CAPSTONE PROJECT

PROJECT TITLE
CURE WELL – HOSPITAL MANAGEMENT WEB APPLICATION

PROJECT BY

B. KUMAR
M. SANTHOSH
P. NAVEEN
R. VINAY
M. MOTHISH

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Welcome to an in-depth exploration of the groundbreaking "Cure Well - Hospital Management Web Application" project. This visionary web application represents a meticulous endeavor aimed at elevating operational efficiency and streamlining administration at the esteemed "Cure Well" hospital in Sydney. Within the following pages, we invite you to embark on a comprehensive journey through the intricate components, overarching objectives, and the profound impact that this state-of-the-art hospital management solution promises to make in the realm of contemporary healthcare administration..

The inception of the "Cure Well - Hospital Management Web Application" project arises from the imperative to optimize the operational efficacy and administration of the esteemed "Cure Well" hospital in Sydney. Within the hospital context, the efficient orchestration and synchronization of medical resources, notably in the realm of surgical procedures, holds paramount significance in the delivery of top-tier patient care. The manual handling of these operational facets can be fraught with errors and consume valuable time. Hence, the hospital's leadership has made the strategic choice to embark on the development of a web application tailored to streamline these intricate processes.

CHAPTER 2
System Overview

A system overview provides a high-level description of the Curewell Hospital Management web application, outlining its components and how they work together to achieve the project's goals. Here's a system overview for your project:

System Overview: Curewell Hospital Management Web Application

1. User Roles:

Admin: Responsible for overall system management, including

2. Key Features:

a. **Doctor Management:**- Admin and hospital staff can add, edit, and categorize doctors based on specializations.

b. **Authentication and Access Control:**- Secure login and user authentication for all roles. - Role-based access control ensures data privacy and security.

c. **Data Security and Compliance:**- Ensures data is encrypted and compliant with healthcare regulations (e.g., HIPAA).

d. **User-Friendly Interface:**- Provides an intuitive and responsive interface for users to access information and perform tasks.

3. System Components:

a. **Frontend Interface:**- User-friendly web interface built using Angular for the client-side. - Bootstrap for responsive design and styling.

b. **Backend Services:**- Developed using C#.NET for server-side logic. - Handles user authentication, data processing, and communication with the database.

c. **Database:**- Stores doctor profiles, patient records, appointment details, and surgical schedules. - Utilizes a relational database system (e.g., MSSQL) for data storage.

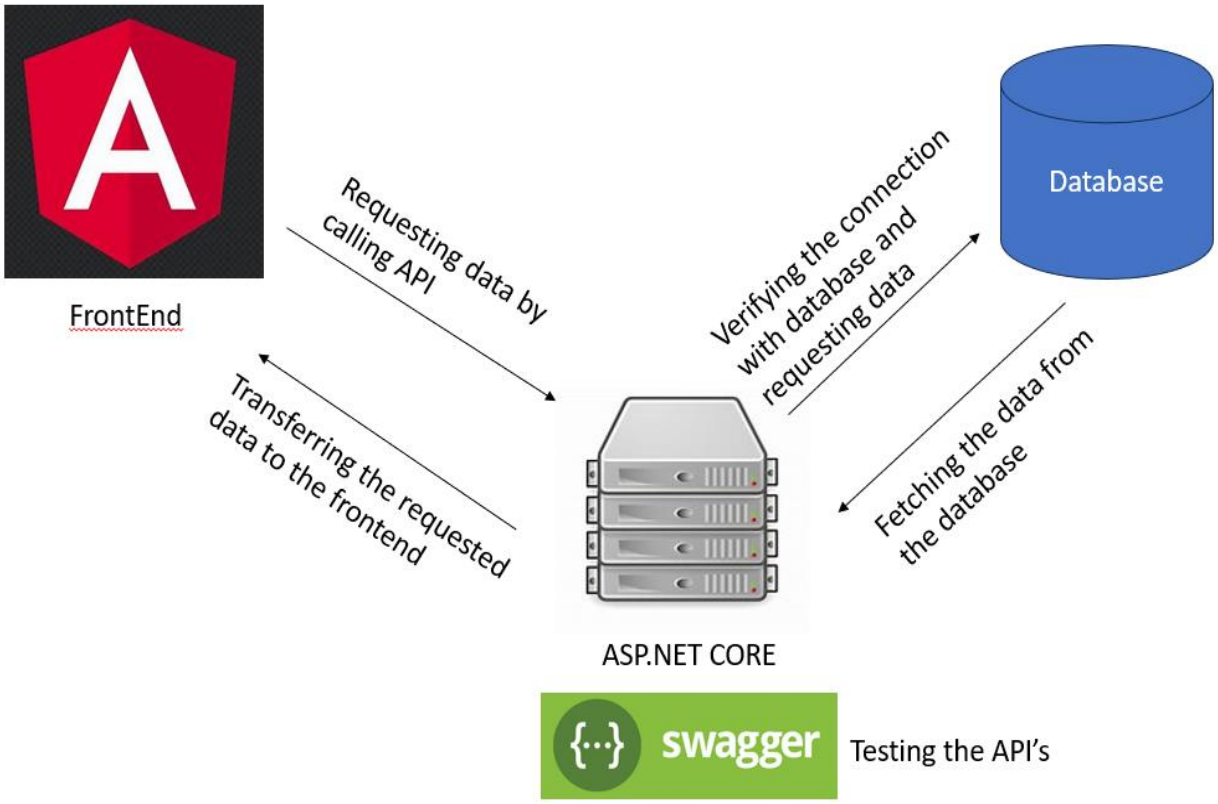
d. **Authentication Services:**- Utilizes JWT (JSON Web Tokens) for secure user authentication. - Manages user sessions and access control.

4. Data Flow:

- Admin access the web application through the frontend interface.
- Admin authentication is handled through the authentication services.
- Data is securely stored and retrieved from the database.

This system overview outlines the core components and functionalities of the Curewell Hospital Management web application, highlighting its user roles, features, architecture, and data flow. It emphasizes the system's commitment to data security, user-friendliness, and compliance with healthcare regulations.

CHAPTER 3
SYSTEM-ARCHITECTURE



The frontend requests data from the backend by calling an API.

The backend verifies the connection with the database and requests the data.

The database fetches the data and returns it to the backend.

The backend transfers the requested data to the frontend.

The diagram also shows that Swagger is used to test the APIs. Swagger is a tool that helps developers generate documentation and test APIs.

1. The frontend requests data from the backend by calling an API.

The frontend is the part of the web application that the user interacts with. It is typically built using HTML, CSS, and JavaScript. The backend is the part of the web application that is responsible for processing requests and generating responses. It is typically built using a programming language such as Python, Java, or C#.

APIs are used to communicate between the frontend and the backend. The frontend calls an API to request data from the backend. The backend then processes the request and returns a response to the frontend.

2. The backend verifies the connection with the database and requests the data.

Once the backend receives a request from the frontend, it needs to verify the connection with the database. If the connection is successful, the backend can then request the data from the database.

3. The database fetches the data and returns it to the backend.

The database is the part of the web application that stores data. It is typically a relational database such as MySQL or PostgreSQL.

Once the backend requests data from the database, the database fetches the data and returns it to the backend.

4. The backend transfers the requested data to the frontend.

Once the backend receives the data from the database, it transfers the requested data to the frontend. The frontend can then use the data to display it to the user or to perform other tasks.

Swagger

Swagger is a tool that helps developers generate documentation and test APIs. It can be used to generate documentation in a variety of formats, including HTML, JSON, and YAML. It can also be used to generate test cases for APIs.

To use Swagger to test an API, the developer first needs to generate a Swagger Spec. A Swagger Spec is a JSON document that describes the API. Once the Swagger Spec is generated, the developer can use a variety of tools to test the API, including the Swagger Editor and the Swagger Hub.

The diagram shows that Swagger is used to test the APIs in the web application. This is a good practice because it helps to ensure that the APIs are working as expected.

CHAPTER 4
SYSTEM FEATURES

- Authentication : CureWell hospital management, boasts a robust authentication system implemented at the login page. This feature ensures the security of your application by employing JSON Web Tokens (JWT) for user authentication, allowing only authorized individuals to access the system. Users are required to provide valid login credentials, such as a username and password, to gain entry. Upon successful authentication, users receive a token that is subsequently used to validate their identity in subsequent interactions with the server, eliminating the need for repeated credential input.
- Doctor Management: One of the core functionalities of your application is Doctor Management, which grants administrators or authorized users the ability to oversee and control doctor-related information within the hospital. This feature encompasses tasks such as adding new doctors, updating existing doctor profiles, and deleting doctor records. It facilitates the storage of crucial data about doctors, encompassing details like their names, contact information, specialization, and work schedules. This data is typically managed through API endpoints and stored in a database.

- Surgery Tracking: Surgery Tracking is another pivotal aspect of your application, enabling the efficient monitoring and management of surgical procedures conducted within the hospital. Users can schedule surgeries, allocate surgeons and support staff, assign resources, and monitor the progress of surgeries through this feature. It includes functionalities for creating surgery records, updating the status of ongoing surgeries, and generating comprehensive reports on surgical activities. Surgery Tracking ensures that the hospital can streamline its surgical operations and enhance patient care.

- User Friendly Interface : To enhance the user experience, your application boasts a User-Friendly Interface. This feature is dedicated to creating an intuitive and visually appealing layout that caters to users of all backgrounds

and skill levels. It incorporates responsive design principles, ensuring that the application functions seamlessly across various devices and screen sizes. Key elements include clear navigation menus, user-friendly forms, informative dashboards, and interactive components, collectively contributing to a user-centric design that facilitates ease of use

Overall, your CureWell hospital management web application combines secure authentication, efficient doctor management, surgery tracking, and a user-friendly interface to provide a comprehensive solution for hospital administration and management. These features collectively contribute to the smooth operation of the hospital and the effective delivery of healthcare services.

CHAPTER 5
USER ROLES AND PERMISSIONS

The project that is being proposed is a machine learning model that can be accessed by anyone who has internet access. This machine learning model can be easily used by any one by simply entering your credentials without revealing any personal information. One of the most appealing features of this model is to check the secure webpage using certain features. Without the permission of the admin, no other user or individual would have access to this model. Because the Machine learning algorithmic technique used in the development of this model is completely free, there will be no service tax charged to the end user. Another unique feature of this application is that, unlike most others, it does not require many installations, only a very few installations can be done and even take up a small amount of space on your device, saving the user's memory storage.

Feasibility study is an important phase in the software development process. Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. Feasibility study should be performed on the basis of various criteria and parameters. The various feasibility studies are:

- Technical Feasibility
- Operation Feasibility
- Economic Feasibility

5.1 Technical feasibility

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment's have the technical capacity to hold the data required to use the new system?

- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?

5.2 Operational feasibility

- **User-friendly:** Not only programmers have to analyze data and Python can be useful for almost everyone in an office job. The problem is non-technical people are scared to death of making even the tiniest change to the code. So, it is very user-friendly.

- **Security:** As we are developing our model on PyCharm, no one can access your own private PyCharm and will be protected from hacking, virus etc.

- **Portability:** The application will be developed using google chrome as there are no restrictions.

So, we can use both on Windows and Linux o/s. Hence portability problems will not rise.

- **Availability:** This software will be available always.

- **Performance:** Use the computing power of the Google servers instead of your own machine. Running python scripts requires often a lot of computing power and can take time. By running scripts in the cloud, you don't need to worry. Your local machine performance won't drop while executing your Python scripts.

5.3 Economic feasibility

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any additional hardware or software. The implementation of the model need not requires any separate software installation, it reduces the budget of the project.

CHAPTER 6
SCREEN SHOT FOR MAIN FEATURES

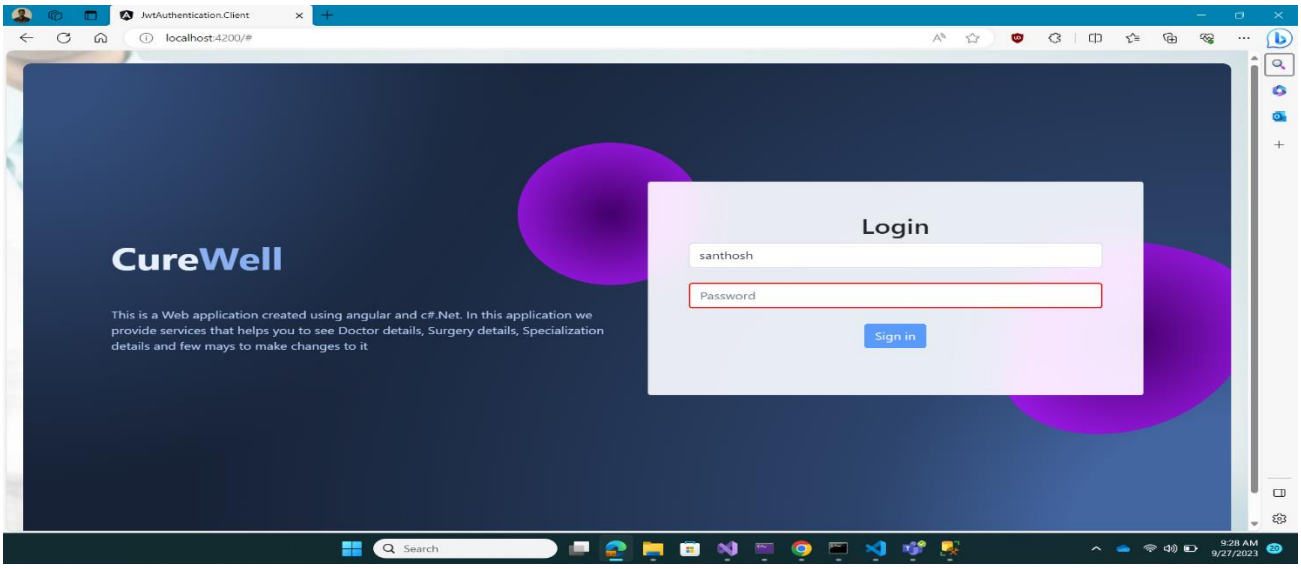


Fig 1: Login Page

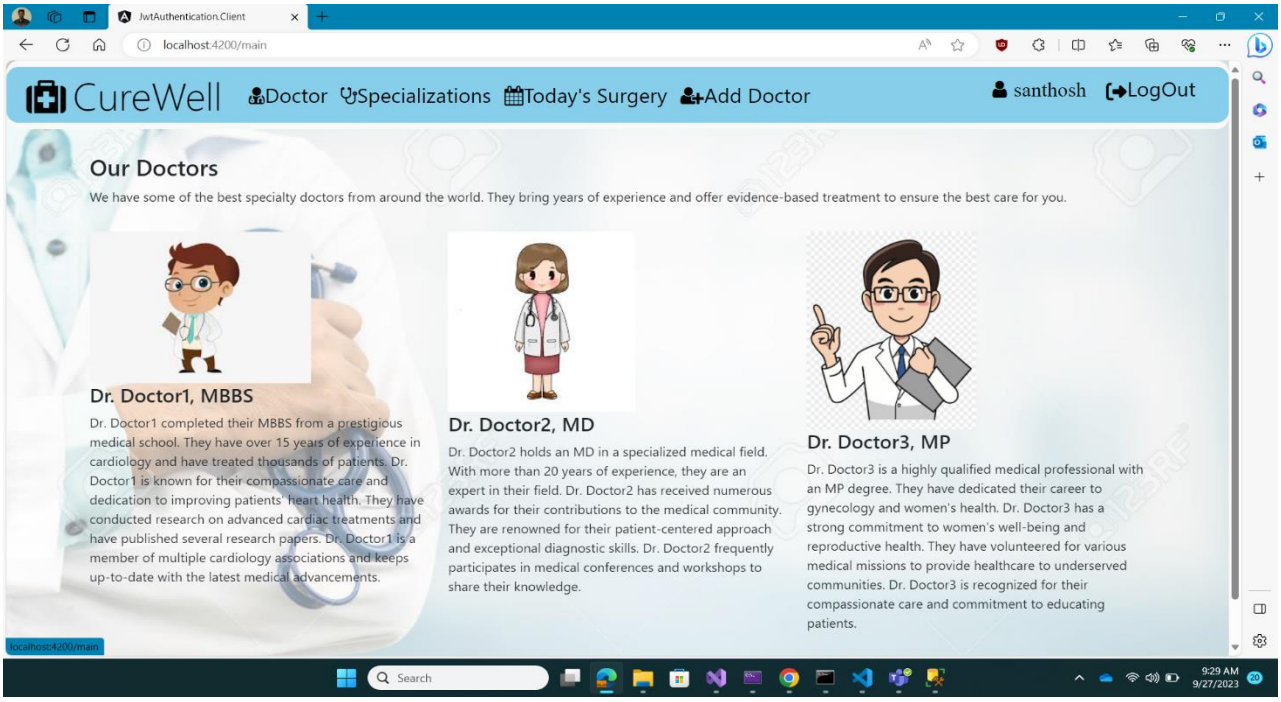


Fig 2. Home Page

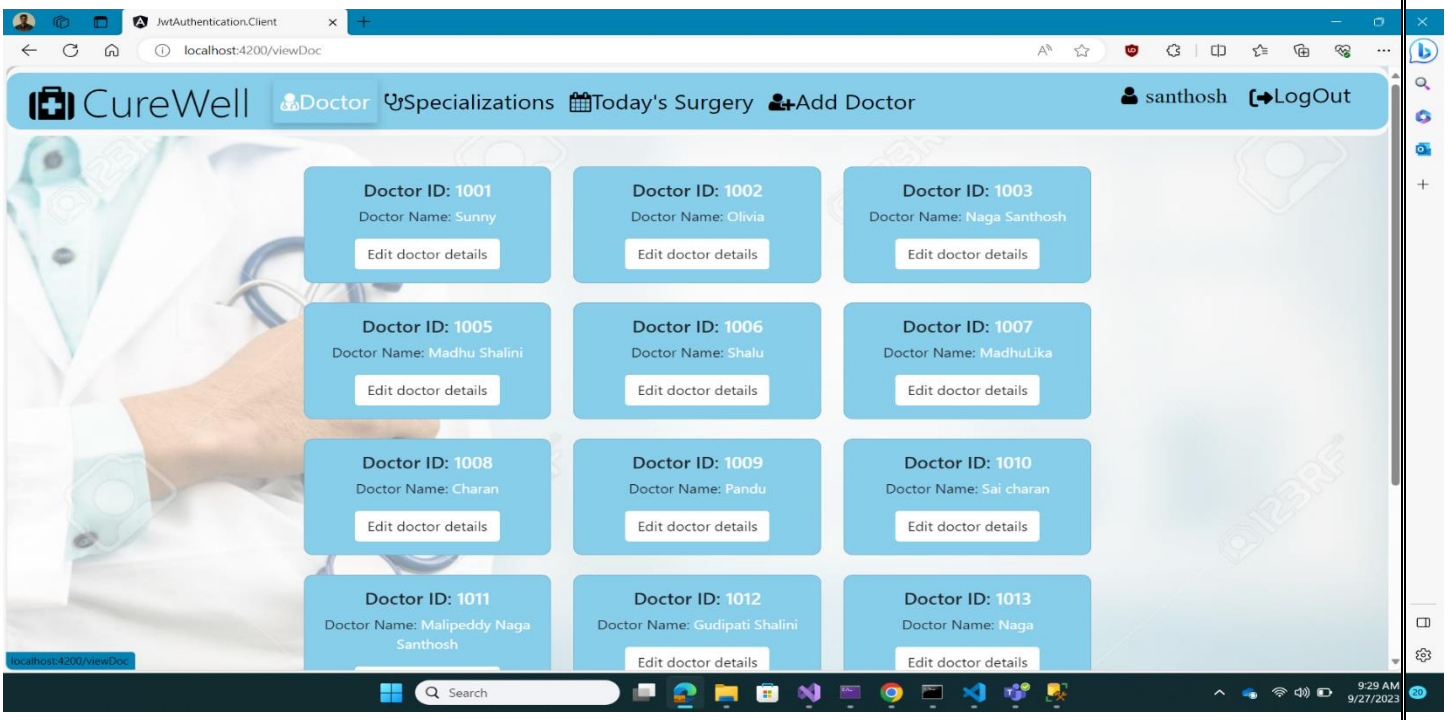


Fig 3. View Doctors Page

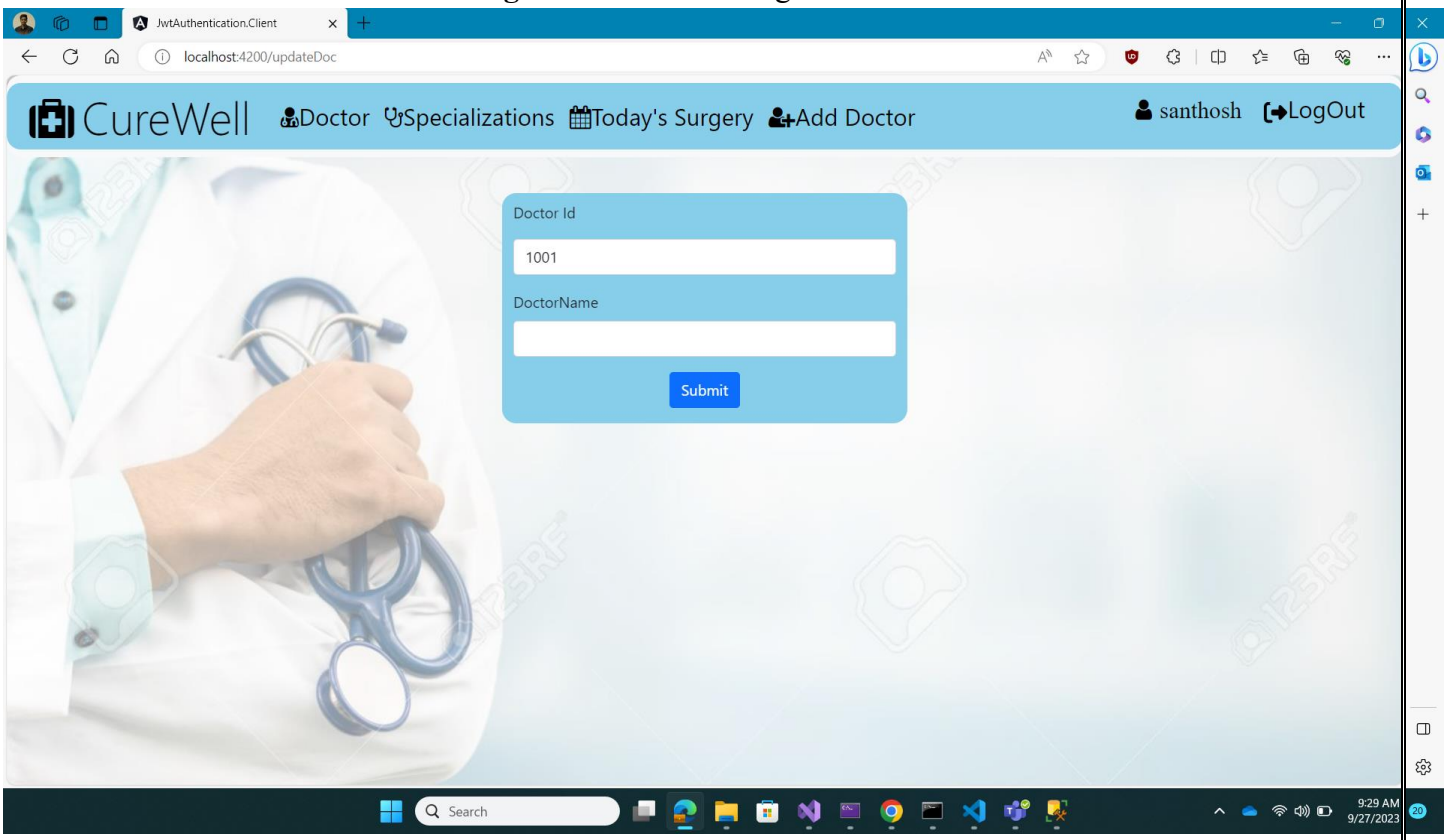


Fig 4. Edit Doctor Details

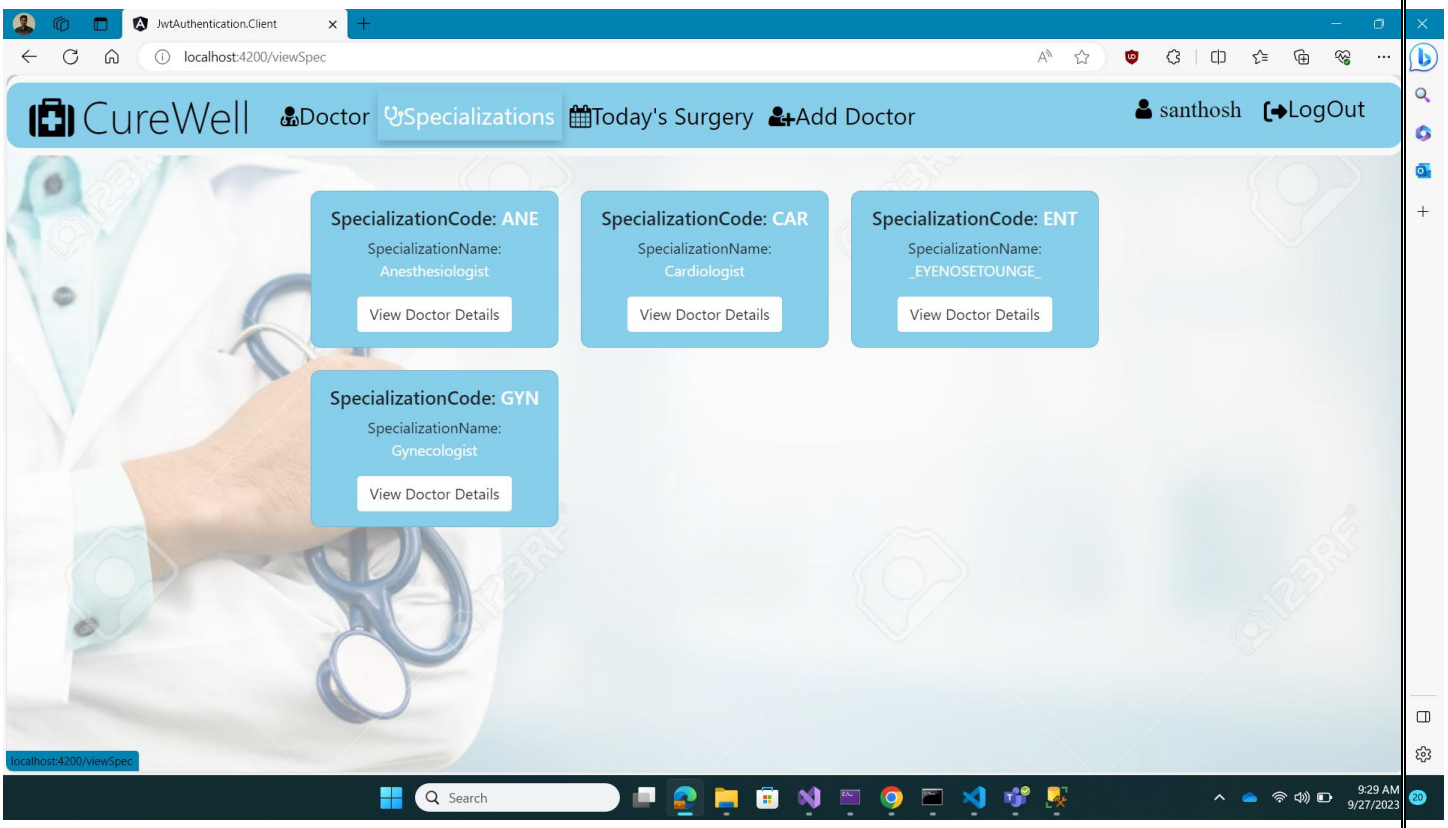


Fig 5. View Specializations Page

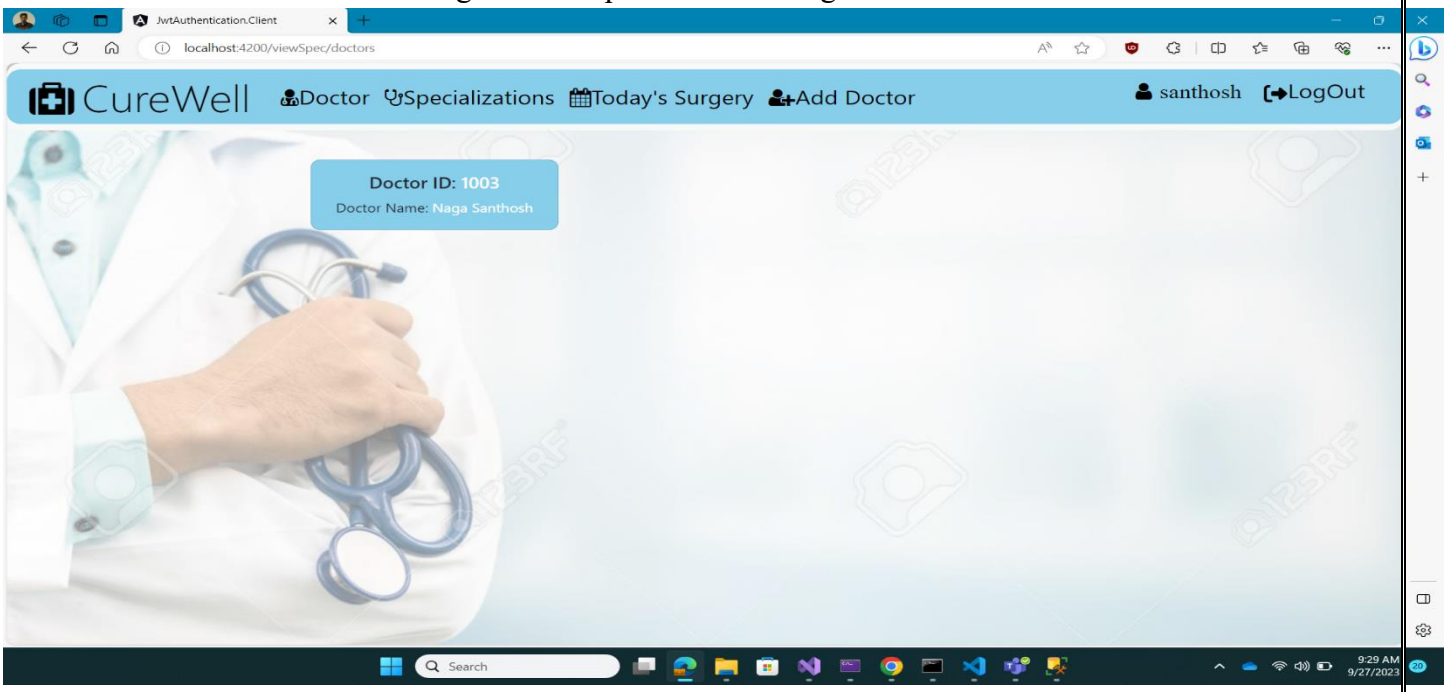


Fig 6. Doctors of Specific Specializations

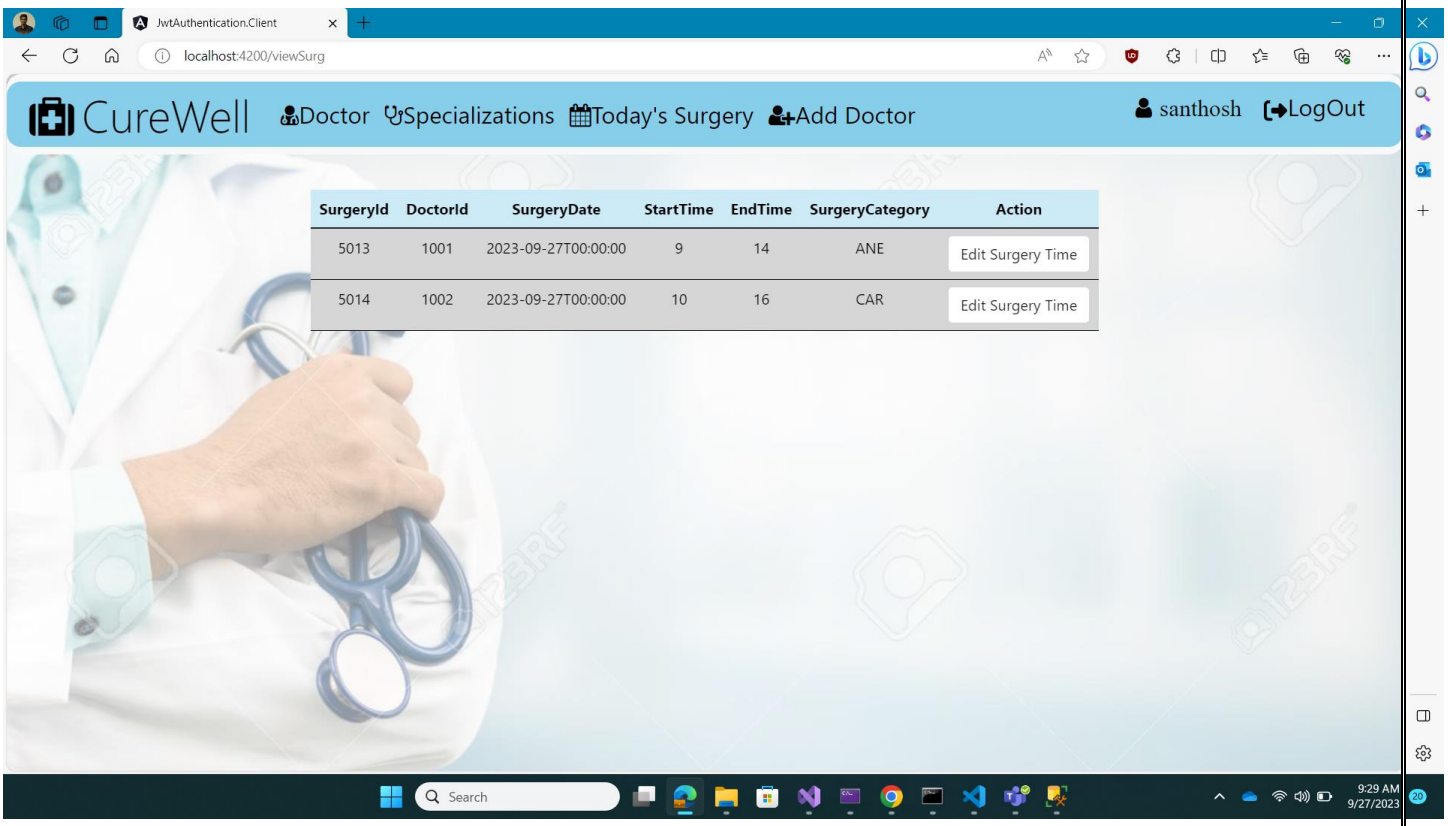


Fig 7. View Surgery

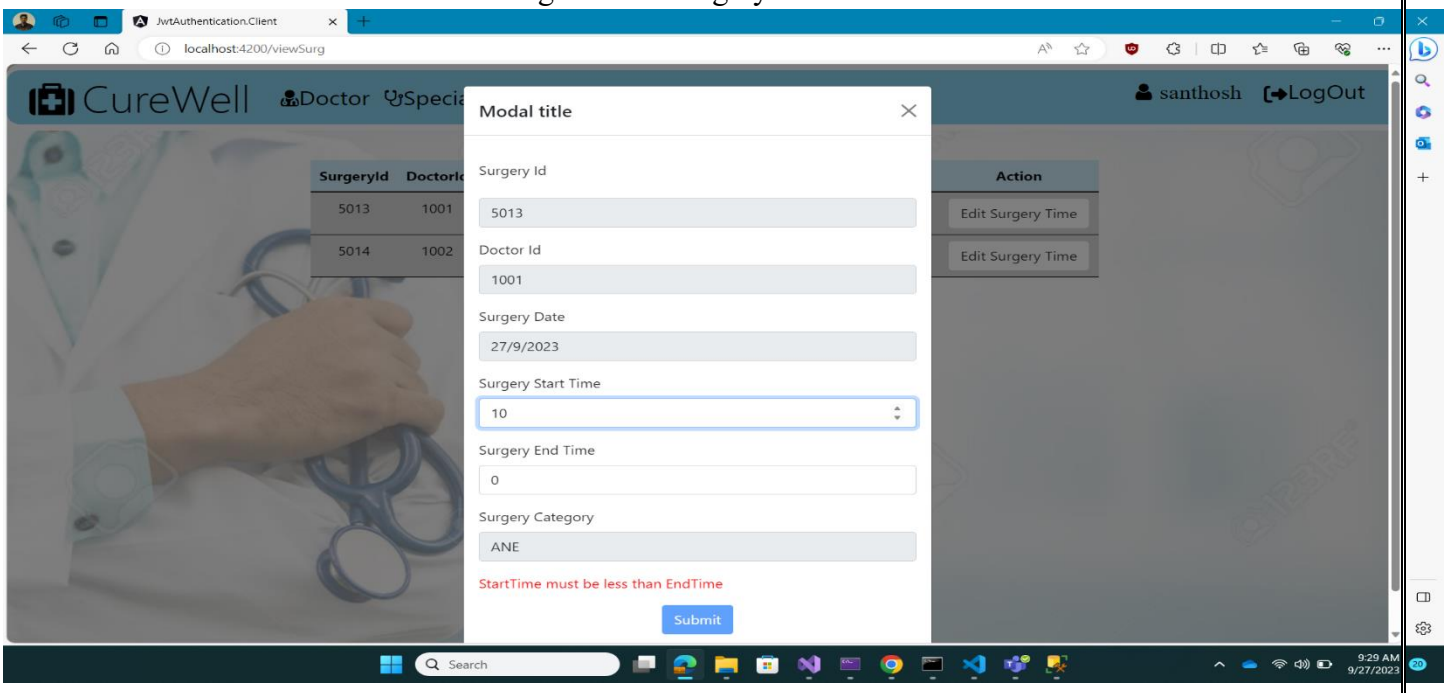


Fig 8. Update Surgery

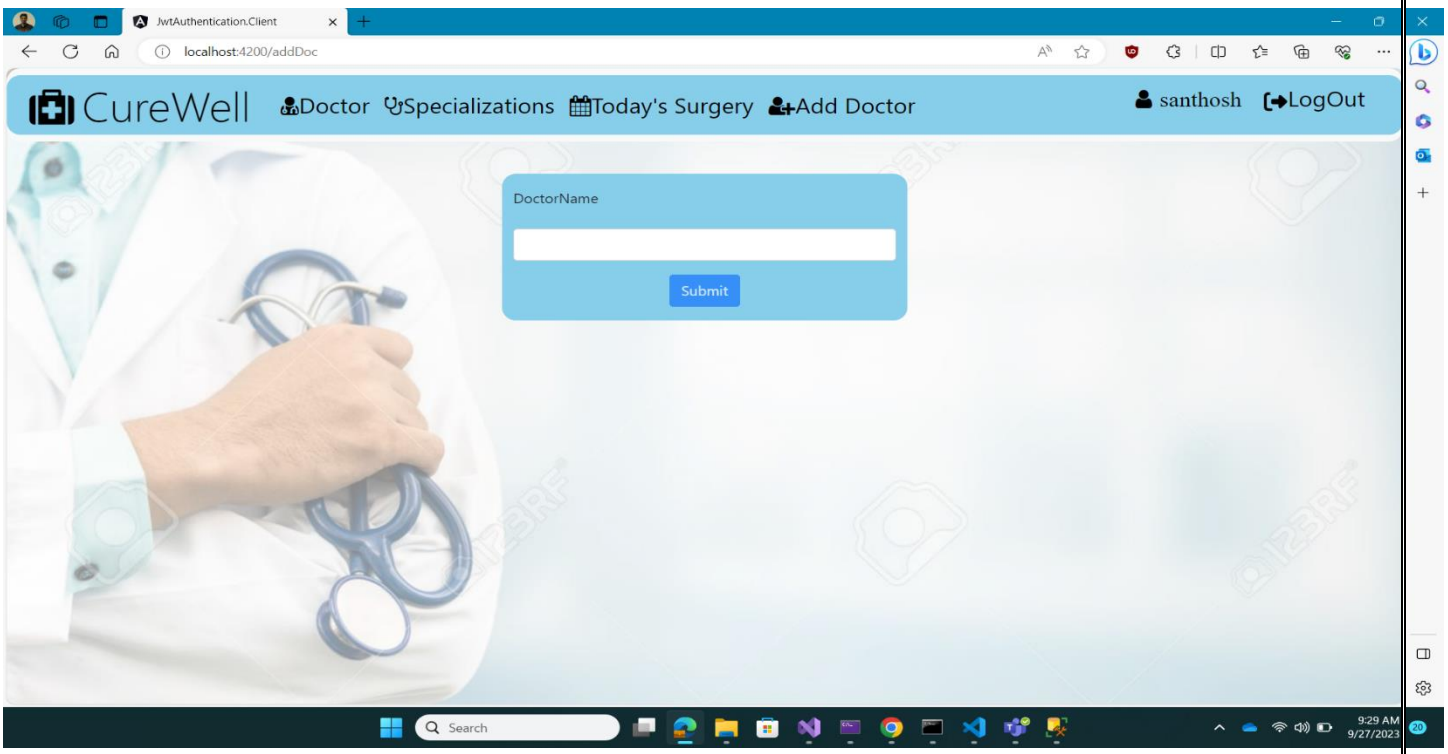
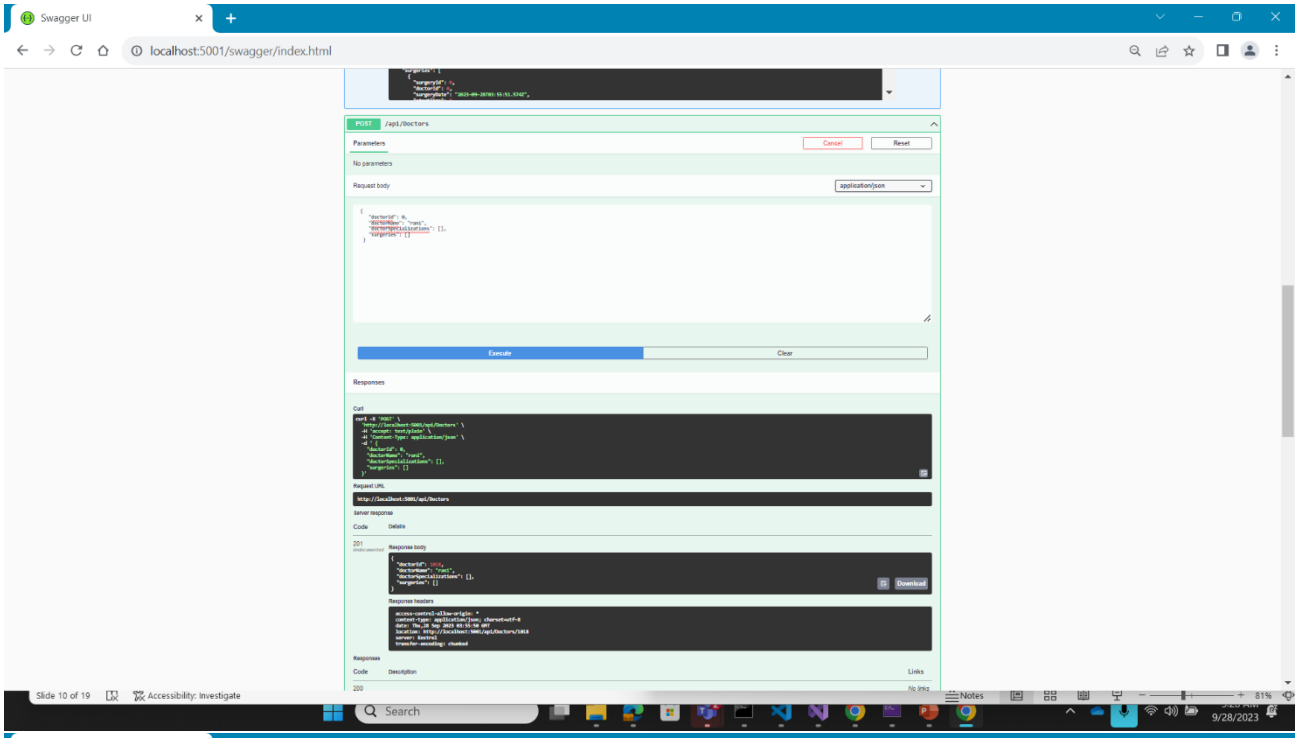
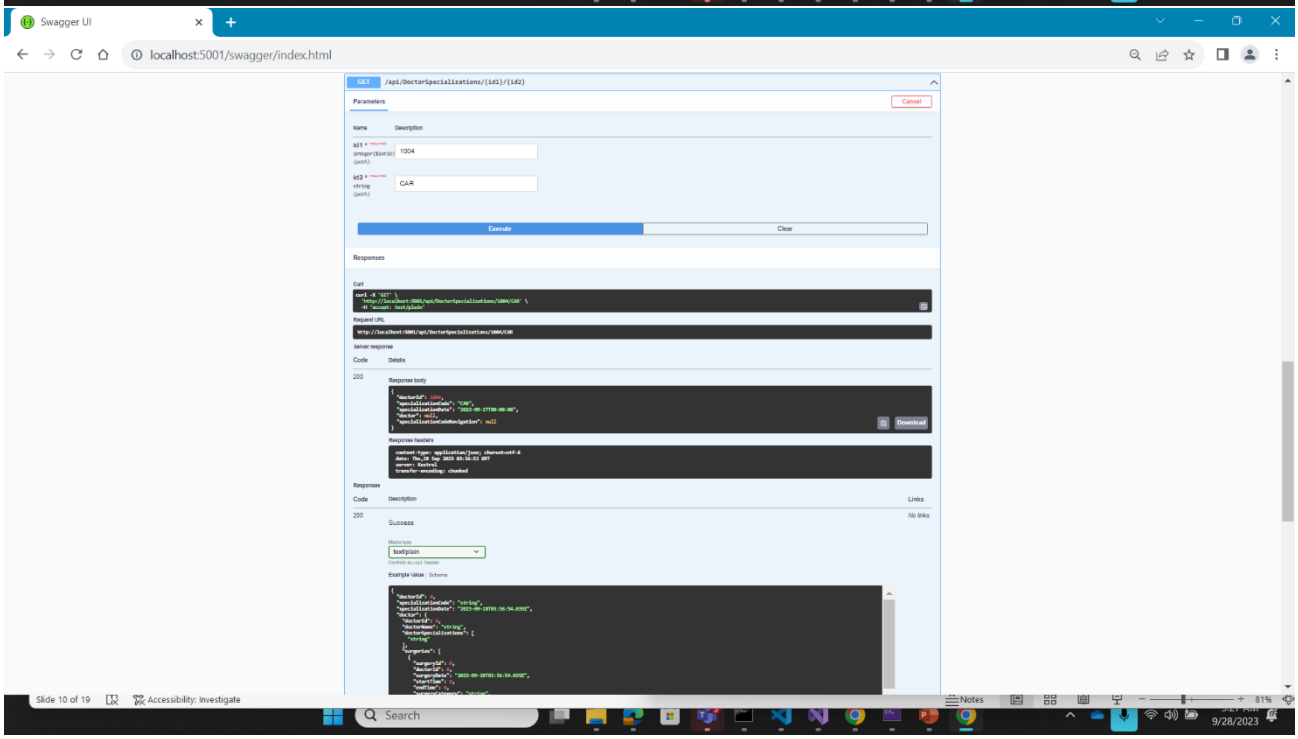
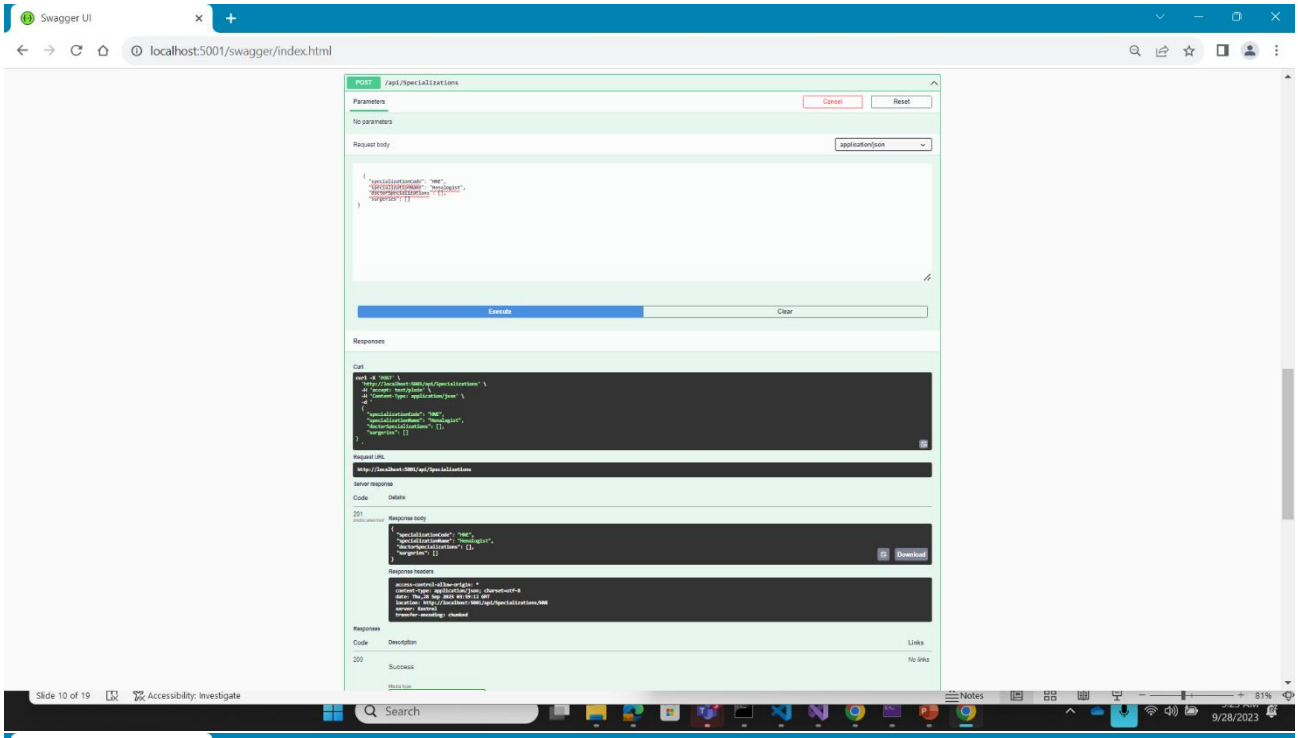
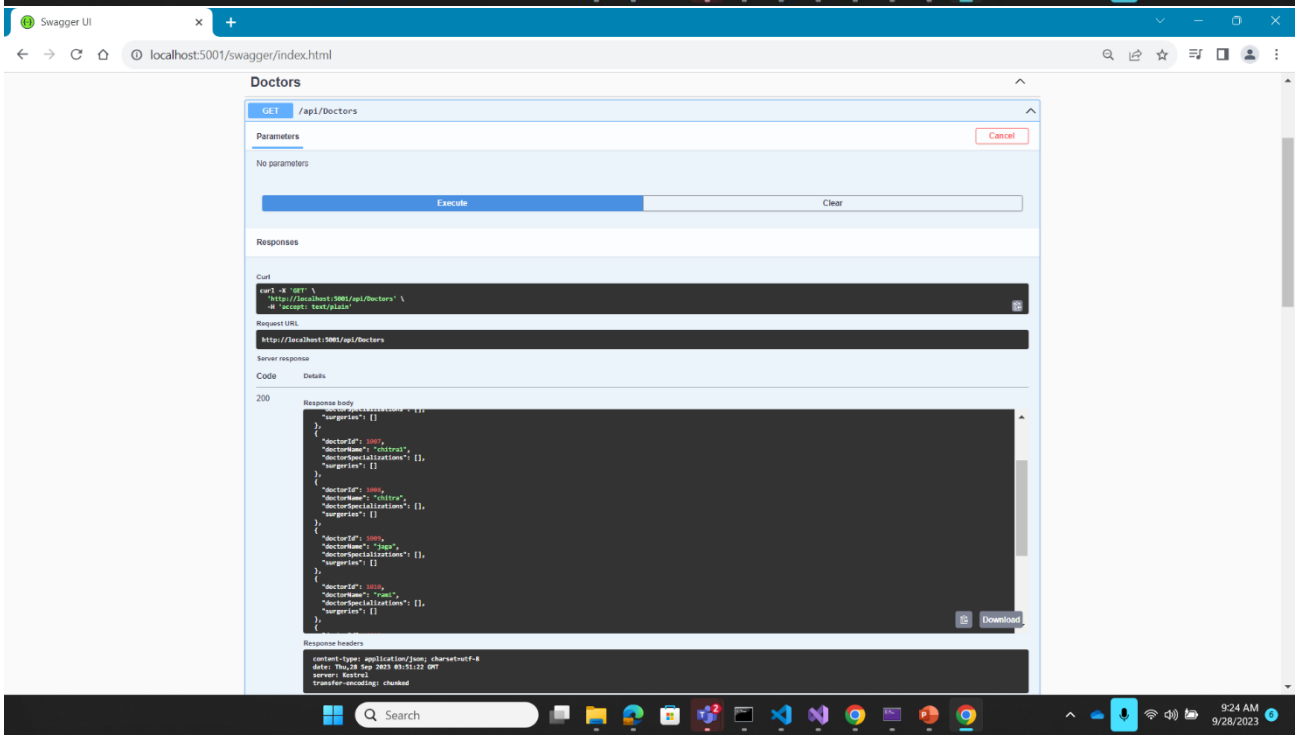


Fig 9. Add Doctor Page


Swagger UI
localhost:5001/swagger/index.html
POST /api/Doctors
Parameters
Cancel
Reset
No parameters
Request body
application/json
Execute
Clear
Responses
Curl
Request URL
http://localhost:5001/api/Doctors
Server response
Code
Details
Response body
Download
Response headers
Responses
Code
Description
Links
No links
Slide 10 of 19
Accessibility: Investigate
Notes
Search
9/28/2023

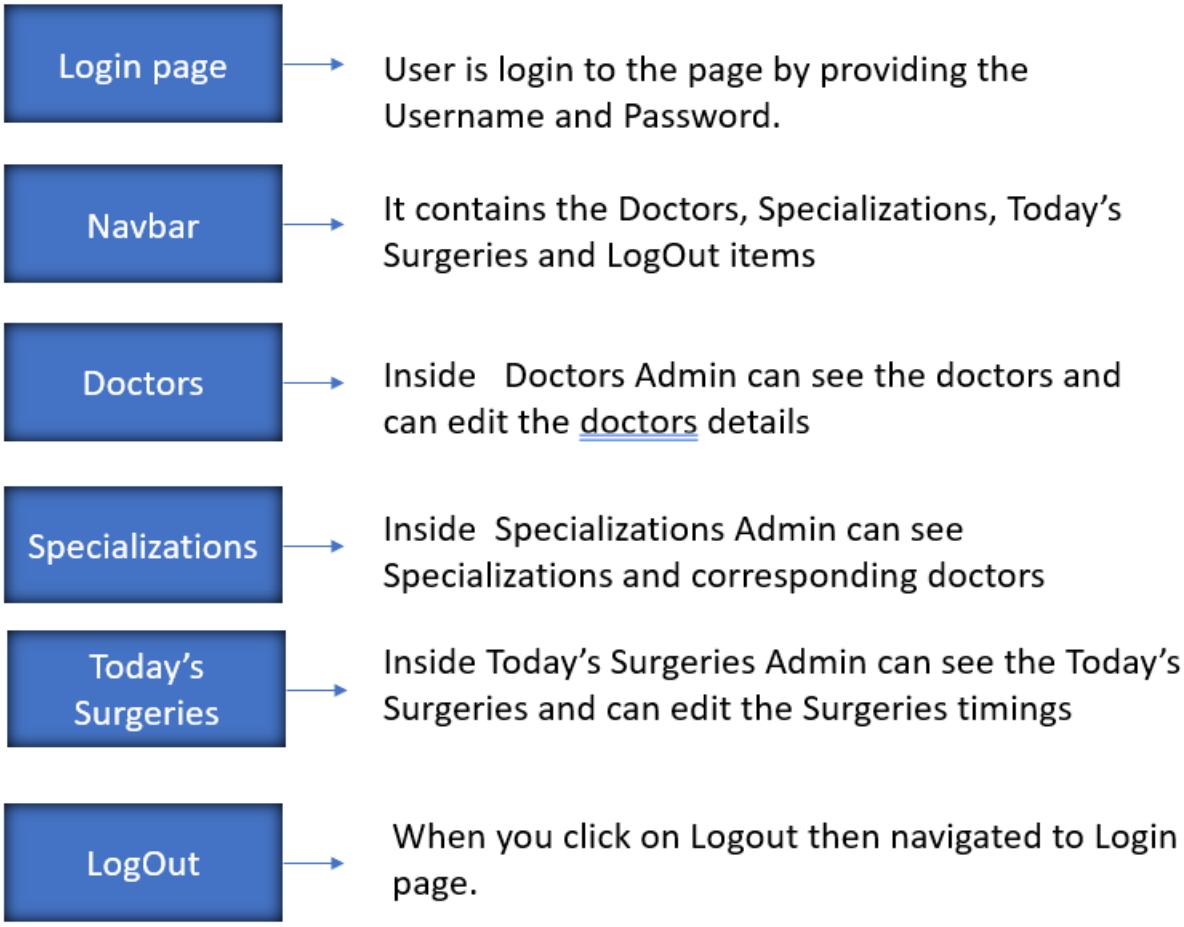

Swagger UI
localhost:5001/swagger/index.html
GET /api/DoctorSpecializations/{id1}/{id2}
Parameters
Cancel
Name
Description
id1
1004
id2
CAR
Execute
Clear
Responses
Curl
Request URL
http://localhost:5001/api/DoctorSpecializations/1004/CAR
Server response
Code
Details
Response body
Download
Response headers
Responses
Code
Description
Links
No links
Success
text/plain
Example Value
Schema
Slide 10 of 19
Accessibility: Investigate
Notes
Search
9/28/2023


Swagger UI
localhost:5001/swagger/index.html
POST /api/Specializations
Parameters
Cancel
Reset
No parameters
Request body
application/json
Execute
Clear
Responses
Curl
Request URL
http://localhost:5001/api/Specializations
Server response
Code
Details
Response body
Download
Response headers
Responses
Code
Description
Links
200
Success
No links


Swagger UI
localhost:5001/swagger/index.html
Doctors
GET /api/Doctors
Parameters
Cancel
No parameters
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://localhost:5001/api/Doctors' \
-H 'accept: text/plain'
Request URL
http://localhost:5001/api/Doctors
Server response
Code
Details
200
Response body
"doctorId": 1007,
"doctorName": "chitrai",
"doctorSpecializations": [],
"surgeries": []
"doctorId": 1005,
"doctorName": "chitra",
"doctorSpecializations": [],
"surgeries": []
"doctorId": 1009,
"doctorName": "jaga",
"doctorSpecializations": [],
"surgeries": []
"doctorId": 1010,
"doctorName": "rami",
"doctorSpecializations": [],
"surgeries": []
Download
Response headers
content-type: application/json; charset=utf-8
date: Thu,28 Sep 2023 03:51:22 GMT
server: Kestrel
transfer-encoding: chunked

CHAPTER 7
USER GUIDE

- **The user logging in into the application.** The admin logins into the application using his own credentials and use the required services provided by the application.
- **The frontend calls the API to make the request.** This is done by sending an HTTP request to the API endpoint.
- **The API verifies the connection with the database and requests the data.** This is done by executing a SQL query against the database.
- **The database fetches the data and returns it to the API.**
- **The API returns the data to the frontend.** This is done by sending an HTTP response back to the frontend.
- **The frontend displays the data to the user.** This could be done by rendering a list of items, displaying a product page, or populating a search results page.


Login page
User is login to the page by providing the Username and Password.
Navbar
It contains the Doctors, Specializations, Today's Surgeries and LogOut items
Doctors
Inside Doctors Admin can see the doctors and can edit the doctors details
Specializations
Inside Specializations Admin can see Specializations and corresponding doctors
Today's Surgeries
Inside Today's Surgeries Admin can see the Today's Surgeries and can edit the Surgeries timings
LogOut
When you click on Logout then navigated to Login page.

CHAPTER 8
MAINTENANCE AND TROUBLESHOOTING

Maintenance and troubleshooting in the Curewell Hospital Management project are essential to ensure the system's ongoing reliability, performance, and security. Here are some key aspects of maintenance and troubleshooting for this project:

Maintenance:

Regular Updates: Maintain and update the system to address bugs, security vulnerabilities, and performance enhancements. This includes applying patches and updates to the underlying software and frameworks.

Data Management: Continuously manage and optimize the database to ensure data integrity, availability, and performance. This includes regular data backups and archiving of old data.

Security Audits: Conduct regular security audits to identify vulnerabilities and ensure that data remains protected. Implement security patches and upgrades promptly.

User Support: Provide ongoing user support to address questions, issues, and training needs. This may involve a helpdesk or support team.

Performance Monitoring: Continuously monitor system performance, including response times, server resource usage, and database performance. Optimize performance as necessary.

Scalability: Assess the system's scalability as hospital operations grow. Ensure that the infrastructure can handle increased data, users, and workload.

Backup and Disaster Recovery: Regularly test backup and disaster recovery procedures to ensure that critical data can be restored in the event of data loss or system failure.

Compliance Updates: Stay up to date with changes in healthcare regulations and compliance requirements. Modify the system and documentation as needed to remain compliant.

Troubleshooting:

Issue Tracking: Implement a robust issue tracking system to log and prioritize reported problems and bugs.

Root Cause Analysis: When issues arise, perform root cause analysis to identify the underlying reasons for the problem. This helps prevent recurring issues.

Testing Environments: Maintain separate testing and staging environments to replicate and troubleshoot issues without affecting the live system.

User Communication: Communicate with users about known issues, their resolutions, and estimated timelines for fixes.

Security Incidents: Respond to security incidents promptly, including breaches or unauthorized access. Investigate the incident, remediate vulnerabilities, and report as required by regulations.

Performance Issues: Troubleshoot and resolve performance problems, such as slow response times, by identifying bottlenecks and optimizing code or resources.

Data Integrity: Regularly validate data integrity to identify and correct data inconsistencies or corruption.

User Errors: Address user errors and issues, such as forgotten passwords or difficulties navigating the system, through support and training.

Integration Problems: Troubleshoot issues related to system integration with other healthcare systems, ensuring data flows smoothly between systems.

Emergency Response: Develop and practice emergency response procedures to ensure the system's availability during critical situations.

Version Control: Ensure that system components and software are kept up to date and aligned with each other. Incompatibilities between components can cause issues.

Documentation Updates: Update documentation as needed to reflect changes and issue resolutions. Well-documented solutions can aid in future troubleshooting.

CHAPTER-9
SECURITY

Implementing JWT (JSON Web Tokens) for security in your CureWell hospital management application is a solid choice. Here's some information about JWT and its role in enhancing security:

Security with JWT in CureWell Hospital Management:

JWT (JSON Web Token) is a widely adopted standard for securing web applications and APIs. It provides a stateless and secure way to transmit information between parties, such as your Angular frontend and ASP.NET backend, in a compact and self-contained manner. Here's how JWT enhances the security of your application:

Authentication: JWT allows you to authenticate users securely. When a user logs in, the server generates a JWT containing a digitally signed payload with information about the user (such as their user ID or role). This token is then sent to the client, which stores it for subsequent requests.

Stateless: One of the key advantages of JWT is its stateless nature. The token itself carries all the information needed for authentication and authorization. This eliminates the need for server-side sessions or database lookups for each request, making your application more scalable and performant.

Authorization: In addition to authentication, JWT can also carry authorization claims. You can embed user roles and permissions within the token. This allows your server to make access control decisions based on the contents of the JWT, ensuring that users only access the resources they are authorized to.

Security Tokens: JWTs are digitally signed by the server using a secret key or a public/private key pair. This signature ensures the integrity and authenticity of the token. Any tampering with the token, such as altering its payload, will result in an invalid token.

Expiration: JWTs can include an expiration time (known as "exp" claim). This adds an additional layer of security, as tokens become invalid once they expire. This reduces the window of opportunity for attackers in case a token is somehow compromised.

Cross-Origin Resource Sharing (CORS): JWTs are an effective means of implementing CORS security. They can be included in HTTP headers to allow or restrict access to resources based on the token's contents and origin.

Revocation: While JWTs are valid until they expire, there are strategies for handling token revocation if necessary. For example, you can maintain a token blacklist on the server to invalidate tokens before their expiration time.

Secure Transmission: JWTs are typically transmitted over HTTPS, providing encryption during transit to protect the token from eavesdropping.

Scalability: JWTs are highly scalable because they don't require server-side storage of session data. This makes them suitable for microservices architectures and distributed systems.

To maximize the security of your application with JWT, it's essential to implement best practices such as proper key management, token validation, and secure transmission. Additionally, keeping your application and libraries up to date to address any security vulnerabilities is crucial. Overall, JWT is a robust choice for securing your CureWell hospital management application, providing a foundation for user authentication and authorization.

CHAPTER-10
FUTURE ENHANCEMENTS

Expanding user roles beyond the current administrator role to include doctors, patients, nurses, and potentially other roles is a valuable future enhancement for your CureWell hospital management application. This expansion not only broadens the user base but also enhances the application's functionality and usefulness. Here's some context about this future enhancement:

Future Enhancement: User Roles for Doctors, Patients, and Nurses:

Enhanced User Experience: Introducing user roles for doctors, patients, and nurses allows you to provide a tailored and enhanced user experience. Each role can access features and information relevant to their responsibilities and needs. For example, doctors can have access to patient records and scheduling, patients can view their medical history and appointment details, and nurses can manage patient care tasks.

Role-Specific Permissions: Implementing role-based access control (RBAC) ensures that each user role has specific permissions aligned with their job functions. Doctors may have the ability to update patient treatment plans, while patients can view their own medical records without editing privileges. Nurses can manage patient assignments and care tasks.

Data Privacy and Security: Assigning specific roles helps maintain data privacy and security. Patient data can be protected by restricting access to authorized personnel only. Role-based permissions ensure that sensitive patient information is not exposed to individuals who do not need it for their roles.

Streamlined Workflows: Role-based access can streamline workflows within the hospital. For example, doctors can electronically prescribe medications and treatments, while nurses can access these orders and administer them. Patients can use the system to schedule appointments and access their health records.

Comprehensive Reporting: Different roles generate and rely on different types of reports. Doctors may need reports on patient outcomes, nurses on medication administration, and administrators on overall hospital performance. Role-based access allows for custom reporting tailored to each role's requirements.

Scalability: As your hospital grows, having user roles for different healthcare professionals and patients ensures scalability. New personnel can be easily onboarded with role-specific access and permissions.

Patient Engagement: For patients, providing role-specific access allows them to actively engage in their healthcare. They can view lab results, appointment schedules, and communicate securely with their healthcare providers.

Customized Dashboards: Role-specific dashboards can provide at-a-glance information relevant to each user role. Doctors might see patient summaries and upcoming surgeries, while patients can access their personal health records.

Future-Proofing: Anticipating the need for additional roles (e.g., specialists, administrative staff) demonstrates forward thinking and ensures that the application remains adaptable to future healthcare trends and staffing requirements.

To implement this future enhancement, you'll need to define the specific permissions, responsibilities, and access levels for each user role, as well as create user interfaces and workflows tailored to their needs. It's also important to consider how user roles will interact within the system and how data will be shared securely among them.

By introducing user roles for doctors, patients, and nurses, your CureWell hospital management application can become a more comprehensive and versatile tool for healthcare professionals and patients, ultimately enhancing the quality of care and the efficiency of hospital operations.

CHAPTER 11
CONCLUSION

The Curewell Hospital Management project is a comprehensive and ambitious initiative designed to revolutionize healthcare operations by implementing a robust web application designed for the healthcare industry. This project aims to streamline and enhance various aspects of hospital management, from doctor information and surgery scheduling to data security and patient care .