**Enhancement Two Narrative**

Sarah Deleppo

Southern New Hampshire University

CS-499

Professor Brooke

July 19, 2021

**Algorithms and Data Structure Narrative**

The artifact developed for the Algorithms and Data Structure category is one piece of a larger project encompassing all three categories of the Computer Science program based on a previous project from class CS-320. In said class, we developed a Contact Manager to allow for users to store, update, and manage their contacts VIA Java code. For my capstone project, I am focusing on the refinement of said Contact Manager and converting the existing project into a Full Stack program utilizing Java and the Spring Framework, React, and MongoDB. For the Algorithms and Data Structure portion of the capstone, my work is based on the backend side of this program, utilizing Java and Spring. The work for this artifact was developed shortly after the development of the frontend UI for the previous Software Design artifact and creates a link between said UI and backend to allow them to function in tandem.

As previously stated in the *ePortfolio Selection and Software Design Document,* I chose to create an all-encompassing, full stack project as it aligns with my career goals of becoming a Full Stack Developer. My main goal with this artifact was to provide a supportive backend for the rest of the project to build off, this involved initially refining the existing Java code from CS-320. This backend code is the core foundation of how I communicate with application users and my audience, it provides the technically sound create, read, update, and delete functions that will be utilized by my frontend UI, and therefore by end-users of the application. The original Java code from CS-320 was in need of modification and refinement in order to function alongside the frontend. In order to accomplish this, I reworked the project to use Spring Boot and REST API to allow the interaction between the backend to the database and the backend to the front end. This mainly involved the refinement of my original ContactService class, which was split into two classes to accommodate the additional functionality: ContactController and Contact Repository.

ContactRepository mainly deals with the interface that extends MongoRepository for CRUD methods, this will be discussed in the final artifact regarding databases. ContactController is a RestController which has request mapping functions for REST requests including getAll Contacts, getContactByID, createContact, updateContact, deleteContact, and deleteAllContacts. This differs greatly from the original ContactService class which revolved around the same CRUD methods except based on a HashMap with no REST functionality. This newly refined code now allows for a functioning backend that can communicate by sending and receiving HTTP responses to the frontend, thus improving the overall projects functionality. My original plan laid out within my flowchart from the *ePortfolio Selection and Software Design Document* was indeed met with slight modification and began with the refinement of existing Java code. Once this was accomplished, I began configuring REST API bootstrapped with Spring and worked to connect to a MongoRepository interface. This allows for the code to be run on port 8080 and connect to both the MongoDB database and the React frontend. In terms of future refinement, I plan to add an additional field within the Contact class down the line for users to submit their contact’s address.

This was one of my first experiences writing API that allowed the backend to communicate to other parts of the stack. Previously, I did not realize that each portion of the stack was run on a different port, with the MongoDB running on port 27017, and the frontend running on port 8081 in order to communicate with the backend which was running on port 8080. Understanding how each piece of the application’s architecture fits together is vital to development, this project provided me a clear picture of how the full stack functions together as one. Understanding that the backend needs to be able to check specific ports in order to communicate with the frontend really helped the whole process click and helped me gain a deeper understanding of full stack development. Not only was this my biggest lesson gained from this artifact, but it was also the biggest challenge, ensuring all of these methods communicate correctly and efficiently with one another. This was my first experience sending and receiving HTTP requests, so understanding which annotations to use where definitely had a learning curve, but I feel I’ve gained a much deeper understanding of full stack architecture through this challenge as well as through sifting through documentation.