**Algorithms & Data Structures Narrative**

**Name:** Anthony DePoalo

**Course:** CS-499 – Computer Science Capstone

**Instructor:** Joseph Martinez

**Date:** October 19, 2025

The artifact for this enhancement comes from the Travlr Getaways full-stack project. This app handles trip data using both server-side code with Node.js and Express, and client-side code with Angular. It uses several algorithms for login, data filtering, and managing what’s shown on screen. For this part, I worked on making the code faster, using better algorithm logic, and improving security through cleaner data handling.

This artifact shows that I can study, build, and improve algorithms to make them both faster and safer. Updating the Travlr Getaways project gave me the chance to use key computer science skills in a real full-stack web app. By improving password hashing, making login checks stronger, and speeding up the front-end search, I showed that I can apply algorithm concepts in ways that make a real difference for users and system performance.

**Enhancements Implemented**

1. PBKDF2 Iteration Increase for Secure Hashing
   1. Increased PBKDF2 iteration count to over 100,000 to align with current NIST and OWASP recommendations for secure password storage.
   2. Verified compatibility with existing user accounts while improving resistance to brute-force and rainbow table attacks.
   3. Files Updated: authentication.js, passport.js
2. Password Policy Enforcement
   1. Added credential strength requirements including minimum length, uppercase/lowercase checks, and special character enforcement.
   2. Implemented validation logic before hashing and storage to reject weak passwords early, reducing unnecessary hashing computations.
   3. Files Updated: authentication.js
3. Angular Client Debounce Implementation
   1. Integrated RxJS debounceTime() operator in the Angular search feature to prevent excessive API calls during typing.
   2. Reduced request overhead and improved UI responsiveness while preserving real-time search feel.
   3. Files Updated: trip-listing.component.ts

**Skills Demonstrated**

* Algorithmic Thinking: I checked how the code used time and memory to make sure the improvements didn’t make it slower or use more resources than needed.
* Security-Focused Implementation: I used PBKDF2 hashing to show how passwords can be safely protected using a secure algorithm.
* Front-End Optimization: I added a debounce feature to stop extra network requests and make the app respond faster.
* Testing and Verification: I ran both manual and automated tests to make sure the new code worked with old and new data without breaking anything that was already working.

This milestone helped me better understand how different algorithm choices can affect how a system works in the real world. Increasing the PBKDF2 iterations made passwords safer, but it also made the system take longer to process. To handle that, I tested it on several devices to make sure it still worked well without timing out.

Adding the debounce feature in Angular showed me how timing and data flow can change how smooth and fast the app feels. I had to adjust the delay to keep the app quick while still giving users a good experience.

The hardest part was making sure these updates worked correctly with the database and front-end without breaking login or other features. I had to do careful testing and debugging step by step. This experience taught me how important it is to use unit testing, check performance, and make sure everything works as expected before releasing it.This enhancement aligns strongly with the following CS program outcomes:

* CO3 – Design and Evaluate Computing Solutions: Applied algorithmic principles to improve efficiency and security of key system functions such as authentication and filtering.
* CO4 – Innovative Tools and Techniques: Used RxJS operators and cryptographic standards to enhance performance and maintain user-friendly interactions.
* CO5 – Security Mindset: Strengthened protection of sensitive data and improved user authentication processes through secure algorithms and validation controls.

Improving the algorithm parts of the Travlr Getaways project showed that I can design, build, and test algorithms that focus on both speed and security. I used encryption methods, front-end improvements, and data checks that follow common industry standards. These changes made the program run faster, kept user data safe, and helped me build the skills I need to create reliable and secure software for industrial control systems.