Infosys Springboard Internship 5.0

TITLE: 'OUTPATIENT APPOINTMENT SYSTEM'

Submitted By: TEAM 4

1. Sagnik Bhunia
2. Rajibul Islam
3. Pravallika Palla
4. Rikitha Oggula

Under the Guidance of:
Mr. Saikumar

Table of Contents

1. Abstract

The **Outpatient Doctor Appointment System** is a sophisticated web-based application designed to optimize the outpatient appointment scheduling process for patients and healthcare providers. It addresses the shortcomings of traditional systems by delivering a user-friendly platform that streamlines the booking process and empowers doctors to manage their schedules efficiently. Administrators benefit from powerful tools to oversee operations effectively. The system integrates essential features such as user authentication, appointment scheduling, doctor availability management, patient information handling, and notifications for reminders and updates, making it a comprehensive solution for outpatient management.

Leveraging modern technologies, the system uses **Java Spring Boot** for a robust backend and **React/Angular** for a seamless frontend experience. A standout feature is the integration of AI tools, which provide smart doctor recommendations tailored to patient needs. Furthermore, a secure payment gateway facilitates online consultation payments, and email notifications keep users informed about critical updates like bookings and cancellations. The modular development approach enabled systematic implementation and comprehensive testing with **JUnit** and **Mockito** which ensured reliability, performance, and data accuracy.

The system's scalability and security make it a versatile solution suitable for hospitals, clinics, and healthcare facilities of varying sizes. Its admin dashboard enhances operational efficiency by offering insightful reports on patient admissions and discharges. This feature supports administrators in making informed decisions and tracking hospital performance through graphical representations and user-friendly interfaces. These capabilities allow healthcare institutions to improve their operational workflows significantly.

By enhancing the patient experience and introducing innovative solutions such as AI-powered recommendations, the Outpatient Doctor Appointment System positions itself as a cornerstone of modern healthcare management. It addresses existing challenges in outpatient operations while laying a solid foundation for future enhancements. Planned advancements, including telemedicine integration and advanced analytics, ensure the system's adaptability to the dynamic needs of the healthcare industry.

2. Introduction

The Outpatient Doctor Appointment System is an advanced web application designed to revolutionize the process of outpatient appointment booking. Traditional methods of scheduling appointments are often time-consuming, prone to errors, and inconvenient for both patients and healthcare providers. These inefficiencies frequently lead to patient dissatisfaction, administrative bottlenecks, and challenges in resource management. By utilizing modern technologies, this project aims to overcome these challenges and provide a streamlined, user-friendly solution for managing outpatient appointments.

2.1 Project Statement:

The project aims to develop a user-friendly web application that facilitates the booking of outpatient appointments with doctors, streamlining the process for both patients and healthcare providers.

Key Outcomes:

- Allow patients to easily schedule appointments with preferred doctors.
- Provide doctors with a convenient way to manage their appointment schedules.
- Improve efficiency in outpatient appointment booking and management.
- Enhance patient satisfaction by offering a seamless and user-friendly interface.

Modules Implemented:

1. User Authentication and Registration: A secure system to allow patients, doctors, and administrators to register and log in.
2. Appointment Scheduling Interface: A user-friendly interface for patients to book appointments with their chosen doctors.
3. Doctor Availability Management: Tools for doctors to manage their availability and scheduling efficiently.
4. Patient Information Management: A centralized system to store and manage patient details securely.
5. Appointment Management: Features to allow modifications, cancellations, and real-time updates to appointments.
6. Notification System: Automated notifications to inform patients and doctors about upcoming appointments or changes.
7. Admin Dashboard: A comprehensive dashboard for administrators to oversee operations, generate reports, and manage resources effectively.

By implementing these modules, the system ensures a cohesive solution that bridges the gap between patients and healthcare providers. The integration of secure data management and a user-friendly interface further enhances the system's reliability and accessibility.

2.2 Problem Definition

The traditional outpatient appointment scheduling process is plagued by inefficiencies that negatively impact patients, doctors, and healthcare administrators. Manual systems are prone to errors, such as double bookings and scheduling conflicts, which lead to operational delays and patient frustration. Additionally, the lack of real-time updates often results in miscommunication, wasted time, and unproductive workflows for healthcare staff.

Key challenges in traditional systems include:

- **Inefficient Booking Processes**: Patients face difficulty in scheduling appointments due to unorganized systems and long waiting times.
- **Errors and Scheduling Conflicts**: Manual errors lead to double bookings and confusion, disrupting both patient care and staff productivity.
- **Lack of Real-Time Updates**: Absence of real-time notifications and updates creates gaps in communication, causing missed or delayed appointments.
- **Inadequate Resource Management**: Administrators struggle to allocate resources effectively due to the lack of centralized tools and data.

The proposed Outpatient Doctor Appointment System addresses these challenges through a web-based platform that simplifies and automates the appointment process. Patients can schedule appointments with ease, view real-time availability of doctors, and receive automated notifications about their appointments. Doctors benefit from tools that help them manage their schedules efficiently, avoiding conflicts and optimizing their time. The system also provides administrators with a dashboard to monitor operations, manage resources, and generate insightful reports to improve hospital efficiency.

By integrating advanced scheduling tools, automated reminders, and secure data management, the system bridges the gap between patients and healthcare providers. It promotes operational efficiency, reduces administrative burdens, and enhances patient satisfaction through a modern, reliable approach to outpatient appointment management.

3. SCOPE OF THE PROJECT

The **Outpatient Appointment System** is designed to simplify and enhance the process of managing outpatient appointments for patients, doctors, and healthcare administrators. The platform provides a seamless experience for users by enabling patients to search for doctors based on specialization, location, availability, and consultation fees. Advanced filters ensure patients can quickly find and book the right healthcare professional.

Secure login functionality allows patients, doctors, and admins to safely manage their profiles and access relevant features. Patients can view their appointment history, download prescriptions or invoices, and reschedule or cancel appointments effortlessly. Real-time notifications are sent via email or SMS, ensuring timely updates and confirmations.

The platform also includes a **Feedback and Review System**, enabling patients to rate and review their doctors, fostering transparency and trust. Doctors can manage their schedules, view patient details, and track appointments through a user-friendly dashboard.

For administrators, the system provides tools to manage doctor profiles, monitor appointment trends, and generate insightful reports to improve operational efficiency. The Outpatient Appointment System is scalable and built to adapt to the evolving needs of healthcare facilities, ensuring a robust and reliable solution for outpatient care.

4. TECHNICAL STACK

Programming Languages

- **Java:** Java is used as the core programming language for backend development. It helps implement the business logic, data handling, and server-side functionality in your application. Specifically, it is utilized with Spring Boot to build RESTful APIs that serve as the bridge between your frontend (React.js) and the MySQL database.
- JavaScript: JavaScript is employed on the frontend to make the application interactive and responsive. React.js, a JavaScript library, is used to build the user interface (UI), handle events, and communicate with the backend. JavaScript helps manage dynamic content updates without requiring a page refresh, providing a seamless user experience.

Frameworks/Libraries

- **Spring Boot**: Spring Boot is a Java-based framework that simplifies the process of building and deploying applications. In your project, it is used to create RESTful APIs, which handle HTTP requests and responses. It also provides built-in features like dependency injection, security, and database connections, making backend development faster and more efficient. Spring Boot's flexibility allows the project to scale well and integrate with various databases and services.
- React.js: React.js is a popular frontend library for building user interfaces. It helps create dynamic, single-page applications (SPAs) where only specific parts of the page are updated, improving performance. React's component-based architecture makes it easy to reuse code, maintain the UI, and ensure a responsive design across devices. React's virtual DOM ensures that updates to the UI are fast and efficient.

Databases

- **MySQL**: MySQL is a relational database management system (RDBMS) used to store and manage the data in your application. In this case, MySQL is used to store crucial information such as patient details, doctor schedules, appointment records, and feedback. MySQL provides a structured way to store data in tables with relationships, ensuring data consistency and integrity. The MySQL database is connected to your Spring Boot backend using JDBC (Java Database Connectivity), with MySQL Connector/J as the JDBC driver.

Tools/Platforms

- **IntelliJ IDEA**: IntelliJ IDEA is a powerful integrated development environment (IDE) used for backend development. It offers code suggestions, debugging tools, version control integration, and other features that improve development productivity. It is particularly suited for Java-based projects like the one you're working on, making it easier to write, test, and manage your Spring Boot application.
- Visual Studio Code (VS Code): VS Code is a lightweight yet powerful code editor used for frontend development with React.js. It supports a wide range of extensions that enhance productivity, such as live server, code linting, and Git integration. It’s especially popular for JavaScript development due to its flexibility and ease of use, making it a great tool for building dynamic user interfaces with React.

- Postman: Postman is an API testing tool that simplifies the process of designing, testing, and validating RESTful APIs. It allows you to simulate different HTTP requests (GET, POST, PUT, DELETE) and inspect the responses from your backend. This ensures that the communication between your React.js frontend and the Spring Boot backend is smooth, secure, and error-free. You can also use Postman for debugging API calls and checking for any issues before deployment.

5. ARCHITECTURE/DESIGN

Frontend Layer

- Technology: React.js
- Description: Provides a dynamic and responsive interface, handling user interactions like login, search, booking, and review submissions. It uses React components for efficient state management and rendering.

Backend Layer

- Technology: Spring Boot
- Description: Manages business logic, handles API requests, and processes data. It exposes RESTful endpoints for authentication, doctor listings, appointment bookings, and reviews, ensuring secure data handling.

Database Layer

- **Technology**: **MySQL**
- **Description**: Stores patient details, doctor information, appointments, and feedback. It handles efficient data retrieval and updates, ensuring smooth appointment management.

API Gateway

- **Description**: Facilitates secure and efficient communication between the frontend and backend, ensuring seamless data flow.

Interactions

- **Frontend & Backend**: The frontend communicates with the backend through **RESTful APIs** for all user actions (e.g., booking, searching, and reviewing).
- **Backend & Database**: The backend interacts with the **MySQL database** for storing and retrieving user, appointment, and feedback data.

6. PROJECT REQUIREMENTS

Functional Requirements

1. Appointment Booking Service:
    - The system allows patients to browse a list of doctors based on various filters like specialization, location, consultation fee, and availability.
    - Patients can view detailed doctor profiles, including qualifications, experience, and ratings, before booking an appointment.
    - The system ensures a smooth booking process with instant confirmation via email or SMS notifications.
2. User Authentication:
    - Secure login and registration functionality for patients, doctors, and admins.
    - Role-based access control to ensure each user type (patient, doctor, admin) sees only the features relevant to them.
    - Features password recovery and secure password storage using encryption techniques.
3. Real-Time Updates:
    - Appointment availability is updated in real time to avoid double bookings.
    - Patients and doctors receive instant notifications for confirmed, rescheduled, or canceled appointments.
4. Search and Filters:
    - Patients can search for doctors based on specific criteria such as specialization (e.g., cardiologist, pediatrician), location, consultation fees, or available time slots.
    - Advanced filters allow sorting by rating, experience, or proximity to the user's location.
5. Feedback System:
    - Patients can provide detailed feedback on their consultation experience, including ratings and written reviews.
    - Doctors can view and respond to feedback for continuous improvement and building trust.
6. Profile Management:
    - Patients can update personal details such as name, email, phone number, and address.
    - Doctors can update their availability, consultation fees, and other professional information.

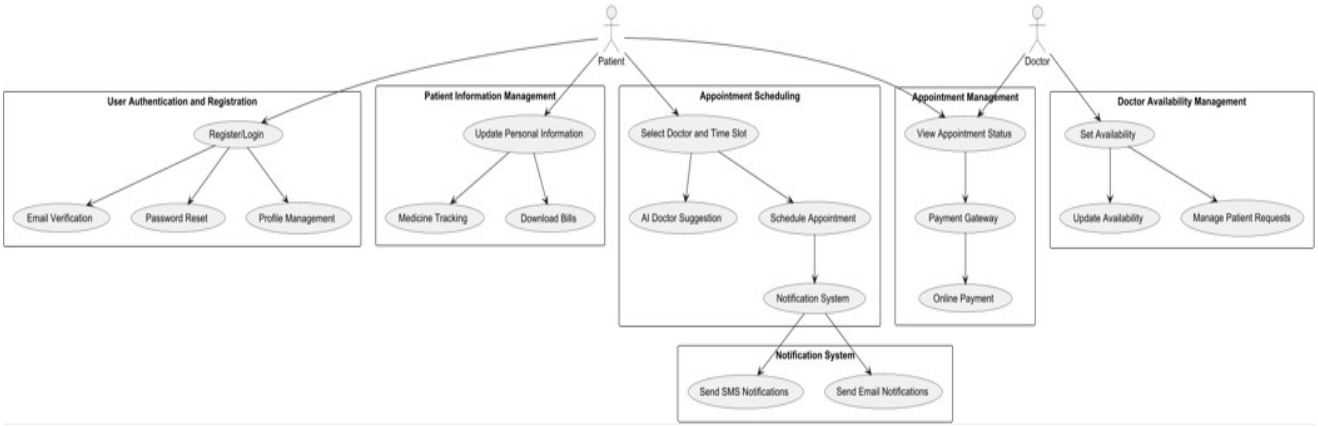
    - Users can view appointment history and download invoices or prescriptions.
7. Doctor Management Module:
    - Admins can add, update, or delete doctor profiles, ensuring the system stays up-to-date.
    - Admins can assign or modify schedules for doctors to maintain appointment slots efficiently.
8. Appointment Dashboard:
    - Doctors have access to a personalized dashboard showing upcoming appointments, patient details, and appointment statuses.
    - Doctors can accept, decline, or reschedule appointments based on their availability.
9. Statistics and Reports:
    - Admins can generate detailed reports on system usage, including the number of appointments booked, most consulted doctors, patient demographics, and peak booking times.
    - Data visualizations such as charts and graphs make it easier to analyze trends and improve system operations.

Non-Functional Requirements

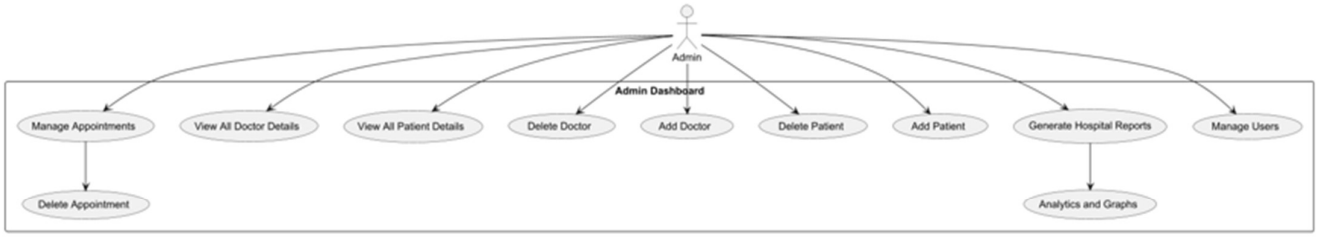
1. Performance:
    - The system ensures quick responses, with search and booking actions completing within 1 second under normal load conditions.
    - The platform supports up to 500 concurrent users without noticeable degradation in performance.
2. Security:
    - Implements industry-standard security protocols, including HTTPS, to ensure secure data transmission.
    - User credentials are encrypted and stored securely in the database.
    - Multi-factor authentication (MFA) is available for additional security.
3. Usability:
    - The interface is designed to be user-friendly, ensuring that even non-technical users can navigate the system easily.
    - A patient can complete the entire appointment booking process in three steps: search, select, and confirm.
4. Maintainability:
    - The code follows a modular architecture, making it easy to update or add new features without affecting the existing functionality.
    - Comprehensive documentation is provided for developers to ensure smooth debugging and system maintenance.

5. Scalability:
    - The system is built to scale seamlessly, allowing the addition of new users, doctors, or clinics without performance bottlenecks.
    - Supports both single-clinic and multi-clinic setups for potential expansion.
6. Availability:
    - The platform guarantees 99.9% uptime, ensuring consistent access for all users.
    - Redundant backups and failover mechanisms are implemented to prevent data loss and downtime in case of server failures.
7. Accessibility:
    - The system adheres to accessibility standards, ensuring users with disabilities can use the platform.
    - Features include screen reader compatibility and keyboard navigation support.

7.Use Case Diagram

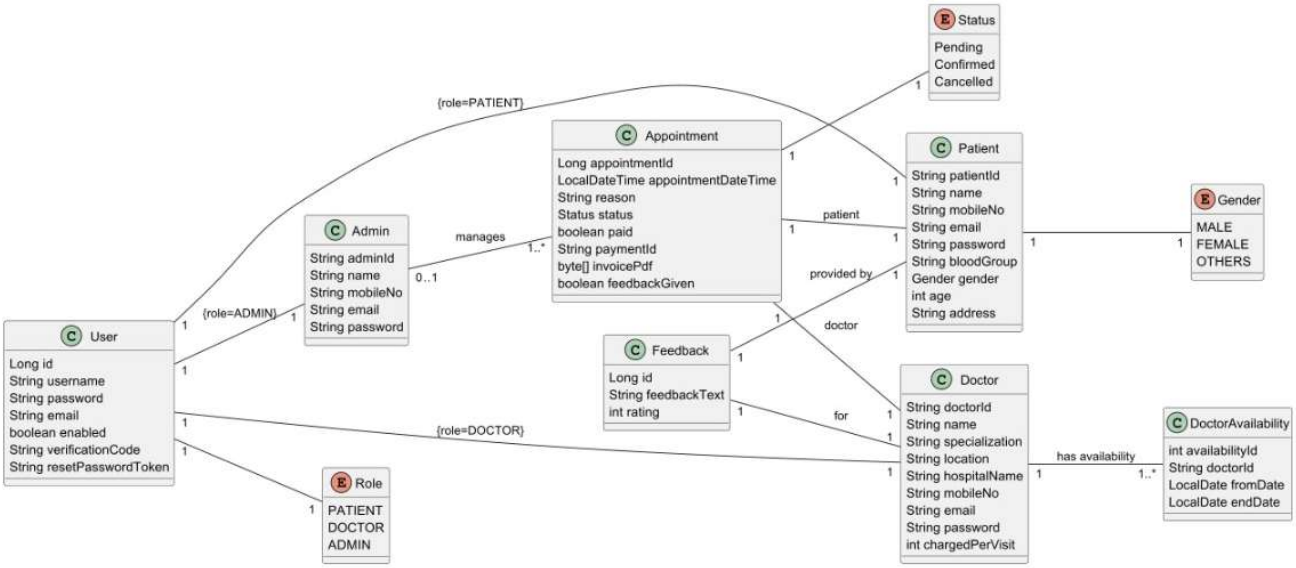


The first diagram outlines patient interactions with the system. It includes **User Authentication and Registration** for secure access, **Patient Information Management** for updating details, tracking medicines, and downloading bills, and **Appointment Scheduling** with AI doctor suggestions and notifications. Additionally, **Doctor Availability Management** enables doctors to set schedules, ensuring efficient service and patient satisfaction.



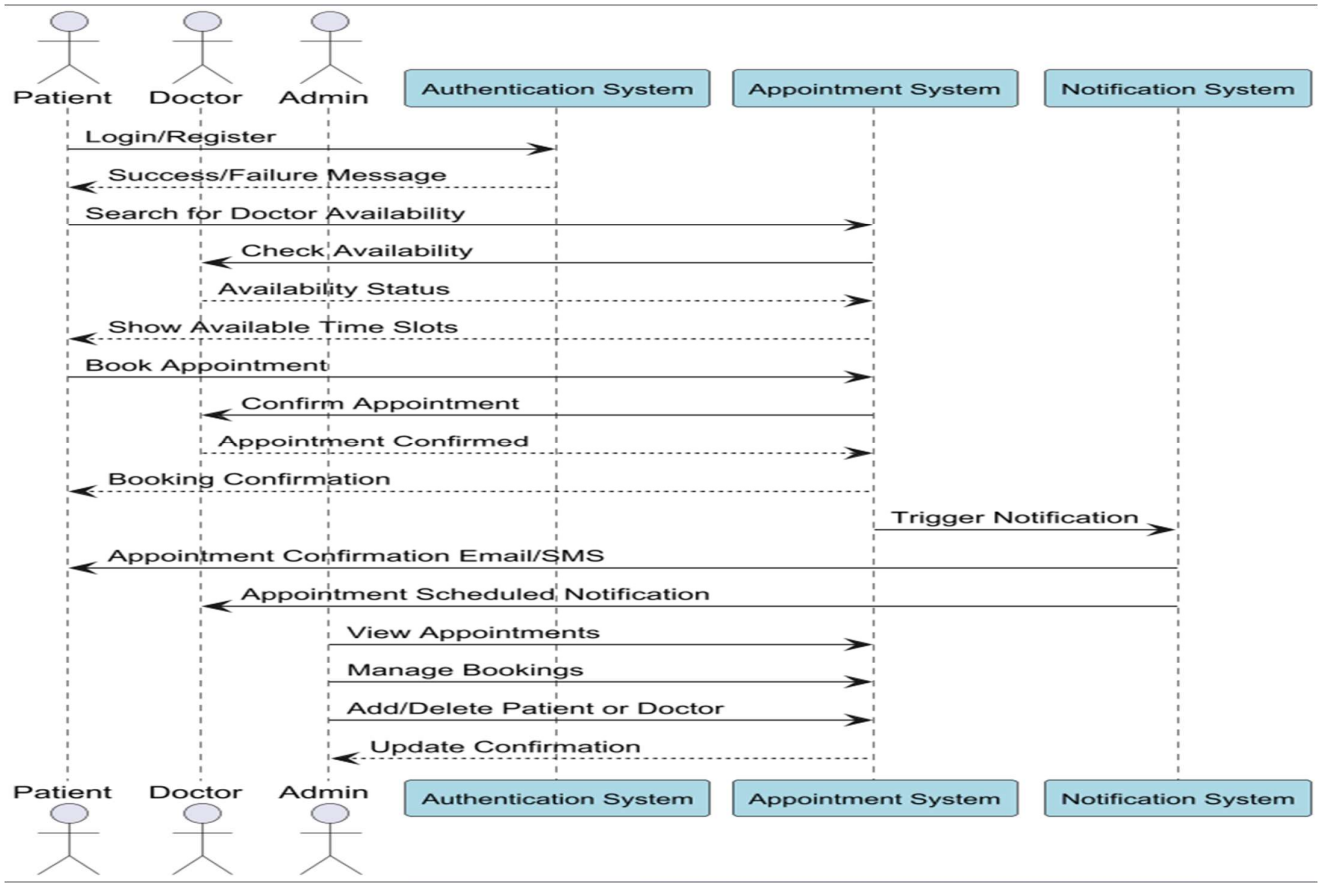
The second diagram highlights the admin's role, including managing appointments, doctor and patient records, and user accounts through the **Admin Dashboard**. It supports generating reports with analytics for data-driven decisions and ensures efficient backend operations, enhancing system functionality.

8.Class Diagram



The class diagram represents a healthcare management system where users can have roles like Admin, Doctor, or Patient. Patients can book appointments with doctors, provide feedback, and view doctor availability. Admins manage the system, while doctors handle appointments and specify their availability. The system also includes attributes like appointment status, payment details, and user-specific information such as gender and roles.

9.Sequence Diagram



The sequence diagram outlines the interaction among Patients, Doctors, Admins, and various system components like Authentication, Appointment, and Notification Systems. Patients can log in, search for doctor availability, and book appointments, which are confirmed by the system. Notifications are sent via email/SMS for appointment confirmation. Admins manage bookings by adding or deleting patients or doctors, while users can view or update their appointments seamlessly.

10. DEVELOPMENT

1. **Java (Spring Boot) - Backend Development:**
    - **Spring Boot**: Simplifies backend development with its pre-configured framework, enabling rapid creation of production-ready applications with minimal setup.
    - **Spring Data JPA**: Streamlines database interactions and CRUD operations by allowing the creation of repositories for data models with minimal boilerplate code. It supports seamless integration with MySQL through the Java Persistence API (JPA).
    - **Spring Security**: Ensures secure authentication and role-based authorization for patients, doctors, and administrators, safeguarding the platform from unauthorized access.
2. **JavaScript (React.js) - Frontend Development:**
    - **React.js**: A versatile JavaScript library used for building responsive and interactive user interfaces. It offers a component-based structure, efficient rendering, and state management for an optimized user experience.
    - **React Router**: Implements dynamic routing, enabling smooth navigation between components like the homepage, booking pages, and user dashboards.
    - **Axios**: Used as an HTTP client for making API requests to the backend, ensuring efficient data exchange through RESTful APIs.
3. **Database:**
    - **MySQL**: A relational database used for storing user details, doctor schedules, appointments, and feedback. It ensures reliable data storage and efficient query performance.
    - **MySQL Connector/J**: The official JDBC driver used to connect the Java-based Spring Boot application with the MySQL database.
4. **Build and Deployment Tools:**
    - **Maven**: Automates dependency management, project builds, and packaging for the Spring Boot application, ensuring a streamlined development process.
    - **Postman**: A powerful tool used for testing and validating RESTful APIs, enabling seamless communication between the frontend and backend during development.

11. TESTING

1. Unit Testing

- **Focus**: Validating individual methods, API endpoints, and backend functionalities in isolation.
- **Tools Used**: JUnit for implementing and executing test cases.
- **Examples**:
    - Testing API endpoints for user registration with valid and invalid input scenarios.
    - Verifying the backend logic for scheduling doctor appointments and ensuring data integrity.

2. Integration Testing

- **Focus**: Ensuring smooth interaction between backend components, APIs, and the database.
- **Examples**:
    - Testing the appointment booking process to confirm correct data linkage with doctor and patient details.
    - Verifying the retrieval of doctor schedules and appointment lists from the MySQL database.

3. System Testing

- **Focus**: Conducting end-to-end testing of application workflows.
- **Scenarios Covered**:
    - Registering a user, logging in, and successfully booking an appointment.
    - Filtering doctor search results based on specialization and availability.
    - Completing the booking process and verifying the confirmation details are accurate.

4. Manual Testing

- **Focus**: Exploring the React.js frontend to ensure the interface is responsive, user-friendly, and intuitive.
- **Activities**:
    - Verifying UI responsiveness on various devices.
    - Testing interactive elements like forms, search filters, and navigation.

Testing Results

1. Unit Testing Results

- Achieved **90% backend code coverage** through test cases.
- Fixed critical bugs, such as:
    - Incorrect validation for appointment time conflicts.
    - Handling invalid inputs during patient registration.

2. Integration Testing Results

- Detected and resolved key integration issues:
    - Mismatched entity-to-database column mappings for appointment details and user profiles.
    - Incorrect API behavior when optional filters for doctor specialization or availability were applied.

3. System Testing Results

- Successfully validated complete workflows with diverse test cases, including edge cases:
    - Overlapping appointment bookings to ensure proper error handling.
    - API response correctness for both valid and invalid booking scenarios.
    - Real-time updates for doctor availability reflected in the frontend.

4. Identified Bugs and Resolutions

- **Issue**: Users unable to cancel appointments after the scheduled time, causing confusion in the workflow.
    - **Resolution**: Modified the backend logic to allow appointment cancellations with appropriate warnings for late actions.
- **Issue**: Appointment confirmation emails occasionally included incorrect date or time due to timezone mismatches.
    - **Resolution**: Synchronized backend and frontend timezone management to ensure accurate timestamps in communications.
- **Issue**: User profile photo URLs were being truncated or improperly formatted during saving.
    - **Resolution**: Adjusted backend logic to handle full URL strings and added validation to ensure the correct format before storing.

This comprehensive testing approach ensured the stability, functionality, and user-friendliness of the Outpatient Appointment System, preparing it for deployment.

12. Deployment for Outpatient Appointment System

AWS Deployment

1. **Set Up EC2 Instance**:
   - Launch an Amazon EC2 instance (Ubuntu or Amazon Linux 2) to host the backend.
   - Configure security groups and SSH access.
2. **Install Java & Dependencies**:
   - Install the required Java Runtime Environment (JRE) and JDK on the instance to run the Spring Boot application.
3. **Deploy Backend**:
   - Upload the Spring Boot .jar file to the EC2 instance.
   - Start the application using: java -jar outpatient-system.jar.
4. **Deploy Frontend on S3**:
   - Build the React frontend using npm run build.
   - Upload the build folder to an Amazon S3 bucket for hosting static files.
5. **Set Up CloudFront**:
   - Use Amazon CloudFront as a CDN to serve static files efficiently from S3.
6. **Set Up Database**:
   - Configure Amazon RDS for MySQL to store application data.
   - Update the backend configuration to connect to the RDS instance securely.
7. **Automate Deployment**:
   - Set up CI/CD pipelines using AWS CodePipeline and CodeBuild to automate deployment and integration.
8. **Monitoring**:
   - Use AWS CloudWatch to monitor application logs, metrics, and performance.

Heroku Deployment

1. **Install Heroku CLI**:
   - Install Heroku CLI and authenticate using heroku login.
2. **Create Heroku App**:
   - Create a new app with heroku create outpatient-app.
3. **Deploy Backend**:
   - Push the Spring Boot backend code to Heroku using Git. Include a Procfile with web: java -jar target/outpatient-system.jar.
4. **Deploy Frontend**:
   - Build the React frontend using npm run build and push the static files to Heroku.

5. **Database Setup**:
    - Use the Heroku Postgres add-on to set up the database with the command: heroku addons:create heroku-postgresql:hobby-dev.
6. **Enable CI/CD**:
    - Configure Heroku pipelines for continuous integration and deployment.
7. **Monitoring**:
    - Monitor application logs using Heroku Logs and app performance with Heroku Metrics.

Azure Deployment

1. **Create Azure Virtual Machine**:
    - Launch an Azure VM (Ubuntu or Windows) for hosting the backend application.
2. **Install Java**:
    - Install JDK on the VM to support Spring Boot backend execution.
3. **Deploy Backend**:
    - Upload the .jar file to the Azure VM and run it with: java -jar outpatient-system.jar.
4. **Deploy Frontend**:
    - Build the React frontend with npm run build and upload static files to Azure Blob Storage.
5. **Set Up Azure CDN**:
    - Use Azure CDN to deliver static assets from Blob Storage efficiently.
6. **Database Configuration**:
    - Use Azure Database for MySQL to manage data and connect it to the backend.
7. **Automation with Azure DevOps**:
    - Implement Azure DevOps pipelines for automated builds and deployments.
8. **Monitoring and Scaling**:
    - Enable Azure Monitor for performance tracking and set up Auto-Scaling for resource management.
9. **Application Insights**:
    - Use Application Insights to monitor real-time performance and diagnose issues.

This deployment strategy ensures scalability, reliability, and efficient resource utilization across platforms, supporting the seamless operation of the Outpatient Appointment System.

13. Outpatient Appointment System User Guide

Backend Setup (Spring Boot)

1. **Open Project Directory**:
   - Navigate to the root directory of the Spring Boot project.
2. **Install Dependencies**:
   - Run mvn install to download and configure all required dependencies.
3. **Run the Application**:
   - Execute the command mvn spring-boot:run or directly run the OutpatientSystemApplication.java file in your IDE (IntelliJ IDEA).
4. **Verify Backend**:
   - Open a browser and go to http://localhost:8080 to confirm that the backend is running.
5. **Database Configuration**:
   - Ensure the MySQL database is running and properly configured in the application.properties or application.yml file.

Frontend Setup (React)

1. **Clone Repository**:
   - Run git clone <frontend_repository_url> to clone the frontend code.
2. **Install Dependencies**:
   - Navigate to the project folder and execute npm install to set up all required packages.
3. **Run the Application**:
   - Start the React app using npm start.
   - Access the frontend in your browser at http://localhost:3000.

Instructions for Use

1. **Initial Setup**:
   - Ensure the following are installed and configured: Java (JDK), Spring Boot, Node.js, React.js, MySQL, and Postman (optional for API testing).
2. **User Registration/Login**:
   - Access the platform and create a new account or log in with existing credentials.
3. **Forgot Password**:
   - Use the "Forgot Password" option on the login screen to reset your credentials.
4. **Dashboard Navigation**:
   - After logging in, use the dashboard to view available appointments, doctors, and clinic details.
5. **Book an Appointment**:

    - Select a doctor, choose an available time slot, and confirm the appointment.
6. **Manage Appointments**:
    - View upcoming appointments, reschedule if necessary, or cancel appointments.
7. **Provide Feedback**:
    - After a completed appointment, use the feedback form to rate the service and provide comments.
8. **Admin Features**:
    - Admin users can log in to manage doctors, schedules, and patient records.

Support

- **For Issues**:
    - Check the logs on the backend and frontend for any errors.
    - Ensure all dependencies are installed and database credentials are accurate.
- **Contact Support**:
    - Reach out to the project administrator for assistance with deployment or usage.

This guide ensures a smooth setup and operation of the Outpatient Appointment System.

14. Conclusion

The Outpatient Appointment System is a comprehensive solution designed to streamline the process of managing appointments for healthcare facilities. It bridges the gap between patients and healthcare providers, ensuring an efficient and user-friendly experience. The system automates crucial tasks such as scheduling appointments, managing patient records, and providing timely notifications, which significantly reduces manual effort and human error.

By leveraging modern technologies like React.js for the front end, Spring Boot for the back end, and MySQL for database management, the system ensures high performance, scalability, and reliability. The integration of secure authentication mechanisms and role-based access control ensures data privacy and protection, catering to the sensitive nature of healthcare information. Furthermore, features such as feedback submission and appointment rescheduling enhance user satisfaction and engagement.

The platform is designed to be adaptable to the needs of various healthcare facilities, from small clinics to large hospitals. Its intuitive interface, real-time updates, and efficient workflows make it accessible to users with minimal technical expertise. By reducing wait times, improving communication, and optimizing resource allocation, the system contributes to better healthcare outcomes and operational efficiency.

In conclusion, the Outpatient Appointment System is a vital tool in modern healthcare, promoting convenience, transparency, and reliability for both patients and providers.

15. APPENDIX:

GitHub Repository: https://github.com/Bhunia-Sagnik/Infosys_EDoctorTeam