



Secure HealthPath: HIPAA Edition

Students: Jason Kildare, Mohammad Kazmi
Mentor: Wenjia Wang, PhD, Product Owner
Instructor/Faculty: Dr. Masoud Sadjadi, Florida International University

PROBLEM

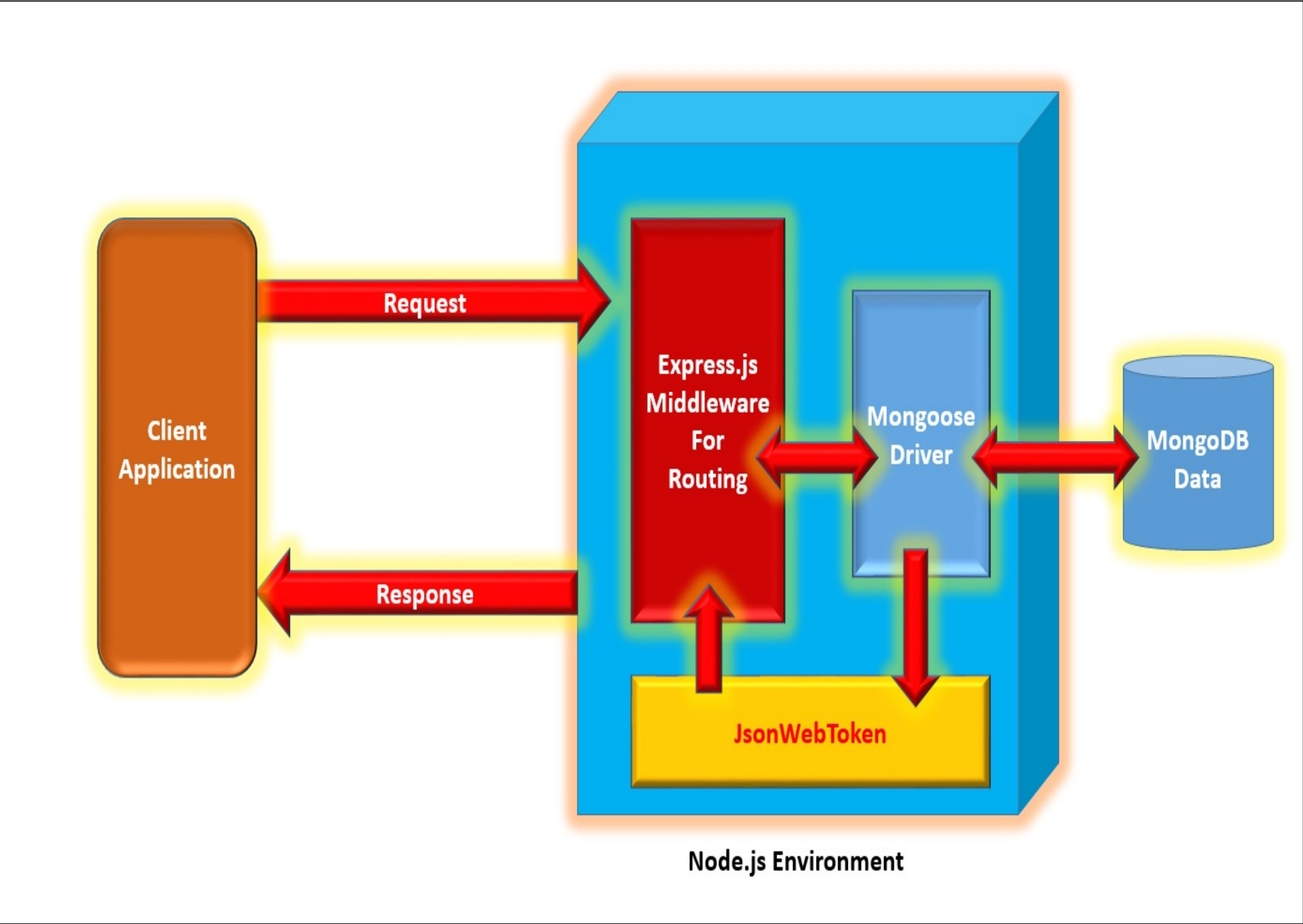
Secure HealthPath was initially introduced as a web application that did not have complete functionality and was not secure under HIPAA compliance.

When dealing with sensitive patient information, it is important to ensure that the protection of patient data and their user account remain secure and private from unauthorized personnel. Examples that elevate additional protection include data encryption, two factor authentication and other security measures. These are some essential ways in securing an application related to personal health related information

To solve this problem, our team implemented security measures related to HIPAA to help guide us to build a more secure application.

SYSTEM DIAGRAM

The three-tier system diagram shows the MERN Stack model utilizing, MongoDB, Express, React, and Node. The system beginnings with the client-side application which sends requests to the server within a NodeJS runtime environment to interact with the MongoDB database. When a request is successful or not successful, the client will receive a response from the server.



SUMMARY

In the initial stages, Secure HealthPath faced challenges, marked by incomplete functionality and security vulnerabilities. Our team responded with determination and implemented significant changes, steering the application towards HIPAA compliance. We reinforced the system with security and breach notification rules, addressing the crucial aspect of data protection. The introduction of advanced encryption and decryption protocols played a pivotal role in securing sensitive patient information. Our overall goal was to achieve an application that was HIPAA compliant and user-friendly within an interactive platform, achieving the standard for healthcare applications.

CURRENT SYSTEM

- The features in the Secure HealthPath project that were either partially or fully completed by our mentor are as follows:
- React frontend library, Node, Express, and other dependencies
 - Mongo DB database with varies collections to store user, doctor, and patient data
 - React components, routes, database models, react database configuration, middleware, Redux for state management, server and client-side code
- Features completed by our team including by not limited to:
- Audit logs of all accesses and changes to electronic health information 164.312(b) (Security Rule)
 - Automation notification system to notify the user in the event of a breach. 164.404(a) (Breach Notification Rule)

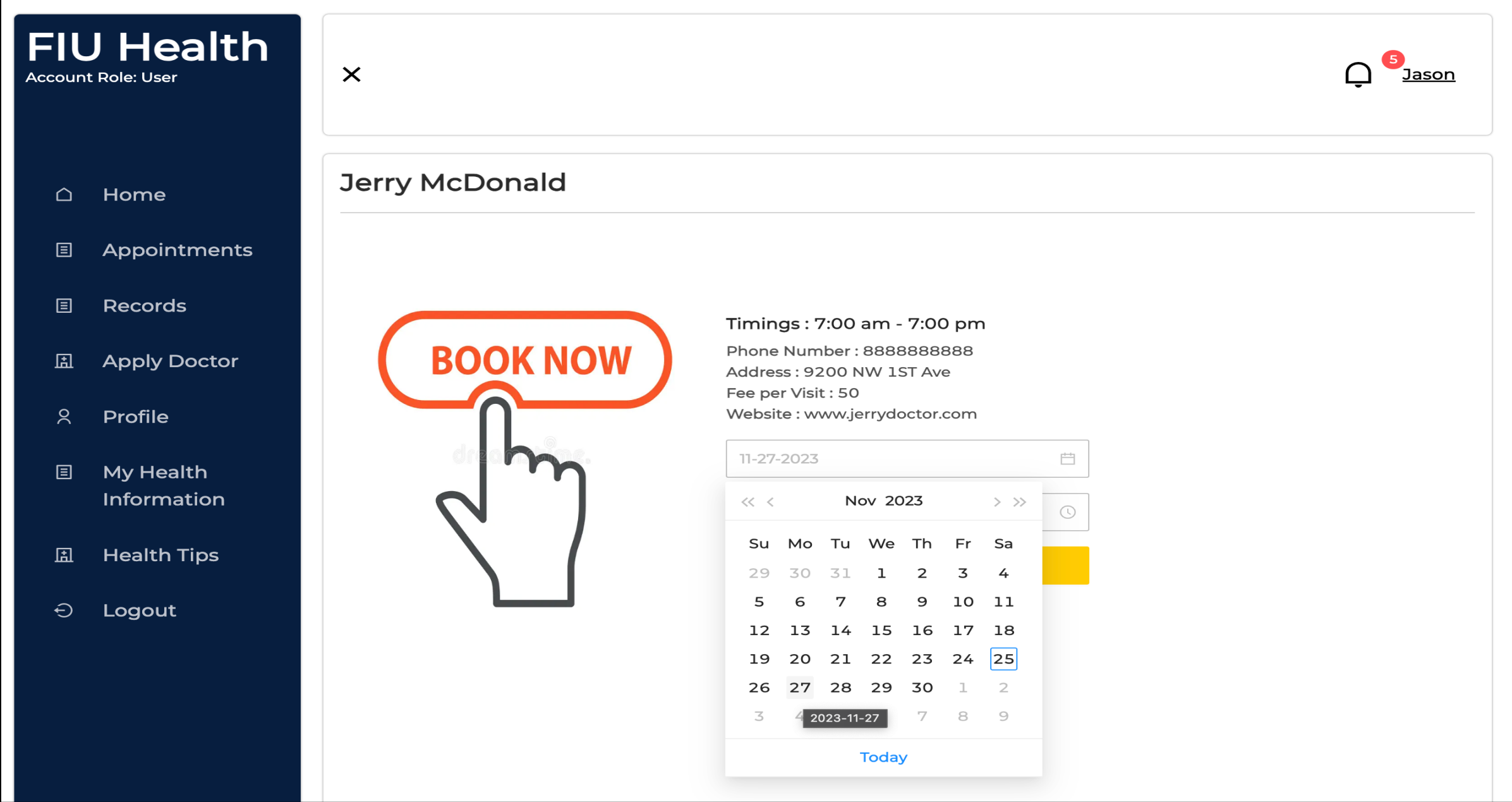
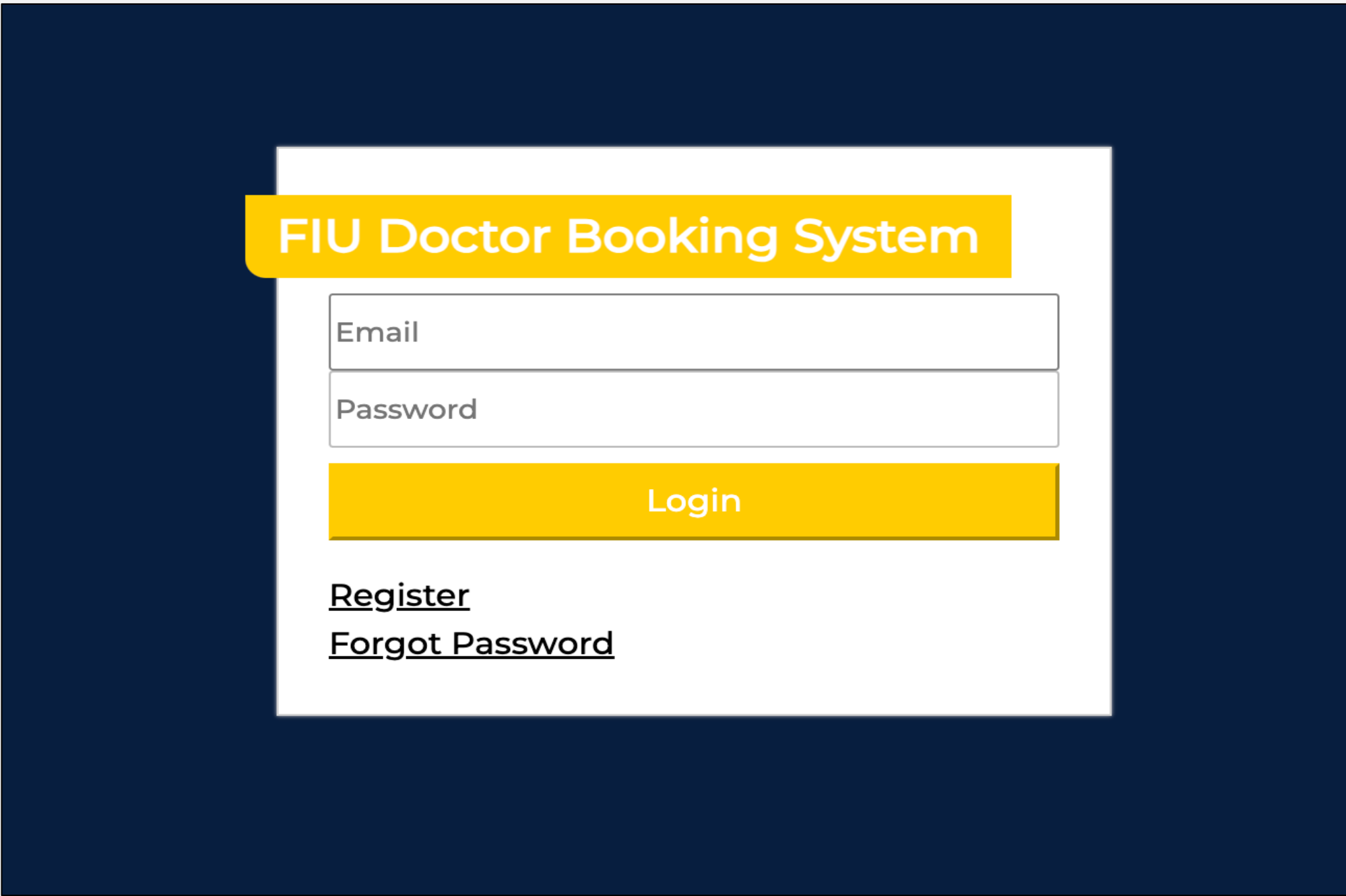
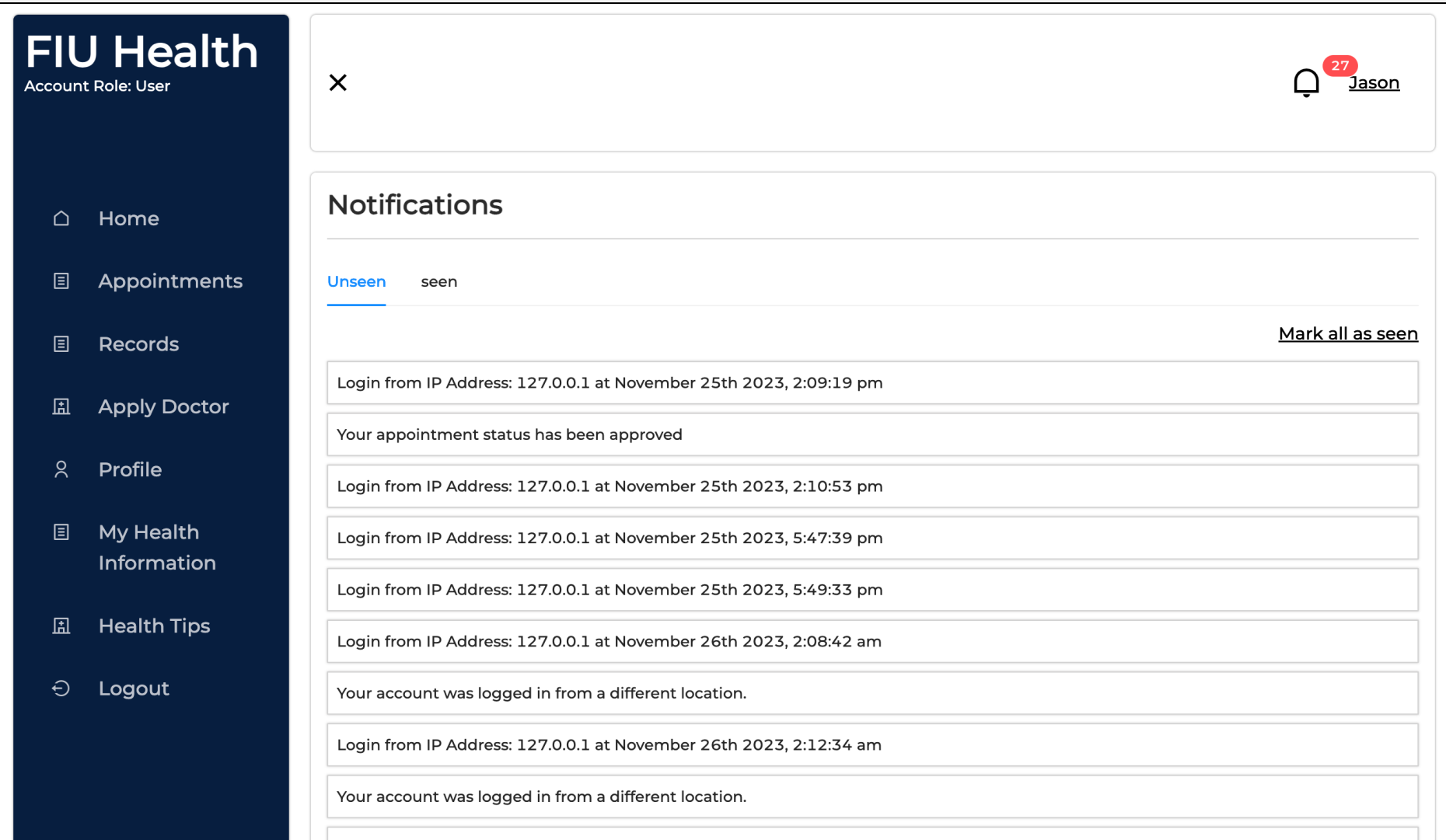
REQUIREMENTS

- The goal of this project was to make it fully functional by creating a secure system and adding additional dynamic features such as:
- Notification system 164.312(b)(Security Rule)
 - Filtering system to discover doctors based on specialization
 - Two-factor authentication
 - Terminate electronic sessions after a predetermined time of inactivity 164.312(a)(2)(iii)(Security Rule)
 - Prohibit users from reusing previous passwords 164.308(a)(5)(ii)(D)(Security Rule)
 - Schedule or cancel appointments
 - Application vulnerability scanner 164.308(a)(1)(ii)(A)(Security Rule)
 - Strong password feature
 - Account deactivation 164.308(a)(3)(ii)(C)(Security Rule)
 - Account recovery
 - Accept / deny changes to health information
 - Provide / update personal health information
 - Data encryption / decryption 164.312.(a)(2)(iv) (Security Rule)

IMPLEMENTATION



SCREENSHOTS



ACKNOWLEDGEMENT