2/9/2026

Project Plan

Alex Wilcox

Alina Zimavaya

Crawford James

Luke Fixari

‘ctrlC+ctrlV’

Table of Contents

Table of Contents

Table of Figures

1. Project Team Name

ctrlC+ctrlV

2. Team Members and Roles

Name	Role	Responsibilities
Alina Zimavaya	Scrum Master	Sprint planning, task tracking, milestone coordination, risk tracking, communication with stakeholders, ensuring deadlines are met
Alex Wilcox	Principal Developer	System architecture, backend development, database integration, performance optimization
Luke Fixari	Quality Assurance Manager	Test plan creation, validation of functional requirements, regression testing, simulation correctness
Crawford James	Documentation Specialist	Requirements documentation, user guides, meeting minutes, formatting consistency, deliverable organization

Table 1: Individual Roles

3. Sponsor/Customer

Panther Cloud Air

4. Work Breakdown

Planning and Coordination

- Sprint planning and milestone tracking
    - Define sprint goals and deliverables for each two-week development cycle
    - Break down major features into actionable tasks with time estimates
    - Track progress using Monday.com Kanban boards

    - Conduct sprint retrospectives to identify improvements
    - Monitor milestone completion against the project schedule
    - Adjust timelines and priorities based on team velocity and blockers

- Meeting scheduling and documentation
    - Schedule bi-weekly team meetings on Wednesdays and Sundays
    - Coordinate with team members to ensure availability and minimize conflicts
    - Prepare meeting agendas with discussion topics and decision points
    - Record meeting minutes including decisions made, action items, and responsibilities
    - Maintain meeting minutes in SharePoint for reference and accountability
    - Document design decisions and rationale in personal engineering journals
- Task assignment and progress monitoring
    - Assign tasks to team members based on expertise and workload balance
    - Set clear deadlines and acceptance criteria for each task
    - Monitor task completion through GitHub commit activity and pull requests
    - Conduct daily async check-ins via iMessage for quick status updates
    - Identify and escalate blockers or dependencies affecting progress
    - Redistribute tasks when team members face unexpected availability issues
    - Maintain transparency through shared GitHub logs and Monday.com boards

Design and Architecture

- Define overall system structure including frontend, backend, and database layers

- Document technology stack decisions (React, Flask, MariaDB, Docker)
- Design API contracts and data flow between components
- Plan for scalability, maintainability, and performance
- Create architecture diagrams showing component interactions

Implementation

- Implement flight timetable generation algorithm
- Apply constraints: airport hours, gate availability, turnaround times, maintenance cycles
- Calculate flight times using great-circle distances and aircraft performance data
- Implement hub-based routing strategy with 3-4 designated hubs
- Handle time zone conversions and daylight saving time
- Validate schedule correctness and feasibility

Testing and Validation

- Functional testing
    - Verify all functional requirements are met
    - Test flight search, schedule generation, and simulation workflows
    - Validate user interface interactions and error handling
    - Ensure database operations perform correctly
    - Test across different browsers and screen sizes

- Performance testing
    - Measure database query response times (target: under 2 seconds)

    - Test system behavior under simulated load
    - Identify and resolve performance bottlenecks
    - Optimize slow queries and resource-intensive operations

- Simulation accuracy validation
    - Verify flight time calculations match expected values
    - Validate disruption probabilities produce realistic results
    - Cross-check passenger flow against demand models
    - Ensure financial calculations are accurate
    - Test edge cases: maintenance conflicts, gate shortages, overnight layovers

- Regression testing
    - Re-test existing functionality after changes or bug fixes
    - Maintain automated test suite for critical features
    - Track and verify bug fixes
    - Ensure new features don't break existing workflows

Documentation and Reporting

- Project plan and requirements
    - Document project scope, goals, and constraints
    - Define functional and non-functional requirements with unique IDs
    - Maintain risk assessment and mitigation strategies
    - Update project plan as scope or timeline changes

- Use case diagrams
    - Create visual representations of system workflows

    - Document user interactions and system responses
    - Maintain consistency between diagrams and implementation

- Environment setup instructions
    - Write setup guide for team members cloning the repository
    - Document prerequisite installations (Node.js, Python, Docker)
    - Provide step-by-step instructions for database initialization
    - Include troubleshooting section for common issues

- Status reports and final documentation
- Prepare weekly status updates for stakeholders
    - Document lessons learned and recommendations
    - Create final project report summarizing objectives, implementation, and results
    - Prepare presentation materials for final demonstration

5. Introduction

Panther Cloud Air is a startup airline seeking to evaluate the feasibility of its business model through a software simulation. The client requires a system capable of generating realistic flight schedules between the top thirty U.S. airports and a daily international route between New York and Paris. The system must model operational constraints such as airport operating hours, aircraft turnaround times, gate availability, maintenance cycles, passenger demand, weather disruptions, and time zone differences.

The simulation must execute two weeks of flight operations, collect operational data in a database, and provide the ability to query performance metrics including passenger counts, operational costs, revenue, and profit. The system is intended for airline staff and administrators only, not passengers.

6. Project Goals

Required:

- Build a flight timetable system that schedules aircraft so passengers can travel between all reachable 31 airports.
- Accurately model flight operations including taxi times, takeoff, climb, cruise, descent, landing, turnaround, refueling, crew changes, and hub constraints.
- Maintain all schedules in a database with clearly defined tables for flights, passengers, aircraft, and airports.
- Simulate two weeks of operations using the provided daily disruption scenarios (weather delays, icing, jet stream effects, cancellations, aircraft failure).
- Provide a language-specific interface, depending on location allowing passengers to search routes from any two airports.
- Generate a final operational report summarizing passengers carried, operating costs, revenue, and profit or loss.

Team Goals:

- User interface needs to be usable by someone with little to no experience.

- Ensure passengers can travel between any two airports using a maximum of one connecting flight.
- Any flight from A to B needs to load in two seconds from the database.
- Implement automated hub selection logic using a specially made algorithm that determines the best hub for profitability and passenger convenience.
- Add multiple performance metric calculators such as percentage for on-time aircraft to better understand the data.

7. Project Scope

Overall System Scope

- Design and implement a software-based airline operations simulation system.
- Simulate airline operations over a defined airport network and fixed time period.
- Emphasize operational realism rather than real-time execution or visualization.
- Focus on producing analyzable outputs rather than live decision-making tools.

Geographic and Network Scope

- Model airline operations across the top 30 U.S. airports.
- Include one daily international route from a selected New York airport to Paris.

- Restrict operations to a fixed, predefined airport list with no expansion.
- Ensure connectivity across all supported airports using direct or connecting flights.

Operational Modeling Scope

- Model aircraft performance characteristics, including:
    - Range
    - Passenger capacity
    - Turnaround time
- Enforce airport constraints, including:
    - Airport capacity limits
    - Hub-based operations
- Simulate operational disruptions over a two-week period, including:
    - Weather delays
    - Icing delays
    - Jet stream effects
    - Cancellations
    - Aircraft failures

Passenger Demand and Routing Scope

- Generate passenger demand based on:
    - Airport population size
    - Hub importance
- Allow authorized users to search for passenger routes between any two airports.

- Limit routing to at most one connection between origin and destination.
- Store and manage all routes and schedules using a database-backed scheduling system.

Data Storage and Persistence Scope

- Store all schedules, routes, and simulation data in a relational database.
- Maintain database tables for:
    - Flights
    - Aircraft
    - Airports
    - Passengers
- Support post-simulation querying and analysis.

Reporting and Outputs Scope

- Generate operational reports, including:
    - Flight schedules
    - Aircraft utilization
    - Disruption statistics
- Generate financial summaries, including:
    - Operating costs
    - Revenue
    - Profit or loss
- Provide data-driven insights designed for analysis rather than real-time use.

User Interface Scope

- Provide a staff-facing user interface for authorized users only.
- Support searching and viewing of:
  - Flight schedules
  - Passenger routes
  - Simulation outcomes
- Prioritize clarity and usability for non-technical users.
- Exclude all passenger-facing functionality.

8. Hardware and Software Requirements

Developer:

Hardware

- Computer capable of running Docker
- Minimum 16 GB RAM recommended
- Stable internet connection
- Multi-core processor
- 15-20 GB of disk space

Software

- Python 3.10+
- Node.js 18+
- Docker Desktop

- MariaDB
- GitHub
- Visual Studio Code
- Web browser (Chrome/Firefox)
- Linux-compatible runtime environment

Client:

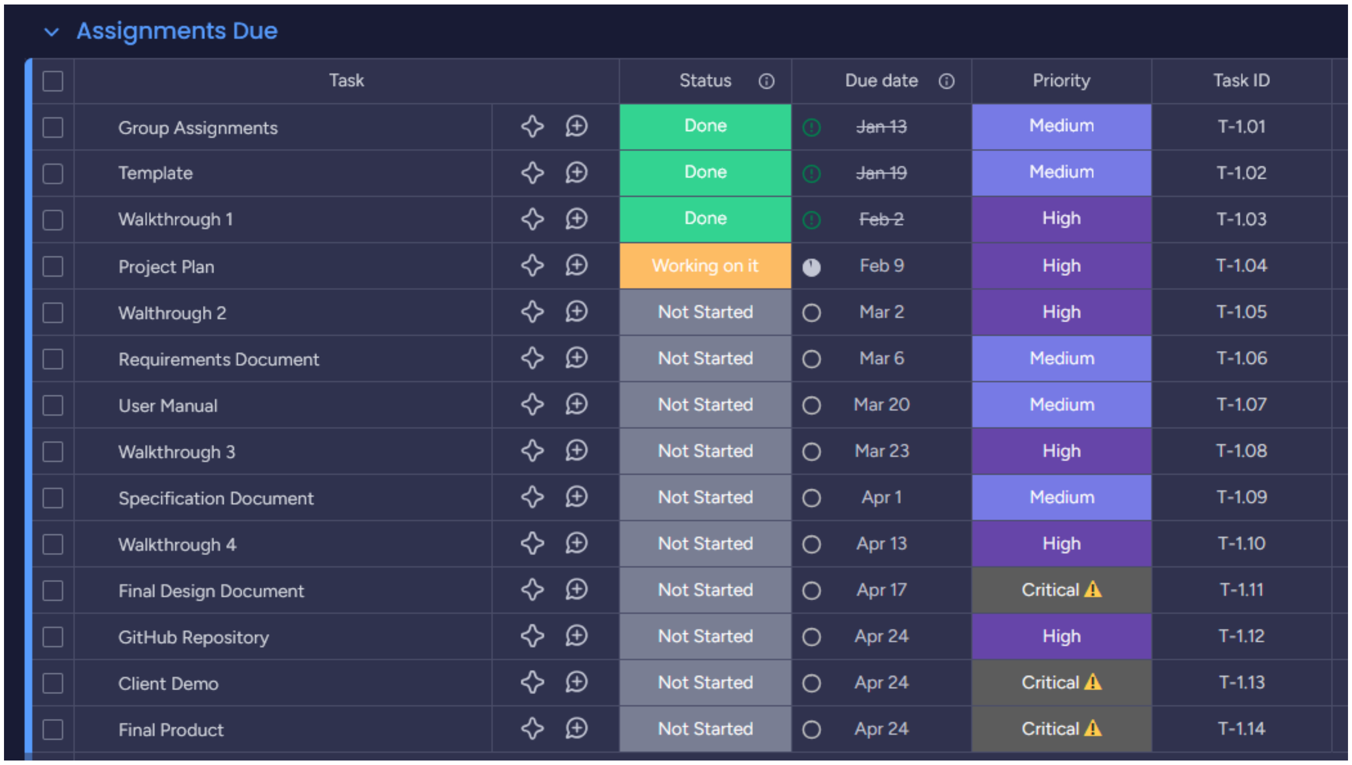
Hardware

- Dell Pro 14 Premium
- 32 Gb of RAM
- Windows 11 Enterprise
- Intel i7
- 8 core processor
- 14-inch monitor
- 120 GB of disk space
- Linux type environment

Software

- Linux, Windows or MacOS
- A web browser (such as Chrome or Firefox)
- Docker
- GitHub

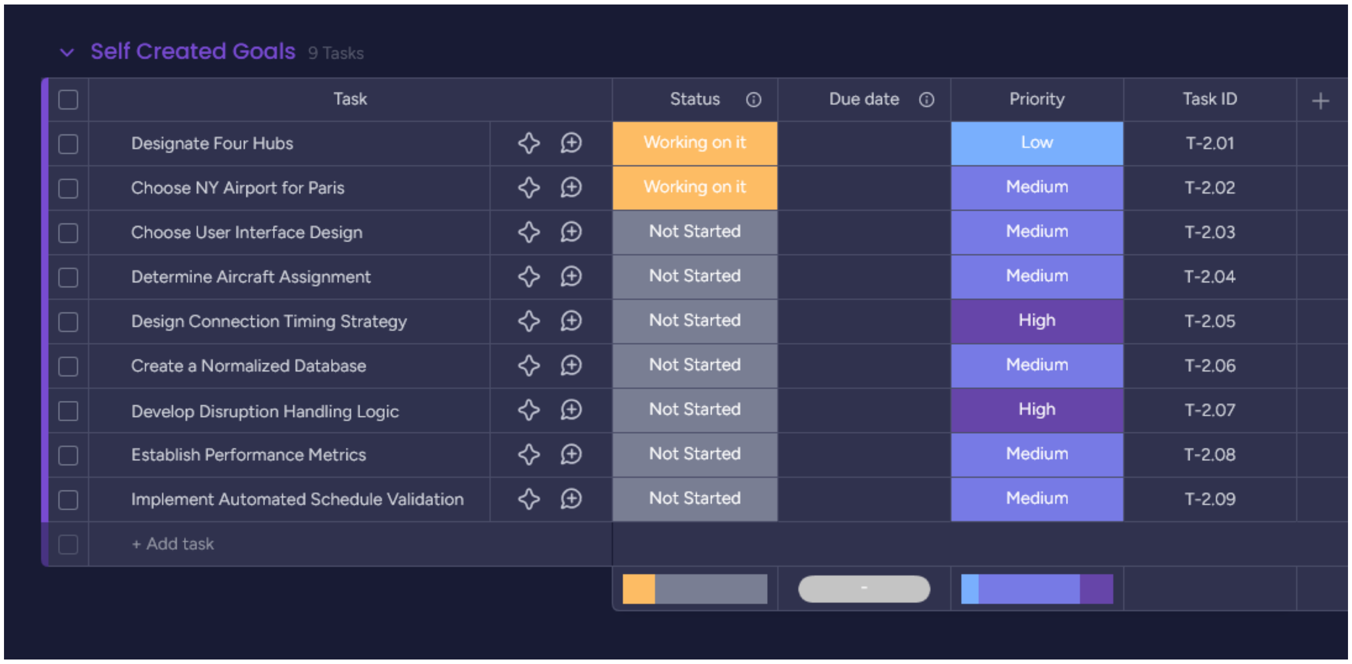
9. Schedule – Using Monday.com



Assignments Due

Task	Status	Due date	Priority	Task ID
Group Assignments	Done	~~Jan 13~~	Medium	T-1.01
Template	Done	~~Jan 19~~	Medium	T-1.02
Walkthrough 1	Done	~~Feb 2~~	High	T-1.03
Project Plan	Working on it	Feb 9	High	T-1.04
Walthrough 2	Not Started	Mar 2	High	T-1.05
Requirements Document	Not Started	Mar 6	Medium	T-1.06
User Manual	Not Started	Mar 20	Medium	T-1.07
Walkthrough 3	Not Started	Mar 23	High	T-1.08
Specification Document	Not Started	Apr 1	Medium	T-1.09
Walkthrough 4	Not Started	Apr 13	High	T-1.10
Final Design Document	Not Started	Apr 17	Critical ⚠	T-1.11
GitHub Repository	Not Started	Apr 24	High	T-1.12
Client Demo	Not Started	Apr 24	Critical ⚠	T-1.13
Final Product	Not Started	Apr 24	Critical ⚠	T-1.14

Figure 1: Project Schedule for Assignments or GitHub Deliverables



Self Created Goals 9 Tasks

Task	Status	Due date	Priority	Task ID
Designate Four Hubs	Working on it		Low	T-2.01
Choose NY Airport for Paris	Working on it		Medium	T-2.02
Choose User Interface Design	Not Started		Medium	T-2.03
Determine Aircraft Assignment	Not Started		Medium	T-2.04
Design Connection Timing Strategy	Not Started		High	T-2.05
Create a Normalized Database	Not Started		Medium	T-2.06
Develop Disruption Handling Logic	Not Started		High	T-2.07
Establish Performance Metrics	Not Started		Medium	T-2.08
Implement Automated Schedule Validation	Not Started		Medium	T-2.09
+ Add task				

Figure 2: Project Schedule for Self-Created Goals

10. Delivery Plan

- Project Plan Document

    - Comprehensive project overview including team structure, roles, schedule, and risk management
- Requirements Specification
    - Functional and non-functional requirements with unique identifiers
    - Client questions document with responses
- Use Case Diagrams
    - Visual representations of system interactions for staff users
    - Includes flight search, schedule viewing, and report generation workflows
- System Architecture Documentation
    - High-level system design showing frontend, backend, and database layers
    - Technology stack justification (React, Flask, MariaDB)
    - Component interaction diagrams
- Test Documentation
    - Test plans, test cases, and testing procedures
    - Functional, integration, and performance test results
    - Bug tracking and resolution logs
- Final Report
    - Complete project summary including objectives, implementation details, testing results, lessons learned, and future recommendations

11. Communication and Reporting

Communication:

- **In-person meetings**
  Used for weekly team meetings, walkthrough preparation, design discussions, and major decision-making.

- **iMessage**
  Used for quick updates, availability checks, and short coordination messages between meetings.

- **GitHub Logs**
  Used to track development progress, code contributions, commit history, and activity levels for accountability and transparency.

Documentation:

- **SharePoint (Word Documentation)**
  Used to store and manage all formal project documents, including the project plan, requirements, meeting minutes, status reports, and templates.

- **Personal Engineering Journals**
  Each team member maintains a bound personal journal documenting design decisions, meeting discussions, individual contributions, and reasoning behind technical choices.

12. Project Constraints

ID	Constraint Name	Constraint Description
C-1.1	Platform Compatibility	The system must execute in a Linux-compatible environment to ensure portability and grading consistency.
C-1.2	Database Deployment	All persistent data must be stored in a MariaDB database deployed using Docker containers.
C-1.3	Access Control	System access is restricted to Panther Cloud Air staff and administrators; no passenger-facing access is permitted.
C-1.4	Airport Operating Hours	Generated flight schedules must comply with defined airport operating hours and prevent arrivals after closure.
C-1.5	Aircraft Resource Constraints	Aircraft scheduling must enforce turnaround time, gate availability, and maintenance requirements.
C-1.6	Fixed Academic Timeline	The project must be designed, implemented, and delivered within a single academic semester.

C-1.7	Limited Team Availability	Total development effort and real-time collaboration are constrained by team members' concurrent academic and employment commitments.
C-1.8	Repository Security	All project artifacts, including code and documentation, must be stored in a secured GitHub repository with controlled access.
C-1.9	Funding Limitations	The project must rely exclusively on freely available or institution-provided software tools due to the absence of external funding.

Table 2: Project Constraints

13. Risk Assessment and Management

ID	Risk Name	Risk Description
R-1.1	Scheduling Complexity	As scheduling rules continue to grow (hubs, maintenance, gates, and time zones), the logic may become difficult to manage and could lead to invalid or unrealistic flight schedules.
R-1.2	Time Zone Errors	Mistakes in handling time zones or daylight saving changes could cause flights to appear at incorrect departure or arrival times.
R-1.3	Simulation Data Overload	The amount of data generated during the two-week simulation may slow down the system or make reports harder to generate and interpret.
R-1.4	Development Environment Issues	Differences in local development setups could cause unexpected bugs or delays when integrating team members' work.
R-1.5	Frontend and Backend Misalignment	Changes made to the backend or frontend without proper coordination may result in broken features or incomplete workflows.
R-1.6	Unrealistic Simulation Outcomes	Randomized events in the simulation may produce results that do not realistically reflect airline operations.
R-1.7	Team Availability and Knowledge Gaps	If a team member becomes unavailable, especially one with knowledge of critical components, progress could slow while others ramp up.

R-1.8	Schedule Underestimation	Some tasks may take longer than expected, which could lead to missed milestones or rushed work near deadlines.

Table 3: Project Risks