FINAL REPORT

SENG3011 REPORT

TELETUBBIES

Yin Huey Tan z5211373
Amanda (Xiaorui) Li z5206613
Sarah Oakman z5206178
Lavanya Sood z5208121
Yiyun Yang z5187469

1.0 USE CASES

1.1 USE CASES OF THE API

Use Case 1.1	**Get articles by start and end date**
Requirements	**Teletubbies API System is able to retrieve articles within the corresponding time frame**
Initiating actor	**User**
Actors goal	**Get articles between start and end date**

Use Case 1.2	**Get articles by start date, end date with a certain timezone**
Requirements	**Teletubbies API System is able to retrieve articles within the corresponding time frame of a specific timezone**
Initiating actor	**User**
Actors goal	**Get articles between start and end date of a specific timezone**

Use Case 1.3	**Get articles by start date, end date and location**
Requirements	**Teletubbies API System is able to retrieve articles of the location/country within the corresponding time frame and**
Initiating actor	**User**
Actors goal	**Get articles between start and end date of the location/country**

Use Case 1.4	**Get articles by start date, end date,location and timezone**
Requirements	**Teletubbies API System is able to retrieve articles of the location/country within the corresponding time frame and timezone**
Initiating actor	**User**
Actors goal	**Get articles of the location/country between start and end date of a specific timezone**

Use Case 2.1	**Post an article**
Requirements	**Teletubbies API System is able to add an article with URL and date of publication required**
Initiating actor	**WHO/Researchers/Reporters**
Actors goal	**Release new information about epidemics and provide updates on the current situation**

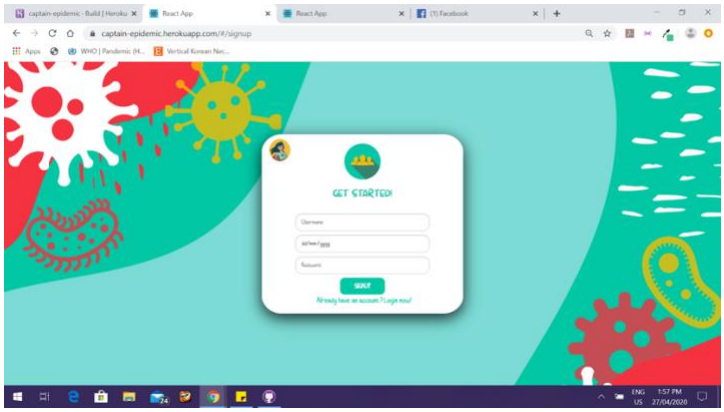
Use Case 3.1	**Put a report**
Requirements	**Teletubbies API System is able to add a new report to an existing article**
Initiating actor	**User**
Actors goal	**Add a new report to an existing article**

Use Case 4.1	**Delete an article**
Requirements	**Teletubbies API System is able to delete an article**
Initiating actor	**User**
Actors goal	**Delete an article and the reports they contain**

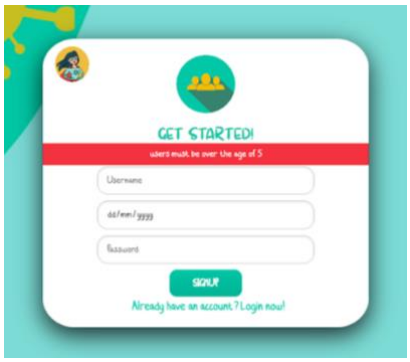
Use Case 4.2	**Show a response when deleting an article**
Requirements	**Teletubbies API System is able to delete an article**
Initiating actor	**User**
Actors goal	**See a response status and informative description to see the outcome of the API call**

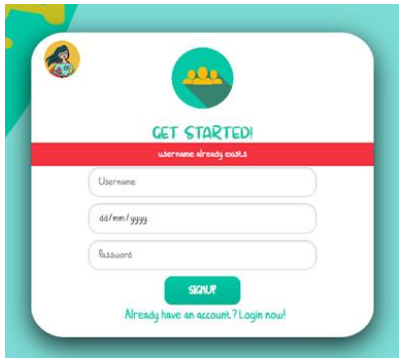
1.2 USE CASES OF ANALYTICS PLATFORM

Use Case 1.1	**Create a new user**
Requirements	**Captain Epidemic System is able to create a new user**
Initiating actor	**User**
Actors goal	**Create an account to access more website features such as games and profile**

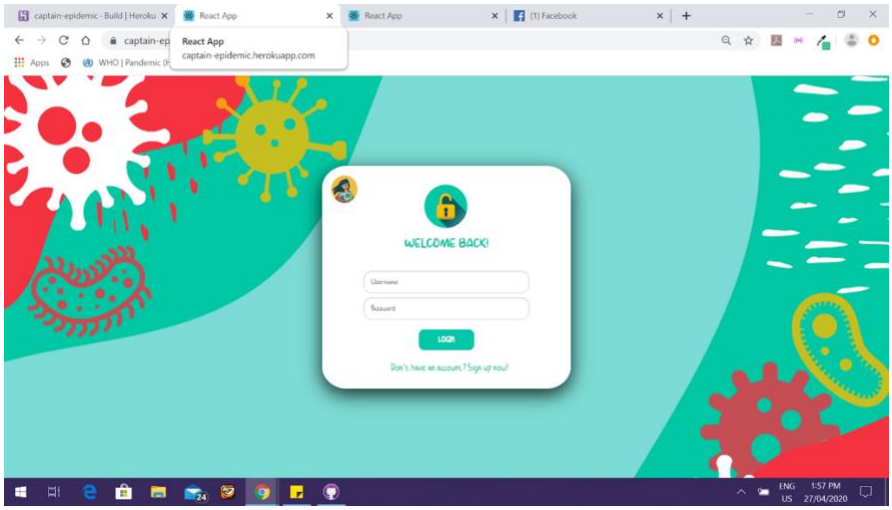


Use Case 1.2	**Signup error messages**
Requirements	**Captain Epidemic System is able to create a new user**
Initiating actor	**User**
Actors goal	**See informative error messages with incorrect signup credentials**

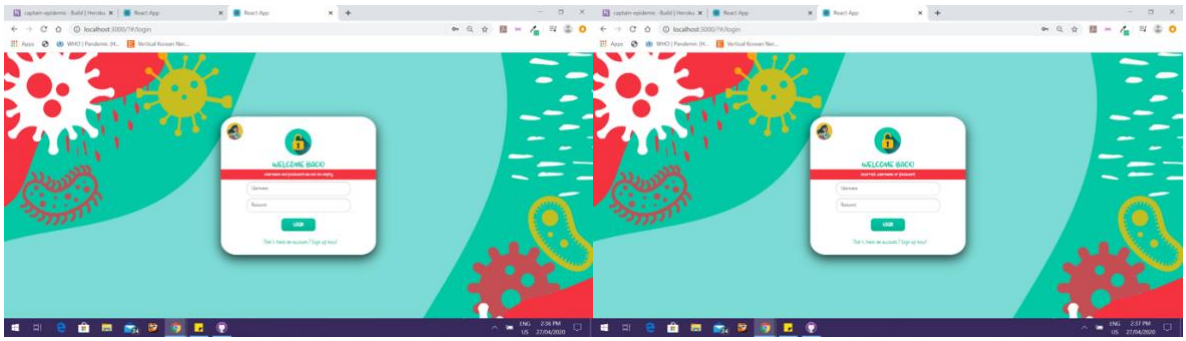




Use Case 2.1	Login with an existing account
Requirements	Captain Epidemic System is able to login an existing user
Initiating actor	User
Actors goal	Login into website to access their profile and extra website features

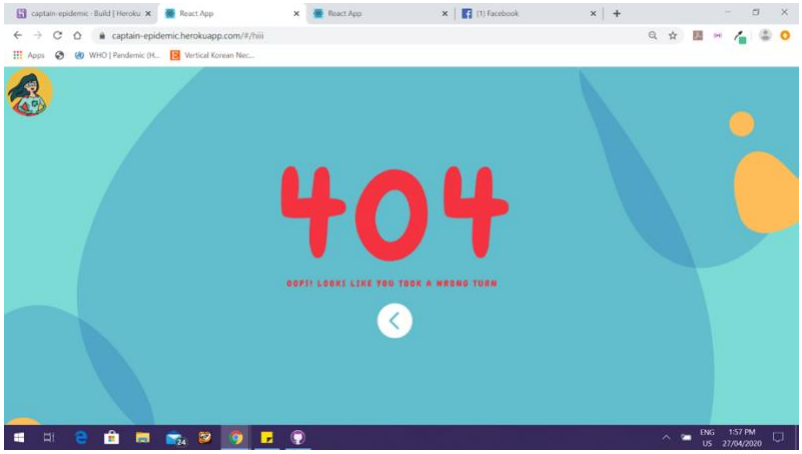


Use Case 2.2	Login error messages
Requirements	Captain Epidemic System is able to login an existing user
Initiating actor	User
Actors goal	See informative error messages with incorrect login credentials

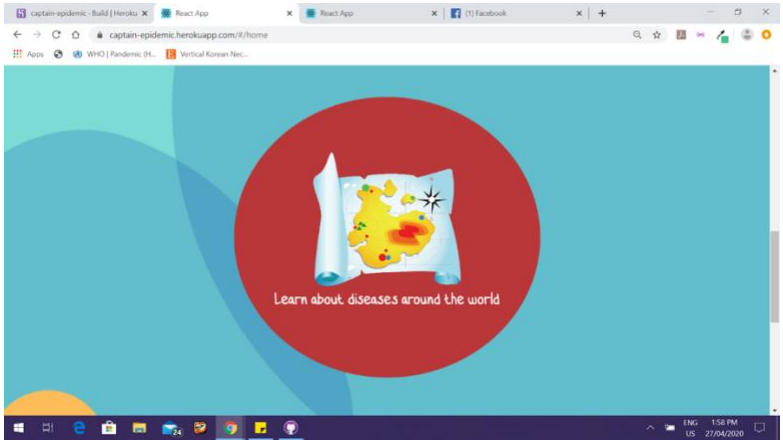


Use Case 3	Logout a current user
Requirements	Captain Epidemic System is able to logout the current user
Initiating actor	User
Actors goal	Logout from the website

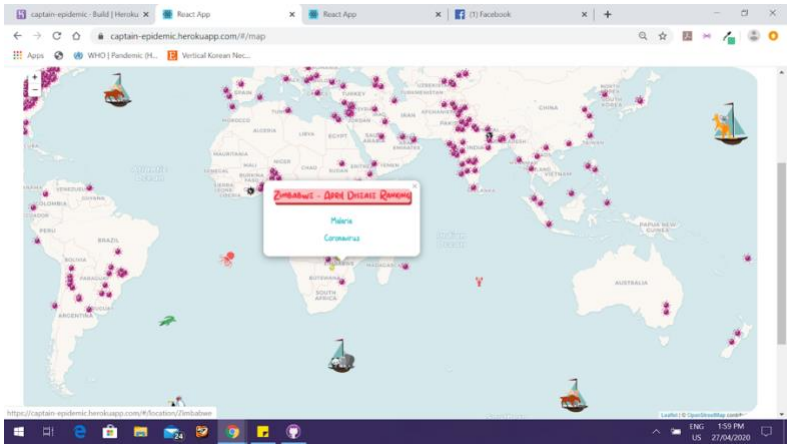
Use Case 4	404 error message
Requirements	Captain Epidemic System is able to show an error page
Initiating actor	User
Actors goal	See an error page when an unknown url is entered where they can go back to their previous page or to the home page



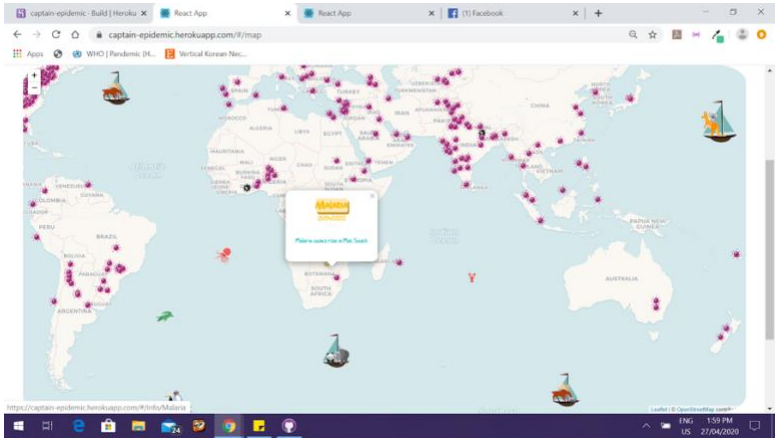
Use Case 5.1	Access global disease map
Requirements	Captain Epidemic System is able to interact with a global disease map
Initiating actor	User
Actors goal	Access the global disease map from the home page



Use Case 5.2	View the monthly ranking of diseases by country
Requirements	Captain Epidemic System is able to interact with a global disease map
Initiating actor	User
Actors goal	See the monthly disease ranking in a country when it is clicked. Also, they are able to navigate to the disease's or country's information pages



Use Case 5.3	View disease reports in a country
Requirements	Captain Epidemic System is able to interact with a global disease map
Initiating actor	User
Actors goal	See the disease report when a disease icon is clicked. Also, they are able to navigate to the disease's information page

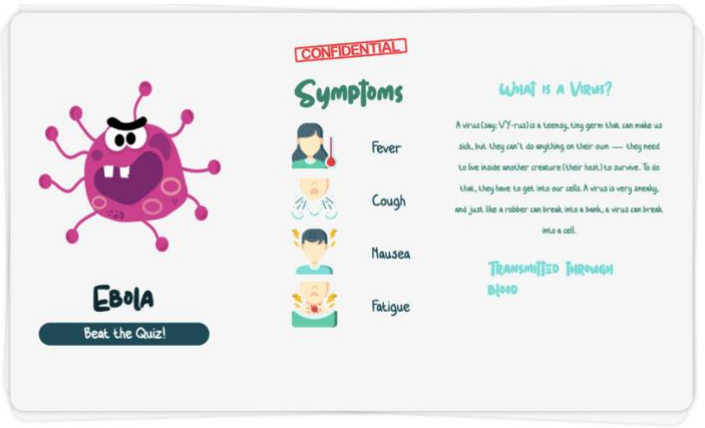


Use Case 5.4	View epidemics for a country
Requirements	Captain Epidemic System displays up to 4 epidemics that have occurred in a country, mentioning states affected and map pinpointing the location of the state
Initiating actor	User
Actors goal	Understand which epidemics are or have affected a location to gauge the risk level

Use Case 5.5	View the latest reports for a country
Requirements	Captain Epidemic System displays up to 5 most recent reports related to a country
Initiating actor	User
Actors goal	Stay up to date on recent news or cases happening in a country for any disease

Use Case 6.1	View disease symptoms
Requirements	Captain Epidemic System displays disease name and information such as symptoms
Initiating actor	User
Actors goal	Learn about what symptoms to watch out for, be self-aware about if they potentially have the disease, in preparation for the quiz/game

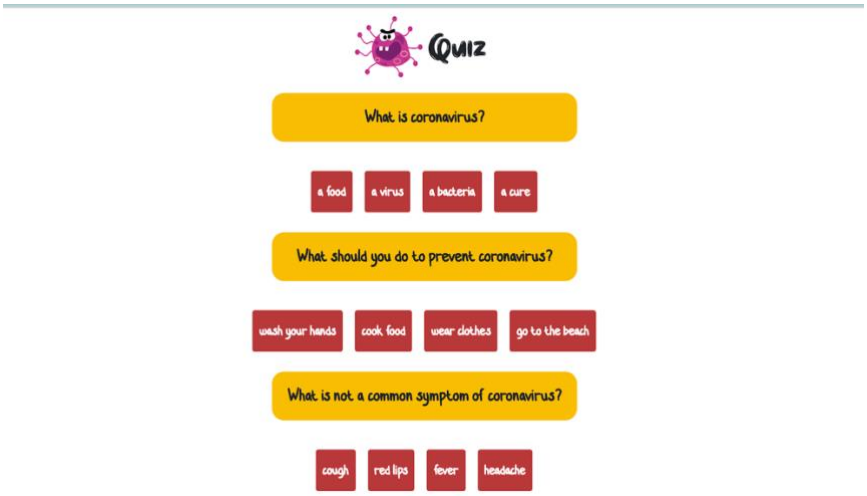
Use Case 6.2	View information about diseases' type
Requirements	Captain Epidemic System displays if the disease is a virus, bacteria, fungus, parasite or another germ with definitions and transmission type
Initiating actor	User
Actors goal	Learn about the classification of disease, how disease spreads to be more aware of surroundings and protect themself



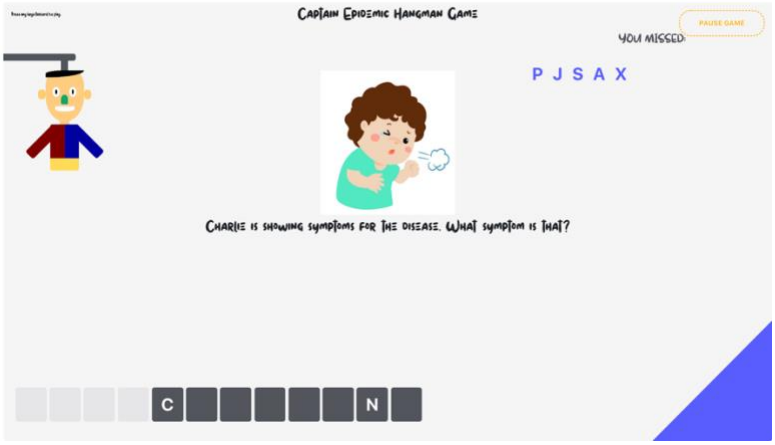
Use Case 6.3	View latest reports about disease
Requirements	Captain Epidemic System displays up to 5 most recent reports mentioning the disease
Initiating actor	User
Actors goal	Stay up to date on news and case reports for a specific disease occurring in any country

Use Case 6.4	View most affected countries of disease
Requirements	Captain Epidemic System displays up to six most affected countries of a disease
Initiating actor	User
Actors goal	Check if a location is affected heavily by the disease and determine if they're at risk

Use Case 7.1	Learn about diseases through quizzes
Requirements	Captain Epidemic System is able to allow user to play quizzes
Initiating actor	User
Actors goal	Play and score a quiz regarding the disease to gain knowledge about it

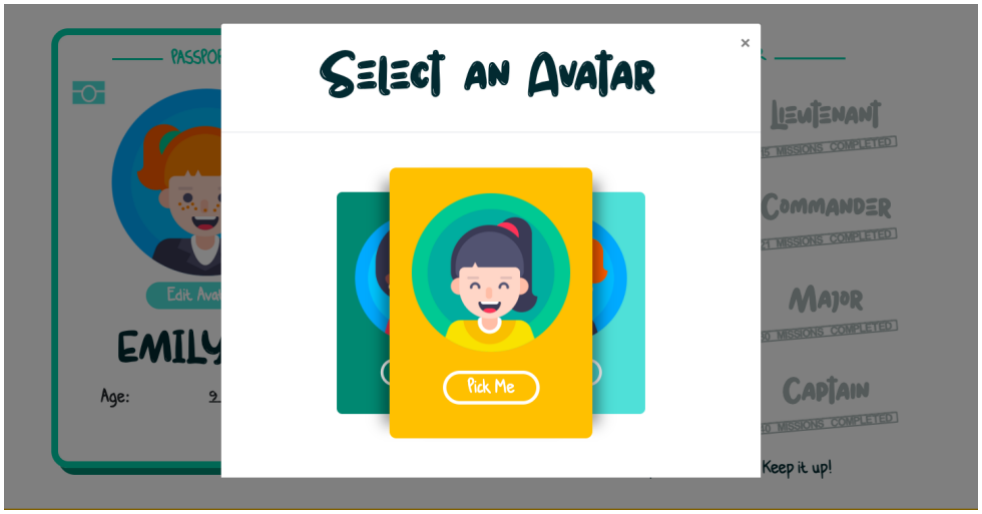


Use Case 7.2	Learn about symptoms of diseases through game
Requirements	Captain Epidemic System is able to allow user to interact with a hangman game
Initiating actor	User
Actors goal	Play and pass the hangman game regarding the symptom of a specific disease to gain knowledge about it



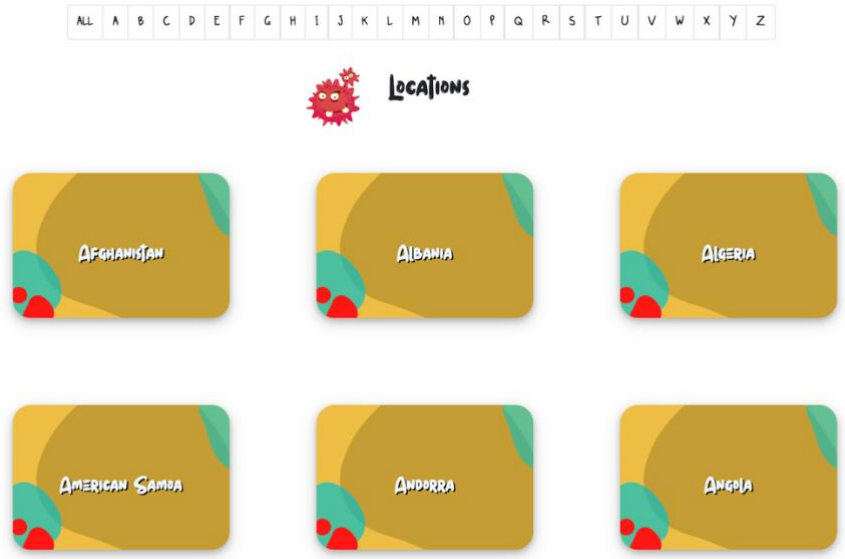
Use Case 8.1	View current game score and achievements
Requirements	Captain Epidemic System is able to allow user to view their current game achievements at their own profiles
Initiating actor	User
Actors goal	View their score and badges earned from games/quizzes played

Use Case 8.2	Edit own profile avatar
Requirements	Captain Epidemic System is able to allow the user to edit their own avatar
Initiating actor	User
Actors goal	Choose a picture that represents the user so they are engaged



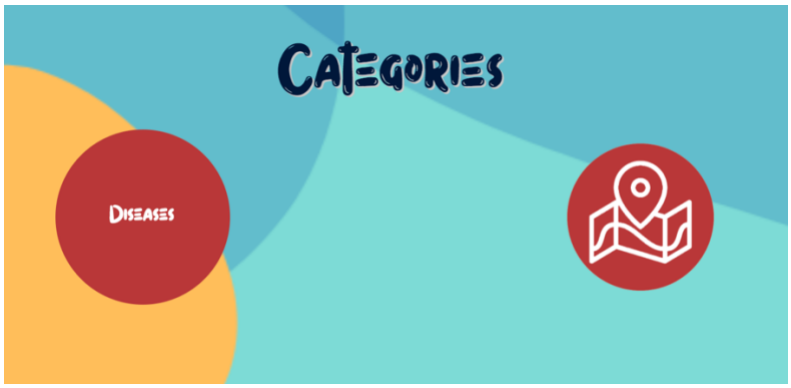
Use Case 9.1	Access location category
Requirements	Captain Epidemic System is able to display location category
Initiating actor	User
Actors goal	Access the location category from the home page

Use Case 9.2	Access location category
Requirements	Captain Epidemic System is able to display a list of locations which linked to the location's information page
Initiating actor	User
Actors goal	View all locations or in selected initial



Use Case 9.3	Access location category with error message
Requirements	Captain Epidemic System is able to display an error message if no result found for locations matched with the selected initial
Initiating actor	User
Actors goal	View an error message showing no countries found with the selected initial

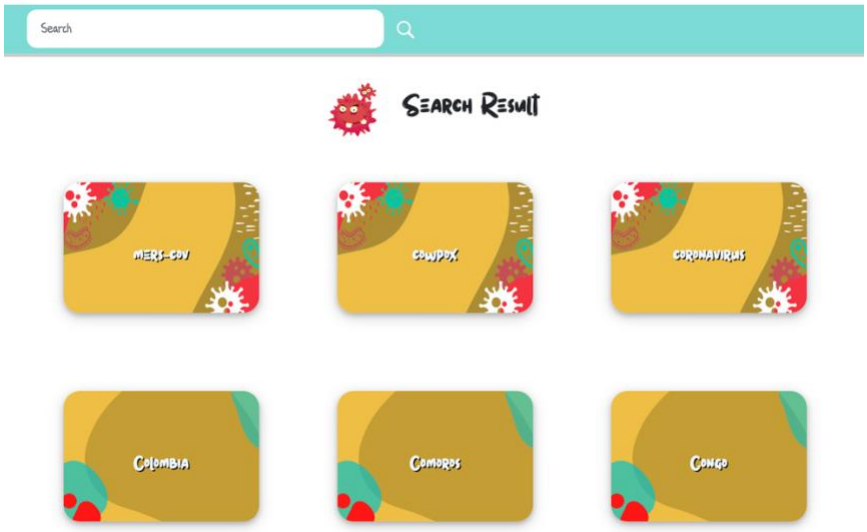
Use Case 10.1	Access disease category
Requirements	Captain Epidemic System is able to view disease category
Initiating actor	User
Actors goal	Access the disease category from the home page



Use Case 10.2	Access location category
Requirements	Captain Epidemic System is able to display a list of diseases which linked to the disease's information page
Initiating actor	User
Actors goal	View all diseases or in selected initial

Use Case 10.3	Access disease category with error message
Requirements	Captain Epidemic System is able to display an error message if no result found for diseases matched with the selected initial
Initiating actor	User
Actors goal	View an error message showing no diseases found with the selected initial

Use Case 11.1	Search through the diseases and locations
Requirements	Captain Epidemic System is able to search through diseases and locations
Initiating actor	User
Actors goal	Search for locations and diseases with input in the navigation bar;Also, they are able to navigate to the disease's/location's information page



Use Case 11.2	**Search through the diseases and locations with error message**
Requirements	**Captain Epidemic System is able to display an error message if no result found for result found with the given input**
Initiating actor	**User**
Actors goal	**View an error message showing no result found with the given input**

2.0 SYSTEM DESIGN AND IMPLEMENTATION

2.1 ANALYTICS PLATFORM AND FINAL SOFTWARE ARCHITECTURE

Our previous web stack can be found as Figure 4.0 in our Design Report (Appendix 1).
Our final technology stack is show below:

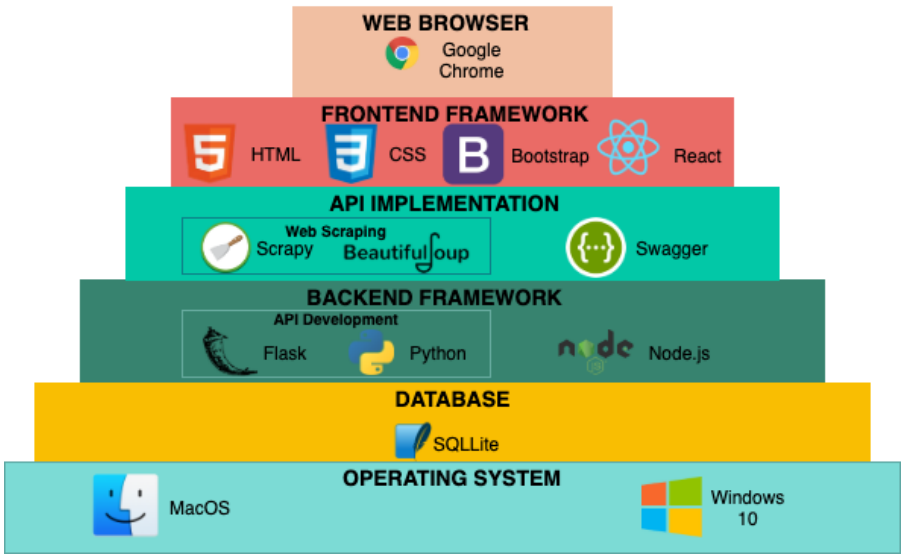


Fig 1.0 – The web stack

Changes to our software architecture include switching to React and Bootstrap from jQuery for frontend framework and adding node.js for backend of the platform (whilst Python and Flask was used for backend of API Development).

We decided to switch from jQuery to React because it utilises 'Components' which are module-like pieces that increase reusability of code. It makes use of the JSX(JavaScript Extension) which is a particular syntax letting HTML quotes and HTML tag syntax to render particular subcomponents (allows for HTML within JavaScript which means simpler and cleaner code) as well as guarantees quicker rendering by using a virtual DOM (Document Object Model). It also offers a very rich JavaScript library which provides more flexibility to the web developers to choose the way they want.

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It is lightweight and customizable with many free and professional templates and plugins. Bootstrap also has in-depth documentation on every component which makes teamwork better and easier. We used Bootstrap for consistent modals and elements that were more dynamic.

We decided on Node.js for the platform's backend because it's compatible with MacOSx and Windows 10 and can be used as a single programming language which allows developers to write both the frontend and backend application in JavaScript. The node package manager (NPM) also allows programmers to install packages so they don't have to write common features from scratch. Additionally, we are using ExpressJS as our framework, and as a result in the future the platform can be extended to a cross platform compatible mobile app. Express provides extra routing features as well to help us connect to external APIs.

Additional APIs used are CalmClams API and Emperor Augustus API from other teams, and Leaflet and countryflags.io API from external sites.

Along with our own World Health Organisation API, we used CalmClams API from the Global Incidents Map website to collate disease report information. CalmClams data was more focused on new cases which was different to our data and increased the range of information we could provide. It also returns location coordinates required for our interactive map so we can display disease icons in the correct state. Also, for the interactive map, we used the leaflet API which is an open source JavaScript library, to allow for panning, zooming and making countries clickable.

We also used Emperor Augustus WHO API for symptoms of each disease as our API used unsuitable complicated medical terms such as Acute Respiratory Syndrome whilst theirs had simple terms that children can understand such as coughing, fever and headache. The last API we used was countryflags.io which helped us link countries to their respective flag.

2.2 HEROKU DEPLOYMENT

The analytics platform was deployed on heroku with the following link https://captain-epidemic.herokuapp.com/

Since Heroku has a buildpack for Node.js, the Captain Epidemic website was easily deployed. It was constructed with automatic deployments by connecting our master branch of the Github repository to Heroku. Also, due to the widely and well documented online tutorials for heroku and React/Express platforms this was a suitable and time efficient option for making our analytics program available on the web.

3.0 TEAM ORGANISATION AND CONCLUSION

3.1 TEAM RESPONSIBILITIES

As shown in appendix 3 the team broke down the whole project using Gannt chart to plan out the timeline and the structure of the project. We used various resources throughout the whole project in order to organise the whole project - Jira Board in order to manage the roles, messenger for communication, google docs for drafting our ideas and github for version control of our code.

For the web application development our whole team worked on both the frontend and the backend for different parts of the website:

Yin Huey Tan: Hangman Game, Quiz
Xiaorui Li: Profile Page, Diseases Page, Locations page
Sarah Oakman: Map, Login, Signup
Lavanya Sood: Base (header and footer), home page, landing page
Yiyun Yang: Search, Categories

3.2 ACHIEVEMENTS OF PROJECT

Through this project our team was able to accomplish various tasks by expanding our knowledge and learning about the different components required in order to build a successful application.

API

We have developed an API that allows users to retrieve news articles obtained from WHO website. The WHO news articles have been scraped and separated into disease reports in the hopes of detecting epidemics by collecting global disease data. Disease reports can be accessed using GET requests whilst the POST, PUT and DELETE request can be accessed by authorised users which manipulates the scraped data stored within an SQL database.

ANALYTICS PLATFORM

Since we identified a lack of resources in the epidemic education market with concise information in simple language and catchy, up-to-date information, we have introduced and developed an engaging and intuitive platform which aims to educate children particularly those above 5 years old about various diseases in the world. The platform consists of different interactive games, map and information implemented through integration with our own API and various other external APIs. We have successfully implemented the platform using react and node.js and users can also access our website hosted online easily. We hope that children will be benefited from our website by experiencing a different way of learning about current diseases.

3.3 ISSUES ENCOUNTERED

As we were learning React and Express in this project, we encountered some difficulties in the beginning. Some team members experienced issues when attempting to retrieve backend and api data with Express and notably, one member's Express communication could not be configured. Also, asynchronous calls in React caused some problems when loading pages, as they would render

before the data was received from our Express backend. Despite these issues, we researched the problems encountered and were able to develop a functioning backend with Express which communicated with the Frontend React application correctly.

To ensure our data was updated regularly and prevent slow loading of data by calling and waiting for an APIs response with every page load
Slow website loading speeds were also experienced on some pages, for instance the information pages for countries. This was because multiple API calls had to be made which sometimes required a few minutes to retrieve data. So, to avoid long loading times whilst having current API data, each API is called only once per day using cron in Node.js and the data is stored in our backend databases.

Problems relating to the interactive map and locations were also experienced. Since country names were used from different sources e.g. geolocation json, our api and the calmclams api, there were issues integrating them. More coding and testing was required to link countries from the maps page to the location information pages. Subset checking was used as well as manually testing and adding extra checks for certain countries. Missing disease reports and data for some countries was another issue encountered. Due to lack of time to use more apis and data, these countries on the map page are not clickable. Also, rendering a map on each location information page of the current country was quite complicated as the latitude and longitude coordinates for the centre of the countries had to be found for the leaflet api.

In terms of disease information, it was difficult finding sources for prevention tips and symptoms that were simple enough to understand for children. Options considered include Google Search API and Bing API, since both engines provide descriptions as shown below.

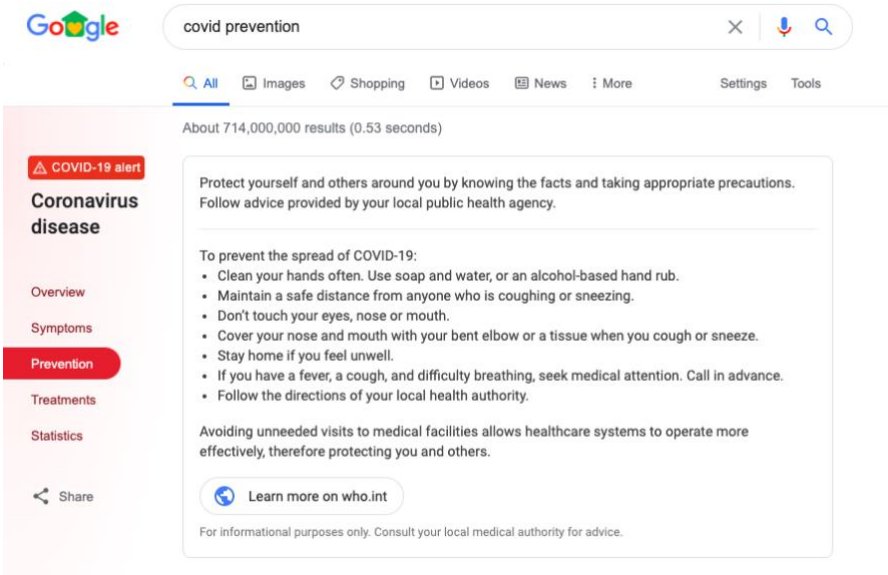


Fig 2.0 – Search results of COVID Prevention example

However, Google's Search API was discontinued in 2009 and alternatives like SerpAPI were not free. Bing's API only returns search links, so it didn't provide the information needed. This issue was fixed by using the other team's Emperor Augustus API. Issues were also encountered with this API though. Some disease searches fetched no information, for example 'Cowpox', and the search had to be expanded to 'Pox' for results to be found.

3.4 CONCLUSION

Before the workshop we believe that having the skills necessary in order to scrape a website and build an API would have helped as it would have allowed the team to have a head start on the project. This is because the first few weeks of the course were spent understanding how to start developing the project so even though our team was successfully able to build and deploy the API, having previous knowledge about building the API could have helped our team use the time in the beginning of the term more productively.

Overall, our team was able to collaborate successfully in order to build a working web application that met most of our goals that we had set out for the project. One thing that we could have improved on starting programming the website earlier as it would have made the amount of work that we needed to implement less and allow us to spend more time on important features.

For additional features of this project we thought of developing a parents and teachers portal which could allow them to track the progress of their children as well as host Kahoot quizzes in order to help them interact with the website more. However due to time constraints we were not able to get these features to work but we wish to implement these features in future iterations.

4.0 APPENDICES

Appendix 1
https://github.com/lavanya-sood/SENG3011_TeleTubbies/blob/master/Reports/Design%20Report.docx

Appendix 2
https://github.com/lavanya-sood/SENG3011_TeleTubbies/blob/master/Reports/Testing%20Documentation.pdf

Appendix 3
https://github.com/lavanya-sood/SENG3011_TeleTubbies/blob/master/Reports/Management%20Information.pdf