Project Status & Participation Report

Team 1

Submission Date: 05/10/2025

Deliverable: Final submission of all deliverables

Overall Project Status

- On track
- At risk: milestones missed but date intact
- High risk: at risk with a high probability of going off-track
- Off-track: date will be missed if action is not taken

Milestones accomplished.

- Final video is ready.
- Final presentation with all documents is prepared.

Planned Milestones Not Achieved

- Null

Team Members' Task Assignment

Completed
In Progress
Deferred
Blocked
Not Achievable

Team Lead - FNU Anuja

Task Assigned	Priority	Hours Spent	Task Status
Prepared the PPT for the final submission	High	4	
Recorded the video and gave voiceover for problem definition, problem solution.	High	5	
Worked on all documentation for final submission.	High	6	

Team member 1 Smit Soneshbhai Patel

Task Assigned	Priority	Hours Spent	Task Status
Refactored Mobile Application Code	Low	4	
Refactored Backend Code	Low	2	
Prepared the PPT for the final submission	High	1	
Recorded & prepared Visuals for the video recording submission	High	5	

Team member 2 (Brad S.)

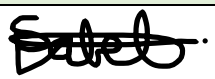
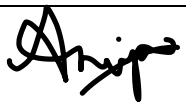
Task Assigned	Priority	Hours Spent	Task Status
Helped add new features to frontend code	High	4	
Conducted video editing project to create final presentation video	High	5	
Participated in team meetings	High	1	
Finished project closeout document	High	2	

Team member 3 (Elisha-Wigwe Chijioke)

Task Assigned	Priority	Hours Spent	Task Status
Updated DALI mock system for final demo	High	2	
Recorded DALI mock presentation for addition to final demo	High	2	

Contributed to final produce video script and recording of demo	High	5	
Updated designs for final submission	High	1	
Participated in team meetings	High	4	

Team Members' Participation

Team Member	Participation	Signature
Smit Soneshbhai Patel	25 %	
FNU Anuja	25%	
Brad S.	25 %	
Elisha-Wigwe Chijioke	25 %	

Cargo Connect: Improving Airport Freight Flow

Feasibility Report vs 4.0

SE 6387 Advanced Software Engineering Project

Prof. R. Z. Wenkstern – UT Dallas

05/09/25

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

Revision History

Version	Date	Description	Authors
1.0	2/11/25	Feasibility Report version 1	AF, SP, BS, CE
2.0	2/18/25	Feasibility Report with few class feedback	AF, SP, BS, CE
3.0	2/26/25	Feasibility report with inclusion of more content and the template	AF, SP, BS, CE
4.0	5/09/25	Final updated version	AF, SP, BS, CE

1. Executive Summary - Management Summary and Recommendation

1.1. Problem

The current freight truck operations at airports face inefficiencies due to fragmented communication and a lack of real-time coordination. Delays, congestion, and misallocated resources impact scheduling, dock assignments, and parking management. A system is needed to integrate real-time route optimization, dock assignments, and airport scheduling while leveraging the DALI system for dynamic navigation. This will enhance efficiency, reduce delays, and streamline freight logistics.

1.2. Project Goal

- Implement real-time tracking and route suggestions for freight trucks using DALI.
- Ensure seamless data exchange with airport systems for flight updates and resource allocation.
- Provide truck operators with real-time access to routing, docking, and parking information.
- Enhance airport logistics by improving freight truck flow and reducing congestion.

1.3. Success Criteria

- Seamless integration with existing airport systems for real-time data exchange.
- Accurate and timely route suggestions through DALI, reducing truck congestion and delays.
- Efficient allocation of docks and parking, improving freight handling operations.
- Real-time visibility and decision support for administrators via dashboards.
- Positive feedback from truck operators on system usability and effectiveness.

1.4. Assumptions/Risk/Obstacles

Assumptions

- The existing airport systems can seamlessly integrate with the new platform for data exchange.
- Real-time data updates from DALI and airport systems are accurate and reliable.
- Truck operators have access to the system interface for receiving and following routing instructions.

Risks

- Potential delays in integration with existing airport and logistics systems.
- Data inaccuracies leading to inefficient routing and resource allocation.
- Cybersecurity threats affecting data integrity and system performance.

Obstacles

- Resistance to adoption by truck operators due to changes in workflow.

- Technical challenges in ensuring seamless interoperability between disparate systems.
- External disruptions like weather conditions, roadblocks, or airport congestion impacting system efficiency.

2. Introduction

Airports serve as critical hubs for global trade. The movement of goods through airports is often hampered by critical issues like poor coordination between freight trucks and airport operations. This disconnect leads to delays, inefficient routing, and scheduling conflicts, increasing the costs and slowing down logistics for both airports and trucking companies. The Cargo Connect will address this challenge by linking freight trucks, existing airport systems, and the DALI system to improve communication and streamline operations. The goal is to cut delays, enhance efficiency, and optimize the flow of freight.This feasibility study evaluates the viability of the project across technical, economic, operational, legal dimensions, and feasibility matrix to determine whether it aligns with stakeholder needs and delivers measurable value.

2.1. Purpose

- This feasibility study evaluates a comprehensive system redesign that will assess the technical and operational viability of the Cargo Connect platform and validate the economic benefits and return on investment.
- It also provides decision-makers with actionable insights for project implementation.

2.2. Scope

- The system will optimize freight truck operations by integrating real-time tracking, dynamic routing, and airport coordination.
- It will provide truck operators with arrival predictions, optimized routes, and designated entry, dock, and parking assignments.
- The system will interact with existing warehouses to exchange scheduling, dock, and parking data.
- It will also ensure seamless departure by providing optimized exit routes and real time tracking. A unified platform will enhance operational efficiency, reducing congestion and delays.

3. Background

The proposed system builds upon previous implementations by addressing challenges related to real-time decision-making, seamless data integration, and optimized routing. It aims to enhance coordination between freight trucks, airport authorities, and logistics operators, ensuring smoother and more efficient cargo movement while minimizing congestion and delays.

3.1. Previous Implementations

Connected Eco-Driving Technology for Freight Trucks (California, USA)

- Used real-time traffic signal data to optimize truck speeds and reduce unnecessary stops.
- Achieved fuel savings of up to 20% under cold start conditions and a 10% reduction in CO_2 emissions.
- Improved traffic flow, reduced brake wear, and lowered noise pollution.

Logistics Companies and Urban Congestion (New York City, USA)

- Major logistics companies like FedEx, UPS, and USPS faced significant delays due to urban congestion and parking issues.
- Companies incurred millions in parking violation fines due to the lack of optimized freight movement systems.
- Highlighted the need for intelligent logistics routing and real-time parking allocation to reduce delays and costs.

3.2. Key Terms

- DALI System: A system that provides real-time optimized suggestions for freight trucks, integrating live data from traffic and logistics sources.
- Real-Time Tracking: Continuous monitoring of freight truck movements to enhance logistics efficiency.
- Dynamic Dock & Parking Allocation: Automated assignment of docking and parking spaces based on real-time airport and traffic conditions.
- Automated Routing: Navigation that dynamically adjusts routes based on congestion and delivery priorities.
- Congestion Management: Strategies to minimize traffic bottlenecks and improve freight flow.
- Decision Support Dashboard: A centralized interface for administrators to monitor freight operations and make data-driven decisions.

4. Alternatives

	FNU Anuja	Smit Patel	Bradely	Chijioke
Description	Predictive models for cargo delivery scheduling. It needs more time than allotted and can face training complexities. Hence not chosen.	Camera-based parking spot detection. It is costly and not feasible everywhere. Hence not chosen.	Sensor-based parking spot detection. It comes with higher maintenance but relatively cheaper as compared to	Integration of RFID technology for tracking. The entry to the airport is directly feasible by DALI integration. Hence not needed.

			Camera. Hence selected.	
Economic Feasibility	High initial cost	Costly hardware, not scalable.	Moderate cost, higher maintenance expenses.	Affordable but redundant with DALI capabilities.
Technical Feasibility	Complex, requires advanced domain specific expertise.	Feasible but needs robust camera systems for image processing.	Simpler, reliable with existing sensors.	Straightforward, overlaps with DALI functions.
Operational Feasibility	Model performance cannot be guaranteed.	Limited by environmental factors like weather.	Practical, aligns with current operations.	Feasible but unnecessary given DALI integration.
Schedule Feasibility	Since project time is limited to 3 months, understanding and implementing all required models would be difficult	Feasible with the schedule.	Feasible with the schedule	Rapid rollout, minimal additional effort.
Legal Feasibility	Compliant with privacy laws, needs validation.	Requires camera regulation compliance.	Minimal legal hurdles, standard tech.	Must ensure RFID meets aviation standards.

5. System Description

5.1. Challenges

The Cargo Connect system will tackle Freight management issues like fragmented data ecosystems where data is scattered across carriers, airports, and logistics providers which causes miscommunication and makes the process slow. Traffic congestion from peak-hour bottlenecks delays trucks, raising costs and disrupting schedules. Inefficient loading and unloading at the bay area due to poor scheduling, slows operations and clogs docks. Real-time rescheduling struggles with sudden changes, like flight shifts, without instant data access. Providing parking spots is a headache, as limited spaces force trucks to circle, adding to delays. Dynamic route updates via the DALI system.

5.2. Features

- Feature 1: Real Time flight and cargo updates

- This feature focuses on monitoring cargo arrivals, likely with real-time updates. Technically feasible by integrating with airport systems for flight schedules.

- Feature 2: Real time truck tracking
  - Feasible with real-time tracking from mobile system to DALI.
- Feature 3: Scheduling the assignments from the cargo provider.
  - Feasible with integrating the system with cargo providers.
- Feature 4: Integrated Data Management Platform
  - Technically feasible with modern tools and APIs to integrate data sources and make data driven decisions.
- Feature 5: Mobile Interface for System Access
  - Feasible with standard app development tools for truck operators.

5.3. Requirements

- **Freight Trucks**

  The system shall display the following information specific to the truck operator:
  - The estimated and expected arrival time to the airport if not at the airport.
  - Real-time information from DALI about the optimised route to the airport.
  - Which terminal should be used to enter the airport?
  - Assigned dock and time of loading.
  - Designated parking area.
  - Real-time information from DALI about the route suggestions from the airport to its next destination.
  - Which terminal to exit the airport through?

- **Existing airport systems:**

  The system shall be capable of receiving the following information from the existing airport system:
  - Current location of the truck.

  The system shall be capable of sending the following information to the existing airport system:
  - Arrival time of freight trucks
  - Target destination of freight trucks
  - Delays/ early arrival information.

- **DALI System:**

- The system shall receive the current location of the truck when travelling to the airport.
- The system shall receive the current location of the truck when travelling to the cargo destination.
- The system shall send real-time route suggestions to the mobile app.

6. Cost-Benefit analysis / Economic feasibility

This project is economically feasible. After doing a cost-benefit analysis, we have determined that the upsides outweigh the cons; we also posit that the return on investment will be substantial.

The work required to do things such as integrating with existing systems/databases/providers, authenticating DALI-registered vehicles, and providing dashboards displaying real-time metrics will be formidable.

However, the cost is that airport traffic will remain log-jammed to a significant degree, with delays across the spectrum at a major expense to logistics carriers. In other words, if things stay the way they are, profits will continue to be lost as a direct result of freight inefficiencies at airports.

We believe that the cost of development is worth it because the opportunity to stop revenue from continuing to flow out this way is an important one to seize. Our solution will require many hours of labour in the form of tasks such as project planning and coding, but this downside is outweighed by the immense likelihood of the current problem staying the same without intervention.

The prospect of fewer delays and increased efficiency overall at airports could boost profits significantly in the long term. A few months' worth of investment in software engineering labour could produce returns for years to come if the software is properly maintained over time by the recipient(s), allowing for longstanding benefits such as greater coordination of resources and real-time visibility of data. Upsides such as these would be indispensable to the bottom line of a business in the long run.

7. Evaluation of Technical Task

7.1. Data Operations Platform

- Aircraft Arrival Monitoring and Freight Release Prediction: Feasible by integrating flight tracking data and using complex algorithms to predict freight release times based on the real-time data.
- Priority-Based Cargo Loading: Requires intelligent scheduling and dynamic interaction with cargo handling to ensure efficient loading based on priority, which can be complex in real-time.
- Emergency Handling and Congestion Management: Real time data and complex algorithms can help, but the system needs to act quickly to minimize disruptions.
- Emergency Handling and Congestion Management: Real time data and complex algorithms can help, but the system needs to act quickly to minimize disruptions.
- Dynamic Route Optimization: Feasible with GPS, real-time traffic, and weather data; challenges involve ensuring fast, efficient updates under varying conditions.

7.2. Overall Feasibility

- The technologies involved are available and proven, but combining them into a seamless, real-time, and scalable system for airport logistics presents challenges in data flow, system integration, and optimization. While technically feasible, a strong infrastructure and skilled and organized team will be essential to bring it all together.

8. Operational Impact

Operational feasibility evaluates whether the proposed system will solve business problems effectively and integrate smoothly into daily workflows.

8.1. Workflow Integration

- Unified Dashboard: Consolidates data from cargo companies, customs, and truck operators into a single interface (DALI) that provides role-based access for making better decisions.
- Dynamic Scheduling: Application will dynamically schedule the loading / unloading time slots for the trucks based on the Cargo arrival time, road traffic with the help of the DALI application.

8.2. User Training

- Minimal Learning Curve: The mobile app's intuitive design requires minimum training sessions.
- Advanced Training: Admin panel receives workshops on dashboard analytics and emergency protocols.

8.3. Stakeholder Collaboration

- Airport Authorities: Automated bay assignments reduce congestion at unloading zones, aligning with their goal to maximize terminal throughput.
- Logistics Carriers: Real-time aircraft arrival notifications let carriers adjust schedules preemptively, minimizing idle time.
- Customs: Secure API integration ensures compliance while accelerating clearance for authorized shipments.

9. Legal Ramifications

In terms of copyright infringements, failing to respect parents could result in the forced removal of critical features. This applies to trademarks as well. Proper patent research and trademark clearance must be planned out ahead of time to avoid any future trouble.

As far as privacy concerns go, privacy laws must be a major focus due to the involvement of customer data in the mix. Tracking the movement of drivers and their behavior in real-time also poses potential issues with surveillance. Transparency of data practices will be a critical

component of the project.

9.1. Privacy Concerns (Drivers of DALI-vehicles and Customers' Information)

- Since this project involves vehicles (and potentially customer data), privacy laws must be a major focus. Additionally, tracking drivers' movements and behaviour in real-time raises potential concerns around surveillance, so transparency of data practices will be critical.

9.2. Labor Laws (Local and Federal Regulations)

- Adhering to local and federal labor laws is essential. Work hours, wages, worker benefits, and more must be in compliance with applicable regulations.

9.3. Violation of Pre-existing Contracts and Agreements

- Sometimes, software or data shared under a licensing agreement may not be used for other purposes, or certain territories may be off-limits. Violating these terms could result in legal action, termination of agreements, or financial penalties.

9.4. Environmental Concerns (As We Are Dealing with Moving Vehicles)

- Assessing environmental impact could involve looking at emissions, fuel consumption, and how vehicles contribute to local air quality. Any project of this scale may require compliance with local environmental laws or regulations surrounding noise, emissions, or resource use.

10. Schedule Analysis

Schedule feasibility evaluates if a 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.

- Estimated project duration: The duration of the project is 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 Month 3
  - Testing and Deployment – Month 4

By limiting the requirements to a few, with 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.

Learning curve duration would be a few days in the final month. Learning curve for users to adapt to a new system can be minimised by providing training sessions and designing a user-friendly interface.

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

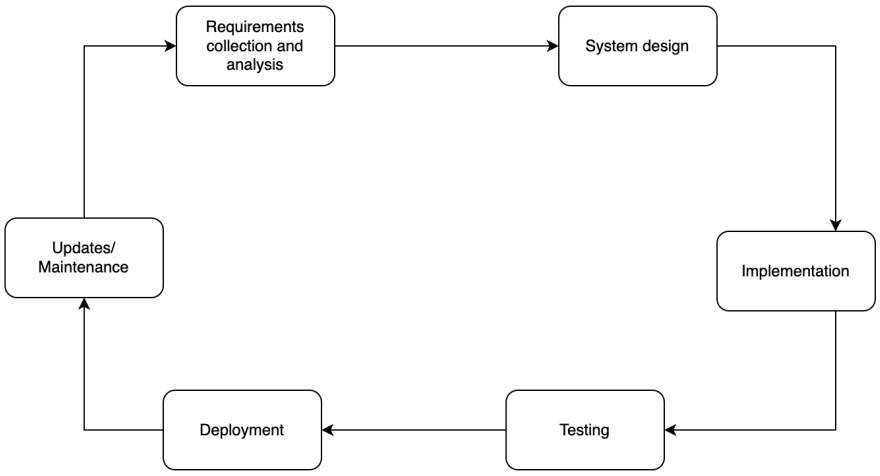


Fig 1: Iterative Software Development Process Model

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.

Cargo Connect: Improving Airport Freight Flow

System Requirements Specification vs 2.0

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

05/09/25

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

Revision History

Version	Date	Description	Authors
1.0	02/26/25	Completed initial draft	BS, PS, AF, CE
2.0	05/09/2025	Final draft updated	BS, PS, AF, CE

Contents

1. Introduction

1.1. Purpose

The purpose of this document is to establish the requirements for the Cargo Connect system.

1.2. Scope

The scope of this document is the functional and nonfunctional requirements as they relate to the system. The objective is to explain as clearly and concisely as possible how the system will meet the needs of stakeholders via fulfilling their specific needs. We aim to establish an understanding between our team and the stakeholders of exactly how our solution will work.

1.3. Overview

Our product will assist in improving the traffic flow of logistics carriers by targeting the bottleneck of airport chaos. By utilizing route optimization, real-time information, and schedule modification, we aim to bring logistics flow at airports to its maximal efficiency as much as possible. Integration with the DALI and Airport system will allow for this to transpire.

2. Overall Description

2.1. Product Perspective

The primary purpose of our product is to maximize traffic flow, reduce logjams, and expedite delivery times for logistics carriers. Through features such as parking optimization, dock assignment, and airport entry routing, we will accomplish this objective. Target users include major logistics carriers across the continental United States. Our product will interact with the DALI system, a traffic optimization solution, and the two will be integrated to form a fully cohesive software product.

2.2. Product Functions

The product should display information to the specific truck operator that includes: arrival time to airport, real-time information from DALI about the up-to-date route to the airport, specific airport entry, assigned dock, loading time, designated parking area, DALI-derived information regarding the route to the next destination, and specific airport exit.

Furthermore, the product should receive information from the logistics company such as cargo scheduling, dock assignment data, and parking assignment data. In terms of sending information, the product should relay to the existing airport system the arrival time of freight trucks, their live GPS location, and their target destination.

Lastly, the product should enforce interaction with the DALI system on the user's behalf on the way to the airport as well as the destination.

2.3. User Characteristics

The characteristics of the intended user base revolve around being a major logistics carrier within the continental United States. The intended user is concerned about efficiency and delivery times, and desires

to make sure everything in terms of traffic flows smoothly at airports. They are obligated to serve their customer base with the fastest delivery possible, and they are in need of a solution to provide enhancement in this regard. They are burdened with the inefficiencies of airports (such as delayed packages in transit) and are presumably willing to try something new in a novel attempt to ameliorate this bottleneck situation.

2.4. Constraints

Our team only has approximately four months for product development and launch, which is a relatively short window, and this is a major constraint. We are limited by privacy laws surrounding user data in terms of what we can do with objectives such as integration with package tracking, real-time oversight of drivers, and GPS location of trucks. Additionally, we cannot control the level of user engagement with the DALI system; drivers can override suggestions with their own free will, potentially limiting the effectiveness of our solution.

2.5. Assumptions and Dependencies

We are beholden to the DALI system for much of the underlying functionality of our product, which constitutes a major dependency. Furthermore, we are dependent on the logistics system to provide scheduling, parking assignment, and dock assignment information; if there are any outages or bugs, this will pose a problem for the ongoing functionality of our product. We are assuming that the end customer will adopt our entire solution and implement it fully, rather than simply taking certain parts and putting them into play. We are also assuming full compliance on the part of the drivers in following directions and suggestions given by the system.

3. Hardware Specification

3.1. Hardware Component 1

We have a physical server for DB and a server for computational tasks.

3.1.1. Functionality

Database Server Stores, retrieves, and manages structured cargo data such as tracking, inventory, schedules, and user information. Computation Server Performs route optimization, cargo allocation algorithms, data analytics, and handles API processing or backend logic.

3.1.2. Operational Requirements

Servers must operate 24/7 with minimal downtime. Must support concurrent access by logistics operators, transporters, and system agents. Database servers must support regular backups, replication (if needed), and transaction integrity. Computation servers must have sufficient compute resources (CPU, RAM) to handle route optimization, ETA calculations, and real-time data processing. Secure network communication between both servers.

3.1.3. QoS Requirements

The CPU will be powerful enough to run algorithms in order to provide the utmost accuracy while filtering out local interference. Correspondingly, the RAM capacity will be robust enough to achieve all of these tasks as well.

3.1.4. Parametric Requirements

Phone/Mobile battery capacity must extend to a lifespan of up to 10 years. RAM should be expandable to accommodate new features and algorithms that come with periodic operating system updates [1].

3.1.5. Design Requirements

Modular architecture separating data persistence and processing logic.

4. External Interface Requirements

4. 1. User Interfaces

- **Truck Driver Interface**: Android app for real-time updates, including arrival times, optimized routes suggestions via DALI, airport entry points, dock assignments, parking availability, and loading schedules.
- **Truck Manager Interface**: Web-based dashboard for monitoring truck movements, dock assignments, and real-time data from the airport and DALI systems.
- **Usability**: Simple, fast, and customizable for different airport operations to minimize distractions and improve efficiency.

4. 2. Hardware Interfaces

- **DALI System**: Real-time communication for optimized traffic routing suggestions.
- **Server Hardware**: Cloud-based backend with processing and storage capacity for real-time data.

4. 3. Software Interfaces

- **Airport Management System**: API integration for exchanging data on flight arrival/delays information.
- **DALI System**: Receives traffic and routing data suggestions for optimized paths.
- **API**: Exposes data for integration with external systems (e.g., freight tracking, logistics platforms) using formats like JSON and XML.

4. 4. Communication Protocols and Interfaces

- **HTTPS/REST APIs**: Secure communication between systems.
- **WebSocket/UDP**: Real-time, low-latency updates for truck locations and traffic data.
- **Data Encryption**: Secure, encrypted communication (TLS 1.2 or higher)

5. System Features

5.1. System Feature A

5.1.1. Description

The system will display heads-up information to the truck driver.

5.1.2. Action/result

The driver will be well-informed about decisions to make in terms of navigating the route.

5.1.3. Functional Requirements

1. The system shall show the arrival time to the airport.
2. The system shall show real-time route suggestions from DALI during navigation to the airport
3. The system shall show which entry to enter the airport through.
4. The system shall show the assigned dock and time of loading.
5. The system shall show the designated parking area.
6. The system shall show real-time information from DALI about the optimized route to the next destination.
7. The system shall show the ideal airport exit for the route.

5.1.4. NFR

The system shall continuously display accurate information via constant synchronization.

5.2. System Feature B

5.2.1. Description

The system will send information to and receive information from the existing airport system.

5.2.2. Action/result

The system will be constantly up-to-date in terms of routing information.

5.2.3. Functional Requirements

1. The system shall receive airplane and cargo scheduling information from the airport system.
2. The system shall receive dock and parking assignment data from the airport system.
3. The system shall send arrival time of freight trucks to the airport system.
4. The system shall send the live GPS location of active freight trucks to the airport system.
5. The system shall send the target destination of freight trucks to the airport system.

5.2.4. NFR

The system shall maintain accurate and timely information via integration with the airport system API.

5.3. System Feature C

5.3.1. Description

The system will be integrated with the DALI system.

5.3.2. Action/result

The driver will have real-time and up-to-date navigation information at his/her fingertips.

5.3.3. Functional Requirements

1. The system shall engage the user with the DALI system while traveling to the airport.
2. The system shall engage the user with the DALI system upon exit from the airport.

5.3.4. NFR

The system shall keep a consistent and reliable connection to DALI with absolutely minimal downtime.

6. Non-Functional Requirements

6.1. Product NFR

1. The system shall circumvent service interruptions using backup servers and load balancing.
2. The system shall use encryption to store sensitive user data.
3. The system shall have an intuitive user interface for freight truck drivers to use.
4. The system shall support integrations with third-party service including airport systems.
5. The system shall respond with traffic suggestions within 5 seconds of receiving current location
6. The system shall allow cargo carriers to enter cargo schedule details

6.2. Process NFR

1. The system shall support the JSON data format for proper communication with external systems.
2. The system shall accurately process data, without loss to ensure the reliability of decisions.
3. The system shall ensure a maximum of 100ms latency when interacting with real-time data

Appendix A: Glossary

Term	Definition
Airport system	The existing infrastructure at the airport that manages scheduling, dock assignments, parking assignments, and other logistics-related data.
API	A set of protocols and tools that allow different software systems to communicate with each other, such as Cargo Connect and airport management systems.
Cargo Scheduling	The process of managing and organizing the loading and unloading of cargo at an airport to ensure efficiency.
DALI System	A traffic optimization system that provides real-time route recommendations for freight trucks traveling to and from the airport.
Dock assignment	The process of allocating specific docks at the airport for freight trucks to load and unload cargo.
Freight trucks	A logistics vehicle responsible for transporting cargo to and from the airport.
NFR	Requirements that define system attributes such as performance, security, reliability, and scalability rather than specific functionalities.
Real time data	Live, continuously updated information regarding truck locations, scheduling, and routing to ensure operational efficiency.
Route optimization	The process of determining the most efficient path for freight trucks to take to minimize delays and maximize fuel economy.
Software interface	The means by which different software components interact, including user interfaces and API integrations.

System features	The specific capabilities of the Cargo Connect system, such as real-time navigation, airport system integration, and parking management.
System constraints	Limitations affecting the project, such as development time, legal restrictions, and driver compliance.
Scalability	The ability of the system to expand by adding more servers or features to accommodate increased demand.

Appendix B: References

[1] *What is Parametric Design in Requirements Engineering? – Valispace*. (n.d.). https://www.valispace.com/what-is-parametric-design-in-requirements-engineering/

[2] *PlacePod*. (2024, November 22). PlacePod. https://placepod.com/

Cargo Connect: Improving Airport Freight Flow

Requirements Analysis Vs 3.0

SE 6387 Advanced Software Engineering Project
R.Z. Wenkstern

Date 05/09/2025

Group 1
FNU ANUJA
SMIT SONESHBHAI PATEL
BRADLEY EVAN STOVER
CHIJIOKE ELISHA-WIGWE

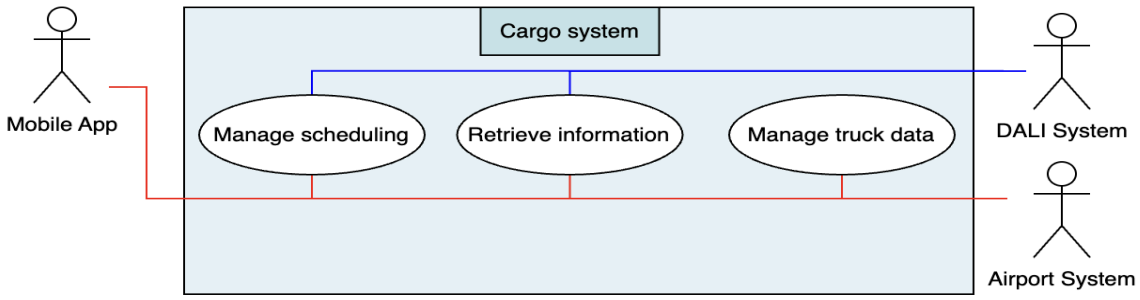
Revision History

Version	Date	Decryption	Authors
1.0	03/10/25	Completed initial draft	AF, SP, BS, CE
2.0	03/25/25	Modified the draft	AF, SP, BS, CE
3.0	05/09/25	Updated the final document	

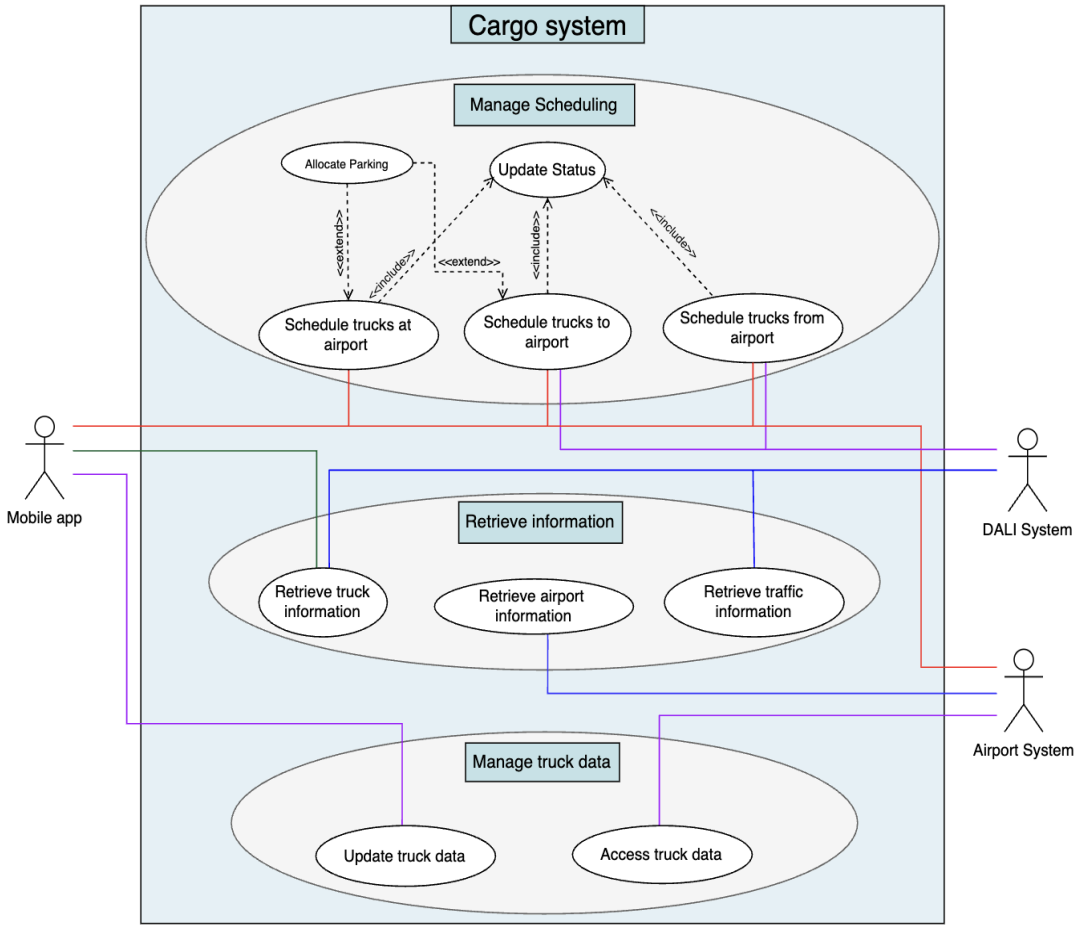
Contents

1. Use Case Diagrams

1.1 Level 0 UCD



1.2 Level 1 UCD



2. Use Case Specification in Brief Format

Update Truck Data	
ID: UC-01	**Type:** base
Primary Actor: Mobile App	
Secondary Actor: Airport System	
Brief Description: - The Mobile App sends the truck update information to the system. - The Airport system updates the information about the truck	

Access Truck Data	
ID: UC-02	**Type:** base
Primary Actor: DALI System	
Secondary Actor: Airport System	
Brief Description: - The DALI system requests information about a particular truck. - The Airport system provides information about the requested truck	

Request Traffic Information	
ID: UC-03	**Type:** base
Primary Actor: Time System	
Secondary Actor: DALI System	
Brief Description: - The system (Time) sends request for traffic information at scheduled time. - The DALI system provides information about the current traffic information	

Retrieve Airport Information	
ID: UC-04	**Type:** base
Primary Actor: Time System	
Secondary Actor: Airport System	
Brief Description: - The system (Time) sends request for airport information at request time. - The Airport System provides information about the current airport status	

Schedule Trucks at Airport	
ID: UC-05	**Type:** base
Primary Actor: Mobile App	
Secondary Actor: Airport System	
Brief Description: The Mobile App contacts the system in order to initiate the scheduling process. The Airport system responds to the scheduling request with either a confirm or a deny.	

Schedule Trucks to Airport

ID: UC-06	**Type:** base
Primary Actor: Mobile App	
Secondary Actors: Airport System, DALI System	
Brief Description: The Mobile App makes a request to the DALI system to schedule a truck going to the airport. The DALI system contacts the airport system, which confirms or denies the request based on preexisting availability.	

Schedule Trucks from Airport	
ID: UC-07	**Type:** base
Primary Actor: Mobile App	
Secondary Actors: Airport System, DALI System	
Brief Description: The Mobile App makes a request to the DALI system to schedule a truck going from the airport. The DALI system contacts the airport system, which confirms or denies the request based on whether it can fit into the existing schedule.	

Retrieve Truck Information	
ID: UC-08	**Type:** base
Primary Actor: Mobile App	
Secondary Actor: DALI System	
Brief Description: The Mobile App requests the status of a particular logistics vehicle to the DALI system. The DALI system responds with an up-to-date status of the truck.	

Update Status	
ID: UC-09	**Type:** Include
Primary Actor: Mobile App	
Secondary Actor: Airport System, DALI System	
Brief Description: This use case ensures real-time updates of truck scheduling statuses, dock assignments, and route information. It is executed every time trucks are scheduled to, at, or from the airport to maintain accurate synchronization among all involved systems.	

Allocate Parking	
ID: UC-10	**Type:** Extends
Primary Actor: Mobile App	
Secondary Actor: Airport System, Mobile App	
Brief Description:	

This use case allocates trucks temporarily to a parking lot when docks are congested**,** or Cargo is delayed.

3. Use Case Prioritization

High Priority: Manage Scheduling

- This is the most crucial use case as it directly impacts how freight trucks operate within and around the airport, involving scheduling, route optimization, and dock assignment. It has a high dependency on airport systems and regulatory requirements.
- It is essential for truck movement within the airport, determining the correct entry point, assigned dock, and parking area. Moderate complexity but high impact.
- It defines optimized routes from external locations to the airport, leveraging DALI for navigation. High business value but slightly less user impact than scheduling at the airport.
- It also ensures trucks exit the airport through the correct route. It has a significant impact on logistics but slightly lower impact than intra-airport scheduling.

Medium Priority: Manage Truck data.

- It involves updating and accessing truck data, which is important for system accuracy but does not directly affect scheduling in real time.
- It allows retrieval of stored truck data but has minimal real-time impact on scheduling operations.

Low Priority: Retrieve information.

- It supports decision-making but does not directly impact truck scheduling. Its dependency on external systems (airport, DALI) increases complexity.
- It retrieves airplane and cargo scheduling details, which indirectly help scheduling but are not directly required for truck operations.

Priority	Use Case	Justification
High	Manage Scheduling (Schedule trucks at airport, Schedule trucks to airport, Schedule trucks from airport)	It has the highest priority since it has a direct impact on truck scheduling, airport coordination, and DALI system integration.
Medium	Manage truck data	It ensures truck records are up to date but does not directly impact scheduling.
Low	Retrieve information	It mainly supports other processes rather than directly influencing scheduling or execution.

4. Use Case 1 Specification in Fully Dressed Format

4.1. Use Case 1

Use Case – Schedule trucks to Airport	
ID: UC-05	**Type:** Base

Brief Description: This use case manages the scheduling of trucks going to the airport. It includes assigning appropriate docks based on real-time dock availability, cargo loading/unloading assignments, and optionally managing parking lot assignments when docks are congested.

Primary Actors: Mobile App

Secondary Actors: DALI System, Airport System, GoogleMaps

Preconditions:

- The truck is registered and authorized within the system.
- Cargo details (type, priority, and loading/unloading requirements) are available.
- Real-time dock availability and parking status information are accessible.

Main Flow:

[include: AuthenticateUser]

1. The MobileApp requests the cargo schedule list.
2. The system returns the list of cargo items to the MobileApp.
3. The user selects an item from the list
4. The MobileApp opens the CargoDetail screen.
5. The Mobile App requests the cargo details from the backend
6. The system returns the cargo details to the MobileApp
7. The user triggers the button to start navigation
8. The Mobile App begins navigation to the cargo's destination using Google Maps.
9. While navigating to the airport (loop every X seconds):
   - 9.1. The MobileApp queries GoogleMaps for the current location.
   - 9.2. GoogleMaps responds with the current location
   - 9.3. The MobileApp sends the current location to the DALI System
   - 9.4. DALI system which responds with traffic alert suggestions.
   - 9.5. The Mobile App displays any alerts to the driver.
   - 9.6. The MobileApp sends the current location and cargo ID to the Airport System.
10. The MobileApp sends a request for the docking port

[extension: DelayedCargo, ParkingFull, ParkingUnavailable, DockReassignment]

11. The Airport sends the parking/docking information
12. The Mobile displays the docking information to the user

Postconditions:

- The truck is successfully assigned to an optimal dock.
- Docking schedules are accurately updated within the Airport System.
- Truck operators receive prompt scheduling confirmation notifications.

Alternative Flows: Dock Congestion (Parking Lot Assignment):

DelayedCargo

1. The Airport System sends a **CargoDelayed** message to the Mobile App.
2. The Mobile App informs the user about the delay with an appropriate message.

ParkingUnavailable

1. If a suitable dock is not available due to congestion, but parking is available:
2. The Airport System sends an **AssignParkingLot** message to the Mobile App.
   - 2.1. The Mobile App displays the assigned parking lot and notifies the operator.
3. The truck proceeds to the parking area and awaits further instructions.

ParkingFull

1. If both docks and parking lots are unavailable:
   1.1. The Airport System responds with a **ParkingFull** message to the Mobile App.
   1.2. The Mobile App displays the suggestion and guides the driver to a nearby resting area.
2. The truck halts temporarily and remains on standby.

DockReassignment

1. Once a dock becomes available:
   1.1. The Airport System sends a **DockReassignment** message to the MobileApp from the parking lot (or resting area) to the optimal dock.
   1.2. The Mobile App notifies the truck driver immediately and updates navigation instructions.
2. The navigation resumes toward the assigned dock for unloading or loading.

Non-Functional Requirements:

- The system should provide dock availability updates within **5 seconds** of a request.
- The system should support **concurrent scheduling requests** with less than **2%** performance degradation.

Technology and Data Variation List:

- The system must integrate with **Google Maps & Directions API** for location updates.
- The dock assignment system should support **API-based communication** with airport systems.
- Data should be stored in a **distributed database** for high availability and fault tolerance.

Open Issues:

- How to handle dock reassignments in case of last-minute scheduling changes?

Use Case – Schedule trucks from airport	
ID: UC-07	**Type:** Base
Brief Description: This use case manages the scheduling of truck departures from the airport. It optimizes truck exit routes based on cargo destination, cargo priority, real-time traffic conditions provided by the DALI system. It ensures that accurate departure information is communicated to relevant systems.	
Primary Actors: Mobile app	
Secondary Actors: DALI System, Airport System, Google Maps	
Preconditions: - Cargo ID is known - The truck has completed loading/unloading operations at the dock. - Real-time traffic data is accessible from the DALI System. - The user is authenticated and has access to delivery operations	
Main Flow: 1. The user enters a CargoID in the Mobile App. 2. The Mobile App sends a request to retrieve the cargo's destination. 3. The system fetches the destination from the CargoItem database 4. The system returns the destination to the Mobile App. 5. The user triggers the button to start navigation 6. The Mobile App initiates navigation toward the cargo's destination. 7. While navigating (loop every 5 seconds until destination is reached): 7.1. The Mobile App queries Google Maps for the current location. 7.2. Google Maps returns the current GPS coordinates. 7.3. The Mobile App sends the current location update to the DALI system. 7.4. The DALI system returns a traffic alert suggestion (if any). 7.5. The Mobile App processes the alert and optionally notifies the user. 8. The user marks the delivery as complete. 9. The Mobile App sends a delivery status update to the system. 10. The system updates the delivery status in the database. 11. The Mobile App receives confirmation that the status has been updated.	
Postconditions: - The cargo delivery status is successfully updated in the system. - All location and traffic updates during navigation are processed in real-time. - The cargo and delivery updates are confirmed to the operator in a timely fashion - Truck departure schedules are accurately confirmed. - The Airport System maintains updated and accurate departure records. - Truck operators receive timely, optimized route and departure information.	
Alternative Flows: TrafficDelay, GPSSignalLost, DALISystemUnreachable - If location retrieval fails (e.g., due to GPS signal loss), the system retries and optionally notifies the user of degraded tracking. - If significant traffic delays, the system recalculates alternative routes immediately. - The truck operator is promptly notified of route changes via the Mobile App.	

- If the DALI system is unreachable, navigation continues without traffic alerts, and a fallback alert is displayed
Non-Functional Requirements: - The system must ensure exit route recommendations are updated every 10 seconds. - The system should maintain secure data transmission when communicating truck departure details to the airport system. - The system must ensure that location updates and suggestions are processed within 3 seconds of the request. - The system shall support concurrent location tracking for multiple deliveries. - The MobileApp must gracefully handle intermittent connectivity during navigation.
Technology and Data Variation List: - Integration with DALI API to ensure up-to-date traffic intelligence and alert suggestions. - Google Maps API is used for real-time GPS location. - Cargo delivery status is persisted in a distributed backend database.
Open Issues: - How to handle unexpected road closures affecting exit routes? - How to escalate critical traffic delays to airport authorities for resolution? - Should delivery confirmation require geo-fencing validation (e.g., driver must physically reach a predefined delivery zone)? - Should offline delivery completion be allowed and synchronized later?

4.2 <<include>> AuthenticateUser

Use Case: Authenticate User	
ID: UC 09	**Type:** Include
Brief Description: This use case handles the process of authenticating a user into the system using a username and password. It is included in other use cases that require a validated and authorized session before continuing (e.g., scheduling, navigation, delivery updates).	
Primary Actors: Mobile App	
Secondary Actors:	
Preconditions: -	
Main Flow: 1. The user opens the Mobile App, which initializes the session. 2. The Mobile App prompts the user to enter their username and password. 3. The Mobile App sends the credentials to the system. 4. The System validates the credentials: [extension: InvalidUsernamePassword, NetworkErrorTimeout] 5. The System forwards the successful authentication response to the Mobile App.	
Postconditions:	

- On success: The user is authenticated, the session is initialised, and the user may proceed with the intended operation.
- On failure: No further operations requiring authentication are permitted until login succeeds.

Alternative Flows:

InvalidUsernamePassword:

1. The system sends an error message back to the Mobile App.
2. The Mobile App displays an error to the user.
3. The truck driver may retry the login or exit the app.

NetworkErrorTimeout:

1. The Mobile App cannot reach the backend due to a network issue.
2. A timeout or connection error is raised.
3. The user is notified with a message like "Unable to connect. Please check your internet connection."

Non-Functional Requirements:

- Authentication should complete within 2 seconds under normal conditions.
- Credentials must be transmitted over HTTPS to ensure security.
- Support for rate-limiting and account lockout must be enforced to prevent brute-force attacks

Technology and Data Variation List:

- Communication with UserDB to ensure proper authentication and authorization

Open Issues:

- What if the UserDB is not responding?
- What if the Truck driver does not exist?

Appendix A: Glossary

Term	Definition
Mobile App	Application used by truck operators for scheduling and tracking.
DALI System	System managing real-time traffic and route optimization.
Airport System	System managing airport operations, including truck dock assignments.
Dock assignment	Allocating a specific loading/unloading bay for a truck.
Parking allocation	Assigning temporary parking spots to trucks due to congestion.
API integration	The method of connecting systems for seamless data exchange.
System availabilities	The uptime and responsiveness of the Airport and DALI Systems.
Network failure handling	Contingency measures when connectivity issues arise.
Include	An include relationship is a relationship in which one use case (the base use case) includes the functionality of another use case (the inclusion use case). The include relationship supports the reuse of functionality in a use-case model.
Extend	Extend relationship is to specify that one use case (extension) extends the behavior of another use case (base).

Appendix B: References

Larman, C. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition). In *Prentice Hall PTR eBooks*.

Cargo Connect: Improving Airport Freight Flow

High Level Architecture vs 5.0

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

Date: 05/09/2025

Group 1
FNU ANUJA
BRADLEY EVAN STOVER
CHIJIOKE ELISHA-WIGWE
SMIT SONESHBHAI PATEL

Revision History

Version	Date	Decription	Authors
1.0	03/25/2025	Completed initial draft	FA, SP, BS, CE
2.0	04/01/2025	Completed second draft	FA, SP, BS, CE
3.0	04/17/2025	Completed third draft	FA, SP, BS, CE
4.0	04/24/2025	Completed fourth draft	FA, SP, BS, CE
5.0	05/09/2025	Completed final document	FA, SP, BS, CE

Contents

1. Introduction

The Cargo Connect Project seeks to make the transportation of freight from various origins to various destinations easier, with particular focus on scheduling trucks to, from airports and at airports. The system provides end-to-end logistics solutions, such as retrieve information, truck allocation, traffic coordination, and real-time status.

Key Goals:

1. Efficiency: Automate scheduling processes to minimize manual coordination and turnaround time.
2. Scalability: Accommodate growing volumes of trucks and routes without compromising performance.
3. Reliability: Provide continuous operation, even with high loads or local system failure.
4. Integration: Be integrated with external services (e.g., Airport System, DALI) to gain access to real-time data and schedule in a more informed way.

At an architectural level, it adopts a layer-based approach with Presentation, Application, Domain, Data Access, and Infrastructure layers isolating concerns. This maximizes code readability and makes it simpler to extend or change the system.

2. Architectural Factors

2.1. Functional requirements

Manage Scheduling	The system shall schedule trucks to and from the airport, at the airport, including dock assignment and tracking truck status.
Retrieving Information	The system shall pull in necessary information such as truck information, airport information (parking conditions, flight schedule), and road traffic information.
Truck data management	The system shall store and display truck data for reporting, analysis, and real-time decision-making (e.g., new truck addition, status update).
External integration	The system shall integrate with Mobile App (drivers/staff), Time service (for scheduling triggers), DALI System (traffic management and optimized routes), and Airport System (flight arrival estimated data)

2.2. Non-Functional requirements

Performance	The system shall be able to process large numbers of scheduling requests and able to react promptly to real-time events (traffic, truck status).
Scalability	The system shall be able to support a growing number of trucks, routes, airports, and external data feeds without affecting performance.

Availability	The system must be operational with minimal downtime and tolerate partial failures.
Security	The system shall protect sensitive data, verify communication with external API's and implement authentication and authorization
Usability	The system shall provide user friendly interface for all users.

2.3. Constraints

1. Dependence on the existing airport system's API and data accuracy.
2. Potential network latency affecting real-time updates.
3. Limited availability of parking and docking slots during peak hours.
4. Freight truck schedules must align with flight schedules, requiring strict adherence to narrow operational windows, particularly during peak commute hours and delivery deadlines.

3. Architectural Decisions

The architectural decisions for Cargo Connect have been chosen to leverage a lightweight, flexible backend with Python Flask, robust data storage with PostgreSQL, modern Android development using Kotlin and Jetpack Compose, and simplified deployment on Heroku. With JWT for stateless security, JSON for efficient communication. This tech stack offers a scalable, maintainable, and secure freight management solution.

1. **Backend Framework: Python Flask**

   We have planned to use Python Flask as the primary backend framework. Flask is lightweight, flexible, and perfect for building RESTful APIs. It favors quick development and boasts a mature extension ecosystem. It facilitates easier development of a JSON-based API for mobile and web client communication. It also facilitates easier integration with JWT-based authentication and PostgreSQL.

2. **Database: PostgreSQL**

   We are using PostgreSQL as the relational database for storing fundamental freight management data (schedules, truck records, user profiles, etc.). This offers excellent ACID compliance, good support for advanced queries, and scalability. It integrates well with Python with libraries like SQLAlchemy. This indeed ensures data consistency and transactional integrity for critical scheduling operations. It supports high-performance querying and reporting on freight data.

3. **Mobile App: Android (Kotlin + Jetpack Compose)**

   We are using Kotlin as the primary language and Jetpack Compose for UI to develop the Android app. Kotlin is new, expressive, and fully supported for Android development. Jetpack Compose declares UI development easier and results in well-maintained code. It provides a modern, responsive, and stunning user interface for

truckers and staff. This also enables faster UI cycles and a reactive UI that stays synchronized with backend changes.

4. **Hosting: Heroku**

   This helps hosting the backend on Heroku. Heroku simplifies deployment and scaling via a managed platform. It provides add-ons (e.g., for PostgreSQL, logging, monitoring) that simplify operations. It also decreases infrastructure management overhead and allows for quick deployments and allows for automatic scaling and integration with CI/CD pipelines without hassle.

5. **Authentication: JWT**

   We are using JSON Web Tokens (JWT) for stateless authentication and authorization. JWTs provide secure, token-based authentication that is best suited to RESTful APIs. Clients send a JWT in the Authorization header (Bearer <token>) with each request. Thus, making security management easier and reduces overhead on backend resources.

6. **Data Format: JSON**

   We have JSON as the default data format between the backend and mobile clients. JSON is lightweight, human-readable, and natively implemented in Python and Kotlin. It is the default for REST APIs, and easier to integrate across platforms. It ensures smooth communication between the Flask API and the Android app and makes debugging and serialization/deserialization of data easier on both sides.

8. Architectural Considerations Overall

Modular Design: Using Flask blueprints or other patterns to split the API into modules (e.g., scheduling, truck management, notifications).

Layered Architecture:

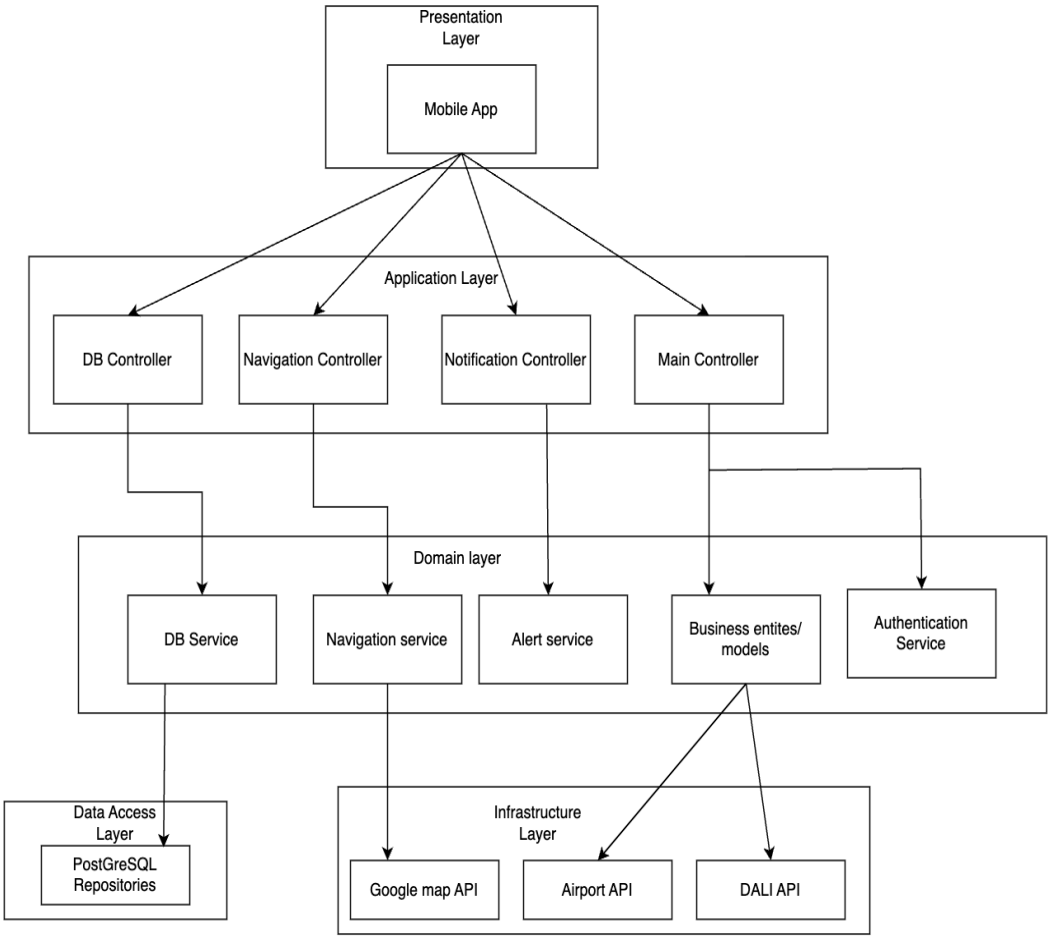
Presentation: UI/UX handled by the Android app (through Jetpack Compose).
Application/Backend API: Python Flask routes receive incoming requests.
Domain/Business Logic: Buried inside Flask services or external modules.
Data Access: PostgreSQL accessed through ORM (e.g., SQLAlchemy) or raw SQL queries.
Infrastructure: Third-party integrations (third-party APIs for traffic/airport data).
CI/CD: Utilize Heroku's pipelines or connect to services like GitHub Actions to perform automatic testing, builds, and deploys.
Security**:** Deploy HTTPS on all endpoints, secure JWT secret handling, and utilize token expiration and refresh to further protect it.

4. Architectural Views

4.1. Logical View

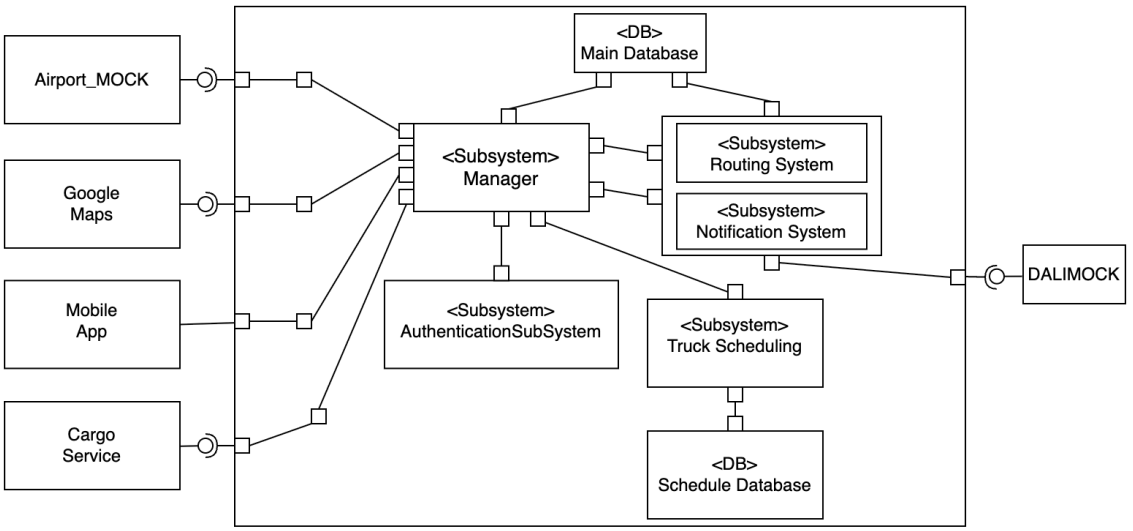
4.1.1. Layered View

Layered perspective is one of the architectures for organizing and visualizing a software system by breaking it into distinct layers or tiers where each layer is used for some specific purpose and typically only depends on the layers that follow it (or is completely independent at times). The idea is to partition the concerns so that each layer deals with some specific aspect of the application, so the system is easier to write, maintain, and extend.



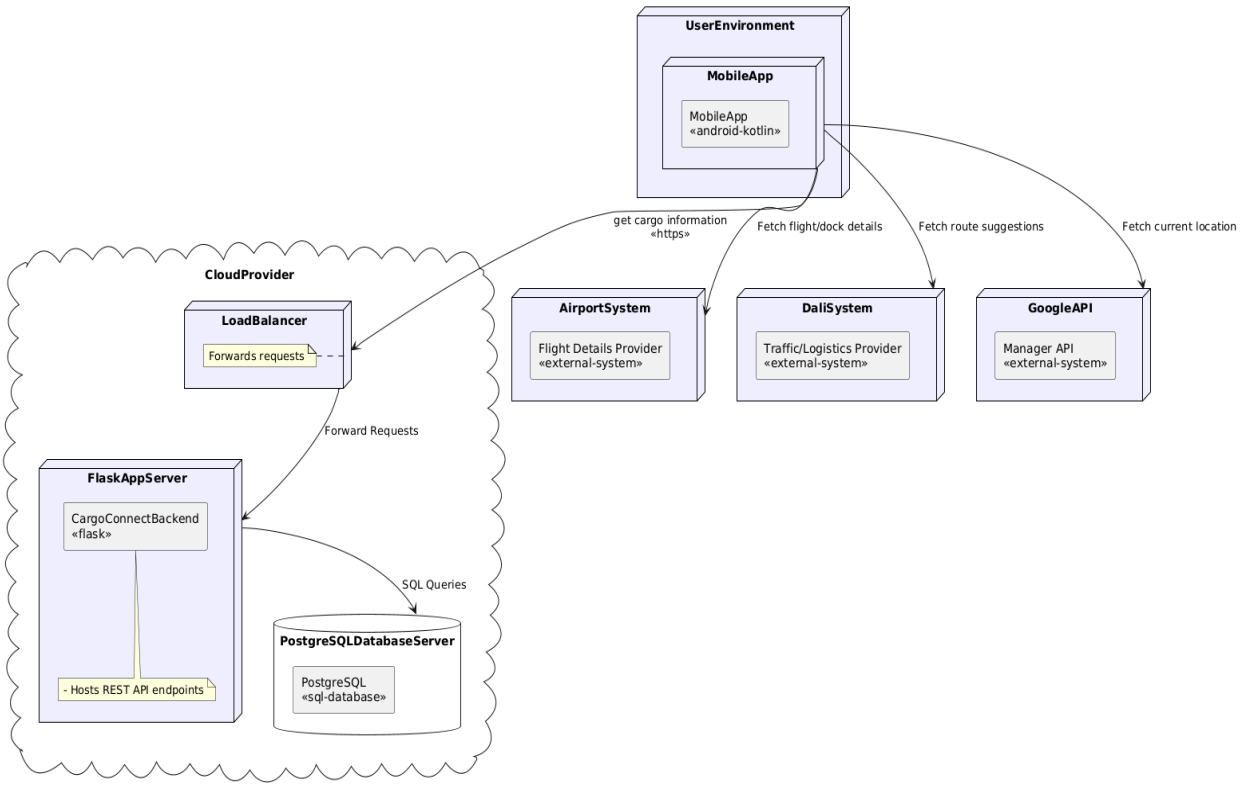
4.1.2. System and Subsystem Component Diagrams

Subsystems are a type of stereotyped component that represent independent, behavioral units in a system. Subsystems are used in class, component, and use-case diagrams to represent large-scale components in the system that you are modeling. We can model an entire system as a hierarchy of subsystems and also define the behavior that each subsystem represents by specifying interfaces to the subsystems and the operations that support the interfaces.



4.3. Deployment View

Deployment diagram is a type of UML diagram that visually represents the physical deployment software artifacts (e.g., programs or databases) onto hardware nodes (e.g., servers or devices), showing the runtime architecture of the system.


UserEnvironment
MobileApp
MobileApp
«android-kotlin»
get cargo information
«https»
Fetch flight/dock details
Fetch route suggestions
Fetch current location
CloudProvider
LoadBalancer
Forwards requests
Forward Requests
FlaskAppServer
CargoConnectBackend
«flask»
- Hosts REST API endpoints
SQL Queries
PostgreSQLDatabaseServer
PostgreSQL
«sql-database»
AirportSystem
Flight Details Provider
«external-system»
DaliSystem
Traffic/Logistics Provider
«external-system»
GoogleAPI
Manager API
«external-system»

Appendix A: Glossary

Term	Definition
Cargo Connect	A freight management solution aimed at improving airport freight flow by automating scheduling, truck allocation, and real-time tracking of freight operations.
API	A set of protocols and tools that allow different software applications to communicate with each other
Python Flask	A lightweight and flexible web framework used for building RESTful APIs. It facilitates rapid development and integration of various services like JWT authentication.
PostgreSQL	An open-source relational database system used to store and manage critical freight management data such as schedules, truck records, and user profiles
Deployment Diagram	A type of UML diagram that illustrates the physical distribution of software components (artifacts) across hardware nodes, showing how the system is deployed in a runtime environment.
Architectural Views	Different perspectives (such as logical, layered, and deployment views) used to analyze and represent the various aspects of the system's structure and behavior
CI/CD	A set of practices and tools designed to automate software integration, testing, and deployment. This ensures that changes are delivered quickly and reliably

Appendix B: References

[1] Requirement analysis

[2] Architectural Views: The State of Practice in Open-Source Software Projects

Cargo Connect: Improving Airport Freight Flow

System Design vs 6.0

SE 6387 Advanced Software Engineering Project
R.Z. Wenkstern

Date : 05/09/2025

Group 1
FNU ANUJA
BRADLEY EVAN STOVER
CHIJIOKE ELISHA-WIGWE
SMIT SONESHBHAI PATEL

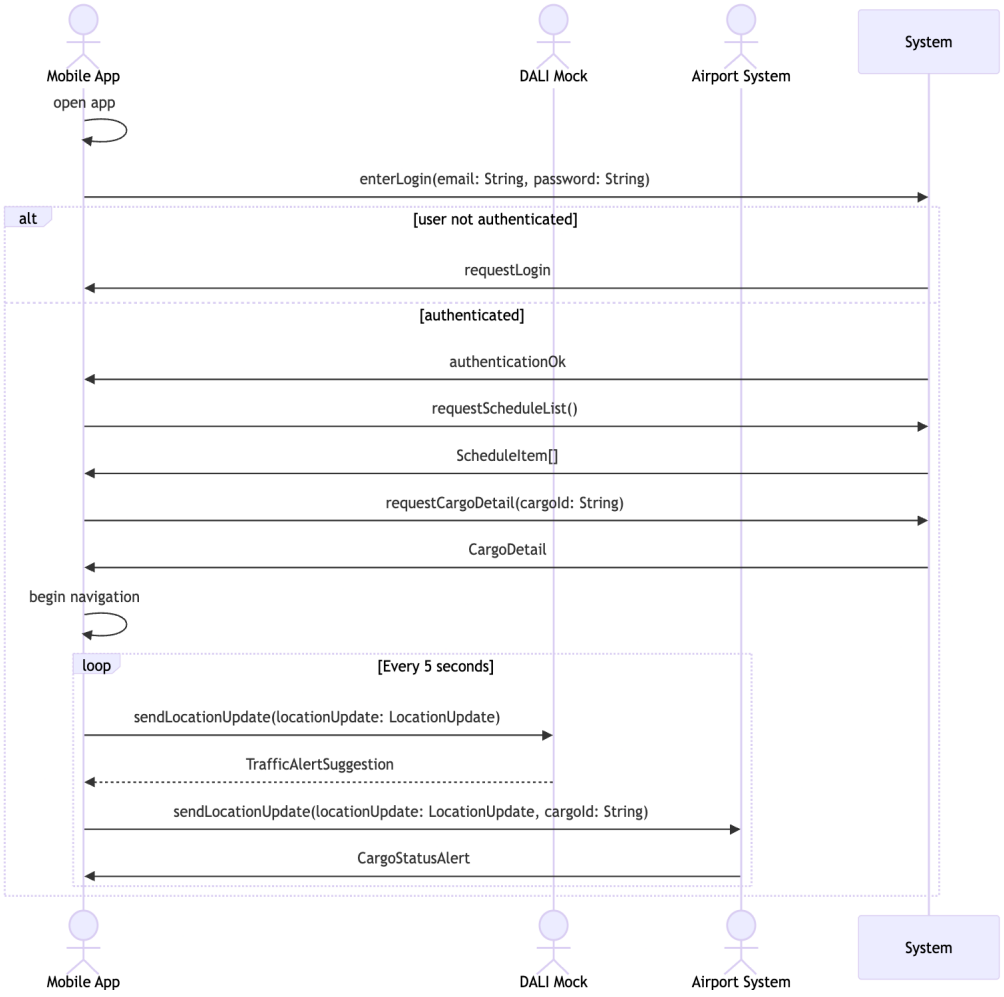
Revision History

Version	Date	Decription	Authors
1.0	03/25/2025	Completed initial draft	FA, CE, SP, BS
2.0	04/01/2025	Completed second draft	FA, CE, SP, BS
3.0	04/17/2025	Completed third draft	FA, CE, SP, BS
4.0	04/24/2025	Completed fourth draft	FA, CE, SP, BS
5.0	04/29/2025	Completed fifth draft	FA, CE, SP, BS
6.0	05/09/2025	Completed final document	FA, CE, SP, BS

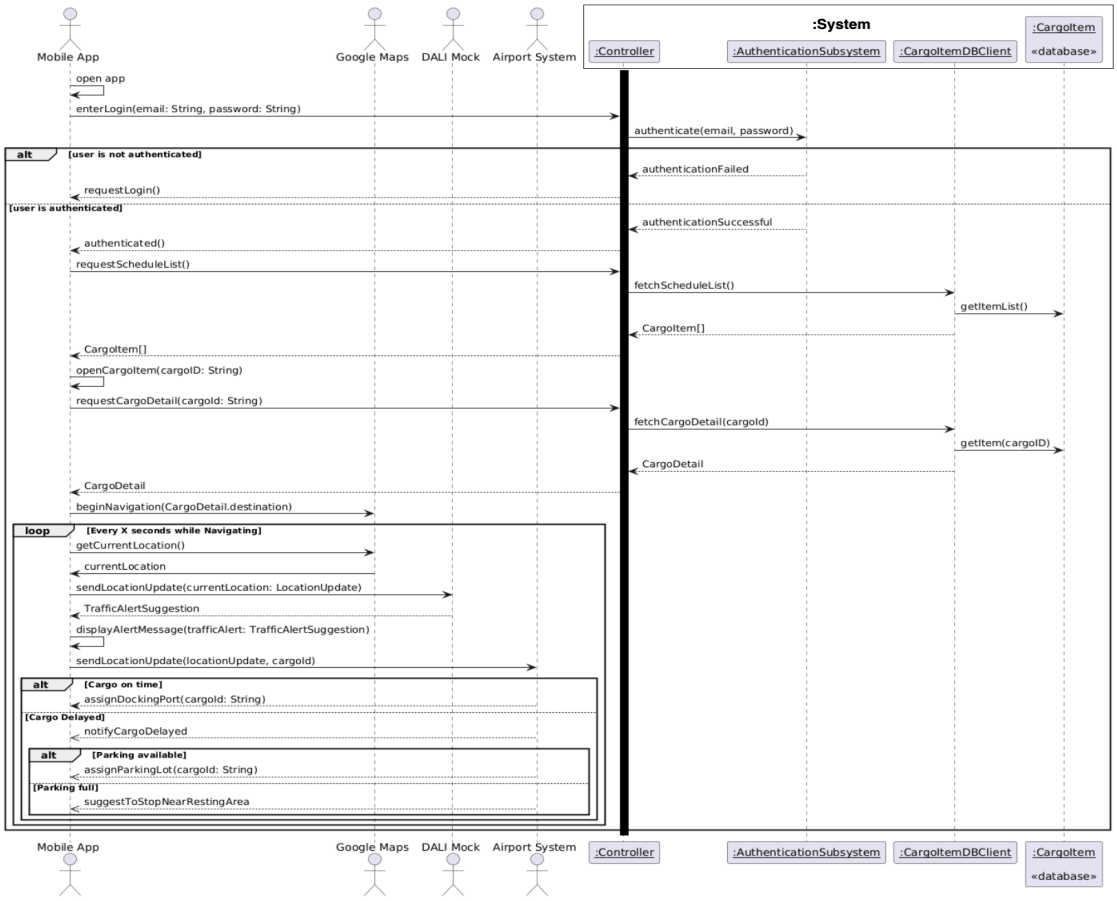
Contents

1. Use Case Realizations – UC1

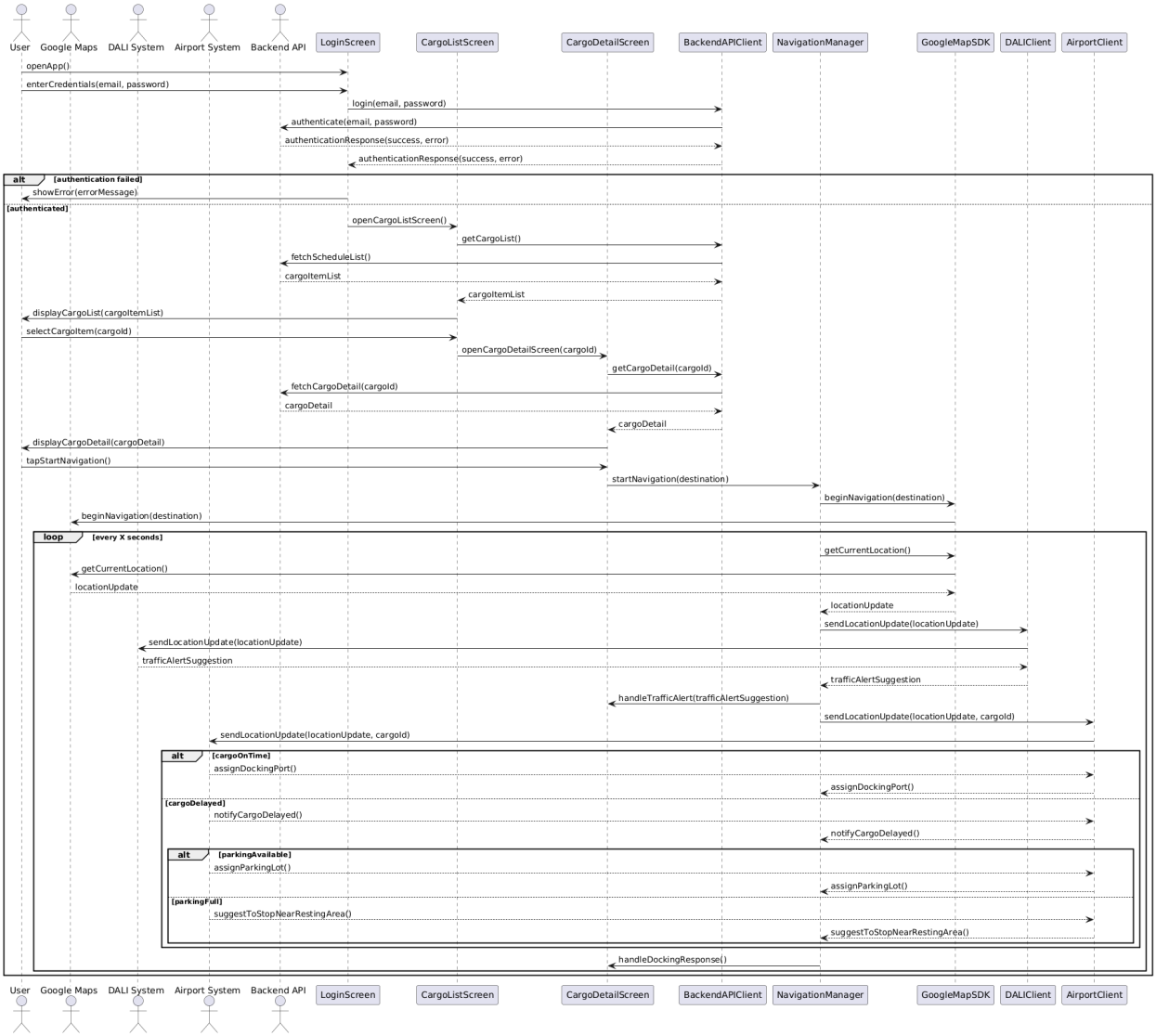
1.1. Black box sequence diagram UC 1



1.2. UC1 UCR - Scheduling trucks to the airport.

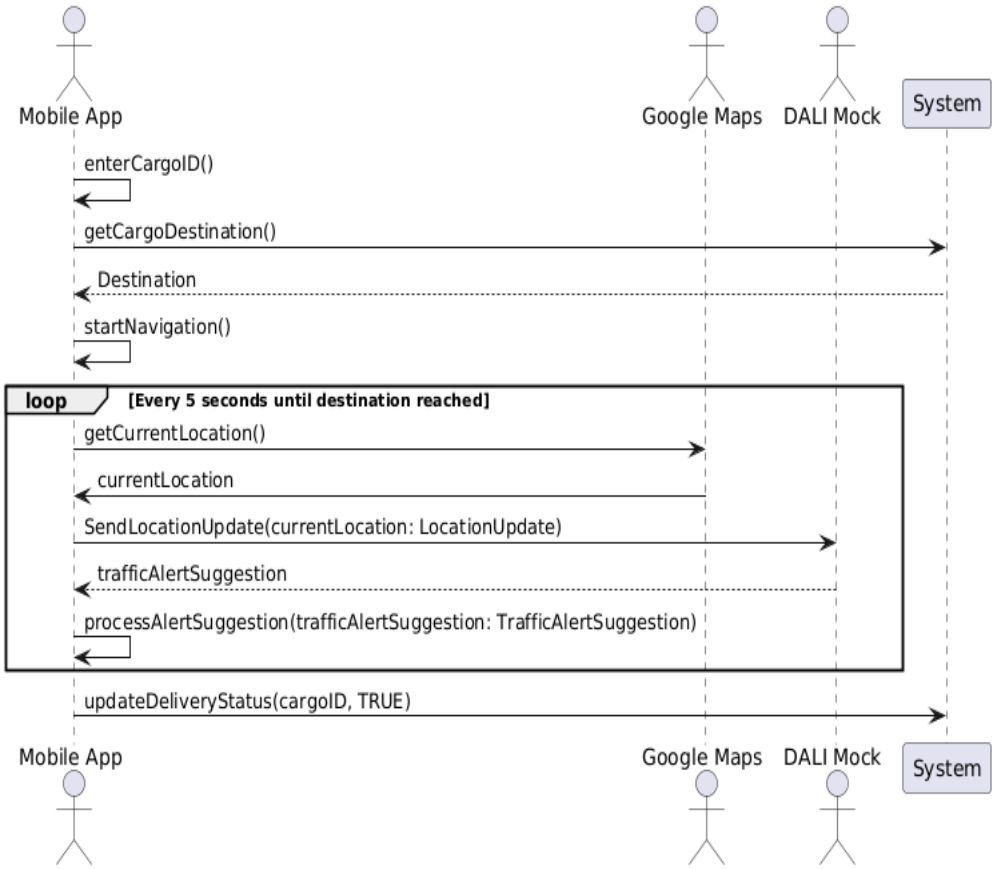


This UCR is focused on Mobile application external services.

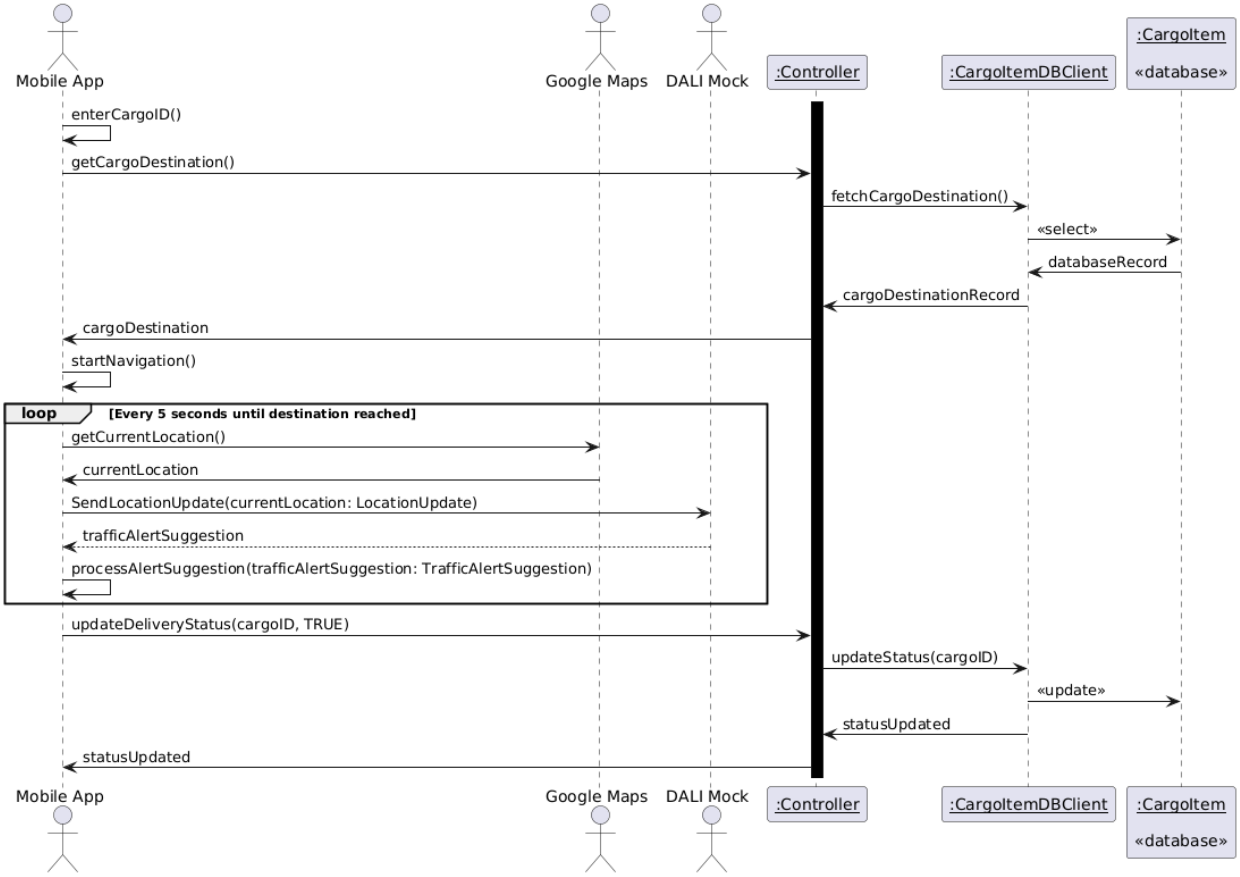


2. Use Case Realizations – UC2

2.1. Black box sequence diagram UC 2

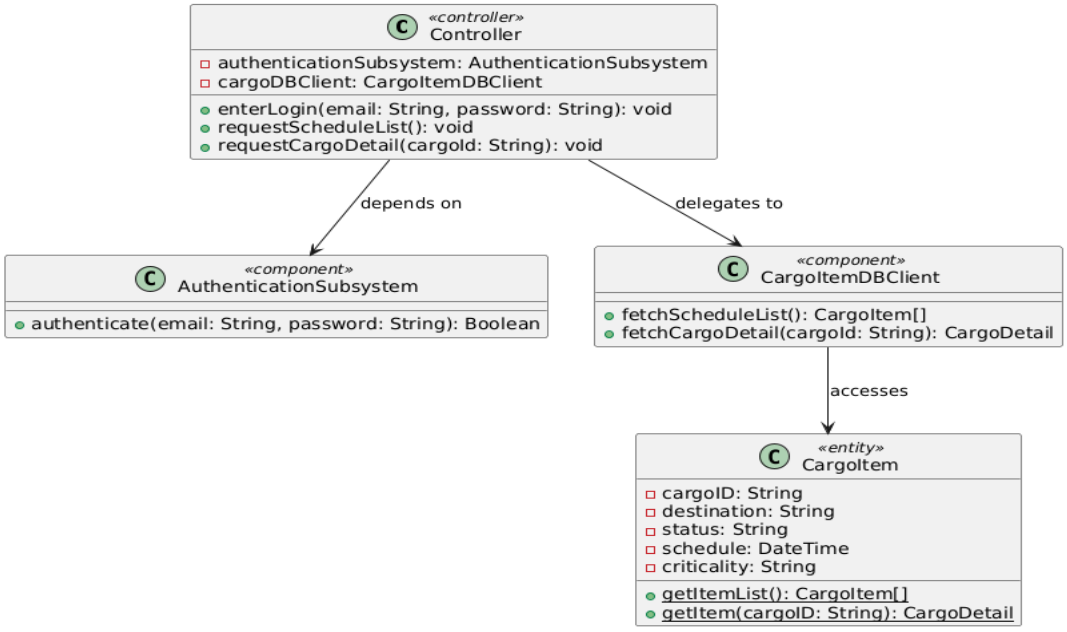


2.2. UC2 UCR - Scheduling trucks from the airport.

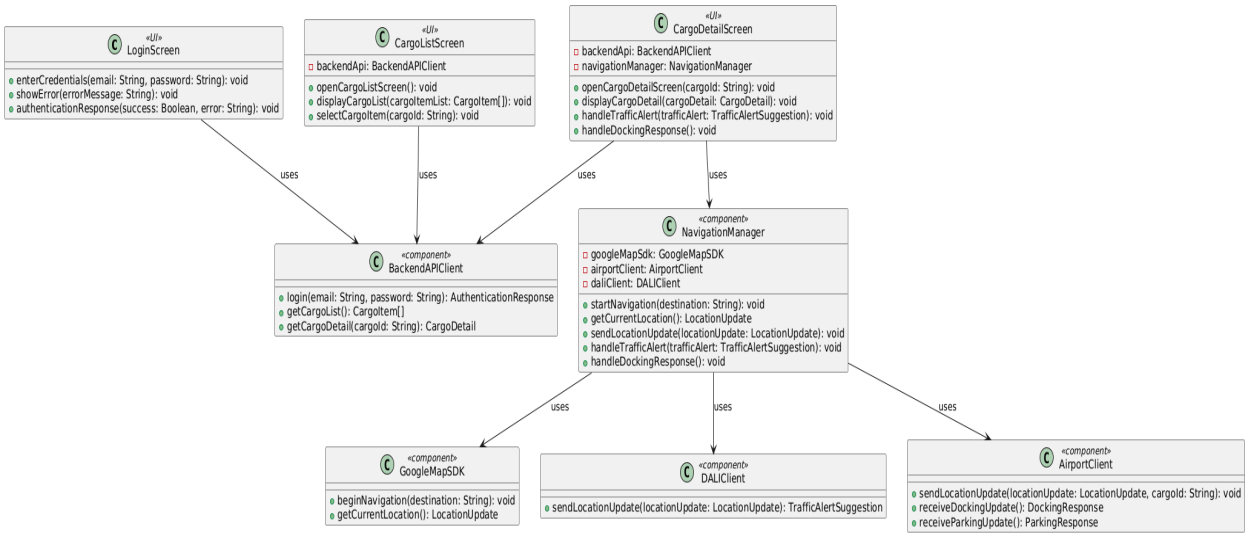


3. Design Class Diagrams

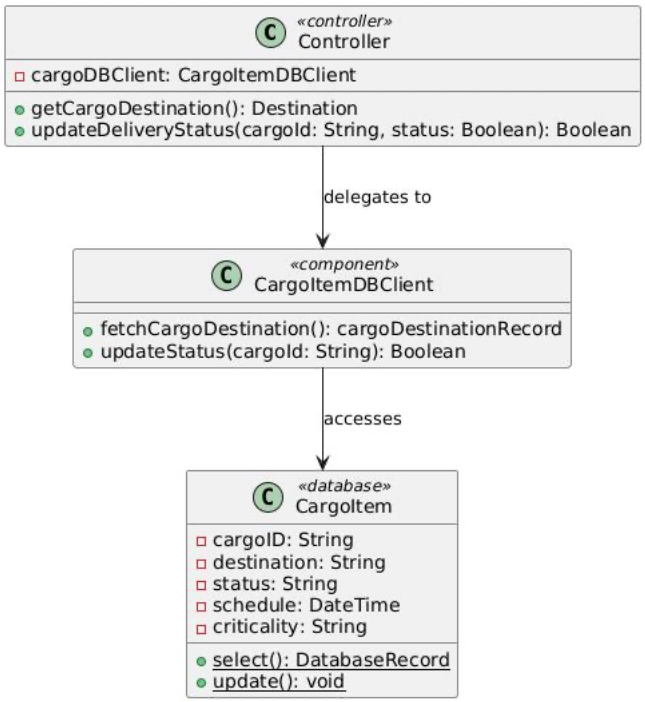
3.1. UC1 DCD : Schedule trucks to the airport



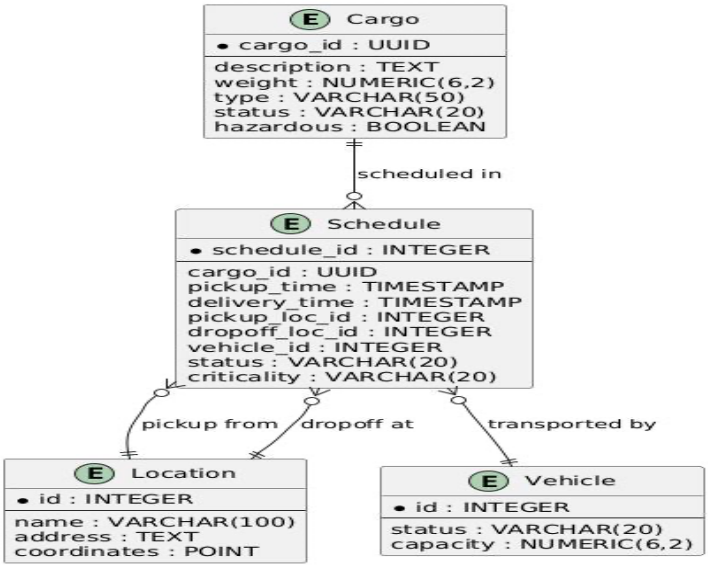
Mobile App DCD :



3.2. UC2 DCD : Schedule trucks from the airport



4. ER Diagram



Appendix A: Glossary

Term	Definition
Cargo Info	Data that includes truck ID, dock location, estimated time of arrival (ETA), and cargo availability status.
DALI	A traffic analysis system providing real-time driving instructions and alerts to ensure efficient navigation for trucks.
Dock	A designated loading or unloading location at the airport for trucks
ETA	The expected time when a truck will reach its designated dock at the airport.
Main System	The core software system responsible for coordinating truck scheduling, managing cargo information, and assigning docks.
Position	A class representing the spatial data of a truck's current location.

Appendix B: References

[1] **https://learn.saylor.org/mod/book/view.php?id=72371&chapterid=67339**

[2]. **https://roshanchi.tripod.com/Documents/Study/OOAD/Notes/DCD.pdf**

Cargo Connect: Improving Airport Freight Flow

Test Documentation vs 2.0

SE 6387 Advanced Software Engineering Project

R.Z. Wenkstern

Date : 05/09/2025

Group *X*
FNU ANUJA
BRADLEY EVAN STOVER
CHIJIOKE ELISHA-WIGWE
SMIT SONESHBHAI PATEL

Revision History

Version	Date	Decription	Authors
1.0	03/25/25	Initial draft completed	FA, CE, BS, SP
2.0	05/09/25	Final draft completed	FA, CE, BS, SP

Contents

1. Organizational Test Process Documentation

1.1 Test Policy

Our goal is to ensure Cargo Connect delivers reliable, scalable, and user-friendly freight scheduling services with minimal downtime. Cargo Connect adopts a quality-driven testing policy that aims to validate all critical functional and non-functional components of the system before deployment. The goal is to ensure automated freight scheduling, mobile interactions, and third-party integrations operate seamlessly.

1.2 Organizational Test Strategy

1. **Unit Testing**: Done for all backend services (Flask) and mobile app modules (Kotlin).
2. **Integration Testing**: Between backend and PostgreSQL, backend, and third-party APIs (DALI, Airport System).
3. **Acceptance Testing**: Based on user stories and UC1 (Schedule trucks to airport).

   Tools: Postman, Android Studio Unit Testing framework, and manual test execution.

2. Test Management Process Documentation

2.1 Test Plan (including a Test Strategy)

The testing scope for Cargo Connect encompasses end-to-end validation of all critical system functionalities and interactions:

- Scheduling: Ensuring trucks are scheduled to, from, and within airport zones based on cargo availability and dock readiness.
- Data Integration: Validating the flow of data between internal modules and external systems like the Airport System, DALI.
- Mobile Updates: Ensuring the mobile app correctly displays updated truck status, routing changes, and dock reassignments in real time.
- Traffic Rerouting: Simulating various traffic conditions and ensuring DALI integration provides intelligent, adaptive routing suggestions.

Test Types

1. **Unit Testing**

- Conducted on individual modules like scheduling logic, JWT token validation, database access, and mobile app components.
- Tools: Pytest for Flask backend, built-in Android testing tools for Kotlin code.

2. **Integration Testing**

- Tests communication between:
    - Backend ↔ PostgreSQL
    - Backend ↔ Airport APIs
    - Backend ↔ Google APIs
    - Mobile App ↔ Backend API

3. **System Testing**

- Validates end-to-end workflows like UC1:
    - Scheduler triggers scheduling
    - Cargo availability is checked.
    - Dock assignment and rerouting if needed.
    - Mobile app receives updates.

4. **Acceptance Testing**

- Tests whether the system meets business requirements and user expectations.

2.2 Test Status Report

- Summary of test progress at any point during the testing lifecycle.
- Number of tests planned, executed, passed/failed, open defects, etc.

2.3 Test Completion Report

The test completion report will summarize:

- All passed test cases (UC1 functionality verified).
- Integration with DALI, and Airport System successful.
- No high/critical issues pending.
- System ready for deployment.

3. Dynamic Test Process Documentation

3.1 Test Design Specification

Designs will validate:

- Correct scheduling based on flight timing.
- Dock assignment and reassignment logic.
- DALI sending instructions behavior.
- Real-time push notifications via OneSignal.
- Mobile app updates.

3.2 Test Case Specification

Test Case 1 : Schedule truck and send the details to the driver via oneSignal.

Test Case 2 : If no cargo available, log the event and no further actions

Test Case 3 : If there is a dock congestion, assign a parking lot. Dock will be reassigned once available.

Test Case 4 : Send traffic notifications to the truck driver.

3.3 Test Procedure Specification

- Start Flask server.
- Trigger scheduling via API.
- Simulate cargo info fetch from Airport System.
- Observe OneSignal notifications.
- Simulate DALI traffic events.
- Verify real-time updates on mobile.

3.4 Test Data Requirements

Effective testing of Cargo Connect requires carefully prepared and representative test data that simulates real-world scenarios. The data should cover all system interactions and edge cases related to truck scheduling, airport conditions, traffic inputs, and mobile updates.

Below is a detailed description of the necessary test data categories:

1. Truck Data

- **Truck ID**: Unique identifiers for each truck used to schedule, track, and log operations.
- **Location**: GPS coordinates or location zones indicating the current position of the truck (e.g., en route, at terminal, near dock).
- **Availability Status**: Whether the truck is currently available for scheduling or in transit/loading.

2. Airport Dock Status

- **Dock ID**: Each dock at the airport has a unique identifier.
- **Status**: Whether the dock is free, occupied, or reserved.
- **Expected Release Time**: If occupied, this indicates when it will become available.

Use Case: This data is used to assign a truck to a dock or reroute it to temporary parking.

3. Flight Schedules

- **Flight Number**: Identifies which flight a truck may be associated with.
- **Arrival/Departure Times**: Required to align freight delivery and pickup accurately.
- **Cargo Ready Time**: Time when freight becomes available for loading/unloading.

Use Case: Flight schedule data is used by the scheduler to coordinate truck timing and avoid delays or congestion.

4. Simulated Traffic Events (DALI Integration)

- **Event Type**: E.g., accident, roadblock, green_light, reroute_needed.
- **Location**: Affected coordinates or area.
- **Severity/Instruction**: Actionable data such as "slow down", "rerouting", or "maintain 40 km/h".

Use Case: This helps validate the DALI system's ability to provide real-time routing instructions to truck drivers.

3.5 Test Data Readiness Report

All test data present in PostgreSQL test schema. Mock services for Airport API and DALI are responsive. All test data including mocks for Airport System APIs are available and yet to be validated. I am using mocked data to run and check my test cases.

3.6 Test Environment Requirements

- Local server for Flask.
- PostgreSQL DB.
- Android emulator/device.
- Internet access for OneSignal and external APIs.
- Postman can be used for manually triggering and testing backend endpoints like scheduling, truck updates, and notifications.
- Mock API Servers - To simulate the behavior of third-party APIs (Airport System) when the actual services are unavailable or unstable.

3.7 Test Environment Readiness Report

Environment setup is complete. Flask app and mobile app are running. All APIs are reachable, and logs configured.

3.8 Actual Results

Actual outputs from test execution were recorded and matched with expected results with few of the test cases. We are still not able to push all kind of notifications from DALI system by tracking real time data from the truck.
Truck schedule details is being sent to the driver and the driver can receive it and start the navigation.

3.9 Test Result

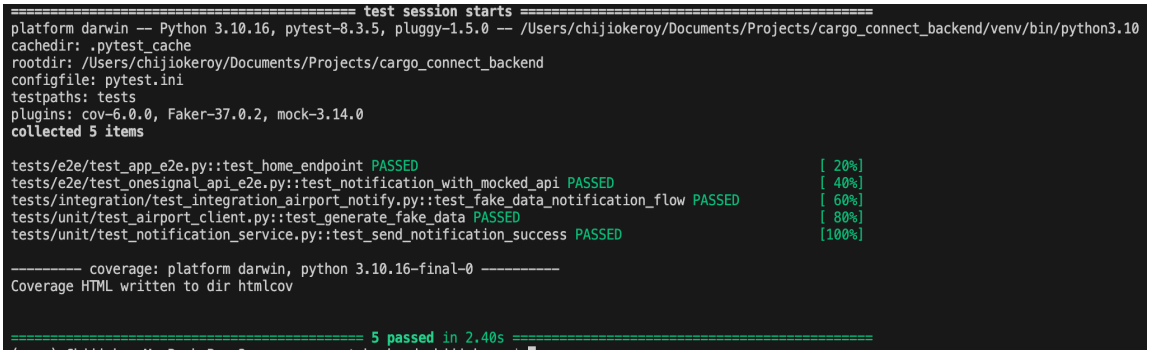
Below are the screenshots of test cases code and results attached.

Few testcases code snippets.

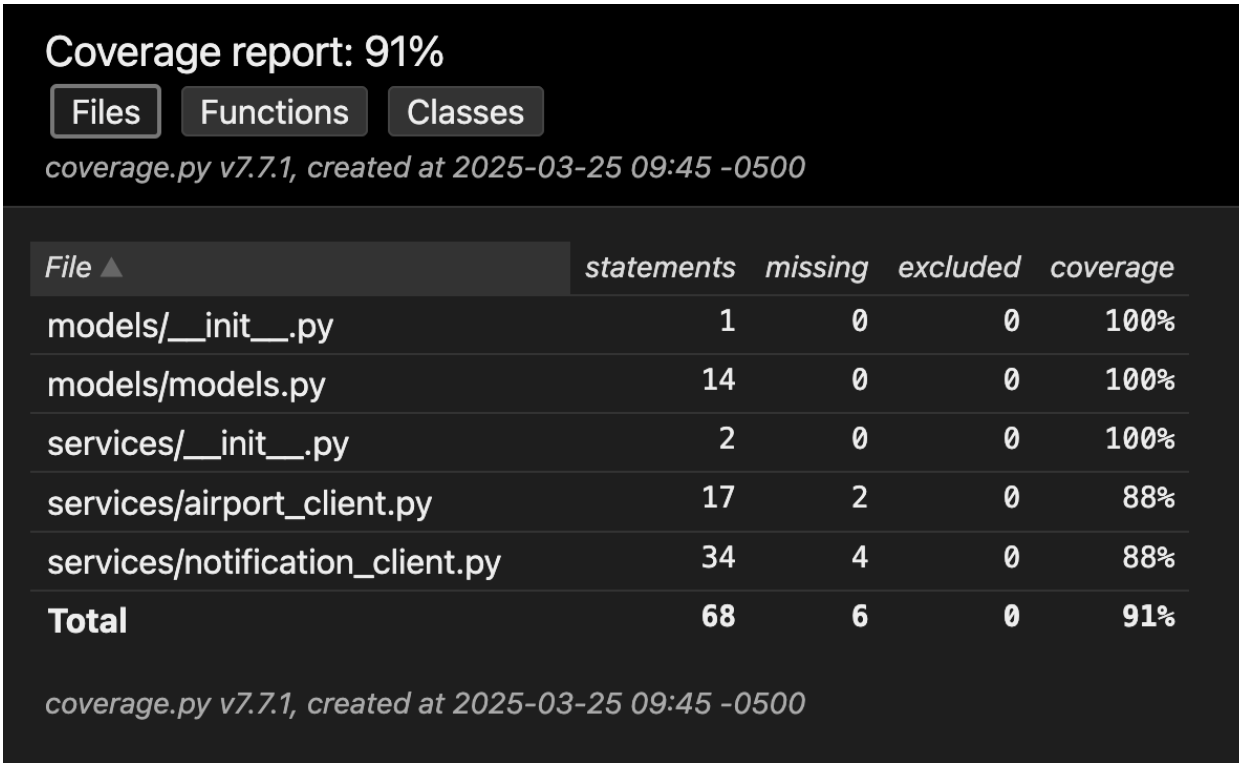
```
@RunWith(MockitoJUnitRunner::class)
class MainActivityTest {

    @Mock
    lateinit var mockContext: Context

    @Mock
    lateinit var mockNotificationClickEvent: INotificationClickEvent

    @Mock
    lateinit var mockPayload: JSONObject

    @Mock
    lateinit var mockIntent: Intent

    @Test
    fun testNotificationClick() {
        // Prepare mock data
        `when`(mockNotificationClickEvent.notification.additionalData).thenReturn(mockPayload)
        `when`(mockPayload.optString("message")).thenReturn("Test Message")
        `when`(mockPayload.optString("latitude")).thenReturn("32.9857")
        `when`(mockPayload.optString("longitude")).thenReturn("96.7502")

        // Initialize MainActivity
        val mainActivity = MainActivity()

        // Simulate the notification click
        mainActivity.onCreate(null)
        mainActivity.openEventHandler.onClick(mockNotificationClickEvent)

        // Verify that the intent is correctly created with the expected parameters
        val expectedUri = "http://maps.google.com/maps?saddr=32.9857,96.7502&daddr=32.9857,96.7502"
        verify(mockContext, times(1)).startActivity(
            argThat { it.action == Intent.ACTION_VIEW && it.data.toString() == expectedUri }
        )
    }
}
```

```
@RunWith(MockitoJUnitRunner::class)
class MainActivityTest {

    @Mock
    lateinit var mockContext: Context

    @Mock
    lateinit var mockNotificationClickEvent: INotificationClickEvent

    @Mock
    lateinit var mockPayload: JSONObject

    @Test
    fun testStartActivityWithCorrectIntentOnNotificationClick() {
        // Set up mock data
        val latitude = "32.9857"
        val longitude = "96.7502"
        `when`(mockNotificationClickEvent.notification.additionalData).thenReturn(mockPayload)
        `when`(mockPayload.optString("latitude")).thenReturn(latitude)
        `when`(mockPayload.optString("longitude")).thenReturn(longitude)

        // Prepare the intent that we want to verify
        val expectedUri = "http://maps.google.com/maps?saddr=32.9857,96.7502&daddr=$latitude,$longitude"

        val mainActivity = MainActivity()
        mainActivity.onCreate(null)
        mainActivity.openEventHandler.onClick(mockNotificationClickEvent)

        // Verify that startActivity was called with the right intent URI
        val argumentCaptor = ArgumentCaptor.forClass(Intent::class.java)
        verify(mockContext).startActivity(argumentCaptor.capture())
        val capturedIntent = argumentCaptor.value
        assertEquals(expectedUri, capturedIntent.data.toString())
    }
}
```

- Results from Unit, Integration, and E2E testing.



- 90%+ test coverage



The first three test cases mentioned above are working well, but the test case where DALI needs to send few contextual information to the app when the driver is on the way in not being shown for now.

3.10 Test Incident Report

1. Notification not received due to delay.

2. Airport System returned malformed data.

3. Triggering the query with periodic time delay was not successful sometimes.

Appendix A: Glossary

Term	Definition
Cargo Info	Includes truck ID, dock, ETA, and cargo availability
DALI	Traffic analysis system used for routing.
Time	Component that triggers time-based scheduling queries.
UC 1	Main use case - Schedule trucks to the airport.

Appendix B: References

- ISO/IEC/IEEE 29119 Test Documentation, 2013
- High Level design document VS 1.0
- Detail design document VS 1.0
- Requirement Analysis VS 2.0

Cargo Connect

Project Closeout Report

SE 6387 Advanced Software Engineering Project
R.Z. Wenkstern

5-10-25

Group 1
Brad Stover
Anuja FNU
Smit Soneshbhai Patel
C.J. Elisha-Wigwe

Revision History

Version	Date	Description	Authors
1.0	5-8-25	Made initial edits to document template	BS

Contents

1. General Information

Provide basic information about the project including: Project Title – The proper name used to identify this project; Project Working Title – The working name or acronym that will be used for the project; Proponent Department/Division – The department/division that will be responsible for the management of the project; Prepared by – The person(s) preparing this document; Date/Control Number – The date the report is finalized and the change or configuration item control number assigned.

Project Title:	Cargo Connect	***Project Working Title:***	Cargo Connect
Proponent Department/Division:	UTD		
Prepared by:	Brad Stover	***Date/ Control Number:***	5-10-25

2. Project Deliverables

List all Project Deliverables and the date each was accepted by the user. Identify any contingencies or conditions related to acceptance.

Deliverable	***Date Accepted***	***Contingencies or Conditions***
PowerPoint Presentation	5-10-25	All parts completed by respective team member(s)
Final Video	5-10-25	Final draft completed
Master Codebase	5-10-25	All code revisions completed
Final Versions of Documentation	5-10-25	All final drafts done
Closeout Document	5-10-25	All sections completed/updated

3. Performance Baseline

Document how the project performed against each Performance Goal established in the Project Plan.

Project Business Objective	***Performance Goal***	***Results***
Get all work done in timely fashion in terms of deliverables	Meet each deadline (1h before minimum)	Done
Create a system that: uses real-time traffic data, makes dynamic routing decisions, helps trucks avoid congestion, and keeps deliveries on time and operations running smoothly	Have all deliverables submitted by May 10th, 2025	Done

4. Cost (Budget) Baseline

State the Planned Cost and Funding for the project, as approved in the Initial Cost Baseline and the Project Charter. State the Actual Cost and Funding at completion. Document and explain all cost and funding variances, including approved changes to the cost baseline.

Expenditures ($000)				
	Planned	***Actual***	***Variance***	***Explanation***
Internal Staff Labor				
Services	$0	$0	$0	
Software Tools	$0	$10/mo	$10	Heroku subscription for hosting
Hardware	$0	$0	$0	
Materials and Supplies	$0	$0	$0	
Facilities	$0	$0	$0	
Telecommunications	$0	$0	$0	
Training	$0	$0	$0	
Contingency (Risk)	$0	$0	$0	

Total	$0	$10/mo	$10/mo	Described above - Heroku

Funding Source ($000)				
	Planned	***Actual***	***Variance***	***Explanation***
General Fund				
Non-General Fund	$0	$0	$0	
Federal	$0	$0	$0	
Other				
Total	$0	$0	$0	No funding necessary

5. Schedule Baseline

Compare the initial approved schedule baseline against the actual completion dates. Enter the planned start and finish dates from the initial schedule baseline. Document all actual start, finish dates, and explain any schedule variances, including approved changes to the schedule baseline

WBS Elements Activity or Task	***Planned Start Date***	***Actual Start Date***	***Planned Finish Date***	***Actual Finish Date***	*Variance*	***Explanation of Variance***
Feasibility Report	2-4-25	2-4-25	2-11-25	2-11-25	None	N/A
Project Plan	2-13-25	2-13-25	2-20-25	2-20-25	None	N/A
SRS	2-19-25	2-19-25	2-26-25	2-26-25	None	N/A
Requirement Analysis	3-4-25	3-4-25	3-11-25	3-11-25	None	N/A
Test Documentation	3-18-25	3-18-25	3-25-25	3-25-25	None	N/A
High-Level Architecture	3-24-25	3-24-25	3-31-25	3-31-25	None	N/A
System Design	3-24-25	3-24-25	3-31-25	3-31-25	None	N/A
UC1	3-24-25	3-24-25	3-31-25	3-31-25	None	N/A
Revised Documentation	4-10-25	4-10-25	4-17-25	4-17-25	None	N/A
Revised Documentation	4-17-25	4-17-25	4-24-25	4-24-25	None	N/A
Revised Documentation	4-22-25	4-22-25	4-29-25	4-29-25	None	N/A
Final Deliverables	5-3-25	5-3-25	5-10-25	5-10-25	None	N/A

6. Scope

Document any changes to the Project Scope and their impact on Performance, Cost, or Schedule Baselines.

Scope Change	***Impact of Scope Change***
Implementing mock DALI system	No effect on cost; number of developers to dedicate to both backend and frontend temporarily decreased while addressing the building of mock DALI system, impacting performance; no effect on schedule
Implementing mock airport system	No effect on cost; number of developers to dedicate to both backend and frontend temporarily decreased while addressing the construction of mock airport system, impacting performance; no effect on schedule

7. Operations and Maintenance

Describe the plan for operation and maintenance of the product, good, or service delivered by the project. State the projected annual cost to operate and maintain the product, good, or service. Identify where and why this projection of cost differs (if it differs) from the Project Proposal. If the operation and maintenance plan is not in place, what is the target date for the plan and what is the impact of not having operations and maintenance for the product, good, or services in place.

7.1 Operations and Maintenance Plan

7.2 Operations and Maintenance Cost

Expenditures ($000)				
	Planned	***Actual***	***Variance***	***Explanation***
Internal Staff Labor				
Services	$0	$0	$0	
Software Tools	$0	$10	$10	Heroku subscription for hosting backend
Hardware	$0	$0	$0	
Materials and Supplies	$0	$0	$0	
Facilities	$0	$0	$0	
Telecommunications	$0	$0	$0	
Training	$0	$0	$0	

Contingency (Risk)	$0	$0	$0	
Total	$0	$10	$10	See Heroku above

Funding Source ($000)				
	Planned	***Actual***	***Variance***	***Explanation***
General Fund				
Non-General Fund	$0	$0	$0	$0
Federal	$0	$0	$0	$0
Other				
Total	$0	$0	$0	None needed

8. Project Resources

List the Resources specified in the Project Plan and used by the project. Identify to whom each resource was transferred and when it was transferred. Account for all project resources utilized by the project.

Resource ***(Describe or name the resource used)***	***Person or Organization Who Received Resource***	***Turnover Date***
Project Team	N/A	N/A
Customer Support	N/A	N/A

Facilities	N/A	N/A
Equipment	N/A	N/A
Software Tools	N/A	N/A
Other	N/A	N/A

9. Project Documentation

Identify all project documentation materials stored in the project library or other repository. Identify the type of media used and the disposition of the project documentation (see Communications Plan).

Report(s) and Document(s)	***Media Used***	***Storage Location***	***Disposition***
Feasibility Report	PDF	UTD Box	Uploaded
Project Plan	PDF	UTD Box	Uploaded
SRS	PDF	UTD Box	Uploaded
Requirement Analysis	PDF	UTD Box	Uploaded
Test Documentation	PDF	UTD Box	Uploaded
High-Level Architecture	PDF	UTD Box	Uploaded
System Design	PDF	UTD Box	Uploaded
UC1	PDF	UTD Box	Uploaded
Feasibility Report	PDF	UTD Box	Uploaded

Project Plan	PDF	UTD Box	Uploaded
SRS	PDF	UTD Box	Uploaded
Requirement Analysis	PDF	UTD Box	Uploaded
Final Deliverables	PDF, PPT, .ZIP files	UTD Box	Uploaded

10. Lessons Learned

Identify Lessons Learned for feedback to the company/organization. Lessons Learned should be stated in terms of Problems (or issues) and Corrective Actions taken. Provide a brief discussion of the problem that identifies its nature, source, and impact. Site any references that provide additional detail. References may include project reports, plans, issue logs, change management documents, and general literature or guidance used that comes from another source.

Statement of Problem	***Discussion***	***References***	***Corrective Actions***
Various bugs in frontend code	Due primarily to two developers working on the frontend		Consulted online resources and each other when necessary to fix bugs
Too many branches for mobile app frontend on Github	Due to hesitancy to overwrite existing code		Decided on one final branch; exchanged working branch names constantly
End-to-end project lifecycle ownership	Due to difficulty of defining clear problem statements and balancing diagrams with documentation	Final PPT	Constantly revised diagrams to match code and rest of documentation
Modular and scalable system design	Due to difficulty of decoupling components	Final PPT	Decoupled components to enable plug-and-play mock systems

Third-party integration and parallelism	Due to difficulty of designing asynchronous workflows	Final PPT	Integrated main system with DALI mock and airport mock
Collaboration and adaptability	Due to not being able to receive feedback on a daily basis on project	Final PPT	Took into account iterative feedback from professor during semester-long demos to improve our system design

11. Dates for Post Implementation Review and Report

Identify the date for completing the post implementation report and the person responsible for this action.

Action	***Date***	***Responsible Person***
Post - Implementation Review	5-10-25	Brad Stover
Post - Implementation Report	5-10-25	Brad Stover

12. Approvals

Position/Title	***Signature/Printed Name/Title***	***Date***

Project Manager	Anuja FNU	5-10-25
Project Sponsor		
Program/Agency Management		

Appendix A: Glossary

Term	Definition
Box	A cloud storage platform used by the University of Texas at Dallas for document storage.
DALI mock system	A third-party logistics API system leveraging Laravel for mock purposes.
GitHub	A web-based platform for version control used to manage source code.
Heroku	A cloud-based platform-as-a-service allowing developers to build, run, and deploy applications.
Jetpack Compose	A modern Android UI toolkit for building native UIs.
Laravel	A PHP framework used to simulate the DALI system in the absence of real access.
PostgreSQL	An open-source relational database used for storing project data.
SRS	Software Requirements Specification; outlines project requirements
UC1	Use Case 1; a specific scenario described and implemented in the project

Appendix B: References

Project closeout report, Virginia Information Technologies Agency

UTD Box. *The University of Texas at Dallas Cloud Storage Service*. Retrieved from https://utdallas.account.box.com