Flight Tracker System

Visual Architecture Guide

Real-Time Aircraft Tracking Application

Target: Ryanair Graduate Programme

Timeline	Dec 11, 2024 - Jan 27, 2025
Duration	7 weeks
Stack	Java Spring Boot + React + PostgreSQL
External APIs	OpenSky Network + Mapbox GL
Deployment	Railway.app + Vercel

Generated: November 28, 2025

Table of Contents

System Diagrams

The following page contains 8 comprehensive visual diagrams covering all aspects of the system architecture.

Technology Stack Details

Backend Technologies

Component	Technology	Purpose
Framework	Spring Boot 3.x	REST API + WebSocket server
Language	Java 17+	Type-safe backend logic
Database	PostgreSQL 15	Time-series flight data storage
ORM	Spring Data JPA	Database abstraction
Real-time	STOMP/SockJS	WebSocket communication
Build Tool	Maven	Dependency management

Frontend Technologies

Component	Technology	Purpose
Framework	React 18	Component-based UI
Build Tool	Create React App	Development environment
Mapping	Mapbox GL JS	Interactive map visualization
WebSocket	@stomp/stompjs	Real-time data connection
HTTP Client	Fetch API	REST API communication
Styling	CSS3 + Glassmorphism	Modern UI design

Key Implementation Examples

1. Scheduled Flight Data Fetching (Java)

```
@Scheduled(fixedRate = 120000) // Every 2 minutes public void fetchFlights() {
OpenSkyResponse response = restTemplate .getForObject(OPENSKY_URL,
OpenSkyResponse.class); List<FlightState> flights = response.getStates() .stream()
.filter(state -> !state.isOnGround()) .map(this::convertToEntity)
.collect(Collectors.toList()); flightRepository.saveAll(flights);
webSocketNotifier.broadcast(flights); }
```

2. WebSocket Configuration (Java)

```
@Configuration @EnableWebSocketMessageBroker public class WebSocketConfig implements
WebSocketMessageBrokerConfigurer { @Override public void registerStompEndpoints(
StompEndpointRegistry registry) { registry.addEndpoint("/ws-flights")
.setAllowedOrigins("*") .withSockJS(); } }
```

3. React WebSocket Connection

```
useEffect(() => { const client = new Client({ webSocketFactory: () => new
SockJS('http://localhost:8080/ws-flights'), onConnect: () => {
client.subscribe('/topic/flights', (message) => { const updates =
JSON.parse(message.body); updateFlightMarkers(updates); }); } }); client.activate();
return () => client.deactivate(); }, []);
```

Essential Database Queries

Get Latest Position for Each Aircraft

```
SELECT DISTINCT ON (icao24) * FROM flight_states WHERE on_ground = false AND timestamp
> NOW() - INTERVAL '5 minutes' ORDER BY icao24, timestamp DESC;
```

This query retrieves the most recent position for each aircraft currently airborne.

Get Flight Trail (Historical Path)

```
SELECT latitude, longitude, altitude, timestamp FROM flight_states WHERE icao24 =
'a1b2c3' AND timestamp > NOW() - INTERVAL '1 hour' ORDER BY timestamp DESC LIMIT 20;
```

Returns the last 20 positions for a specific aircraft to draw its flight path on the map.

Calculate Dashboard Statistics

```
SELECT COUNT(DISTINCT icao24) as total_flights, AVG(altitude) as avg_altitude,
MAX(altitude) as max_altitude, COUNT(*) FILTER (WHERE velocity > 200) as fast_count
FROM flight_states WHERE timestamp > NOW() - INTERVAL '5 minutes' AND on_ground =
false;
```

Aggregates statistics for the dashboard display showing current air traffic metrics.

Development Phases Breakdown

Phase	Duration	Key Deliverables	Success Criteria
Setup & Research	3 days	• OpenSky account\n• Database design\n• First API call	Can fetch real flight data
Backend Foundation	1 week	• Scheduled polling\n• Database persistence\n• REST endpoints	API serves flight data
Real-Time System	1 week	• WebSocket setup\n• Live broadcasting\n• React component	Map updates automatically
Core Features	1 week	• Interactive map\n• Flight info panel\n• Search & filter	Full user interaction
Polish & Deploy	2 weeks	• UI refinement\n• Testing\n• Production deployment	Live, shareable URL
Final Prep	1 week	• Documentation\n• Demo video\n• Application ready	Portfolio-ready project

Core Features Summary

Feature	Technology	User Benefit
Real-time Updates	WebSocket (STOMP)	See aircraft move without refresh
Interactive Map	Mapbox GL JS	Pan, zoom, explore globally
Flight Details	React State + REST	Click any plane for information
Historical Trails	PostgreSQL Queries	Visualize flight paths over time
Search Function	Array Filtering	Find specific flights quickly
Altitude Filters	Client-side Logic	Focus on relevant aircraft
Statistics Dashboard	SQL Aggregation	Overview of air traffic
Responsive Design	CSS3 + Glassmorphism	Works on mobile and desktop

Why This Project for Ryanair Graduate Programme

Direct Relevance to Aviation Industry

• Demonstrates understanding of real-time flight operations

• Shows ability to work with aviation data standards (ICAO24, callsigns)

• Visualizes complex geographic and temporal data

Technical Skills Demonstrated

• Full-stack development (Java + React)

• Real-time system architecture (WebSockets)

• Database design for time-series data

• External API integration

• Production deployment and DevOps

Soft Skills Evidenced

• Project planning and time management (7-week timeline)

• Self-directed learning (new technologies)

• Problem-solving (architectural decisions)

• Attention to detail (UI/UX polish)

Next Steps

1. Review all diagrams and ensure you understand each component

2. Set up development environment (Java, Node.js, PostgreSQL)

3. Create OpenSky and Mapbox accounts

4. Start Day 1 of the roadmap: Make your first API call

5. Commit to GitHub daily - build a strong contribution history

6. Reference this guide whenever you're unsure about architecture

7. Deploy early and iterate - don't wait for perfection

Remember: This project isn't about replicating FlightRadar24. It's about demonstrating your ability to build a real-time, full-stack application from scratch with clean architecture and professional deployment. Focus on learning and documenting your decisions - that's what will impress in interviews.