



Bachelor (Hons) of Science in Computer Science

Honours Project

May 2020

**Increasing Usability of Flight Trackers using Augmented Reality**

|  |  |
| --- | --- |
| *Author*  Lee Patrick Robbie | *Supervisor*  Dr John P Isaacs |

**Abstract**

< To be written >

**Acknowledgements**

I would like to thank Dr John Isaacs, Head of the School of Computing Science and Digital Media at Robert Gordon University, for his dedication in providing guidance throughout the year and for his guidance throughout my time at the university. I would also like to take the time to thank and highlight the extreme professionalism of the university staff in dealing with a difficult transition to remote teaching with continued support during the Coronavirus pandemic. I would also like to thank Mark Scott-Kiddie, Fatima El-Ghanduri and Calum McGuire for their support and motivation throughout the years at university. Also, I’d like to thank all the participants who participated in any user testing that was conducted as the feedback was a crucial part of my project. Finally, of course, I would like to thank my family for their continued support throughout my time at university, especially emphasising the support received this year above all.

**Declaration**

I confirm that the work contained in this Honours project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

|  |  |
| --- | --- |
| A close up of an animal  Description automatically generated  Signed: | Date: 18/05/2020 |

Contents

1.8 Figures

Figure 1: How ADS-B works (Richards et al., 2010)

Figure 2: How MLAT works (“Multilateration (MLAT) - FlightAware,” 2019)

Figure 3: 27 OpenSky Network sensors 2017 (Strohmeier et al., 2015)

Figure 4: OpenSky world-wide coverage (“OpenSky’s reception coverage (July 2017) based on crowdsourced ADS-B... | Download Scientific Diagram,” 2019)

Figure 5: FlightAware ADS-B/MLAT coverage (“FlightAware ADS-B Coverage Map - FlightAware,” 2019)

Figure 6: FlightAware satellite coverage (“FlightAware ADS-B Coverage Map - FlightAware,” 2019)

Figure 7: Marker based AR (“example of marker-based AR | Download Scientific Diagram,” 2019)

Figure 8: Location based AR (Vakoms, 2019)

Figure 9: Uber colour palette (“31 Inspirational Brand Colors And How To Use Them | Piktochart Blog | Piktochart,” 2019)

Figure 10: Heetch logo (“Heetch raises $12 million to reboot its ridesharing service | Tech | Tech companies, Tech, Company logo,” 2019)

1 Introduction

2 Literature Review

This chapter aims to research and review the literature surrounding current flight tracking technologies available, flight tracking application programming interfaces using relevant data sources, how augmented reality is used in mobile applications and aids in user experience, and techniques used to assist in the usability of mobile applications.

* 1. Background

Multiple flight tracking applications are available for download from the App Store and Google Play, such as FlightAware, Plane Finder, Planes Live, and Flight Tracker. The most profound flight tracking application on the market is Flightradar24, being ranked number one on the App Store in over 130 countries and the number one travel app on the App Store in over 150 countries. The application boasts over 40 million downloads with one of the largest ADS-B networks in the world, with over 20,000 receivers. The application tracks 180,000 flights per day with an active user base of over 2 million users per day. This user base also contains major airlines using the application with other well know names within the industry such as Airbus, Boeing, and Embraer. (“About us - Find out about the world’s most popular flight tracker | Flightradar24,” 2019)

The application comes in three forms, standard (free) and premium (Silver/Gold). The standard version enables users to view aircraft in real-time, view departure and arrival times, and search aircraft by flight number, airport, or airline. Additionally, it has content specially targeted towards aviation enthusiasts such as photos of the aircraft being tracked, historical flight data, and a 3D pilot view. The premium version of the application, Silver charged at $1.50 a month and Gold charged at $4 a month, provides users with increased flight history, live weather overlays, aeronautical charts, and other additional features. The application is also supported on Apple Watch and Android Wear for an alternative device choice for the user. (Hill, 2018)

Flightradar24 initially started as a hobby in 2006, where a network of ADS-B receivers was built over Northern and Central Europe. However, it was not until 2009, where the network was opened to the public and allowed anyone with an ADS-B receiver to upload data to the network. The network began to cover the globe quickly; however, complete coverage is ongoing. (“About us - Find out about the world’s most popular flight tracker | Flightradar24,” 2019) Other flight tracking applications started in a similar way, such as Flightradar24’s main competitor, FlightAware. Both networks are competing to expand their network with the most accurate data sources available. The networks both rely on users to expand their network compared to waiting for governments to create the infrastructure to support technologies like ADS-B.

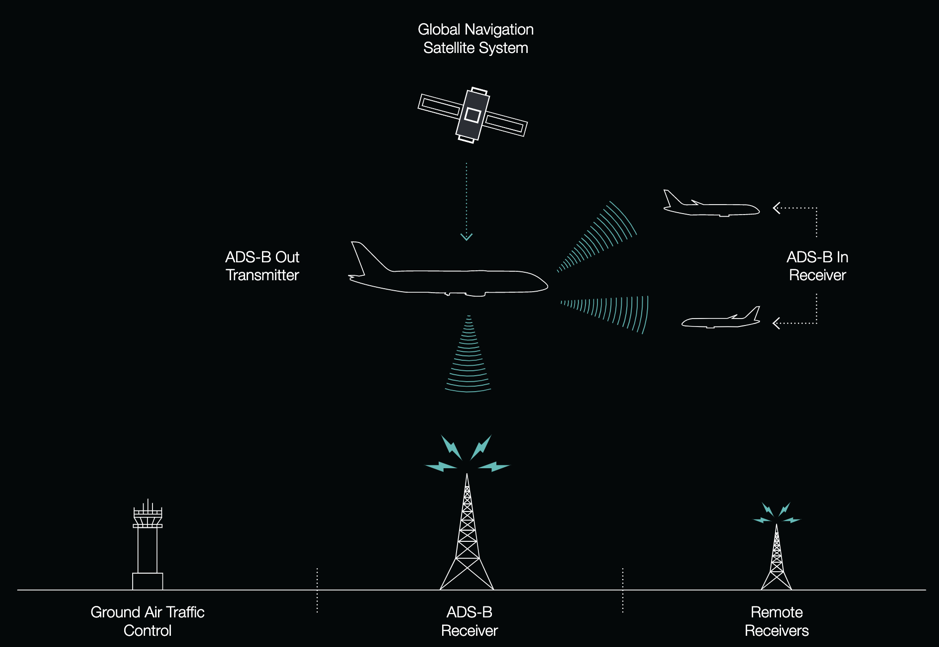
* 1. Flight Tracking Technology

The technology behind flight tracking comes from combining multiple data sources such as ADS-B and MLAT. This data is combined with aircraft schedules and statuses of flights which are acquired from airlines and airports. Newer aircraft such as all Airbus models, Boeing models between 737-787, are equipped with Automatic Dependent Surveillance-Broadcast (ADS-B) transceiver whereby it transmits signals containing data about the flight such as location and altitude. (“How flight tracking works - Learn how we track flights | Flightradar24,” 2019) The data is transmitted at a frequency of 1090 MHz with a transmitting pulse length of 120 μs allowing for data to be received by anyone/network with the appropriate ADS-B receivers. (Huang, Narayanan, & Feinberg, 2008) Older aircraft which are not equipped with the newer ADS-B transceivers can be located by calculating their position using Multilateration (MLAT). MLAT uses a method called the Time Difference of Arrival, which measures the time a signal is received from an aircraft using an older transponder, the Mode S, whereby the position can then be calculated. (“How flight tracking works - Learn how we track flights | Flightradar24,” 2019) The literature will convey the benefits and disadvantages of the data receivers to convey whether ADS-B is better suited to retrieve flight data compared to MLAT or if flight data is better retrieved using a combination of both ADS-B and MLAT.

2.2.1 Automatic Dependent Surveillance-Broadcast (ADS-B)

Air Traffic Management systems will face considerable challenges over the coming decades due to rapid growth in air traffic and demand. America alone expected in 2015 for air traffic to increase by 25-30% and in some cases, exceed that prediction. (Huang et al., 2008) Modernisation of flight tracking has been gradual with places such as western China beginning to consider ADS-B tracking due to restrictions of terrain and meteorological conditions preventing construction of new radar stations. (Zhang, Liu, & Zhu, 2011) Developed countries/continents such as Europe, United States, Australia, and Canada are beginning to enforce/drive ADS-B as a mandatory requirement on aircraft by 2020 (“How flight tracking works - Learn how we track flights | Flightradar24,” 2019) within their respective airspace. Enforcement of ADS-B tracking pushes applications to favourite ADS-B data sources for future data retrieval of flight tracking applications due to the drive to expand the network world-wide. (Davidson, 2019) ADS-B is a composition of CNS/ATM (Communication, Navigation, and Surveillance/Air Traffic Management) using Mode S transponders capable of ADS-B transmissions. ADS-B is the recommended surveillance method by the ICAO (International Civil Aviation Organisation) for the future generation of ATM.

With the use of ADS-B, air traffic control will change from a radar-based system into a satellite-derived location system. The change will increase safety as aircraft will no longer rely solely on ATC as aircraft will have surveillance of other aircraft. (Part III Department of Transportation Federal Aviation Administration 14 CFR Part 91 Automatic Dependent Surveillance-Broadcast (ADS-B) Out Performance Requirements To Support Air Traffic Control (ATC) Service; Final Rule mstockstill on DSKH9S0YB1PROD wit, 2010) This surveillance will significantly improve a pilot’s situational awareness of the traffic environment due to data of location and bearings being transmitted by aircraft in close proximity, as shown in Figure 1. (Huang et al., 2008) Additionally, this change brings enhanced accuracy and speed of data beneficial for precise flight positioning for tracking applications. (Part III Department of Transportation Federal Aviation Administration 14 CFR Part 91 Automatic Dependent Surveillance-Broadcast (ADS-B) Out Performance Requirements To Support Air Traffic Control (ATC) Service; Final Rule mstockstill on DSKH9S0YB1PROD wit, 2010)



*Figure 1: How ADS-B works* (Richards, O’Brien, & Miller, 2010)

Although many clear advantages of ADS-B for airlines related to safety and fuel efficiency, from more direct routings, (Richards et al., 2010) airlines such as JetBlue Airways are challenging to persuade due to historical tendencies to not invest in technologies unless a compelling safety and business case are given. (Moorman, 2016) However, due to the lesser cost of older technologies and the reliability of current global infrastructure, ADS-B in comparison, lacks the overall world-wide coverage its counterparts have established. (Zhang et al., 2011) Hence airlines are discussing whether to outfit their aircraft with newer ADS-B enabled Mode S transponders compared to the traditional Mode A, B, S transponders rendering some aircraft hidden to ADS-B tracking. Flight tracking applications require accurate data for users, and with limitations of reduced aircraft activity shows an obstacle for app developers. The problem shows that the data source is crucial for the most accurate data.

2.2.2 Multilateration (MLAT)

The International Civil Aviation Organisation (ICAO) in the early 1990s approved the model of the Future Air Navigation System (FANS) to be based on satellite and data link technology, later this would be known as CNS/ATM. As traditional air traffic control surveillance had limitations that would have constrained future air traffic growth. The solution was to upgrade to newer technologies such as ADS-B, as already discussed, SSR and MLAT. (Xu, He, Tang, & Li, 2015)

Aircraft that do not broadcast their latitude and longitude through ADS-B transponders such as older Boeing models (737-200) Bombardier CRJ/Dash models, Embraer models, Fokker 50, most helicopters and propeller aircraft (“How flight tracking works - Learn how we track flights | Flightradar24,” 2019) use another tracking technology called Multilateration. MLAT uses 1090 MHz signals broadcasted by Mode A, B, or S transponders to determine the aircraft’s location from locating the source of the transmission (Xu et al., 2015) using a method called the time difference of arrival (TDOA).

This method involves using four or more receivers/ground stations to detect aircraft by taking the time for a signal to be received by one receiver at a stationary point and the time taken for the signal to be received by at least three other different receivers. (“Multilateration (MLAT) - FlightAware,” 2019) As the data is transmitted, the position of the aircraft will be at different distances to each receiver. Therefore, the data will be received at marginally different times. The different times at which the transmissions are received can be used to determine the aircraft’s position accurately. (Xu et al., 2015) The data is transmitted to a server to be combined to calculate the latitude and longitude, as shown in Figure 2. The signals also broadcast the aircraft’s transponder identification and the altitude. Real-time flight tracking can be provided by collaborating the data. However, aircraft must be within line-of-sight with the receivers for an accurate position to be determined. (“Multilateration (MLAT) - FlightAware,” 2019) Although considered real-time, calculation delays and processing latency hinder true real-time flight activity with a 4-6 second delay. (“Multilateration (MLAT) - FlightAware,” 2019) MLAT coverage is limited to areas with receivers/ground stations present and normally only achieved at altitudes between 3000-10000 feet. Due to this limitation, general aircraft flying below the range may be hidden to MLAT surveillance. (“How flight tracking works - Learn how we track flights | Flightradar24,” 2019)

A picture containing sky, green

Description automatically generated

*Figure 2: How MLAT works* (“Multilateration (MLAT) - FlightAware,” 2019)

Another disadvantage to MLAT can be the positioning accuracy as it can be significantly affected by the factor of Geometric Dilution of Precision (GDOP). This can be due to a large surveillance area and positioning of fixed ground stations/receivers. As the distance increases from the aircraft to the polygon created by the multiple receivers, the larger the GDOP and consequentially, the less accurate the positioning of the aircraft. (Xu et al., 2015)

As current airspace contains a variety of old and new aircraft, successful tracking applications must be open to multiple tracking technologies to give users the full airspace traffic. Users limited to one dimension of traffic would not be drawn to an application with reduced availability. The primary audience of flight trackers being plane enthusiasts diverges into different categories of interest, such as airplanes (single-engine/multi-engine), rotorcraft (helicopter/gyroplane), and gliders. The different audiences should be taken into account with the choice of the data source to maximise user interest in an application.

2.2.3 ADS-B and MLAT Comparison

All flight tracking technologies available have different advantages and disadvantages against their tracking methods. ADS-B uses dependent surveillance based on GPS, making position accuracy greater than MLAT TDOA tracking techniques. MLAT, however, has proven to be less costly than older methods such as radar by 20-35% (Multilateration era Corporation, 2019) and currently outfitted by more aircraft world-wide compared to ADS-B. However, with the current push for ADS-B tracking to be the world-wide choice of flight tracking, the current weight will shift to ADS-B being more prominent compared to MLAT. (“How flight tracking works - Learn how we track flights | Flightradar24,” 2019)

As both technologies use Mode S transponders, they share a high update rate. In the case of ADS-B, an update rate of two positions per second (Neufeldt, 2017) allowing for precise and accurate data for flight tracking applications. The data for both ADS-B and MLAT are combined at central servers, which are used to retrieve flight data. Although ADS-B only requires one single ground station/receiver for coverage and MLAT requires multiple, the cost of ground equipment and infrastructure are relatively equal, with both having an overall low lifecycle cost. (Neufeldt, 2017)

Current security concerns only relate to ADS-B. Such security concerns are ADS-B spoofing and ADS-B meaconing. ADS-B signals are not authenticated or encrypted, so spoofing can occur where falsified data is inserted into the ADS-B system by transmitting a signal on ADS-B frequencies. The FAA claims, “We have ways of validating the data that shows up on a controller’s screen so that spoofed targets are filtered out … An FAA ADS-B security action plan identified and mitigated risks and monitors the progress of corrective action. These risks are security sensitive and are not publicly available”. This validation technique used by ATC is not publicly available, so flight tracking applications are at risk from ADS-B spoofing as they have no method to filter false data. (Thurber, 2012) The process of ADS-B meaconing involves the capturing of transmissions whereby are altered and retransmitted after a short delay. (Gouripeddi, 2016) The attack is primarily a man-in-the-middle attack, which is another way inaccurate data is accepted by flight tracking applications.

To ensure accurate data for users of flight tracking applications, a combination of old and new, MLAT and ADS-B, tracking technologies should be used to give a broader range of aircraft traffic, so limitations are not preventing the application's user base.

2.3 Flight Tracking Application Programming Interface

An Application Programming Interface (API) is an intermediary software which allows two applications to communicate. An application will connect to the internet sending data to a server. The data is received by the server and interpreted where the server will execute the required action(s) and transmit data back to the application. The application will interpret the received data and present the data in a format to be used by the user. The process is achieved through an API. (“What is an API? (Application Programming Interface) | MuleSoft,” 2019) The general term of an API is a connectivity interface to an application. However, modern APIs have characteristics that extend their value and usefulness. They follow standards such as HTTP and REST, making them developer-friendly and easily accessible. New APIs are well documented for easy consumption and versioning. (“What is an API? (Application Programming Interface) | MuleSoft,” 2019) For a developer, APIs provide a range of operations that developers may use to their advantage with the documentation for ease of implementation. The fundamental work is done by the API reducing the level of code required to be implemented by the developer. (Hoffman, 2018)

APIs provide a layer of security for users while retrieving data as applications are not fully exposed to the server in which the data required is held, nor is the server exposed to the user. The communications only comprise of small packets of data containing only relevant data. (“What is an API? (Application Programming Interface) | MuleSoft,” 2019) APIs also control access to resources, hardware, and software. (Hoffman, 2018)

A range of APIs exist for developers providing flight data. Such APIs are the OpenSky Network, FlightAware, Flight Tracker API of Aviation Edge, ADS-B Exchange, and RadarBox. The literature will communicate the variety of data available given a select two APIs. The APIs to be investigated are the OpenSky Network and FlightAware.

2.3.1 OpenSky Network

The OpenSky Network was developed as a research tool to conduct experimental studies based on real flight data. A partnership between the University of Oxford (UK), the University of Kaiserslautern (Germany), and the armasuisse (Switzerland) had developed the participatory sensor network. (Strohmeier, Martinovic, Fuchs, Schäfer, & Lenders, 2015) The network uses ADS-B sensors distributed to volunteers throughout central Europe. In 2014 the network covered 720,000 km2 and captured 30% of the commercial aircraft traffic in Europe (Schäfer, Strohmeier, Lenders, Martinovic, & Wilhelm, 2014), and by 2015 the network captured 40% of the traffic covering over 1 million km2. The network comprised of 27 sensors in 2015, as shown in Figure 3, which are low-cost and connected over the internet. (Strohmeier et al., 2015) OpenSky currently has more than 2000 sensors around the world with coverage in all continents shown in Figure 4. (“Opensky network,” 2019) The network continues to grow as more volunteers add sensors. Due to the low-cost of equipment participants can join with little difficulty. (Schäfer et al., 2014) The network exhibits the largest dataset of aircraft surveillance data of its kind. (“Opensky network,” 2019)

*A close up of a map

Description automatically generatedA picture containing tree, light, outdoor, text

Description automatically generatedFigure 4: OpenSky world-wide coverage* (“OpenSky’s reception coverage (July 2017) based on crowdsourced ADS-B... | Download Scientific Diagram,” 2019)

*Figure 3: 27 OpenSky Network sensors 2017*

(Strohmeier et al., 2015)

Commercial flight tracking networks such as the market leader Flightradar24, FlightAware, and Plane Finder do not store the raw ADS-B or Mode S messages for experimental purposes. (Schafer et al., 2016) OpenSky offers more than 15 trillion historical ADS-B, Mode S, and FLARM messages. (“Opensky network,” 2019) It records all messages as they are received by the sensor nodes and stored on a MySQL database server. (Schäfer et al., 2014) The data is accessible by a REST API represented as state vectors were Java or Python can be used to bind to the API. The state can be retrieved as a vector in the form of a JSON object. (The OpenSky Network, 2017) The state vector of an aircraft is a summary of the tracking data from the ADS-B and Mode S messages. The data within the state consists of primarily the position, velocity, and identity at a given time. Each aircraft/transponder is identified by a unique ICAO 24-bit address presented by its hexadecimal representation. As a message is received to the server, a record for the aircraft is created (state vector). The data required to track the aircraft, such as the ICAO address, call sign, Unix timestamp, and spatial data (position, velocity, and heading), will coincide with the state vector. (The OpenSky Network, 2017) If a member of the OpenSky Network, developers can use the 15 trillion historical ADS-B, Mode S, and FLARM messages. However, if not a member, the user may only use current flight data, i.e. real-time data. (“OpenSky REST API — The OpenSky Network API 1.4.0 documentation,” 2019)

Flight vectors are updated periodically; however, if no updated position or velocity is received within 15 seconds, the position and velocity are omitted from the state vector. The API would consider the state vector obsolete and not return further state vectors. No received data before 15 seconds; the position and velocity remain the same to the previous received.

Several functions are available to receive state vectors, flights, and tracks for the entire OpenSky Network or to receive a particular aircraft from the REST API. The root URL of the API “http://opensky-network.ord/api” allows for the functions to be appended to the endpoint of the path. The following parameters are a sample of requests that can be made to the API. (“OpenSky REST API — The OpenSky Network API 1.4.0 documentation,” 2019)

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| time | integer | Time in seconds since epoch (in Unix time stamp to retrieve states at given time or current time used if omitted) |
| icao24 | string | One/multiple transponder address to give property of that address (if omitted all aircraft state vectors are returned) |
| lamin | float | The minimum bound of the latitude (decimal degrees) |
| lomin | float | The minimum bound of the longitude (decimal degrees) |
| lamax | float | The maximum bound of the latitude (decimal degrees) |
| lomax | float | The maximum bound of the longitude (decimal degrees) |

A query request example to the REST API with the parameters of time and ICAO transponder address using URL:

### http://opensky-network.org/api/states/all?time=1572618734&icao24=c0ff33

The JSON response by the API gives properties of the following:

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| time | integer | The time which the aircraft state vectors from the response are associated. |
| states | array | The aircraft state vectors |

The property “states” are a two-dimensional array containing multiple state vectors with fields such as icao24, callsign, origin\_country, longitude, latitude, velocity, and vertical velocity.

The OpenSky Network API gives accurate and detailed data ideal for use as a data source in a flight tracking application. However, coverage of the network does not cover the main areas of Scotland or outskirts of Europe. It focuses on central areas of Europe and the United States, providing a lesser range of air traffic available and excluding users from these key locations. The network is primarily used for research where the data is optimal for that area of use, however not optimal for the use of tracking individual flights a prominent use of flight trackers today.

2.3.2 FlightAware

FlightAware, founded in 2005 in Houston Texas, is an aviation company operating the world’s largest flight tracking and data platform. The network has global connectivity in every segment of aviation. The network receives data from multiple different data sources in collaboration with other companies. They receive data from ATC systems in over 45 countries, have ADS-B ground stations/receivers in 195 countries, gain data from Aireon space-based ADS-B, and have datalink access to every major satellite provider such as ARINC, SITA, Satcom Direct, Garmin, and Honeywell GoDirect. (“About FlightAware - FlightAware,” 2019; “ADS-B Flight Tracking - FlightAware,” 2019) FlightAware’s receivers are distributed world-wide with over 20,000 currently issued to FlightAware users. The network’s ground coverage is shown in Figure 5, yellow representing MLAT and ADS-B represented in green. The network’s satellite coverage is shown in Figure 6.

A picture containing tree

Description automatically generated

*Figure 5: FlightAware ADS-B/MLAT coverage* (“FlightAware ADS-B Coverage Map - FlightAware,” 2019)

A picture containing outdoor

Description automatically generated

*Figure 6: FlightAware satellite coverage* (“FlightAware ADS-B Coverage Map - FlightAware,” 2019)

FlightAware collaborates the sources of data using AI models and algorithms, this collaboration is achieved by FlightAware’s HyperFeed engine. (“About FlightAware - FlightAware,” 2019) The data can be accessed through FlightAware’s API called FlightXML. Applications can query the API for the live flight data similarly to the OpenSky Network. The network servers store data of up to two weeks before the current time. (“Documentation - Flight Status API / Flight Tracking API / FlightAware API - Commercial Services - FlightAware,” 2019) (“FlightXML2 Web Services,” 2019) Significantly less than the vast network data store OpenSky possesses. When queried, the user can request the API to return a set of matching aircraft based on different attributes example attributes follow:

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| type | string | The aircraft type ID |
| origin | string | Airport code of origin |
| destination | string | Airport code of destination |
| airline | string | Airline code of the carrier |
| flightno | string | Flight number of the aircraft |

The API once queried responds and returns aircraft matching the query giving data of flight number, aircraft type, origin, destination, longitude/latitude, and groundspeed. A feature FlightXML provides, unlike the OpenSky Network is airport queries, a beneficial feature for the user base of plane spotting hobbyists. Queries specific to airports can return a list of scheduled flights, departed flights, en route flights to the airport, and arrived flights. (“Documentation - Flight Status API / Flight Tracking API / FlightAware API - Commercial Services - FlightAware,” 2019; “FlightXML2 Web Services,” 2019)

Developers can access the FlightXML data by having a FlightAware account. When queries are made to the API, requests must contain the user’s username and API key. The requests are transmitted by the HTTP Authentication standard for each request made to the server. Similarly to OpenSky, developers can retrieve data from FlightAware using Representational State Transfer (REST) giving responses encoded in JSON format. The network can also be accessed using Simple Object Access Protocol (SOAP); however, responses returned in JSON allow for the data to be used in environments such as mobile phone applications and web browser applications. Unlike the OpenSky Network, multiple languages can be used to bind to the REST API, such as JavaScript, Microsoft Classic ASP, Objective C, and Ruby. (“Documentation - Flight Status API / Flight Tracking API / FlightAware API - Commercial Services - FlightAware,” 2019)

The FlightXML API gives accurate data through the many data sources it compiles its data from. The use of HyperFeed allows for the most accurate data through filtering multiple data sources using the AI models and algorithms. Although similar to the OpenSky Network, having users deploy receivers to advance the network due to the collaboration with companies and ATC systems allows for vastly more extensive coverage in comparison. This coverage would allow for a higher proportion of flight traffic to be accessible to the proposed application users.

* 1. Augmented Reality

Augmented Reality (AR) is defined as a real-time or indirect view of a real-world physical environment that has been enhanced by the addition of virtual computer-generated data creating a mixed reality. Augmented reality is interactive by combining real and virtual objects to users in the aim to simplify the user’s life and enhance their perception by bringing virtual information to a surrounding. The first form of Augmented Reality dated back to the 1950s. A cinematographer, Morton Heilig, thought by bringing the viewer into the onscreen activity by taking in all the senses would encourage the viewer perception. Heilig built a prototype in 1962 of the concept called Sensorama, which predated any digital form of computing. (“Handb. Augment. Real.,” 2011)

Augmented reality today is presented as a futuristic technology with users primarily being exposed to it by companies incorporating the technology within mobile applications. (Emspak, 2018) Mobile devices provide advanced technical abilities such as motion sensors, a GPS sensor, and a high-resolution camera. Mobile applications using technologies such as AR have to be developed for different operating systems to ensure the spread of user base may use the application. However, developers must deal with the current capabilities the operating systems support while simultaneously utilizing the technical capabilities proposed to be implemented. (Geiger, Schickler, Pryss, Schobel, & Reichert, 2014)

Current applications available using AR are Vito Technology’s Star Walk app, which allows users to point their phone camera towards the sky to see the names of stars and planets. An application which uses similar technology to the proposed flight tracking application uses the user’s GPS location and their camera to collect information about the user’s surrounding. From this, data is displayed about the nearby restaurants, stores, and points of interest within the user’s surrounding. (Emspak, 2018)

Two forms of simple augmented reality exist, Marker-based AR and Location-based AR. Both use the user’s camera; however, Marker-based uses visual cues where Location-based uses positional data from the user’s mobile phone, such as the GPS and compass. (Patkar, Singh, & Birje, 2013) Marker-based AR uses images that can be detected by a user’s camera and used to project a virtual item into the scene, as shown in Figure 7.

A picture containing person

Description automatically generatedA picture containing person, sewing machine

Description automatically generated

*Figure 8: Location based AR (Vakoms, 2019)*

*Figure 7: Marker based AR*

*(“example of marker-based AR | Download Scientific Diagram,” 2019)*

Location-based AR, unlike Marker-based, is not bound to a select area to project its image. The application using Location-based AR does not require a marker to display content; it only requires the GPS location of the user’s phone, the direction in which it is pointed and the location of where the content is to be displayed as shown in Figure 8. Location-based AR is more interactive in comparison to Marker-based, and its functionality more aligned to how content would be displayed for the use of augmented flight tracking. (Patkar et al., 2013) Location-based AR is a robust technology suitable in large-scale environments; however, it tends to be inaccurate in its positioning. (Burkard, Fuchs-Kittowski, Himberger, Fischer, & Pfennigschmidt, 2017) Using Augmented Reality in applications greatly increases the user experience for the applications users as it brings the content to the user dependent on the actions of the user.

Developers can use a range of SDKs to incorporate Augmented Reality into native mobile applications; however, several limit SDKs are available for web-based AR. The biggest complication with WebAR is browser compatibility, still a current issue to date with WebAR experiences. Limited browsers have support for the different sensor’s API. Additionally, not all devices have the appropriate sensors equipped where the App Store can filter whether an application can be downloaded based on the requirements of the application; however, web applications have no control of these checks. (“Web vs App (AR edition) - Agora.io - Medium,” 2019)

2.4.1 AR.js

AR.js is an online library accessible on the web for the use of Augmented Reality within web applications to create WebAR, which works within default browsers with no additional special browser application. The library has features that enable Marker-based AR and Location-based AR. (Etienne, 2018) As it runs purely on the web browser, it requires no installation. AR.js runs on all mobile platforms such as Android, iOS11, and Windows mobile. (Etinenne, 2017) The advantages of AR.js are the running efficiency on mobile phones. Newer mobiles have higher performance graphics cards, so the frames per second are greater than older mobiles; however, mobile phones four years old measure frames per second at 60fps, still relatively efficient. As it is web-based, no installation is required for the user, and the JavaScript is based on three.js with its jsartoolkit5 tool kit. The AR.js library also works with any browser with WebGL. (Etienne, 2018)

AR.js can be implemented using A-Frame. This combination allows for the smooth implementation of Augmented Reality by combining the two frameworks. (Etinenne, 2017) Using the A-Frame implementation provides custom components that allow for the integration of data from GPS sensors. GPS entities can be created that have specific longitude/latitude values. The values can be added using a script which will be loaded by the FlightAware API. Once an entity/entities have been added and a camera entity created, the system calculates the user’s position and the distance between each GPS entity for every frame. From this, AR.js can use the user’s sensors to acquire the phone's orientation/position and display content for each GPS entity using the user’s camera. As the camera moves, the orientation and position are updated. (Etienne, 2018)

2.4.2 ARKit

ARKit is a specific Augment Reality SDK for iOS 11 to create AR components for native mobile applications. (“Web vs App (AR edition) - Agora.io - Medium,” 2019) The SDK can be used to combine multiple libraries giving the opportunity to create different varying AR applications.

When an ARSKView session is created, it creates a virtual world centred on the user’s mobile phone’s current location. Virtual objects are created and positioned relative to that current location. The disadvantages to ARKit are that it will not render anchors that are over 100 meters away from the current position. The AR session will also not update the world origin as the user moves. (Smith, 2018)

To rectify the two issues, CoreLocation is used. CoreLocation determines a mobile device’s geographic location, such as its altitude, orientation, and position relative to a neighbouring iBeacon device. (“Core Location | Apple Developer Documentation,” 2019) The CoreLocation framework collects the relevant data using any available components present on the mobile device such as Wi-Fi, GPS, and Bluetooth. If the location is further than 100 meters away from the current user location, the pin needs to be moved closer to the user. Then as the user approaches the pin, the pin should move further away. This is accomplished by recentering the AR world’s origin. Recentering is achieved by checking if the user is a specific distance away from the original position of the AR origin. If the user is further than the specified distance, the current pins are removed, the AR session is restarted, so the world is recentered to the new user position, and the pins are placed. (Smith, 2018)

* 1. Usability

Usability is considered one of the main elements regarding software quality. It validates the ability of the software to be understood if the software is functional and appealing to users. (Rivero, Kawakami, & Conte, 2014) Usability on web applications can have the effect of the users accepting the application or not accepting the application. It is necessary to use technologies to assist in the evaluation and improvement of usability features in the application development process as the cost and time of correcting software problems later into development can cause more disruption than if they had been corrected earlier in development. Despite the importance of usability, developers tend to neglect the concept as just one of several aspects to determine software quality. The most common method of evaluating web applications is through user testing. A user would interact with the software where a spectator would identify usability problems affecting the user. The method can become time-consuming and costly due to multiple user tests being conducted to measure a range of user feedback. (Rivero et al., 2014) Different usability inspection methods have been created to filter usability issues in different mediums.

2.5.1 Usability Inspection Methods

Usability inspection methods use experts experience in the field of usability and previous user research to ascertain obvious usability issues throughout application development. Using the methods avoids the need to have user testing where users are exposed to a version of a developed application whereby the users critique the application against a set of criteria. These methods allow for quicker and less costly alternatives to the traditional user testing method to establish usability faults. However, this methodology represents a minimum standard where experts may not establish all issues present compared to user testing, where issues revealed were not considered by the methodology. (Baxter, Courage, & Caine, 2015) While in initial mock-up stages of design, usability inspection methods should be used to avoid redesigns of implemented prototypes to prevent time wastage and to adhere to standards to ensure heightened usability for the user base of the flight tracking application.

2.5.1.1 Heuristic Evaluation

One usability inspection method is the heuristic evaluation. Jacob Nielson and Rolf Molich introduced the method as a means for practitioners to save time and money as an alternative to lab-based usability studies. (Nielsen, 1989; Nielsen & Molich, 1990) The heuristic evaluation is the most commonly used method in user-centred design to identify usability constraints. (Wilson, 2014) The method adheres to ten heuristics that applications should follow for good user experience. However, not all ten can always be adhered to as they may conflict with the application. Nor where all ten heuristics are adhered to will the application be guaranteed to meet users’ needs, yet it is less likely difficulties of poor design will emerge. (Baxter et al., 2015) The ten heuristics are as follows:

|  |  |
| --- | --- |
| **Heuristic** | **Definition** |
| Visibility of system status | Keeping the user informed of the progress of the application and feedback given in a sensible time. |
| Match between system and the real world | Avoid use of technical jargon. Information presented in a logical format. |
| User control and freedom | Allow users to control the movement through the system. Able to return to previous states. |
| Consistency and standards | Be consistent throughout application such as jargon, layout and actions. Follow known conventions and principles. |
| Error prevention | Avoid the possibility for users to make errors. If an error has been made notify the user to correct before advancing to the next state. |
| Recognition rather than recall | Make options or information accessible across application when required. Users not relied to remember how to use application. |
| Flexibility and efficiency of use | Make shortcuts available for expert users but hidden for novice users. Application to be customizable for users based on regularity of use. |
| Aesthetic and minimalist design | Avoid irrelevant information. Minimal design to avoid overloading user. |
| Help users recognise, diagnose, and recover from errors | In event of errors inform users with error messages and instructions on how to recover. |
| Help and documentation | Instructions on how to use application if needed by users. Instructions should be brief, easy to locate and focused on specific task. |

There are three general approaches to conduct a heuristic evaluation. An object-based heuristic evaluation, where evaluators examine particular user interface objects for problems related to the heuristics; such objects can be phone screens, web pages, windows, dialog boxes, menus, and controls. A second approach is a task-based heuristic evaluation, where evaluators are given tasks to complete and are asked to report problems related to the ten heuristics. The third approach is a hybrid of the object-based and task-based approaches. Evaluators are first given tasks to complete reporting issues related to the heuristics then evaluate the user interface objects against the heuristics. (Wilson, 2014)

The ten heuristics can be used as a requirement specification while designing mock-ups of the flight tracking application to ensure the application follows standard usability conventions of design.

2.5.1.2 Cognitive Walkthrough

A second usability inspection method is the cognitive walkthrough, where an evaluator(s) work through a sequence of tasks and asks a set of questions from the viewpoint of the user. The cognitive walkthrough aims to gauge the system’s learnability for new users. The initial purpose of the cognitive walkthrough was aimed to evaluate small walk-up systems such as kiosks and ATMs, where users would be exposed to the systems without any form of training. The tool has now been deployed to gauge the learnability of more sophisticated software systems on new users. The advantages of using a cognitive walkthrough are the ability to deploy the tool during any phase of development, provides suggestions on improvements to increase learnability, the efficiency of applying the tool, and may be completed by user(s) without any form of previous access to the system. However, the tool is limited by the value of data given due to the skills of the evaluators. Additionally, it does not provide a record of the frequency or severity of issues encountered by users. (Baxter et al., 2015) (“Cognitive Walkthrough | Usability Body of Knowledge,” 2019)

2.5.2 System Usability Scale (SUS)

Usability can be measured using a scale called the System Usability Scale. The scale provides a quick and rudimentary measurement of the usability of a variety of products or services such as hardware, software, websites, and applications. John Brooke created the SUS in 1986 and consisted of ten questions with five possible answers ranging from strongly agree to strongly disagree, using a 1-5 metric. An industry-standard measurement tool with references in over 1300 articles/publications. The benefits of the tool include its scalability to participants, reliable results from a small sample size of participants, and its ability to accurately determine usable systems compared to unusable systems. (“System Usability Scale (SUS),” 2017)

Scores are interpreted by converting the participant's score for each question into a new number. Scores that are odd are subtracted by 1; for example, a score of strongly agree giving a value of 5 would be subtracted by 1, giving an overall score of 4. Scores which are even numbered are subtracted from 5. (Sauro Jeff, 2013) The conversion changes the scale of values between 0-4 with 4 being the most positive answer. All scores for the participant are added and multiplied by 2.5, changing the range from 0-40 to 0-100. Based on research, a System Usability Scale score of over 68 would be considered above the average. (“System Usability Scale (SUS),” 2017)

2.5.3 Comparison of Features and Design

A study was conducted comparing two ride-sourcing applications available in Stockholm, Uber and Heetch. The study was conducted to measure the desirability and usability of the two’s mobile applications analysing which factors influence a user’s choice on the application. The results from the study showed the application Uber to have superior desirability and usability out of 14 of 16 participants. The study aimed to determine the behaviour of users, such as if design and usability influenced the user’s preferred choice of application, what factors are most important for the user, and to what degree does the design and features impact the choice of application. (Heikinaho, Villarin, & Vilarin, 2018)

Both applications fundamentally follow the same purpose; however, they differ in design and features provided. Uber is the more recognised application operating in 600 cities and seen as the pioneer of the business model. Heetch operates on a significantly smaller scale operating in only 9 cities and has not been running in Stockholm long in comparison to Uber. This advantage to Uber having built a larger brand may have assisted in preference of design due to recognisability.

Part one of the experiment was the investigation of desirability using a method developed by Microsoft for measuring factors of “fun” and “desire” while using a software product. The method uses reaction cards where user subjects select a word from a given list to give feedback on either a specific part of the software or the overall software. Two categories of positive and negative cards were created using words such as “Fun”, “Creative”, “Confusing” and “Inconsistent”. The method is flexible and designed so users can give their impressions of software. Part two of the experiment on measuring usability used the System Usability Scale as reviewed previously. (Heikinaho et al., 2018)

The results of the experiment showed that users felt the application colour choice of Uber using muted and subtle tones of black, white, and grey gave the application a feeling of professionalism and higher quality. In comparison to Heetch’s bright hot colouration being described as simplistic and fun. (Heikinaho et al., 2018) Uber’s colour palette is shown in Figure 9 and Heetch’s logo containing the colour theme of the application shown in Figure 10. Users also felt the user interface of Heetch to be more cumbersome where Uber’s user interface was clean and professional.

A screenshot of a cell phone

Description automatically generated

*Figure 9: Uber colour palette* (“31 Inspirational Brand Colors And How To Use Them | Piktochart Blog | Piktochart,” 2019)



*Figure 10: Heetch logo* (“Heetch raises $12 million to reboot its ridesharing service | Tech | Tech companies, Tech, Company logo,” 2019)

2.6 Conclusion

From the literature, it has shown there to be many varieties of techniques to be used within the different sections. It was found that with a combination of tracking technologies, ADS-B and MLAT, would be more beneficial for accurate and diverse data. Additionally, the future advancement of ADS-B will render it the most prominent tracking data type world-wide in years to come. The literature has revealed FlightAware’s FlightXML API to have the greatest current world-wide coverage compared to the OpenSky Network; furthermore, compiling multiple data sources allows for highly accurate data to be viewed by users. Reviewing the literature relevant to Augmented Reality showed Location-based AR as the appropriate choice in order to create entities of different aircraft in a live environment. AR.js allows for the AR library to be implemented onto different operating systems and web browsers with WebGL enabled. Where ARKit is specifically implemented onto native applications, not suitable for web-based applications. The ARKit also is confined to the iOS operating system, so it would limit users on different mobiles using operating systems such as Android or Windows. The usability literature showed a case of multiple usability inspection methods to improve the usability of applications early into development or at the end of development. System usability scales also give a measurement of how usable an application is.

3 Design / Methodology

3.1 Requirements Analysis

The project aims to design and develop a usability centric flight tracking web application where a user can track flights globally on a map and view flights locally using augmented reality. The project will be achieved using a front-end web application where user requests can be made and visualised. Requests will be sent to a server with an API returning the flight data. The requirements specification has been derived from the literature carried out in the research conducted in the literature review. The areas of research conducted were ‘Flight Tracking Technology’, ‘Flight Tracking Application Programming Interface’, ‘Augmented Reality’ and ‘Usability’.

The requirements are composed using the MoSCoW method to prioritise the different requirements into ‘must’, ‘should’ and ‘could’. The highest priority requirement ‘must’ if not met the web application would not fulfil the purpose of the project. The second priority ‘should’ if not met does not affect the functionality of the project; however, does improve the final application. The last priority ‘could’ are requirements which are not necessary to the overall functionality of the application; however, if development time allows for their inclusion, they would give value. From the carried out research, the following requirements were created:

3.1.1 Functional Requirements

* Must use ADS-B data source
* The application must use the most relevant data source of the current time
* Accuracy of data
* Must use MLAT data source
* The application must use the traditional data source used by older aircraft
* Ensures diversity of aircraft to be visible to users
* Must use a Flight Tracking API
* Data must be requested from an API containing live flight data.
* The flight data source must have an extensive global coverage.
* APIs such as the OpenSky Network or FlightAware must be used.
* The API must be rigorously tested
* API must be tested to ensure correct data is retrieved from the API to the user. Testing must be conducted with different parameters being passed through the request to the server.
* Must be able to make user requests to an API
* Must be able to return flight data dependent on request parameters supplied by the user.
* Must have a usable interface in order to make requests.
* Must display requested flight data onto a map
* Could implement a 3D map to display flight data.
* Must use an Augment Reality library
* Must use a web application based AR library such as AR.js in comparison to a native application based AR library such as ARKit.
* Must have interlinking screens to navigate the application
* The application navigation must be able to transfer between screens without any issues.
* Should have the ability to return to previous pages saving prior requests made rather than users having to resubmit requests.
* Should have seamless transition from the main page to the camera function of the Augmented Reality.
* Should only request the permission of use of the users camera once throughout the current visit of the web application.
* Should display all flights initially on the map
* The application should display all flights globally before any user requests on specific flight data are made.
* Should be responsive to users
* The application should be able to deal with user input and handle interrupts and be able to return to previous states.
* Could be implemented to function on a smartwatch device
* The application could be implemented or have a separate application to function on smartwatches.
* Integrate the camera from the user’s mobile phone and view augmented reality footage on the smartwatch.
* The watch could view a map of local flights.

3.1.2 Non-Functional Requirements

* Must have a scalable user interface
* The user interface must scale to different mobile devices due to the variation of screen size.
* Must be available to be used at all times
* The application must not have little or no downtime and be available for users at any time of day.
* A percentage uptime of between 95% to 99% should be achieved.
* Must be user-accessible throughout the application
* Must use usability inspection models to determine the usability of the application in order to adapt prototype to a refined product based on the feedback given by the models.
* Such models to be used are the Heuristic Evaluation, Cognitive Walkthrough and System Usability Scale.
* Must use HTML, CSS and JavaScript for development
* Due to the product being a web-based application, HTML must be used for the foundation to build the application.
* Styling of the web application to be implemented using CSS to engage visual appeal to users.
* JavaScript should be used in order to develop functionality of the web application such as calls made to the API and general functionality of the application.
* Must run on a range of web browser
* Users must be able to use the application on web browsers such as Google Chrome and Firefox.
* Must be cross-platform accessible Augmented Reality
* The AR library must be accessible on different web browsers such as Google Chrome and Firefox.
* Should have the same theme throughout all pages of the web application
* The application should follow a theme through each individual page of the application.
* Should have application documentation
* Should provide documentation to user on how to use application.
* Should have a short processing time
* The application should request data from the server API in an appropriate time of 3 seconds
* Could implement a splash screen or visual loading progression mechanism to give the user the visual cue of the loading process of the data requested. Diverting the attention of the user focusing on time and directing onto the visual cue.

3.1.3 Changes in Scope

3.2 Project Design

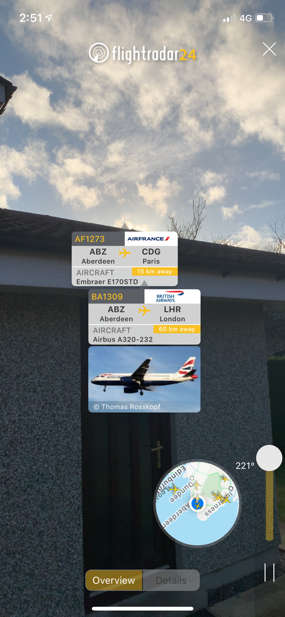
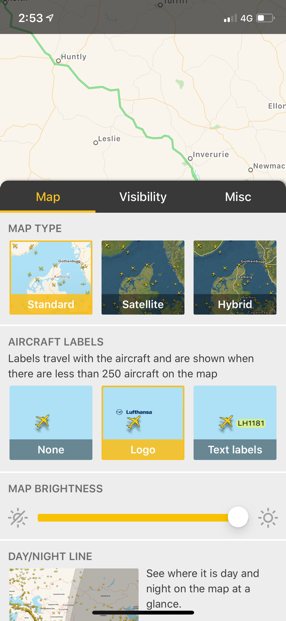
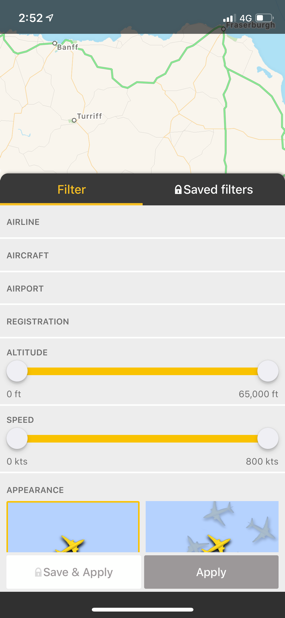
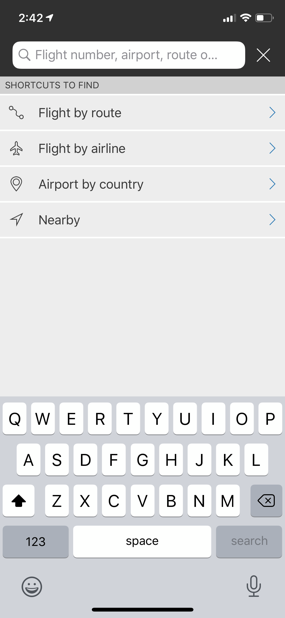
Before any implementation of the project, logos and designs were created in order to gauge user opinion and gain feedback. As a user-centric project, constant user feedback is essential for a successful usability related application. Firstly research into current applications was conducted to gain knowledge as to what design principles each application used. Specific features were selected from said applications in order to adapt and take forward into wireframes. From the created wireframes mockups would then be created which would be subjected to user feedback. Where lastly colour schemes would be selected and used on the preferred designs selected from user feedback to get then further user feedback to take to a finalised design for implementation.

3.2.1 Investigation of Current Native Based Applications

As there are no flight tracking applications specifically designed for running off of a web browser, an investigation was conducted on design principles and features on three native based flight tracking applications. The applications investigated was the market-leading, based on the number of downloads through the App Store, application “Flightradar24” and two competitors being “Flight Tracker” and “FlightAware”.

3.2.1.1 Flightradar24

The initial design is seen as simplistic with the screen being filled with a map. The header bar and options bar presented at the bottom transparently covers the map; however, with subtlety to still give the illusion that the entire page is fitted with the map.



Filter flight options global map

Search specific flight

Changes visuals on map

AR displaying flights details

Button directing users to AR functionality.

Button to take users back to their current location.

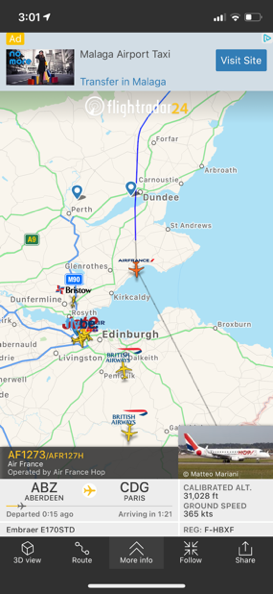
Bright colour of contrast of plane icon representing a live aircraft.

Footer option bar giving multiple options of alterations to the live feed/map.

Search box which allows user to search specific flights by different parameters.



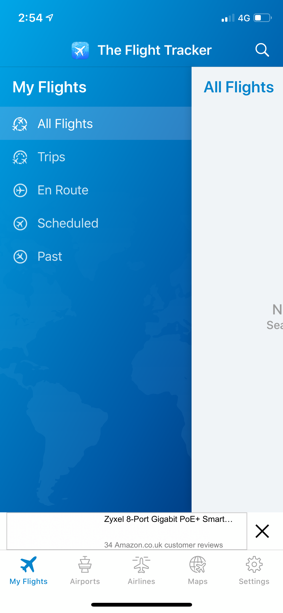
On link of specific plane icon shows details of flight.



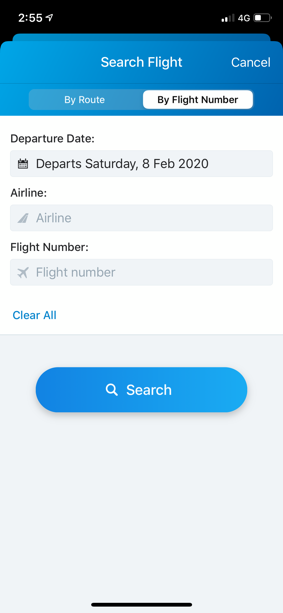
3.2.1.2 Flight Tracker

The main page is designed not to take the user directly to the content of the application. With the cooling blue colour scheme used throughout gives a calming effect while using the application. The design requires the user to search a specific flight to display on a map.

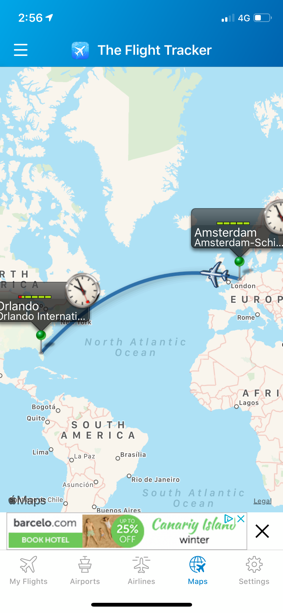
Footer bar for different search types, map view and settings menu.



“My Flights” side menu gives different options from flights already searched.



Search flights by either the airline and flight number or the route.



Flights added to the “My Flights” list are displayed on the map reducing the clutter of the map.



Search button to search flights to track.

None cluttered visual introductory page. Displays users’ flights selected to track.

Button to take the user to their “My Flights”.

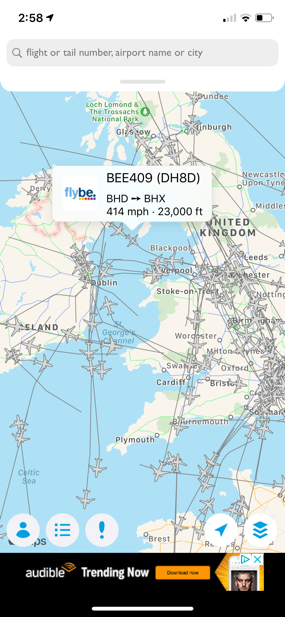
Consistent header of the title and logo of the application used throughout the pages.

3.2.1.3 FlightAware

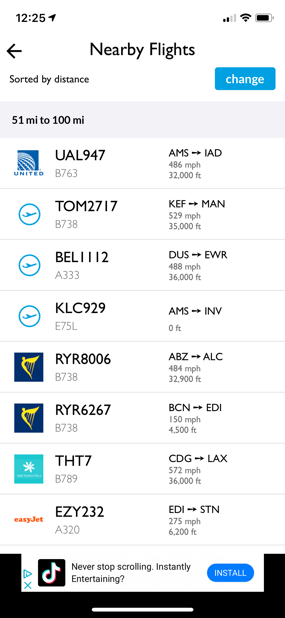
The design of the FlightAware application follows a simple format similar to the design of Flightradar24. As the user enters the application, the user is presented with a search window displaying recent searches and flight alerts. The window can be swiped to reveal the main interface of the map. It follows a similar format where the map fills the entire page accompanied by the addition of option buttons located at the bottom of the page.

Search bar to reveal dropdown window to track specific flights.

Option buttons giving user additional content and available alterations.



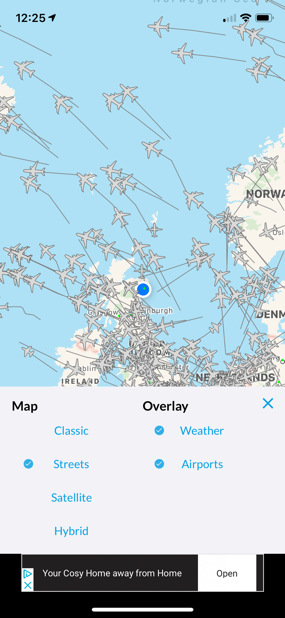
Users’ selected flights to track.



Nearby flights based on users’ location.



Search window and alert setup.



Settings window to alter map.

Trails indicating recent flight path of aircraft.

On click of plane icon displays information about the flight.

Button to direct user back to their current location.

Plane icon follows colour scheme of the page. Not contrasting with other features, blends into the design.

3.2.2 Logo Design

Before designs where created possible names of the application were investigated. Taking inspiration from the applications previously studied all make reference to “Flight” and reference to awareness of said flights. Two names indicate a form of discovery with “Flightradar24” denoting a radar signifying some form of tool to track the flights. Were “Flight Tracker” takes the literal route of using the word tracker to imply its intention as a flight finding application. The final application “FlightAware” uses the word of aware to give the sense of the application being alert to the surrounding aircraft around a user. A mind map was created, as shown in Figure 11, to run through ideas of potential names. The names were created following the same naming principles as the three investigated apps have used. A miniature survey was conducted by asking ten individuals their opinion of a preference of name yielding the results in Figure 12. The name Flight Detector was hence taken forward into the development of the logo.



*Figure 11: Application Name Mind Map*

*Figure 12: Application Name Doughnut Chart*

3.2.2.1 Logo Sketches

With a name for the application having been chosen, from the multiple possible names, the name would be taken forward to create sketches of potential logos for the application. The logo sketches would take inspiration from the two key elements of the application. These elements being aircraft and the tracking technology, such as radar, in order to track the aircraft. The application “FlightAware” has no form of imagery in its logo; however, it does make use of a plane in its native based app icon visible on mobile devices. “Flight Tracker” and “Flightradar24” both make use of imagery where one uses a plane, and the other uses a radar symbol in order to give imagery related to the roots of the function of the application. The following sketches where drawn, as shown in Figure 13-16, further accompanying the name of the application to give a further indication to the user over the purpose of the application however through the means of imagery. The imagery used was the use of planes with a variety of different vectors of movement, also using the imagery of radar pulses to associate the user to tracking. The sketches were later taken forward into the next stage of mockups.

*Figure 16: Logo Sketch 4*

*Figure 15: Logo Sketch 3*

*Figure 13: Logo Sketch 1*

*Figure 14: Logo Sketch 2*



3.2.2.2 Logo Mockups

*Figure 17: Logo Mockup 1*

*Figure 18: Logo Mockup 2*



*Figure 19: Logo Mockup 3*

*Figure 20: Logo Mockup 4*



*Figure 21: Logo Mockup 5*



*Figure 22: Logo Mockup 6*

A close up of a logo

Description automatically generatedA close up of a logo

Description automatically generated

*Figure 23: Logo Mockup 7*

*Figure 24: Logo Mockup 8*

*Figure 26: Logo Mockup 10*



*Figure 25: Logo Mockup 9*

A picture containing airplane

Description automatically generated

*Figure 27: Logo Mockup 11*

From the mockups, Figure 17-27, user feedback was taken asking participating users to vote on their preferred logo design. The survey allowed for a multiple choice answer gauging a trend as to which logo designs were preferred. From the doughnut chart, Figure 28, it is shown that two designs took particular favour over the given selection to users. “Logo Mockup 5” and “Logo Mockup 6” were omitted from the doughnut chart due to no users taking preference to either design. Although “Logo Mockup 1” and “Logo Mockup 11” took quite similar favour, “Logo Mockup 11” received 5% more of the vote share resulting in the design to be taken as the applications chosen logo.

*Figure 28: Logo Mockup Doughnut Chart*

3.2.3 Web Application Page Designs

Before the implementation of the project, wireframes were created, taking inspiration from design features of the three investigated applications. The created wireframes would give a base plan, and an indication of where design elements could be situated. It also provided an opportunity to visualise how pages would be linked, showing the flow of progression throughout the application and how features could be accessed. The first wireframes created were for the home page of the web application. With a standard design, the flow could be conceptualised, showing what buttons could provide what functionality. Once an application flow was established the remaining pages and functionality could be composed into wireframes.

3.2.3.1 Home Page Wireframes

The wireframes followed the principle of creating a simplistic design. This principle rendered the design to only incorporate a minimal number of buttons. The entire page would consist primarily of the map, taking inspiration from “Flightradar24” and “FlightAware” following this same strategy of design. Buttons residing on the home page would also follow the design that “FlightAware” had adopted overlapping the buttons on the map rather than a footer containing the buttons. This would continue to give the impression that the map is the predominant element of the page. Due to the simplicity in nature of the proposed design the wireframe designs represent this with only using three buttons of Search, Clear and AR.

*Figure 31: Home*

*Wireframe 3*

*Figure 30: Home*

*Wireframe 2*



*Figure 29: Home*

*Wireframe 1*

3.2.3.2 Search / Flight Data Wireframes



*Figure 32: Search Wireframe*

*Figure 33: Flight Data Wireframe*

3.2.3. Navigation Structure

For the purposes to show page flow the first wireframe from each page, Figure 29, , will be used to demonstrate the proposed navigation structure.