Casey Dzimiela CIDM 6330/Spring2023

Final Project: Bicycle Logistics and Route Planning

Validation and Justification

I will start by making in clear that I did need to drastically reduce the scope and the grand scale at which my original domain idea originally started with. I understand that this was an intentional hurdle that Dr. Babb realized far in advance but me as a very amateur realized it was probably well outside my wheelhouse to attempt to do the entire scope of my original domain plan but now with some greater knowledge and understanding of both the process and my own skillset realize that it was more appropriate to look towards space travel in the time allotted.

Now with all of that said, it was relatively simple to reduce my domain down to a much more chewable project than the original elephant that I had created to try and chew and swallow in one bite. Looking at my Domain model essentially I would need to distill the project down to just Route, Ride and Rider data entities. Attempting to integrate the Datasources, Weather, and Traffic in is just far more advanced than is realistic and I knew that going into this but I was not sure what the scope of the project would be. I do understand why Dr. Babb chose to have students shoot for the moon and create any Domain model that they are passionate about because this subject is difficult enough for myself, but having a topic that I am passionate about makes it easier to relate to because you really already understand the inner workings of how the Domain flows together as if you were the owner of your own business and you needed to explain to your vendor or even to your co worker what you are attempting to accomplish or how processes flow through your organization.

While removing half of the entities in my domain is a huge step backwards in the complexity of my Domain and ultimately its functionality and capabilities it is not really going to hamper the opportunity for me to show my understanding of Architecture Patterns with Python or Test-Driven Design patterns. Essentially my redacted domain model turns into a very basic sort of Bark or even Flask type of tutorial depending on what type of interface that I choose to apply to it. I have learned in this class and also as Bob Martins clean architecture diagram shows that the external framework and drivers that support what we are doing are far less important than the chosen entities. I just know that I would not be able to get to a place where I could even start to provide a User or External interface at all if I chose the full domain I originally came up with.

In working in the class with Percival and Gregory code as well as what Dr. Babb has provided in the form of “Barky” form Deal Hillard I am seeing the different service layers and what is needed in order to accomplish the example tasks such as create a Bookmark repository. That process and depending the interface is probably not that far off from similar in the sense of my reduced complexity domain. Regardless of the complexity a user would be creating a ride/route and then that route would be referenced by other users. So for certain tests that were done in the Barky example where we are looking to do certain actions such as “CRUD” Create, Read, Update, and Delete all of these things would take place the difference would be that these routes and the overarching repository is going to be used for all users of the program. So that said the idea of Reading, Updating, and Deleting is valid but not final.

Unlike in the Barky or bookmark example each user will have their own routes that they will contribute to the overall repository but as that data is required to stay and build on an aggregate the concept of a Delete is not the same. So it would not be a database Delete it would again fall under more the context of another “Update” to that users instance and not a Delete to the overall domain. The goal is to expand the scope of the amount of routes and rides that data can be pulled from and searched. Due to lack of understanding of the complexity of integrating the more complex DataSource Element of my Domain model, I do know that this is really the foundation of what is going to make it successful. So in the context of the simplified Domain that my code will represent the tests that will take place will support this concept of Create Read Update without a true delete action. The Delete action will just be remove the pointer or association of that database entry from that specific user and the entry will just exist by itself but serve as data for the larger aggregate repository.

Essentially in order to have something that is chewable and attainable for a personal domain I have elected to remove all of the External Adapters that exist between the internal repositories and the Weather, Traffic, and DataSource Entities. I am unsure if in reality if this data and API’s are even available for me to tap into. I know that I would have access to my own personal accounts that I hold on a few of these platforms and that may in a poorly secured construct might work for the needs of what the Weather and Traffic adapters would need to function, but the DataSource would fall catastrophically short of the intended design intentions. This realistically is because I would need to have rights to all of the ride data of multiple bicycle GPS unit platforms then determine a way to get all of their schema to align and then create a query mechanism integrating that information with my own platform as well as making some sort of route creation program that would then interface back with all of those different types of GPS middleware programs.

I think that there is a common standard for what each brand or manufacturer of GPS device uses and the file format so that it can be read and function with the device so that would not be anything to hurdle but it very much would need to be taken into consideration. I also know that these files do not take into consideration and live data. There are alerts that can be built into certain locations within the route for instance if there is a known sharp corner and your route includes this there can be an alert. To have live weather or traffic data would be much different and almost turn into its own application separate form the GPS file if I am understanding the way these devices function. So the route is static and works based on GPS satellite position and is pre-loaded but if you wanted to use live traffic or weather data you would need to have a live connection. So more than likely this would require there to leverage a Bluetooth connection to your cell phone and assume that you have connectivity in that location along with an assumption you bring your cell phone along. At this point your phone would be in constant update with your GPS route as well as both weather and traffic apps then providing pertinent updates. It almost seems like it would be easiest if this app was completely cell phone driven and not use a bicycle GPS device as it is almost just a unnecessary tangent of connectivity.

The Event publisher and Event consumer processes in the paragraph discussion above is far beyond what I am capable of. My domain and Final Project will relate to much more of a Bark or flask type operation with the Bicycle Logistics and Route application user doing a manual entry of their ride data that exists in the Route and Ride Elements of the Domain. From here it is really showing the AAP and TDD functionality that it is all able to work together. Can a user Create a new ride, can a user update the speed, bicycle type or time that the ride consisted of. I am in all honestly going to need to remove the actual route from the logic all together as that is beyond my capability to attempt to integrate that type of data into the Message Bus, Unit of Work and Repositories. It is far too complication I would not even know where to start with having my simple app ingest a file from a GPS device and be able to create a repository from it to be queried later. A more simple data entry format is what I strive to do and then show tests to support its function.