Professional Self-Assessment

William P. Giittinger

Email: [william.giittinger@snhu.edu](mailto:william.giittinger@snhu.edu)

GitHub Pages: <https://williamgiittinger.github.io/>

Southern New Hampshire University

**Reflection**

Throughout my time in the Computer Science (CS) program at Southern New Hampshire University, I have worked from the ground up to build the skills and knowledge needed to be a professional web developer. At the beginning, it was learning the basics with mathematics and simple scripting with Python. By the end, I was creating full-stack websites, object-oriented programs in a variety of programming languages, and even reverse engineering basic programs to use the Assembly instructions to recreate the program in C. Today’s developers need to be adept in more than just making a functioning program. To that end, the CS program has exposed me to a variety of scenarios and taught me industry best practices to approach each challenge.

Early in the CS program, I had coursework in the software development lifecycle and collaborating in team projects. The scenarios I worked through and the development team simulations we worked through taught us many aspects of working in today’s software development fields. We covered many approaches to the software development lifecycle, from Waterfall to Prototyping. The primary focus of my studies was the Agile methodology. As a team we learned how to communicate and coordinate development efforts with ourselves, stakeholders, and clients. Learning to discuss with stakeholders about the design of a program and the needs of everyone involved before beginning development was a critical learning step for me that proved to be vital throughout the rest of my time in the CS program. I quickly grew to appreciate that asking for clarity and identifying specific needs and expectations as early as possible is one of the most surefire ways to avoid issues, mistakes, and wasted resources later.

After learning how to be an effective communicator and the preparatory steps for program development, much of the rest of my time in the CS program was spent covering a wide range of software topics. An entire course was dedicated to learning different data structures and algorithms, from which I learned the concepts necessary to work with more complex data structures. For example, it was this course that taught me how to develop and interact with binary trees which I have since implemented in my capstone project, a full-stack website created from the ground up. I also learned about a variety of databases. The first database I learned to use was a MySQL database, including how to implement CRUD (create, read, update, delete) functionality. This was expanded upon with more advanced concepts such as aggregation. In later courses, I was exposed to the document-oriented MongoDB which I also utilized as the database for my capstone project. I also learned about software security and the potential harms that can occur if it is not taken seriously. While reverse engineering, I learned a wide array of security issues that occur in program logic such as dangling pointers, stack smashing (buffer overflow), broken algorithms, and more. Furthering my security knowledge in practice, the website I created in my full-stack development course included full password encryption, session support using the node.js Passport module, and discussion on other attack vectors such as injection attacks.

**Capstone Project**

For my capstone course, I needed to address three main areas of expertise and showcase the skills and understanding I had of each. Specifically, the three focal points were software design and engineering, algorithms and data structures, and databases. Considering my desire to work as a web developer, instead of using three different programs or improving my previous programs, I decided to create a full-stack website from scratch and focus on those areas. Before starting development, I performed an informal code review of some of my previous work to identify specific ideas that I could use to demonstrate my capabilities. This review also focused on the strengths and weaknesses of those programs and how I could improve and mold the functions in them for my website. It also helped me identify a theme for my website, a website for a zoo where zookeepers could login and keep track of the zoo’s animals (e.g. add newly acquired animals, keep track of their status, etc.).

In my full-stack development course, I had used the MEAN stack (MongoDB, Express.js, AngularJS, Node.js), and I decided that stack was appropriate for the website I was creating. I decided not to include AngularJS for this project as it was not a piece I identified as necessary for the three focus points, and development time constraints meant it was an unnecessary component. For the first development artifact targeting the software design category, I created a front-end login system which utilized Passport to authenticate users and for session support. Since the database artifact was not yet created, all user data was stored in text files. The passwords were not fully encrypted, but the bcrypt module was used to hash the passwords which is a strong first step for security. At the completion of the artifact, the zookeepers could register for the site with their name, email address, and password, and then use his/her email and password to login to the website. The server handled routing the user between pages, and once the user successfully logged in, a welcome page was displayed by a dedicated controller. This welcome page was inaccessible to anyone who was not authenticated, even with the correct route.

The goal of the second artifact was to implement a binary search tree which could be used for sorting, searching, adding, deleting, etc. animals to the tree. At some level, the creation of this binary search tree is redundant as much of the functionality is already supported by MongoDB. However, the goal was to showcase my capacity to create the data structure and properly implement the algorithms needed to use it. Again, at this point the database is not implemented and the animal data is stored on json files and imported / initialized as “Animal” class objects. In this artifact I also displayed my ability to work more extensively with HTML, CSS, and the EJS view-engine for Express.js. The website’s visual design was not a point of focus for the project, but drastically improves the usability of the website by moving from displaying plain text to stylized elements.

The final artifact was to implement the document-oriented MongoDB. Ideally this would have been implemented much earlier on, but the structure of the capstone course placed this artifact last. The database was created through MongoDB Atlas and is hosted by Amazon Web Services (AWS). As such, the database is stored in the cloud and can be accessed from any device connected to the internet, which is a fundamental requirement for a website of this kind. The structure of the MonoDB is not complex and was designed with two collections, one for the user (zookeeper) data and another for the animal data. Once the database was online and connected to the server (using hidden .env variables for authentication), large portions of the website were then refactored to utilize the data stored on the database rather than the text and json files. An unforeseen benefit of having to rework parts of the website was the demonstration of my ability to work with an existing code base and change it to meet changing requirements. For added benefit, the mongoose module was utilized for data modeling on the server side. This provided numerous benefits, but primarily allows the defining of schemas and data enforcement. With this artifact completed, and a few extra web design improvements, the full-stack website project was complete.

Project’s working GitHub repository (not GitHub pages): <https://github.com/WilliamGiittinger/ZooAuthenticationSite>