Live Delivery Tracking For Medication Drones

Catherine Currie - s2524237

December 2025

1 Overview

For my extended implementation of the ILP drone delivery service, I have developed a real-time delivery tracking system. The service enables users to request medication delivery to their Edinburgh address, visualise the live drone on a map and receive collection notifications.

This system is designed for Edinburgh residents requiring medication delivery, particularly those who cannot easily access pharmacies. Traditional medical delivery systems often lack transparency [1], leaving patients unable to effectively plan their day around deliveries. This uncertainty creates anxiety, especially for time-sensitive or emergency medications [2]. By providing live tracking with precise arrival notifications and a transparent user interface, the service widens accessibility and reduces delivery anxiety.

2 Innovation and Benefits

Unlike static delivery systems providing periodic status updates, my service enables live drone tracking with position updates every 250ms via WebSocket. Users can see their drone's exact position on the map, with path fading visualisation(completed path shown in grey, remaining path in blue), providing clear visual delivery progress. The system also automatically handles refrigerated medication by looking up the medication in the H2 database to check for requirements, ensuring safe transport. This automated approach ensures an intuitive user experience prioritising accessibility for medical services.

3 Implementation and Justification

I integrated the A* pathfinding algorithm from CW2, including no fly-zones. Drones calculate a path to the delivery location, simulate landing until user confirms collection, then return to their service point. Since the calculated path is used multiple times, I have stored it as a JSON, eliminating duplicate calculations. I updated the pathfinding algorithm to use haversine distance calculations instead of Euclidean distance, providing more accurate real-world distance measurements for latitude/longitude coordinates.

I implemented WebSocket with STOMP and SockJS fallback for real-time tracking. The system broadcasts position updates at 4 updates/second, ensuring smooth animation. Features include order-specific filtering, automatic order status detection (QUEUED → IN-TRANSIT → ARRIVED → COLLECTED → COMPLETED), and bi-directional communication.

I chose H2 database with Hibernate for quick development. The database auto-creates on startup and seeds 105 medications. The Medication entity catalogues available medications, while DeliveryOrder tracks each delivery lifecycle, enabling inventory tracking and automatic drone selection.

The system uses queue processing with a fixed 5 second interval. Orders are dispatched FIFO (first in first out), when a drone becomes available. This could be later extended to prioritise emergency cases. This implementation is also event driven, ensuring status changing trigger actions in the drone simulator. Flight speed is set to 160 km/h (100 mph), an upper estimate (for demonstration purposes) based on realistic drone speeds [3].

3.1 User Interface Design

For the user interface, I used Leaflet.js for map rendering and OpenStreetMap Nominatim for address autocomplete. I initially attempted Google Places API, but encountered configuration issues, so I opted for Nominatim's API key-free approach. The UI design is inspired by other delivery services' UI (notably Deliveroo [4]). I prototyped it in Figma before implementation (Figure 1), and created custom map markers for a customised experience (Figure 3).

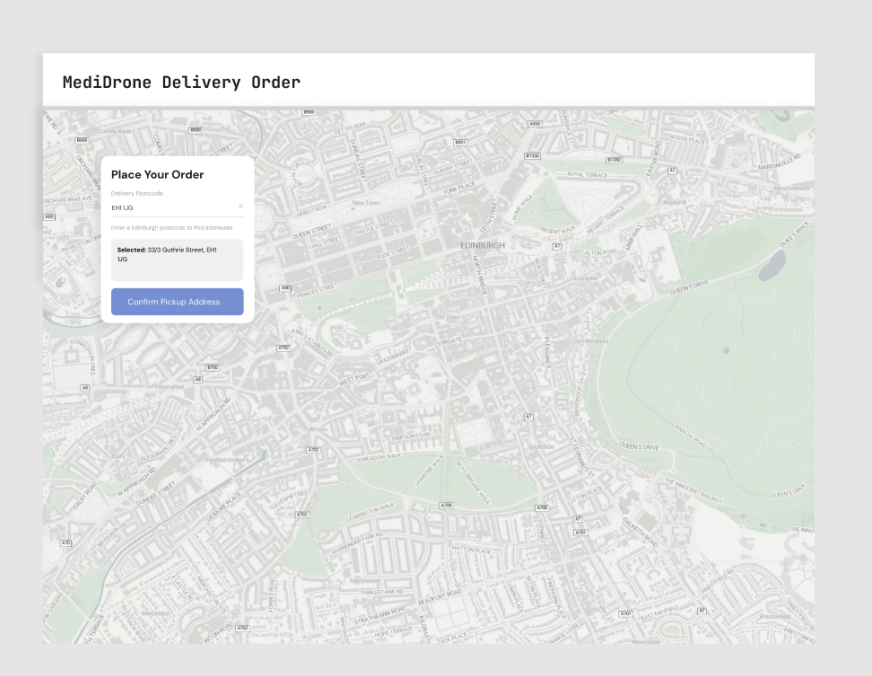

Figure 1: Initial Design Concept in Figma

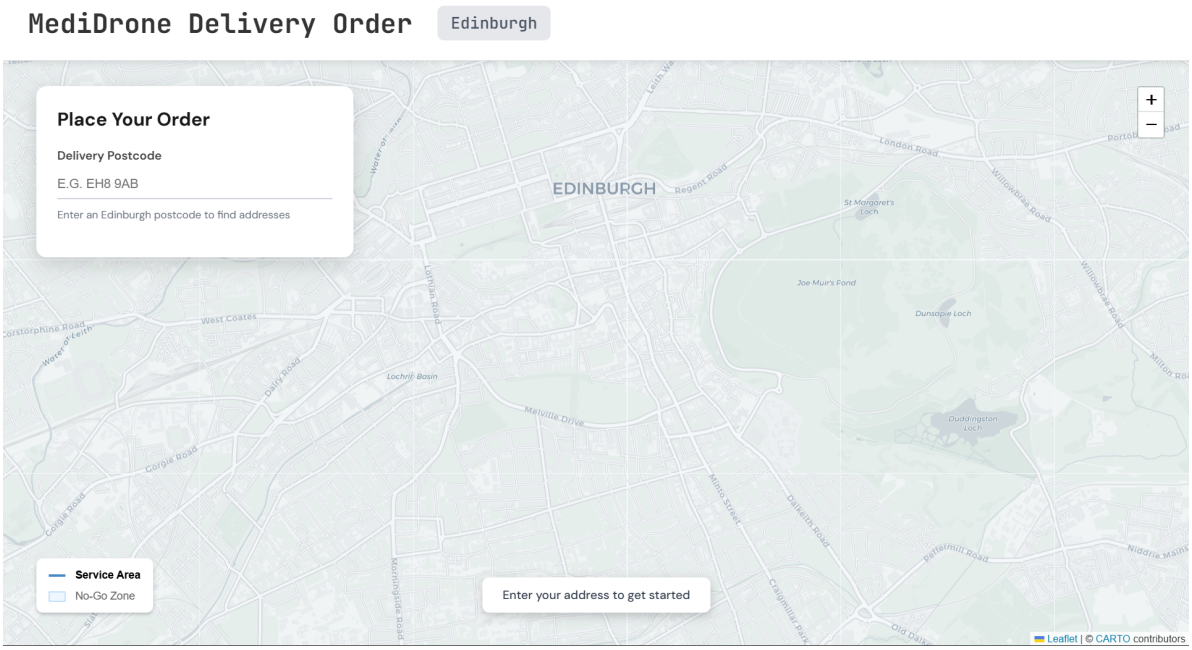

Figure 2: Final Implemented Design

Address input uses Google Maps Geocoding API to convert addresses to coordinates, prioritising ease of use. Nominatim handles autocomplete suggestions, reducing user effort and errors. All inputs are validated, to accept only Edinburgh postcodes (EH prefix). Users can refine their exact pickup location by moving the map beneath a fixed centre pin - if they move more than 10 metres from the geocoded address, reverse geocoding updates the address field dynamically.

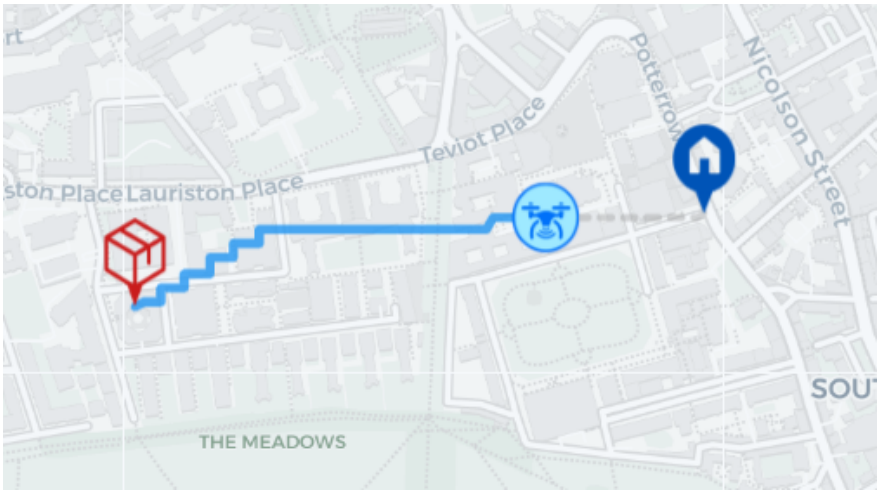

Figure 3: Custom Icons and Path Line with Real-Time Fading Visualization

Since this service is Edinburgh-specific, I added a GeoJSON boundary at the city limits, inspired by Voi bikes service area restrictions UI[5]. Users cannot place markers outside this boundary (Figure 4), preventing invalid orders and providing immediate visual feedback.

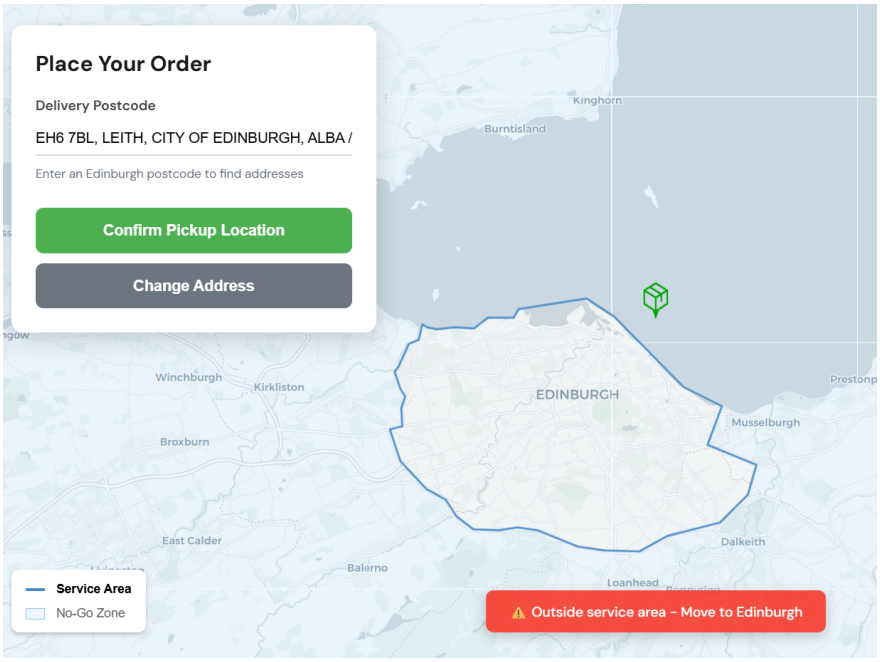

Figure 4: Edinburgh Boundary Line Preventing Out-of-Bounds Markers

Development Approach: I used AI assistance extensively throughout development for syntax guidance, code generation and debugging, and learning new technologies like WebSocket and H2 database integration. However, all architectural decisions, including the pathfinding integration, JSON path storage, and UI/UX design choices, were my own. AI served as an development tool for implementation details, and accelerated development during a constrained time frame. I maintained control over system design, user experience decisions, and problem-solving approaches.

4 Limitations Encountered

While this system successfully implements real-time drone tracking, I encountered several limitations:

UI Blocking: My initial implementation blocked the UI during pathfinding and return journey calculations. I resolved this by pre-calculating paths during the preview stage and storing them in the database as JSON, then making the return journey calculation asynchronous.

Cross-Order Drone Visibility: When refreshing the page mid-delivery, the system would display the previous order's drone on the new order's map. I fixed this by strengthening the WebSocket filtering with order number matching, added state machine guards and ensured no old markers appeared on the map.

Double Pathfinding: The system originally calculated paths twice (preview and dispatch), wasting CPU resources. I eliminated this by storing paths as JSON in the database, deserialised when needed. This improved user experience by eliminating delays between order placement and dispatch.

System Constraints: The system currently only supports one active delivery at a time. The H2 database resets on restart, meaning stock levels don't persist between sessions. Additionally, I removed ETA and distance calculations for the UI, due to time constraints and complexity in achieving accuracy during flights.

5 Conclusion

My system demonstrates how the ILP drone service can be successfully implemented as an intuitive application with a complete customer interface and an automated delivery system.

I have integrated real-time tracking and automated order processing to ensure a practical and intuitive user experience. The system handles real world requirements: address geocoding, inventory management and live position updates.

Future enhancements could include multi-drone support, persistent database storage, user authentication and realistic drone physics, including battery management, altitude, and weather conditions. The current system serves as a strong proof-of-concept, ready for further development.

References

[1] National Center for Biotechnology Information. *Transparency Issues in Medical Delivery Systems*. https://pmc.ncbi.nlm.nih.gov/articles/PMC11995619/. Accessed: 2025-12-01.

[2] National Association of Theatre. *Medication Time, Every Time: Reducing Delivery Anxiety*. https://nat.org.uk/news/statement-medication-time-every-time-people-shouldnt-fear-going-hospital/. Accessed: 2025-12-01.

[3] JOUAV. *How Fast Can a Drone Fly?* https://www.jouav.com/blog/how-fast-can-a-drone-fly.html. Accessed: 2025-12-01. 2024.

[4] Deliveroo. *Deliveroo UI*. Mobile Application Interface. Accessed: 2025-12-01.

[5] Voi Technology. *Voi Bikes Service Area Restrictions*. Mobile Application Interface. Accessed: 2025-12-01.