NAME: SWAFIYAH GICHUKI

ID. NO.: 665725

SEMESTER: FALL 2025

COURSE: SWE4900A

PROJECT PROGRESS REPORT

1. Executive Summary

This report details the progress on the Smart Medical Emergency Response System (SMERS) prototype as presented on 31st October 2025. The project is in a strong position, with the primary objective of creating a high-fidelity, interactive front-end simulation now functionally complete.

The presented system successfully simulates the entire dispatcher and admin workflow, from a polished, role-based login to a dynamic, multi-tabbed dispatch dashboard. The completed work includes AI-assisted *triage*, a manual dispatch workflow with a "nearest-responder" algorithm, and real-time map animation.

This report also documents the valuable feedback received during the presentation (see Section 2), which will now guide the project's next steps. The top priority is to evolve the system's logic to incorporate a fully AI-driven dispatch engine and further enhance the user interface.

2. Professor Feedback

Following the recent project progress presentation, valuable feedback was received. This feedback provides a clear direction for enhancing the project's quality and sophistication.

The suggestions are:

1. **Enhanced AI Logic:** The recommendation is to move beyond the current AI-assisted *triage* and develop a more advanced system that uses AI for **both** real-time case prioritization and the optimal, **automatic allocation of responders**.
2. **Improved User Flow:** The initial user flow was noted as needing refinement. A more streamlined, multi-tabbed interface was suggested to improve the dispatcher's workflow.
3. **Better User Interface (UI):** The need for a more professional, data-rich UI was highlighted, drawing inspiration from real-world emergency management systems.

These suggestions will be implemented in the next phase of development, as detailed in Section 4.

3. Work Completed (As Presented)

This section details the functionality of the prototype *as it was presented*.

3.1. System Design & Documentation (SDD)

A comprehensive Software Design Document (SDD) has been drafted, formalizing the project's design and architecture. This includes a full suite of UML diagrams (Use Case, Class, Object, Sequence, ERD, State, and Activity).

3.2. Core Application Architecture

The front-end is a robust Single-Page Application (SPA) using React.

- ➔ **Centralized State Management:** All state (requests, riders) is managed in the `App.js` component to ensure a single source of truth.
- ➔ **Role-Based Routing:** The application uses `react-router-dom` to create distinct, nested navigation flows for Dispatchers and Administrators.
- ➔ **Persistent Layout:** A main `Layout` component provides a consistent header, navigation tabs, and a floating action button to all authenticated pages.

3.3. User Interface (UI) & Experience

The UI has been built with a "high-fidelity" goal, using Material-UI and a custom theme. This includes data-rich tables, Key Performance Indicator (KPI) cards, and integrated mini-maps.

- ➔ **Login Page:** A responsive, two-column login page with mock authentication.
- ➔ **Triage Form:** A dedicated, full-page form for creating new emergency requests.
- ➔ **Tabbed Dashboard:** A multi-tabbed interface for navigating `Dashboard`, `Jobs`, `Units`, and `Profile` views.
- ➔ **Admin Dashboard:** A separate, tabbed interface for `Open Cases`, `Completed Missions`, `Payment Oversight`, and `User Management`.

3.4. Dispatcher Workflow & Simulation

The entire dispatcher workflow is fully simulated and interactive:

- ➔ **Dynamic Job Generation:** The system uses a generator to create an endless stream of new, randomized emergency jobs.
- ➔ **AI-Assisted Triage:** The Triage Form features a real-time "Suggested Priority" chip that updates as the dispatcher types symptoms.
- ➔ **Manual Dispatch Simulation:** The system flags new jobs, allowing the dispatcher to review them in the "Jobs" tab and manually click a "Dispatch" button.
- ➔ **Nearest Responder Logic:** Upon a manual dispatch, the system runs an algorithm to find the nearest *available* responder to the incident.
- ➔ **Map Animation:** Dispatched units are animated on the map, moving from their starting point to the patient, and then to the hospital.
- ➔ **Notification System:** A pulsing floating action button notifies the dispatcher that a new job has been generated and is awaiting manual review.

4. Pending Work & Next Steps

The remaining work is now prioritized based on the professor's feedback, followed by the original plan for back-end integration.

1. **Implement Professor Feedback :**
    - **Develop Full AI Dispatch Engine:** The system's logic will be upgraded. Instead of requiring manual dispatch, the system will be enhanced to automatically analyze new incidents (priority, location) and dispatch the nearest, most appropriate responder, all without dispatcher intervention.
    - **Refine User Interface:** The "Jobs" and "Units" dashboards will be further polished based on the "police dispatch" inspiration to improve data density and clarity.
    - **Streamline User Flow:** The tabbed layout and routing structure have successfully addressed this feedback. This item is considered complete.
2. **Back-End Integration (DB Connection):**
    - Set up and configure the PostgreSQL server.
    - Build a Node.js/Express back-end API to handle requests.
    - Refactor the `App.js` component to `fetch` data from the new API instead of using the local mock data generator.
3. **Advanced Routing & UI Functionality:**
    - **Protected Routes:** Implement routing logic to automatically redirect unauthenticated users from the app's internal pages back to the login page.
    - **Make Profile/Admin Forms Functional:** Connect the "Save Changes" and "Add User" forms to the future API endpoints.
4. **Future Enhancements (Post-Prototype):**
    - **Road-Based Routing:** Integrate a routing engine (e.g., OSRM) to make responder units follow the road network.
    - **Telephony Integration:** Connect the system to a service like Twilio to simulate the incoming call event via an API.

5. Conclusion

The SMERS prototype, as presented, is in an advanced state and successfully demonstrates all core features of a manual dispatch system. The front-end simulation is complete, stable, and provides a comprehensive, high-fidelity experience. The next steps will focus on implementing this feedback by evolving the system's logic from a manual tool into a fully AI-driven dispatch engine. Following this, the project will move into its final phase: building the back-end infrastructure to power the system with a live database.