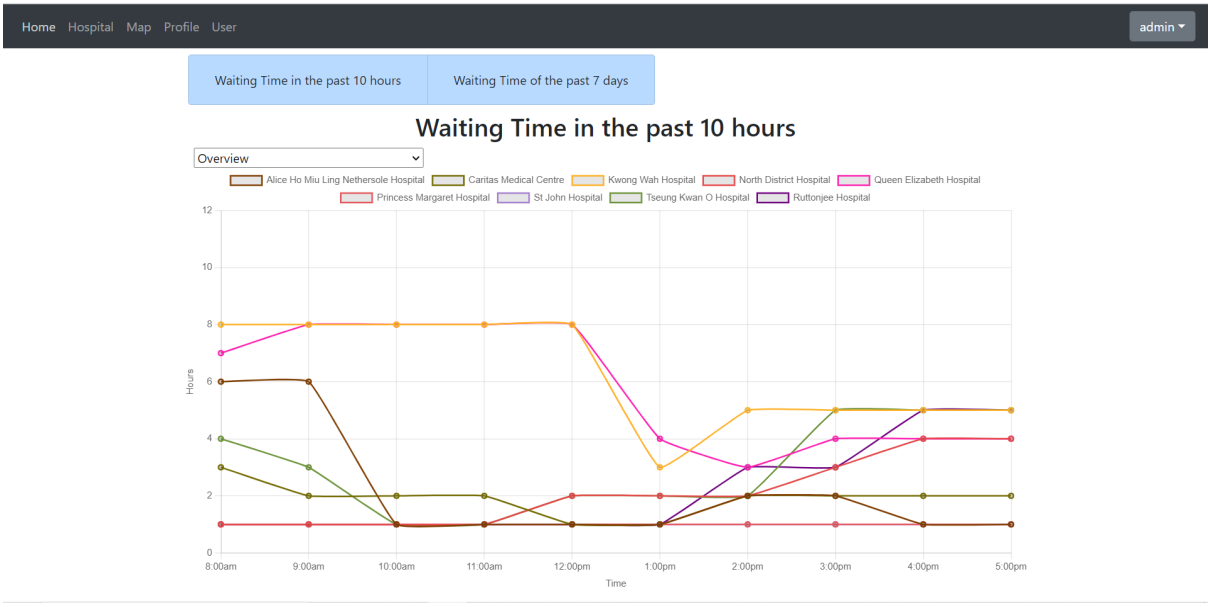
CSCI2720 Group20 ProjectReport

Members:
SZETO Win Key (1155109549)
CHEUNG Kam Fung (1155110263)
WONG Shing (1155109027)
IP Shing On (1155109011)

1. Abstract

This web application aims to provide simple information about hospitals and display these data in a high user friendliness manner with an easy to use and highly understandable user interface to enhance user experience. Users can check the waiting time of each hospital, as well as bookmark them and view them in their profile page. A map function is implemented in this app to provide users an overview of where the hospital locates. Details of each hospital will be shown in their specific detail page. Under the detail page, users can freely leave any comments below regarding that hospital.



2. Methodologies

2.1 Programing language:

This web application is built using the MERN stack. Javascript will be our main programming language as both NodeJS and ReactJS use JS to code with.

2.2 Bcryptjs:

Bcryptjs is a module we use to handle **encryption** of user passwords. Because of **security** concerns, we would only store the encrypted password in the database. It is also used for login verification to compare the password sent by the user and the encrypted password stored in the user database.

2.3 Cookies:

Cookies is a technology we used to store the identity in the browser to enhance **user experience**. After login, several data will be sent from the server for client side to store these information in the cookies for later purpose. The cookies stored are as follow:

2.3.1 userId:

Object ID of this user document in the database. It is used on the client side to include the userId in some API as parameters.

2.3.2 username:

Username of this user, and it is displayed on the top right corner of the header of this application.

2.3.3 accessToken:

A **JsonWebToken** (JWT) for identity check of this user. As this application requires the user to login before he/she accesses any features of this app, this JWT is used for identity check for all actions of this user except login. Note that we set the expiration time of this accessToken to be 30 seconds to **prevent malicious** people from stealing this token and stealing your identity in this application. We deliberately set the expire time in a short period for **security** concern, and the expiry of accessToken will be handled by the refreshToken sent together with this JWT.

2.3.4 refreshToken:

A JWT used as the session tokens for this user. This JWT can be used to generate a new accessToken for this user when the old accessToken expires. This JWT will not expire until the user logout, so the **user experience** will not be affected as long as the session has not ended. We store these refreshToken in a list of tokens in the user database, so it supports multiple logins of the same user. When the user logs out from the application, the program will delete this refreshToken, to prevent any malicious people from stealing your identity.

2.4 Performance:

We have to deal with plenty of array loopings in our project. To facilitate the **performance**, we use array.map(), array.filter() and Lodash Library. These functions and libraries provide an acceptable algorithm complexity.

2.5 Axios:

Axios is a module for the client side to call API and communicate with the server and interact with the database.

2.6 Modularity:

We build our application in a **top-down design** and **bottom-up implementation** approach, to enhance **divide and conquer**, as well as the capability of understanding. We write our code in a general manner with high incrementality, to anticipate changes in the future.

2.7 Google Map API:

We use the library @react-google-maps/api to render the Google Map on our website, which has high understandability and **maintainability**.

2.8 Chart.js:

Chart.js is used to show the past data. The x-axis shows the date while the y-axis shows the waiting time. It allows to show **multiple datasets** at the same time.

2.9 isDeleted and cleanup()

We included a flag (isDeleted) to check if the document is going to be deleted. We use this approach instead of removing the object in the usual way because our three schemas have **joined together** with relations. Instead, we set isDeleted = true for this document, and call a cleanup function whenever the server restart to delete all documents with isDeleted set to true, to **minimize the risk** of a reference from one table pointing to a deleted document, and crashing the whole program.

2.10 createAdmin()

As the **CRUD** operation of users is handled by admin, instead of using the approach of registration in normal applications. We implemented a createAdmin function when the server starts. This function checks if the admin exists in the database, and if admin cannot be found, it will automatically create a new admin.

2.11 Design of data schemas and models of database:

User	**Hospital**	**Comment**
1. username: String 2. password: String 3. favHospital: [{ObjectId, ref 'Hospital'] 4. isDeleted: Boolean 5. sessionTokens: {[String]}	1. name: String 2. waitingTime: Number 3. lastUpdateTime: String 4. latitude: Number 5. longitude: Number 6. district: String 7. comments: [ObjectId, ref 'Comment'] 8. isDeleted: Boolean 9. past10HoursWaitingTime: [{updateTime: String, waitTime: String}] 10. past7DaysWaitingTime: [{updateTime: String, waitTime: String}]	1. content: String 2. date: Number 3. sender: ObjectId, ref 'User' 4. isDeleted: Boolean

2.12 Comparison table

	Advantages	Disadvantages
Mongoose/MongoDB	1. Many different types of data, structured, unstructured, can be stored and retrieved more easily. 2. Easy updates to schema and fields.	1. Joining documents in MongoDB is not easy. 2. MongoDB does not support transactions.
ReactJS	1. Fast and responsive by selective rendering. 2. High modularity, small and reusable modules which are easier for maintenance.	1. Only covers the UI layer and nothing else, we need a bunch of libraries for development. 2. HTML in JavaScript, it is a kind of 'spaghetti code', which causes a hard learning curve.
NodeJS with Express	1. Highly extensible 2. High-performance for real-time applications	1. API Instability 2. Decreased performance during complex computation tasks
RESTful API	1. Allows standard-based protection with the use of OAuth protocol 2. Flexibility and portability	1. lose the ability to maintain state in REST 2. Lack of Security that SOAP is not imposed
ChartJS	1. Interactive 2. Lightweight and fast	1. Only Canvas based 2. Only accept predefined charts

3. References

[1] "# Chart.js," *Chart.js* | *Chart.js*. [Online]. Available: https://www.chartjs.org/docs/latest/. [Accessed: 25-April-20211].

[2] *React Google Maps Api Style Guide*. [Online]. Available: https://react-google-maps-api-docs.netlify.app/. [Accessed: 28-April-2021].

[3] "*The Good and the Bad of ReactJS and React Native*", 27-Feb-2020. [Online]. Available: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-reactjs-and-react-native/. [Accessed: 07-May-2021].

[4] J. T. Mark Otto, "Bootstrap." [Online]. Available: https://getbootstrap.com/. [Accessed: 25-April-2021].