1. **Briefly describe the artifact. What is it? When was it created?**

The original artifact I am basing my improvements on is the CS-465 travlr project, which was a full stack web application I created in July of 2023.

2. **Justify the inclusion of the artifact in your ePortfolio. Why did you select this item?**

I chose this item because creating a full stack web application would provide an excellent portfolio item for applying to web development jobs.

3. **What specific components of the artifact showcase your skills and abilities in software development?**

The artifact required me to use the MEAN stack, showing the ability to use a MongoDB database, angular frontend and node/express backend. As a part of that it also showed skills with Javascript.

4. **How was the artifact improved?**

My improvements to the artifact begin with using the Phoenix Framework and the elixir language. The elixir language is a functional language which is quite different from Javascript, and more niche. The Phoenix framework likewise is very different from using the MEAN stack, and being less popular is more difficult to learn due to a paucity of resources. For this milestone specifically I added an SQL database using postgresql and the Ecto toolkit to manage interaction between the web app and that database.

5. **Did you meet the course outcomes you planned to meet with this enhancement in Module One?**

Yes, I was planning to make improvements which make the web app useful for a specific business use case relevant to my job as an underwriter. With this enhancement, that is taking shape because you can upload a list of quotes and then browse them by brokerage and broker. This is useful when calling a broker on an account because you can see other open quotes you have with them to ask them about. Surprisingly, my company's systems don't have an easy or fast way to get that information at present. I also added authentication to the app, which then allows quotes to be stored in the database with the user's email as a field, allowing the app to then show only the user's quotes. For the broker and brokerage views it shows all the quotes in the database which match that datapoint, because for the business use case that is the data you'd want to see. This is because brokers are usually assigned to a specific underwriter, so if I have one quote outstanding with a broker, I'd want to know about the 20 they might haver outstanding with a colleague.

6. **Do you have any updates to your outcome-coverage plans?**

For this milestone the code I produced maps quite well to the plan I put in the Module One Assignment. I intended to create an SQL database to hold the quotes, as well as authentication to allow a user to see their own quotes. I expanded that slightly, allowing the broker and brokerage views to show the matching quotes for all users.

7. **Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it?**

I learned a huge amount about the Phoenix Framework and Elixir in creating this enhancement. I especially learned a lot about the Ecto Database management module included in Phoenix works. I also learnt quite a bit about managing authentication using a 3rd party library. However, I would say I'm not especially comfortable with that and it will be an area I will seek to expand my knowledge of in the future.

8. **What challenges did you face?**

In contrast to something like the MEAN or MERN stacks, the Phoenix Framework definitely has a steeper learning curve. This is in part to do with its relative lack of popularity which results in significantly fewer learning resources. This is complicated by changes in the Phoenix Framework, such as the introduction of LiveView which have changed the framework significantly even in the last few years. Additionally, the Phoenix Framework is much more opinionated than the MEAN or MERN stacks. This requires more up front learning, but I can now definitely appreciate how it makes for more maintainable apps. For instance, with the MEAN stack, you could develop the angular front end with the node/express backend completely before adding the MongoDB database on at the end. The Phoenix Framework in contrast requires you to create the database for the project before even running it, and when you create contexts/models you have to create and run database migrations. I found this difficult but it is certainly a good practice to get used to.

Checkpoint	**Software Design and Engineering**	**Algorithms and Data Structures**	**Databases**
Name of Artifact Used	**Artifact Name:** Travlr, travel full stack application Origin: CS-465 CS-465 Final Project	CS-465 Final Project	CS-465 Final Project
Status of Initial Enhancement	Complete	Complete	Complete
Submission Status	Submitted	Submitted	Submitted
Status of Final Enhancement	Complete	Complete	Complete
Uploaded to ePortfolio	Yes	Yes	Yes
Status of Finalized ePortfolio	Complete	Complete	Complete