Project Falcon - Technical Notes

Objective: Develop a real-time logistics dashboard for visibility and analytics across the supply chain.

The system integrates IoT truck sensors, GPS tracking, and warehouse management APIs.

Frontend: Built using Streamlit with modular widgets, map visualizations using Folium, and alert notifications.

Backend: FastAPI and PostgreSQL with Celery for background task queues.

Kafka serves as the streaming backbone, ingesting data from edge devices with Avro schema validation.

Monitoring: Prometheus and Grafana dashboards are integrated to monitor latency and uptime.

Authentication: OAuth2.0 using Auth0; access control via role-based RBAC.

Current Progress: Core data models are complete; front-end dashboard is ~70% done.

Next Steps: Integrate vendor APIs, finalize alert rule engine, and conduct internal pilot with 5 users.

Project Falcon - Technical Notes

Objective: Develop a real-time logistics dashboard for visibility and analytics across the supply chain.

The system integrates IoT truck sensors, GPS tracking, and warehouse management APIs.

Frontend: Built using Streamlit with modular widgets, map visualizations using Folium, and alert notifications.

Backend: FastAPI and PostgreSQL with Celery for background task queues.

Kafka serves as the streaming backbone, ingesting data from edge devices with Avro schema validation.

Monitoring: Prometheus and Grafana dashboards are integrated to monitor latency and uptime.

Authentication: OAuth2.0 using Auth0; access control via role-based RBAC.

Current Progress: Core data models are complete; front-end dashboard is ~70% done.

Next Steps: Integrate vendor APIs, finalize alert rule engine, and conduct internal pilot with 5 users.

Project Falcon - Technical Notes

Objective: Develop a real-time logistics dashboard for visibility and analytics across the supply chain.

The system integrates IoT truck sensors, GPS tracking, and warehouse management APIs.

Frontend: Built using Streamlit with modular widgets, map visualizations using Folium, and alert notifications.

Backend: FastAPI and PostgreSQL with Celery for background task queues.

Kafka serves as the streaming backbone, ingesting data from edge devices with Avro schema validation.

Monitoring: Prometheus and Grafana dashboards are integrated to monitor latency and uptime.

Authentication: OAuth2.0 using Auth0; access control via role-based RBAC.

Current Progress: Core data models are complete; front-end dashboard is ~70% done.

Next Steps: Integrate vendor APIs, finalize alert rule engine, and conduct internal pilot with 5 users.