Professional Self-Assessment

Mohamed Aziz Zaghdoudi

## Introduction

This professional self-assessment reflects my growth throughout the Computer Science program at SNHU and summarizes how my CS 499 capstone an Airline Dispatcher application demonstrates readiness for professional software development roles. Building a full-stack system and curating a cohesive ePortfolio allowed me to integrate software engineering, algorithms and data structures, and databases into a practical, real-world solution.

## Strengths and Professional Preparation

The capstone has consolidated my strengths in full-stack development, system design, and technical communication. I am confident in planning, executing, and iterating software that must balance user needs, maintainability, and security. I can translate ambiguous requirements into concrete architecture, choose appropriate tools, and measure trade-offs in complexity, performance, and team velocity.

### Collaboration in a Team Environment

Throughout the program, I worked in team settings that mirrored industry practices breaking down features, writing clear tickets, and reviewing code for readability and correctness. I learned to use Git strategically (feature branches, descriptive commits, and pull requests) and to resolve conflicts effectively. In the capstone, I applied these habits to structure the repository into a clean frontend/backend split, making it easier for collaborators to contribute without stepping on each other’s work.

### Communicating With Stakeholders

I improved at adapting the depth and tone of my communication to different audiences. The informal code review video and enhancement narratives required me to explain architecture and design choices clearly, justify trade-offs, and present outcomes connected to course objectives and professional goals. This reinforced my ability to document APIs, write concise READMEs, and visually present progress in a way that non-technical stakeholders can follow.

### Data Structures and Algorithms

In my Dispatcher app, I designed assignment logic that prioritizes flights by ETA/ETD while distributing workload fairly among available agents. I evaluated alternative approaches (e.g., heaps vs. sorted lists; maps/queues for fast lookups and ordering) and implemented strategies that are performant for realistic workloads and easy to maintain. I articulated complexity, tie-breaker rules, and fallbacks, demonstrating the ability to design and reason about algorithmic solutions under practical constraints.

### Software Engineering and Databases

From a software engineering perspective, I migrated the frontend to React + TypeScript + Vite, structured features cleanly, and implemented role-based routing guards. On the back end, I built a Node/Express REST API with MongoDB persistence using Mongoose models (Flights, Agents, Assignments, Users). The system now supports realistic workflows: dispatchers run and publish schedules; agents retrieve their assigned tasks. The architecture is modular, testable, and ready for incremental features like audit logs and notifications.

### Security Mindset

I integrated a security mindset by avoiding hardcoded roles, introducing JWT-based authentication, and hashing passwords with bcrypt. I enforced server-side authorization (RBAC) and reduced trust in client-provided data. Configuration is externalized via environment variables, and the system design considers principle of least privilege and defense in depth. Future work would include rate limiting, stronger input validation, and more granular permissions as the feature set grows.

## How the Artifacts Fit Together

The three enhancements form a coherent whole. The software design work (routing, TypeScript migration, and guarded views) provides a robust UI foundation. The algorithms and data structures enhancement adds the core business logic for flight-to-agent assignment. The database enhancement introduces durable persistence and a clean API layer. Together, they demonstrate end-to-end capability from user experience and system design to algorithmic correctness and operational data management.

## Learning, Challenges, and Feedback

Key learning included structuring a monorepo-like layout (frontend/server), defining shared types, and ensuring alignment between frontend models and backend schemas. A recurring challenge was balancing simplicity with realism, keeping the scheduling logic understandable while reflecting real constraints like arrival vs. departure priority. Instructor feedback helped me clarify requirements and improve documentation. I refined naming, added comments, and improved the README and narratives to communicate decisions more clearly.

## Career Goals and Next Steps

I plan to pursue full-stack or software engineering roles where I can contribute to scalable, user-focused systems. Near-term improvements include adding integration tests, role-based dashboards with richer analytics, and CI/CD automation. Long term, I want to deepen my expertise in distributed systems and security engineering while mentoring junior developers and contributing to high-impact product teams.

## Conclusion

The CS 499 capstone and ePortfolio showcase my readiness to deliver value as a software engineer. By combining solid engineering practices, algorithmic thinking, and secure database-backed design, I created an artifact that is both technically sound and practically relevant. I am confident this work represents my growth and my potential to contribute meaningfully in professional settings.