James (JJ) Hatzibiros

CS 499 Milestone Two: Enhancement One

**1. Briefly describe the artifact. What is it? When was it created?**  
 The artifact is the Animal Shelter Dashboard web app I built in spring 2024 for my CS 340 class. It shows a list of animal records (dogs, cats, etc.) in a table. It’s meant to house all of the records of animals in a specific shelter, although the app can be customized to be used for almost any other record-keeping scenario. Animal shelter volunteers can add animal data to describe animals that are in the shelter and their adoption status so that the shelter can keep track of which animals are at the shelter and which are available to adopt. Users can add new animals, update existing records, or delete entries. It also displays a pie chart of breed distribution and a map showing each animal’s location. Originally, I used Flask + Dash to connect to a MongoDB database and render everything on one page.

**2. Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?**  
 I chose this dashboard for my ePortfolio because it ties together back-end database work with a live front-end interface. When I first published it, only one person at a time could run CRUD operations (create, read, update, delete). If someone else had it open in another browser, they would not see new changes until they manually reloaded. I improved it so that whenever one user changes a record, say they mark a dog as “adopted”, everyone else who is viewing the dashboard sees that update instantly, without refreshing. This matters because all volunteers share the same data. As an example, if I am about to assign a foster home to a dog, I need to know right away if another volunteer just reported that the dog was adopted. By broadcasting each change over a WebSocket channel (SocketIO), the dashboard stays in sync for every connected user.

I also added a live comment box under the table so users can post notes (e.g., “I am meeting this cat at 3 PM”) and see others’ comments in real time. I also built an opt-in setting so that only volunteers who truly need instant alerts will get them. People who just want to view the data without interruptions can leave notifications turned off. This way, those who must coordinate on a new adoption (shelter volunteers) will see live updates, and those who only need to glance at the dashboard (such as administrative staff, etc.) will not be bothered.

**3. Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?**  
 Yes. In Module One I aimed to meet two specific course outcomes:

* **Outcome 1: “Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.”**
* **Outcome 4: “Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals.”**

By adding real-time broadcasts and a shared comment panel, the dashboard now lets multiple volunteers work together on the same data. Everyone sees changes as they happen and can leave live feedback. That fulfills Outcome 1, because it creates a collaborative space where decisions (e.g., “Which animal do I schedule for adoption?”) happen in real time.

Blending RESTful API requests with WebSocket messaging (SocketIO) and Dash’s graphical components shows Outcome 4. I combined several technologies: Flask for the server, MongoDB for storage, SocketIO for live updates, and Dash components for charting, into one cohesive app that delivers immediate value. I also added “opt-in” notifications as well. Now volunteers who want instant updates can turn notifications on, and employees who only need to review data can keep them off.

**4. Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?**  
 When I first tried to add live updates, I mistakenly added SocketIO() inside the constructor of my AnimalShelter class. That meant every time I called the CRUD methods, a new SocketIO instance was created, and nothing got broadcast properly. Learning to store the socket instance as self.socketio taught me how to structure class-based code correctly: create one server-attached socket object at startup, then reuse it for every method.

At first, I wasn’t sure why “everyone needs to know” when a change is made. I realized that all volunteers share the same data: if one person marks a cat as “adopted,” every other volunteer should see it immediately so they do not schedule the same cat for a different foster home. That clarity helped me understand real-time collaboration’s business value.

I then built the comment panel by writing a Dash callback that takes text input, wraps it in an HTML <p> tag, and appends it to the comment list. That taught me how to use Dash’s callback structure to dynamically render new elements. I also learned how to integrate Flask, SocketIO, and Dash in the same Python file rather than treating them as separate apps. Designing the opt-in toggle showed me why it is important to plan exactly who needs live updates. Rather than broadcasting to all connected clients, only those who have checked “Notify me” will receive every change. This keeps users who only “view” the dashboard from being distracted.