	<b>ABSTRACT</b>	v
	<b>LIST OF FIGURES</b>	viii
	<b>LIST OF ABBREVIATIONS</b>	ix
<b>1.</b>	<b>INTRODUCTION</b>	1
	1.1 GENERAL	1
	1.2 OBJECTIVES	3
	1.3 CONTRIBUTION OF THE PROJECT	5
<b>2.</b>	<b>SYSTEM ANALYSIS</b>	8
	2.1 LITERATURE SURVEY	8
	2.2 EXISTING SYSTEM	10
	2.3 PROPOSED SYSTEM	11
<b>3.</b>	<b>SYSTEM SPECIFICATION</b>	12
	3.1 HARDWARE SPECIFICATION	12
	3.2 SOFTWARE SPECIFICATION	12
<b>4</b>	<b>SYSTEM DESIGN</b>	13
	4.1 SYSTEM ARCHITECTURE	13
	4.2 UML DIAGRAMS	16
	4.2.1 Usecase Diagram	18
	4.2.2 Sequence Diagram	19
<b>5</b>	<b>SYSTEM IMPLEMENTATION</b>	21
	5.1 METHODOLOGY	21
	5.2 IMPLEMENTATION DETAILS	40

<b>S.NO.</b>	<b>DESCRIPTION</b>	<b>PAGE NO</b>
	5.3 MODULES AND ITS DESCRIPTION	44
<b>6</b>	<b>SYSTEM TESTING</b>	46
	6.1 TYPES OF SYSTEM TESTING	47
	6.2 PHASES OF SYSTEM TESTING	50
	6.3 PERFORMANCE EVALUATION	53
<b>7</b>	<b>CONCLUSION AND FUTURE ENHANCEMENTS</b>	56
	7.1 CONCLUSION	56
	7.2 FUTURE ENHANCEMENTS	57
<b>8</b>	<b>RESULT AND ANALYSIS</b>	58
<b>9</b>	<b>REFERENCES</b>	62
	<b>APPENDIX</b>	63
	<b>TECHNICAL BIOGRAPHY</b>	85



## LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
Fig 4.1	ARCHITECTURE DIAGRAM	15
Fig 4.2.1	USECASE DIAGRAM	19
Fig 4.2.2	SEQUENCE DIAGRAM	20
Fig 8.1	HOME PAGE	59
Fig 8.2	UPLOAD PAGE	59
Fig 8.3	PREVIEW SCREEN	60
Fig 8.4	OUTPUT SCREEN	61

# CHAPTER 1

## INTRODUCTON

### 1.1 GENERAL

In today's fast-paced healthcare environment, efficiently managing appointments between doctors and patients is essential for improving communication and optimizing resources. The **AppointMed** system is developed to address the challenges associated with scheduling and managing appointments in healthcare settings. This system streamlines the process by providing a user-friendly interface for both doctors and patients, significantly enhancing the efficiency of appointment handling and contributing to a better patient experience.

At the core of this system lies a structured approach that allows patients to log in with their credentials, view available doctors by city, and submit appointment requests. The system facilitates clear communication by notifying patients of the status of their requests—whether confirmed or cancelled—thus minimizing confusion and improving the overall user experience. For doctors, **AppointMed** offers a straightforward interface to view and manage incoming appointment requests, helping them organize their schedules effectively and be prepared for each appointment.

To ensure optimal performance, the **AppointMed** system is built using a robust three-tier architecture. The **Client Tier**, developed with React.js, offers a dynamic and responsive user interface, enabling smooth interactions with the server via API calls. The **Application Tier** is built with Spring Boot, featuring a **Controller Layer** for handling HTTP requests, a **Service Layer** for managing business logic, and a **Repository Layer** for executing CRUD (Create, Read, Update, Delete) operations. The **Data Tier** utilizes a MySQL database to securely

store essential data, including user credentials, appointment details, and system logs. This architecture enhances communication efficiency, reduces manual intervention, and provides a solid foundation for future enhancements, ensuring scalability, maintainability, and operational excellence in healthcare appointment management.

## **1.2 OBJECTIVES**

The primary objective of the **\*\*AppointMed\*\*** system is to develop an efficient and user-friendly healthcare appointment management platform that simplifies the scheduling process for both patients and healthcare providers. By providing patients with a seamless interface to search for doctors based on their location and availability, the system aims to enhance accessibility and convenience in booking medical consultations. AppointMed strives to improve communication between patients and doctors by offering real-time notifications and updates on the status of appointment requests, ensuring transparency and reducing missed or mismanaged appointments.

A critical objective of the project is to optimize the management of healthcare resources by enabling doctors and administrative staff to efficiently organize their schedules, prevent appointment conflicts, and maximize clinic productivity. The system is designed with a focus on delivering a dynamic and responsive user experience through a React.js-powered client interface that interacts with the backend via API calls. Additionally, AppointMed aims to ensure data security and privacy by implementing robust authentication and data storage mechanisms, thereby safeguarding sensitive patient and doctor information in compliance with healthcare regulations.

Another important objective is to build a scalable and maintainable architecture using a three-tier design comprising the Client Tier, Application Tier,

and Data Tier. The Application Tier, developed with Spring Boot, is structured with a Controller Layer to handle HTTP requests, a Service Layer to manage business logic, and a Repository Layer for database operations. This architecture supports future enhancements, integration with other healthcare systems, and the ability to handle a growing user base, making it a versatile and sustainable solution for healthcare appointment management.

### 1.3 CONTRIBUTION OF THE PROJECT

The **AppointMed** system makes significant contributions to the healthcare industry by addressing the challenges associated with appointment management and improving the overall efficiency of healthcare services. One of its key contributions is the creation of a **centralized, accessible platform** for patients to book appointments with healthcare providers at their convenience, eliminating the need for traditional scheduling methods like phone calls or in-person visits. This enhanced accessibility is especially beneficial for patients in remote or underserved areas, enabling them to connect with medical professionals quickly and efficiently.

The system also contributes to **enhanced operational efficiency** by automating the appointment scheduling process, thereby minimizing the administrative workload on clinic staff. This allows healthcare providers to focus more on delivering quality care rather than managing appointment logistics. Additionally, AppointMed reduces the incidence of missed appointments by sending **real-time notifications and reminders** to patients, ensuring better utilization of available appointment slots and increasing overall clinic productivity.

Another significant contribution of the project is its ability to provide **data-driven insights** for healthcare administrators. The system logs detailed information about appointments, user interactions, and system performance, which can be analyzed to optimize resource allocation, manage patient loads, and improve service delivery. These insights empower healthcare providers to make informed decisions that enhance both operational efficiency and the patient experience.

AppointMed's **modular and scalable architecture** allows for seamless integration with other healthcare management systems, such as electronic health records (EHR), telemedicine platforms, and billing systems. This capability fosters a more interconnected healthcare ecosystem, improving data flow and coordination across various healthcare services and creating a comprehensive solution for healthcare management.

In addition to its functional benefits, the system **empowers healthcare professionals** by providing advanced tools for efficient schedule management. Doctors can view and manage appointment requests in real-time, helping them prepare adequately for each consultation and allocate sufficient time to each patient, thereby improving the quality of care delivered.

Moreover, AppointMed demonstrates the effectiveness of a **three-tier architecture** in building scalable, secure, and maintainable applications. This architecture supports the current functionality while providing a solid foundation for future expansions, such as incorporating AI-driven appointment recommendations, patient feedback systems, and integration with wearable health devices.

Ultimately, the AppointMed system enhances the **patient experience** by offering a transparent, efficient, and user-friendly appointment process, contributing to higher patient satisfaction and retention. It positions itself as a valuable tool for healthcare providers seeking to optimize their operations, deliver better care, and make a meaningful impact on the overall healthcare delivery system.

**The AppointMed system contributes to healthcare management in the following ways:**

**1. ImprovedAppointmentAccessibility:**

By providing a centralized platform for scheduling, AppointMed ensures that patients can book appointments anytime, anywhere, reducing the need for phone calls or in-person visits.

**2. EnhancedOperationalEfficiency:**

Automating the appointment scheduling process minimizes manual intervention by clinic staff, allowing healthcare providers to focus more on patient care.

**3. ReductionInNo-Shows:**

By sending real-time notifications and appointment reminders to patients, the system reduces missed appointments, improving clinic productivity and patient service.

**4. Data-DrivenInsights:**

The system logs appointment and user data, providing healthcare administrators with valuable insights for optimizing resource allocation, managing patient loads, and improving service delivery.

#### 5. **SeamlessIntegrationwithHealthcareSystems:**

The modular and scalable architecture of AppointMed enables easy integration with other healthcare management systems, such as EHR and telemedicine platforms, fostering a comprehensive healthcare solution.

#### 6. **ContributiontoPatientSatisfaction:**

By offering a transparent, efficient, and user-friendly appointment process, AppointMed enhances the patient experience, contributing to higher satisfaction levels and better patient retention for healthcare providers.

#### 7. **SupportforHealthcareProfessionals:**

The system empowers doctors with better schedule management tools, helping them prepare for appointments and allocate adequate time for each patient, thereby improving the quality of care provided.

## **CHAPTER 2**

### **SYSTEM ANALYSIS**

#### **2.1 LITERATURE SURVEY**

##### **Study on Appointment Scheduling Systems**

Efficient appointment management systems are crucial for improving healthcare services. **Shah, P., and Mehta, K. (2022)** explored the impact of online appointment booking systems in healthcare, highlighting how they reduce administrative workload and improve patient satisfaction. Their study demonstrates that automated systems not only minimize manual scheduling but also provide patients with greater flexibility in booking appointments. This

shift significantly enhances operational efficiency and reduces wait times in healthcare facilities.

#### Integration of Electronic Health Records (EHR)

The integration of appointment systems with Electronic Health Records (EHR) has shown significant advantages in healthcare management. **Patel, R., and Gupta, M. (2021)** analyzed the benefits of such integration, emphasizing improved data flow and coordination between departments. Their research highlights that EHR-integrated systems streamline the sharing of patient information, ensuring that healthcare providers are better informed and can deliver more personalized care.

#### Automated Reminders and Reduction in No-Shows

**Kim, J., Lee, S., and Park, H. (2021)** focused on the role of automated notifications in reducing appointment no-shows. Their study found that sending timely reminders via SMS or email significantly decreases the rate of missed appointments, leading to better utilization of available time slots and increased clinic productivity. The research suggests that incorporating automated reminders can enhance patient compliance and satisfaction.

#### Data Analytics for Resource Optimization

Data-driven insights play a critical role in optimizing healthcare operations. **Wang, Y., and Zhang, T. (2020)** investigated how appointment systems collect and analyze data to manage patient load and allocate resources efficiently. Their findings indicate that predictive analytics can help healthcare administrators anticipate demand patterns, optimize staff schedules, and reduce operational bottlenecks, ultimately improving the quality of care.

#### Modular and Scalable Design in Healthcare IT

The scalability and modularity of appointment management systems are essential for their adaptability to evolving healthcare needs. **Smith, J., and Brown, L. (2020)** discussed the importance of designing systems that can integrate with telemedicine platforms, billing systems, and other healthcare services. Their research illustrates that a modular architecture not only supports current functionalities but also facilitates future expansions, such as incorporating AI-based appointment recommendations and patient feedback mechanisms.



## Real-Time Appointment Management

Real-time appointment management systems provide immediate updates and improve the overall patient experience. **Cheng, H., and Li, Q. (2019)** developed a real-time scheduling algorithm using OpenCV and Python. Their system dynamically updates appointment availability based on cancellations and rescheduling, ensuring optimal slot utilization. The research highlights that real-time systems enhance patient engagement and reduce administrative delays in appointment management.

## Impact on Patient Satisfaction and Retention

Patient satisfaction and retention are critical metrics for healthcare providers. **Johnson, K., and Williams, M. (2019)** examined how transparent and user-friendly appointment processes contribute to higher patient satisfaction. Their study concludes that systems offering easy access, real-time updates, and personalized scheduling options significantly improve patient retention and foster long-term loyalty to healthcare providers.

These studies collectively emphasize the importance of automated, integrated, and data-driven appointment management systems like **AppointMed** in enhancing healthcare delivery, improving operational efficiency, and increasing patient satisfaction.

## 2.2 EXISTING SYSTEM

The existing systems for real-time video manipulation primarily focus on either static image processing or limited video effects that often lack interactivity. Traditional software solutions typically rely on post-processing techniques, where effects are applied after recording, making them unsuitable for live interactions.

These systems often employ basic filtering or overlay features, failing to integrate advanced technologies like deep learning, which limits their adaptability to dynamic environments. Platforms such as Snapchat, Instagram, and Zoom have incorporated basic AR features and face filters. These systems often rely on facial landmark detection and lightweight image processing techniques to apply real-time effects.

However, most existing systems are limited in terms of the complexity and realism of their effects. They are typically optimized for mobile platforms but often lack the seamless quality required for more professional video applications. While there are powerful software solutions such as OBS Studio (Open Broadcaster Software) and tools powered by OpenCV.

these tend to focus more on video streaming and basic overlays rather than advanced AI-based transformations. Current systems also face challenges in achieving high-quality, real-time face swaps or video manipulations without latency or performance issues, especially on lower-end hardware.

The system also faces challenges such as poor performance in low-light settings, lack of real-time interaction capabilities, and minimal support for automation. These limitations hinder the user experience, making it inefficient for scenarios requiring quick adjustments, such as live events, virtual meetings, or streaming. Addressing these gaps in the current system highlights the need for a robust and integrated solution like which aims to bring advanced, AI-driven, and user-friendly functionalities into a single, seamless platform.

## **2.3 PROPOSED SYSTEM**

The AppointMed system is designed to overcome the limitations of existing healthcare appointment solutions by providing a comprehensive, automated platform for managing appointments in clinics and hospitals. Key features of the proposed system include:

- **User-Friendly Interface:** A responsive, modern interface that enables patients and doctors to log in, register, and manage appointments seamlessly, improving accessibility and ease of use.
- **Online Appointment Booking:** Patients can submit appointment requests using an intuitive online form, selecting preferred dates and times, which minimizes errors and miscommunication during scheduling.

- **Real-Time Notifications:** Patients receive automatic notifications regarding the status of their appointment requests, ensuring transparency and timely updates throughout the booking process.

- **Centralized Dashboard for Doctors:** Doctors can view and manage all incoming appointments from a single dashboard, allowing for streamlined appointment handling and organized scheduling.

- **Flexible Appointment Management:** Patients have the ability to view appointment history and, if necessary, cancel upcoming appointments, which enhances control and convenience.

- **Scalable Three-Tier Architecture:** AppointMed is built on a three-tier architecture, ensuring scalability and maintainability to support future system enhancements and feature integration.

Overall, AppointMed is designed to simplify the appointment booking process, improve communication, and enhance user satisfaction for both patients and doctors.

## **CHAPTER 3**

### **SYSTEM SPECIFICATION**

#### **3.1 HARDWARE SPECIFICATION**

- **Processor** : Intel Xeon E3-1240 v6 or equivalent.
- **Speed** : 3.7 GHz base clock.
- **RAM** : 16 GB.
- **Hard Disk** : Minimum 256 GB SSD

### **3.2 SOFTWARE SPECIFICATION**

- Operating System : Ubuntu 20.04 LTS or higher o Windows Server 2019 (optional).
- Programming Languages :
  - Frontend: HTML5, CSS3, JavaScript (ES6+)
  - Backend: Java 11 or higher
- Framework :
  - Frontend: React.js
  - Backend: Spring Boot
- IDE/Code Editors :Eclipse for Java; Visual Studio Code for React.js
- Database :
  - Database Management System: MySQL 8.0 or higher
  - Database Connection: JDBC for Java

## **CHAPTER 4**

### **SYSTEM DESIGN**

#### **4.1 SYSTEM ARCHIRECTURE**

The Doctor Appointment Booking System is designed to facilitate effective scheduling and management of appointments between doctors and patients.

Key features include:

- User Registration and Role-Based Profiles: AppointMed allows both patients and doctors to register and create user profiles. Patients and doctors have distinct access rights, with user roles stored securely in the database, ensuring each user has a tailored experience based on their role.

- **City-Based Doctor Search:** Patients can filter doctors by city, simplifying the process of finding local healthcare providers. This feature enhances user convenience, allowing patients to view only the doctors available in their preferred location.

- **Efficient Appointment Booking:** Patients can view the available timeslots for a specific doctor on their selected date and book an appointment directly from the system. This streamlines the booking process, enabling patients to find suitable time slots quickly and easily.

- **Personalized Appointment Dashboard for Patients:** Patients have a dedicated section where they can view and manage all their scheduled appointments. This dashboard provides a clear and organized view of upcoming and past appointments, enhancing patient engagement and accountability.

- **Appointment Management Dashboard for Doctors:** Doctors can view a comprehensive list of all appointments made by patients. They also have the ability to cancel an appointment if necessary, providing greater control over their schedule.

- **Secure Role-Based Access Control with JWT Authentication:** AppointMed uses JSON Web Tokens (JWT) to enforce security and restrict access to certain API endpoints based on user roles. Patients and doctors are only permitted access to functionalities relevant to their specific roles, ensuring data security and user privacy.

- **Scalable and Maintainable Architecture:** Built using Spring Boot and React, AppointMed follows a modern, three-tier architecture that enhances responsiveness and maintainability. This layered structure allows for scalability, making it adaptable for future updates and increased user demand.

- **Robust Data Management with MySQL Database:** The system's backend uses MySQL to store essential information, including user profiles, appointment details, and available timeslots. This ensures that data is securely stored and accessible for quick retrieval, enhancing the system's reliability and efficiency.

- **User-Friendly Design:** Responsive design that works well on various devices (desktops, tablets, and smartphones). Easy navigation and clear layout to enhance user experience.

- **Responsive and Mobile-Friendly Interface:** AppointMed's frontend, built with React, is designed to be responsive, ensuring that users can access and use the application seamlessly from any device, including tablets and smartphones, making healthcare more accessible.

- **Data Encryption for Sensitive Information:** To enhance security, AppointMed implements encryption techniques to protect sensitive patient and doctor information, ensuring that personal data, medical information, and login credentials are securely stored and managed.

- **Comprehensive Appointment History Tracking:** Patients and doctors can both view a full history of past appointments. This feature allows patients to track their healthcare interactions, while doctors can review previous consultations with patients for continuity in care.

These features collectively make AppointMed a comprehensive, user-friendly application designed to streamline the appointment scheduling process and foster better communication between patients and doctors.

## 4.2 DATA FLOW DIAGRAM (DFD)

A DFD visually represents the flow of data within the system, including interactions between users and the system a data flow diagram (DFD). It illustrates how data moves through the system, showing interactions between various components such as patients, doctors, and the database. Here's an overview of the DFD at two levels:

### Context Level DFD (Level 0)

The Level 0 DFD, or context diagram, provides a high-level view of the AppointMed system. It shows the system's primary entities and how they interact with the system without going into detailed processes.

#### Entities:

- Patient - Registers, logs in, views available doctors and timeslots, requests appointments, and views appointment history.
- Doctor - Registers, logs in, views received appointments, and cancels appointments if necessary.
- Database - Stores all data related to users, appointments, and timeslots.

#### Process:

AppointMed System - Acts as the central process, interfacing between patients, doctors, and the database to manage appointment bookings.

#### Data Flows:

- ❖ Patient → AppointMed System: Registration/login requests, appointment requests, view history requests.

- ❖ Doctor → AppointMed System: Registration/login requests, view appointments, cancel appointment requests.
- ❖ AppointMed System → Database: Save/retrieve user data, timeslots, and appointment details.
- ❖ Patient Registration and Login → Patient submits registration details or login credentials → System
- ❖ verifies credentials → System responds with login success or error message.
- ❖ Doctor Registration and Login → Doctor submits registration details or login credentials → System
- ❖ verifies credentials → System responds with login success or error message
- ❖ Appointment Request Process → Patient selects city and doctor → System displays available timeslots →
- ❖ Patient submits an appointment request with selected time → System updates Appointment Database.
- ❖ Appointment Confirmation / Rejection → Doctor views incoming requests → Doctor approves or rejects an appointment → System updates Appointment Database → System sends status notification to the patient.
- ❖ Appointment Viewing and Management → Patient views their upcoming appointments, or doctor views
- ❖ all appointments assigned to them → System retrieves data from Appointment Database and displays it
- ❖ Appointment Cancellation → Doctor cancels an appointment if needed → System updates Appointment
- ❖ Database → System sends cancellation notification to the patient.



## Level 1 DFD

The Level 1 DFD breaks down the AppointMed System process into more specific actions, showing the primary operations involved in managing appointments.

Expanding further, we can break down each major process:

### 1. User Authentication

Patient or Doctor enters login credentials → AppointMed's Authentication System validates credentials → System grants access to the appropriate user interface (Patient or Doctor dashboard).

### 2. Submit Appointment Request

Patient selects a doctor, chooses a date and timeslot → System records the appointment request and associates it with the selected doctor → Notifies the doctor of the new appointment request.

### 3. Manage Appointment Requests

Doctor views pending appointments in their dashboard → Doctor can choose to cancel an appointment if necessary → System updates the appointment status accordingly.

### 4. Notify Patient

System sends a notification to the Patient (via email or system notification) updating them on the status of their appointment, especially if there is a cancellation.

## 5. View Scheduled Appointments

Both Patient and Doctor can view all scheduled appointments in their dashboards → System retrieves and displays appointment details from the database based on user role

- **Appointment Management Dashboard for Doctors:** Doctors can view a comprehensive list of all appointments made by patients. They also have the ability to cancel an appointment if necessary, providing greater control over their schedule.
- **Secure Role-Based Access Control with JWT Authentication:** AppointMed uses JSON Web Tokens (JWT) to enforce security and restrict access to certain API endpoints based on user roles. Patients and doctors are only permitted access to functionalities relevant to their specific roles, ensuring data security and user privacy.
- **Scalable and Maintainable Architecture:** Built using Spring Boot and React, AppointMed follows a modern, three-tier architecture that enhances responsiveness and maintainability. This layered structure allows for scalability, making it adaptable for future updates and increased user demand.

- **Robust Data Management with MySQL Database:** The system's backend uses MySQL to store essential information, including user profiles, appointment details, and available timeslots. This ensures that data is securely stored and accessible for quick retrieval, enhancing the system's reliability and efficiency.
- **User-Friendly Design:** Responsive design that works well on various devices (desktops, tablets, and smartphones). Easy navigation and clear layout to enhance user experience.
- **Responsive and Mobile-Friendly Interface:** AppointMed's frontend, built with React, is designed to be responsive, ensuring that users can access and use the application seamlessly from any device, including tablets and smartphones, making healthcare more accessible.
- **Data Encryption for Sensitive Information:** To enhance security, AppointMed implements encryption techniques to protect sensitive patient and doctor information, ensuring that personal data, medical information, and login credentials are securely stored and managed.
- **Comprehensive Appointment History Tracking:** Patients and doctors can both view a full history of past appointments. This feature allows patients to track their healthcare interactions, while doctors can review previous consultations with patients for continuity in care.

These features collectively make AppointMed a comprehensive, user-friendly application designed to streamline the appointment scheduling process and foster better communication between patients and doctors.

## DATA FLOW DIAGRAM

The DFD provides a structured view of the main functionalities and data interactions within AppointMed, ensuring that the data flow is efficient and easily accessible by both patients and doctors. It shows the flow of data within the system for scheduling and managing doctor appointments.

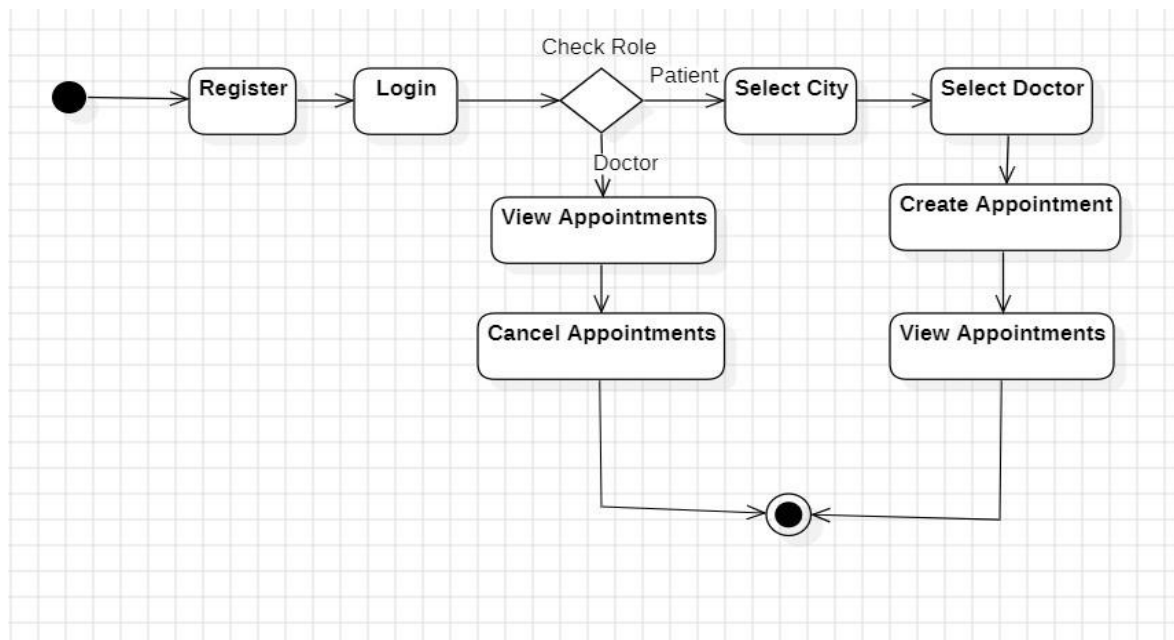
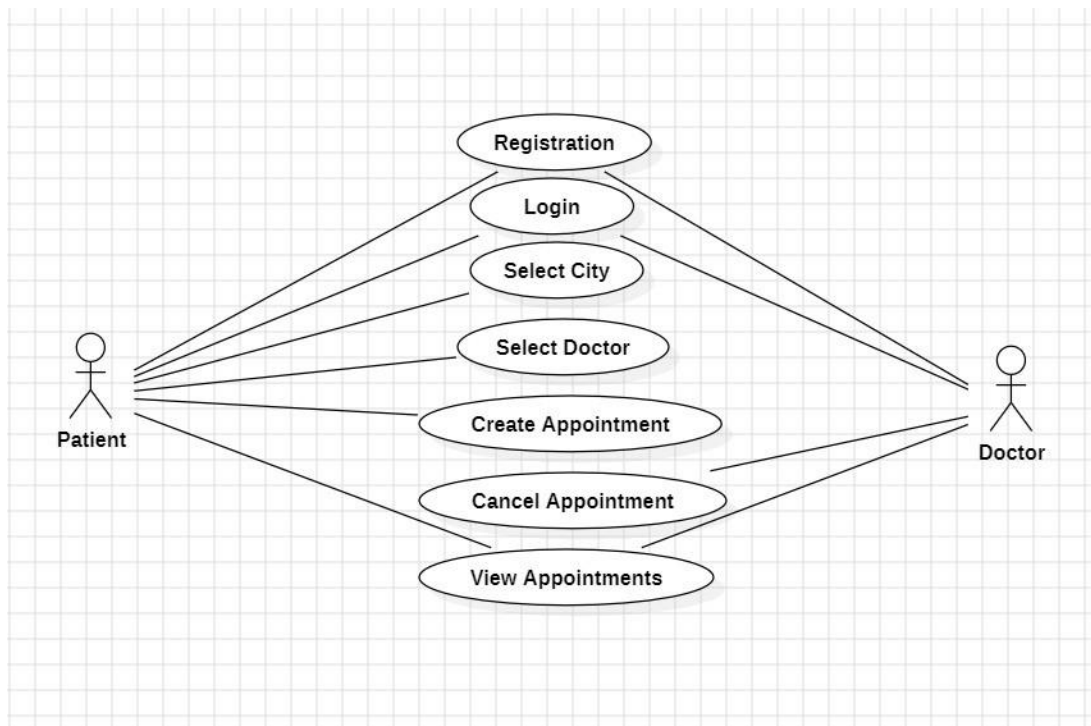


Fig. 1. Data flow diagram

## **4.3 USE CASE DIAGRAM**

A use case diagram visually represents the interactions between users (actors) and the system, highlighting the different functionalities (use cases) that the system provides. The Use Case Diagram for AppointMed visually represents the interactions between the system's primary actors (patients and doctors) and the various functions they can perform.

Fig. 2. Use Case Diagram



#### 4.4 DATABASE DESIGN

- For the AppointMed application, the database design consists of an Entity-Relationship Diagram (ERD) that defines the core entities, their attributes, and relationships, along with the MySQL database table structure for each entity.

##### ▪ Entities:

- User: Stores details for both patients and doctors, including registration information and user roles.
- Patient: Stores details specific to patients, such as personal information, contact details, and appointment history.
- Doctor: Contains details for doctors, including specialization, availability, and associated appointments.
- Appointment: Tracks appointment requests and statuses, including details of the selected date, time, and assigned patient-doctor pair.

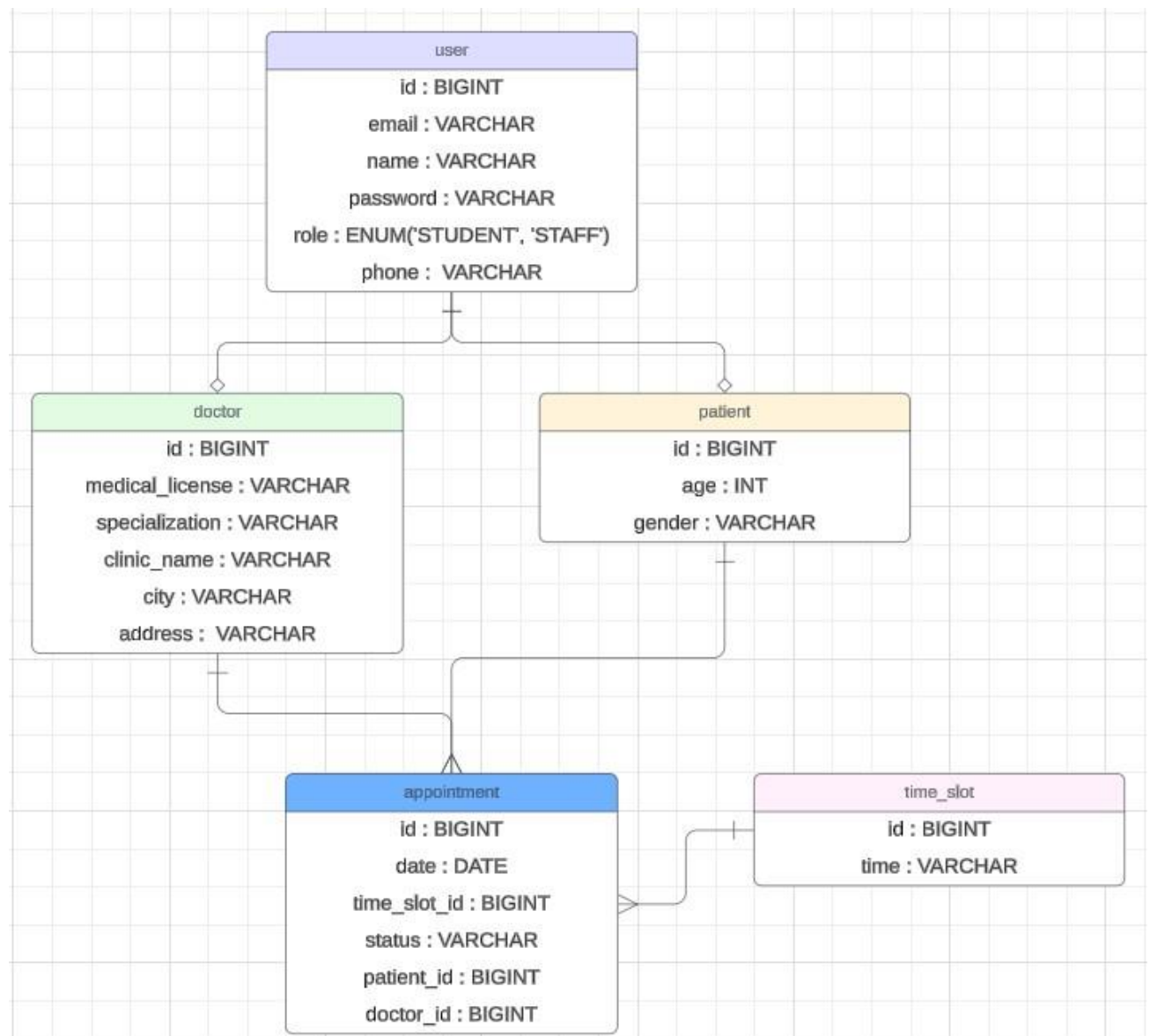
- Timeslot: Manages available time slots associated with each doctor, allowing patients to select specific dates and times for appointments.

### Relationships:

- Patients can book appointments with Doctors by selecting a Timeslot.
- Appointments link Patients and Doctors and hold details of the scheduled meeting.

This structure supports a scalable and efficient AppointMed system, providing seamless management of doctor-patient appointments and timeslots while ensuring data transparency and maintainability.

### DATABASE DESIGN



## **CHAPTER 5**

### **SYSTEM IMPLEMENTATION**

#### **5.1 METHODOLOGY**

##### **IMPLEMENTATION DETAILS**

- **Feasibility Study**

A feasibility study evaluates the practicality of the proposed system in terms of operational, technical, and economic aspects. The results of the feasibility study are as follows:

- **Operational Feasibility:**

The Proposed system addresses the operational needs of healthcare practices by automating appointment booking, enhancing communication, and streamlining management tasks for both patients and doctors. This automation improves the efficiency of healthcare providers and enhances the patient experience. User training and support materials can be easily implemented to ensure all users quickly adopt and become familiar with the system.

- **Technical Feasibility:**

AppointMed is built with established technologies such as React.js for the frontend and Spring Boot for the backend, aligning with modern industry standards. The system's three-tier architecture supports scalability, enabling easy integration of new features and simplifying maintenance. The software and hardware requirements are readily achievable with most healthcare providers' current resources.

- **Economic Feasibility:**

Initial setup costs for developing and deploying the system will be incurred, but AppointMed provides significant long-term benefits, including reduced administrative workload, improved scheduling efficiency, and better patient satisfaction. Increased efficiency and minimized manual intervention can lead to cost savings over time, making the investment in AppointMed economically viable and advantageous for healthcare practice.

## **5.2 MODULES AND ITS DESCRIPTION**

The system is designed with a three-tier architecture. Each module has specific responsibilities for handling users, appointments, notifications, and system logs. The modules are based on the architecture's client, application, and data tiers.

### **1. User Module**

This module handles user authentication and management for both patients and doctors, allowing for streamlined access and profile handling within the application.

- **Responsibilities:**

- Authentication and authorization for patients and doctors with role-based access.
- Management of patient and doctor profiles, including fetching and updating user information.

- **Key Components:**

- PatientController.java and DoctorController.java: Handle HTTP requests related to user actions for patients and doctors, respectively.



- UserService.java: Implements business logic for user authentication and profile management.
- UserRepository.java: Manages database interactions for user-related operations.

## 2.Appointment Module

This module manages the scheduling, updating, and tracking of appointments between patients and doctors.

- **Responsibilities:**
  - Patients can request appointments with doctors by selecting a time slot and date.
  - Doctors have the ability to view and manage incoming appointment requests, with the option to cancel appointments if necessary.
  - Both patients and doctors can view appointment histories and upcoming bookings.
- **Key Components:**
  - AppointmentController.java: Exposes endpoints to create, update, and retrieve appointments.
  - AppointmentService.java: Implements business logic for appointment creation, availability checking, and status updates.
  - AppointmentRepository.java: Manages appointment data interactions with the database.

## 3.Time Module

The Time Module manages the available timeslots for appointments, ensuring that both doctors and patients can schedule appointments based on predefined slots. This module supports efficient time management and availability display for booking.

- **Responsibilities:**
  - Initialization of default timeslots available for appointments, ensuring that times are pre-set for patient selection.
  - Management of time-based operations, including querying available slots and updating availability when appointments are created or canceled.
- **Key Components:**
  - TimeSlotInitializer.java: Initializes a set of default timeslots in the system, ensuring that each doctor has predefined slots available for patient bookings.
  - TimeService.java: Handles business logic related to time management, such as checking slot availability, marking slots as booked, and releasing slots if appointments are cancelled.
  - TimeRepository.java: Interacts with the database to store, retrieve, and update timeslot data, ensuring accurate availability records.

## USER TABLE

Column Name	Data Type	Description
id	BIGINT	Primary Key, auto-generated user ID
name	VARCHAR(255)	User's name
email	VARCHAR(255)	User's email address (used as username)
password	VARCHAR(255)	User's hashed password
role	VARCHAR(255)	Enum type role indicating user type (e.g., Patient or Doctor)
phone	VARCHAR(20)	User's phone number

PATIENT TABLE

Column Name	Data Type	Constraints	Description
id	BIGINT	PRIMARY KEY, FOREIGN KEY (User.id)	Inherits User ID, references <code>User.id</code>
age	INT	NOT NULL	Patient's age
gender	VARCHAR(20)	NOT NULL	Patient's gender (can be ENUM)

DOCTOR TABLE

Column Name	Data Type	Constraints	Description
id	BIGINT	PRIMARY KEY, FOREIGN KEY (User.id)	Inherits User ID, references <code>User.id</code>
medical_license	VARCHAR(255)	NOT NULL	Doctor's medical license number
specialization	VARCHAR(255)		Doctor's area of specialization
clinic_name	VARCHAR(255)		Name of the doctor's clinic
address	VARCHAR(255)		Clinic address
city	VARCHAR(100)		Clinic city

Appointment Table

Column Name	Data Type	Constraints	Description
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each appointment
patient_id	BIGINT	NOT NULL, FOREIGN KEY (Patient.id)	References the patient in the <code>Patient</code> table
doctor_id	BIGINT	NOT NULL, FOREIGN KEY (Doctor.id)	References the doctor in the <code>Doctor</code> table
time_slot_id	BIGINT	NOT NULL, FOREIGN KEY (TimeSlot.id)	References the time slot in <code>TimeSlot</code> table
date	DATE	NOT NULL	Date of the appointment
status	VARCHAR(50)	NOT NULL	Appointment status (e.g., "Scheduled", "Completed")

## Time Slot Table

Column Name	Data Type	Constraints	Description
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each time slot
time	VARCHAR(50)	NOT NULL	Represents the time of the time slot

## CHAPTER 6

### SYSTEM TESTING

System testing is a critical phase in the software development lifecycle, ensuring that a complete, integrated software system meets specified requirements. It focuses on verifying the interactions and functionalities of individual modules within the entire system. By performing system testing, developers can identify discrepancies, defects, and issues in performance or security before deployment, making it an essential step toward delivering a reliable product.

One of the main objectives of system testing is to verify that all components work together as expected. It simulates real-life usage scenarios to validate that each module within the software interacts as it should, without any unexpected behavior or performance issues. This is especially crucial for complex applications with multiple interconnected modules, as it ensures the system delivers a smooth and consistent user experience.

In system testing, both functional and non-functional aspects of the system are assessed. Functional system testing focuses on evaluating if the software performs its intended functions accurately. It involves testing core features, input and output processing, user interactions, and data flow. For instance, in a web application, functional system testing may validate features like login, navigation, data input, and processing. Non-functional testing, on the other hand, assesses attributes like performance, security, and usability. Performance testing within system testing measures the system's response time, scalability, and reliability under various loads. By stressing the system, developers can identify bottlenecks and optimize the application for better performance, ensuring it handles expected workloads efficiently.

## Benefits of system testing:

- **Software performance:** Performance-based system tests can help understand changes in a system's performance and behavior, such as memory consumption, central processing unit utilization and latency. These tests raise red flags if system performance degrades significantly, enabling developers to take proactive action.
- **Ensures Complete System Functionality:** System testing checks the software as a whole, verifying that each module functions together as expected. It helps ensure that individual components, like face detection or swapping modules in a project, work seamlessly, leading to a cohesive user experience.
- **Identifies Integration Issues:** By testing the integrated system, it reveals compatibility and interaction problems between different modules, such as potential lag between the video feed and face-swapping functions.
- **Improves User Satisfaction:** System testing identifies issues that could impact end users, such as UI flaws, processing delays, or incomplete functionality. By detecting and fixing these issues, system testing contributes to a more reliable and user-friendly product, enhancing user satisfaction and trust in the application.
- **Reduces Maintenance Costs:** Thorough system testing reduces the chances of critical issues arising after deployment, lowering long-term maintenance costs. Identifying and resolving issues during the testing phase prevents costly fixes and rework once the software is live.
- **Security:** Well-tested products are reliable. They ensure that the tested system doesn't contain potential vulnerabilities that can put end users and system data at risk of potential threats.

## 6.1 TYPES OF SYSTEM TESTING

Various testing methodologies can be applied to verify that the system is working as expected. These methodologies focus on different aspects of the system, such as functionality, performance, security, and usability.

## 1. Unit Testing

- Objective: Ensure that individual components or units of the system (such as services, controllers, or repositories) work correctly in isolation.
- Tools: JUnit (for Java), Mockito (for mocking dependencies), Spring Test (for testing Spring Boot applications).
- Example: Test the create Appointment method in the Appointment Service to ensure appointments are correctly created.

## 2. Integration Testing

- Objective: Validate that multiple components work together as expected when integrated.
- Tools: Spring Boot Integration Tests, Postman (for API testing), MySQL test database.
- Example: Test the interaction between Appointment Controller, Appointment Service, and the Appointment Repository to verify the flow of creating an appointment through the API.

## 3. Functional Testing

- Objective: Verify that the system performs its intended functions according to the requirements. It includes testing user interactions with the system (e.g., logging in, booking appointments).
- Tools: Selenium (for web UI testing), Postman (for API testing), manual testing (for user experience verification).
- Example: Test the user flow where a patient creates an appointment and a doctor accepts or rejects the request.

#### 4. Acceptance Testing

- Objective: Validate the system against the business requirements and ensure it meets the acceptance criteria defined by the stakeholders.
- Tools: Manual testing or automation tools like Cucumber (for behavior-driven development).
- Example: Test the entire workflow from user registration to managing appointments, including login, request submission, and notification handling.

#### 5. Performance Testing

- Objective: Assess the system's performance under load, ensuring it can handle multiple simultaneous users without degradation.
- Tools: JMeter, Gatling.
- Example: Simulate multiple users logging in and submitting appointment requests to assess the system's response time and stability under stress.

#### 6. Security Testing

- Objective: Ensure that the system is secure and protects sensitive user data, such as user credentials and appointment details.
- Tools: OWASP ZAP, Burp Suite (for vulnerability scanning), manual code reviews.
- Example: Test for SQL injection vulnerabilities in user authentication and ensure secure password hashing.

## 6.2 PHASES OF SYSTEM TESTING

### TEST CASES

Below are example test cases covering various functionalities of the Doctor Appointment Booking System. These test cases can be executed manually or using automation tools to verify that the system works as expected.

#### Test Case 1: User Login

- Test Case ID: TC001
- Description: Verify that users can log in with valid email credentials.
- Preconditions: The user must already be registered in the system.
- Test Steps:
  1. Navigate to the login page.
  2. Enter valid email and password.
  3. Click on the "Login" button.
- Expected Result: User is successfully logged in and redirected to the dashboard.
- Actual Result: User is successfully logged in and redirected to the dashboard.
- Status: Pass/Fail

#### Test Case 2: Create Appointment

- Test Case ID: TC002
- Description: Verify that an end user can create an appointment.
- Preconditions: User must be logged in.
- Test Steps:
  1. Select a city and doctor.
  2. Choose an available date and time slot.
  3. Click "Schedule."



- Expected Result: Appointment is successfully created and marked as "SCHEDULED" in the database.
- Actual Result: Appointment is successfully created and marked as "SCHEDULED" in the database.
- Status: Pass/Fail
- 

### **Test Case 3: View Appointment Status**

- Test Case ID: TC003
- Description: Verify that a patient can view the status of their appointments.
- Preconditions: Patient must have created an appointment.
- Test Steps:
  1. Navigate to the "View Requests" page.
  2. Check the status of the previously created appointments.
- Expected Result: The correct status (SCHEDULED or CANCELLED) is displayed for each appointment.
- Actual Result: SCHEDULED.
- Status: Pass/Fail

### **Test Case 4: Cancel Appointment (Doctor)**

- Test Case ID: TC004
- Description: Verify that a doctor can cancel an appointment.
- Preconditions: A patient must have created an appointment to the doctor.
- Test Steps:
  1. Doctor logs in and navigates to the "View Requests" page.
  2. View the list of scheduled appointments.
  3. Select an appointment and cancels it.
- Expected Result: Appointment status is updated in the database, and the patient is notified.
- Actual Result: Appointment status is updated in the database, and the patient is notified.
- Status: Pass/Fail

### **Test Case 5: Logout**

- Test Case ID: TC005
- Description: Verify that users can log out successfully.
- Preconditions: User must be logged in.
- Test Steps:
  1. Click on the "Logout" button.
- Expected Result: User is logged out, and the login page is displayed.
- Actual Result: (To be filled in after testing)
- Status: Pass/Fail

## **CHAPTER 7**

### **CONCLUSION AND FUTURE ENHANCEMENT**

#### **7.1 CONCLUSION**

The AppointMed system has been successfully developed to address the challenges of scheduling and managing doctor appointments, providing patients and healthcare professionals with an efficient, streamlined solution. By automating the process, AppointMed reduces manual administrative tasks, improves communication, and offers a structured approach to handling appointment requests. With its user-friendly interface and role-based functionality, the system enhances both patient experience and doctor efficiency, ensuring a smooth and transparent appointment management workflow.

Built on a robust three-tier architecture—consisting of a React.js frontend, a Spring Boot application layer, and a MySQL database—AppointMed offers scalability, maintainability, and high performance. The system underwent thorough testing, including unit, integration, and functional testing, to ensure it

meets functional requirements and operates reliably in different conditions. Real-time notifications improve communication, while the clear separation of roles ensures that both patients and doctors have access to the tools they need for seamless interaction.

Overall, AppointMed represents a substantial improvement in appointment management within the healthcare sector. By leveraging modern technology and incorporating a flexible, scalable design, the system is well-suited for future enhancements, such as calendar integrations or mobile app support. It provides a robust solution that not only addresses current scheduling needs but also offers potential for continuous growth and expansion.

## **7.2 FUTURE ENHANCEMENT**

The AppointMed system is designed to be scalable and adaptable, making it well-suited for future enhancements that can further improve its functionality and user experience. As healthcare needs grow and patient expectations evolve, several features can be added to the system to ensure it remains a valuable tool for managing doctor appointments effectively.

### **1. Integration with Calendar Systems**

One of the most beneficial future enhancements would be integrating AppointMed with popular calendar platforms such as Google Calendar, Microsoft Outlook, or Apple Calendar. This would allow patients and doctors to automatically sync their appointments with their personal calendars, ensuring they stay updated on scheduled appointments and avoid conflicts. This integration could also provide reminders and updates directly to the user's calendar, enhancing the convenience and reliability of the booking process.

## 2. Mobile Application

To further improve accessibility, developing a mobile application for both Android and iOS devices would allow users to manage their appointments on the go. The mobile app would offer key features, including booking appointments, viewing appointment statuses, and receiving realtime notifications. Additionally, a mobile app could leverage features like push notifications and alerts to keep users informed instantly about any changes. This would enhance the overall user experience, particularly for patients and healthcare providers who may need to check appointments quickly.

## 3. Fee Payment Integration

Incorporating a fee payment feature would streamline the appointment process by allowing patients to pay directly through the AppointMed system when booking their appointments. This could include secure payment gateway integration with options for credit/debit cards, UPI, and digital wallets, ensuring flexibility for users. By automating fee collection, the system can help reduce no-shows and ensure timely payments for healthcare providers. A digital invoice could also be sent to patients post-payment, enhancing transparency and record-keeping.

## 4. Feedback Mechanism and Analytics


Another useful enhancement would be implementing a feedback system where patients can rate their appointment experiences and provide suggestions for improvement. Along with this, adding analytics and reporting tools could help healthcare providers gather insights into appointment trends, patient satisfaction levels, and potential improvements in scheduling. This data could also be used to optimize doctor availability and improve the efficiency of the booking process.

## CHAPTER 8

### RESULT AND ANALYSIS


#### SCREEN LAYOUTS


##### LOGIN PAGE :



**Appointmed**

Welcome Back

Email 

Password 

☒ Patient ☐ Doctor

**Sign In**

Don't have an account ? [Sign Up](#)

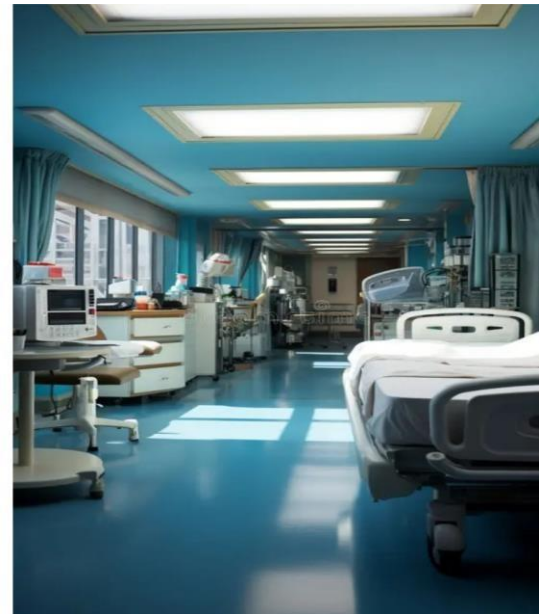



Fig.6. Login Page for Appointmed


##### SIGNUP PAGE :





**Appointmed**


Register Now


☒ Patient ☐ Doctor


Name 

Email 

Phone 

Password 

Age 

Gender 

**Sign Up**

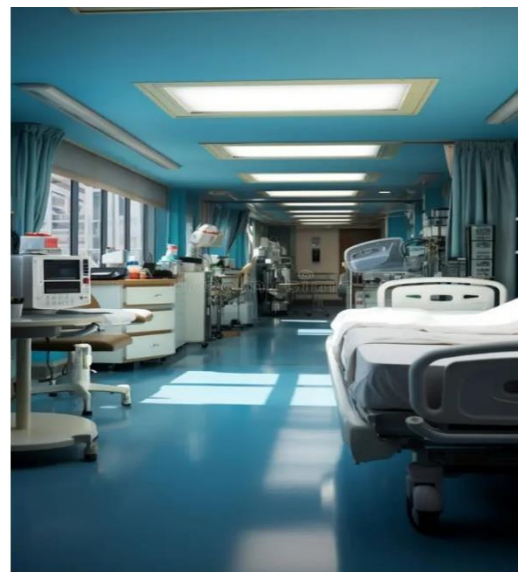


Fig.7. Signup Page for Appointmed

HOME PAGE :

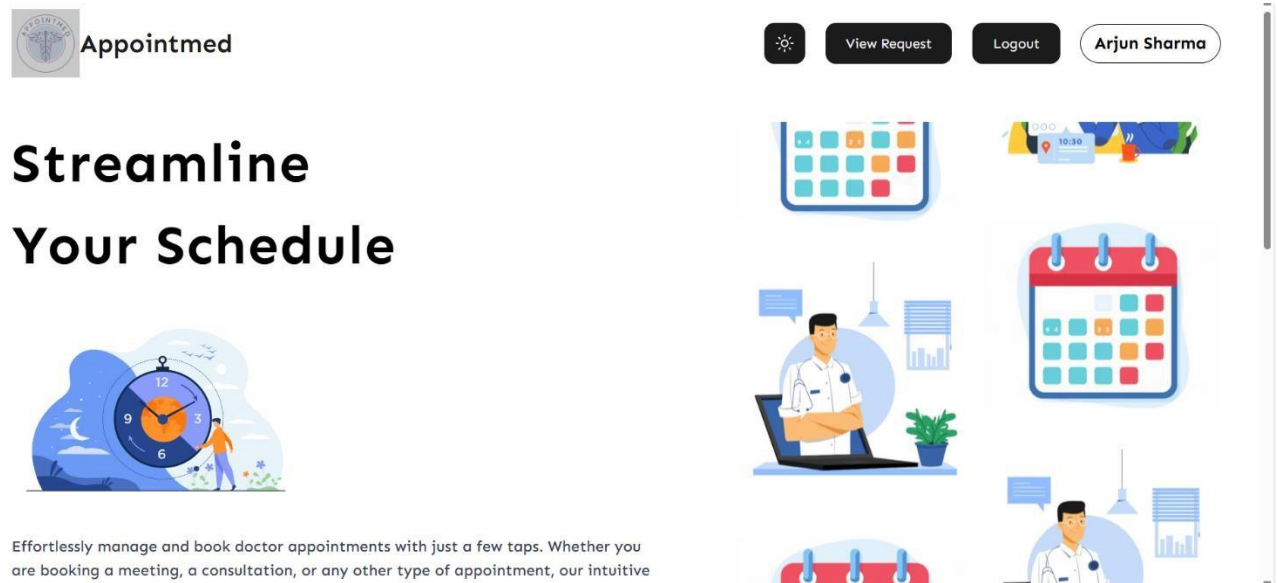


Fig 8. Home page

## Choose your city




Fig 9. Home page – select city

This interface lets the user view the available departments and on clicking the desired department, it leads into the faculty page.

## DOCTORS VIEW PAGE :


### Meet Our Esteemed Doctors

Our faculty members are the backbone of our institution, bringing years of experience, knowledge, and passion to the classroom. Each educator is dedicated to fostering a learning environment that encourages curiosity, innovation, and academic excellence. Explore their profiles to learn more about their qualifications and schedules.




Dr. Ramesh Kumar  
General Medicine

[Schedule](#)




Dr. Priya Deshmukh  
Cardiologist

[Schedule](#)



Dr. Karthik Nair  
Pediatrician

[Schedule](#)



Dr. Sunita Rao  
Gynecologist

[Schedule](#)

Fig. 10. View doctors

## APPOINTMENT BOOKING INTERFACE :

Meeting Date

24-10-2024



Select Available Time Slot

11.00 AM - 11.30 AM




Cancel


Schedule



Fig.11. Booking Appointment

VIEW REQUESTS PAGE :

Appointmed


[View Request](#)[Logout](#)


Arjun Sharma

### View Request

Doctor Name	Date and Time	Status
Dr. Meera Gupta	2024-11-04 09.00 AM - 09.30 AM	SCHEDULED
Dr. Sunita Rao	2024-10-24 11.00 AM - 11.30 AM	SCHEDULED

Fig.12. View request status – patient

Appointmed

[View Request](#)[Logout](#)

Dr. Ramesh Kumar

### View Request

Patient Name	Date and Time	Status
Ravi Patel	2024-11-06 10.00 AM - 10.30 AM	SCHEDULED

Cancel

Fig.13. View request status – doctor



## CHAPTER 9

### REFERENCES

- <https://reactjs.org/docs/getting-started.html>
- <https://tailwindcss.com/docs/installation>
- Dribbble. (n.d.). Dribbble. Retrieved October 13, 2024, from <https://dribbble.com/>
- FreeCodeCamp. (2022). Learn React Firebase full course. Retrieved October 13, 2024, from <https://www.freecodecamp.org/news/learn-react-firebase-full-course/>
- [https://www.researchgate.net/publication/316502021\\_Web-Based\\_Medical\\_Appointment\\_Systems\\_A\\_Systematic\\_Review](https://www.researchgate.net/publication/316502021_Web-Based_Medical_Appointment_Systems_A_Systematic_Review)
- <https://pmc.ncbi.nlm.nih.gov/articles/PMC5425771/>

## APPENDIX

### SOURCE CODE

#### User.java:

```
package com.project.appointmed.dto;
import java.util.Collection;
import java.util.Collections;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import jakarta.persistence.Entity; import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType; import jakarta.persistence.Id; import
jakarta.persistence.Inheritance; import jakarta.persistence.InheritanceType;
import lombok.AllArgsConstructor; import lombok.Getter; import
lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;
```

```

@Entity
@Inheritance(strategy = InheritanceType.JOINED)
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor @ToString
public abstract class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)    private Long id;
    private String name;    private String email;    private String password;
    private Role role;
    private String phone;

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return Collections.singletonList(new SimpleGrantedAuthority("ROLE_" +
role.name()));
    }

    @Override    public String getPassword() {
        return this.password;
    }

    @Override
    public String getUsername() {
        return this.email; // You can use email as username
    }

    @Override
    public boolean isAccountNonExpired() {
        return true; // Implement custom logic if necessary
    }

    @Override

```

```

    public boolean isAccountNonLocked() {
        return true; // Implement custom logic if necessary
    }

    @Override
    public boolean isCredentialsNonExpired() {    return true; // Implement
custom logic if necessary
    }

    @Override
    public boolean isEnabled() {
        return true; // Implement custom logic if necessary
    }
}

```

### **Patient.java:**

```
package com.project.appointmed.dto;
```

```
import jakarta.persistence.Entity; import lombok.Getter; import
lombok.NoArgsConstructor; import lombok.Setter;
```

```

@Entity
@Getter
@Setter
@NoArgsConstructor
public class Patient extends User {    private int age;    private String gender;

        public Patient(Long id, String name, String email, String password, Role role,
String phoneNumber,
                                int age, String gender) {
            super(id, name, email, password, role, phoneNumber);
            this.age = age;            this.gender = gender;
        }

}

```

### **Doctor.java:**

```
package com.project.appointmed.dto;
```

```
import jakarta.persistence.Entity; import lombok.Getter; import  
lombok.NoArgsConstructor;  
import lombok.Setter;
```

```
@Entity
```

```
@Getter
```

```
@Setter
```

```
@NoArgsConstructor public class Doctor extends User {    private String  
medicalLicense;    private String specialization;    private String clinicName;  
private String address;  
    private String city;
```

```
        public Doctor(Long id, String name, String email, String password, Role role,  
String phoneNumber, String medicalLicense,  
                        String specialization, String clinicName, String address, String  
city) {  
            super(id, name, email, password, role, phoneNumber);  
            this.medicalLicense = medicalLicense;            this.specialization =  
specialization;            this.clinicName = clinicName;            this.address =  
address;  
            this.city = city;  
        }
```

```
}
```

### **Appointment.java:**

```
package com.project.appointmed.dto;
```

```
import java.time.LocalDate;
```

```
import org.springframework.stereotype.Service;
```

```
import jakarta.persistence.Column; import jakarta.persistence.Entity; import  
jakarta.persistence.GeneratedValue; import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id; import jakarta.persistence.JoinColumn; import  
jakarta.persistence.ManyToOne; import lombok.AllArgsConstructor; import  
lombok.Getter; import lombok.NoArgsConstructor;  
import lombok.Setter;  
import lombok.ToString;
```

```
@Entity
```

```
@NoArgsConstructor
```

```
@AllArgsConstructor
```

```
@Getter
```

```
@Setter
```

```
@ToString
```

```
public class Appointment {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)    private Long id;
```

```
    @ManyToOne
```

```
    @JoinColumn(name = "patient_id", nullable = false)
```

```
    private Patient patient;
```

```
    @ManyToOne
```

```
    @JoinColumn(name = "doctor_id", nullable = false)
```

```
    private Doctor doctor;
```

```

    @ManyToOne
    @JoinColumn(name = "time_slot_id", nullable = false)    private TimeSlot
timeSlot;
40
    @Column(nullable = false)
    private LocalDate date;

    @Column(nullable = false)    private String status;
}

```

### **TimeSlot.java:**

```

package com.project.appointmed.dto;

import jakarta.persistence.Column; import jakarta.persistence.Entity; import
jakarta.persistence.GeneratedValue; import jakarta.persistence.GenerationType;
import jakarta.persistence.Id; import lombok.AllArgsConstructor; import
lombok.Getter; import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Entity
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@ToString public class TimeSlot {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)    private Long id;

    @Column(nullable = false)
    private String time;
}

```

