SENG3011 Engineering Workshop 3

Deliverable 4

ANALYTICS PLATFORM FOR PREDICTING EPIDEMICS: Project Overview and Implementation Report

By team4masters

Navid Bhuiyan (z5260384), Jin-Ao Olson Zhang (z5211414), Adrian Borjigin (z5205791), Aaron Guan (z5208046), Kieran Nguyen (z5208469)

Contents

Project Background

Project Specification

Implementation

- **Features**
  - **Prototypes**
  - **Final Implementation**
- **Design**
  - **Frontend**
  - **Backend**
    - **API Implementation**
  - **System**
    - **Deployment**
    - **Testing**
  - **Future Extensions**

Conclusion

Project Background

The Integrated Systems for Epidemic Response (ISER) which is a NHMRC Centre for Research Excellence located at UNSW, has requested to design a scraping, storage, search and reporting system to automate outbreak detection and surveillance. As part of Phase 1, the report will break down our software and platform design that fit the major aspect of the report gathering and search aspects of the project specification. Our project management practices are also described here to allow for flexibility if requirements or our understanding of the project changes.

For Phase 2, it has been requested that it is to be investigated the viability of utilising these APIs being developed to seek any viable business use case for them. Upon the assessment of potential user groups in the travel sector (refer to Use Cases and Requirements Report), there was a justification to transform the information obtained by the design of our APIs from Phase 1. MediTravel is a customisable data dashboard platform, which provides medical and outbreak threat information to travellers who need to plan for their travels or get sick abroad. Using the Phase 1 APIs, the information will interpolate region based information on the threat of outbreaks per region, and present medical information to combat these diseases.

Project Specification

The project specifies the development of an outbreak surveillance system, broken up into two stages. This report will focus on the implementation of Phase 2, which specified the development of a web service which integrates Phase 1 into a general business model where the services developed previously could be used to present this system and the information behind it differently.

As observed from our Use Cases and Requirements Report, meeting the needs of travellers impacted by COVID-19, shall extend into the feature in preventing loss of health and damages caused by outbreaks and local disease while travelling abroad. There is a unique integration that can be found using the Reports API developed during Phase 1, where we indirectly interpolate and present information obtained from the reports to the user via a data dashboard. The project also requires the use of external APIs to complement information obtained by our scraper, to mould information to better suit our chosen business case, which is MediTravel.

Implementation

Features

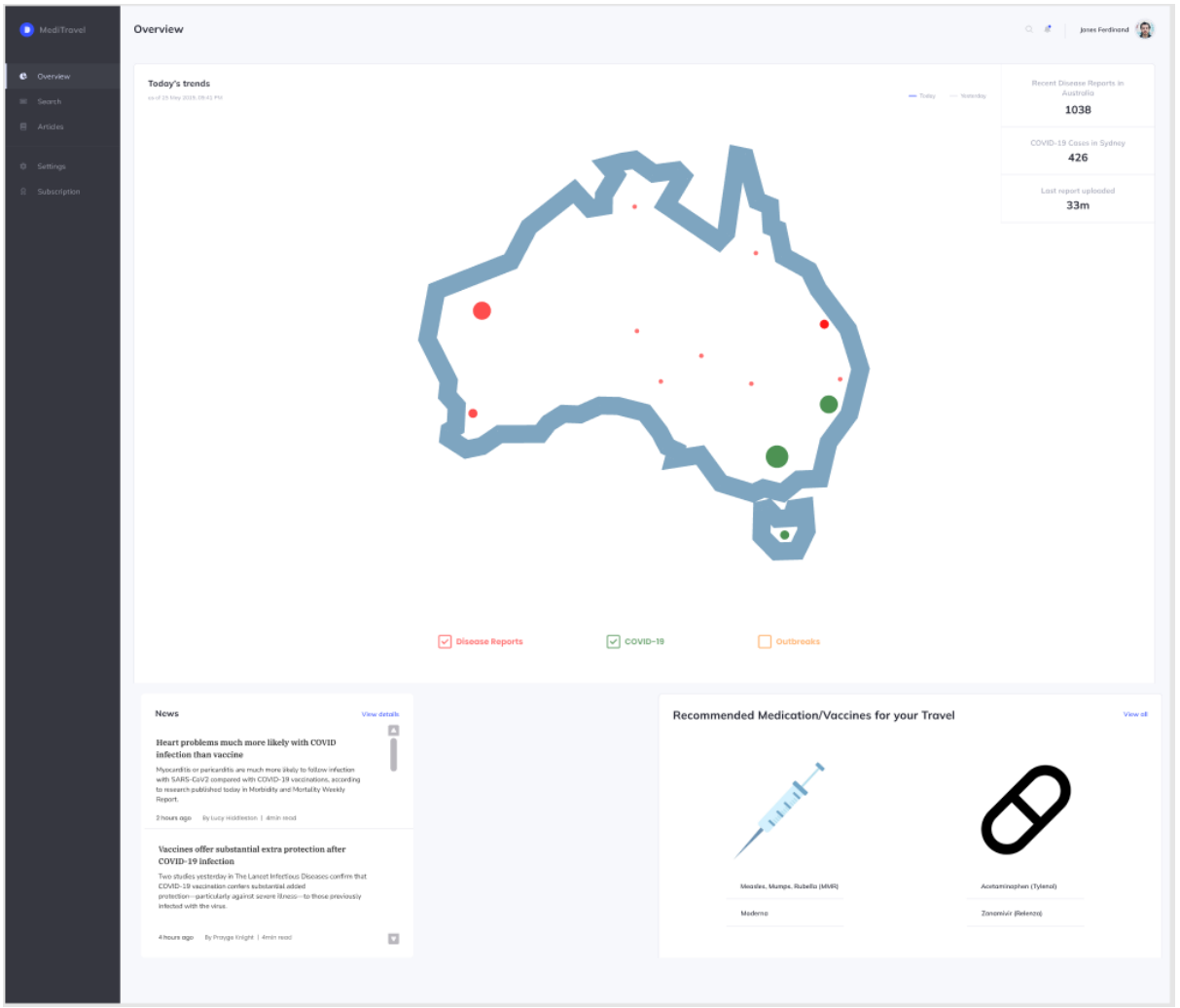
The features of our implementation follow on from the requirements outlined in the Use Cases and Requirements Report.

Requirement	**Description**	**Specific Features**
Dashboard	Get the most recent tweets about local diseases from the current location. By selecting a location on the map, travel advice on that location will be reported, including (travel restrictions, quarantine, vaccines and medication).	- Twitter Feed - Prevention, vaccine and medication information
Map	A map that will be displayed on the dashboard and can be dragged. Allows for another way to search for information.	- Circle highlights to display where reports are occurring - Modal popups upon hovering over a circle, displaying common diseases and number of reports in that area
Search & Report Generation	Allows users to search any specific location and provide detailed reports regarding various health information	- Search bar to enter a location - Report of relevant disease articles at that location
Subscription	Enables users to subscribe to the product to access additional features	- Ability to select different tiers of subscription and apply changes to the user's account

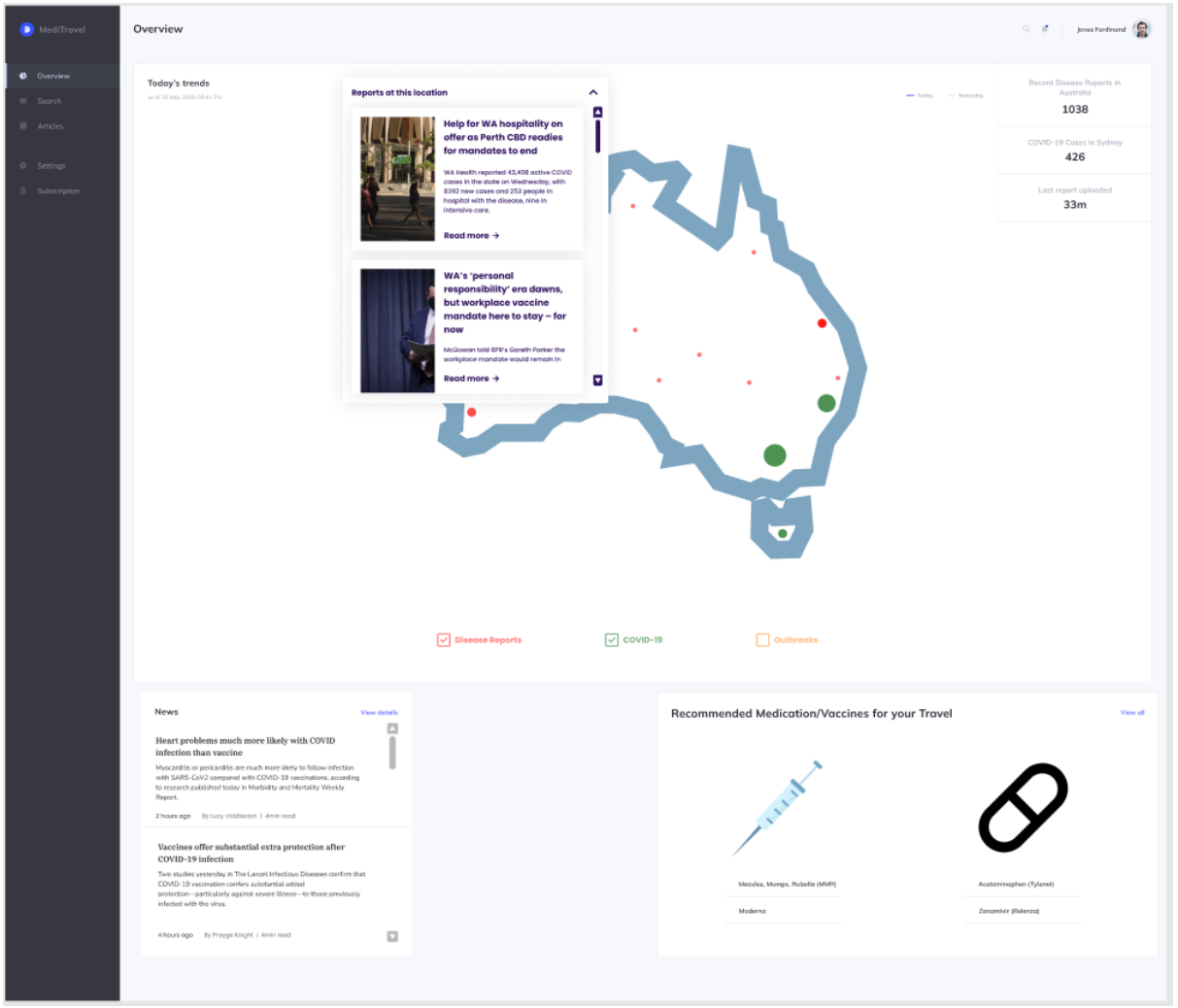
Prototypes

The following screenshots are from an initial prototype developed in D3 using Figma, which represent our original proof of concept for the product. This provided a starting point upon which we could begin to model the frontend and formulate ideas for what features to begin working on first.

Dashboard:

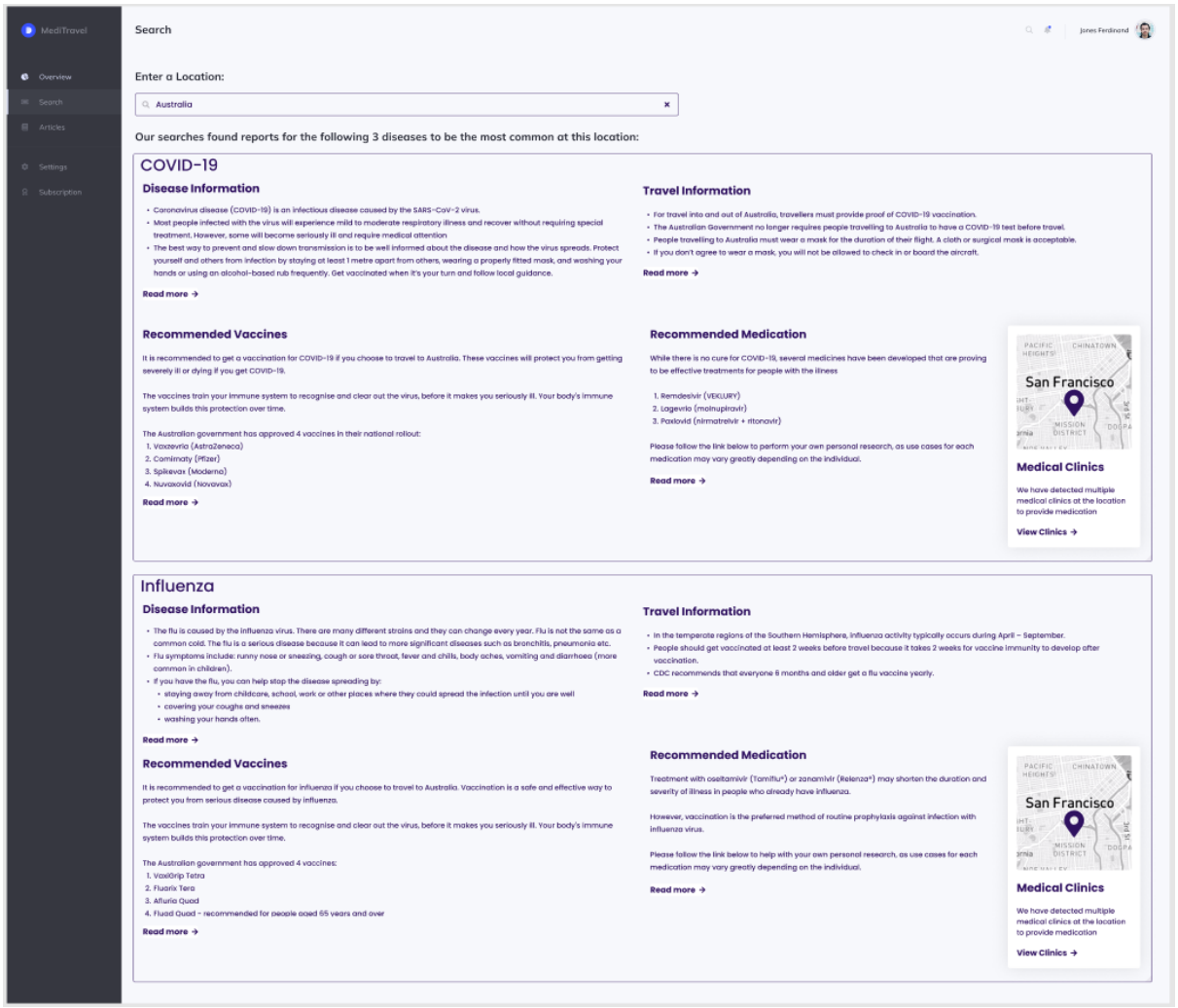


The users' landing page is the personalised dashboard where there is an interactive map designed through the Google Maps API, which shows any diseases reported in the area being viewed. Below that will include any local news in the area regarding the relevant diseases and any recommended medication or vaccines for visiting the location.



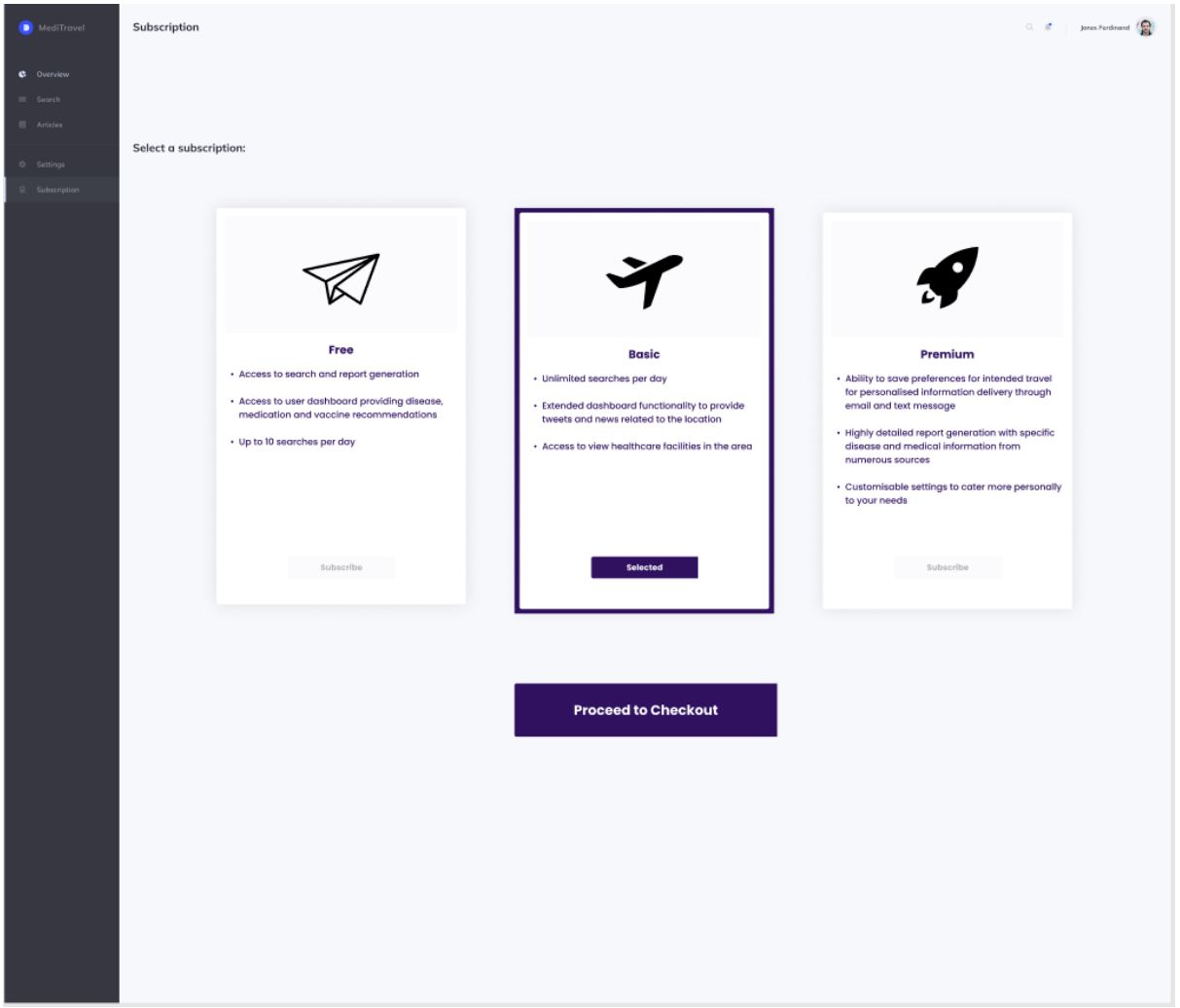
The interactive map can be clicked on in certain locations indicated by a coloured circle, showing specific diseases in the area. By clicking on the circles, a scrolling modal will appear above the location with relevant reports about the disease.

Search & Reports:



On the search page, you will be prompted to enter a location into a text box and by doing so, the search system will generate the user with a detailed yet succinct report of each disease in the location searched. Some useful information included excluding the information that can also be found on the dashboard would be travel information which talks about any relevant quarantine procedures or mandates required by the government in the desired location. The search function also provides the location of medical clinics that can aid the user in case of an emergency or to provide medication.

Subscription:



The subscription page allows the user to subscribe to one of the packaged deals in order to receive perks and benefits from the site. We have a free, basic and premium subscription deal which all provide varying benefits. We have created these three packages in order to fit our three user groups: casual, frequent and business.

Final Implementation

Upon development of our product, the initial prototype went through a few changes that we deemed to be more appropriate for the service we provided.

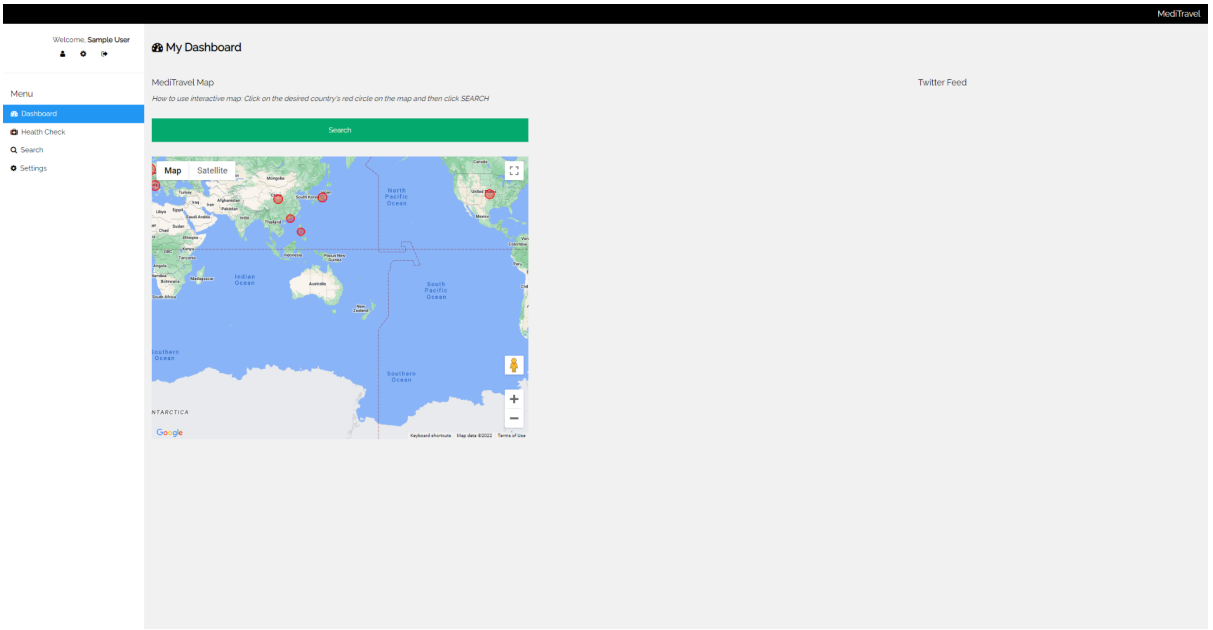
Firstly, the map on the dashboard was modified to simply display common diseases and the number of reports of such diseases at a location upon clicking on the red circle. This modification was made due to the dashboard being intended as a quick checkup service, where the user is able to receive highly summarised information, and would ultimately reduce information overload. Instead the functionality for reports being provided was moved to the search function, where users who are interested in finding relevant

reports to their travels can navigate to in order to find more detail should they require it.

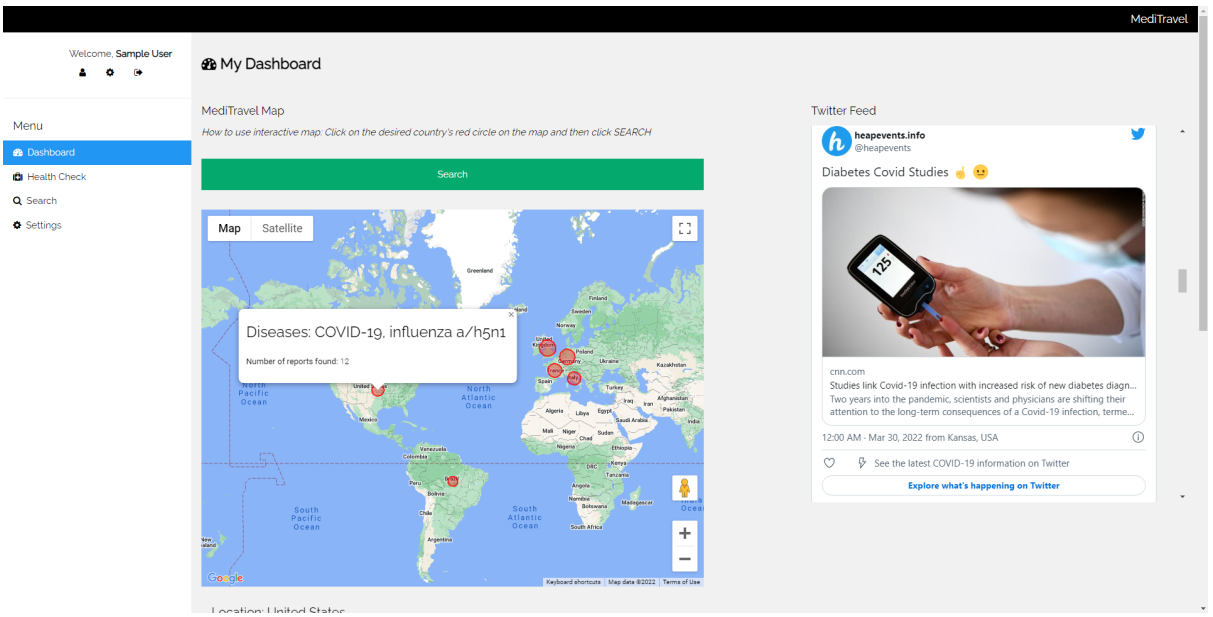
Additionally, most of the information that was originally displayed in the search and report page, including prevention, vaccine and medication information, was also moved back to the dashboard, whereby the user can click a circle and then press “Search” in order to view the information pertaining to that country. Our team determined that this was quite important to display on the dashboard as opposed to in the reports as it is more convenient to the user, where the dashboard aims to provide fast, personalised information whilst the report section provides additional detail.

The following screenshots show the final product which we were able to achieve at the end of the timeline:

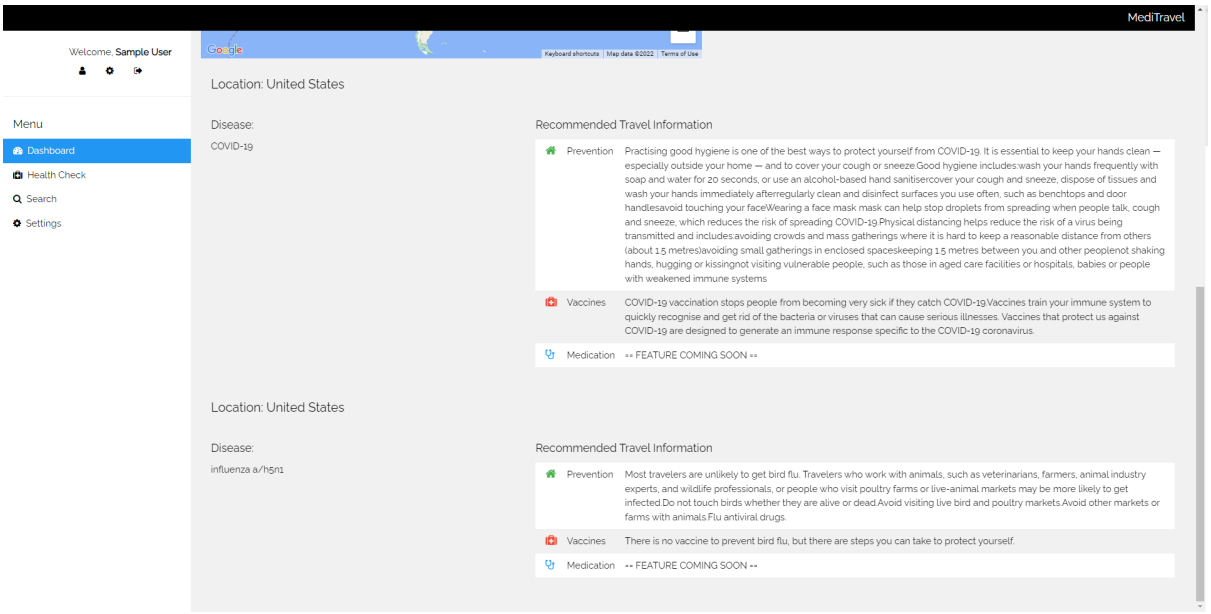
Dashboard:



Original state of the dashboard. From here, the user can interact with the Google Maps API, to explore which diseases are present in each country. The user can have the ability to scroll through to see which countries are afflicted with a disease, marked with a red circle, and upon hovering over this circle, the site will show which diseases are present.



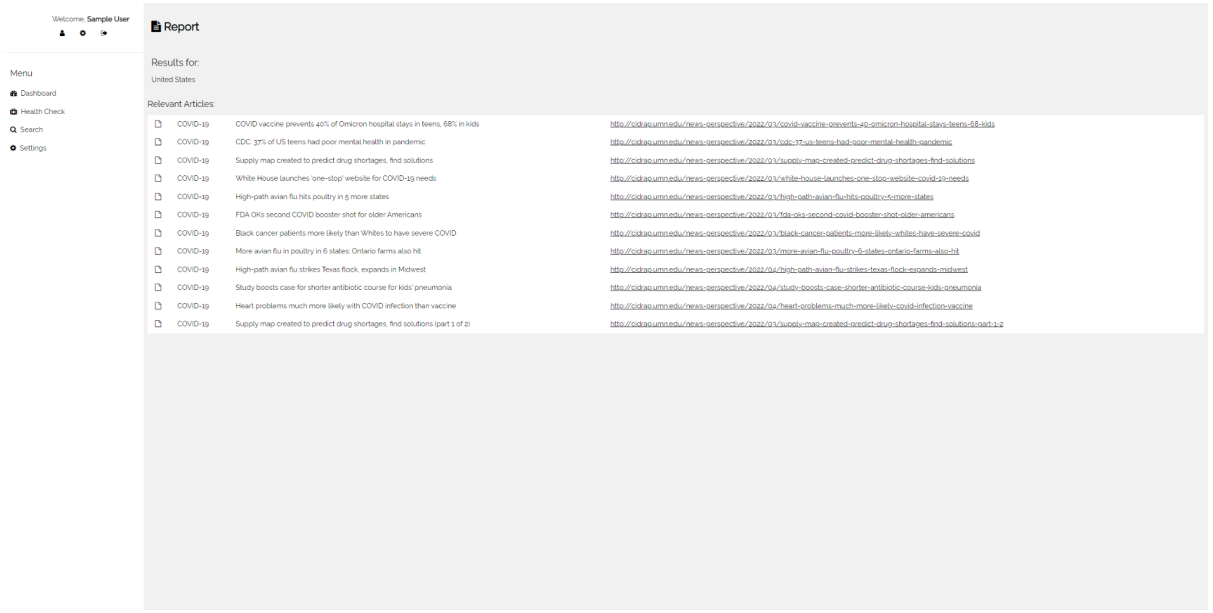
Upon clicking one of the red circles and pressing search, the dashboard is filled with a "Twitter Feed", which allows them to access relevant tweets in that country which discuss the current situation.



In addition to the twitter feed, information on prevention, vaccines and medication is provided underneath, specific to the common diseases identified at that location. This would fulfil the intended use case for travellers to select a country which they are curious about travelling to, and then be provided information about how they may prepare for their trip.

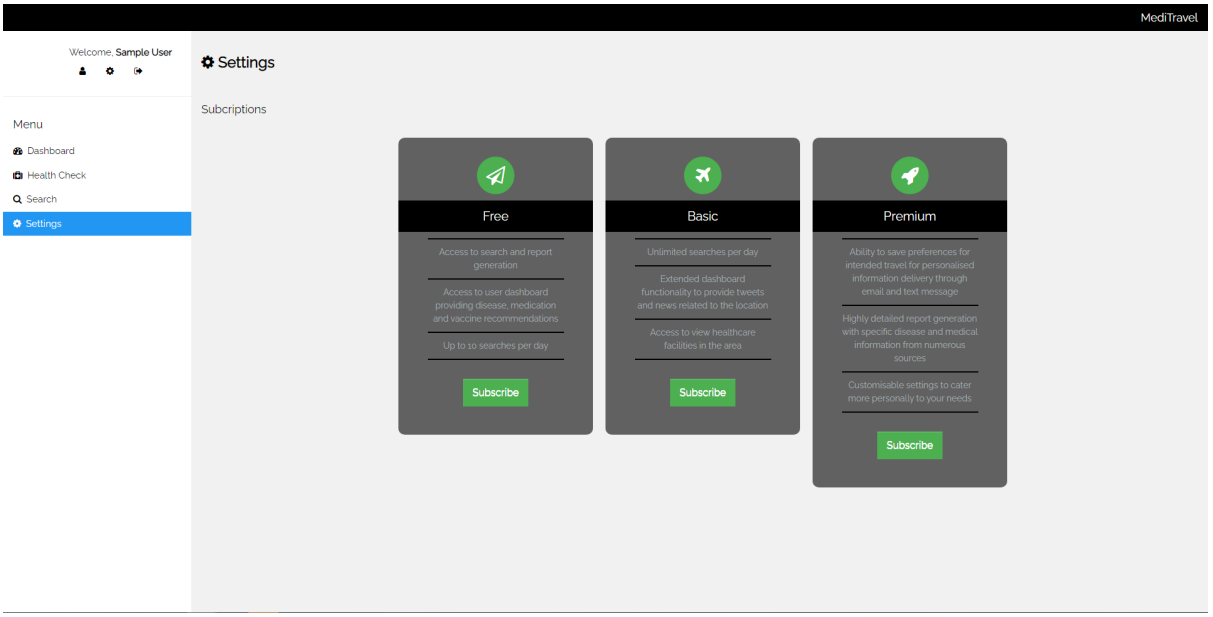
Search & Report:

The user will be prompted by a textbox where they can search for a country to look for relevant reports in that country.



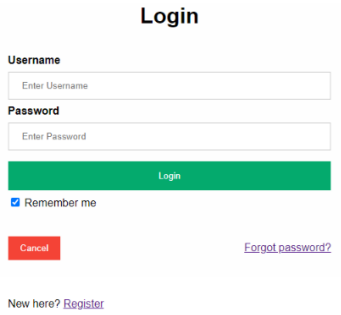
Upon clicking search, the user will be presented with a list of all the relevant reports to the location, where they will show three columns: type of disease, headline of the report, and URL to the report.

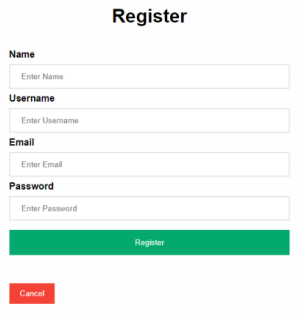
Subscription:



The subscription page will allow the user to subscribe to one of the three package deals we present: free, basic, and premium which reflects our user groups and business model.

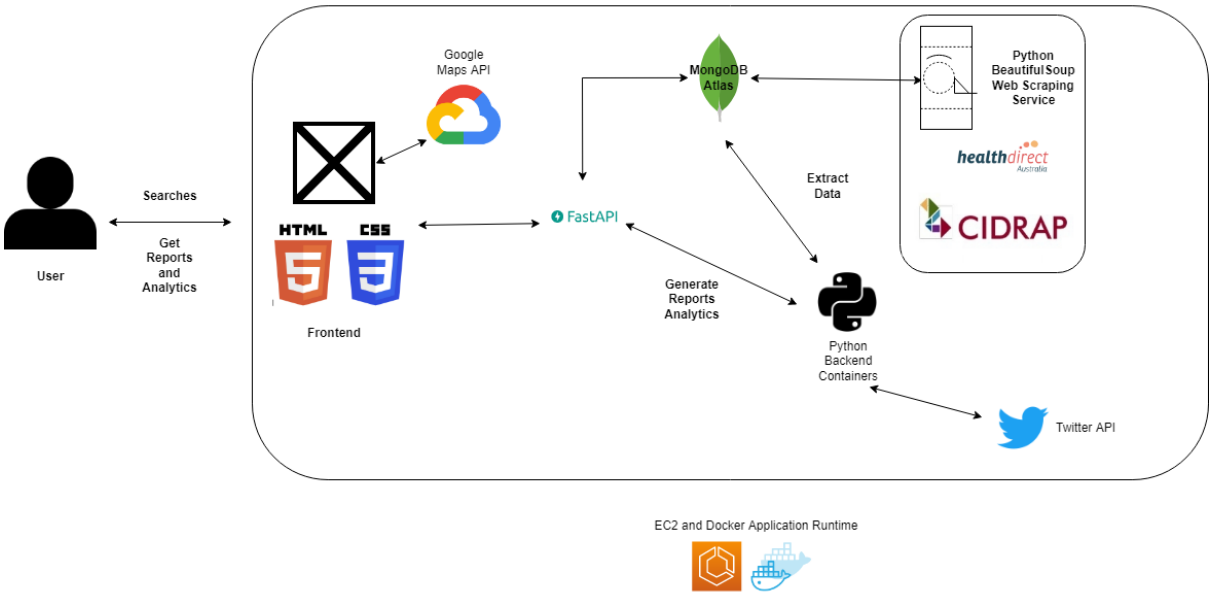
Login & Register:





A simple register and login page.
The register page will require a name, username, email and password.
The login page will just require the username and password.

Design



Frontend

With our Flask framework, we decided to go with **HTML+CSS** for the frontend and **vanilla JavaScript** to provide the logic for the frontend. This was due to the fact that many of us were fairly new to frontend programming, so we decided to keep it simple and go with a stack that has plenty of resources online to help guide us.

Initially we did consider frameworks such as React and Angular, however not all of us were well versed in those frameworks so we decided against it, despite its developmental advantages such as it having an open-source library with a plethora of tools and its ability for CI/CD.

Other libraries such as Bootstrap and jQuery were also considered. Bootstrap would have provided us with plenty of prebuilt components to use in HTML+CSS and jQuery would have given us a simpler way to manipulate HTML traversal and the display of dynamic elements, which are both key to our application. Both tools would have helped in the development of our application, however for simplicity's sake, we decided against it as none of us have had prior experience with these libraries. We also thought that it wouldn not enhance our application immensely to the point where the time it took to learn it would outweigh the benefits of it.

The use of the **Google Maps API** was primarily due to the fact we initially designed in our prototype for an interactive map and Google Maps is one of the most well known APIs with plenty of documentation and support to aid us in the production of the site.

To manage our frontend, we used the **Flask** framework to help us with the functions of a web app. Navigation was primarily done through flask, as well as calling upon external APIs such as the Google Maps API as well as our own. Data was processed either in the Flask python file or parsed through to the frontend's JavaScript. The biggest reason why we chose Flask over Django, the other popular framework, was the fact that Flask can support APIs whilst Django does not. Flask is also traditionally more suited to smaller web apps, such as this project, so it was more suitable than Django.

Backend

For the backend server of our application, Python and Fast API were used. This was primarily due to the ease of use for both frameworks, with all of our team members having previous experience with Python development, and the use of asynchronous request processing benefits of Fast API, as well as maintaining API documentation.

The application required multiple sources of information to be fetched from the backend, which were as following:

Information Source	**General Purpose**
User Authorisation	Allows for customisation of the service through user specific settings, such as subscription tiers or service variability catered to their needs
Disease Reports	Primary information from which the other services are built upon. Based on the number of reports for a given disease, a list of common diseases at a location are found, which are then passed to other functions to use
Twitter API	Form of news and discussion content provided to the user to view the current state of diseases at a location
Disease information	Primary information on disease prevention, vaccination and medications. Shows specific details on how to prevent the spread, what vaccinations are available and what medications to take for a specific disease.

User Functions

To use our product, users will need to register an account the first time they use it, after which they can log in to their account directly to access the product.

Once on the main page, the user is presented with a map which can be dragged around to select areas and view information about places or destinations of interest to the user. Our product will automatically grab the latest relevant tweets about covid-19 from users

in that location on Twitter and display them to the user. In addition, users have the option to click on a location on the map to get information about the disease in that area, including (travel restrictions, quarantine, vaccines and medication).

Users can enter the disease and region they are looking for in the 'Search' page, which will bring up news or articles about it. The search bar will also display the top five most searched diseases.

Disease Information

To retrieve information on diseases at certain locations, we first utilised our own API's search functionality, developing an alternate version to search through the database with a list of diseases at the location, and determine the most common diseases based on the number of reports found related to that disease. This was performed using python's request library, making a call to API and constructing a list based on the results.

This list of common diseases at a location could then be utilised to match the disease information, such as prevention, vaccine and medications, which were then passed to the frontend to be displayed in the necessary forms. Information for all of these were then retrieved from their own relevant scraper (as justified below under ***Scraper Implementation***), which were developed by our team. This was done as opposed to using external libraries in order to provide control over the results from the searches, allowing us to scrape sites that we knew had the desired information that we were looking for and thus allowing formatting to become much easier in the frontend.

Current News

In order to retrieve real time updates regarding certain diseases, we chose to implement a Twitter scraper using the snscrape library in Python. This library gave us the scalability to be able to scrape other social media sites such as Reddit or Facebook for future development, whilst also being relatively simple to use. To scrape relevant tweets, the scraper was simply passed the location, and would filter tweets that were within 10 kilometres of the given location, returning a json of relevant articles to the common list of diseases at the given location.

We also chose to parse the json into HTML within the backend using Python, as due to our given frontend framework, without JQuery it would be difficult to manipulate the HTML. For this we utilised the urllib library in order to convert the url string into a utf-8 string, and then the requests library to the Twitter API in order to retrieve the embed code for that URL.

Scraper Implementation

The scraper was implemented using the BeautifulSoup Python library. We chose to use BeautifulSoup as our main scraper as it turns the HTML on the webpage into a parse tree. Then, to navigate through the tree, we can call specific tags with their unique id and name to scrape the data that we require. BeautifulSoup is also the most beginner friendly Python scraper to use and implement, and most suitable for smaller projects.

The articles presented in our database were scraped from CIDRAP. The articles in CIDRAP are all clearly sorted by disease, date, and country, making it easy to extract the data required and sort the data accordingly in our database. CIDRAP also allows for filtering out specific requests. Since we were mostly looking for articles to scrape, CIDRAP can filter out scans on the page. CIDRAP has the countries that are used in their articles displayed in their 'By Country' filter. This allows us to scrape the list of countries used in their articles. Within the articles, the disease/s that is referenced in the report is shown on top of the page. To scrape the cities and the event dates, because the data was all in the article itself, we decided to use regular expressions to match the data we need. If there was no event date in the article itself, the date of publication was used for the event date instead. For cities, a list of cities downloaded from the internet was used as a reference to match using regex.

The disease information such as prevention and vaccine was scraped from the Healthdirect website. On Healthdirect, most pages on the disease information are separated into different headings, with easily identifiable ids. For example, on the mumps information page, the article is split by different ids, with information on prevention below the id prevented, and treatment being under the id treated. To extract the information, BeautifulSoup's find function was used to find the section and then the paragraphs of text that accompanied it were scraped. For some diseases, the CDC website was used instead as Healthdirect did not provide sufficient information. For avian flu, the Healthdirect website instead links to articles on different sites instead of having the data on site.

Platform

Our platform was planned to be native on AWS EC2 Container Services, because of a few key things which allow for a clean integration of CI/CD for automated testing, deployment and scaling.

The IaaS and PaaS paradigms are present in this service, which enables us to both develop our own platform but configure it in a much more intuitive way. This allows for faster development, focusing on the core product rather than infrastructure set up. We just simply set up the services and develop them. However cost can also be considered, and we can restrict certain resources which may not anticipate much use and traffic, but have the flexibility to expand our services further if needed. Docker is also a popular runtime which enables optimised instances running for its desired task with minimum

overhead.

However these deployment and testing capabilities weren't fully realised in Phase 2 due to project constraints as elaborated on in our Project Management Report, as resources needed to be dedicated to entirely focusing on the bottom line for the application.

CI/CD

Although initially planned from Phase 1, utilising services such as Docker, AWS ECS, AWS Codedeploy and GitHub Actions were planned to be used to automate all testing and deployment of the code.

As of now the application uses a manual approach to upload our code and facilitate it onto a EC2 Ubuntu 20.04 hosting an NGINX server as a reverse proxy for our backend and frontend code. This option was chosen instead due to its simplicity to upload and test our code manually.

If the application was fully realised, the initial platform design would have been executed.

Testing

In terms of testing, we opted for a manual blackbox testing approach, as our product is focused on functionality and it does not require testing the logic of the code like white box testing does, to simply measure our baseline capabilities.

In D2, we first tested the logic of our code, corrected the code according to the output from the server, and finally tested the functionality of the backend code. In D4, we wrote another http request based test script to test whether the client could be accessed and to determine what the problem was by the status code returned.

We opted for the manual, logic and response testing, due to constrained resources to further delve our testing ideology into maintaining the Quality Assurance of our features. If resources were not so constrained, then non-functional testing scripts for frontend and backend could have been developed.

Future Feature Extensions

Within the scope of the project, our team aimed to focus on functionality, and thus much of our development time was focused on creating a solid foundation for our backend functions to be extended in the future. For example, our scrapers are able to parse from a list of 70 common diseases, many of which rarely appear in the reports. As such, due to time constraints, certain features were prioritised in development of our MVP, and thus

with more time to develop the application, our product's functionality could be improved upon greatly.

In regard to current features, there are numerous improvements that could be made to improve delivery of certain functions. The dashboard currently takes a long time to load (~5 seconds) due to needing to call the database for every country in the world, which would be frustrating to interact with from an end user perspective. As such, before final deployment, this would be improved through automating the process, storing each country's related reports and diseases directly to the database, and retrieving results from the database rather than having to query each article in the database for the location and relevant disease.

Another change would be improving the report generation of our search function. As of now, the search function simply displays the results from our disease scraper, with the relevant title, URL and diseases that were found in the article. The original aim of the search function was to provide additional detail at a given location, and as such, cater towards the business traveller group. This could be extended through providing a summary of each article, displaying relevant information that transforms the functionality from a simple search engine into a tool that generates a separate use case for those seeking detail, but in a summarised form. In addition to the articles, it is also important to display information regarding the disease, and specific to business users, health regulations at the location would be another extension possibility. This would be extended through our scraper to search the country's government health website and summarise the relevant information regarding the regulations.

Aside from existing features, there were also many personalised features for each user group, which were intended to be developed in our initial business proposal, that were ultimately cut from our MVP. With more time to develop in the future, these features would act to create stronger distinctions between each user group and their intended use cases for our product, and thus improve the overall user experience through a more customised experience based on the user's needs.

One such example of this as outlined in the Use Cases and Requirements report is the notification service, which would deliver phone and email notifications to the user about local health threats based on their location. This would cater towards the frequent traveller, who intends to use our service in order to receive information about how they may stay more safe in their travels, and as such, real time updates would act as a service to keep them updated during their travel, as opposed to the casual user who may only seek preventative measures.

Another feature that would provide a similar benefit is the provision of nearby health clinics, based on either where the user currently is or where they intend to travel to. These recommendations would streamline the service to not online provide passive

information regarding the diseases, but give the user an active way to seek assistance with the disease, both during and in preparation of their travels. This would act to extend our application from a simple information source into a live service that travellers can use whilst they are out.

Business Model

MediTravel offers three different services depending on the type of traveller the user is and will utilise a subscription base model to generate revenue and profit. There are three tiers, which go up in subsequent levels to encapsulate users basic to niche needs.

Tier Level	**Description**	**Justification**	**Price**
Freemium	- Travellers use the app once in a while - Limited to ten searches per month - Allow report and dashboard generation per country	Serves the absolute basic needs of all travellers to grip onto the service (refer to the Use Cases and Requirements Report). User data collection could be used to gain profit.	AU$0
Basic	- For frequent travellers who need to search up country information frequently - Dashboard is extended to show social media (tweets) related to diseases at the destination - News information appears related to the diseases at the destination	Serves the needs for frequent travellers (refer to the Use Cases and Requirements Report) that need this information frequently.	AU$10
Premium	- To meet the demands of specialised business travellers who need deeper information on where they are travelling to and how outbreaks are	Serves the needs for specialised business travellers (refer to the Use Cases and Requirements Report) that need customised information	$100

	impacting their work - The platform becomes more of a data analytics and information hub, customisable to their needs. - Gather notifications about disease and its impact at their chosen locations.	frequently.	

The pricing models are based on affordability and the demand of users requiring certain features. Although to note these are approximate pricing, further clarification of business costs would need to be delved into to find out operating costs, preferred profit margins and expected user demand.

Conclusion

The overall service we have produced at the end of Deliverable 4 is a reflection of the MVP we have identified from the Use Cases and Requirements report. Whilst there are many extensions and desirable features which could be added to fulfil the requirements better, the current implementation represents the core functionalities which are necessary as they apply to all users. As such, there exist solid foundations upon which more personalised features can be added in order to more specifically cater towards our target user groups, and whilst not fully implemented, the ideas have been formulated and with more time, would be relatively simple to implement.

Through this, the business use cases will be satisfied, thus providing a service that we believe will alleviate the shift in public perception towards diseases that has occurred through the COVID-19 pandemic, and assist in facilitating international business activity again.