Milestone 3 Enhancement

William P. Giittinger

william.giittinger@snhu.edu

Southern New Hampshire University

**Zoo Authentication Website Data Structure / Algorithm Artifact**

This milestone submission contains a variety of improvements and new features to the Zoo website, including the primary new feature, the binary search tree. To implement this, a variety of classes were constructed to implement data structures for the zoo animals, the nodes for the tree, and then the tree itself. As should be obvious, each node contains the data for the animal it represents, and references to two “leaf” (a.k.a. “child”) nodes. The root of the tree is the first node placed. From there, the tree branches left and right based upon the id of the animal being inserted. If the new animal’s id is less than the current node, the program attempts to place it down the left branch. If the id is greater than the current node, it attempts to place it down the right branch. The tree contains a variety of utilities including adding new animals, searching for animals based on id number, displaying an ordered list of the animals, and removing animals from the list based on id. There are also minor improvements to the login system which now includes some css and images. Lastly, the comments and documentation were improved by adding more detailed comments and ensuring conformity in commenting style throughout the application.

As mentioned in the initial project plan, creating a binary search tree (BST) when a MongoDB is to be used in the next improvement is slightly redundant, since MongoDB’s indexes work essentially the same. However, this artifact proves that I understand this type of data structure and the algorithms that can be used to interact with it. This tree structure can also have many other features added to it if desired, such as depth-first searching, breadth-first searching, greedy methods, etc. These types of algorithms are important for efficiently traversing, sorting, searching, etc. through data structures. It also showcases that I’m able to create these systems from scratch and use my own data. A json file of sample data was created that can be imported into the BST with the click of a button. While extremely limited, the data structure also has some basic security checks in place to ensure that, at the very least, the unique identifiers are indeed unique, and the tree’s structure remains sound.

I believe that I have achieved the course objectives in this artifact with the development of a BST from scratch with many helper methods. Importantly this follows the project’s initial plan to create CRUD functionality with the data structure. I learned a lot working with this data structure in a website application as well. For example, the resulting files and methods are more “WET” (write everything twice) than initially anticipated. In particular, the controller for the welcome page which handles the BST and its data should contain much more of the functionality than it currently does. For example, the insert function could be written in just the controller and BST files. However, it’s also contains pieces in the animals.js file for creating instances of the animals. This will change once the database is implemented and wasn’t something I had anticipated, but I can see potential changes to the application that would make it more “DRY” (don’t repeat yourself). The only major update to the outcome-coverage plan is to improve the design of the welcome page by adding more css, images, etc. This wasn’t initially planned but goes a long way in making it feel like a professional development effort.

As with essentially all development efforts, there were some challenges. As previously mentioned, due to using JSON files to store and import data, I unexpectedly started to have some repeat functionality. To be fair, the functionality isn’t so much repeated, but the methods could be condensed and centralized to the controller rather than the auxiliary files. I also got to practice more with the EJS view engine which was admittedly difficult to get working as intended, but I’m learning in more detail how it is used and how it works with other tools (e.g. needing to manually call to rerender the page to display updated data). For a final challenge, it was my first time working with a plain json file and interacting with it with JavaScript. The parsing and reading methods used for the file were refactored 3 times. Finally, I learned that node.js’ require statements automatically turn the file into a usable JavaScript object. Lastly, I learned that no matter how perfect my method was for reading from the file, if the path is wrong at the API endpoint, it still won’t work as intended.