Cargo Connect: Improving Airport Freight Flow

Project Plan vs 4.0

SE 6387 Advanced Software Engineering Project
Prof. R. Z. Wenkstern – UT Dallas

5/09/25

Group *1*
Stover, Bradley E.
Anuja, FNU
Patel, Smit
Elisha-Wigwe, Chijioke

Revision History

Version	Date	Description	Authors
1.0	2/20/25	Completed initial draft	BS, PS, AF, CE
2.0	2/21/25	Completed revisions and final draft	BS, PS, AF, CE
3.0	2/26/25	Revised Process Model and added Figure	BS, PS, AF, CE
4.0	5/09/25	Final updated project plan	BS, PS, AF, CE

List of figures

Contents

1. Overview

1.1 Purpose, scope and objectives

Purpose:

The purpose of this project is to create a system that effectively oversees truck operations at an airport by integrating with the DALI system and current airport systems to offer real-time tracking, routing suggestions, and smooth dock and parking assignments.

Scope:

- Show truck drivers up-to-date arrival, routing, dock assignment, and exit information in real-time.
- Retrieve scheduling and parking information by integrating with the airport system.
- Integrate with the DALI system to get route advice and suggestions
- Reduce freight handling delays and increase logistical efficiency.

Objectives:

This project aims to improve the efficiency of freight truck operations at the airport by guaranteeing the timely and accurate delivery of information, streamlining route planning to minimise traffic, improving integration with current airport management systems, and automating tasks to minimise human intervention. The project's goal is to increase operational efficiency and reduce freight movement delays [3] while adhering to regulatory and airport security standards by utilising real-time GPS monitoring and optimised routing through the DALI system.

1.2 Assumptions and constraints

Assumptions:

- The airport system offers precise and up-to-date scheduling and parking information.
- Reliable route suggestions and advice are provided by the DALI system.
- The required GPS tracking and communication equipment is installed on goods vehicles.
- Automated entrance and exit management is supported by the airport's infrastructure.

Constraints:

- Dependence on the existing airport system’s API and data accuracy.
- Potential network latency affecting real-time updates.
- Limited availability of parking and docking slots during peak hours.
- Required alignments with flight schedules and strict adherence to narrow operational windows, particularly during peak commuter hours and delivery deadlines.

1.3 Project deliverables

- A functional system with UI for truck operators displaying required information.
- Feasibility report.
- Project Plan.
- System requirement specification.
- Requirement analysis
- High-level architecture
- System design
- Test documentation.
- Presentations about cargo connect systems.

1.4 Schedule and budget summary

Schedule:

Project can be finished in the allotted amount of time. It involves evaluating the project's schedule, goals, and possible obstacles to make sure it can be completed on schedule and with all its goals met.

Timeline:

- Estimated project duration: 4 months
- Can all the milestones and completion date schedules be met?
    - For the freight transportation problem, milestones include:
        - Requirement gathering, Planning and design – Month 1
        - Development and integration – Month 2 and Moth 3
        - Testing and Deployment– Month 4
    - By limiting the requirements to a few, as well as implementing proper project management and team planning, the milestones can be met in 4 months.

For this project, a 4-month deadline is mandatory, as this project is confined to only one semester. Hence, the requirements are limited.
The learning curve duration would be a few days in the final month. The learning curve for users to adapt to a new system can be minimized by providing a training session and designing a user-friendly interface.

Tentative project plan

- Month 1
    - Design and Planning
    - Collect and analyze requirements.
    - Complete the design architecture, features, and scope.
    - Create prototypes.
- Month 2 and Month 3
    - Integration and Development
    - Create the essential functions, such as scheduling, real-time tracking, and notifications.

    - Connect to external systems (such as airport systems, traffic APIs, and GPS).
    - Perform preliminary testing on each module separately.

- Month 3:
    - Deployment, and Testing
    - Conduct thorough testing, including security, performance, and functional testing.
    - Organize end-user training sessions.
    - Complete the system based on user input and fully implement it.

2. Project Organization

2.1. Roles and Responsibilities

The assignment of roles and responsibilities to individuals is yet to happen. This will keep changing and everyone will be involved in every aspect of the project.

- Project Managers: In charge of development, resource management, and meeting deadlines.
- System Architect: Creates the integration points and system architecture.
- Developers: Put system integrations and functions into action.
- QA & Test engineers: Examine the security, performance, and dependability of the system.
- Deployment engineers: In charge of system rollout and make sure everything runs smoothly.

3. Managerial process plan

3.1 Start-up plan

The start-up plan begins with gathering input from stakeholders to define the project's scope. Next, it identifies the main technologies and key integration points. Finally, it sets up systems for tracking progress and keeping communication smooth.

3.1.1 Estimation Plan:

The duration of the project is 4 months. The 4-month deadline is mandatory, as this project is confined to only one semester. The plan is to develop time estimates for each phase of the project, ensuring resource allocation aligns with project objectives.

3.1.2 Staffing Plan:

The team members include FNU Anuja, Smit Soneshbhai Patel, Bradley Evan Stover, and Chijioke Elisha-Wigwe. Documentation and project plans have been done as a team, and individual work will be allotted in the coming time based on their expertise and knowledge.

3.1.3 Resource Acquisition Plan:

Determining the necessary software and infrastructure requirements and securing them within the project timeline to avoid delays.

- Together with the core DALI system, some potential additional software requirements include software for real-time data processing, cloud computing resources, among others.

3.2 Work plan

3.2.1 Work activities

- Sprints every week using an Agile Framework.
- Iterative testing and regular status meetings.

3.3 Risk management plan

Technical Risks:

- API compatibility issues with airport and DALI systems.
- Data accuracy and latency are affecting real-time decision-making.

Environmental Risks:

- Weather conditions impacting route optimization and travel time predictions.
- Regulatory compliance changes requiring system updates.

4. Technical process plans

4.1 Process model

Inception Phase: Requirements Gathering and Initial Analysis

Goal: Establish the project scope, identify key stakeholders and use cases, and define high-level system requirements.

Activities:

- Conduct stakeholder interviews and workshops
- Identify and prioritise use cases
- Document brief descriptions for top use cases:
- Define key system constraints and assumptions

Outputs:

- Initial Requirements Specification
- Use Case Prioritisation
- Preliminary Risk Assessment.

Elaboration & Construction Phase (Iterative)

Iteration 1: Architecture & Implementation for UC1 - Schedule to the airport

Goal: Fully develop Use Case 1 with architecture, design, implementation, and validation.

Activities:

- Define system architecture to support UC1 (including relevant subsystems)
- Create design artefacts (sequence diagrams, class diagrams, component diagrams)
- Implement core components of UC1
- Conduct unit, integration, and scenario-based testing
- Review and adjust based on internal feedback

Outputs:

- Architecture and Design Documents
- Tested and validated implementation of UC1
- Internal Feedback Report

Iteration 2: Architecture & Implementation for UC2 - Schedule to Cargo Destination

Goal: Extend and refine the architecture to support UC2 and implement full functionality.

Activities:

- Update the architecture to accommodate additional navigation and routing features
- Design diagrams focused on UC2 interaction and data handling
- Implement UC2 logic, ensuring integration with UC1 modules
- Conduct unit, integration, and regression testing

Outputs:

- Updated Design & Architecture
- Fully integrated and tested UC2 module
- Test Reports

Iteration 3: Architecture & Implementation for UC3 (Authenticate/Authorise User)

Goal: Implement secure user authentication and authorisation to support controlled access across UC1 and UC2.

Activities:

- Extend the system architecture with a simple authentication/authorisation layer

- Create detailed design artefacts: Sequence diagrams for login, token issuance, and access control, Class diagrams for user roles, permissions, and session management, Component diagrams showing integration with UC1 and UC2 flows
- Implement core components - Login interface, Authentication backend (e.g., username/password authentication), Access control filters for route-level permissions
- Conduct: Unit and security testing (e.g., input validation, brute force protection), Integration testing with navigation and scheduling flows (UC1 and UC2), Penetration test simulations (optional, for higher assurance)

Outputs:

- Updated Design & Architecture
- Fully integrated and tested UC2 module
- Test Reports

Transition Phase

Iteration 4: System Integration, Validation, and Deployment

Goal: Validate the entire system in real-world conditions and prepare for release

Activities:

- Full system acceptance testing (including UC1 and UC2 in integrated scenarios)
- User training and operational documentation
- Deploy to the staging or production environment

Outputs:

- Acceptance Test Results
- Deployed System Version 1.0
- Training and Deployment Materials

Iteration 5: Feedback Collection & Maintenance Planning

Goal: Collect feedback and prepare for ongoing support and iteration

Activities:

- Collect feedback from users and stakeholders
- Log bugs, enhancement requests, and usability issues
- Define roadmap and backlog for the next iteration cycle

Outputs:

- Maintenance & Enhancement Plan

- Updated Backlog and Risk List

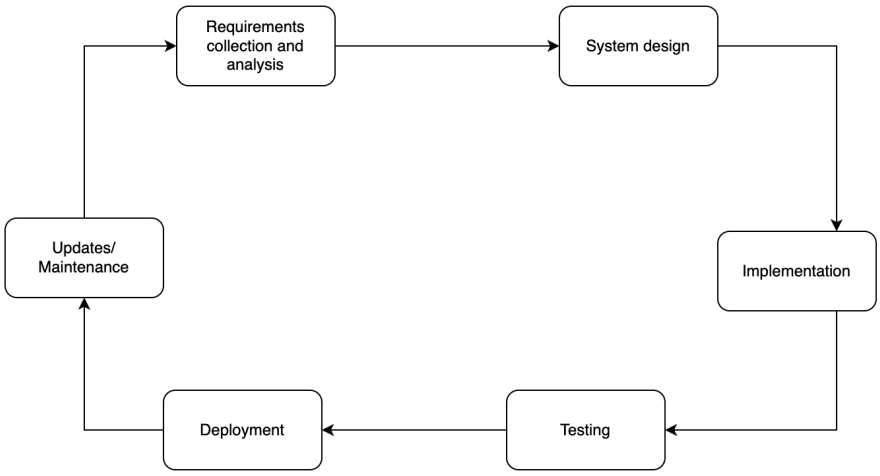

Fig 1: Iterative Software Development Process Model

4.2 Methods, tools and techniques

- **Data Collection and Integration**
  - **GPS/Location Tracking**: Real-time vehicle locations via Google Maps and Directions APIs.
  - **API Integrations**: RESTful APIs for interfacing with third-party systems, including airport systems, DALI systems, etc
  - **Middlewares**: Authentication and authorisation services to manage secure communication between systems
- **Data Processing**:
  - **Backend system:** To manage and process data from different sources
  - **Frontend systems:** Mobile app for drivers tailored to specific interactions**.**
- **Routing**:
  - **Mobile Navigation**: Based on suggestions from DALI and integrated with Google Maps APIs.
  - **Real-Time Traffic Management**: Integrate with existing traffic management systems to automate traffic flow and vehicle routing

4.3 Infrastructure plan

Network Architecture

- **Cloud Infrastructure**
  Core subsystems including the Manager, Routing System, Truck Scheduling, and Notification System are hosted on a scalable cloud platform.
- These provide:
  - Centralized deployment of microservices
  - Auto-scaling for handling varying cargo volume.
- **Edge Devices & Integration Adapters**
  Integration with external systems such as Airport_MOCK, Google Maps, Cargo Service, and DALIMOCK happens via cloud-hosted middleware with API adapters for secure and efficient communication.
- **Internal Communication**
  RESTful APIs based communication is used for synchronous subsystem interactions, while **message queues** may be employed between asynchronous components like Notification System and Mobile App.
- **Real-Time Communication**
  WebSockets enable push-based updates from the Notification System to DALIMOCK or Mobile App clients.
- **Connectivity & Access**
  Secure 5G or Wi-Fi networks provide connectivity between cloud infrastructure and mobile/edge users, ensuring timely data updates.

Data Management

- **Primary Databases**
  - The Main Database (used by multiple subsystems) stores operational data such as user profiles, cargo entries, bay availability. Recommended technologies: PostgreSQL
  - The Schedule Database, dedicated to the Truck Scheduling Subsystem, holds ETAs, slot reservations, and scheduling rules.
  - Data Synchronization - The Manager Subsystem acts as a coordination hub that manages updates across all integrated databases to avoid duplication and ensure consistency.

System Security

- **Authentication & Authorization**
  The **Authentication Subsystem** ensures secure user access across all frontends (Mobile App, Cargo Service).

4.4 Product Acceptance Plan

- Acceptance Criteria
  - **Usability**: The software must be intuitive and user-friendly, meeting all usability

specifications previously outlined.
    - **Functionality**: The software must fulfil all functional requirements posited in the SRS document.
    - **Performance**: The system must meet performance specifications as defined by requirements.
    - **Compatibility**: The system must successfully run on the specified operating systems and devices.
    - **Quality assurance**: The system must be free of critical bugs, and all test cases need to pass during the final testing phase.

- Acceptance Process
    - **Initial testing**
        - Unit and integration testing will be done at this point for the validation of individual modules.
        - If any testing failures arise, acceptance will not occur.

    - **User acceptance testing**
        - End users and stakeholders will evaluate the software to determine whether it conforms to their expectations and can adapt to a real-world scenario.
        - Stakeholders will provide the acceptance criteria.
        - User acceptance testing will span two weeks.
        - In the case that user acceptance testing is not successful, a meeting will be held to resolve all issues.

    - **Final review**
        - Our project manager, in tandem with stakeholders, will conduct a final review of the software.
        - All documentation (such as user manuals) will be reviewed as well.

    - **Sign-off**
        - Once the criteria are fulfilled, formal acceptance will be provided by the stakeholders via the sign-off document.
        - Any criteria not being met is grounds for the software to be sent back for development and re-testing.

- Deliverables for Acceptance
    - Source code and executable software
    - Testing reports
    - Documentation
    - All other artefacts

5. Supporting process plans

5.1 Configuration management plan

To ensure effective version control and collaboration, the following configuration management strategies will be implemented:

- Version Control Tool: Git
- Repository Setup: A centralized repository will be established to manage the codebase.
- Branching Strategy:
    - Separate branches for different development sectors:
        - Frontend branch for UI/UX-related code.
        - Backend branch for server-side logic.
    - A main branch will serve as the stable release branch.
- Commit Guidelines: Clear commit messages will be used to maintain traceability.

5.2 Test plan

The testing process will focus on ensuring the functionality, quality, and reliability of the application. The following testing strategies will be employed:

- Unit Testing: Test cases [4] will be defined to validate individual functional requirements.
- Code Coverage Testing: Coverage tests will be conducted to assess code quality and ensure all critical paths are tested.
- API Testing: Tools like Postman will be used for backend API testing to verify data exchange between systems.

5.3 Documentation plan

Comprehensive documentation will be created to support development, maintenance, and user guidance. This includes:

- **Technical Documentation**:
    - System requirement Specification
    - Requirement Analysis
    - High-Level System Architecture
    - System Design
    - Test Documentation

- **Project Documentation**:
    - Project Planning Documents
    - Feasibility Reports

- **User Documentation**:
    - User Manual for end users explaining app functionality.

5.4 Quality assurance plan

To maintain high-quality standards throughout the project lifecycle, the following quality assurance measures will be implemented:

- **Code Reviews**: Regular peer reviews to ensure adherence to coding standards.
- **Static Code Analysis**: Automated tools will analyse code for potential bugs, vulnerabilities, and inefficiencies.
- **CI/CD Pipeline**: Continuous Integration pipelines will automate testing and deployment processes.

Appendix A: Glossary

Term	Definition
DALI System	Distributed, Agent-based traffic Lights system for traffic management.
Agile Framework	A project management methodology involving iterative development, sprints, and continuous stakeholder feedback.
Cloud Infrastructure	Scalable computing resources for data storage, processing, and integration with external systems.
RBAC	Role-Based Access Control: A security method that restricts system access based on user roles.
TLS	Transport Layer Security: A protocol for encrypting data in transit to ensure security and privacy.
SRS	Software Requirements Specification: A document outlining the functional and non-functional requirements of the Cargo Connect system.
User Acceptance Testing	Evaluation by end users and stakeholders to confirm the system meets expectations and works in real-world scenarios.
Configuration Management	The process of tracking and controlling changes to the software codebase using tools like Git.

Appendix B: References

[1] K. Schwaber and M. Beedle, Agile Software Development with Scrum. Upper Saddle River, NJ, USA: Prentice Hall, 2001.

[2] D. Uckelmann, M. Harrison, and F. Michahelles, "An architectural approach towards the future of the Internet of Things," in Architecting the Internet of Things, D. Uckelmann, M. Harrison, and F. Michahelles, Eds. Berlin, Germany: Springer, 2011, pp. 1–24.

[3] J.-P. Rodrigue, *The Geography of Transport Systems*, 5th ed. New York, NY, USA: Routledge, 2020.

[4] P. C. Jorgensen, Software Testing: A Craftsman's Approach, 4th ed. Boca Raton, FL, USA: CRC Press, 2013.