Department of Computer Science

Loughborough University

BSc Computer Science

20COC251

Developing a 'COVID-19 aware' Flight Finder Web Application

Author:

Nana Aboagye

Supervisor:

Dr. Hossein Nevisi

May 2021

Abstract

The COVID-19 pandemic has forever changed our outlook on global travel and will continue to affect how we book travel online for the foreseeable future. Travel bans, rising surges of COVID-19 cases, and lockdown restrictions, amongst others, mean it is no longer as simple as deciding on a destination and subsequently booking a flight. The travel booking landscape is now a maze that requires effort on the holidaymakers' behalf to navigate through, and current solutions do little to assist them with it. Current solutions lack awareness of COVID-19 related restrictions affecting travellers, and don't incorporate COVID-19 information within their application to advise users, resulting in users scouring the internet through numerous government websites looking for answers.

In this project, the gaps within other solutions were explored to form requirements for the development of a 'COVID-19 aware' flight finder web application, that itself, provides users with information and data regarding the COVID-19 situation in each country, to assist holidaymakers with the newly surfaced challenges the global pandemic has brought. These requirements were further developed upon through the testing of wireframe prototypes by a focus group of potential users, and this feedback was used to design the actual system that would be implemented. Using Amazon Web Services extensively, a serverless cloud web application was built, with React.JS used for the front end and Python for the back end. Through the use of CI/CD, iterations of the application were continuously deployed and tested by end users, with their feedback being incorporated into future iterations. In the end, a web application that could potentially be commercially viable was created, solving the aforementioned issues.

Contents

Acronyms

API Application Programming Interface. 24, 60, 61

AWS Amazon Web Services. 64, 65

CI/CD Continuous Integration/Continuous Deployment. 59, 98, 108, 110

CORS Cross-Origin Resource Sharing. 89

CSS Cascading Style Sheets. 60

DOM Document Object Model. 28

FaaS Function-as-a-Service. 29

HTML HyperText Markup Language. 27, 60

HTTP HyperText Transfer Protocol. 29, 64

JS JavaScript. 28

JSON JavaScript Object Notation. 68

REST Representational State Transfer. 29, 65, 89

SPA Single-Page Application. 28

UI/UX User Interface/User Experience. 11, 112

UML Unified Modeling Language. 48

Web App Client-side, internet facing application, running on a remote web server. 28, 43

XML Extensible Markup Language. 63

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Problem Domain

Due to the COVID-19 pandemic, many countries around the world are imposing restrictions on travel into their countries and currently, it is very difficult for holidaymakers who still wish to travel to, find information on these restrictions, without having to browse through numerous government websites and news articles. Many potential holidaymakers see this as a blocker and instead decide not to go on holiday.

This project's aim is to develop a web application that will better serve travellers during the pandemic, helping them to clearly view the restrictions imposed on the places they want to travel to, whether they will have to quarantine when they return from these locations and more. This application will effectively be a 'COVID-19 aware' flight and hotel finder, with a dashboard tailored to users based on the country they reside in, and using a clear User Interface/User Experience (UI/UX), it will provide users with information on the COVID-19 restrictions of the destinations they want to travel to, all in one place.

1.2 Aims and Objectives

The goal of this project is to help holidaymakers access COVID-19 restriction information more easily, within the same interface they would use to find and search for their flights and hotels, so as to better inform their decisions. Users will be able to search for a location they wish to go to and instantly see information on restrictions based on their country of origin and the destination country alongside flight and hotel prices. Recommendations of destinations will be made based on the origin country of the user taking restrictions into account.

The objectives of the project are to:

1. Carry out a literature review in order to find a suitable technologies to use for this project

2. Carry out a gap analysis researching the pros and cons of similar applications and systems in industry
3. Source and collate accurate data regarding each country's COVID-19 situation to use within a web application
4. Develop a web application that will both serve holidaymakers and help them make better informed decisions on their travels in the pandemic
5. Build a solution that is, or has the potential to be, commercially viable and have real-world value

1.3 Motivations

The primary motivations for undertaking this project are:

- To help holidaymakers make better informed decisions on their holidays during the pandemic by helping them access information on restrictions more easily
- To serve end users by helping them find flight and hotel prices alongside restriction information all in one place
- To learn new skills in Data Science and Machine Learning using Python libraries amongst others
- To utilise and improve my cloud computing skills gained during university and on my placement

1.4 Potential Risks and Constraints

As with any project, there are a number of potential risks and constraints that can affect this project. A quantitative risk assessment method will be used where the risk rating will be the probability of the risk multiplied by its potential impact. These have been identified as:

- **Not collecting enough data to train the machine learning models.**
  This could have an adverse effect on the web application's ability to accurately determine whether a country is restricting international arrivals and will cause the application to be ineffective, so this can be seen as having a high potential impact. The likelihood of this occurring is also quite high too, as data like this isn't necessarily readily available through APIs, so web scraping techniques would have to be applied to source this data.
- **Collecting unusable data due to inaccurate web scraping.**
  Web scraping can sometimes be unreliable, and the data may not be presented in a way that is suitable for it to be used as data to clean and use in a machine learning model. This has a relatively

low chance of occurring but, if it does occur, it will have a high impact on this projects success, as it is very reliant in having good quality data and having it in abundance.

- **Not being able to accurately report on a country's restrictions.**
  This is important as the web application will not be reputable if it is not in line with the information found on government websites. Providing false information on COVID-19 restrictions is also unethical, so this must be avoided or at least managed as best possible. The possibility of this happening is medium and the impact it may have is very high.

- **Scraping for data from websites being within the ethical guidelines.**
  While it is legal to scrape websites for information in most cases, there are varying cases for each website, and care must be taken for each one. By not complying to a website's restriction as set in their robots.txt file, access could be possibly restricted from this website and in so doing restrict the size of the data set meaning this could have quite a high impact, but the probability of this happening is very low when being responsible, such as keeping requests to 1 request at an interval of 12-15 seconds for example.

Scale used: 1 – Very Low, 2 – Low, 3 – Medium, 4 – High, 5 – Very High P = Probability I – Impact

Table 1: Quantitative Risk Assessment of risks and constraints the project could encounter

Risk	**P**	**I**	**Risk Rating**	**Counter-measures**
Not collecting enough data to train the Machine Learning Models	4	4	16	This risk can be managed by collating data from a wide variety of source not limited to just government websites but also national news websites, and national tourism websites.
Collecting unusable data due to inaccurate web scraping	2	4	8	By using APIs where possible, or when web scraping, avoiding the use of Optical Character Recognition (OCR) techniques, the risk of not collecting usable data is mitigated.
Not being able to accurately report on a country's restrictions	3	5	15	Providing disclaimers that the information may not be entirely accurately, and official government websites should always be used as a first point of call.
Scraping for data from websites being within the ethical guidelines	1	4	4	Complying to a websites rules as set out in the websites' robots.txt and keeping to a strict set of guidelines when web scraping such as keeping requests to 1 request at an interval of 12-15 seconds for example and not republishing scraped data without permission.

Chapter 2

Literature Review

2.1 Introduction

These days it is far more common to book a flight or hotel online than any other existing method – travel agents, over the phone etc. It is estimated 83% of US adults now prefer to book their travel online, with a whopping 148.3 million travel bookings made online every year [1]. It is very evident that this is now the primary way to book a flight or hotel, and due to this, there is a wide array of options available to the public, such as travel comparison sites like Agoda or Kayak, online travel agencies like Expedia, and the travel providers' own websites such as British Airways.

In this Literature review, an in depth look into some of these different online options of booking travel will be made, along with an attempt to find gaps in the current online travel booking landscape, which this solution can be built upon. These different existing options will be assessed based on a number of factors, including but not limited to, usability, how COVID-19 aware' it is – does it give clear information on current travel restrictions due to the COVID-19 pandemic, and how involved this information is in the overall booking process.

2.2 Gap Analysis

2.2.1 Criteria to be used

Detailed below is the criteria that will be used to help assess each existing solution in the form of questions:

1. Does this solution give clear information in regard to the COVID-19 restrictions for the user's destination within their own website?

2. Does this solution factor in travel restrictions into search results (i.e. highlighting in the results if

the user's chosen destination country allows international visitors or is on lockdown)?

3. When it comes to usability, how difficult is it for the user to access information on COVID-19 restrictions if available? – measured based on number of clicks, pages navigated, time taken etc.

2.2.2 Flight and/or hotel Finder

Agoda – Travel Comparison Hotel Specific Solution

Agoda doubles up as an online travel comparison search engine and travel agency offering over 2 million properties globally [2]. More emphasis is placed on hotel accommodation, long-term rentals and non-hotel accommodation, however, in recent years they have also added the ability to book flights directly through their website too. It also provides details of places of interest such as landmarks and transport links nearby to the property, helping users make well advised decisions on what accommodation to choose based on their requirements.

Question 1

When it comes to this question Agoda seems to miss the mark. While they do have a page on the website regarding coronavirus [3] they only advise users themselves to check local travel restrictions prior to booking a trip as seen in the screenshot in Figure 1.

Figure 1: Agoda homepage COVID-19 status bar screenshot

Agoda doesn't give any information in regard to the COVID-19 restrictions in any specific country, so therefore doesn't meet the criteria in this first question.

Question 2

Agoda doesn't fare much better in this question as well. It doesn't factor in any specific coronavirus travel restrictions into the search at all. It however does give information on the hygiene measures in place, shown in Figure 2, for some of the properties, which can provide some assurance to potential users.

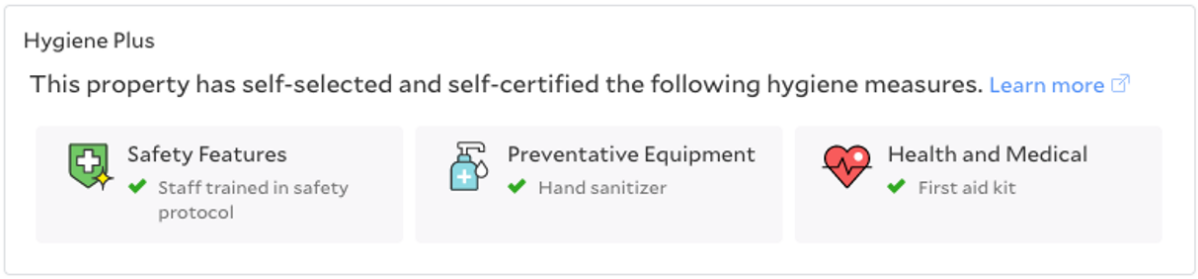


Figure 2: Agoda hygiene information screenshot

Question 3

- Number of clicks -1
- Pages navigated - 1
- Time taken - 5 seconds

Basing the usability of the website in terms of finding information on COVID-19 on these 3 factors, it is clear that the Agoda website has achieved a good level of usability. It took just one click and only one page navigated to find this information, taking around 5 seconds. Just under the navigation bar near the top of the website displays a status bar discussing COVID-19 briefly and link to 'Learn more' is given. Due to its prominent position on the webpage, it is very hard to miss. Information about coronavirus can also be found in the help section of the website, which can be found by clicking the link to it located on the footer of the homepage, as seen in Figure 3.

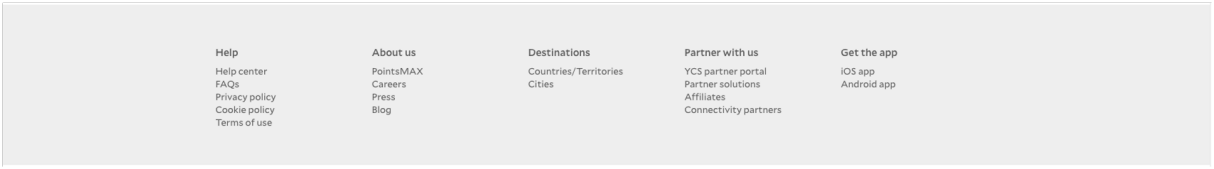


Figure 3: Footer of the Agoda website screenshot

Advantages

- Information of the hygiene measures in place at properties.
- COVID-19 policy easily accessible.

Disadvantages

- No specific travel-related coronavirus information.
- Cannot tell from website if a country is on lockdown.
- No indicator if the user's chosen destination country allows international visitors or is on lockdown.

Skyscanner – Travel Comparison Flights Specific Solution

Skyscanner is an online travel comparison search engine and travel agency with a greater onus on finding flights as opposed to hotels, although it does offer a feature to find the latter. Its site is available in 30 countries and is used by over 100 million people every month [4]. Its services also include car rental, and it has a search feature called 'Search Everywhere' where it shows users the cheapest destinations they can fly to from their airport and country of origin. Users are also able to track prices of flights with alerts of price changes sent to the users' email.

Question 1

Skyscanner definitely fits this criterion and it does so very well. It has a live COVID-19 travel information map [5] where users can click on their desired destination country, and are given an in-depth summary of travel restrictions to that country as shown in Figure 4.

Figure 4: Screenshot of Skyscanner's live COVID-19 travel information map

For each country, information such as whether or not users would have to quarantine on arrival; the proportion of COVID-19 cases out of 100,000 people, and its percentage increase from the last week; and the country's detailed restrictions policy, is available. Users are also able to sign up for travel updates for a country in case their policy changes. Countries are grouped into four main categories: Low restrictions, Moderate restrictions, Major restrictions and Restrictions unknown.

Question 2

Skyscanner achieves this to some extent, clearly specifying if the traveller will have to quarantine on return from a country, factoring in the traveller's country of origin. This feature isn't in their main search function, but through the main COVID-19 travel restrictions page, there is a search feature where users can search for destinations to go to where there are low to high cases, no quarantine on arrival, and are accepting visitors. Within the main search results, Skyscanner has a feature where it discusses the airlines COVID-19 safety measures, and whether the airlines have a flexibility or cancellation policy, seen in Figure 5.

Figure 5: COVID-19 safety measures on Skyscanner's website

Question 3

- Number of clicks - 1
- Pages navigated - 1
- Time taken - 2 seconds

It is clear Skyscanner has put a lot of thought and care into the usability of its website when it comes to finding information on coronavirus. It makes use of a banner at the top of its homepage entitled "COVID-19 travel info". This banner offers a button that takes users to the aforementioned live COVID-19 travel info map. Similar to Agoda, information on COVID-19 can also be accessed by going to the help section, where there is a link to in the footer section of the website.

Advantages

- Clear visual live map displaying countries with colour coded restriction levels set by the country of origin.
- Information on a country's cases per 100,000 people in the last week and the rate of increase.
- Displays conditions of a country's current travel policy in relation to COVID-19.

Disadvantages

- Unclear if a country is on lockdown or not.
- COVID-19 information on restrictions is not displayed within search results, and the user has to navigate to a separate page to find this information.

Expedia – Online Travel Agency

"Expedia is one of the world's leading full-service online travel brands helping travellers easily plan and book their whole trip with the widest selection of vacation packages, flights, hotels, vacation rentals, rental cars, cruises, activities, attractions, and services" according to [6].

Question 1

When looking at hotels or flights for a destination, Expedia does suggest that the location may have travel restrictions, and clicking on the 'Find out more' link leads users to France's official government advice on travel, as shown in in Figure 6. Similar results occur for other destination countries, taking users to that country's official government travel advice.

Although this information isn't directly available within Expedia's site, this is still an extremely helpful feature, and saves users having to scour the internet for their destination country's government advice on travel.

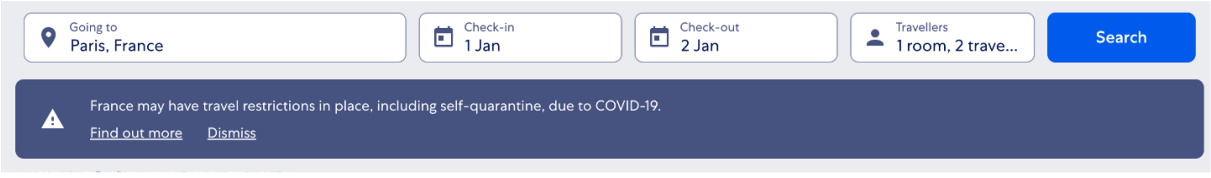


Figure 6: Expedia's search results COVID-19 banner

Question 2

Expedia only offers information on the destination country's travel restrictions within the search results and doesn't consider or provide information on whether holidaymakers may have to quarantine on return to their country of origin.

Question 3

- Number of clicks - 3
- Pages navigated - 2
- Time taken - 15 seconds

The Expedia website in comparison to the previous two solutions, proved more difficult to access information on COVID-19 with. Unlike the others, there was no banner of sorts on the homepage discussing COVID-19, and the footer also didn't have a link to a help page. At the top of the page, a link to a 'Support' page was found in the navbar and clicked on. After clicking on the relevant option on the accordion menu, information on coronavirus was located.

Figure 7: Screenshot of the footer of Expedia's website

This way seemed a little long-winded, instead of being immediately presented with information on COVID-19 via methods such as banners or pop-ups, it had to be searched thoroughly for. In retrospect, it can be seen that there is indeed a link to the 'Support' page located in the footer of Expedia's homepage, however this link finds itself among many other menu options and inside a footer with a lot of other text and information as shown in Figure 7. This makes its quite hard to see initially, and the footer menu text colour being the same as that of the actual text doesn't do much to differentiate both, further causing issues to the user in terms of usability.

Advantages

- Provides links to pages of a users destination country's official government travel advice.

Disadvantages

- Not clear if the destination country is on lockdown.
- Difficult to find Expedia's general policy and information on coronavirus.

British Airways - Travel Provider Website

British Airways is the flag carrier of the UK and transports over 40 million customers a year [7]. It is possible to book both flights and hotels on their website and they also offer car rentals. They have a rewards scheme called "Executive Club" where users can collect points called Avios.

Question 1

The British Airways website manages to provide very generic information on COVID-19, but it fails to provide specific information for a country a user may wish to book a flight to within its own website. It does however provide links to the relevant government websites in order to find this information.

Question 2

British Airways' website only mentions information on travel restrictions on their COVID-19 travel and service updates page seen at [8]. It doesn't offer any information regarding COVID-19 or travel restrictions within the search results.

Question 3

- Number of clicks - 1
- Pages navigated - 2
- Time taken - 5 seconds

In terms of usability, when it comes to accessing information on COVID-19, the British Airways performs strongly. Just under the navbar on the home page is a banner prompting the user to "find out more" on COVID-19 updates. This means information is literally one click away and only 2 pages need to be navigated. This banner can be seen in Figure 8.

Advantages

- Very simple and quick to access information on COVID-19.

Disadvantages

- Fails to give any information on a country's travel restrictions within their own website.
- Information on coronavirus isn't incorporated into search results at all.

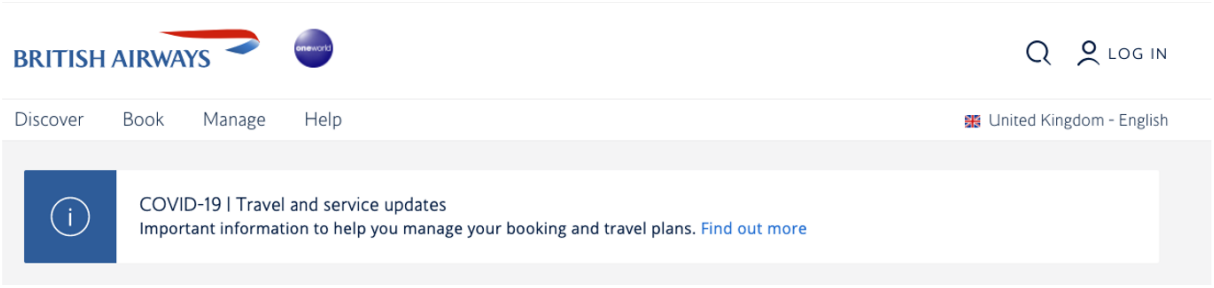


Figure 8: Screenshot of a banner on British Airways' website

2.2.3 Conclusion on existing solutions

All of the solutions attempt to give some information to their customers regarding COVID-19, albeit it very minimal in most cases. They all seem to miss the mark in incorporating travel restriction information, within the search function, or search results of their websites. This is a gap that this project's solution will look to address, allowing users to adopt the booking process they're used to, not having to scour help pages or government websites which have an overload of information, to find information on the country they wish to travel to. Another gap that is aimed to be addressed is the lack of knowledge on whether a country is on lockdown or not. This will be a great benefit to the holidaymakers, as they won't be able to participate in their desired activities if a country has lockdown restrictions. Most of the solutions did well to make the information on COVID-19 very available and accessible with ease, a feature again desired in this project's solution.

2.3 COVID-19 Data and Travel Industry Restrictions Research

To be able to provide users with accurate information and data on COVID-19 and its related travel restrictions, it was required to find the typical information given by governments on travel restrictions and COVID-19 data. There's a lot of data given by governments, so it is important to choose data which will be most useful to users in making decisions on travel. The UK government's current travel restrictions will be used as the baseline data, as it can be envisioned most users will be departing from the UK. These restrictions are correct as of 20th December 2020 and are subject to change.

Baseline Travel Restriction Information

As stated at [9]:

- If you test positive for coronavirus you are likely to need to get treatment locally and stay there until you have recovered.

- If you live in a Tier 4 area in England, you should not travel abroad. You can only travel internationally where you first have a legally permitted reason to leave home i.e., work or education purposes

Tier 4 areas can be found on the UK Government website [9].

Baseline COVID-19 Data

The following data is sourced from the UK Government website [10]:

- Cases by specimen date (daily): 32,461
- Recent 7-day case rates by specimen date: 345.3
- Daily deaths with COVID-19 on the death certificate by date of death: 547
- Cumulative deaths with COVID-19 on the death certificate by date of death: 87304

Conclusion

With the data collated above, a good understanding of current restrictions and how it may all fit into the web application has been formed. It will form very useful in the design stage and testing. It is important to note that information on restrictions is subject to change very frequently, and can become out of date and useless at less than days' notice. This goes to show how useful this solution could be, helping users access and keep track of the ever-changing information, to help them form decisions on their travels.

2.4 Technology Research

In this section it is aimed to research into and ultimately decide upon the technologies that will be used in this project. The type of technologies that will be needed are outlined as server-side (back end) technologies, Application Programming Interfaces (APIs), client-side (front end), web scraping and cloud services.

2.4.1 External APIs

Part of the web application's intended functionality is to return results on flights and hotels based on a search query. In order to do this, it is planned to utilise and hit the endpoints of APIs for both flights and hotels and return the received information to the end users. Research will be conducted into some of the APIs available, so the best fit for this use case can be chosen.

Flights APIs

Skyscanner Flight API

"The Skyscanner API lets users search for flights & get flight prices from Skyscanner's database of prices, as well as get live quotes directly from ticketing agencies" as specified in [11]. This would serve this use case well as an API is needed that lets users search for flights and their prices. Through RapidAPI, the Skyscanner API is free to use as long as 50 requests per minute aren't exceeded which should be fine for this use case.

Upon further research on Skyscanner's own website, it was discovered that the same functionality could be implemented with their Flight Search Widget found at [12]. This could save a lot of time in creating front end code logic to display the data received from the API, however it has its downsides in that the search takes users to Skyscanner's own website. Earlier in the literature review in the Gap Analysis, it was aimed to find gaps in Skyscanner's approach and they can be seen as a competitor, so it wouldn't be a good decision to defer to them completely in order to provide this functionality for the users. Users may assume they're better off just using Skyscanner's own website instead of this project's solution.

Amadeus Flight Offers Search API

The Flight Offers Search API offered by Amadeus can be used to query over 500 airlines based on the given itinerary [13]. This API will provide a list of flights for the itinerary including their price, airline names, departure terminals and more. This suits this use case well and will help provide end-users with a comprehensive flight search feature. It is free to use this API as long as there are less than 1 request every 100ms and less than 2000 requests per month. A request quota of 2,000 a month should be appropriate to begin with. It is important to note that flights from low-cost carriers and American Airlines aren't available currently within the API.

Conclusion

Both API solutions suit this project's needs, however, after researching both more in depth, specifically the response payload they return, it was discovered that the Skyscanner API doesn't provide the flight departure and arrival times, whereas the Amadeus option gives the necessary departure and arrival times. For this reason, it was decided to go with the Amadeus API as this will better aid the flight searching experience for the end users. It is important for them to know the flight length and when the flight is leaving and arriving.

Hotel APIs

Hotels.com unofficial API

This API allows users to query hotel rooms and their information, including but not limited to price, images, local amenities, reviews, and transport links. It is an unofficial API for the Hotels.com website, so there is a great range of hotels available. The API is free if requests don't exceed 500 a month and it has a rate limit of 5 requests [14]. This API will definitely suit this project's needs well, giving access to a wealth of information.

Amadeus Hotel Search API

The Amadeus Hotel Search API offers deals at over 150,000 hotels globally and also gives detailed information on each hotel [15]. Its offering is very similar to what the Hotels.com API has, however, this API allows queries involving more filters for the search. This will facilitate a search feature that is very customisable and allows users to really filter the search to their needs.

Conclusion

Both APIs allow queries with many filters which will help create a really quality search feature in the web application, however the Amadeus API was superior in this case, and its return payload is also a lot clearer to read and understand, which will help when parsing the data into the front-end of the application.

2.4.2 Web scraping

In order to gain access to data that is provided by government websites and news articles web scraping will be utilised. Few government websites offer APIs and the data collected from news outlets such as BBC News and Sky News, who mostly don't offer APIs, will likely be unstructured data.

Selenium

Selenium is a Python library made initially to facilitate the automated testing of web applications [16], but is now used for other purposes such as web scraping. Why it may be useful, is that it is capable of scraping dynamically populated web pages, which is a category a lot of modern websites fall in to. It can also mimic human like behaviour and click on buttons and interact with other modern web page features such as carousels.

Beautiful Soup

Beautiful Soup is a Python Library that can be used to parse HyperText Markup Language (HTML) pages and extract data from them conveniently [16]. When used in combination with other Python packages such as *requests*, it becomes a very useful tool in capturing data from web pages. It also works well with poorly designed HTML pages, which could serve this use case well.

UiPath

UiPath is a company that offers Robotic Process Automation (RPA) tools and solutions, and it offers web scraping software too. It claims to be 100% accurate as alluded to in [17] and it is a solution that requires no programming, which could save time in the long run. Like Selenium it is adept at interacting with modern web page features.

Conclusion

All three are good solutions that would serve well, however Beautiful Soup was chosen which will be used in combination with other Python packages such as Requests and Pandas. This will allow for the parsing and cleaning of the unstructured data from the various news outlets and websites and make it useful. UiPath would've been another good solution, but beyond a free trial it is a paid service so doesn't suit this project's needs. Selenium may still be utilised in the case where Beautiful Soup and Requests aren't fit for the task.

2.4.3 Client-side framework

Choosing the right front-end framework is always important. This is how users will ultimately make use of and interact with the web application. Three JavaScript frameworks/libraries have been chosen to review and decide between. Vanilla JS (plain JavaScript) and jQuery has been excluded as this involves everything being written from scratch and reinventing the wheel to create the end product.

React

React is "a JavaScript library for building user interfaces" according to Facebook [18]. It is the most popular framework/library out of the three to be discussed, and it is still being maintained by Facebook. It claims to use declarative views to make code more predictable and easier to debug, and it also has the concept of components that manage their own state. With React, users have access to a vast collection of React code snippets and components, which can make building user interfaces faster.

Vue

Vue is a "progressive framework for building user interfaces" according to its official website [19]. It is said to be perfectly capable of powering Single-Page Applications (SPAs) when used with modern JavaScript (JS) tools and libraries. It is known for its lightweight size, which improves performance in the web browser and improves SEO (Search Engine Optimisation) and UX (User Experience). Vue also allows developers to write in vanilla (plain) JS instead of JSX (JavaScript XML) with React and TypeScript with Angular, so it builds on any pre-existing JavaScript knowledge well.

Angular

Angular is an "application design framework and development platform for creating efficient and sophisticated single-page apps", as stated by Google [20]. Backed and maintained by Google, it is another very popular JavaScript framework, and like React and Vue, it makes use of modular code and components and allows users to create templates. One factor that makes Angular different from the other two options, is that it uses the real Document Object Model (DOM) instead of using the virtual DOM. This makes it slower than the others usually and can sometimes mean users encounter several bugs.

Conclusion

It wasn't too difficult choosing what JavaScript library/framework to go for out of the three for the client-side of the Web App. Due to more prior experience of using React, it was the obvious choice as time would be saved from not having to learn something from scratch. This however wasn't the only factor in the decision. React compared to the other two frameworks is heavily supported by Facebook, and has excellent documentation which will help make the development process an easier one. An argument could be made for Angular which is backed by Google having similar benefits, however since it offers everything such as routing and testing utilities in its package, this can mean it is quite bloated in terms of its package size and whilst also keeping in mind the aforementioned performance issues it has due to its use of the real DOM, this could pose to be an issue.

2.4.4 Backend/bespoke API

Here the decision-making process in choosing the server-side (or serverless) framework will be discussed. A few different traditional frameworks were shortlisted, including a Python framework Flask, and Express JS, a JavaScript framework, but more specifically a Node.js framework. A cloud option of AWS Lambda was also shortlisted.

Express JS

Express is a minimal Node.js framework that provides tools for web and mobile applications [21]. At its disposal are a set of HyperText Transfer Protocol (HTTP) utility methods and middleware, such as Helmet and Morgan, which makes creating APIs quick and easy. It would also link quite well with the front-end framework choice of React, as they can both run on top of Node.js and are JavaScript based.

Flask

Flask is a "simple to get started" Python web framework that is very lightweight, but can be easily extended [22]. It is said to be well suited to beginners, as there is boilerplate code available to get a simple app started. It offers good integration with popular database choices, such as SQLAlchemy, and it is un-opinionated in the way it allows users to add different helpful plugins that suit their needs.

AWS Lambda

"AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers" according to [23]. With this Function-as-a-Service (FaaS) service, it's possible to write Lambda functions in languages such as Node.js, Python and more which could be very useful for this application, as it is planned to do the web scraping with Python libraries such as Beautiful Soup. By combining AWS Lambda with other AWS services such as an Amazon S3 bucket to store the front-end code, Amazon DynamoDB to store collected data and Amazon API Gateway to facilitate Representational State Transfer (REST) API calls to trigger the Lambda functions, a web app can be built more efficiently without having to worry about building a bespoke API with Flask or Express for example.

Conclusion

It was decided to push ahead and go with the serverless option of AWS Lambda as this will save time building an API from scratch and worrying about hosting code on servers. This time can instead be better spent working on the actual code - the web scraping scripts, the client-side UI and any potential extensions such as some machine learning to provide insights and analytics for the end users. Going serverless will also go a way in fulfilling one of the motivations for this project, found in Section 1.3, of utilising and improving on pre-existing cloud computing skills.

2.4.5 Illustration of the stack

Two diagrams were drawn to show what the technology stack for the web application could look like and how they link together. The first one, as shown in Figure 9, was drawn before doing any research,

based on prior knowledge, and the second, as shown in Figure 10, was drawn after being advised by the research conducted.

The first diagram was drawn out with a naïve approach. It was based entirely on prior knowledge and it resembles a typical MERN (MongoDB, Express.JS, React, Node.js) stack – without the M as MongoDB here – for this web app's architecture. In the initial diagram, no specific database (DB) was defined. While the MERN stack is a very popular stack for web apps, due to clear time constraints and a lack of resources being that there will be only a single developer, time can be saved by going for a serverless cloud method, and focusing more on the actual business logic code, which in this case will be the web scraping scripts and the front-end logic that will take into account information from the data scraped from the Internet. This possibility was only highlighted after conducting research on the various methods, which goes to show how effective the research process has been.

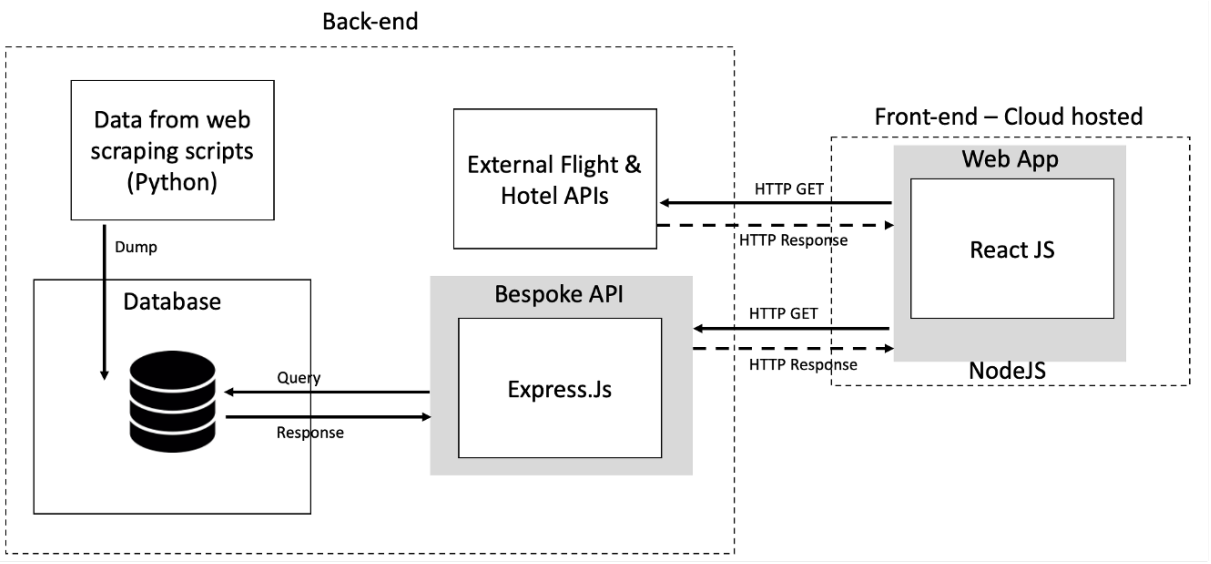


Figure 9: Initial proposed technology stack of the web application

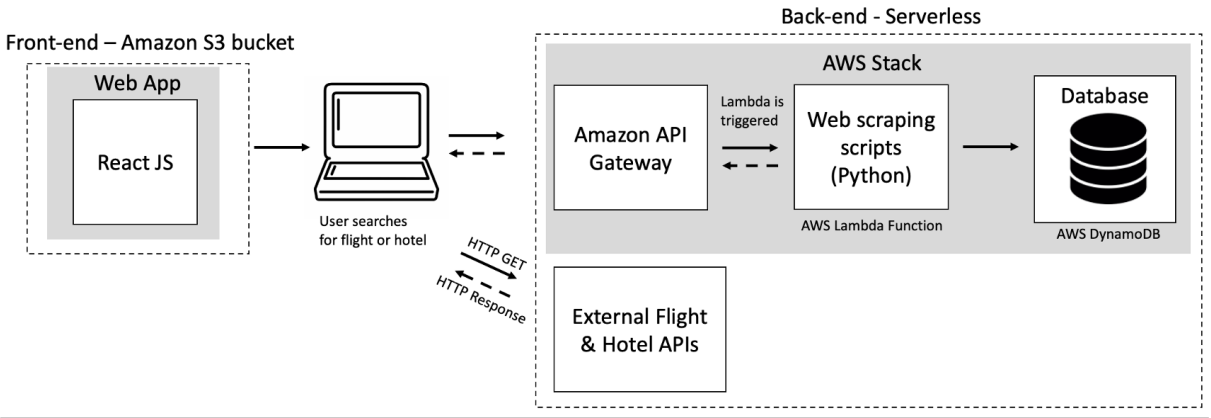


Figure 10: Updated proposal of the technology stack of the web application

The second diagram is a much improved version of the first. It is a lot clearer how each part of the architecture relates to each other and how information is passed from one component to another. After doing the research it was a lot more clear on how different services should link and relate to each other. User actions were also accounted for in the second diagram whereas with the first they weren't. Initially it was difficult to see where exactly the user fit in to the web application's architecture, but as more was read about the services AWS provides, this also became clearer.

2.5 General Conclusion

This literature review has been conducted thoroughly and will serve well as this project proceeds. It has also changed previous views on ideas on what technology architecture to choose, and it has highlighted the gaps in other solutions where this project can aspire to fill. It is now clear what technologies will be used to realise this project and how they will all relate to each other. By doing this literature review, a pivot has been made in the decision to keep the web application as a tool to find flights solely instead of both flights and hotels.

This decision was heavily influenced by researching into the hotel APIs available. It was quickly discovered the huge amount of data the response payload had for both, with the Hotels API [14] being a clear example, and it was realised that it would take a lot on the front-end code logic for all this information to be displayed. It would be counterproductive to continue to go ahead with including both flights and hotels, and excluding hotels won't affect the ability of this project to achieve its main aim and objective of helping holidaymakers' access COVID-19 restriction information more easily, within the same interface they would use to find and search for their flights, so as to better inform their decision.

Chapter 3

Requirements

3.1 Introduction

In this section, the process used to gather requirements, and analyse them in terms of priority using the MoSCoW method will be detailed, along with the reasoning behind some of the decisions in forming the requirements.

3.2 Requirements Gathering

Using the main discussion outcomes of the gap analysis, discussed in Section 2.2, an outline of the minimum, core and additional product functions were drawn out as specified below.

The product functions of the project are as follows:

Minimum:

- Users are able to search for flights and their prices and refine their search based on various COVID-19 information.
- Users will be able to see information on restrictions based on their country of origin and the destination country alongside flight information.
- Users will be able to check if their desired destination country has lockdown restrictions.

Core:

- Users are able to sign up to receive updates on a destination country in the case that its restrictions regarding COVID-19 changes.
- Information on COVID-19 will be accurate as of the day of being viewed.

- Travel restriction information will be incorporated within the search results.

Additional:

- Users will be able to see analysis and predictions using ML on if a country will enter lockdown in the near future or be added to their country of origin's quarantine list.
- Users will be able to receive recommendations of all the countries they can visit based on factors such as their country of origin, quarantine restrictions and lockdown restrictions.

3.3 User Stories

Using the product functions, outlined in Section 3.2, some example user stories were created to better divide the functions into different pages/views that can be used to create wireframes to be used as part of an Agile prototyping methodology to elicit requirements. An Agile prototyping methodology was chosen as it is an effective way of validating the requirements early on in the development, and can provide guidance for the design phase whilst also accommodating frequent changes in a project [24]. This means potential end users can see a rough idea of what to expect from the product, and can influence its progression, more than they can within the limits of a traditional rigid software development model.

3.3.1 Homepage/Main Dashboard Page

Summary:

This page should allow users to search for flights and refine their search using filters based on various COVID-19 information. It should also allow users to see suggestions and recommendations of all the countries they can visit based on factors such as their country of origin, quarantine restrictions, lockdown restrictions etc.

Detailed Functionality:

- Users can search with text(s) inputted into a search box.
- Users can apply filters to their search such as “”lockdown¿‘, ”quarantine restrictions¿‘, ”low number of COVID-19 cases“, and more.”
- Users should be able to view a grid of recommended countries for them to visit factoring in the current travel policy of their country of origin.

3.3.2 Search Results Page

Summary:

This view should provide the user with a list of results from their search query, displaying the flights. This view should also clearly highlight to the user whether their country of destination has lockdown and/or quarantine restrictions or if travelling to this destination is not permitted by their country of origin; all within the same view as the search results.

Detailed Functionality:

- Users can see a list of search results from their query, displaying flight options they can choose.
- Users can get a snapshot of information on their destination country's COVID-19 information without leaving the page.
- Users can click on a flight itinerary of their choice in the search results, and be redirected to the flight provider to proceed and book the flight if they wish

3.3.3 Update Sign Up View

Summary:

This view should allow users to sign up for updates regarding COVID-19 via email on a country of their choice they may wish to travel to.

Detailed Functionality:

- Users should be able to input their email address and preferred destination country, and then receive regular updates on their country of choice, regarding it's COVID-19 situation.

3.4 Wireframe Prototyping

Before the wireframes were created using Figma, low-fidelity prototype 'paper' sketches were created to help aid with designing the software version. They can be found in Appendix A. Low-fidelity prototypes don't look like or provide the same functionality as the final product but can help to explore ideas and are simple and quick to modify [25].

Whilst creating the high-fidelity wireframe prototypes, a decision was made to add a new *Recommendations Page*. The rationale behind this was that, whilst recommendations is included on the homepage, users may want dig deeper and see a snapshot of COVID-19 related information related to each destination, similar to that of the search results view discussed in Section 3.3.2.

A decision was also made to create a very similar view to that of the *Update Sign Up View* mentioned in Section 3.3.3, but to be used to get price updates on the flights of a destination. This was added to enable users to also get updates if the prices for flights for destinations they intend to travel to go up or down. Another page created was the Travel Advice page, which is an interactive world map colour coded according to daily COVID-19 cases, alongside other COVID-19 related information. It is intended to help users see on a global scale what the COVID-19 situation is like.

3.4.1 Wireframe Designs

Homepage/Main Dashboard Page

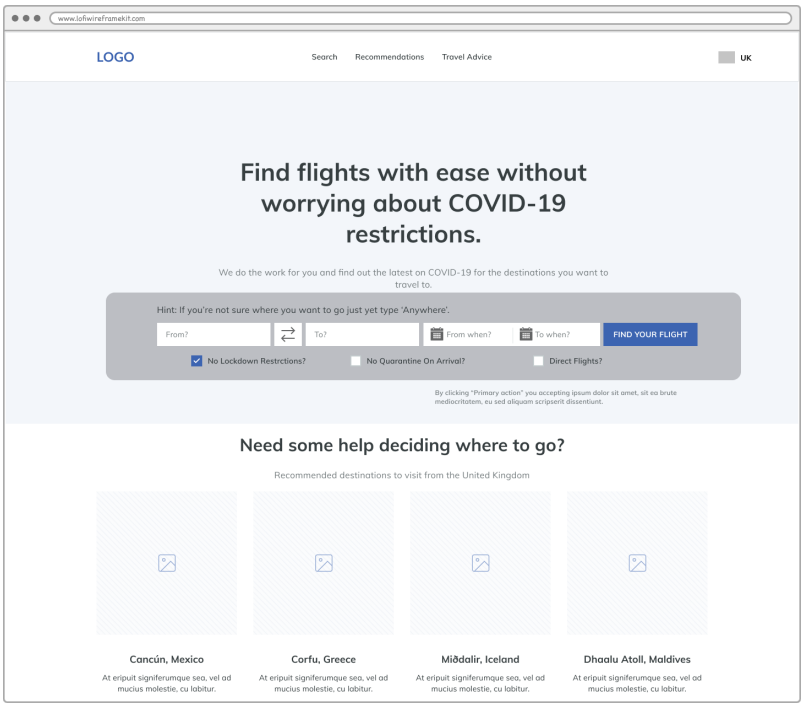


Figure 11: Main Dashboard Wireframe

Search Results Page

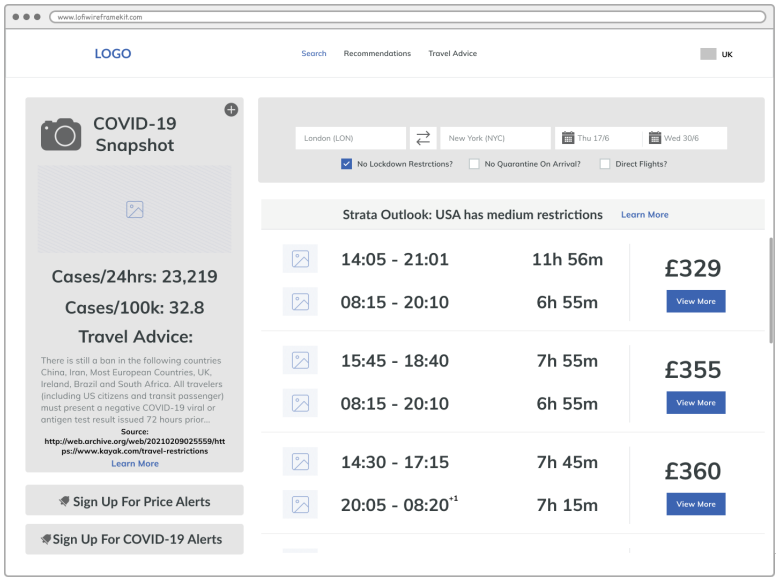


Figure 12: Search Results Wireframe

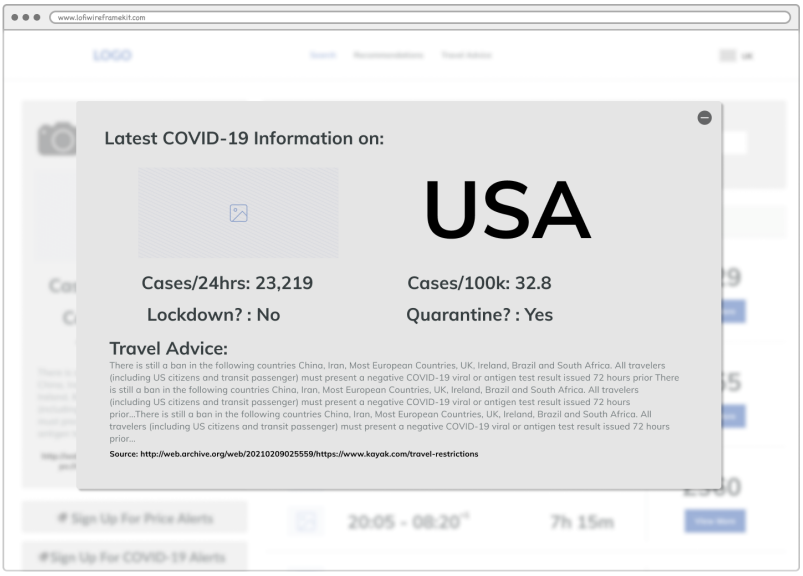


Figure 13: Expanded Snapshot Wireframe

Update Sign Up Views

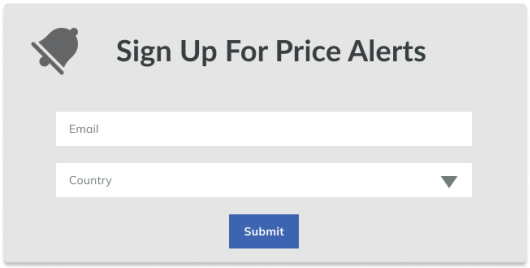


Figure 14: Price Alerts Wireframe

Figure 15: COVID-19 Alerts Wireframe

Travel Advice Page

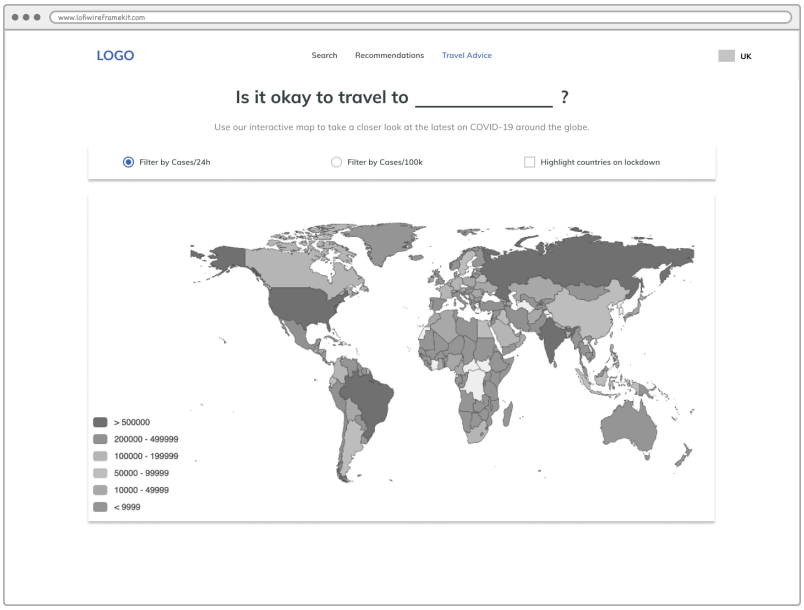


Figure 16: Travel Advice Wireframe

Recommendations Page

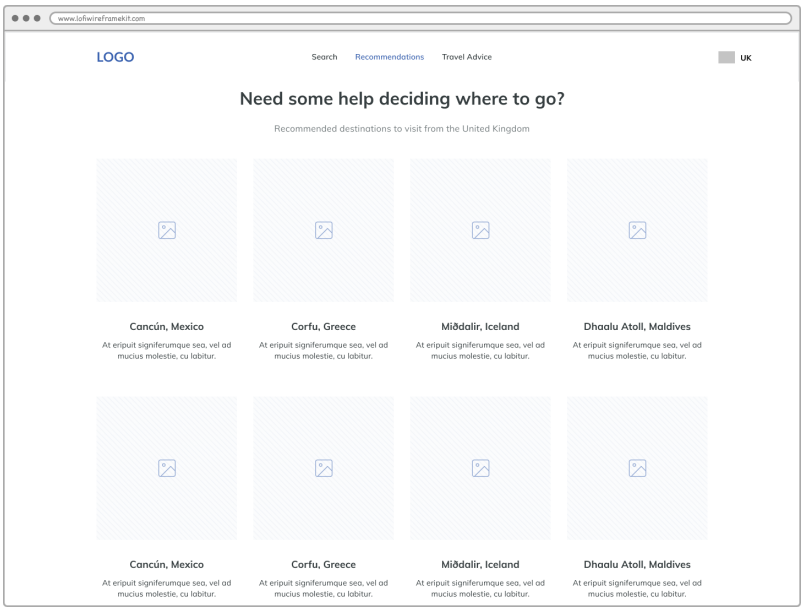


Figure 17: Recommendations Wireframe

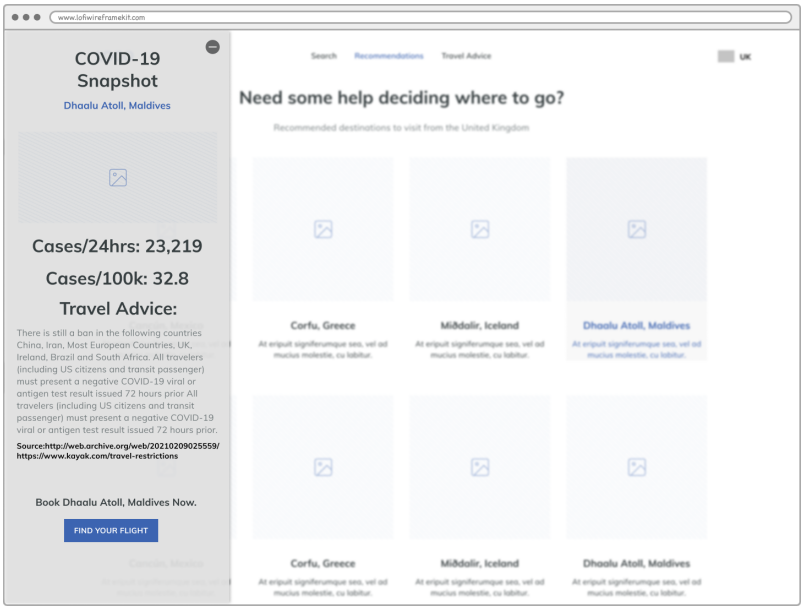


Figure 18: Recommendations with Snapshot Wireframe

3.4.2 User Testing

Aims of test

By conducting the following wireframe prototype test, it is aimed to narrow the communication gap between the end-users in deciding the system requirements. It is hoped, an agreement on what capabilities must be delivered in the final product, in order to fulfil a typical users' expectations, by using prototypes to present something that end users can react to, will be established [26].

It is hoped answers to questions such as "Can users understand the purpose of the web application?", "What do they like and dislike about the application?", and "Can they complete activities on it?" will be found.

Test Approach

In order to effectively test the high-fidelity wireframes created, a set of scenario-based tasks were created for users to walk through. Scenarios are informal narrative descriptions used to describe a task, that allows exploration and discussion of contexts, needs and also requirements [25, p.408] which is the main focus. This approach can be more effective than just supplementing the prototypes with arbitrary tasks, with no given context. Scenarios instead help to explain some aspects of the user's goals and put more emphasis on the usability and user experience [25, p.410]. It was decided not to create a user persona to complement the scenarios, and this decision will explained later when discussing the profiles of members in the focus group.

Three scenarios-based tasks will be given, which will involve several of the users stories described in Section 3.3.

Scenario 1

> "The UK government has just announced international travel will be allowed in a months time. You and 3 of your friends want to travel abroad as soon as this date comes, but are not sure of what destinations allow UK travellers to visit them."

Find a destination for you and your friends to travel to, that allows UK travellers.
Expected Actions: User either scrolls down and views the recommendations on the homepage or they click on 'Recommendations', on the navbar and navigate to the dedicated 'Recommendations' page. An alternative approach would be to type 'Anywhere' in the destination box and then search for a flight as suggested on the homepage wireframe design.

Scenario 2

> "You are searching for flights to travel to New York for a music festival you want to go to in June. Whilst searching for flights, you realise you aren't aware of the current restrictions imposed in the United States due to COVID-19, and have no information on the number of COVID-19 cases there."

Search for a flight to New York and also find out the latest travel advice for the USA and the rate of cases per 100,000 people in the last 7 days in the USA.
Expected Actions: User types in keywords for flights to New York, clicks find your flight and then uses the 'COVID-19 snapshot' to see COVID-19 related information.

Scenario 3

> "You are considering going to Paris in August but you're hesitant to book now since August is still a bit far away and you're worried the outlook of COVID-19 may change in France, meaning you won't be able to travel there and could lose out on the money paid for your ticket."

Search for flights to Paris and sign up to both price updates for flights to Paris and alerts on COVID-19 in France.
Expected Actions: User types in keywords for flights to Paris, clicks find your flight and then clicks on the 'sign up for updates' buttons on the search results page.

Test Plan

In addition to the given scenarios, the wireframe prototypes will also be tested for general usability and user experience, with a greater onus on the experience users have when accessing COVID-19 related information. As mentioned in the gap analysis conclusion in Section 2.2.3, the existing solutions did well to provide information on COVID-19 in a fashion that is easily accessible for users, so again, it is important this standard is upheld and tested for in the design of this application.

According to [27] Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts, ISO 9241-11:2018, usability can be described as "the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". In order to get a good overall picture of the usability of the wireframe prototypes, each of the goals described will be measured with statistics highlighted below:

- Effectiveness - Percentage of users completing the task successfully.
- Efficiency - The average task completion time.

- Satisfaction - User rating of their experience using the wireframes from 1 - 5 (1 = poor, 5 = good).

To aid with the collection of data from the focus group participants', an objective performance capture sheet was created which can be found at Appendix B. Due to the current COVID-19 restrictions, all the tests were conducted remotely via various platforms such as Microsoft Teams and Zoom. It is also important to note that participants were interviewed one-by-one individually. This is because it was difficult to get everyone onto the same video call at the same time due to differing schedules.

Focus Group Profiles

As alluded to in the *Test Approach*, see Section 3.4.2, it was decided not to create a user persona alongside the test scenarios. This was decided because the users participating in the focus group are reflective of the typical users' of this web application. They can be summed as holidaymakers' who are open to travelling internationally, despite the uncertainty around restrictions imposed due to COVID-19. Since the participants all fall into this category, the given scenarios are in line with their own views and it isn't necessary for them to *"put themselves in someone else's shoes"*, in order to handle the scenarios effectively.

The makeup of the participants in the focus group can be seen in Table 2 below.

Table 2: A table summarising the profile of the focus group participants

ID	Age Group	Gender	No. of holidays in last 3 years
0	18-25	Male	12
1	18-25	Female	5
2	18-25	Female	4
3	18-25	Male	4
4	18-25	Male	7

The profile of the participants are those that travel and go on holidays quite frequently, with 60% of the participants going on 5 or more holidays in the last 3 years. This makes them suitable candidates to test the prototypes, as it can be assumed that they have used similar flight finder web applications before to book travel.

3.4.3 Feedback

In this section, the responses from the scenario-based testing will be discussed, and at the end, key takeaways from each scenario will be listed that will help guide the final designs, and also help with refining the user requirements.

Scenario 1

This scenario proved difficult for the participants to follow with only one user demonstrating the expected actions. One user went straight to the travel advice view to find out this information. When asked why they did this, they said "they thought of advice after hearing the scenario and recommendations seems more like its based on other people's experience, like a ratings and reviews section".

Another user used the search feature and typed 'Anywhere' for the destination which is an alternative expected response. When asked why they didn't go through the route of recommendations they made a really good point of the recommendations not being "for a specific date" and that she wants to see recommendations for the dates she wants to travel, not just in general. This user also mentioned she went via this route, as searching would mean she also gets an idea of prices, and the recommendations have no indication of prices at all currently.

It is clear this workflow was difficult for users to follow and was quite confusing for some, especially with the presence of 'travel advice' alongside the wording of 'recommendations'. Below key takeaways from this scenario are listed.

Key takeaways:

- Change the wording in the navbar to clear up confusion between 'Recommendations' and 'Travel Advice'.
- Add indications of prices on the recommendations, for e.g. Maldives "from £300".
- Allow users to search for recommendations based on dates they want to travel.

Scenario 2

The participants fared much better with this scenario, and all of their actions followed that of the expected action for this scenario. The participants liked the idea of the 'COVID-19 snapshot', however a few of the participants were unclear of what some of the information was referring to. For instance, one user was unsure of what 'Cases/100k' was referring to. Another user also mentioned that the country the snapshot is referring to, should be stated on the minimised form of the snapshot, like it is on the larger version.

A few users also mentioned they would like to see a 'last updated date' under the cases information, so that they know how recent this information is. They appreciated that there was a source for the information, with a participant saying "it helps make the information more reputable when the source is stated, so I would trust what I'm seeing more".

This workflow seemed to make a lot more sense to the participants, and the search process was easy for them to follow as it is inline with common design patterns they are used to. In terms of the snapshot, a

few things could be clarified to make it a more useful feature, such as stating what some of the information actually means. The key takeaways from this scenario are as follows.

Key takeaways:

- Adding tooltips or information symbols next to labels, e.g. Cases/100k, to explain what they mean to users. Users could just hover and quickly see this information, meaning the design is still kept simple with minimal text.
- Adding a last updated date to the cases information would help make the information more reputable, and give users an idea of how recent the data is.
- Stating the name of the country on the minimised COVID-19 snapshot would be useful to users to verify they are seeing the right information for their destination.

Scenario 3

This scenario was straightforward for the participants to grasp. This could be because the workflow is largely similar to that of scenario 2, searching for flights, and they likely would've also noticed the 'sign up for update' buttons as well. The participants liked the idea of being able to sign up for updates, with one participant mentioning that "they can sit back and not have to continuously scour the internet for updates".

Whilst most agreed this was a helpful feature, some of the participants weren't pleased with its execution. One participant said "there should be a way to sign up for both price and COVID-19 updates in one instead of having to do them separately". Another user even described the process as "tedious" and complained about having to enter their email twice to sign up for both updates, including that this is something that could "put them off" using the update feature and possibly the Web App itself.

One participant said, adding a description of what updates the users expect to receive would help to make it more intuitive. Adding to this he said, it would be better to be able to pick a destination to receive updates on rather than a country as when buying flights he think's of cities or regions rather than a whole country.

Initially it was thought users would want to receive updates on typically just one of either, the COVID-19 information, or price updates, but after this scenario was tested, it became clear that users were willing, and actually desired, to get information on both. The key takeaways discovered from prototyping this scenario are described below.

Key takeaways:

- Combining the update feature so users can sign up to receive both update types, and don't have to enter their email, for example, multiple times.

- Adding a description to the alert sign up form to properly communicate to the user what information they can expect to receive.
- Changing the option of picking a country in the form to a destination instead.

General Feedback

After going through the three scenarios and the prototype wireframes, users were asked for their general opinion of the web application, and were asked if it was something they would actually use to search for flights.

Generally the participants really liked the general concept of having COVID-19 information within the same place or product they use to search for flights. Participants made comments such as "everything's in one place so I don't have to go and Google information", "keeping up with news and changing circumstances is difficult, so a website like this could be really helpful" and "everyone loves a one-stop shop so this is perfect, I don't have to search on government websites to find out COVID-19 related travel restrictions for where I want to go".

These comments serve well in highlighting how some of the problems discussed in the Problem Domain in Section 1.1 are attempted at being solved, with a key point being to help holidaymakers "find information on these restrictions without having to browse through numerous government websites and news articles".

Usability and User Experience

Table 3 gives a summary of the data collected from the objective performance sheets, see Appendix B, whilst doing the prototype testing. Issue frequency and task completion frequency refer to the percentage of participants facing issues while doing the task and completing the task respectively. An issue can be described as the participant having to ask for help in order to proceed further in doing the task.

Table 3: A table summarising the usability statistics collected during prototype testing

Scenario No.	Ave. Task Time (mm:ss)	Issue Freq.	Task Completion Freq.
1	01:33	80%	20%
2	00:25	20%	100%
3	00:19	40%	100%

3.5 Requirements Specification

Using the feedback from the prototype testing, requirements were drawn out and logged using Jira, and the rationale for this will be detailed further in the implementation section. The top-level requirements

are detailed in Table 4 and prioritised using the MoSCoW method. The MoSCoW method was chosen as it is an intuitive and simple way to prioritise requirements in terms of their importance of meeting the aims and objectives of a project [28] and according to the sentiments of the end-users. The four MoSCoW priority levels are Must have, Should have, Could have and Would have. This is further detailed by Figure 19.

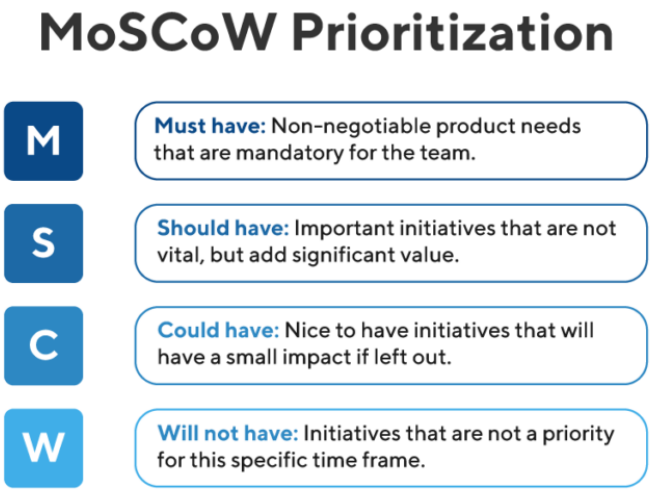


Figure 19: MoSCoW prioritisation levels. Adapted from [28]

The full requirements specification can be found in Appendix C.

Table 4: Top-level requirements

Begin of Table		
Summary	Priority	Description
Main Dashboard	Must	As a user I want to be able to search for flights direct from the main dashboard And I want to be able to navigate easily to the different views of the application to access other features So that I can use all of the application's features efficiently

Continuation of Table 4

Summary	Priority	Description
Flight Search	Must	As a user I want to be able to search for flights to any destination And I want to be able to apply relevant filters to my search And I want to be able to see COVID-19 information for my preferred destination within the same search interface So I can find prices and flight information for the destination I want to travel to
COVID-19 Snapshot	Must	As a user I want to be able to see the current daily rate of COVID-19 for a destination And I want to be able to see the weekly rate per 100,000 people And I want to see that country's latest travel advice And I want to be able to check if that country is on lockdown or has a mandatory quarantine on arrival And I want to see the source of this aforementioned information And I want to see when this information was last updated So that I can then use it as part of my decision, when deciding to buy a flight to a destination
Destination Updates	Should	As a user I want to be able to input my email address and preferred destination country and then receive regular updates on my country of choice, regarding it's COVID-19 situation. So I can keep track of changing circumstances
Travel Advice Map	Should	As a user I want to be to able to view an interactive world map, colour coded according to daily COVID-19 cases And to be able to change this colour coding using other information as the deciding factor, such as if a country is on lockdown, or the weekly rate of COVID-19 cases And I want to be able to click on a country on the map to get information on this country's COVID-19 situation So I can get see on a global scale what the COVID-19 situation is like in order to help me make a decision on what regions I may want to avoid travelling to

Continuation of Table 4		
Summary	Priority	Description
Recommendations	Could	As a user I want to be able to receive suggestions on places I can go to from my country of origin, incorporating COVID-19 travel restrictions So I can get ideas of destinations I can travel to without being affected by COVID-19 restrictions
Lockdown Prediction	Would	As a user I want to be able to see predictions on countries that may enter a national lockdown in the near future So I can avoid booking travel to these countries
End of Table		

3.6 Conclusion

This requirements analysis has been a success in achieving its aim of narrowing the communication gap between the end users in order to build on the initial requirements gathered from the Gap Analysis in Section 2.2, and then also prioritise the requirements according to the sentiments of the end-users, discussed during the prototyping testing.

A benefit of using prototyping for requirements elicitation means that there are already high fidelity wireframe prototypes that can be used as as a guideline to create the designs, and by testing the usability and user experiences of the wireframe designs, the feedback can be used to improve and create final designs that have been influenced greatly by end-user input.

Chapter 4

Design

4.1 Introduction

In this section, the final user interface designs will be discussed, including decisions on the theme and font chosen, and the usability heuristics that have been factored into the final design. The designs also reflect the outcomes of testing the high-fidelity wireframe prototypes, highlighted in Section 3.4.3.

This section will also look to describe how the system design has evolved as a result of the requirements analysis and will also reflect the new requirements.

4.2 System Design

In order to help further verify the requirements, two Unified Modeling Language (UML) diagrams were drawn: a Use Case diagram which is a behavioural UML diagram, and a Component diagram which is a structural UML diagram.

4.2.1 Component Diagram

A component diagram is more focussed on modelling the structure of an overall system and the dependencies of components in the system. It has the concept of a *provided interface*, which shows the services that instances of the component in question can offer to their clients. It is represented in the diagram by a small circle connected to a component. It also has what is called a *required interface*, which highlights what services a component needs in order to perform its own function, and provide services to its own clients. It is represented by a semicircle attached to a component[29]. The component diagram for the proposed system is pictured in Figure 20.

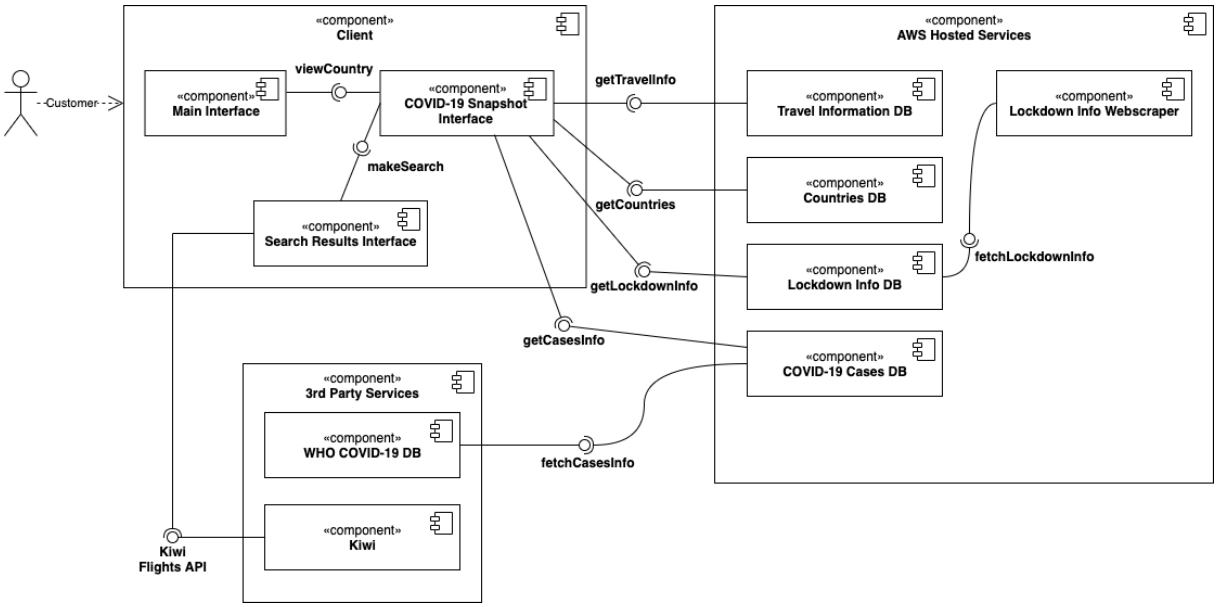


Figure 20: A component diagram for the web application

The sole user is a customer who interacts with the client-side of the web application. There are three subsystems - the client, AWS Hosted Services and 3rd Party Services. AWS Lambda functions are used to collect data from 3rd party services and store them in the AWS DynamoDB Databases. The client side calls AWS API Gateway endpoints in order to pull information from the databases and it also directly calls external APIs such as the Kiwi Flights API [30]. The rationale for using these services will be further discussed in the Implementation chapter.

A decision was made to pivot from using Amadeus's API, chosen in the literature review, and use Kiwi's Flight API due to them offering deep links to a booking platform in their API. It seemed more natural allowing the user to search for flights, and by clicking on a listed flight and its details, be redirected to an external link to proceed to potentially book their chosen flight, as opposed to just listing flights and their times with no where for the user to go further in the booking process. This is actually also a feature described in the user stories in Section 3.3.

4.2.2 Use Case Diagram

Below in Figure 21 is the use case diagram used to model the intended behaviour of the application, derived from the functional requirements.

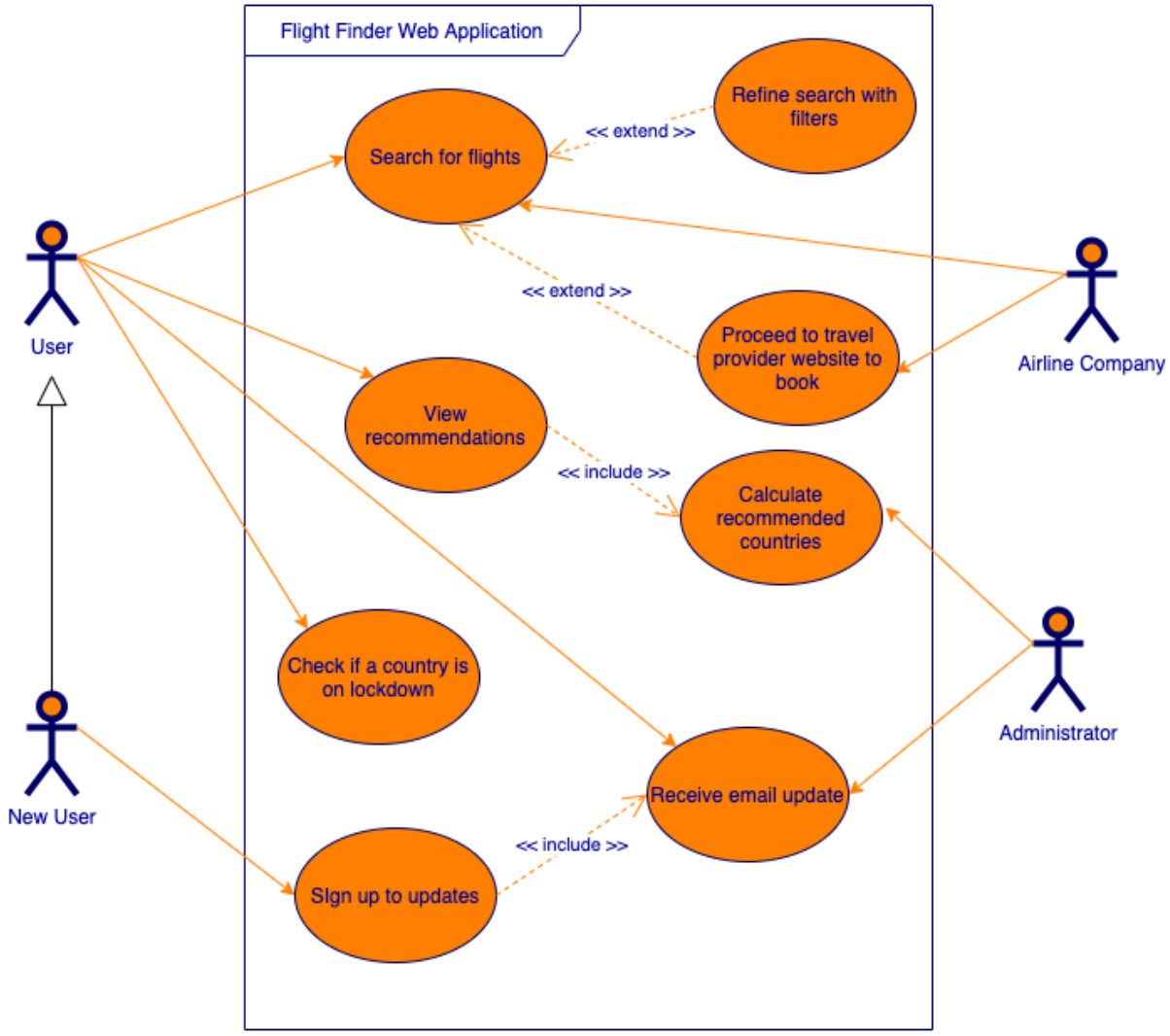


Figure 21: A use case diagram of the web application

Use case diagrams are an industry standard in describing a system's software requirements and they provide a "black box view" of the system, presenting the external behaviour of the system rather than its internal workings [31, p.1]. This view has a greater focus on how the system interfaces with its actors, including but not limited to the potential end-users in this situation [31, p.2].

There are three actors involved in the system: a user/customer, the airline companies and the site administrator. There is a generalisation of the user, that being a new user, as it would be a new user signing up to updates, as opposed to a existing user who is assumed to have already signed up for updates and will be receiving the updates by email.

4.3 User Interface Design

4.3.1 Design Approach

In order to fulfil the set out requirement of ease of navigation and a good user experience, as referred to in the Full Requirements Specification, see Appendix C, it was decided to use Jakob Nielsen's 10 Usability Heuristics [32] as a rough guide when designing the final designs.

While creating the final designs, the 9 principles of good web design discussed by Marianne [33], were also used as a reference point. Principles such as simplicity; involving colour, typography and imagery; F-shaped pattern reading, which involves the way people generally scan and read pages naturally; and navigation were strongly taken into account and will influence styling choices in the following sections.

4.3.2 Key Styling and Palette Choices

Looking back to the 9 principles of web design, it discussed using effective colour palettes that fit the brand and influence customer's behaviour towards the brand [33]. With this in mind the colour palette, see Figure 22, was chosen to be used in the final designs. Since this application will be used to search for flights, using colours that can be associated with key words such as "flight", "sky" and "clouds", were important. To this effect, different shades of blue were used. The colour blue was also chosen due to the colour psychology behind it.

Cherry [34], highlights how blue is a colour commonly found in nature, and because of this it is described as bringing "feelings of calmness and serenity" to mind. Other words used to describe blue's psychology are "peaceful, tranquil, secure and orderly" as mentioned by Cherry [34]. When thinking of anything to do with air travel, these are the type of feelings people want to associate a brand with. People typically want to be assured that the brand is safe, secure and orderly, as air travel can be dangerous if stringent measures aren't in place. Although air travel itself isn't actually being provided, it still felt important to convey these feelings across through the web application and branding, to give customers that sense of calm when searching for flights.

Figure 22: Chosen Colour Palette

Another key styling decision made was the typography to be used within the web app. It was decided to primarily use the Nunito font [35], which Google describes as a "well balanced sans serif typeface

superfamily". Figure 23 shows a few examples of the Nunito's font styles. This font family was chosen as it is sans serif, which typically is more legible when used on websites as opposed to serif typefaces. Again, taking into account comments by Marianne [33], "typefaces should be legible" and play an important role on websites, so choosing a font that was simple but also different to what is typically seen was key.

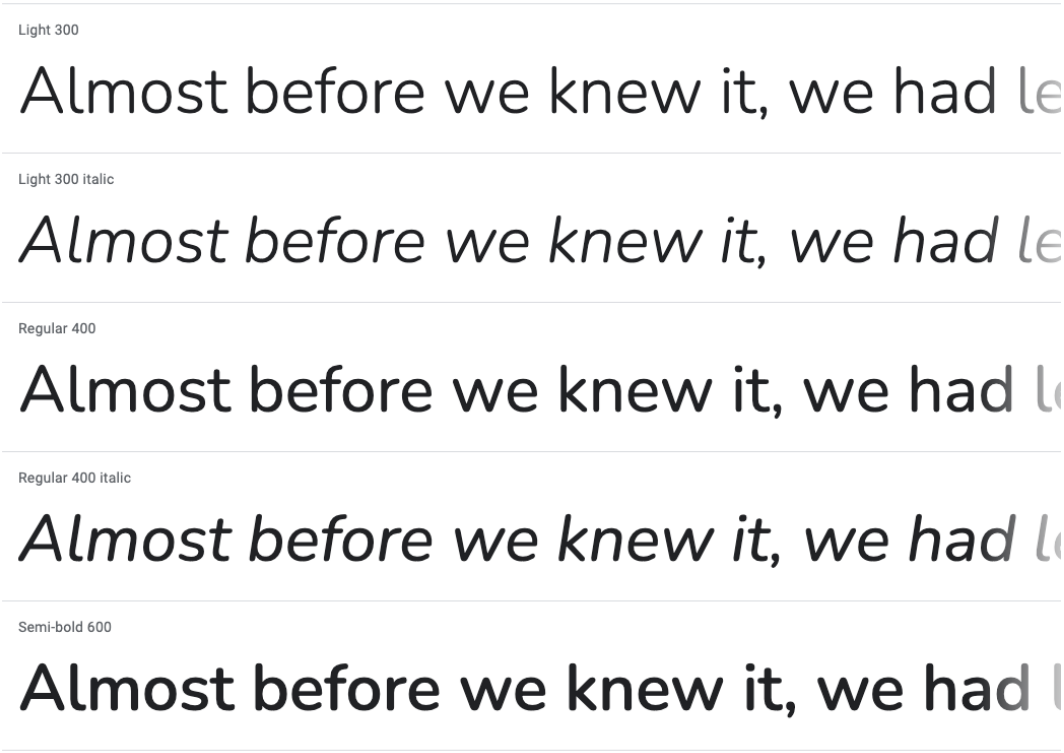


Figure 23: Nunito Font Family

4.3.3 Final Designs

Please note that as per the feedback given in Section 3.4.3, the Recommendations page has been renamed to Travel ideas.

Homepage/Main Dashboard Page

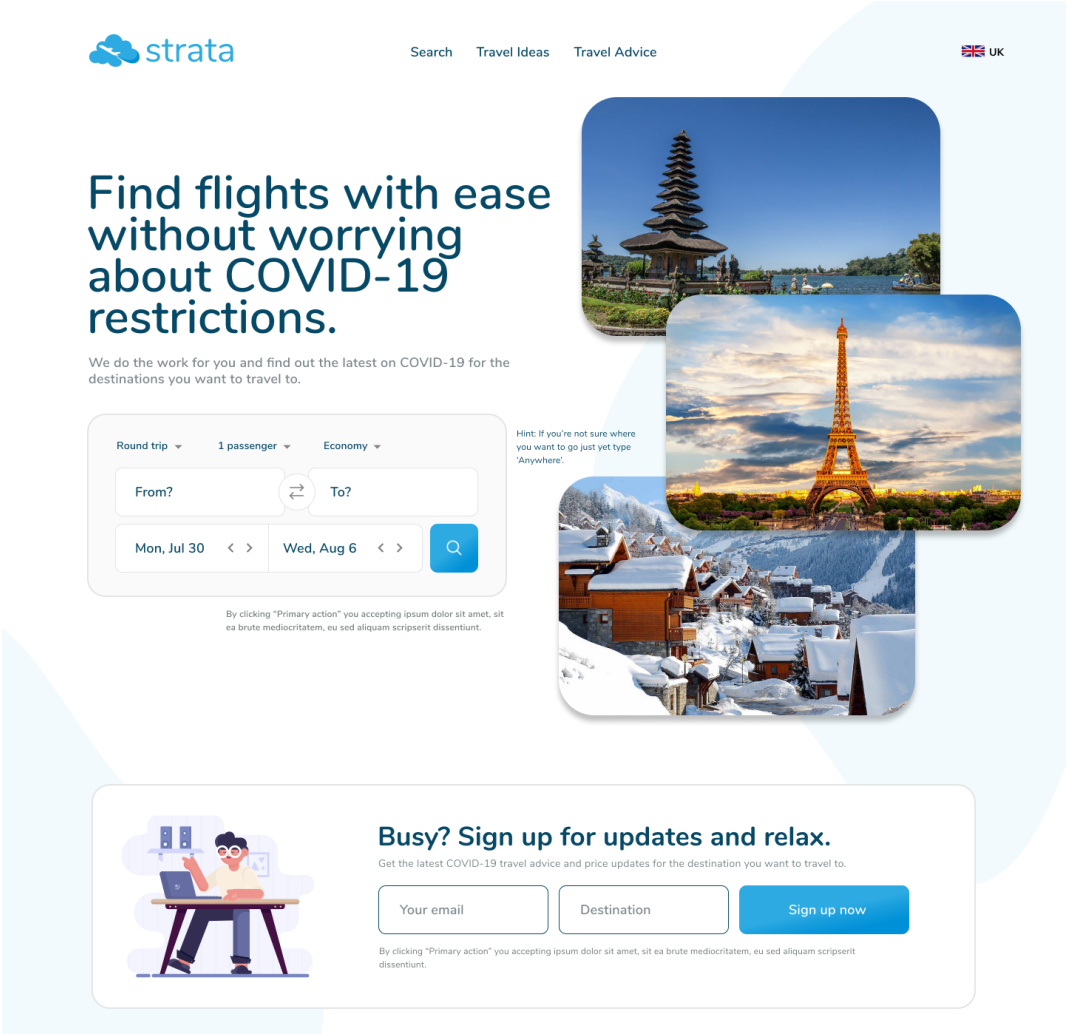


Figure 24: Main Dashboard Design

Search Results Page

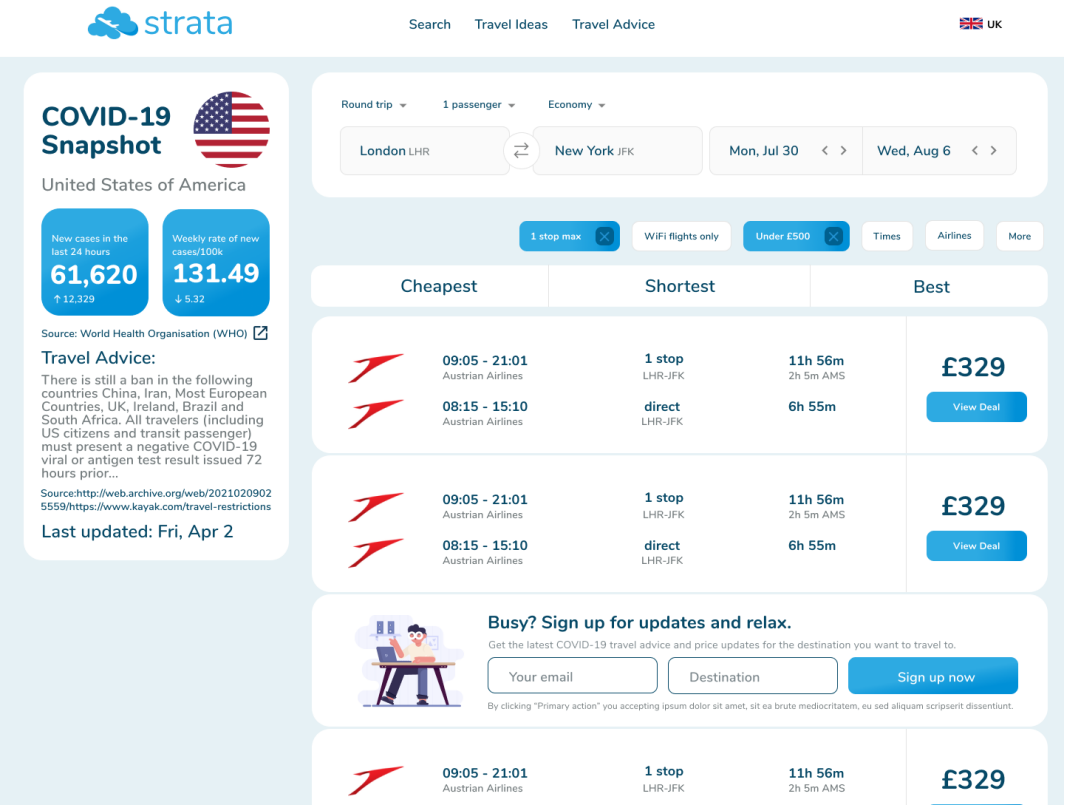


Figure 25: Search Results Design

Figure 26: Expanded Snapshot Design

Update Sign Up View

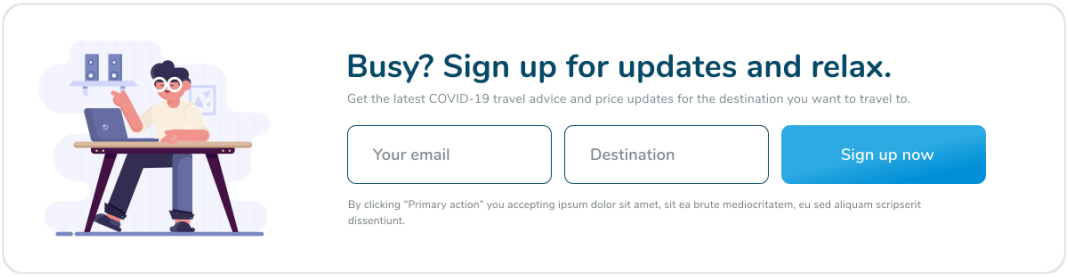


Figure 27: Update Sign Up Design

Travel Advice Page

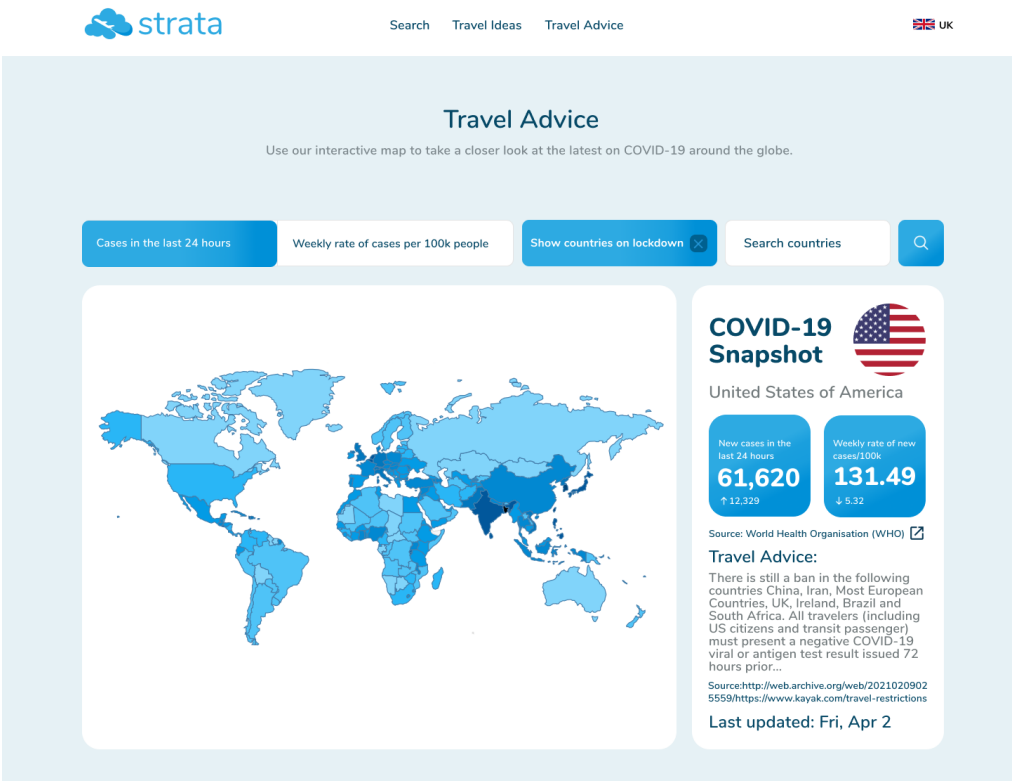


Figure 28: Travel Advice Design

Travel Ideas Page

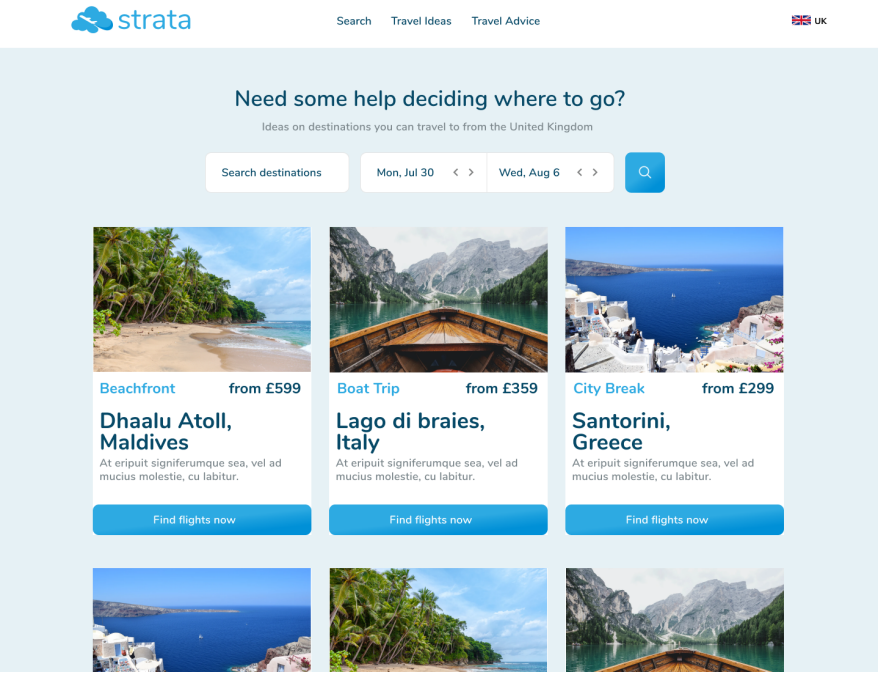


Figure 29: Travel Ideas Design

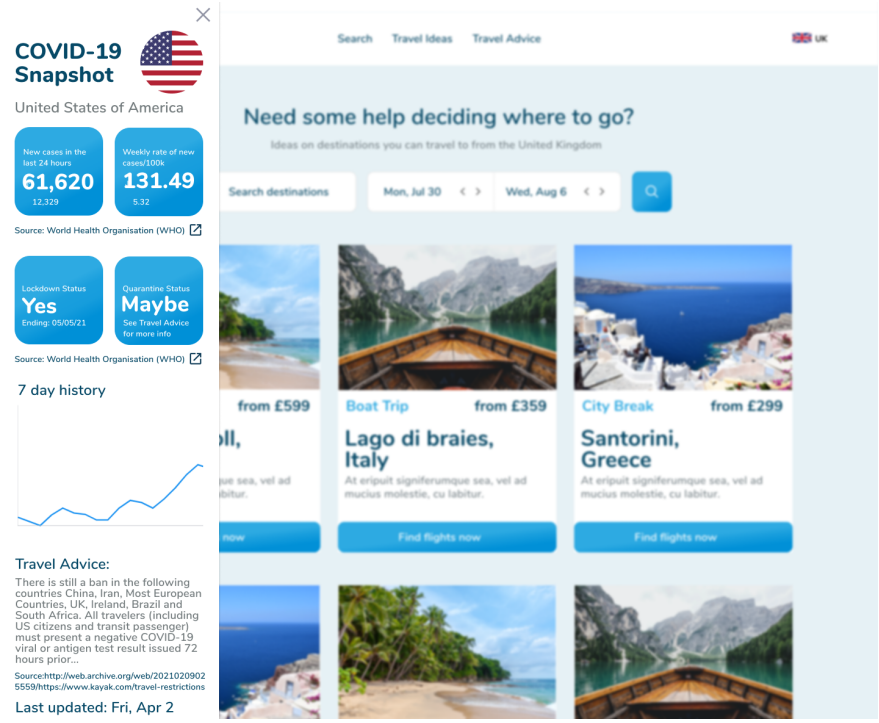


Figure 30: Travel Ideas with Snapshot Design

4.4 Conclusion

The final designs, as mentioned in the introduction, incorporated the focus group feedback and this is evident with features such as the newly worded navigation bar which states "Travel Ideas" instead of "Recommendations", to avoid any confusion. The update feature has also been combined into one, and has a description to make its objective and purpose clearer. The COVID-19 Snapshot also includes features discussed by the focus group participants, in Section 3.4.3, such as information sources and last updated dates, to make the information more verifiable.

The designs also begins to factor in Nielsen's heuristics, such as "User Control and Freedom", which with the exit symbols on the expanded COVID-19 Snapshot and slider is implemented, as users can leave a potential "unwanted action" [32]. Another heuristic already visible from the final designs is having an "Aesthetic and Minimalist Design", with each design kept uncluttered and as simple as possible [32]. More of Nielsen's heuristics will become evident in the actual implementation of the application, such as error handling and prevention.

Key design decisions made in this section such as the colour palette and typography, alongside the use of web design principles, will aid greatly with the implementation of the web application.

Chapter 5

Plan

5.1 Introduction

This section will take a role in describing the tools and environments that will be used during development, as well as the libraries that will be used, and the methodology to be used during the implementation stage.

5.2 Methodology

In retrospect, the methodology for this project could've been chosen in the Gap Analysis, however it was decided to choose the methodology later on in the project, as the scope of what has to be achieved off the back of the requirements analysis would be clearer, and can then influence the development methodology to be used.

Taking into account new constraints, namely resource constraints such as time and man power, alongside the Full Requirements Specification, a decision was made to use an Agile Methodology. Within Agile project management, more of an onus is placed on software being built in the earliest possible time [36]. This is in sharp contrast with the more traditional waterfall method where one step can't begin until the preceding step has finished causing delays. Koi-Akrofi et al. [36] goes on to say, "Agile management is often said to work best with tiny teams.".

Both statements go a way in showing that, with the limited resources in man power and time available, using an Agile methodology will best aid the delivery of an MVP (Minimal Viable Product) for this project, that can be continually updated and improved. Following the agile rapid prototyping method used to create the designs, using a similar process to develop the product was also a welcomed transition.

5.3 Tools and Environments

5.3.1 IDE/ Text Editor

Visual Studio Code (VS Code) was chosen as the text editor to be used throughout the development of this project. VS Code is a lightweight but powerful source code editor that has built-in support for JavaScript, and extensions for other languages such as Python [37]. Since the main programming languages to be used to develop were JavaScript and Python, as mentioned in the Gap Analysis, VS Code was clearly fit for purpose. It was also the source code editor that the authors were more familiar with using, so the time that may have been used to get over the learning curve with another text editor was factored out.

5.3.2 Version Control

The chosen version control system for this project is Git, due to it being the de facto distributed version control system and its use in GitHub, which allows for private hosting of source code repositories. An added bonus is that GitHub and Git are tools the authors have experience in using already, meaning no time is used in learning a new system.

Version control is a key enabler in efficient agile development as it keeps tracks of changes to files over time [38]. This helps support the high and frequent output in changes to code, as it provides a full audit trail of any changes, providing the ability to revert back to previously working versions in the case that there is a problem with the current version. Version control is also one of the main building blocks of Continuous Integration/Continuous Deployment (CI/CD).

5.3.3 CI/CD

Continuous Integration/Continuous Deployment is described by Sacolick [39] as an "agile methodology best practice". It is set of principles and practices that help development teams deploy code changes frequently, by automating aspects such as testing, the build of the source code and its actual deployment. This leaves developers to focus on the actual code and user requirements.

Earlier when discussing the reasons for using a serverless backend for this project in Section 2.4.5, being able to have more time to focus on the business logic and requirements was a big factor, and for this same reason, implementing CI/CD would help facilitate the agile methodology by allowing code changes, pushed to a GitHub repository, to be deployed automatically. CircleCI was chosen as the CI/CD tool due to it being very quick to set up, and easily integrable with GitHub and AWS, which are other tools being used on this project.

5.3.4 Project Management

In Section 3.5, it was mentioned that Jira was used for the logging of requirements, and each was logged in term of its priority. Jira was useful for this, as it has priority levels that easily link with the MoSCoW prioritisation levels. Jira was used to log the requirements, as by doing this, meant it could also be used to facilitate the organisation of related tasks derived from the requirements into sprints, which could then be used to manage time effectively and have an overview of the tasks that needed attention. Figure 31, shows tasks organised into a sprint in Jira.

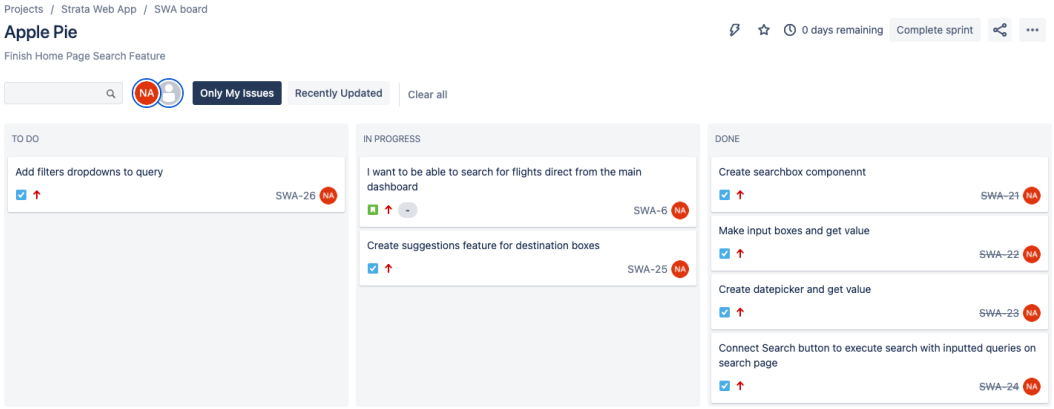


Figure 31: View of a sprint in Jira

5.3.5 Google Chrome DevTools

As described by Google [40], Chrome DevTools are a set of web development tools inbuilt into the Google Chrome browser, which allows users to inspect and edit HTML elements and their Cascading Style Sheets (CSS) on the fly, allows users to view the console output, which is very helpful during debugging, and view network requests, which is helpful in managing API calls to third parties. Figure 32 is a screenshot of DevTools being used to inspect elements on a web page.

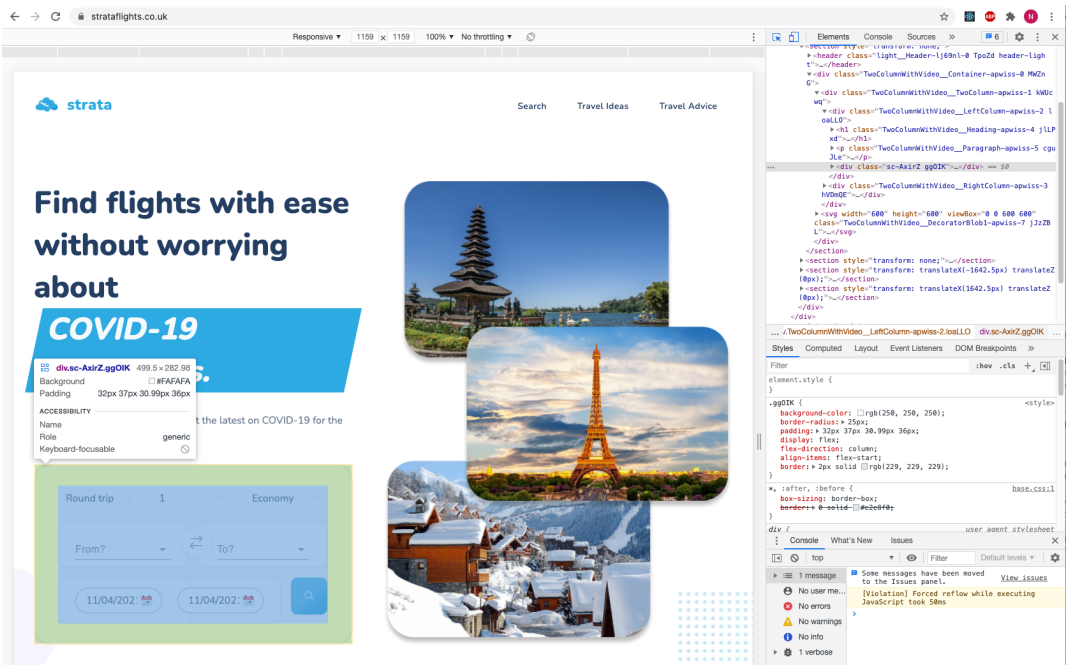


Figure 32: Inspecting elements in Chrome DevTools

5.3.6 Postman

When working with APIs, it's very important to test them before calling them through the application. Developers need to know what the response data of the API looks like, in order to use it in their applications, and it is best practice to not use it programatically initially, before testing its endpoints and reviewing its documentation, as the application may make too many requests, potentially violating the API's rules or using up the request limit.

To get around this, Postman [41], an API platform, is widely used to test and explore APIs before using them in applications, and for the aforementioned reasons, it will be used on this project as several third party APIs will be used, as can be seen in the component diagram in Section 4.2.1.

5.3.7 Jupyter Notebook

It can sometimes be difficult working with data tables in formats such as CSV and pandas just using Python on its own, as users have to constantly save and recompile code to see changes, and the visualisation of the table in the terminal environment is usually not up to standard. Jupyter Notebook [42] is an open-source web application that allows users to create documents containing live code and visualisations, making the data cleaning process much swifter.

As alluded to in the conclusion of the literature review section of web scraping, see Section 2.4.2, this project involves the cleaning of potentially unstructured data, so using Jupyter Notebook will help with

visualising data, and selecting the correct attributes and discarding those not helpful. Figure 33, shows Jupyter Notebook being used to visualise a pandas dataframe.

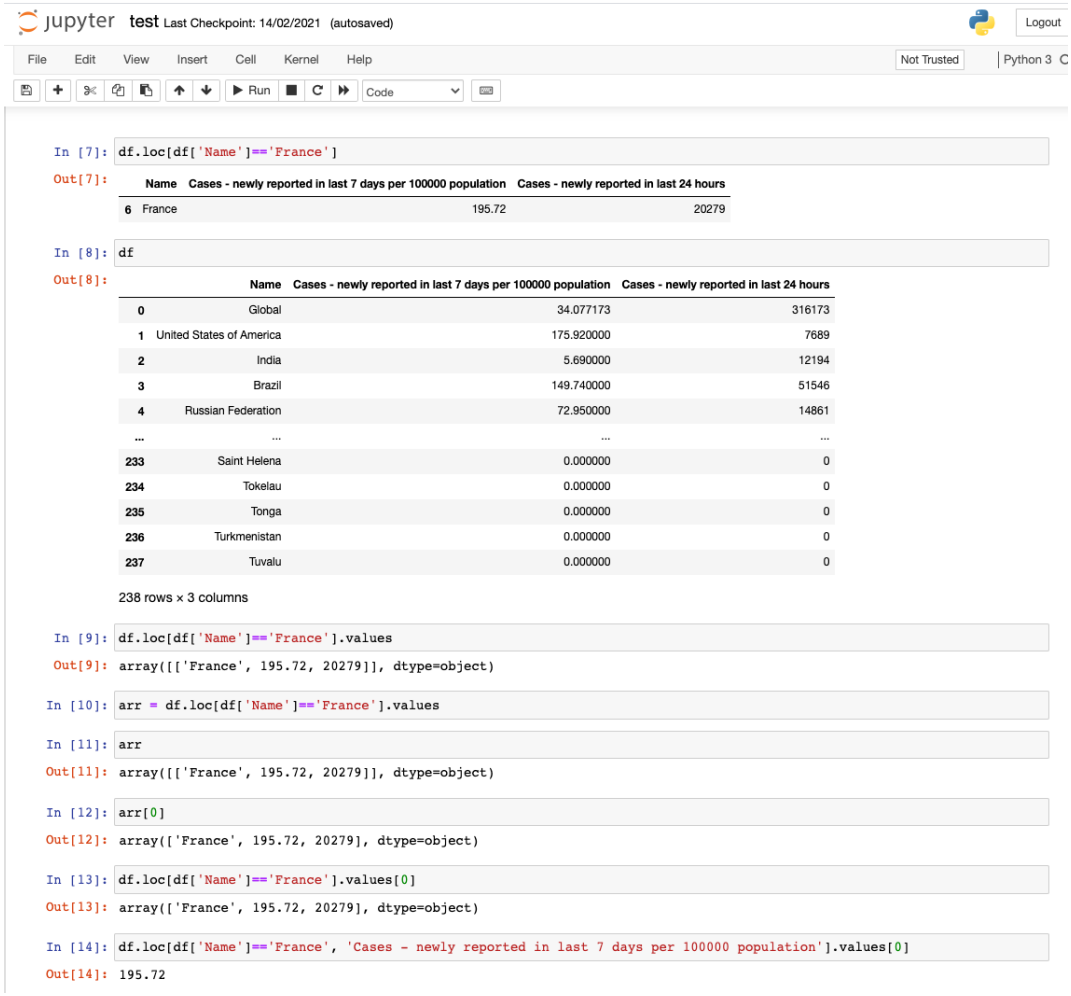


Figure 33: Jupyter Notebook being used to visualise a pandas dataframe

5.4 Libraries

5.4.1 Client

React Rainbow - React Rainbow is a UI component library for React. It was used for components such as date pickers, sliders and radio button groups.

Material-UI - Material-UI is a popular UI framework for React. It was used for the auto complete input boxes, and grids were also used to make the application responsive for devices of all sizes.

Tailwind CSS - Tailwind CSS is a "utility-first CSS framework" that allows developers to quickly apply CSS to style elements using pre-existing utility classes [43]. This means custom CSS and classes doesn't have to be written and the utility classes can be reusable meaning a large CSS file doesn't have to be written, improving the web application's performance.

Styled components - Styled components is another library used for CSS styling, letting developers write actual CSS code within the JavaScript. This was used when more control of the CSS of an element was needed than the Tailwind utility classes offered. This was used more in styling custom components derived from the final designs, see Section 4.3.3.

Axios - Axios is a JavaScript library that can be used to make XMLHttpRequests directly from the browser to interact with APIs [44]. XMLHttpRequests [45] enables the user to retrieve data (not just Extensible Markup Language (XML) but any type of data) from a URL without having to do a full page refresh. This was used in making API requests to both third parties, and APIs created with AWS API Gateway, as part of the serverless backend.

React Router - React Router is a routing framework for React and it was used here to conditionally render and display components based on the current URL route.

Match-Sorter - Match-sorter is a JavaScript library used to sort arrays in a deterministic manner. It was used to create the lookup filter, as part of the input fields for the destination in the flight search, to list airports and cities based on a user's text input in the box.

React Simple Maps - React Simple Maps is a library that can be used to make SVG maps in React, underpinned by d3-geo and topojson [46]. It was used to transform the data collated on COVID-19 cases and lockdown statuses to create an interactive map.

React-tooltip - React tooltip is a lightweight library that solely provides a tooltip component, and it was used as it implements a tooltip in a simpler manner than in React Rainbow and Material-UI. It was used to provide the tooltip for the map on the Travel Advice page.

5.4.2 Serverless backend

Pandas - Pandas is a Python based data analysis and manipulation tool which works well with tabular data. Using pandas DataFrames (data tables in Pandas) within Jupyter Notebook made it a seamless process cleaning the data, and removing unnecessary columns (features) from datasets collected through scraping on the internet.

Requests - Requests is the standard Python library for making HTTP requests. It was used for making API requests to receive data from third parties, to then go on and parse with Beautiful Soup and analyse and clean with pandas.

Beautiful Soup - Refer to Section 2.4.2 in the Literature Review, for a description of Beautiful Soup and the justification for its use.

Boto3 - Boto3 is the official Python SDK (Software Development Kit) for Amazon Web Services (AWS), allowing users to interact with AWS services directly from Python code [47]. It was used within the code in the AWS Lambda Functions, to store or get data from the DynamoDB databases.

Docker - Docker is a platform that "provides the ability to package and run an application in a loosely isolated environment called a container" according to Docker Inc [48]. Its use case here was to mimic AWS's Lambda runtime environment, so dependencies could be installed within it and then packaged to be used as a layer in Lambda.

5.5 Conclusion

This section has answered questions on what methodology will be used going forward into the implementation stage of this project, and has provided rationale on the chosen libraries and tools that will play a part in implementing the set out requirements. This section will serve as a key reference point during the implementation stage.

Chapter 6

Implementation

6.1 Introduction

In this section, the process undertaken to implement the given requirements, as specified in Appendix C, will be detailed, describing how each feature was implemented, alongside the libraries used in their implementation. Some code excerpts will be shortened using an ellipsis(....).

6.2 Serverless Backend

This section will detail how each feature of the serverless backend was implemented, discussing any tools, services and libraries used in its implementation.

6.2.1 Introduction

The first few sprints were dedicated to writing the scripts to be used to collect, clean and process the required data. Following this, these written scripts were set up within a number of AWS services to form the serverless backend.

The AWS services used are as follows:

API Gateway - Amazon API Gateway is a service that lets developers create, publish and secure REST APIs that act as the "front door" to other AWS services [49]. It was used to interact (get or put) with information stored in DynamoDB, and to trigger Lambda functions performing business functions, such as the COVID-19 email update.

DynamoDB - Amazon DynamoDB is a serverless NoSQL, key-value and document database service that gives applications low-latency data access [50]. It was used on this project to store data collected from third parties, to then be polled through a Lambda function and API Gateway, to be used in the front-end of the application. It was also used to store a mailing list of people signing up to alerts.

Lambda - See AWS Lambda found in the Backend/bespoke API section of the Literature Review.

CloudWatch Events - Amazon CloudWatch Events is mainly used for logging and monitoring purposes but can also be used to "schedule automated actions that self-trigger at certain times using cron or rate expressions" [51], which is its use case in this situation. It was used to set lambda functions collecting, cleaning and then storing the data, to run daily.

Simple Email Service (SES) - Amazon [52] describes SES as a "cost-effective, flexible, and scalable email service that enables developers to send mail from within any application". This service was used to securely send emails to users signed up for COVID-19 updates for a chosen destination.

Identity Access Management (IAM) - AWS IAM allows users to manage access to AWS services and resources securely, giving the ability to create roles and "manage permissions to control which operations can be performed by the entity, or AWS service, that assumes the role" as stated by AWS [53]. It was used to create roles with the least-privileges required for executing different AWS services.

6.2.2 COVID-19 Case Data

The data source chosen for data on global COVID-19 cases was the World Health Organisation (WHO)'s COVID-19 Dashboard [54]. It was chosen as the WHO is a very reputable source, and provide accurate, up to date data on COVID-19 for all countries. Using a reliable source helps to meet the requirement on *Sources of data* found in the Full Requirements Specification. Another factor in deciding to use this data source was the ability to get the data in a CSV format, thus removing the need to collect the data through web scraping.

Now the data was sourced, the next step was to clean the data so it contains only the data actually required. As per the Full Requirements Specification, the required data to be able to fulfil the requirements is the number of new COVID-19 cases in the last 24 hours, and the weekly rate of new cases per 100,000 people for each country. To clean the data, a Jupyter Notebook was used as described in Section 5.3.7, and the CSV file was read straight into a pandas DataFrame. Using pandas, any columns that weren't required were then dropped and then any rows with missing values were also removed, as can be seen in Listing 1.

```
#source of data from WHO
url = 'https://covid19.who.int/WHO-COVID-19-global-table-data.csv'

# Reading CSV file into pandas DataFrame and filtering required columns
df = pd.read_csv(url)
df = df.filter(['Name', 'Cases - newly reported in last 7 days per 100000 population'
, 'Cases - newly reported in last 24 hours'])
# removing rows with missing values
df.dropna(axis = 0, how = 'any', thresh = None, subset = None, inplace = True)
```

Listing 1: Using pandas to collect and clean data

After the script to collect and clean the data had been written, and tested in a Jupyter Notebook to be able to visualise it properly, it could now be wrapped into a Lambda function. Boto3 was used to instantiate the connection with the proposed DynamoDB tables where the data would be inserted each day. Looping through each row of the pandas DataFrames containing the WHO COVID-19 data, the row is inserted into the table alongside a date timestamp in the form YYYY-MM-DD, the ISO 3166-2 [55] country code, country full name and the TTL (Time To Live) for the database item, which is a week from its date of entry in the Epoch time format.

Using the ISO 3166-2 code for each country was important as most APIs use this as the standard for uniquely referring to a country, so in the same manner this is the sole way a country was identified throughout the application. An example of the data being inserted into the DynamoDB table in a Lambda function can be found in Appendix D.

At this point the Lambda function was able to successfully source, clean and insert COVID-19 case data into a DynamoDB table. The next task was to schedule this Lambda function to run each day to collect up to date information. To do this CloudWatch Events was used, with rules set up to trigger the lambda function to run on a schedule defined using a cron expression [56]. Figure 34 shows the cron expression of '0 8 * * ? *', being used to set the Lambda Function to run each day at 8AM GMT.

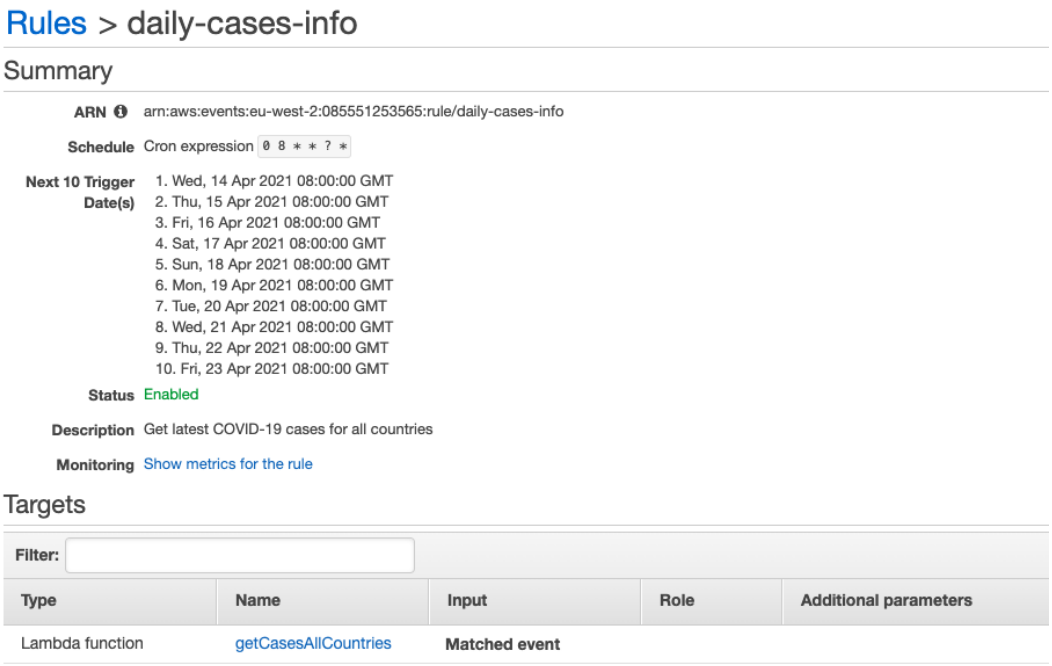


Figure 34: Cron expression defined schedule for Lambda Function triggers

6.2.3 Travel Restrictions Information

The source of the data used to implement a daily sourcing and storage of each country's travel restrictions is the COVID19 API [57], which uses the COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University [58] as its source for data.

Before using this external API programmatically, it was explored using Postman as described earlier in Section 5.3.6. The specific end point used was the *premium travel data by country* and a sample API call can be seen below:

```
https://api.covid19api.com/premium/travel/country/AU
```

Again the use of ISO 3166-2 codes for each country can be seen, with AU representing Australia.

After identifying the required attributes from the JavaScript Object Notation (JSON) response in Postman, using the *requests* library a GET request was made to the API endpoint, and the relevant information including the travel restriction information and their URL source, was inserted into the *country_travel_advice* DynamoDB table for each country alongside the country name, ISO 3166-2 code, and the TTL. Similar to the implementation of providing *COVID-19 Case Data*, CloudWatch Events was used to schedule the Lambda function to run each day, this time using a cron expression of '15 8 * * ? *', to schedule it for 8:15AM GMT.

6.2.4 Lockdown Information

The source of data for lockdown information on countries was a table named *Table of pandemic lockdowns* from Wikipedia [59]. Since the data wasn't available for download or via an API, the data had to be collected through web scraping, using pandas which uses lxml and Beautiful Soup [60] via its *read_html* function [61] to parse the contents of HTML tables into DataFrames.

Since the data wasn't structured in a way where it could be used immediately besides only removing unused attributes, like in the case of the CSV data from WHO on COVID-19 cases, a lengthy cleaning process was required using pandas. The data required from this table was to find the countries currently experiencing a national lockdown as of the start of the current week and the end dates for these lockdowns if specified. An excerpt of the code required to clean and process the data using various pandas functions is shown below in Listing 2, with the resulting DataFrame pictured in Figure 35.

```
# Determining the column with the end date of the latest lockdown
result = []
for index, row in df.iterrows():
    if row['L2'] == 0:
        result.append(row['L1'])
    elif row['L3'] == 0:
        result.append(row['L2'])
    else:
        result.append(row['L3'])

df['Lockdown End'] = result

# Dropping byproduct attributes in determining end date of lockdowns
df.drop('L1', inplace=True, axis=1)
df.drop('L2', inplace=True, axis=1)
df.drop('L3', inplace=True, axis=1)

# Removing dates that are ranges and replacing not "set dates" with zero values
df['Lockdown End'] = df['Lockdown End'].apply(format)

# Dropping any rows that don't have an end date in the format YYYY-MM-DD
df['Lockdown End'] = pd.to_datetime(df['Lockdown End'], format='%Y-%m-%d', errors='
coerce')
df.dropna()

# Applying function to determine countries with a current lockdown
df['Current Lockdown?'] = df['Lockdown End'].apply(testDate)
df = df.loc[df['Current Lockdown?'] == True]
```

Listing 2: Cleaning unstructured data using pandas

Out[46]:

	Country	Lockdown End	Current Lockdown?
17	Bangladesh	2021-04-13	True
19	Belgium	2021-04-24	True
45	France	2021-05-03	True
49	Germany	2021-04-19	True
62	Ireland	2021-04-12	True
68	Italy	2021-04-30	True
118	Poland	2021-04-18	True

Figure 35: DataFrame showing countries currently on lockdown

The countries that are on lockdown are added to the *countries_lockdowns* table in DynamoDB each day, with the Lambda function run each day at 8:20AM, scheduled by CloudWatch Events.

6.2.5 Red List Countries

To be able recommend to the users destinations that they can travel to, it was required to find a list of countries the UK had banned travel to. The official UK government website lists these countries, and defines them as "Red list travel ban countries" [62]. Since this data is not made available via an API, web scraping was called upon.

Using Beautiful Soup and Requests; in combination with Chrome DevTools to inspect and find the HTML element, relating to the list of red list countries, these countries are scraped and parsed, and subsequently added to the *red_list_countries* DynamoDB table. The Lambda function encompassing the aforementioned steps, is scheduled to run each morning at 8:00 AM, scheduled by CloudWatch Events. Listing 3 shows how Beautiful Soup identifies the HTML element relating to the list.

```
1 URL = 'https://www.gov.uk/guidance/transport-measures-to-protect-the-uk-from-variant-
    strains-of-covid-19#red-list-travel-ban-countries'
2   page = requests.get(URL)
3   # find div elements with the 'govspeak' class
4   soup = BeautifulSoup(page.content, 'html.parser').find('div', class_='govspeak')
5   # find all list elements within the unordered list in the govspeak class div
6   countries = soup.find_next('ul').find_all('li')
```

Listing 3: Scraping data from a web page using Beautiful Soup

6.2.6 Email Services

As illustrated in the use case diagram in Section 4.2.2, the email feature can be split into two interactions. The first being when a user initially signs up to receive alerts on a destination of their choice, and the second is the user then receiving weekly updates on the COVID-19 situation of the destination after this. As a result two Lambda functions were written to handle each situation.

The 'new user' function, simply sends the user an email to verify their email address, as can be seen in Figure 36, and after verification redirects them to a page on the website confirming the success or failure of the verification. This is facilitated through Amazon SES, with a custom verification email template being created to be able to include a custom message and redirection links. Using the *newUserEmail* Lambda function, the verification email, using the created template, can be sent to a user, executed by an API call to the API Gateway endpoint targeting it. Within the same Lambda function, the users' email address, their chosen destination, and that destination's country code, is inserted into the *Subscriptions* DynamoDB table.

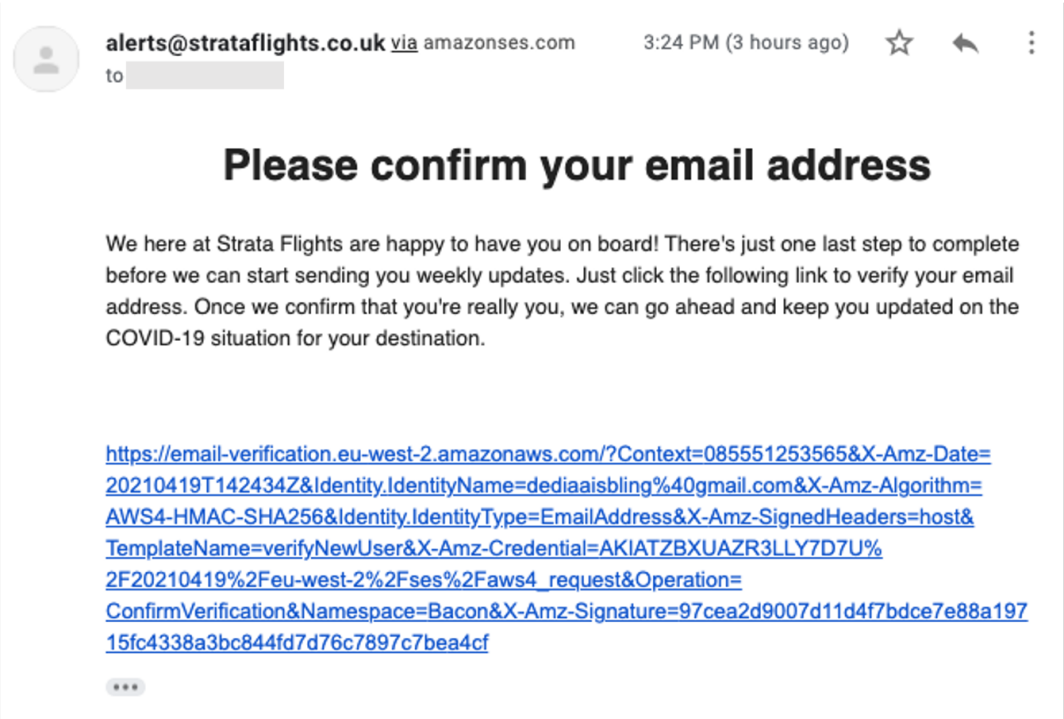


Figure 36: Screenshot of custom Amazon SES verification email

The second Lambda function, *CovidUpdateEmail*, is scheduled to run weekly in CloudWatch Events, and sends emails to everyone in the mailing list contained in the DynamoDB table *Subscriptions*, with COVID-19 information for the country of the destination they subscribed for alerts for. Only users who have had their email verified are sent emails. This was done to add an extra layer of security, ensuring

the email address is valid to reduce the risk of issues such as bounces - when an email is returned to the sender because it cannot be delivered. An example of an email sent to a user can be seen in Figure 37.

Figure 37: Screenshot of a COVID-19 update email

6.3 Client

This section will detail how each feature of the client was implemented, discussing any tools, services, and libraries used in its implementation.

6.3.1 Introduction

Alongside the use of some of the UI component libraries mentioned in Section 5.4.1, the Treact [63] template was used as a base template for the client-side of this application. Already packaged with tools such as TailwindCSS and styled components, this template made it simpler to get started and ensured the wheel wasn't being re-invented setting up a new React project with react-router amongst other packages. As a result, more time was spent creating new components that were to be used for the business requirements of the application, as opposed to trivial components such as navbars and footers which contribute mainly to the design. As stated by Khan [63] in Treact's License, "Treact is free for personal as well as commercial use", so using it in this situation doesn't violate its terms of use in any way.

In the main stylesheet for the project, *tailwind.config.js*, the font and base colours were changed to use the Nunito font family and colour palette, both designated in the Key Styling and Palette Choices section. The default logo was also replaced with a separate logo image and the brand name of "Strata" chosen.

6.3.2 Homepage/Main Dashboard Page

The Homepage has 3 main features: the 'mini' search feature with limited filters, the update sign up form, and a preview of the Travel Ideas feature.

'Mini' Search feature

The 'mini' search feature allows users to start their search, right from the homepage, before reaching the dedicated search page. It allows the user to input the minimal information required to query flights, that will then be listed on the search results page.

To facilitate this, the LookupInput component was created. This component allows users to search either using city or airport code, and suggests options based on the current input in the text box. An example of this can be seen in Figure 38.

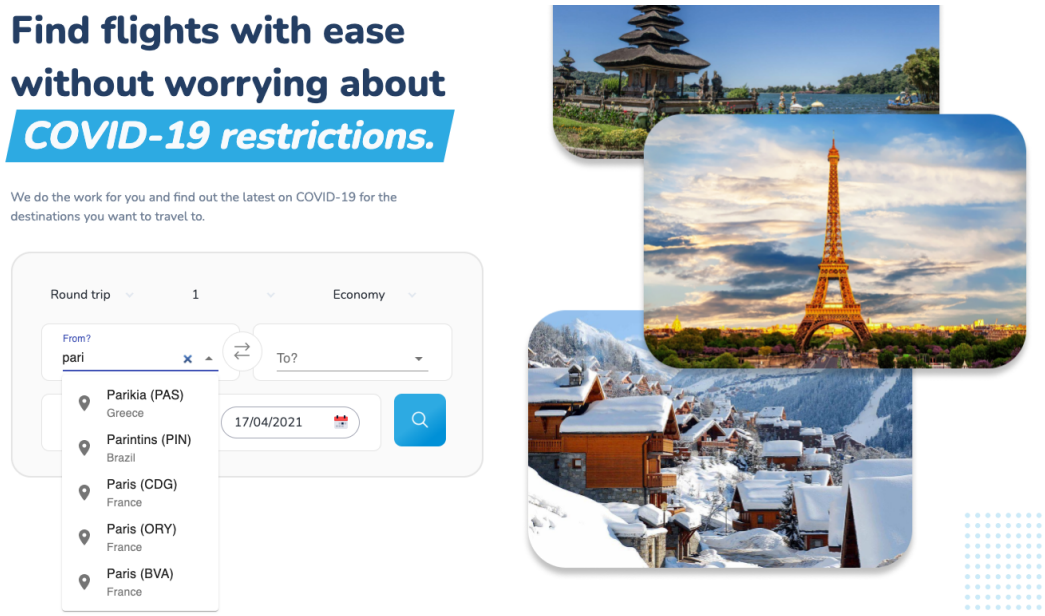


Figure 38: LookupInput component in use on the homepage

On every letter input, through the onInput change event of Material-UI's AutoComplete component, the *query* state variable is set to the the text that has been inputted.

Via the useEffect hook, an API call is made to the Kiwi API's Locations endpoint using the *query* state variable, which returns a list of locations that are then listed as options. The useEffect hook [64] tells React to perform its contents any time the *query* state variable is changed, as can be seen in Listing 4.

```
useEffect(() => {
  const fetchData = async () => {
    if (query !== "") {
      await axios
        .get(`
          https://xtogjhen60.execute-api.eu-west-2.amazonaws.com/dev/locations/query?
  term=${query}&locale=en-US&location_types=airport&limit=50&active_only=true`
        )
        .then((response) => {
          setLocations(response.data.locations);
        });
    }
  };
  fetchData();
}, [query]);
```

Listing 4: UseEffect Hook in React

The returned list of locations is filtered with the user's current input using the match-sorter library, with the airport code (id) taking precedence and then the city name. This is highlighted in Listing 5.

```
  const filterOptions = (options, { inputValue }) =>
  matchSorter(options, inputValue, {
    keys: [
      "id",
      { threshold: matchSorter.rankings.STARTS_WITH, key: "city.name" },
    ],
  });
```

Listing 5: Match-sorter library filtering locations

The date picker and the filters use components from the React Rainbow UI component library. Since the date picker, filters, and the LookupInput, are child components of the parent component searchContainer, their state has to passed upstream in order for them to be accessed by searchContainer whenever it changes. Each has a callback function that is called during its onChange event, that passes the changed value to the parent component searchContainer via a function passed down in the *search* props, which then in turn sets the relevant state variable in searchContainer with that value. An example of a callback function that passes a value to the searchContainer parent component can be seen in Listing 6.

```
    submit = (keyword) => {
    if (typeof this.props.search === "function") {
      this.props.search(keyword);
    }
  };
```

Listing 6: Callback function to pass state from child to parent component

Now the searchContainer component can access the values of all the fields, it can then pass the values, through its state variables, with the React Router useHistory hook. This passes the state variables to be stored in the browser's history, so that it can be used on another page, and in this case it will be used to form the query to display the search results on the search page. This is executed when the user clicks on the search button.

The update sign up form and the travel ideas preview will be discussed in the Update Sign Up Form and Travel Ideas Page sections respectively.

6.3.3 Search Results Page

The search results page also has three main components, including the 'max' search feature which is very similar to the mini search but with a greater selection of filters to choose from, such as picking direct flights only, or changing the times of departure or return. The other components are the flight card, which is used to display the results of the flight query, and the COVID-19 snapshot, which gives the user COVID-19 information based on the country of the chosen destination.

'Max' Search Feature

The max search feature is essentially the same as the mini search with the addition of more filters below it. If the user reaches the search results page via a search performed on the homepage, the max search is populated with the data the user entered in the original search. This is made possible using the state variables passed in the browser history by react router. The variables are accessed using the useLocation react router hook.

The filters are a button group and through each button's onClick event, its value is set to either true or false, depending on if it had been previously selected or not, and then via a useEffect hook, the current values of all the filters are passed up to the parent component SearchResults. This means whenever a filter is selected or deselected, its value becomes part of the API parameters used to get data, that will then go on and generate the flight cards, giving the effect of the search refreshing each time a filter is applied. Using styled components and a ternary operator, when a filter button is selected, it is shaded blue instead of the default white, and it also then has a cancel button on it to give the users the option to deselect it, of which an example can be seen in Listing 7.

```
background: ${(props) =>
  props.selected
    ? css`
        linear-gradient(
          270deg,
          ${(props) => props.theme.colors.dodgerBlue2} 11%,
          ${(props) => props.theme.colors.dodgerBlue3} 23%,
          ${(props) => props.theme.colors.dodgerBlue} 50%
        );
      `
    : props.theme.colors.white};
```

Listing 7: Changing button colour depending on its state

The effect of this is pictured Figure 39.

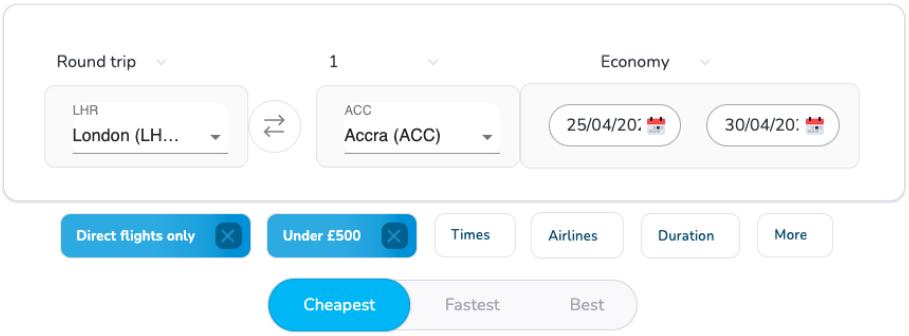


Figure 39: Search filters as part of the max search feature

Flight Card

The state variables set by the users search, and any applied filters, are passed as props to the SearchCard component. These variables then become part of the parameters of an API call to Kiwi's Search endpoint, and the return data is mapped across the FlightCard component, generating a number of FlightCards containing the information for each flight itinerary. Any time any of these variables change, the SearchCard component, via the useEffect hook, re-renders and makes a new API call to the Search endpoint, and in so doing, generates new FlightCards with the the data from the new search query. This gives the feature of the FlightCards updating every time the user changes a search parameter or selects/deselects one of the filters. Listing 8, shows the API response, stored in the *flights* variable, being mapped across the FlightCard component.

```
flights.map((flight, index) => {
  if (props.oneWayOrReturn === "Round trip") {
    return (
      <StyledCard key={index}>
        <FlightCard
          obdDptTime={GetISOTime(flight.route[0].local_departure)}
          obdArrTime={
            checkStops(flight, "num", "obd") === "direct"
              ? GetISOTimeArr(
                  flight.route[0].local_arrival,
                  flight.route[0].local_departure
                )
              : GetISOTime(flight.route[1].local_arrival)
          }
          obdDuration={DurTime(flight.duration.departure)}
          obdAirline={flight.route[0].airline}
          obdDptAirportCode={flight.flyFrom}
          obdArrAirportCode={flight.flyTo}
          ....
          ....
        />
      </StyledCard>
    )
  }
  ....
}
```

Listing 8: Mapping data across FlightCards

Figure 40 shows the FlightCards displaying the flight itinerary, with information from the response of a call to Kiwi's API Search endpoint.

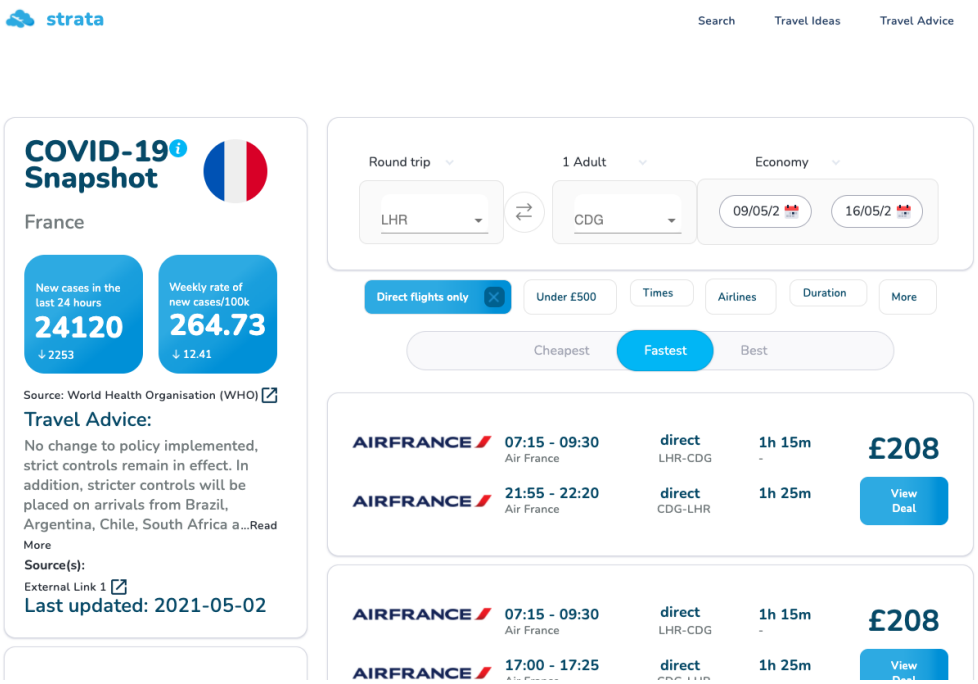


Figure 40: Screenshot of the search results page

COVID-19 Snapshot

Figure 40 also displays the COVID-19 snapshot, which is one of the key features of this web application. It displays COVID-19 information for the country of the destination selected in the search box, such as the number of new cases in the last 24 hours, the weekly rate of new cases per 100,000 people, along with the country's latest travel advice, and sources and links to all the data.

Figure 41: Screenshot of the COVID-19 snapshot expanded

It can be expanded when clicked on to display more information, such as whether a country is on lockdown, or if they have any quarantine restrictions, as seen in Figure 41.

Here is where the API endpoints created using Amazon API Gateway are consumed in order to display the information. The country code (code) and current date (date) are used as parameters for each API call, to get information specific to the current date and destination country's code. A concurrent API call, facilitated using the axios.all function, is executed within the useEffect hook, so that each time the destination changes, another API call is made to get new information, specific to the newly selected destination. This can be seen in Listing 9.

```
useEffect(() => {
  const fetchData = async () => {
    await axios
      .all([
        axios.get(
          `https://3tmuo3iuhk.execute-api.eu-west-2.amazonaws.com/dev?code=${code}&date
=${date}`
        ),
        axios.get(
          `https://hmbrr2y0jg.execute-api.eu-west-2.amazonaws.com/dev?code=${code}&date
=${date}`
        ),
        axios.get(
          `https://0glnns5f5h.execute-api.eu-west-2.amazonaws.com/dev?code=${code}&date
=${date}`
        ),
      ])
      .then((response) => {
        setData(response[0].data.body.Item);
        ....
        ....
      .catch((err) => {
        if (err.response) {
          setData("N/A");
          setCases({ dailyCases: "N/A", rate: "N/A" });
          setDiffs("N/A");
          setLockdowns("N/A");
        }
      });
  };
  code.length === 2 && fetchData();
}, [code, date]);
```

Listing 9: Execution of concurrent API calls

To expand the snapshot, a modal is used from React Rainbow's library with the same data as the unexpanded COVID-19 Snapshot, alongside additional data such as information on lockdown restrictions, displayed on the modal due to the additional space. The modal is opened when the user clicks on the snapshot, and can be exited using the exit symbol on the top right hand corner, or by clicking the area outside the modal.

There is also a small blue information icon on the COVID-19 snapshot, pictured in Figure 42. When the user hovers over this icon, they can view a disclaimer regarding the information seen on the snapshot. This was implemented to satisfy the requirement of including a disclaimer as a counter-measure to the risk of "Not being able to accurately report on a country's restrictions", specified in Table 1. The HelpText component in the React Rainbow library was used to facilitate this.

Figure 42: Disclaimer message displayed hovering the icon

6.3.4 Update Sign Up Form

The majority of inputs across the application are limited to pre-defined options, such as with the destinations and dates inputs. This limits any risk of an injection attack. The email address input as part of the form to sign up for updates cannot of course being predefined. and the user can technically input whatever they wish to. This means that this field requires input validation.

Since the form uses the *email* input type [65], its contents are validated to ensure it is a properly formatted email address, before the form can be submitted. This doesn't necessarily ensure that it is a valid email address, but it at least ensures malicious scripts can't be injected into the form and submitted. Figure 43, shows input validation in action on the email input of the form.

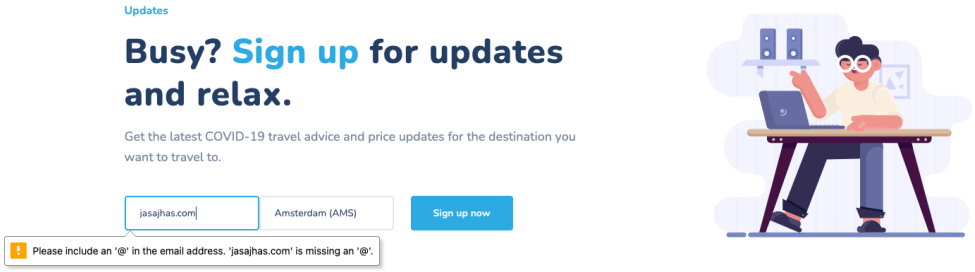


Figure 43: Screenshot of the sign up form

To ensure the email address is validated, it is verified through Amazon SES, with the user being asked to click the link sent to their email to confirm their email address, before any new emails are sent through to them. On the success of verification, the user is redirected to a page on the website confirming the success of the verification, or if the process failed, a separate page asking them to try verifying their email address again, as both seen in Figure 44.

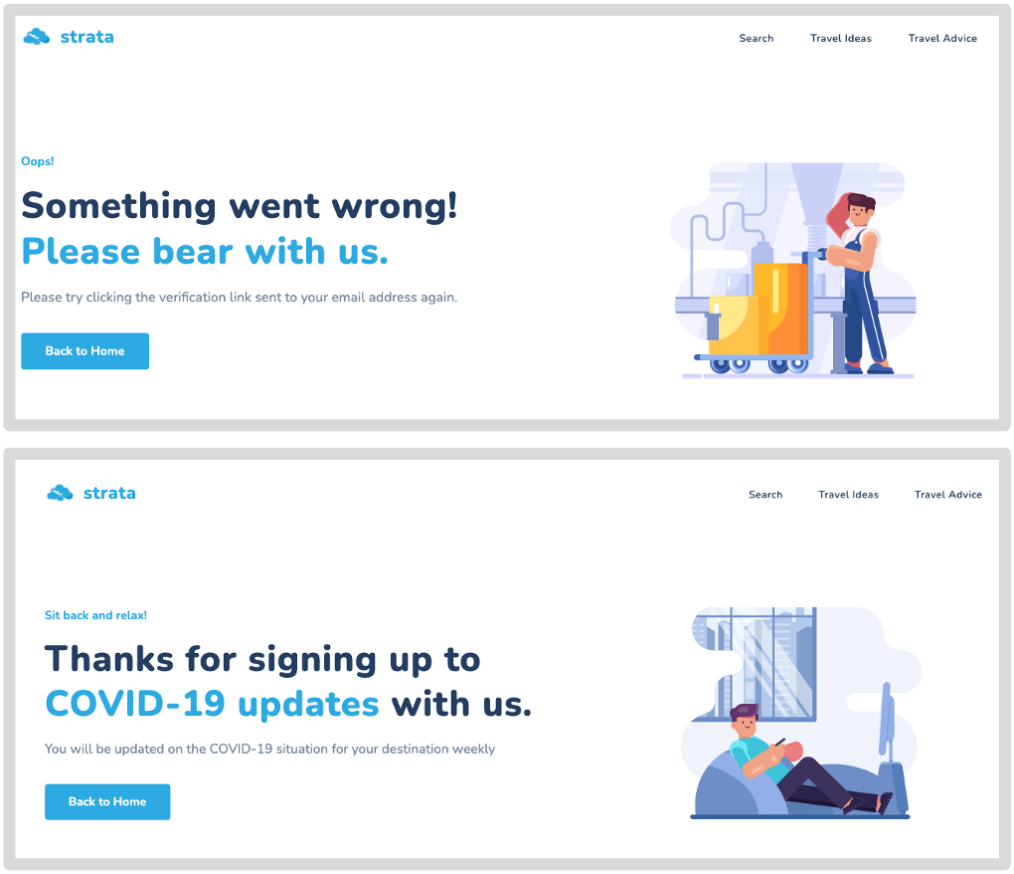


Figure 44: Confirmation pages after verifying email address

6.3.5 Travel Advice Page

The travel advice page focuses on delivering information on COVID-19. It's focal point is the TravelMap component, which is an interactive map created with React Simple Maps, supplied with data from the API Gateway endpoint to the getMapData Lambda function. A screenshot of the page can be seen in Figure 16.

Figure 45: Screenshot of Travel Advice Page

For React Simple Maps to generate the map, it is supplied coordinate and geometry data for countries globally using a topojson file [66] publicly available. Data made available from the API Gateway endpoint to the getMapData lambda function is also supplied to the map, sampled as follows:


```
{
  "code": "IS",
  "dailyCases": "18",
  "rate": "26.64",
  "lockdown": "No"
},
{
  "code": "IT",
  "dailyCases": "10398",
  "rate": "151.65",
  "lockdown": "Yes"
},
```

As both the topojson file and the data from getMapData used ISO 3166-2 code to identify countries, the relevant country on the map was associated with its corresponding COVID-19 information. With the data, the map could now be colour coded based on a set scale as seen in Listing 10.

```
const colorScale = scaleLinear()
  .domain([0, sort === "rate" ? 350 : 20000])
  .range(["#ffedea", "#ff5233"]);
```

Listing 10: Setting scale to colour code countries on the map

For cases in the last 24 hours, the scale chosen was 0 to 20000, and for weekly rate of cases, the scale was 0 to 350. The colour range chosen is between a very light shade of red, hex colour #ffedea, and a very dark shade of red, hex colour #ff5233. Using the the corresponding COVID-19 data for each country, it is designated a colour within that range based on the given numerical scale.

The data was also used in other ways, such as to display the number of cases in the last 24 hours, when a user hovers over a specific country, as seen in Figure 46. This was made possible using the tooltip component from React-tooltip.

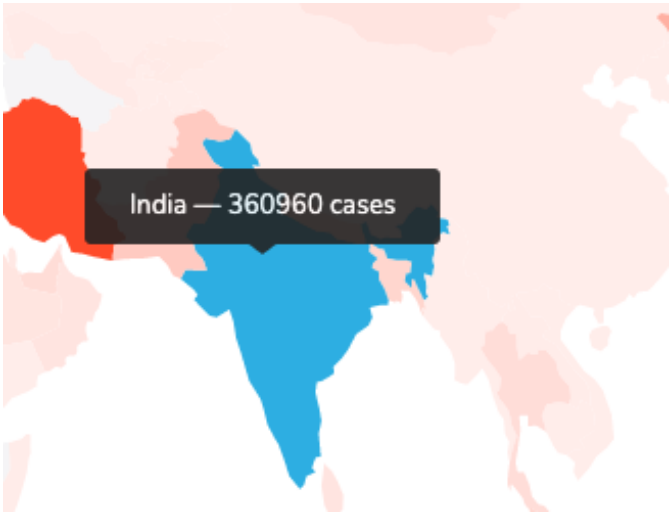


Figure 46: Tooltip showing number of COVID-19 cases

With the data, it was also possible to highlight on the map which countries are on lockdown, as seen in Figure 47, where countries on lockdown are highlighted green.

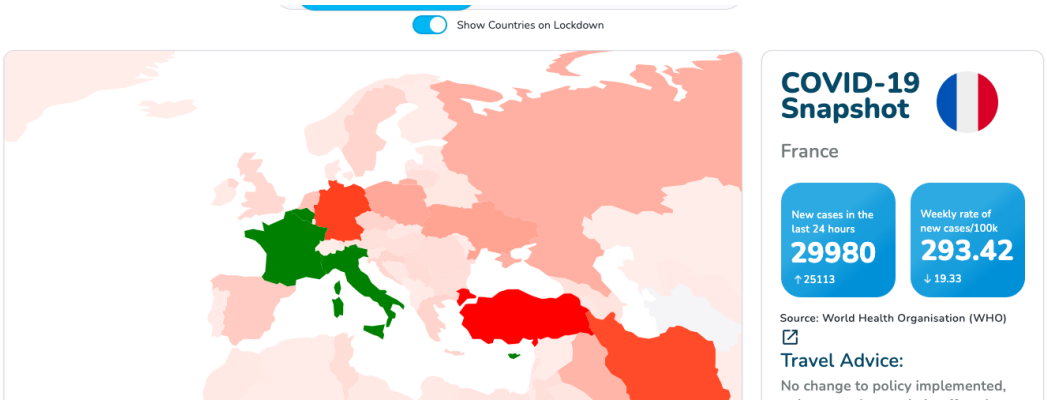


Figure 47: Countries on lockdown being highlighted

A toggle is used which can set the *lockdown* state variable as true or false. When set true, countries that are on lockdown are set to be filled with green, as can be seen in the code excerpt in Listing 11.

```
fill: `${
  lockdown && d !== undefined
    ? d["lockdown"] === "Yes"
      ? "green"
      : geo.rsmKey === selectedBtn
      ? "#064967"
      : d
      ? colorScale(d[sort])
      : "#F5F4F6"
    : geo.rsmKey === selectedBtn
    ? "#064967"
    : d
    ? colorScale(d[sort])
    : "#F5F4F6"
}`,
```

Listing 11: Conditionally setting the colour of a country

As also seen in Figure 16, by clicking on a country, it is highlighted blue, and its corresponding COVID-19 Snapshot is shown to the right of the map, in a similar manner as how it is used on the Search Results page, using the ISO 3166-2 code of a country to call and display it's data.

6.3.6 Travel Ideas Page

The Travel Ideas page, seen in Figure 48, is used to recommend destinations that users from the UK can travel to and are not in one of the current red list travel ban countries [62].

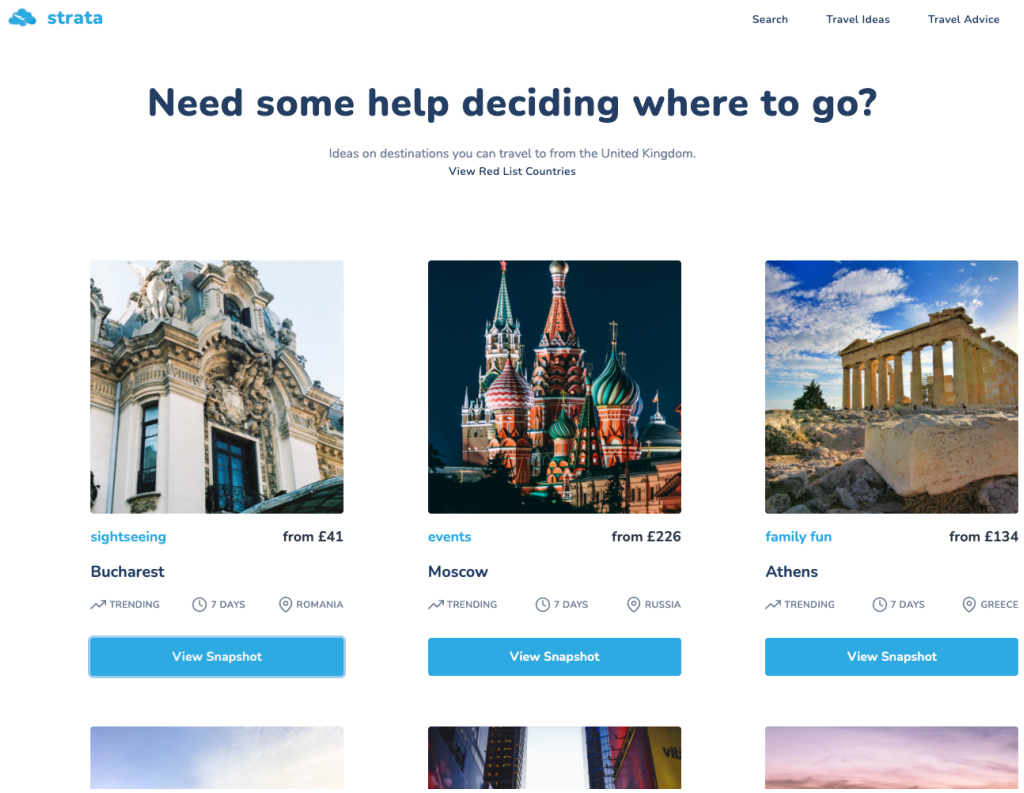


Figure 48: Screenshot of the Travel ideas page

To do this, Kiwi's API /locations/topdestinations endpoint is used to retrieve a list of the most popular destinations searched for by UK based users of Kiwi. These destinations are then verified to ensure they aren't in any of the red list countries, and then subsequently displayed on the page. This is made possible with the data collected daily by the Lambda function discussed in Section 6.2.5. An API Gateway endpoint is called, which retrieves the list of red list countries and their ISO 3166-2 codes. This verification process can be seen in Listing 12, where *'codes'* is the list of codes of red list countries, and *'countries'* is the list of codes of the popular destinations from Kiwi's API.

```
.then((countries) => {
  countries.forEach((country) => {
    if (codes.indexOf(country.country.id) !== -1) {
      countries.splice(countries.indexOf(country), 1);
    }
  });
  return countries;
})
```

Listing 12: Checking if country is not in the red list

Using Kiwi's /aggregation_search/price_per_city endpoint, the price of flights to each destination were

retrieved, and Unsplash's API was used to source and display an image of each destination.

Additionally, a list of the countries on the red list can be viewed in a modal, by clicking on the "View Red List Countries" text just above the grid of destinations, as seen in Figure 49. It also includes a link to the official UK government website, where the red list countries are sourced from, in the footer of the modal, also pictured in Figure 49. React Rainbow was again used to provide the modal for this feature.

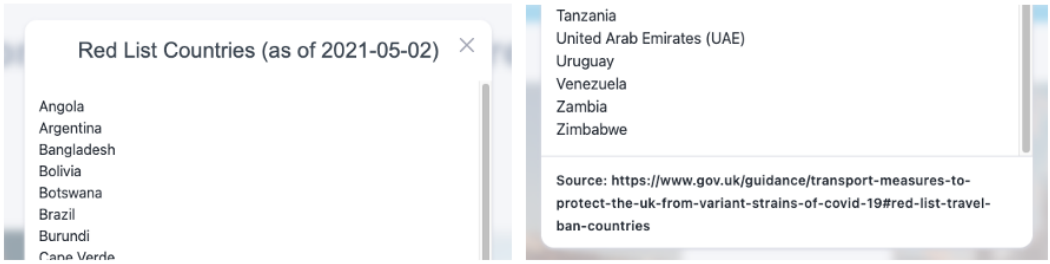


Figure 49: Modal listing countries on the red list

By clicking on the "View Snapshot" buttons, users can also view the COVID-19 snapshot for the country of the destination, similar to its implementation on the Travel Advice and Search Results pages, however this time contained in a React Rainbow drawer component, as seen in Figure 50.

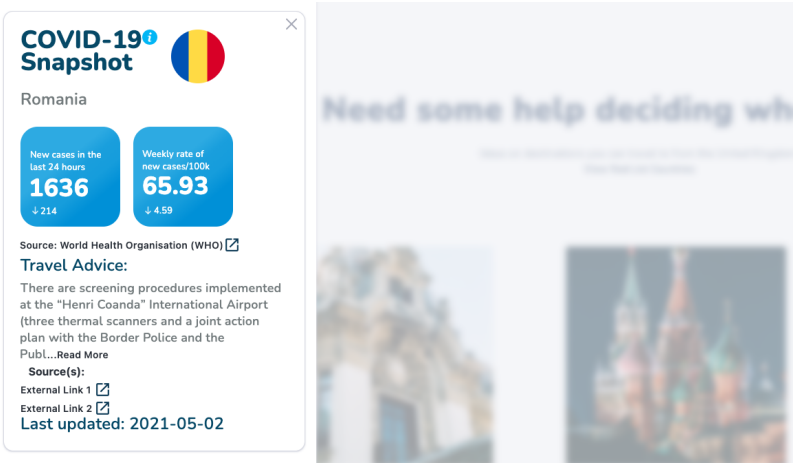


Figure 50: COVID-19 Snapshot in a drawer

6.4 Other Considerations/Issues encountered

6.4.1 Lambda Additional Layers

Amazon Lambda ships Python 3.8 which comes with the default Python standard library, however when using external packages such as *requests* and *pandas* from PyPI (Python Package Index) - Python's official repository of packages [67], Lambda layers were required to provide these PyPI libraries for the lambda functions. Lambda layers are zip files that can contain code libraries, custom runtimes and other dependencies [68].

The steps taken to build custom layers to upload to use with the lambda functions are detailed below.

1. Saving a list of dependencies used in the projects virtual environment to a requirements.txt

```
pip freeze > requirements.txt
```

2. Creating a python folder which will contain the libraries used for the layer.

```
mkdir python
```

3. Building dependencies in a docker container that mimics Lambda's runtime environment (provided by lambci [69])

```
docker run --rm \
--volume=$(pwd):/lambda-build \
-w=/lambda-build \
lambci/lambda:build-python3.8 \
pip install -r requirements.txt --target python
```

4. Creating a zip file of the python folder with the dependencies now installed inside it.

```
zip -r layer python/
```

5. Finally, uploading the zip file to AWS Lambda to be used a layer, as shown in Figure 51.

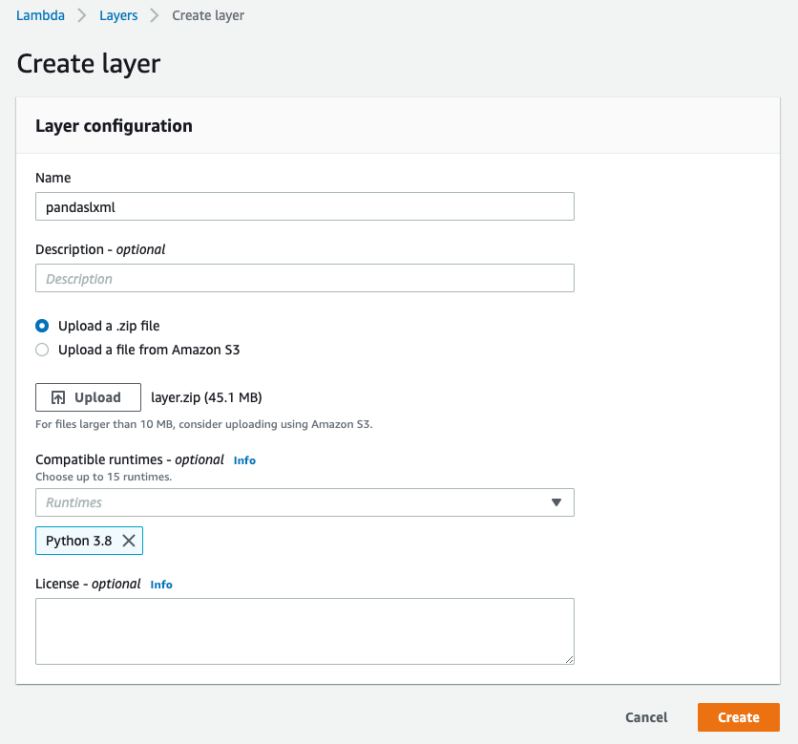


Figure 51: Uploading zip file to use to create a layer on Amazon Lambda

Once these steps were taken, by adding the layer to a Lambda function, any external PyPI libraries used in any of the lambda functions could be imported inside of the default lambda runtime. The instructions required for this process were derived from [70].

6.4.2 API Gateway Setup

Separate Lambda functions were required to retrieve data from the DynamoDB tables to be called by an API Gateway endpoint. Figure 52, which also includes Amazon S3, of which its involvement will be explained when discussing the deployment of the application, highlights the interactions between the different AWS services in a serverless web application. An example of a Lambda function called by API Gateway to retrieve data stored in DynamoDB can be found in Appendix E.

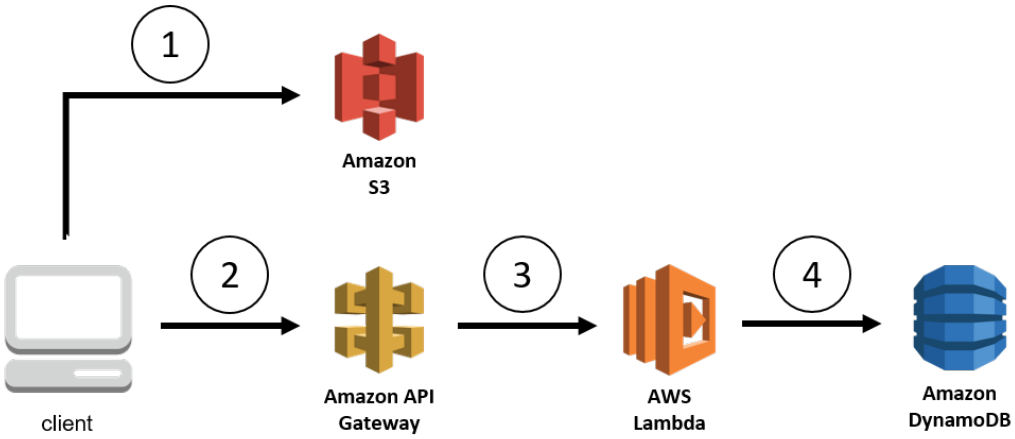


Figure 52: Serverless 101 architecture. Adapted from [71]

Creating a REST API on API Gateway was relatively straightforward, however after deploying an endpoint, it was realised that the API couldn't be consumed by the front end due to a XMLHttpRequest error. The Cross-Origin Resource Sharing (CORS) policy had not yet been set up. According to Mozilla [72], CORS is "an HTTP-header based mechanism that allows a server to indicate any other origins (domain, scheme, or port) than its own from which a browser should permit loading of resources". An example of a cross-origin request is illustrated in Figure 53. For security reasons, browsers typically block cross-origin HTTP requests unless the response from the other origin includes the appropriate CORS headers, which wasn't the case with the API Gateway endpoint that had been deployed. To enable CORS headers on the API Gateway endpoints, the following response headers and default values had to be set:

- X-Requested-With: '*'
- Access-Control-Allow-Headers: 'Content-Type,X-Amz-Date,Authorization,X-Api-Key,x-requested-with'
- Access-Control-Allow-Origin: '*'
- Access-Control-Allow-Methods: 'POST,GET,OPTIONS'

The Access-Control-Allow-Origin header value allows the request to be made by any origin; the Access-Control-Allow-Methods specifies the accepted HTTP methods; the Access-Control-Allow-Headers indicates which headers can be used as part of the request; and the X-Requested-With header specifies the request can be requested by any type of request, e.g. XMLHttpRequest.

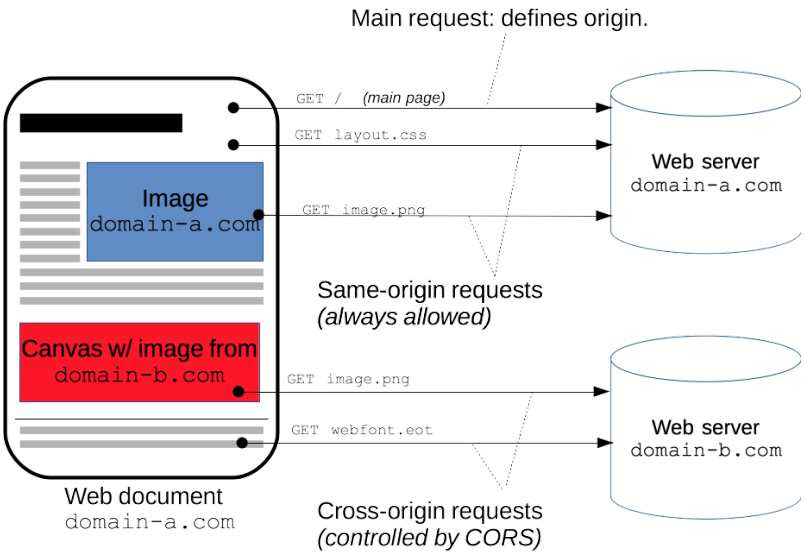


Figure 53: Example of a cross-origin request. Adapted from [72]

6.4.3 Database Archiving

Within some of the DynamoDB tables, data is being stored each day, resulting in tables increasing in size and becoming very large if not dealt with. This can incur high costs and use up any set limits on database sizes, but also as Olson [73] states, if data is kept in operational systems long after it is being used, there are greater possibilities for data quality erosion occurring. With this in mind, a TTL was enabled on items added to databases, which have new items being added to them each day.

The Time to Live (TTL) functionality automatically deletes expired items from DynamoDB tables [74]. It uses an attribute that has been designated as the TTL attribute, which refers to Epoch time of when an item should be deleted. As mentioned previously when discussing the data inputted into the tables, a TTL (Time To Live) attribute for each database item was added, equal to a week from its date of entry in the Epoch time format.

Epoch time can be described as the "number of seconds that have elapsed since January 1, 1970 (midnight UTC/GMT)" [75], and it is used in Unix systems and also in other systems that using scheduling. This tells DynamoDB to automatically archive or delete the item when this Epoch time has passed. A week was chosen as this is the date where there is no longer a business requirement deeming the data operational beyond this time. Figure 54 shows TTL being enabled on the *countries_covid_cases* DynamoDB table, with a preview of items expiring a week from their input date.

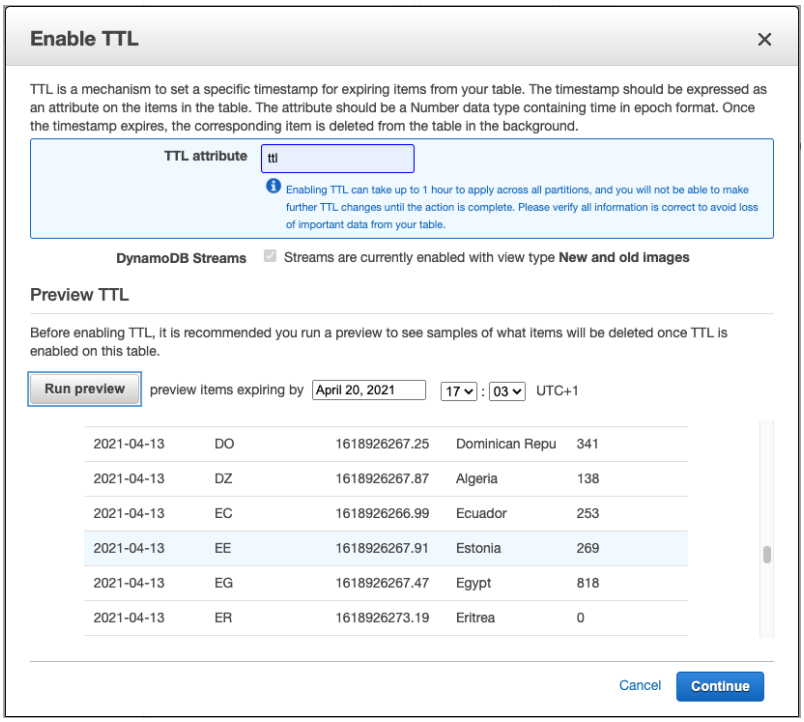


Figure 54: TTL enablement on a DynamoDB table

6.4.4 Cloud Security Considerations

Cloud applications, like the one being developed in this project, are vulnerable because they are internet facing; are usually associated with handling lots of data, which can be can sold off or held to ransom by the attacker; and because there are also many tools available to carry out such attacks, including but not limited to DDOS-as-a-service.

In a serverless model, the burden of responsibility for security is shared between the application owner and the cloud service provider, in a concept called the Shared Responsibility Model. In this case, the cloud service provider, AWS, must handle the security of computing and network infrastructure, operating system configuration, encryption of data and network traffic, and the installation of network and security patches[76]. This leaves the application owner to ensure the application is written keeping security in mind; reducing the number of potential attack vectors by limiting errors and vulnerabilities in code, for e.g. exposing app secrets; and proper monitoring and logging, where any confidential data is not exposed in logs.

With all of this in mind, when developing the serverless backend, and in particular, the Lambda functions, the OWASP (Open Web Application Security Project) Serverless Top 10 [77] was consulted. The Serverless Top 10 discusses the top 10 threats to serverless applications, including potential attack vectors, their potential impact, how to prevent them, and example attack scenarios. An example is the

Sensitive Data Exposure threat, which alludes to how exposed keys in the Lambda function code can lead to unauthorised access or actions, and storing sensitive data in plaintext or with weak cryptography can lead to exposure.

A number of steps were taken in the implementation of the serverless backend considering its security and are listed below:

- Following the least-privilege principles and having an IAM role for each Lambda function that only has enough permissions to carry out its specific task. For e.g. a function that only needs to retrieve data from DynamoDB shouldn't be able to write data to DynamoDB. Figure 55 shows the *fetchLockdownInfo* function, which only retrieves data from DynamoDB, having the read-only policy being given to its execution IAM role, which is only used for the *fetchLockdownInfo* function.

Figure 55: Giving an execution role DynamoDB read-only permission

- Validating any uncontrolled inputs (inputs where the user is not limited in what they can enter) to prevent the risk of injection. Again not using roles that have liberal access helps to limit potential attack vectors.

- Not stating any application secrets such as third-party API keys in plaintext in the function code, but instead keeping them abstracted from any code pushed to a Git repository. This was done by using API Gateway's HTTP proxy integration to make calls to third parties through the serverless backend, and return the response directly to the frontend. The front end calls an API Gateway endpoint, and never knows the API key of the third party.

- Preventing DoS (Denial-of-Service) attacks and/or financial exhaustion by setting timeouts on Lambda functions and making use of the API request limit and throttling settings provided by API Gateway. In Figure 56, the throttling settings have been set way below the default values of 10,000 request per second and 5,000 burst, to align more with the actual current needs of the application.

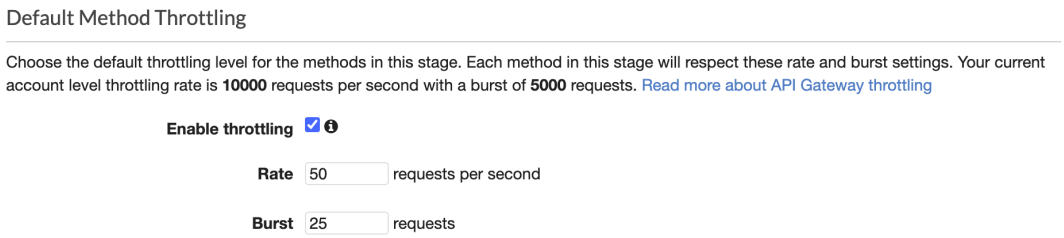


Figure 56: Setting request rate and throttling on API Gateway

6.4.5 Responsiveness

In order to make the web application responsive, Material-UI's Grid system and Hidden utility were used extensively, alongside TailwindCSS's breakpoint utility classes. Listing 13 shows the items within the Grid item being set to have a column width of 12 when within the 'xs' breakpoint, and a width of 6 from the 'sm' breakpoint and above. The Hidden utility is also used to hide the SwitchCircle element from the xs breakpoint downwards.

```
<Grid item xs={12} sm={6}>
  <FromBox>
    <LookupInput term="From?" search={search} />
  </FromBox>
  <Hidden xsDown>
    <SwitchCircle>
      <img alt="switch" src={switch_icon} />
    </SwitchCircle>
  </Hidden>
</Grid>
```

Listing 13: Implementing responsiveness in components

Figure 57 shows Google Chrome DevTools being used to test the responsiveness of the web application at different viewports sizes.

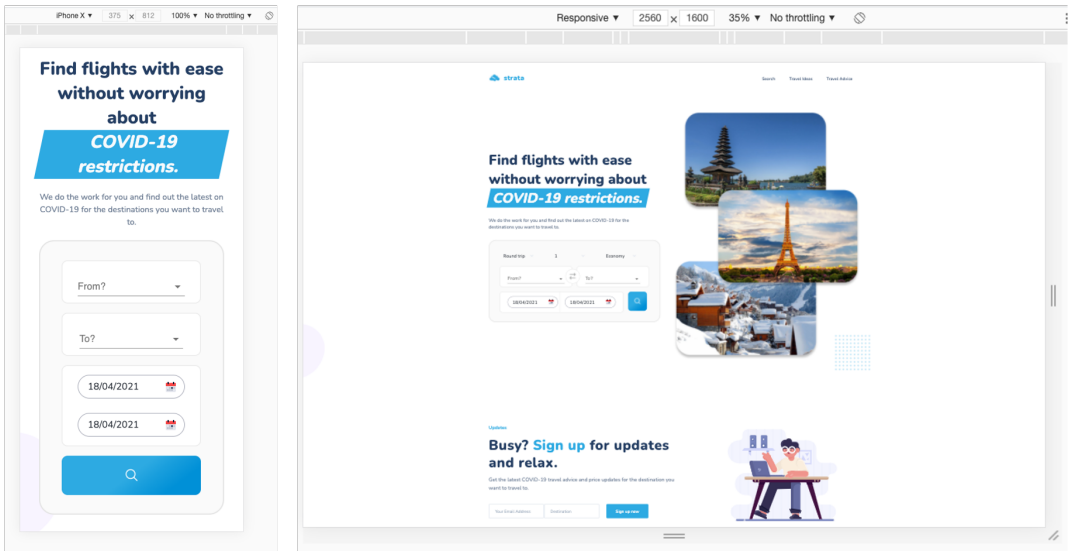


Figure 57: Testing responsiveness with different device sizes

6.5 Conclusion

The Agile approach to implementation helped to stand up the application very quickly, albeit incomplete and very basic, but with the help of automated deployments, discussed further in the Deployment chapter, new features and improvements were continually pushed out to fulfil the requirements specified in the Full Requirements Specification. A wide range of tools were used to build the solution, which all played a significant part in creating the features to meet the requirements.

Chapter 7

Deployment

7.1 Introduction

This section will describe the process of initially deploying the web application, and then setting up a CircleCI pipeline in order to facilitate automatic deployments, each time the development branch was merged with the master branch in GitHub. An initial step was to obtain a custom domain name that would be used for the web application. The domain "strataflights.co.uk" was acquired on LCN.com [78].

7.2 AWS Services

The AWS services involved in facilitating the deployment of the web app are as below:

S3 - "Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance." according to AWS [79]. It was used here to host the web application.

CloudFront - "Amazon CloudFront is a fast content delivery network (CDN) service" that securely delivers data, and in this case, it was used to deliver the application hosted in S3 securely, distributed across locations globally so that users access the site from the location nearest to them, decreasing latency, whilst also offering HTTPS support alongside "both network and application level protection" [80].

Route 53 - Route 53 is a "cloud Domain Name System (DNS) web service" [81]. It was used to route users to the web application hosted in S3 via CloudFront using a custom domain.

Certificate Manager - AWS Certificate Manager (ACM) "is a service that lets users easily provision,

manage, and deploy public and private Secure Sockets Layer/Transport Layer Security (SSL/TLS) certificates for use with AWS services" according to AWS [82]. It was used in this case to create an SSL certificate to allow the application to be deployed over HTTPS securely when used with CloudFront. Using HTTPS means that data incoming and outgoing on the website is encrypted.

7.3 Initial Setup

Before the deployment was set up within a pipeline to be automated, a manual deployment was performed after the initial configuration and setup of the AWS services used. The first action was to setup the S3 buckets that would host the application. Two S3 buckets were created, one named after the root domain, "strataflights.co.uk" and another named "www.strataflights.co.uk". Static website hosting was enabled on both buckets, with the root domain bucket set up for bucket hosting, and the www bucket used to redirect requests to the root domain bucket. The following bucket policy was then set for both buckets to make them publicly accessible:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadForGetBucketObjects",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::strataflights.co.uk/*"
        }
    ]
}
```

Following this, using the command *yarn build*, the react app was bundled into static files to be used for production and subsequently, the contents of the generated *build/* folder was uploaded to both of the S3 buckets. Ahead of the next steps, a new hosted zone was created with the custom root domain (strataflights.co.uk) in Route 53. NS type records were created automatically after creating the zone, which were then copied to LCN.com to change the nameservers to use AWS' nameservers instead of LCN.com's, and essentially assign control of DNS routing to Route 53.

Next, to begin the process of routing the application through CloudFront, using ACM an SSL certificate was requested and then validated by adding a CNAME record to the DNS configuration in Route 53 for the domain. Figure 58, shows the SSL certificate for both the root domain name "strataflights.co.uk" and "www.strataflights.co.uk" being issued and successfully verified.

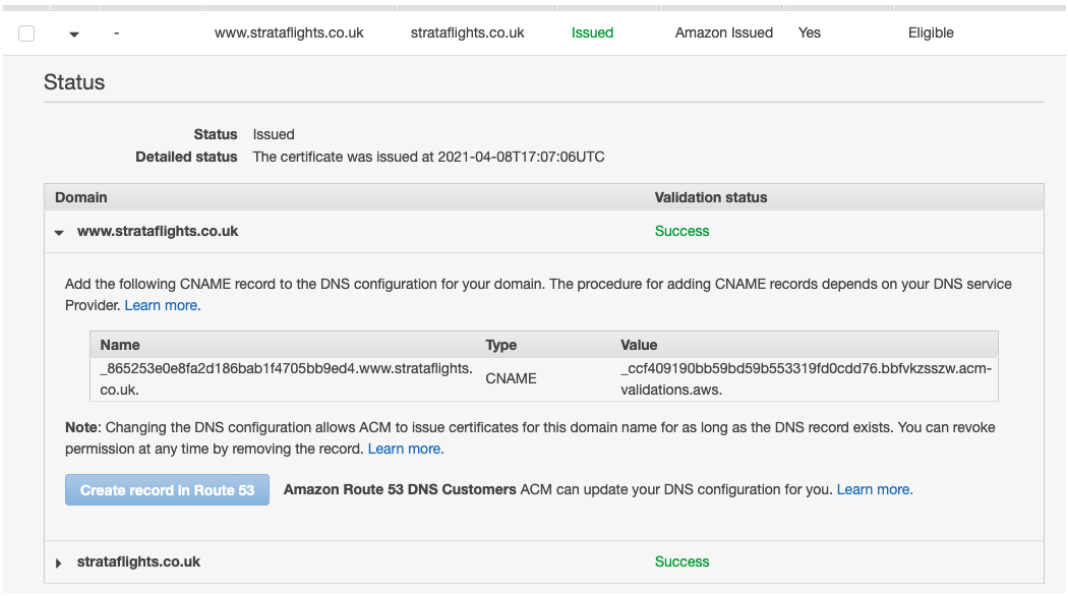


Figure 58: Requesting an SSL certificate in ACM

Next a CloudFront distribution was created using the SSL certificate, set up to use all edge locations which gives the best performance. After the CloudFront distribution had been deployed, the last stage was to create 'A' and 'AAAA' records in Route 53 to redirect traffic from the domain name acquired, "strataflights.co.uk", and it's www counterpart, to the CloudFront distribution domain (xxx.cloudfront.net), where the application is now hosted. 'A' records are usually used to map a hostname and its IPv4 address, and 'AAAA' records used for IPv6, but in this case they were mapped to the CloudFront distribution as an alias target. This setup can be seen in Figure 59.

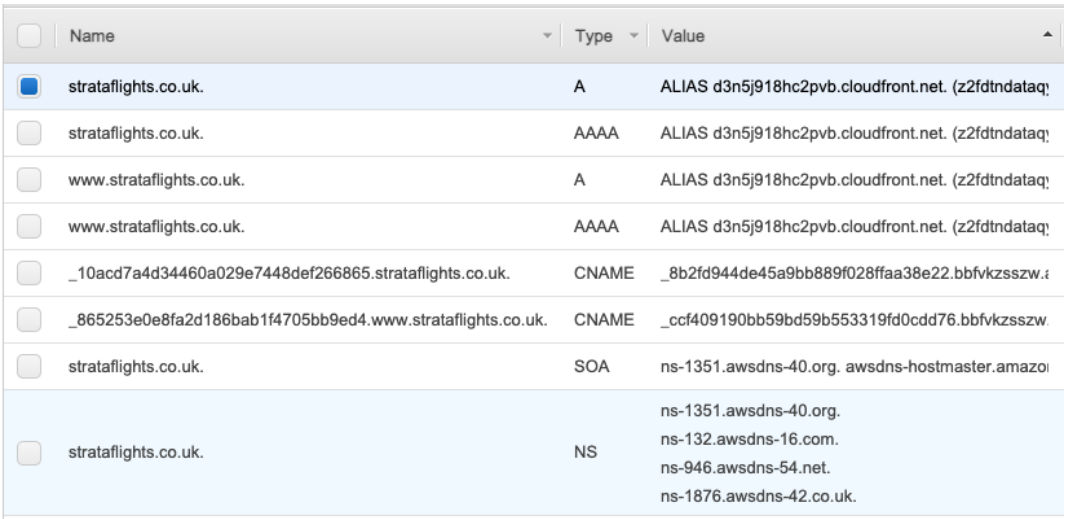


Figure 59: DNS configuration within Route 53

This concluded the manual deployment and initial configuration, resulting in the web application being accessible at `https://strataflights.co.uk` as well as `https://www.strataflights.co.uk`.

7.4 Continuous Deployment

As discussed in Section 5.3.3, CircleCI was the tool of choice for CI/CD. To start, the GitHub repository the web application's code was stored in was connected to CircleCI and subscribed to. This meant that any time code is committed to the master branch, the steps configured in the config.yml would be triggered and ran.

The config.yml is a configuration file, stored in the repository along with the code, that tells CircleCI what to do within the pipeline that is triggered. In this use case, using a docker image, a yarn install was executed to install all of the react app's dependencies, then a yarn build was executed in the same manner as discussed in the manual deployment, and then likewise, the contents of the build folder were then synced to the two S3 buckets. For CircleCI to gain access to AWS, an AWS access key and secret key were created in AWS and then added to CircleCI as environment variables, in order to abstract them from any code stored in the repository.

7.5 Conclusion

Deploying this web application properly with an actual domain has helped to give this solution real-world value as it can now be accessed by anyone over the internet. Using AWS services to host the application was seamless and a good alternative to hosting it the traditional way using a hosting company or on a server. Configuring the deployment in a CI/CD pipeline made it very easy to deploy new changes by just pushing code to the GitHub repository, and facilitated the desired agile approach to development as changes reached production more quickly, giving users more opportunities to test the application and give their feedback to be able to affect further new changes.

Chapter 8

Testing

8.1 Introduction

This section will detail the testing process undertaken in testing the application and its fulfilment of the proposed business requirements. This includes functional testing and user acceptance testing primarily, however, non-functional testing such as usability, performance, and security testing, was also carried out.

8.2 Functionality Testing

Testing was carried out manually after the end of each sprint, each consisting of a page in the user stories, to ensure the completed Jira issues met the outlined requirements they were created to cover. Testing the Lambda functions was carried out through the inbuilt testing facility in Amazon Lambda, where test cases can be defined and ran, as can be seen in Figure 60. It allows users to create a test event that mimics the API parameters would typically be passed to the Lambda function via an API Gateway endpoint.

Testing the API endpoints built in API Gateway also could've been realised with the inbuilt testing offered by Amazon, but it was instead opted to use the industry standard Postman. An example of its use can be seen in Figure 61, with the ISO 3166-2 code for Spain - ES, and a date in the format YYYY-MM-DD, passed as parameters to the API and the resulting response body being outputted, along with a response status of 200 [83], which is the HTTP response status code denoting the request has succeeded.

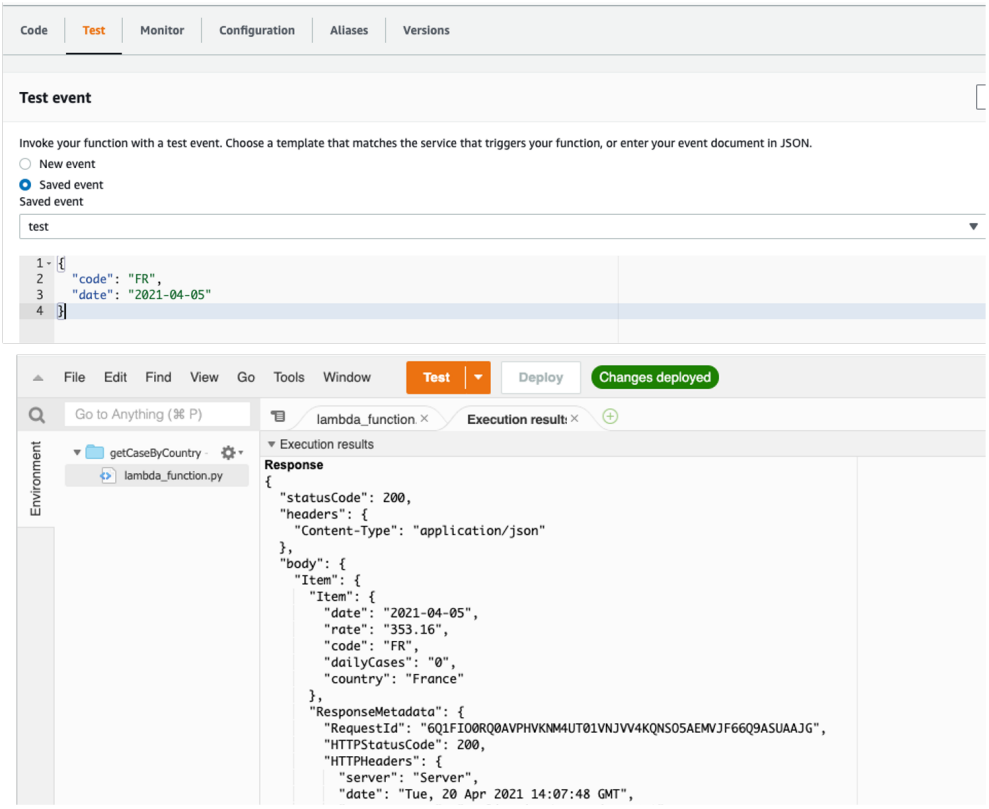


Figure 60: Screenshot of Amazon Lambda testing facility

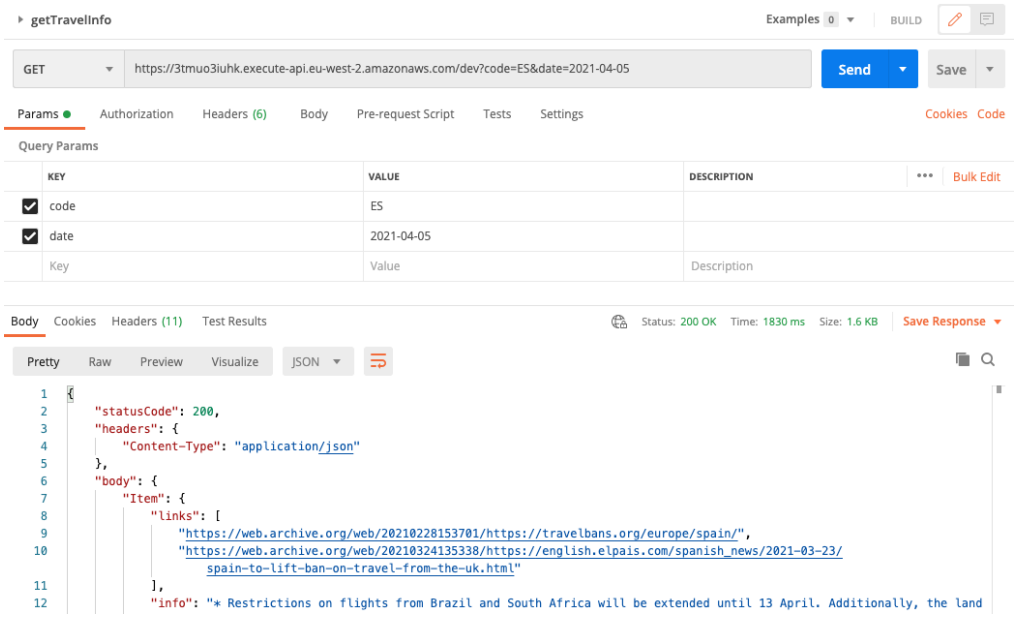


Figure 61: Screenshot of testing an API in Postman

Testing of the functionality, more relevant to the client-side, was done manually using the React development environment which is underpinned by Node.JS, allowing the React app to be hosted locally. With Chrome Dev Tools used to view the console logs when needed, testing was done in the Chrome browser to ensure all the functionality worked as intended. The results of this testing is detailed in the following tables organised in sections based on each page of the web application, with the exception of the COVID-19 Snapshot View and Update Sign Up View, which both exist across multiple pages of the web app.

Homepage

Element	Expected Result (i.e. User can...)	Status	Remarks
Homepage Flight Search	Search for flights to any destination (airport) in the world	Success	
Homepage Flight Search	Search directly from homepage	Success	
Homepage Flight Search	Can apply “trip type (round or one way)” filter to the flight search	Success	
Homepage Flight Search	Can apply “number of adults” filter to the flight search	Success	
Homepage Flight Search	Can apply “trip class (economy, business class, etc)” filter to the flight search	Success	

Search Results Page

Element	Expected Result (i.e. User can...)	Status	Remarks
Filters	Can apply “direct flights only” filter to the flight search	Success	
Filters	Can apply “under £500” filter to the flight search	Success	
Filters	Can apply “times” filter to the flight search	Success	
Filters	Can apply switch circle to switch around origin and destination	Success	
Filters	Can apply “cheapest, fastest, best” filter to the flight search	Success	
Search Results	Can view the itinerary (times, airline, duration, price) for each flight	Success	
Search Results	Can click on a flight and be taken to an external booking facility	Success	
COVID-19 Snapshot	See COVID-19 information for the destination flights have been searched to	Success	
COVID-19 Snapshot	See COVID-19 information alongside the search results	Success	

COVID-19 Snapshot View

Element	Expected Result (i.e. User can...)	Status	Remarks
Daily cases	See the current amount of COVID-19 cases for the country a destination is in for the last 24 hours	Success	
Weekly rate of cases	See the current weekly rate of COVID-19 cases for the country a destination is in	Success	
Travel advice/policy	See the current travel policy/advice for the country the destination is in	Success	
Updated dates	See when data was last updated	Success	
Data sources	See the source for the data provided	Success	
COVID-19 Snapshot modal	Can maximise the snapshot by clicking on it	Success	
COVID-19 Snapshot modal	Can see further information such as lockdown and quarantine in maximised snapshot	Success	

Update Sign Up View

Element	Expected Result (i.e. User can...)	Status	Remarks
Email Input	Can input their email	Success	Validation of email was also checked for
Destination Input	Can select a destination	Success	
Submit Button	Can submit the form	Success	
Email verification	Receive a request to verify their email	Success	Checked by viewing emails
Email verification	Be redirected to the success page upon successful email verification	Success	Also verified by email being added to mailing list in database
Email Updates	Receive weekly email on the COVID-19 situation for the destination signed up for	Success	Rate of emails sent was changed from weekly to daily to test this feature

Travel Advice Page

Element	Expected Result (i.e. User can...)	Status	Remarks
Map	View an interactive world map colour coded according to daily COVID-19 cases	Success	
Map	View an interactive world map colour coded according to the weekly rate COVID-19 cases per 100k people	Success	
Map Filter	Apply filter to highlight countries currently on lockdown	Success	
COVID-19 Snapshot	Click on a country on the map to get information on this country's COVID-19 situation	Success	

Travel Ideas Page

Element	Expected Result (i.e. User can...)	Status	Remarks
Recommendations	View recommendations of destinations that are not in red list countries	Success	
Recommendations	View prices of flights to recommended destinations	Success	
Red list countries	View a list of the current red list countries according to GOV.UK, accurate to the day of viewing	Success	
COVID-19 Snapshot	Click on the "view snapshot" button to get information on the recommended destination's COVID-19 situation	Success	

8.2.1 User Acceptance Testing

Since the web application had been deployed early during its development, it was tested by potential end-users throughout, and any feedback received was incorporated back into development as new features or bugs, logged in Jira.

Of the 27 set out requirements defined in the full requirements specification, see Appendix C, all of the Must and Should requirements were completed, 9 out of the 11 Could requirements were completed and none of the 4 Would requirements were completed. The completed requirements all passed their user acceptance testing based on upon the criteria in their descriptions.

8.3 Non-functional Testing

Within the Full Requirements Specification, there were also a few non-functional requirements defined, and these were also tested. Alongside this, other general non-functional aspects were tested such as usability and security. Table 5 displays the testing results of the non-functional requirements.

Table 5: A table summarising the profile of the focus group participants

Element	Expected Result (i.e. User can...)	Status	Remarks
COVID-19 Snapshot	Clearly see COVID-19 information visually standing out within search results	Success	
COVID-19 Snapshot	Clearly view COVID-19 information to help make an informed decision	Success	
Navigation	Navigate through the site efficiently using a Navbar	Success	
Navigation	Access the applications's main features (search page) in less than 3 clicks	Success	
Compatibility	Application is still effective on devices of smaller screen sizes such as smartphones and tablets	Success	Tested with Chrome DevTools & physical devices

8.3.1 Performance and Security Testing

Using tools such as KeyCDN Website Speed Test [84] the performance of the website was tested, and the necessary changes such as compressing images and font files were taken, as advised. For example some images were converted from PNG files to WebP as it offers 26% smaller file sizes than PNG, whilst still providing the same quality [85]. As a result of the smaller file size, WebP images load faster, improving the web application's overall performance, especially if there a lot of images used.

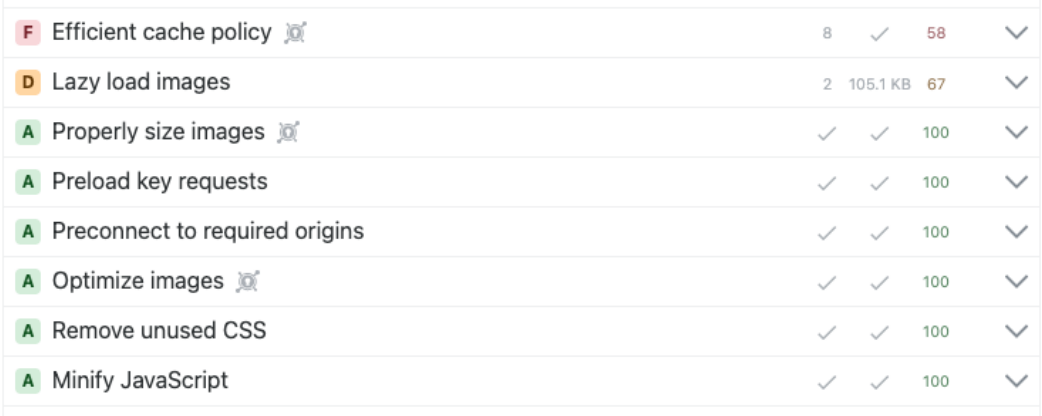


Figure 62: Screenshot of KeyCDN performance grades

The SSL certificate of the website was also tested to ensure all traffic incoming and outgoing was being encrypted. Using the Qualys SSL Server Test [86], a deep analysis of the configuration was ran and the results of it can be seen in Figure 63, where a grade A was achieved.

SSL Report: strataflights.co.uk

Assessed on: Tue, 20 Apr 2021 13:35:02 UTC | Hide | Clear cache

Scan Another >>

	Server	Test time	Grade
1	2600:9000:2146:7600:3:9587:ddc0:93a1 Ready	Tue, 20 Apr 2021 13:19:51 UTC Duration: 76.286 sec	A
2	2600:9000:2146:7200:3:9587:ddc0:93a1 Ready	Tue, 20 Apr 2021 13:21:07 UTC Duration: 74.924 sec	A
3	65.8.158.25 Ready	Tue, 20 Apr 2021 13:22:22 UTC Duration: 76.216 sec	A
4	65.8.158.108 Ready	Tue, 20 Apr 2021 13:23:39 UTC Duration: 76.430 sec	A
5	65.8.158.77 Ready	Tue, 20 Apr 2021 13:24:55 UTC Duration: 78.155 sec	A
6	65.8.158.129 Ready	Tue, 20 Apr 2021 13:26:13 UTC Duration: 77.68 sec	A
7	2600:9000:2146:8c00:3:9587:ddc0:93a1	Tue, 20 Apr 2021 13:27:30 UTC	A

Figure 63: Screenshot of Qualys SSL Server Test report

As mentioned in Section 6.4.4, efforts were made to ensure API keys weren't visible within the source code and network requests, and using Chrome Dev Tools this was confirmed, with network calls to third parties that use API keys made indirectly through API Gateway endpoints, ensuring the threat of cyber criminals gaining access to secrets such as API keys is minimised.

8.4 Conclusion

Through a cycle of continued testing and user feedback; facilitated by the agile approach of deploying the application early in the development process, requirements were completed effectively according to their priority, and as a result, all of the Must and Should requirements were completed and a large amount of the Could requirements were also satisfied. Non-functional testing was also completed, testing aspects such as performance, security and usability.

Chapter 9

Evaluation

9.1 Introduction

This section will look to analyse if and how well the project objectives have been met, and if the motivations for undertaking the project have been fulfilled. It will also look to do an analysis on how the project solution now compares to competitors discussed in the Gap Analysis. Towards the end of the section a reflection on what has been learnt throughout this project will be carried out.

9.2 Project Evaluation

Objective 1 - Carrying out a literature review in order to find a suitable methodology and suitable technologies to use for this project

A detailed literature review was carried out, as per Section 2.4, to determine technologies that would help facilitate the project. However, a decision was made to choose the methodology to be used for the project at a later stage after determining and analysing the requirements, to make a more informed decision on what methodology would best fit the project having then determined the requirements. The methodology was chosen in the Plan chapter in Section 5.2.

Objective 2 - Carry out a gap analysis researching the pros and cons of similar apps and systems in industry

A gap analysis was carried out, see Section 2.2, where similar works - in this case web applications used to search for flights, were analysed in depth based on a number of , which helped in sourcing requirements and user stories covering areas and gaps with other solutions, aimed to be addressed by this project.

Objective 3 - Source and collate accurate data regarding each country's COVID-19 situation to use within a web application

This objective was met using a number of different sources such as the World Health Organisation (WHO)'s COVID-19 Dashboard, the COVID19 API which uses COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University, alongside data scraped from the UK's official government website, for example. Data was sourced and cleaned, and then stored in databases to be used by the front end of the application via API Gateway endpoints.

Objective 4 - Develop a web application that will both serve holidaymakers and help them make better informed decisions on their travels in the pandemic

To achieve this objective, input and feedback from potential users alongside research into travel restrictions, and their influence by COVID-19, were taken aboard to build an application fulfilling the requirements that arose out of the aforementioned discussions and research. By following lean and agile principles and making use of CI/CD, it was quick to stand up and deploy the application and then continually improve it through a feedback loop with potential users, in order to complete requirements, with 21 of the 27 set out requirements being fulfilled, including 100% of the requirements prioritised as "Must" and "Should".

Objective 5 - Build a solution that is, or has the potential to be, commercially viable and have real-world value

This objective will be discussed in the section on Commercial Viability.

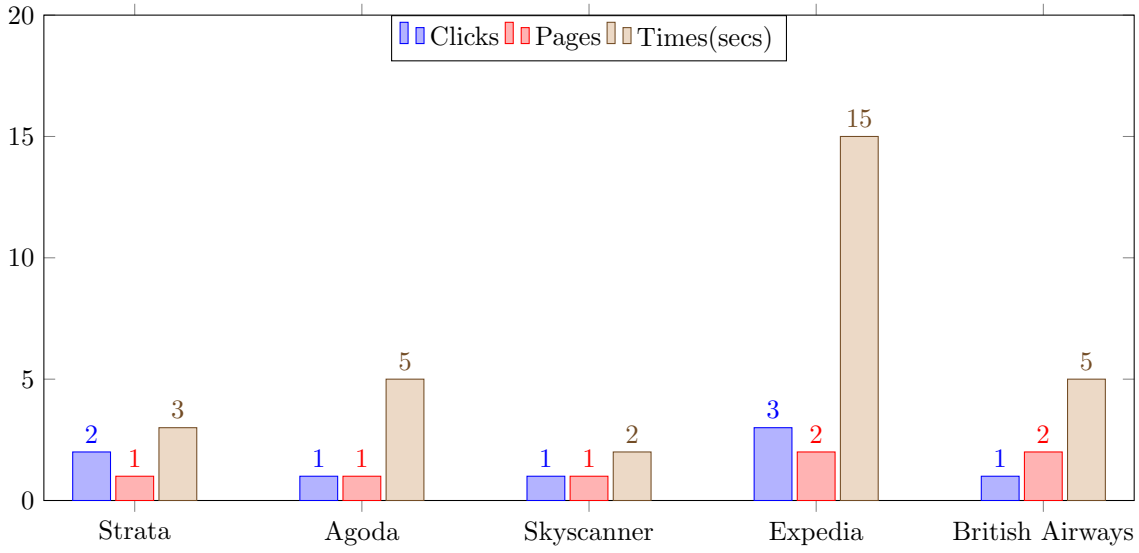
9.3 Competition Analysis

This project's solutions was compared with the existing solutions discussed in the Gap Analysis, using the same criteria upon which the existing solutions were assessed upon. Table 6 highlights how this project's solution, called Strata, has addressed some of the gaps found in its competitors, with it being the only solution to incorporate COVID-19 information within search results for instance. Figure 64 also highlights Strata being on a level playing ground with the other solutions in terms of the usability criteria: number of clicks, pages navigated, and time taken to access COVID-19 information.

Table 6: A comparison of this project's solution with existing solutions

Criterion	Strata	Agoda	Skyscanner	Expedia	British Airways
Gives clear information on COVID-19 restrictions for destinations within its website	✓	✗	✓	✗	✗
Factors travel restrictions into search results	✓	✗	✗	✗	✗

Figure 64: Comparison of the usability of this project's solution with existing solutions



9.4 Future Work and Improvements

This section will look to discuss any features this project could incorporate in the future, and will touch upon improvements that could help take the solution built in this project further.

9.4.1 Country Lockdown Prediction

This was one of the uncompleted requirements, prioritised as "Would". A feature not offered by any competitors, this feature would be one where the end users could stand to gain a lot, especially in the current circumstances where some countries in Europe and around the world are having second or third surges of COVID-19 cases. Users being able to see countries that could potentially have national lockdown restrictions in the near future, could help users avoid the countries in question when choosing their holiday destination.

Data already being collected as part of the application can be used for this purpose. As discussed in Section 6.4.3, data is deleted from tables after their Time to Live (TTL). These data items can be deleted from the DynamoDB tables but then pushed and archived in an Amazon S3 data lake, where machine learning and data mining can be performed on this collected historical data, to potentially find knowledge patterns and go on to build a model that can make predictions on countries entering a national lockdown imminently.

9.4.2 Testing within CI/CD

Although CI/CD was implemented in this project, it only tackled the build of the code and its deployment. A vital step of a standard CI/CD pipeline that was missed out was testing, and this was instead carried out manually which was time consuming. Adding unit tests as part of the CI/CD pipeline would save a great deal of time when it come to future deployments.

9.4.3 AWS CLI

Interaction with AWS services was done entirely through the AWS console which can be very manual and tedious when setting up services, especially when the configuration between different instances of services hardly vary. This project could've been improved by using AWS Command Line Interface (CLI) [87], which allows users to automate cloud infrastructure and interact with multiple services at once, in few commands. This means scripts could've been written on this project to do standard tasks that use the same configuration, such as creating a new IAM role for a lambda function, and in so doing also maintaining consistency between configuration of services, limiting any human error that can occur due to the large number of manual steps involved in using the AWS console.

9.4.4 Redux

Managing the state of the application without a library such as Redux proved very difficult, with a lot of callback functions used to pass state from child components to their parents, which can have an effect on the performance. It was decided not to use Redux due to the steep learning curve involved and time constraints. In hindsight, maybe taking extra time to learn the basics of Redux could've helped greatly with the management of state, especially in an application like the one created on this project where the state is passed from components regularly, i.e. state is passed from filter components to the search components to redefine the search query and return new search results.

9.5 Commercial Viability

One of the objectives of this project was to build a solution that actually had some real-world value, and that had potential of being commercially viable. One of the ways this was achieved was actually acquiring a domain and then deploying the solution to make it public and easily accessible. The next step was to join the Kiwi.com affiliate program [30], which can be used to monetise travel websites, giving a 3% commission on sales made by users proceeding to book flights they find on the web application. This fact means this solution can already begin to become profitable when also considering the very low cost it required to build, with the domain being acquired for free due to a deal from LCN [78], and costs for using AWS services currently at less than £7.50 a month due to being within the free limit of many of the services used.

9.6 Reflection on Learning

Doing this project has given me the opportunity to learn a great deal of things and has helped me utilise skills and knowledge gained through modules on my course, my industrial placement year and through other avenues such as hackathons. I will use this section to reflect on what I have learnt through undertaking this project.

Before beginning this project, I had previously worked with services from cloud technology providers such as Google Cloud Platform through the Team Projects and the Cloud Computing modules but as stated in my motivations for undertaking the project, I wanted to take my cloud computing skills to the next level beyond the scope of what I had learnt previously, and this was achieved by deciding to go ahead and opt for a serverless model, which led to me using a number of AWS services including some I was unfamiliar with such as Amazon SES and Amazon CloudWatch.

Another big learning curve for me was making use of Data Science offerings in Python such as Pandas and BeautifulSoup. I had long desired to do some web scraping and data mining, as also highlighted in

this project's motivations, and this project was a perfect way for me to learn new data science skills, tackling the sourcing of data through web scraping with BeautifulSoup, and manipulating data and doing feature selection on data with Pandas. Both were libraries I had no prior experience with, but have now gained a strong foundation in.

One area of the project which I particularly enjoyed was the design stage. Doing the Human Computer Interaction module in the first semester of my final year led perfectly into my Final Year Project, and equipped me with knowledge on how to conduct wireframe prototyping, and incorporate user feedback effectively into new iterations of designs. It also taught me general principles on human-centred design, and popular heuristics and guidelines such as Nielsen's usability heuristics [32] which I then utilised in my designs. This alongside some experience I gained using UI/UX design tools such as Figma through hackathons, led to a fruitful design stage and also to the prototyping-led approach to requirements elicitation this project followed.

In terms of general web development, the Programming for the WWW module gave me a good introduction to basic JavaScript and PHP, but it was through my industrial placement year I learnt and familiarised myself with more modern languages and frameworks used for creating web applications such as React.JS, and Python which was used for the backend. Having used both extensively on placement and extra-curricular events such as hackathons, it seemed logical to make use of these skills on this project and only improve on them which I definitely did.

Creating a public facing, cloud-based web application required me to consider security in more depth. Knowledge gained through both the Cyber Security and Cloud Computing modules, made me more aware of threats and vulnerabilities that can exist within internet-based applications, and guidelines, such as OWASP Serverless Top 10, that can be used to help prevent the vulnerabilities from being taken advantage of, such as in the manner of Distributed Denial of Service (DDOS) attacks and injection attacks. Learning this helped me define possible attack vectors and vulnerabilities within my application, and to code with security in mind, as per Section 6.4.4.

9.7 Conclusion

In concluding this chapter, it was pointed out how objectives were successfully met and motivations subsequently fulfilled. An analysis of how this project's solution compares to others solutions, initially discussed in the Gap Analysis, was also carried out. Towards the end, key areas where improvements could be made and future work implemented were discussed, alongside how far this project has gone in attempting to be commercially viable at present, or at least what steps have been taken for it to be commercially viable in the near future. Closing this chapter was a reflection on learning through the undertaking of the project.

A Low-fidelity Prototype Wireframes

Figure 65: Main Dashboard Lo-Fi Wireframe

Figure 66: Alerts Lo-Fi Wireframe

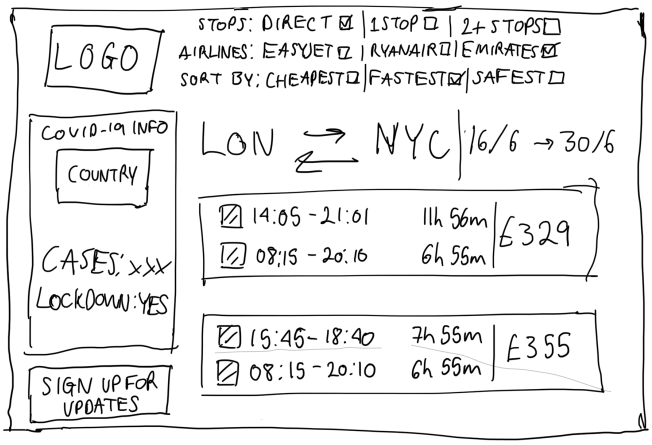


Figure 67: Search Results Lo-Fi Wireframe

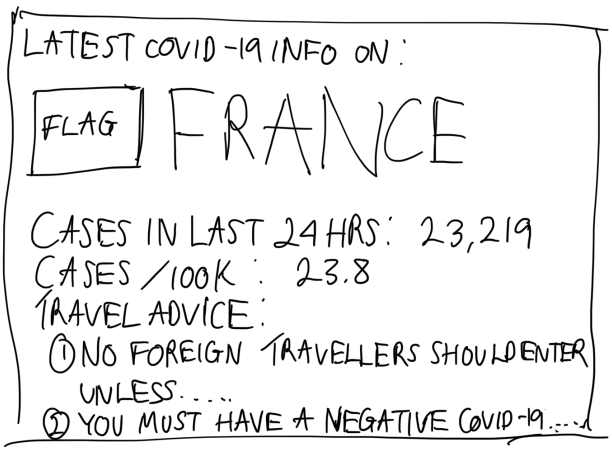


Figure 68: COVID-19 Snapshot Lo-Fi Wireframe

-

B Objective Performance Capture Sheet

Objective Performance Capture Sheet

Scenario Task and comments	Task Time (mm:ss)	No. of Issues	Task completed?
1. *"The UK government has just announced international travel will be allowed in a months time. You and 3 of your friends want to travel abroad as soon as this date comes but are not sure of what destinations allow UK travellers to visit them."*			
2. *"You are searching for flights to travel to New York for a music festival you want to go to in June. Whilst searching for flights, you realise you aren't aware of the current restrictions imposed in the United States due to COVID-19 and have no information on the number of COVID-19 cases there."*			
3. *"You are considering going to Paris in August but you're hesitant to book now since August is still a bit far away and you're worried the outlook of COVID-19 may change in France, meaning you won't be able to travel there and could lose out on the money paid for your ticket."*			

Other general feedback

Figure 69: Objective Performance Capture Sheet

C Full Requirements Specification

Table 7: Full Requirements Specification

Begin of Table		
Summary	Priority	Description
Homepage Flight Search	Must	As a user I want to be able to search for flights to any destination And I want to do this directly from the homepage So I don't have to navigate to the flights page to begin my search
Navigation	Must	As a user I want to be able to navigate through the site efficiently using a Navbar And I want to be able to access the applications's main features in less than 3 clicks So I can have a good user experience using the application
Flight Destinations	Must	As a user I should be able to search for flights for any destination (airport) in the world So I can choose anywhere to travel to
COVID-19 Information within Search	Must	As a user As a user I want to be able to see COVID-19 information for the destination I am searching for flights And I want to see this alongside the search results And I want the information to visually stand out within the search results So that it can be easily visible and so that I can clearly view COVID-19 information to help me an informed decision
COVID-19 cases in the last 24 hours	Must	As a user I want to see the current amount of COVID-19 cases for the country a destination is in for the last 24 hours And I want to see when this data was last updated So I can get a picture of the COVID-19 situation in the destination I want to travel to and see how recent this information is

Continuation of Table 7		
Summary	Priority	Description
COVID-19 weekly rate	Must	As a user I want to see the current weekly rate of COVID-19 cases for the country a destination is in And I want to see when this data was last updated So I can get a picture of the COVID-19 situation in the destination I want to travel to and see how recent this information is
Travel Advice	Must	As a user I want to see the current travel policy/advice for the country the destination I want to travel to is in And I want to see the source for this information And I want to see when this information was last updated So I can get a verifiable source of information of the travel policy for my destination to make an informed decision
Sources of data	Should	As a user I want to see the sources of any COVID-19 information So that I can verify myself that this information is correct if I wish to and have confidence the information is from reputable sources
Date and time of information	Should	As a user I want to see when COVID-19 information was last updated So that I can know how recent the information I am look at is
Lockdown and Quarantine Information	Should	As a user I want to be able to check if country my destination is in has lockdown or quarantine restrictions So that I can know whether my stay in that country will be hampered by things not being available or having to quarantine
Disclaimer	Should	As a user I want to be able to see a disclaimer regarding the COVID-19 information seen within the application So that I am aware that the data may not always be 100% accurate, and should be taken at face value
COVID-19 Updates	Should	As a user I want to be able to enter my email and preferred destination And receive weekly updates on the COVID-19 situation in that country So that I can keep track of any changes without having to go on the web application all the time

Continuation of Table 7		
Summary	Priority	Description
Link to flight booking	Could	As a user I want to be able to click on a flight search result And then be redirected to an external page where I could go ahead and book the flight if I wished to So I don't have to search for a link to book a flight I find manually
Update Sign up within search	Could	As a user I want to be able to search for flights to any destination And I then be able to sign up for updates, with this searched for destination pre-filled in the form So I don't have to re-enter the destination to sign up for updates and/or navigate back to the homepage to sign up for updates
Chart of COVID-19 cases for the last week	Could	As a user I want to be able to see a chart of new COVID-19 cases for each country for the last 7 days And be able to hover over each bar to see its value And this feature should be part of the expanded COVID-19 snapshot So I can see what the COVID-19 situation is for my desired destination on a weekly scale
Refresh on filter change	Could	As a user I want the search results to refresh any time any of the search information is changed So I don't have to keep pressing the search button any time I apply a filter or change information in the search query
Search Filters	Could	As a user I want to apply common filters to my flight search such as number of stops, preferred airlines and price ranges So I can refine my search to fit my travel need
Suitable for all devices	Could	As a user I want the application to be suitable for all device sizes So I can use it on any of my devices and on the go on my smartphone
Suggestions	Could	As a user I want to see suggested destinations based on COVID-19 advice for the UK So that if I don't know where exactly I want to go to I can get ideas of destinations I can go to
Colour Coding	Could	As a user I want to see countries colour coded on a map according to the relative daily COVID-19 cases So that I can easily compare how different countries are being affected by COVID-19

Continuation of Table 7		
Summary	Priority	Description
Filter Interactive Map	Could	As a user I want to be able to change the colour coding using other information as the deciding factor such as if a country is on lockdown or weekly rate of COVID-19 cases So that I can easily compare how different countries are being affected by COVID-19
Search Interactive Map	Could	As a user I want to be able to search for a country in an input box And for that country to be highlighted within the map So that I can easily locate my desired destination, even without knowing its location on the global map
Switch Locations	Could	As a user I want to be able to quickly switch around my departure and arrival destination So I can compare prices for each route the alternate way
Suggestions based on country of origin	Would	As a user I want to see suggested destinations based on COVID-19 advice for my country So that if I don't know where exactly I want to go to I can get ideas of destinations I can go to
Suggestions dates	Would	As a user I want to see suggested destinations based on COVID-19 advice for my country So that if I don't know where exactly I want to go to I can get ideas of destinations I can go to
Price Updates	Would	As a user I want to be able to enter my email and preferred destination And then receive updates if the lowest price of flights for my destination and dates changes So that I can keep track of any changes without having to go on the web application all the time
Lockdown Prediction	Would	As a user I want to see predictions on what countries could potentially go on lockdown in the next few months So that I can plan my holidays avoiding countries that are at risk of a potential lockdown
End of Table		

D Example of Lambda function inserting data into a DynamoDB table

```
# Instantiating connection objects with DynamoDB using boto3 dependency
dynamodb = boto3.resource('dynamodb')
client = boto3.client('dynamodb')
# Getting the table objects
countries = dynamodb.Table('country_covid_cases')
countries_codes = dynamodb.Table('country_codes')

try:
    with countries.batch_writer() as batch:
        for index, row in df.iterrows():
            chunk = {
                'date': datetime.now().strftime('%Y-%m-%d'),
                'code': getCode(row['Name'], countries_codes),
                'country': row['Name'],
                'dailyCases': str(row['Cases - newly reported in last 24 hours']),
                'rate': str(row['Cases - newly reported in last 7 days per 100000 population']),
                'ttl': Decimal(str(time.time() + 604800))
            }
            batch.put_item(Item=chunk)
    return {
        'statusCode': 200,
        'body': json.dumps('Successfully inserted global COVID-19 cases!')
    }
except Exception as e:
    print('Closing lambda function')
    print(e)
    exception_type, exception_object, exception_traceback = sys.exc_info()
    filename = exception_traceback.tb_frame.f_code.co_filename
    line_number = exception_traceback.tb_lineno
    print("Exception type: ", exception_type)
    print("File name: ", filename)
    print("Line number: ", line_number)
    return {
        'statusCode': 400,
        'body': json.dumps('Error saving the global COVID-19 cases')
    }

```

Listing 14: Inserting data into DynamoDB table

E Example of Lambda function retrieving data from a DynamoDB table

```
import json
import boto3
from datetime import datetime as dt
import datetime

def lambda_handler(event, context):
    dynamodb = boto3.resource('dynamodb')
    travel_info = dynamodb.Table('country_travel_advice')

    # querying DB table using parameters of API call (date and country code)
    response = travel_info.get_item(Key={
    'date': event['date'],
    'code': event['code']
    })

    try:
        # check if DB table has been updated today otherwise use yesterdays data
        if 'Item' not in response:
            response = travel_info.get_item(Key={
                'date': (dt.fromisoformat(event['date'])-datetime.timedelta(days=1)).
strftime("%Y-%m-%d"),
                'code': event['code']
            })
        return {
            'statusCode': 200,
            "headers": {
                "Content-Type": "application/json"
            },
            'body':  response
        }
    except Exception as e:
        print('Closing lambda function')
        print(e)
        return {
            'statusCode': 400,
            'body': json.dumps('Error retrieving the travel info')
        }

```

Listing 15: Retrieving data from a DynamoDB table

References

[1] Condor Ferries. (2020) Online travel booking statistics 2020. [Online]. Available: https://www.condorferries.co.uk/online-travel-booking-statistics [Accessed: Nov. 2, 2020]

[2] Agoda. (2021) Agoda official site. [Online]. Available: https://www.agoda.com/en-gb/ [Accessed: Nov. 15, 2020]

[3] Agoda. (2020) Update on covid-19 from agoda. [Online]. Available: https://www.agoda.com/en-gb/coronavirus [Accessed: Nov. 15, 2020]

[4] Skyscanner Ltd. (2020) We're the travel company who puts you first. [Online]. Available: https://www.skyscanner.net/about-us [Accessed: Nov. 15, 2020]

[5] Skyscanner Ltd. (2020, Nov.) Covid-19 travel info. [Online]. Available: https://www.skyscanner.net/travel-restrictions [Accessed: Nov. 15, 2020]

[6] Expedia. (2020) Fast facts. [Online]. Available: https://newsroom.expedia.com/fast-facts [Accessed: Dec. 28, 2020]

[7] IAG. (2021) British airways. [Online]. Available: https://www.iairgroup.com/en/our-brands/british-airways [Accessed: Jan. 13, 2021]

[8] British Airways. (2021, Jan.) Covid-19 travel and service updates. [Online]. Available: https://www.britishairways.com/en-gb/information/incident/coronavirus/latest-information [Accessed: Jan. 13, 2021]

[9] GOV.UK. (2020, Dec.) Full list of local restriction tiers by area. [Online]. Available: https://www.gov.uk/guidance/full-list-of-local-restriction-tiers-by-area#tier-4-stay-at-home [Accessed: Dec. 22, 2020]

[10] GOV.UK. (2020, Dec.) UK summary. [Online]. Available: https://coronavirus.data.gov.uk/ [Accessed: Dec. 21, 2020]

[11] RapidAPI. (2020, Nov.) Skyscanner flight search api documentation. [Online]. Available: https://rapidapi.com/skyscanner/api/skyscanner-flight-search/endpoints [Accessed: Dec. 28, 2020]

[12] Skyscanner Ltd. (2020) Flight search widget. [Online]. Available: https://www.partners.skyscanner.net/affiliates/widgets-documentation/flight-search-widget [Accessed: Dec. 28, 2020]

[13] Amadeus IT Group SA. (2020) Flight offers search. [Online]. Available: https://developers.amadeus.com/self-service/category/air/api-doc/flight-offers-search [Accessed: Dec. 28, 2020]

[14] RapidAPI. (2020, Jun.) Hotels api documentation. [Online]. Available: https://rapidapi.com/apidojo/api/hotels4 [Accessed: Dec. 28, 2020]

[15] Amadeus IT Group SA. (2021) Hotel search. [Online]. Available: https://developers.amadeus.com/self-service/category/hotel/api-doc/hotel-search [Accessed: Jan. 13, 2021]

[16] A. Sharma. (2020, Apr.) 5 popular python libraries to perform web scraping. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/5-popular-python-libraries-web-scraping/ [Accessed: Dec. 22, 2020]

[17] UiPath. (2020) Web scraping software that works everywhere. [Online]. Available: https://www.uipath.com/solutions/technology/web-scraping-software [Accessed: Dec. 23, 2020]

[18] Facebook Inc. (2020) React. [Online]. Available: https://reactjs.org/ [Accessed: Dec. 26, 2020]

[19] E. You. (2020, Dec.) Introduction. [Online]. Available: https://v3.vuejs.org/guide/introduction.html#what-is-vue-js [Accessed: Dec. 26, 2020]

[20] Google. (2020) Introduction to the angular docs. [Online]. Available: https://angular.io/docs [Accessed: Dec. 27, 2020]

[21] StrongLoop and IBM. (2020) Express 4.17.1. [Online]. Available: https://expressjs.com/ [Accessed: Dec. 27, 2020]

[22] M. Makai. (2020) Flask. [Online]. Available: https://www.fullstackpython.com/flask.html [Accessed: Dec. 27, 2020]

[23] Amazon Web Services. (2020) AWS lambda. [Online]. Available: https://aws.amazon.com/lambda/?c=ser&sec=srv [Accessed: Dec. 27, 2020]

[24] S. Tanvir, M. Safdar, H. Tufail, and U. Qamar, "Merging prototyping with agile software development methodology," in *International Conference on Engineering, Computing & Information Technology (ICECIT 2017)*, Kuala Lumpur, Malaysia, Oct. 2017, pp. 50–54.

[25] H. Sharp, J. Preece, and Y. Rogers, *Interaction Design : Beyond Human-Computer Interaction*. John Wiley & Sons, Incorporated, 2019.

[26] M. Mannio and U. Nikula, "Requirements elicitation using a combination of prototypes and scenarios." in *Anais do WER01 - Workshop em Engenharia de Requisitos*, Buenos Aires, Argentina, Jan. 2001, pp. 283–296.

[27] *Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts*, International Organization for Standardization Std. ISO 9241-11:2018, Mar. 2018. [Online]. Available: https://www.iso.org/standard/63500.html

[28] Product Plan. (2021) Moscow prioritization. [Online]. Available: https://www.productplan.com/glossary/moscow-prioritization/ [Accessed: May. 1, 2021]

[29] L. A. Maciaszek, *Requirements analysis and system design*. Pearson Addison Wesley, Jun. 2007.

[30] Kiwi. (2021) Affiliates. [Online]. Available: https://partners.kiwi.com/industries/affiliates/ [Accessed: Apr. 23, 2021]

[31] H. Gomaa and E. Olimpiew, "The role of use cases in requirements and analysis modeling," Ph.D. dissertation, Dept. Info. and Soft. Eng., George Mason Univ., Fairfax, VA, Jan. 2005.

[32] J. Nielsen. (2020, Nov.) 10 usability heuristics for user interface design. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/ [Accessed: Apr. 6, 2021]

[33] Marianne. (2021, Apr.) 9 principles of good web design. [Online]. Available: https://www.feelingpeaky.com/9-principles-of-good-web-design/ [Accessed: Apr. 1, 2021]

[34] K. Cherry. (2020, Feb.) The color psychology of blue. [Online]. Available: https://www.verywellmind.com/the-color-psychology-of-blue-2795815#:~:text=The%20Psychology%20of%20Blue,-According%20to%20color&text=Blue%20calls%20to%20mind%20feelings,sign%20of%20stability%20and%20reliability. [Accessed: Apr. 7, 2021]

[35] Google. (2021) Nunito. [Online]. Available: https://fonts.google.com/specimen/Nunito#standard-styles [Accessed: Apr. 7, 2021]

[36] J. Koi-Akrofi, A. H. Matey, and G. Koi-Akrofi, "Understanding the characteristics, benefits and challenges of agile it project management: A literature based perspective," *International Journal of Software Engineering & Applications*, vol. 10, pp. 25–44, Sep. 2019.

[37] Microsoft. (2021) Documentation for visual studio code. [Online]. Available: https://code.visualstudio.com/docs [Accessed: Apr. 9, 2021]

[38] J. Mann. (2021) A brief introduction to version control. [Online]. Available: https://docs.gearset.com/en/articles/2287242-a-brief-introduction-to-version-control [Accessed: Apr. 11, 2021]

[39] Sacolick. (2020, Jan.) What is CI/CD? continuous integration and continuous delivery explained. [Online]. Available: https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html [Accessed: Apr. 11, 2021]

[40] Google. (2016, Mar.) Overview. [Online]. Available: https://developer.chrome.com/docs/devtools/overview/ [Accessed: Apr. 11, 2021]

[41] Postman Inc. (2021) Api testing with postman. [Online]. Available: https://www.postman.com/api-platform/api-testing/ [Accessed: Apr. 11, 2021]

[42] Project Jupyter. (2021) Jupyter. [Online]. Available: https://jupyter.org/ [Accessed: Apr. 11, 2021]

[43] Tailwind CSS. (2021) Rapidly build modern websites without ever leaving your html. [Online]. Available: https://tailwindcss.com/ [Accessed: Apr. 12, 2021]

[44] Axios. (2021) Getting started. [Online]. Available: https://axios-http.com/docs/intro/ [Accessed: Apr. 12, 2021]

[45] Mozilla. (2021) XMLHttpRequest. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest [Accessed: Apr. 12, 2021]

[46] Z Creative Labs. (2021) React simple maps. [Online]. Available: https://www.react-simple-maps.io/ [Accessed: Apr. 29, 2021]

[47] R. Bolovan. (2021) Python, boto3, and AWS S3: Demystified. [Online]. Available: https://realpython.com/python-boto3-aws-s3/ [Accessed: Apr. 12, 2021]

[48] Docker Inc. (2021) Docker overview. [Online]. Available: https://docs.docker.com/get-started/overview/ [Accessed: Apr. 27, 2021]

[49] Amazon Web Services. (2021) Amazon api gateway. [Online]. Available: https://aws.amazon.com/api-gateway/ [Accessed: Apr. 12, 2021]

[50] Amazon Web Services. (2021) Amazon dynamodb. [Online]. Available: https://aws.amazon.com/dynamodb/ [Accessed: Apr. 12, 2021]

[51] Amazon Web Services. (2021) What is amazon cloudwatch events? [Online]. Available: https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/WhatIsCloudWatchEvents.html [Accessed: Apr. 12, 2021]

[52] Amazon Web Services. (2021) Amazon simple email service. [Online]. Available: https://aws.amazon.com/ses/ [Accessed: Apr. 12, 2021]

[53] Amazon Web Services. (2021) AWS Identity and Access Management (IAM). [Online]. Available: https://aws.amazon.com/iam/ [Accessed: Apr. 25, 2021]

[54] World Health Organization (WHO). (2021) WHO Coronavirus (COVID-19) Dashboard. [Online]. Available: https://covid19.who.int/table [Accessed: Apr. 13, 2021]

[55] *Codes for the representation of names of countries and their subdivisions – Part 2: Country subdivision code*, International Organization for Standardization Std. ISO 3166-2:2020, Aug. 2020. [Online]. Available: https://www.iso.org/standard/72483.html

[56] Amazon Web Services. (2021) Schedule expressions for rules. [Online]. Available: https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/ScheduledEvents.html [Accessed: Apr. 13, 2021]

[57] K. Redelinghuys. (2021) A free api for data on the coronavirus. [Online]. Available: https://covid19api.com/ [Accessed: Apr. 13, 2021]

[58] Johns Hopkins University. (2021) Covid-19 data repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University. [Online]. Available: https://github.com/CSSEGISandData/COVID-19 [Accessed: Apr. 13, 2021]

[59] Wikipedia. (2021) Table of pandemic lockdowns. [Online]. Available: https://en.wikipedia.org/wiki/COVID-19_lockdowns#Table_of_pandemic_lockdowns [Accessed: Apr. 14, 2021]

[60] Pandas Development Team. (2021) Html table parsing gotchas. [Online]. Available: https://pandas.pydata.org/docs/user_guide/io.html#io-html-gotchas [Accessed: Apr. 14, 2021]

[61] Pandas Development Team. (2021) pandas.read_html. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.read_html.html#pandas-read-html [Accessed: Apr. 14, 2021]

[62] GOV.UK. (2021) Coronavirus (COVID-19): red list travel ban countries. [Online]. Available: https://www.gov.uk/guidance/transport-measures-to-protect-the-uk-from-variant-strains-of-covid-19 [Accessed: May. 2, 2021]

[63] O. Khan. (2020, Apr.) Treact - free react & tailwindcss based ui kit for building landing pages. [Online]. Available: https://owaiskhan.me/post/free-tailwindcss-react-ui-kit [Accessed: Apr. 17, 2021]

[64] Facebook Inc. (2021) Using the effect hook. [Online]. Available: https://reactjs.org/docs/hooks-effect.html [Accessed: Apr. 17, 2021]

[65] Mozilla. (2021) ¡input type="email"¿. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/email#validation [Accessed: Apr. 19, 2021]

[66] Z Creative Labs. (2021) world-110m.json. [Online]. Available: https://raw.githubusercontent.com/zcreativelabs/react-simple-maps/master/topojson-maps/world-110m.json [Accessed: Apr. 29, 2021]

[67] Python Software Foundation. (2021) Find, install and publish python packages with the python package index. [Online]. Available: https://pypi.org/ [Accessed: Apr. 14, 2021]

[68] Amazon Web Services. (2021) Lambda layers. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/configuration-layers.html [Accessed: Apr. 14, 2021]

[69] Lambci. (2021) lambci/lambda. [Online]. Available: https://hub.docker.com/r/lambci/lambda [Accessed: Apr. 14, 2021]

[70] M. Mascioni. (2020, Dec.) Using external python packages with AWS lambda layers. [Online]. Available: https://dev.to/mmascioni/using-external-python-packages-with-aws-lambda-layers-526o [Accessed: Apr. 14, 2021]

[71] T. Becker. (2018, Mar.) A production-grade CI/CD pipeline for serverless applications. [Online]. Available: https://medium.com/@tarekbecker/a-production-grade-ci-cd-pipeline-for-serverless-applications-888668bcfe04 [Accessed: Apr. 15, 2021]

[72] Mozilla. (2021) Cross-origin resource sharing (CORS). [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS [Accessed: Apr. 16, 2021]

[73] J. Olson, "Data quality and database archiving: The intersection of two important data management functions," in *The Fifth MIT Information Quality Industry Symposium*, Cambridge, MA, Jul. 2011, pp. 57–74.

[74] A. Wagner and Amazon Web Services. (2017, Jun.) Automatically archive items to s3 using dynamodb time to live (TTL) with AWS Lambda and Amazon Kinesis Firehose. [Online]. Available: https://aws.amazon.com/blogs/database/automatically-archive-items-to-s3-using-dynamodb-time-to-live-with-aws-lambda-and-amazon-kinesis-firehose/ [Accessed: Apr. 13, 2021]

[75] Epoch Converter. (2021) Epoch & unix timestamp conversion tools. [Online]. Available: https://www.epochconverter.com/ [Accessed: Apr. 13, 2021]

[76] U. Kamil. (2020, Mar.) How to take care of the security of serverless applications in AWS? [Online]. Available: https://ttpsc.com/en/blog/how-to-take-care-of-the-security-of-serverless-applications-in-aws/ [Accessed: Apr. 16, 2021]

[77] Open Web Application Security Project. (2017) OWASP top 10 (2017) interpretation for serverless. [Online]. Available: https://github.com/OWASP/Serverless-Top-10-Project/raw/master/OWASP-Top-10-Serverless-Interpretation-en.pdf [Accessed: Apr. 16, 2021]

[78] LCN.com. (2021) Domains, hosting, cloud servers, and ssl - lcn.com. [Online]. Available: https://www.lcn.com/ [Accessed: Apr. 23, 2021]

[79] Amazon Web Services. (2021) Amazon s3. [Online]. Available: https://aws.amazon.com/s3/ [Accessed: Apr. 25, 2021]

[80] Amazon Web Services. (2021) Amazon cloudfront. [Online]. Available: https://aws.amazon.com/cloudfront/ [Accessed: Apr. 25, 2021]

[81] Amazon Web Services. (2021) Amazon route 53. [Online]. Available: https://aws.amazon.com/route53/ [Accessed: Apr. 25, 2021]

[82] Amazon Web Services. (2021) AWS certficate manager. [Online]. Available: https://aws.amazon.com/certificate-manager/ [Accessed: Apr. 25, 2021]

[83] Mozilla. (2021) 200 ok. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200 [Accessed: Apr. 28, 2021]

[84] KeyCDN Tools. (2021) Website speed test. [Online]. Available: https://tools.keycdn.com/speed [Accessed: Apr. 29, 2021]

[85] A. Żółciak. (2018, Sep.) Why webp is the rockstar of image formats for web designers. [Online]. Available: https://insanelab.com/blog/web-development/webp-web-design-vs-jpeg-gif-png/#:~:text=WebP%20vs.&text=PNG%20supports%20transparency%2C%20which%20is%20a%20critical%20element%20of%20web%20design.&text=Essentially%20WebP%20offers%20the%20following,file%20size)%20than%20PNG%20images. [Accessed: May. 2, 2021]

[86] Qualys Inc. (2021) SSL server test. [Online]. Available: https://www.ssllabs.com/ssltest/ [Accessed: Apr. 29, 2021]

[87] Amazon Web Services. (2021) AWS command line interface. [Online]. Available: https://aws.amazon.com/cli/ [Accessed: Apr. 22, 2021]