MODULE 5-2

ENHANCEMENT THREE: DATABASES

Poritosh Mridha

CS-499 Computer Science Capstone

Southern New Hampshire University

June 9, 2024

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

   The chosen artifact for the databases category is the Salvare Search for Rescue Web App, developed during the CS340 Client/Server Development course. This web application interacts with an existing database from animal shelters to identify and categorize dogs available for rescue training. It is built using Python, the Dash framework, and MongoDB, with the PyMongo driver facilitating the connection. The application can be executed in a Jupyter Notebook for testing or in a computer terminal with an internet browser.

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 selected this artifact for my ePortfolio because it demonstrates my ability to develop a complex multi-tier application using the Model View Controller (MVC) pattern and the RESTful protocol. Specific components that showcase my skills and abilities in software development include:

   Database Integration: Leveraging MongoDB for efficient data storage and management.

   MVC Pattern Implementation: Ensuring separation of concerns into the model (data management), view (user interface via Dash), and controller (data retrieval and modification using PyMongo).

   API Development: Extending the HTTP protocol to provide a robust application programming interface (API).

   The artifact was improved by recreating the web app in a Windows OS environment, updating the source code to accommodate changes in the PyMongo driver version, and using an upgraded MongoDB platform. Additionally, I revised the documentation to ensure clear instructions for setting up and running the web app in both Linux and Windows environments.

3. **Did you meet the course objectives you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?**

I met the course objectives planned for this enhancement in Module One. My outcome-coverage plans were successfully implemented, showcasing my proficiency in applying established and innovative techniques, skills, and tools in computing practices to deliver value and achieve industry-specific goals.This setup process, along with updating the source code functionality due to changes in the PyMongo driver version and the use of an upgraded MongoDB platform, enhanced my ability to apply well-established and innovative techniques, skills, and tools in computing practices to implement computer solutions that deliver value and achieve industry-specific goals.

```
# Create method to implement the C in CRUD.
    def create(self, data):
        if data is not None:
            docs = self.collection.insert(data)  # data should be a list of one or more dictionary
            print("[+] Document Created Succesfully %s" % len(docs))
            print("-------------------------------------")
            for doc in docs:  # iterate docs to list the ObjectId of the documents created
                print("ObjectId => %s" % doc)
        else:
            raise Exception(
                "[-] ERROR: Nothing to save, because data parameter is empty.")
```

Figure 4 Original code of Create Method using PMongo v3.0

```
# Create method to implement the C in CRUD.
    def create(self, data):
        if data is not None:
            docs = self.collection.insert_many(data)  # data should be a list of one or more dictionary
            print("[+] Document Created Succesfully %s" % len(data))
            print("-------------------------------------")
            for doc in docs.inserted_ids:  # iterate docs to list the ObjectId of the documents created
                print("ObjectId => %s" % doc)
            return print("-------------------------------------\n*** End of List ***")
        else:
            raise Exception("[-] ERROR: Nothing to save, because data parameter is empty.")
```

Figure 5 Modified Code of Create Method Using PyMongo v4.0

In this artifact, I adhere to industry-standard Python coding best practices and techniques, including in-line comments, proper naming conventions, and formatting that aligns with appropriate coding standards. This approach makes the code easy to read and enhances the organization of the application code. The program code is readable and follows industry-defined formatting best practices, such as proper indentation. The source code is well-structured, stylistically consistent, and properly formatted, including appropriate line breaks. We use appropriate syntax and conventions based on best practices in programming.

The implemented data structures are programmatic, allowing stored variable values to be efficiently used in other methods. Method names are verbs, reflecting the actions they perform. All cases are addressed in an IF-ELIF block, including ELSE or DEFAULT clauses to ensure comprehensive coverage.

```
151
152   ############################################
153   # Interaction Between Components / Controller
154   ############################################
155
156   # DONE: This callback add interactive dropdown filter option to the dashboard to find dogs per category
157   # or interactive button filter option to the dashboard to find all cats or all dogs
158 @app.callback(
159     Output('datatable-id', 'data'),
160     [Input('filter-type', 'value'),
161      Input('submit-button-one', 'n_clicks'),
162      Input('submit-button-two', 'n_clicks')]
163   )
164 def update_dshboard(selected_filter, btn1, btn2):
165     if (selected_filter == 'drit'):
166         df = pd.DataFrame(list(shelter.read(
167                 {
168                     "animal_type":"Dog",
169                     "breed":{"$in":["Doberman Pinscher","German Shepherd","Golden Retriever","Bloodhound","Rottweiler"]},
170                     "sex_upon_outcome":"Intact Male",
171                     "age_upon_outcome_in_weeks": {"$gte":20},
172                     "age_upon_outcome_in_weeks":{"$lte":300}
173                 }
174             )
175         ))
176     elif (selected_filter == 'mwr'):
177         df = pd.DataFrame(list(shelter.read(
178                 {
179                     "animal_type":"Dog",
180                     "breed":{"$in":["German Shepherd","Alaskan Malamute","Old English Sheepdog","Siberian Husky","Rottweiler"]},
181                     "sex_upon_outcome":"Intact Male",
182                     "age_upon_outcome_in_weeks":{"$gte":26},
183                     "age_upon_outcome_in_weeks":{"$lte":156}
184                 }
185             )
186         ))
187     elif (selected_filter == 'wr'):
188         df = pd.DataFrame(list(shelter.read(
189                 {
190                     "animal_type":"Dog",
191                     "breed":{"$in":["Labrador Retriever Mix","Chesapeake Bay Retriever","Newfoundland"]},
192                     "sex_upon_outcome":"Intact Female",
193                     "age_upon_outcome_in_weeks":{"$gte":26},
194                     "age_upon_outcome_in_weeks":{"$lte":156}
195                 }
196             )
197         ))
198     # higher number of button clicks to determine filter type
199     elif (int(btn1) > int(btn2)):
200         df = pd.DataFrame(list(shelter.read({"animal_type":"Cat"})))
201     elif (int(btn2) > int(btn1)):
202         df = pd.DataFrame(list(shelter.read({"animal_type":"Dog"})))
203     else:
204         df = pd.DataFrame.from_records(shelter.read({}))
205
206     data = df.to_dict('records')
207
208     return data
```

Figure 6 App Source Code IF-ELIF Example

As a document database, MongoDB offers a convenient way to manage a large amount of data, making it easy to store both structured and unstructured data. It uses a JSON-like format to store documents, which maps directly to native objects in modern programming languages. This makes MongoDB a natural choice, as it eliminates the need to normalize data. The web application's CRUD operations are transparent and straightforward. However, developing the dashboard using the Dash framework was more time-consuming. Understanding

and exploring how Dash core, HTML components, and callbacks work were

crucial to producing an efficient and straightforward coding structure.

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?**

Enhancing and modifying this artifact taught me several key lessons:

Technical Adaptability: Reproducing the web app in a different environment required adapting to new challenges, such as setting up Python and MongoDB on Windows, initially configured by others in a Linux environment.

Version Management: Updating the source code to accommodate changes in the PyMongo driver version and the upgraded MongoDB platform improved my skills in managing dependencies and maintaining compatibility.

Documentation Skills: Revising and refining the web app documentation to ensure it is clear and comprehensive for both Linux and Windows environments enhanced my ability to create professional and coherent technical documentation.

Security Mindset: Validating input data and designing with a default-denial approach fostered a security mindset, enabling me to anticipate potential adversarial exploits and ensure privacy and data security.

Complexity Management: Understanding and exploring the complexities of the Dash framework, including core and HTML components and callbacks, was crucial to producing an efficient and straightforward coding structure.

The primary challenge I faced was setting up the Windows environment from scratch, which required careful attention to detail and troubleshooting skills to ensure all components were correctly installed and configured. This process gave me a deeper understanding of the practical aspects of software development and deployment in different environments, enhancing my overall technical proficiency and problem-solving abilities.