CSC3131 Building Systems for People: Storyboard for e-clinic web application built with React, ASP.Net, MongoDB, Docker, and other dev-ops tools

Introduction

E-clinic is a clinic management web application, built for patients and medical service providers such as GP clinics. It will be developed using React, ASP.Net, MongoDb, and Docker, as well as other dev-ops tools such as unit testing and CI/CD.

There is role-based access control. Doctors and patients will be provided with different web pages depending on their account roles.

Personas

- The client will be a medical service provider, such as GP clinics. Clinic management software is commonplace and essential in current medical industry, and helps doctors manage appointments, keep logs of medications and past diagnoses, facilitate communication between doctors and patients, etc. The client has requested an easy to use, lightweight web application that is rapidly scalable and hence requires continuous integration and deployment.
- Patients can range from any gender to age range, and hence it is required that the application is easy to use and intuitive. Accessibility concerns for patients are also essential. For example, the text should be easy to read for older patients, and there should be good support for text-to-speech readers for visually impaired users. Following the Web Content Accessibility Guidelines (https://www.w3.org/WAI/standards-guidelines/wcag/) will be essential to ensure that the software is open to all types of user personas.
- Clinic staff such as doctors and nurses will be familiar with basic usage such as answering patients messages, seeing appointment history. The design has to be intuitive and easy to use, as hospitals and clinics can often be understaffed and requires fast turnaround times for patients. Easy to use software are essential to facilitating such aims. Other clinic staff such as hospital receptionists will need access to the web application, for administration such as adding new clients, changing user details such as address, etc.
- As the client has requested an easily scalable web application, it is important for the developers use agile related methodology such as continuous integration and deployment, unit testing, etc. It will also be important that the implementation and the codes are well documented for future scalability.

Design decisions

Figure 1: Overall storyboard

As the web application requires role based access control, only the login screen and the message screen is the same for both staff and patients. The homepage and appointment pages for the users are personalized to each user roles.

The overall design uses bootstrap components for ease of development of the front-end. Reusable components by React also ensures that design concerns such as spacing, fonts, etc remain the same throughout the website.

I have put emphasis on readability as well as ease of use, to accommodate numerous types of users, including users who require accessibility such as visually impaired users where large fonts and sufficient contrast between text and background is essential.

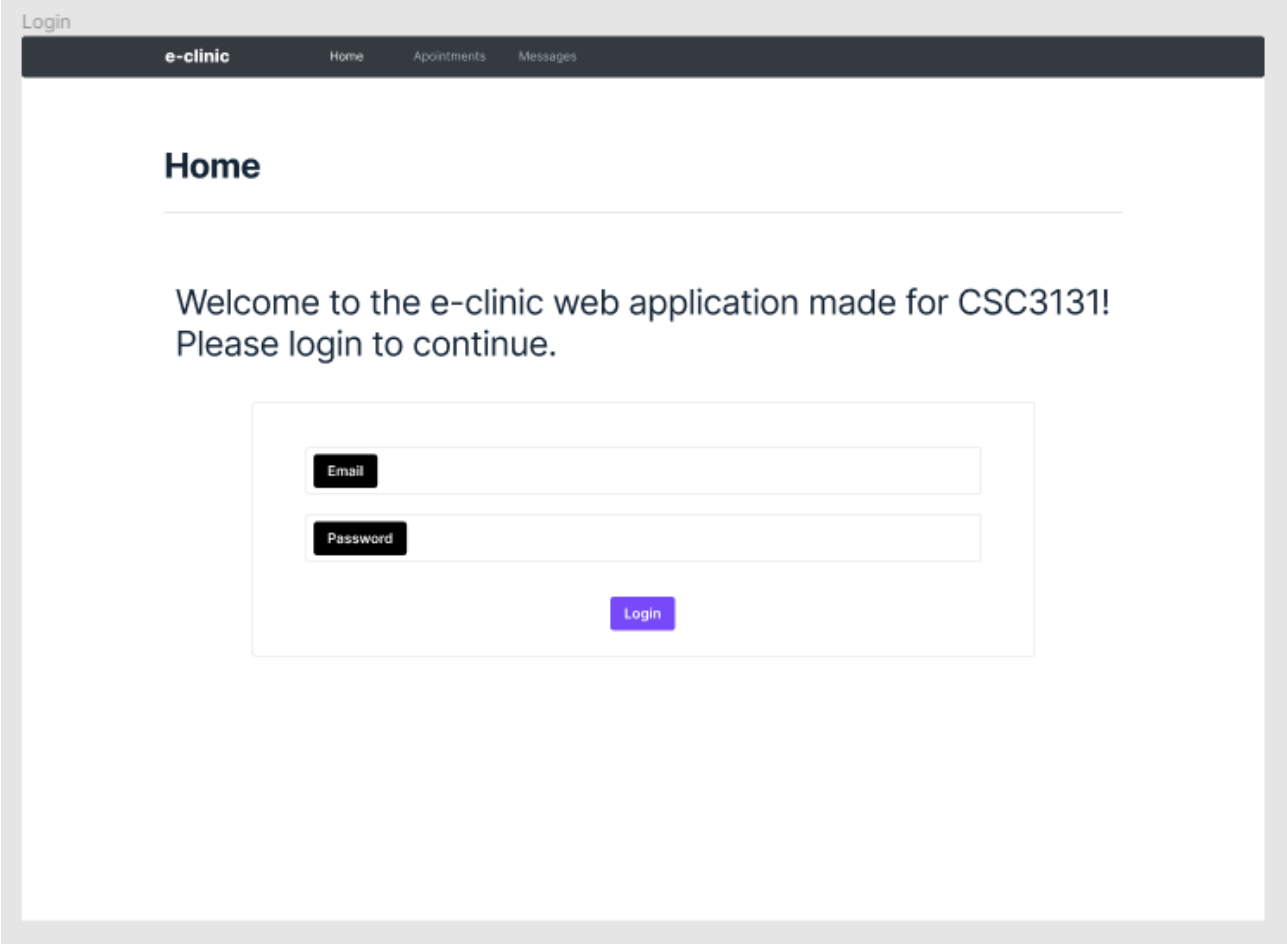

Figure 2: Login screen

Users will be first greeted with a login page. From here, depending on the roles, different versions of the website will be shown depending on if they are staff or patients. Sign-up functionality has been omitted as it is assumed that login information for new users will be given by the clinic reception to ensure that only authorized users can use the clinic software.

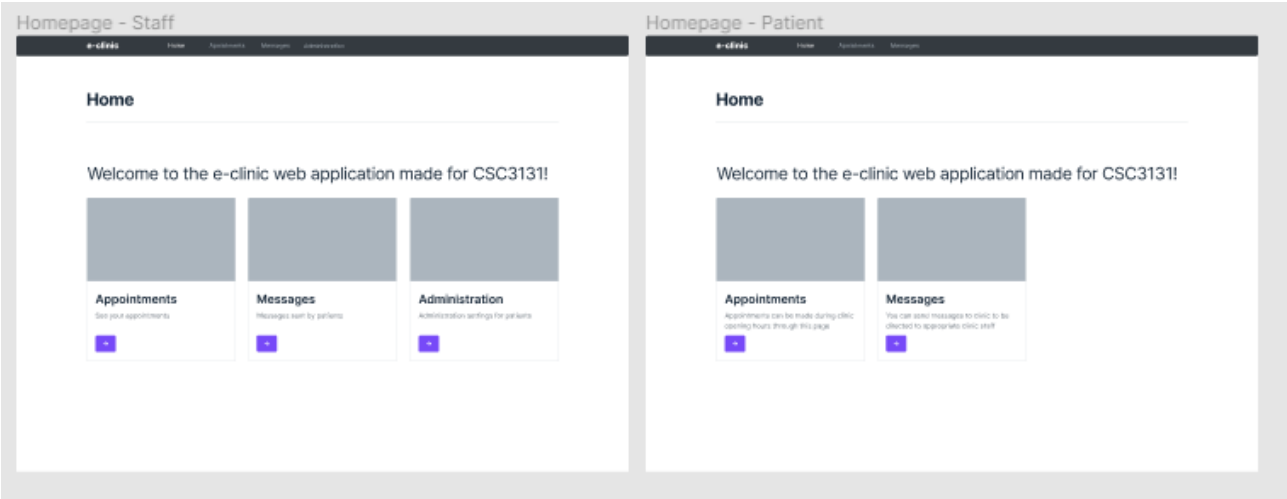

Figure 3: Website homepage

After the login, user is directed to the homepage. It includes links to different site functionality for ease of use. For the staff, administration page has been added.

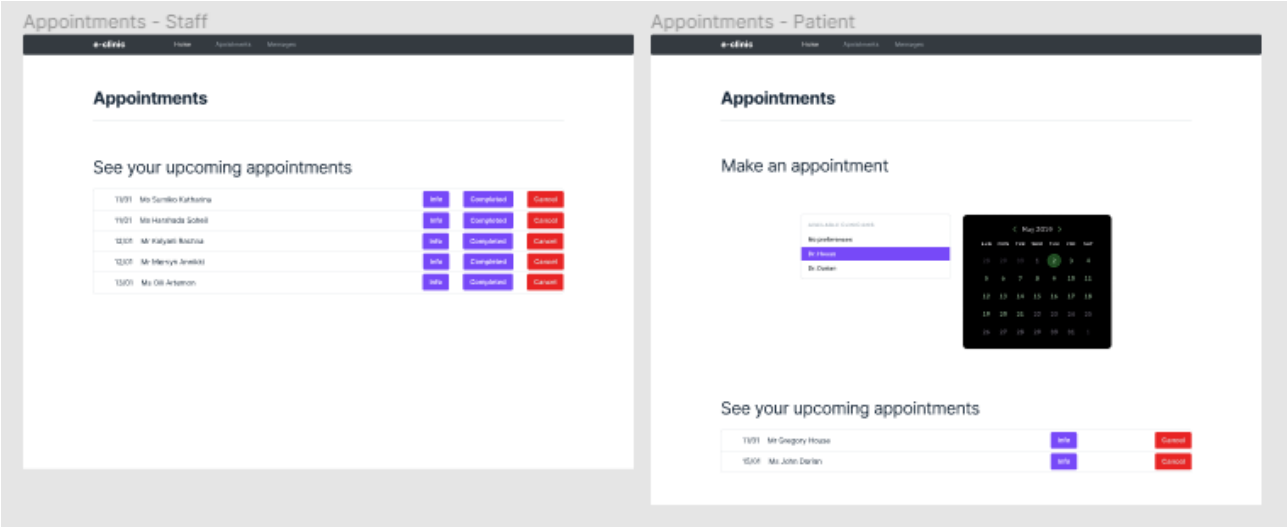

Figure 4: Appointments screen

Both staff and patients can see upcoming appointments, but there is an added functionality for the patient to request new appointments. When making a new appointment, the patient can choose available doctors and their respective available days from the calender to create the appointment. Staff are able to view information about the appointment, mark the appointment as complete to clear it from the list, as well as cancel the appointment.

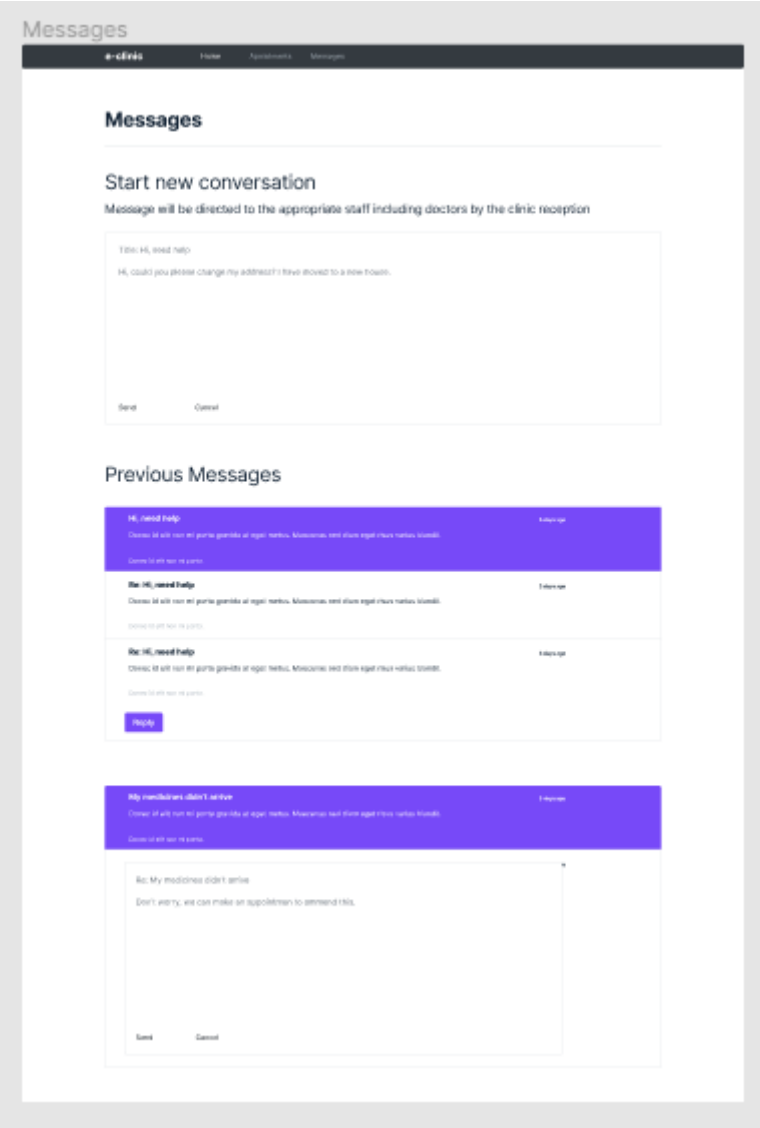

Figure 5: Messages screen

The message screen is same for both staff and patients. The message screen allows messages to be sent to any user in the system, as well as view previous messages and reply to users.

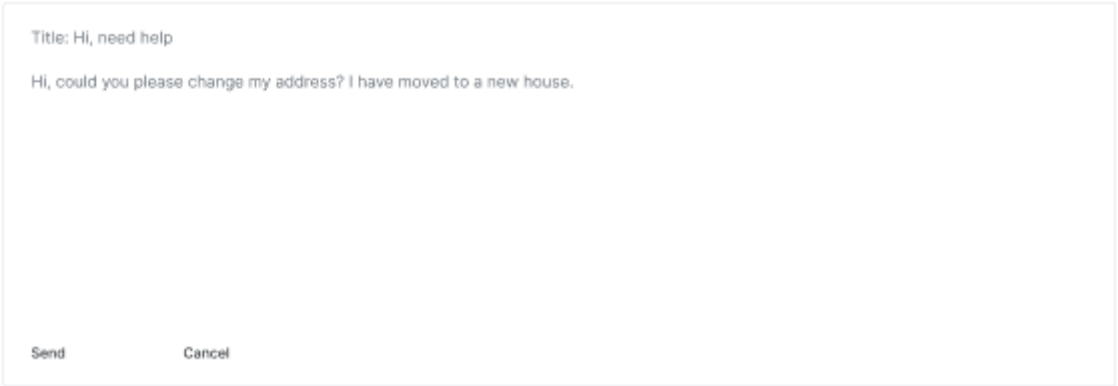

Figure 6: Reusable send message box to be implemented as React component

The send message box has been reused for both making new messages and replying to existing messages, to reduce repeated code in the implementation.

Technical specifications

Technologies

Frontend: React with Bootstrap for layout

Backend: ASP.NET and MongoDB

Dev-ops: Docker, xUnit for unit testing, CI/CD Jenkins

REST API Design

Figure 7: Level 0 data flow diagram

The architecture is a simple three tier system with one back-end service running on ASP.NET. Data flows from the patient to the online clinic, to the doctor in a two way exchange. There is a singular persistent storage as MongoDB database running locally.

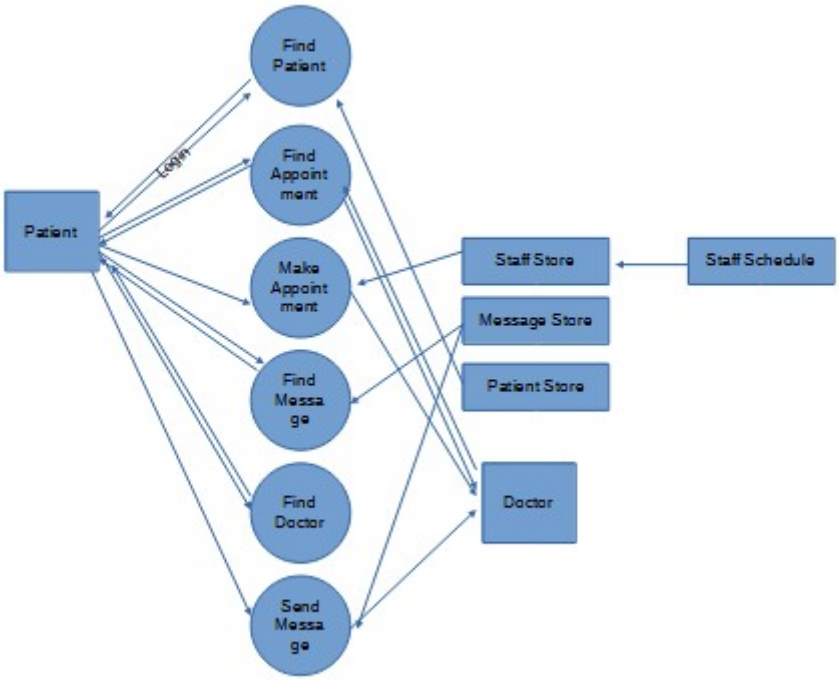

Figure 8: Level 2 dataflow diagram representing dataflow between personas and processes

Figure 8 is an extension of level 0 dataflow diagram, showing the specific processes as well as how these processes interact with the data stores. Currently, there are two main features that needs to be implemented, which are appointment and message management features.