Flight Tracking API

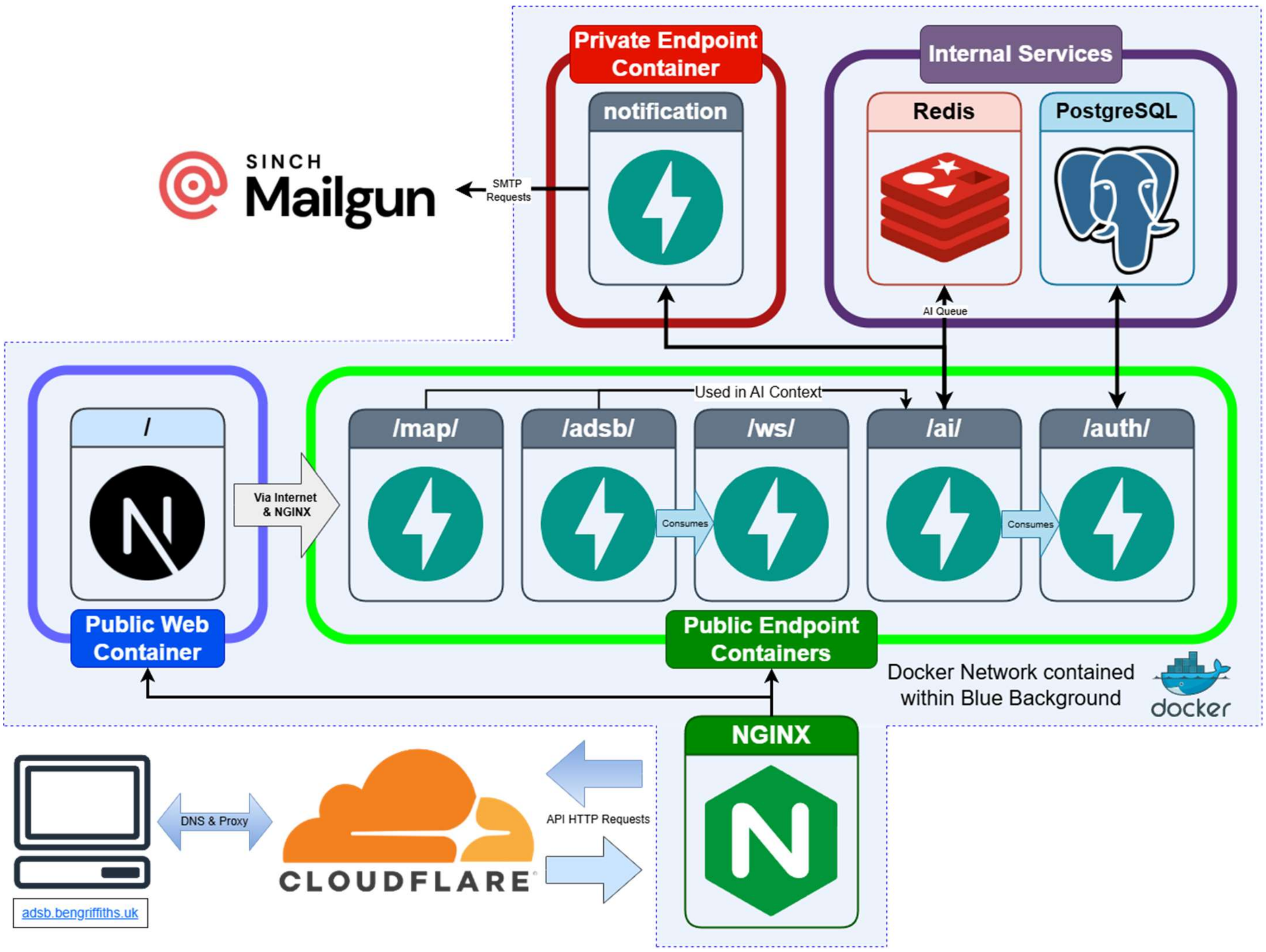
Architecture

This service is designed as a unified API for aircraft tracking and information retrieval for planning, monitoring, and operations. It allows real-time data to be combined from numerous sources into a single API, providing additional ability such as mapping, as well as natural language summaries of the operations in a given area.

While the majority of features are provided free of charge, AI natural language summaries (powered by GPT-4.1-mini) are fulfilled using a credit system, with 1000 free credits granted on sign-up. For these requests, authorisation is always required using the authentication system provided.

AI summary jobs are processed asynchronously, queued in the background using Redis, and provided via a SSE address to allow polling until the response is ready.

All API features are consumed by the demo frontend implemented at https://adsb.bengriffiths.uk.



Accessing Microservices

The above diagram demonstrates the design of the service, which is comprised of a combination of microservices to maximise horizontal scalability when deployed in the cloud. However, this microservice implementation is made transparent to consumers via the use of NGINX as a reverse-proxy.

As shown, microservices are accessed by their different addresses. For example, to login, one would submit a correctly formatted POST request to:

https://adsb.bengriffiths.uk/auth/login

When NGINX receives the request, it will automatically strip the /auth element of the address and forward it to the authorisation management container – meaning that the forwarding implementation is also transparent to the microservices themselves, simplifying development.

Technology Stack

The following technologies have been utilised, with a more detailed justification available in the design proposal document.

Component	Technology	Rationale
API Library	FastAPI	High-performance, easy to develop and scale with native OpenAPI docs via Swagger UI.
Auth Management	JWT with FastAPI-Users	Stateless and secure following industry-standard.
Database	PostgreSQL	Relational, performant, provides security features, and is the most used appropriate database.
Job Queueing	Redis	Buffering of AI requests, especially if moving to bare-metal inference in the future.
AI Summary	OpenAI's API GPT 4.1-mini	Balances intelligence and cost, provided with no concerns for scalability.
Notifications	MailGun Emails	Enables SMTP capability with secure domain registration for best user experience.
Containerisation	Docker	Ensures universal cloud deployment with easy scalability with Docker Swarm or later Kubernetes.

Note that the database, Redis queue and notification system are used internally-only, and cannot be accessed externally.

Endpoints

/adsb/

Authorisation is NOT required.

SwaggerUI available at https://adsb.bengriffiths.uk/adsb/docs

The ADSB microservice provides all real-time information for aircraft for a given area or designation. Typical usage involves receiving a list of aircraft within a bounding box from a south-west corner and a north-east corner via the /radius endpoint. As more information is desired for a specific aircraft, it can be attained using the /hex endpoint with the aircraft's unique hex value used as a parameter.

The /hex endpoint does also support finding an image of the designated aircraft, though performance testing shows this to increase latency beyond a second, and hence is recommended to be used as a secondary call. Where an image is NOT used, multiple aircraft can be queried when separating their unique hex values by commas, e.g "hex=123ab,345bc,456df" etc.

Note: ALL calls to /adsb/ are rate limited UNIVERSALLY across all users due to usage limitations from the external data provider.

```
{
 "ac": [
  {
   "hex": "06a2b4",
   "lat": 51.477511,
   "lon": -0.490189,
   "track": 89.62,
   "flight": "QTR98Q  "
  },
  {
   "hex": "3c5ef5",
   "lat": 51.467422,
   "lon": -0.4842,
   "track": 267.7,
   "flight": "N/A"
  },
  {
   "hex": "40688b",
   "lat": 51.462056,
   "lon": -0.400314,
   "track": 115.33,
   "flight": "BAW11   "
  }
 ]
}
```

/radius example response

```
{
 "r": "G-DBCC",
 "t": "A319",
 "dbFlags": null,
 "gs": 127,
 "ias": 128,
 "tas": 128,
 "desc": "AIRBUS A-319",
 "alt_baro": -50,
 "alt_geom": 375,
 "seen": 0.6
}
```

/hex example response with single aircraft requested.

```
[
 {
  "r": "G-DBCC",
  "t": "A319",
  "dbFlags": null,
  "gs": 110,
  "ias": 128,
  "tas": 128,
  "desc": "AIRBUS A-319",
  "alt_baro": 220,
  "alt_geom": 645,
  "seen": 11.4
 },
 {
  "r": "G-VIIJ",
  "t": "B772",
  "dbFlags": null,
  "gs": 231.5,
  "ias": 217,
  "tas": 222,
  "desc": "BOEING 777-200",
  "alt_baro": 1825,
  "alt_geom": 2225,
  "seen": 0
 }
]
```

/hex example response with multiple aircraft requested.

/map/

Authorisation is NOT required.

SwaggerUI available at https://adsb.bengriffiths.uk/map/docs

The mapping server provides two endpoints – one is an OpenStreetMap pass-through to receives tiles (/tile), and the other is the primary provider (/). The tile endpoint follows the same structure as OSM, with the selection of a zoom level, an x value, and a y value, providing a single OSM tile in accordance to SlippyMaps designation.

The primary endpoint takes a bounding box as input, the same format as /adsb/radius, and provides a stitched image of numerous OSM tiles with automatic selection. The map it returns will exceed the bounds of the requested area as OSM tiles will extend beyond in almost all cases, and so the API will return "extent" parameters in its response. These should be used when drawing maps with libraries such as Leaflet or Matplotlib and should be used as the "actual" drawn position for the map, allowing it to expand beyond its originally requested bounds, usually off-screen.

Cropping is not performed due to the inherent risks in rounding errors in calculating the correct bounding area. By instead providing an over-extending image, accuracy is prioritised.

Extent is provided in the following format in the header:

extent: sw_longitude,sw_latitude,ne_longitude,ne_latitude

e.g.

extent: -0.52734375,51.6180165487737,-0.3515625,51.39920565355377

/ai/

Authorisation is ALWAYS required.

SwaggerUI available at https://adsb.bengriffiths.uk/ai/docs

AI requests must be sent with a Bearer Authorisation token included as part of the request header, and the associated user must have sufficient balance. Should they not have sufficient balance, an error 400 will be returned.

Requests must include a "question", along with a bounding box once again in the same format as previous. From this, the backend will automatically collate the bounding map, all aircraft in the vicinity, and upon request from the AI agent, who has access to the /adsb/hex endpoint, will automatically retrieve additional information about specific aircraft.

Successful requests will be acknowledged with a status code 202 to indicate a successful queue. Within the response content is a json containing keys for "job_id" and "remaining_credits". The job_id is used to access the message stream whilst awaiting an AI response. The stream can be accessed at:

https://adsb.bengriffiths.uk/ai/stream/{job_id}

The only message that will be sent is a one-off response message that can be directly as the AI response via event.data. Job IDs and messages are one-to-one, and multi-turn AI conversations are not supported.

/auth/

Authorisation PROVIDED by endpoint.

SwaggerUI available at https://adsb.bengriffiths.uk/auth/docs

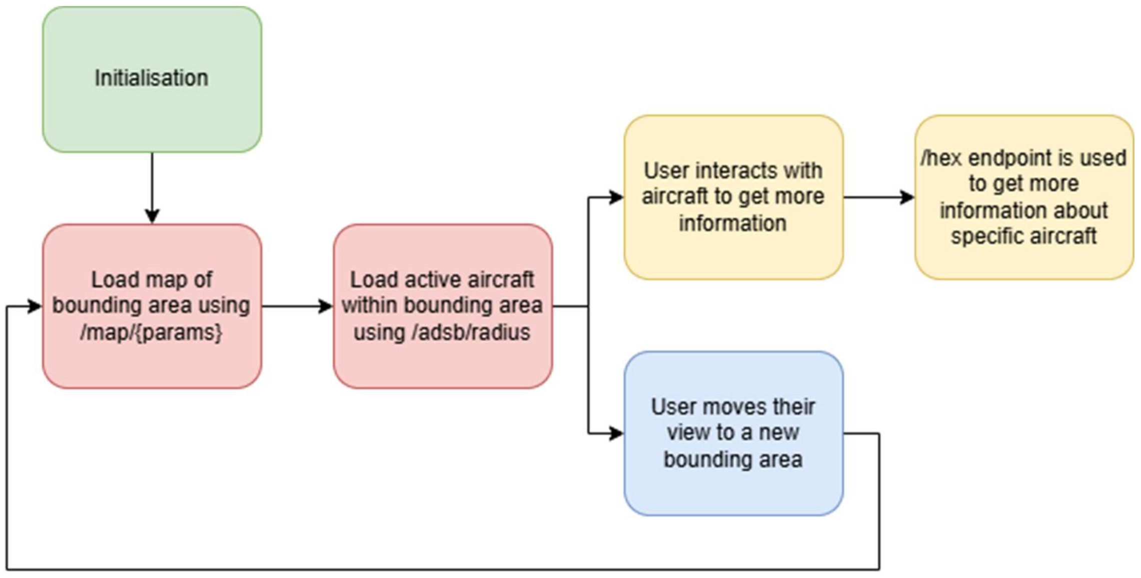
The purpose of the authorisation endpoint is primarily to manage the distribution of JWT authorisation tokens to users, providing the ability to login and access credit-gated features using their access token. This allows the backend to identify the user and check their balance / charge them for any charged services utilised.

Once registered and logged in, calls can use the provided access key to check user balance using the /balance endpoint, and their information using the /me endpoint. Historical usage can also be monitored at /usage. For all of these information endpoints, no parameters are needed, as all information is fetched by the backend database based on the access key.

Common Workflows

Free Use

For basic interaction with the service, a user will typically wish to explore local aircraft in the area on a map view. Using the specified endpoints, this is achievable with the following common flow.

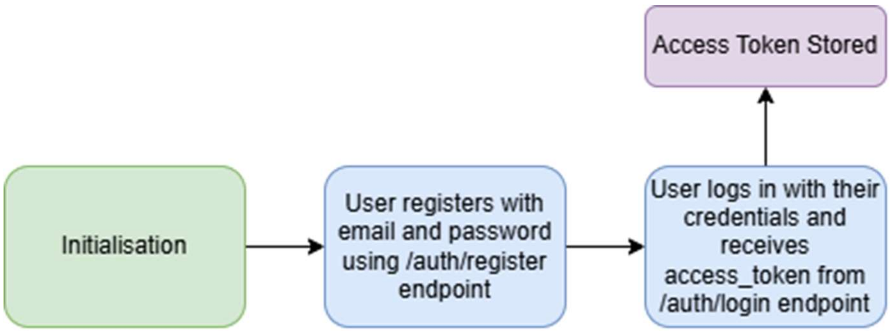

Initialisation
Load map of bounding area using /map/{params}
Load active aircraft within bounding area using /adsb/radius
User interacts with aircraft to get more information
/hex endpoint is used to get more information about specific aircraft
User moves their view to a new bounding area

As the user pans around, the current state of the map and the surrounding aircraft must be updated.

When using the WebSocket endpoint, available at /ws/, the loading of active aircraft within the bounding area is automatically polled at approximately 1Hz, and is useful for real-time web applications. However, this is still ultimately relying on the same API calls to /adsb/, albeit automatically.

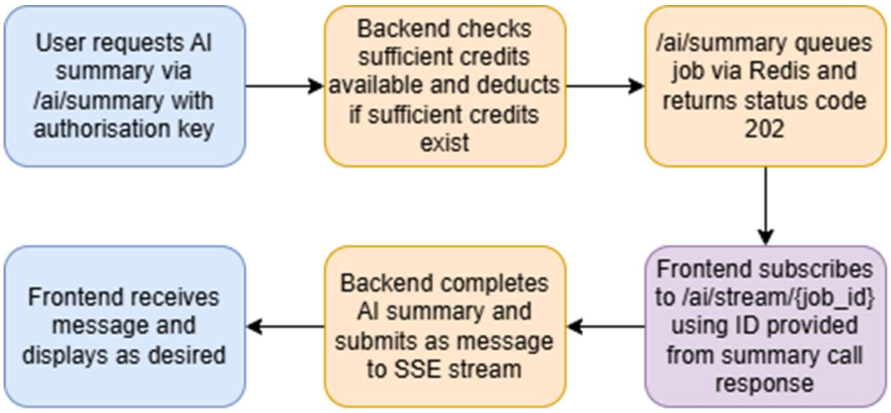
Credit-based AI Summary

To begin using premium features, the user must register and login to receive an access token.


Access Token Stored
Initialisation
User registers with email and password using /auth/register endpoint
User logs in with their credentials and receives access_token from /auth/login endpoint

Once provided, they are then able to use this access token within the header for all of their future requests, providing it using “Bearer {token}” as is standard in the industry.

With the login complete, they can now utilise the AI summary system to ask questions about the aviation activity in a given area.


User requests AI summary via /ai/summary with authorisation key
Backend checks sufficient credits available and deducts if sufficient credits exist
/ai/summary queues job via Redis and returns status code 202
Frontend subscribes to /ai/stream/{job_id} using ID provided from summary call response
Backend completes AI summary and submits as message to SSE stream
Frontend receives message and displays as desired

The user only needs to provide a question and a bounding box. As discussed previously, all information required to answer the question is automatically collated or requested by the AI agent.

Note for this workflow: It should be noted that the credits system does not currently take payment, and hence a payment integration must be implemented. In the future, new accounts are likely to no longer receive 1000 credits on sign-up.