ILP CW3

Prakhar Sangal
s2479386

1 Innovation & Benefit

Hospital logistics coordinators and pharmacy staff manage time-sensitive medical deliveries but lack programming expertise. The CW2 REST API requires constructing JSON payloads with precise coordinates and understanding nested responses, preventing domain experts from using the system independently.

Traditional form-based dashboards with dropdown menus and coordinate fields fundamentally mismatch how these users conceptualize their work. They think conversationally: "I need 5kg of insulin with cooling to Western General by 10am tomorrow," not as discrete form fields introducing friction and errors.

My solution implements two synergistic innovations: conversational planning through Anthropic's Model Context Protocol (MCP) and real-time animated visualization. MCP eliminates structural complexity by translating natural language requests into technical constraints automatically. Users never see JSON structures or coordinate systems. The visualization component provides immediate visual confirmation through animated flight paths with color-coded drones, transforming abstract route planning into observable outcomes. In production deployment, this visualization would track actual drone positions during scheduled delivery windows.

This provides measurable benefits. Delivery planning reduces from five minutes to 30 seconds. More critically, conversational interaction enables exploratory decision-making impossible with traditional interfaces. Users naturally ask "Why are we using Drone X over Drone Y here?" and the system maintains context, comparing strategies without manual parameter manipulation. The visualization amplifies this by showing exactly which hospitals will be visited, in what sequence, with which drones - immediately showing errors that would be invisible in JSON responses. Together, these innovations transform ILP from a technical tool requiring intermediaries into an operational asset enabling independent, informed decision-making.

2 Implementation & Technical Choices

The architecture employs Next.js 14 with server-side API routes managing Claude credentials securely while maintaining conversational context. The MCP server runs as a subprocess using stdio transport for efficient bidirectional communication without network overhead.

Five MCP tools bridge conversation to operations. query_available_drones transforms natural language into structured availability queries, handling temporal and capacity constraints. plan_delivery_path orchestrates complex single and multi-drone routing, returning diagnostics if ranges are exceeded. Crucially, explain_why_unavailable articulates specific conflicts (e.g., "No 15kg drones available"), transforming opaque failures into actionable insights. Finally, get_drone_details and find_drones_with_cooling provide conversational access to specifications and filtering, eliminating manual documentation lookups.

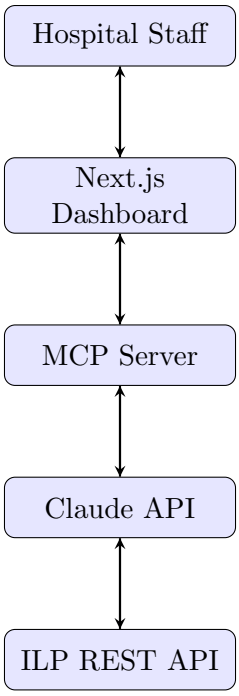

Figure 1: High-level data flow through system components

TypeScript ensures type safety across complex nested structures like flight path coordinate arrays, capability metadata, and animation progress tracking. This prevents runtime errors from Claude's unstructured responses. Zustand manages concurrent deliveries via immutable updates, chosen over Redux for minimal boilerplate and superior inference. Its atomic transactions resolved race conditions between animation frames and user state changes.

Leaflet.js implements real-time animation via frame-based interpolation at 2 moves/second. For n waypoints, duration calculates as $t = n/2$ seconds. Position interpolation between waypoints uses: $\text{position}(t) = p_i + (p_{i+1} - p_i) \cdot \alpha$ where α represents local progress. This provides 60 FPS using requestAnimationFrame with automatic cleanup preventing memory leaks during React's lifecycle management.

Three intelligent parsers handle Claude's conversational responses. Location intelligence matches coordinates against known hospitals using Euclidean distance with 220m tolerance, automatically translating "-3.2351, 55.9623" to "Western General Hospital" in both chat and map interfaces. Cost extraction employs regex patterns handling both singular and plural drone assignments, enabling automatic cost comparison without structured responses. The confirmation workflow implements a finite state machine preventing accidental scheduling: users remain in a "planned" state where exploratory questions are permitted before explicit "confirm" triggers scheduling.

3 Completeness

The solution addresses conversational planning and real-time visualization scenarios. Single-drone multi-stop routes render as connected paths showing sequential hospital visits. Multiple-drone operations display up to nine color-coded drones simultaneously with independent progress tracking, enabling coordinators to monitor parallel deliveries. In production, this would track actual GPS positions during scheduled windows; for demonstration, animations begin immediately upon

confirmation for route verification without waiting for scheduled times.

Error scenarios receive conversational handling. When deliveries exceed flight range, explain_why_unavailable articulates constraints ("No drones can reach coordinates -4.5, 56.2"). Network failures produce graceful fallbacks with clear messaging. The system handles incomplete information through follow-up questions: if users omit dates or cooling requirements, Claude requests missing details conversationally rather than displaying validation errors.

The confirmation workflow supports exploratory planning. Users remain in a 'planned' state to review costs and visualize routes before explicit commitment, which mirrors real-world logistics validation.

4 Architecture & Design

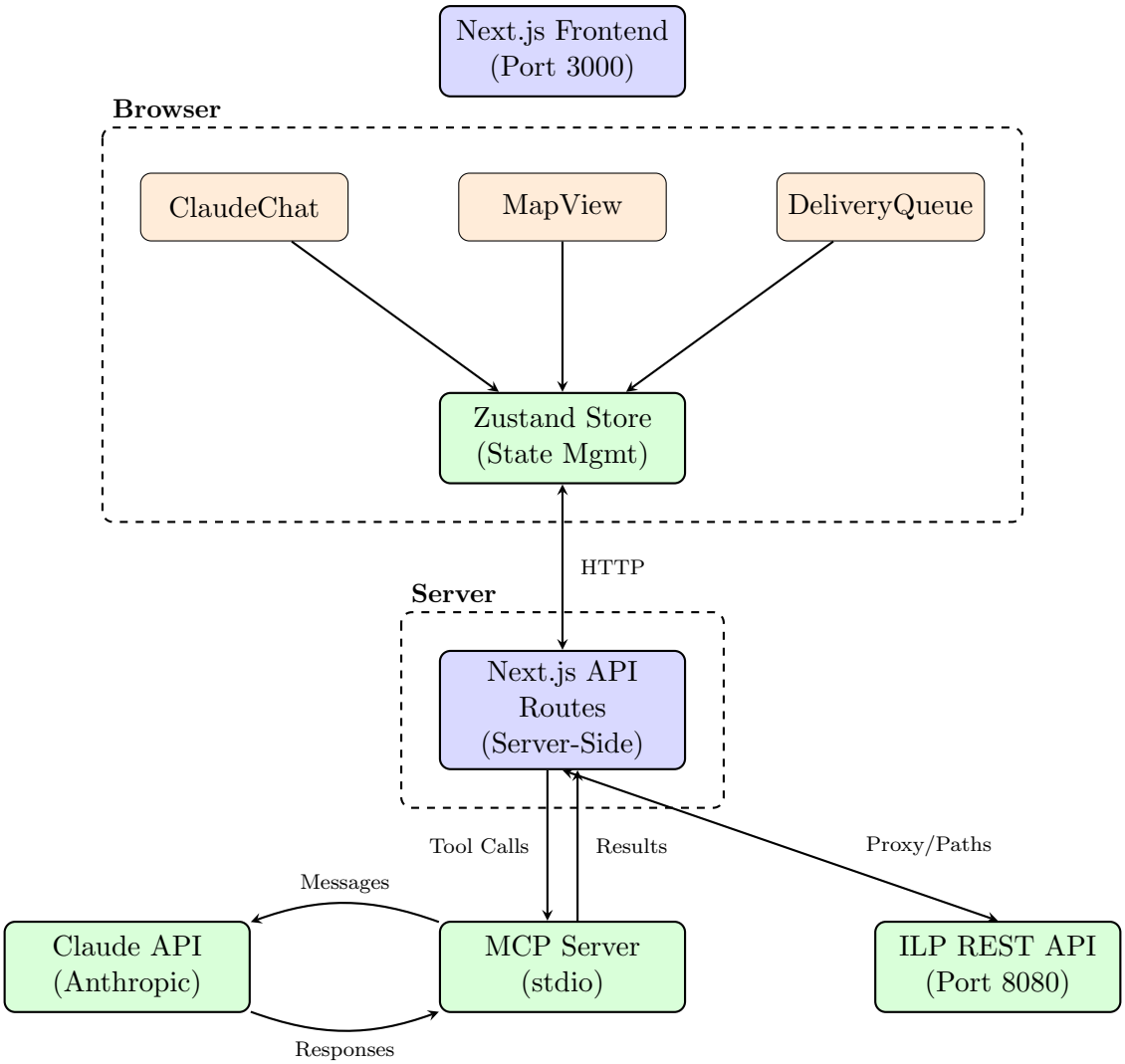

Figure 2: Component and deployment architecture

The three-layer architecture ensures separation of concerns. ClaudeChat manages message history and tool orchestration, while MapView implements visualization logic independently, de-

coupled from conversational state. DeliveryQueue handles the operations dashboard. These components communicate exclusively through the centralized Zustand store, preventing tight coupling while enabling real-time synchronization when state changes in one component must reflect in others.

Type definitions centralize data structures as a single source of truth. The PathSegment interface, Delivery type, and DroneColor mapping provide compile-time guarantees preventing shape mismatches. When backend formats evolved during development, central type updates immediately surfaced affected code through TypeScript's compiler checks.

Performance optimizations target the animation pipeline. Calculations execute via requestAnimationFrame outside React's render cycle, preventing thrashing. Zustand's atomic transactions batch state updates, avoiding cascade re-renders. Map rendering employs lazy loading, React.memo for markers, and persistent polylines.

The MCP architecture allows extensibility without UI refactoring. Features like weather routing or battery tracking require only new tool definitions and prompt updates, maintaining architectural consistency.

5 Acknowledgements

Anthropic's Claude 4.5 Sonnet assisted with React component boilerplate generation, Tailwind CSS class suggestions, and MCP SDK integration guidance during development.

External Libraries: The solution integrates Leaflet.js (visualization), Zustand (state management), and the Model Context Protocol SDK (LLM integration). Map tiles are provided by OpenStreetMap.