ENHANCEMENT THREE: DATABASES

Abigail Sibazeu

CS-499 Computer Science Capstone

Southern New Hampshire University

02/09/2025

This narrative accompanies the enhancements made to the database artifact, providing a comprehensive rationale for its inclusion in this section of the ePortfolio. It offers a detailed reflection on the methodologies employed during its development and highlights the key insights gained throughout the creation process (Southern New Hampshire University, 2022).

The selected artifact for the databases category is the **Sibazeu Search for Rescue Web App**. This web application is designed to interface with an existing animal shelter database to identify and categorize dogs suitable for specialized rescue training. Developed as part of the **CS340 Client/Server Development** course, the application was meticulously planned, designed, and implemented using **Python** with the **Dash framework** and **MongoDB**, a non-relational database. The integration of Python and MongoDB is facilitated through the **PyMongo** driver. The application can be executed both as a testing tool within **Jupyter Notebook** and as a fully functional interface through a web browser via the computer terminal.

The application's core functionality involves importing **CSV data files** containing shelter dog records into **MongoDB**, alongside incorporating essential dependencies such as the **PyMongo driver, Python libraries, Dash framework, and a Python-based CRUD module** for data manipulation. This artifact was selected due to its implementation of a **multi-tier architecture**, adhering to the **Model-View-Controller (MVC) pattern** and **RESTful protocol design**, which extends HTTP functionalities to provide a robust **Application Programming Interface (API)**. A key advantage of the MVC approach is the **separation of concerns**, ensuring modular and maintainable code. In this implementation:

* The **Model** handles data management using MongoDB and Python data structures.
* The **View** determines the user interface and presentation via the Dash framework.
* The **Controller** is responsible for retrieving, modifying, and delivering data to the user, facilitated by the PyMongo driver.

Each record stored in the **MongoDB database** is represented as a **document in BSON format**, a binary structure that is retrieved and displayed in **JSON format** by the web application. When the main application file is executed in the terminal, users can access the web interface by opening a browser and navigating to the corresponding local address (e.g., [**http://127.0.0.1:8050/**](http://127.0.0.1:8050/)). Upon loading, the browser renders a **dynamic and interactive dashboard**, providing an intuitive platform for managing and analyzing shelter dog data.

**A screenshot of a computer

Description automatically generated**



*Figure 1 Web App Client/Database Dashboard*

The artifact demonstrates the application of best practices in engineering, including input data validation and designing with a **default-denial approach** when accessing database records. This approach fostered the development of a **security-focused mindset**, enabling me to anticipate potential adversarial exploits within software architecture. By addressing vulnerabilities, mitigating design flaws, and prioritizing privacy and data security, this artifact highlights the integration of robust security measures into the development process.

The source code is thoroughly documented, adhering to **Python standards** with a clear and maintainable commenting style. This promotes clean code principles through the use of descriptive function names and variable names, ensuring readability and ease of maintenance. Additionally, the **CRUD module** that manages database records is designed with **code reusability in mind**, allowing it to be imported and utilized as a module in other Python scripts.

Text

Description automatically generated

*Figure 2 AnimalShelter Class CRUD Module*

The implemented data structure is programmatically designed to enable efficient utilization of stored variable values across various functions and callbacks within the web application. This design approach enhances the development and evaluation of computing solutions by leveraging **algorithmic principles** and adhering to **computer science practices and standards** tailored to the specific problem. Additionally, it effectively manages the trade-offs inherent in design choices, ensuring a balanced and optimized solution.

Text

Description automatically generated

*Figure 3 App Source Code Example Screenshot*

The artifact enhancements focused on recreating the web application in a **Windows OS environment**, as the original version was developed within the **Apporto Virtual Lab**, a university-provided remote desktop Linux platform. Leveraging the web application’s initial documentation, I followed its guidance to replicate the web app environment while updating and revising the documentation to reflect the actions required for successful deployment and execution in a Windows terminal.

This process presented challenges, as the original Linux environment had been configured by others. Consequently, I had to independently set up Python and MongoDB before proceeding with the steps outlined in the documentation. Additionally, the setup process required updating the source code functionality due to changes introduced by the new version of the PyMongo driver and the upgraded MongoDB platform.

This experience enhanced my ability to apply **innovative and well-founded techniques, skills, and tools** in computing practices. By overcoming these challenges, I successfully implemented a solution that delivers value and aligns with **industry-specific goals**, reinforcing my adaptability and technical proficiency in system development and deployment.

Text

Description automatically generated with low confidence

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

Text

Description automatically generated

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

The artifact demonstrates the use of **industry-standard Python coding best practices and techniques**, including in-line comments, appropriate naming conventions, and proper formatting. These practices ensure the code is highly readable, maintainable, and well-organized. The source code adheres to industry-defined formatting standards, such as consistent indentation, proper line breaks, and a uniform coding style throughout.

The program leverages appropriate syntax and conventions, adhering to best practices in programming. Implemented data structures are designed programmatically, allowing stored variable values to be efficiently utilized across other methods. Method names are action-oriented verbs, reflecting the operations they perform. Logical structures, such as **IF-ELIF blocks**, include comprehensive handling of all cases with **ELSE** or **DEFAULT** clauses, ensuring robust and predictable program behavior.

Text

Description automatically generated

*Figure 6 App Source Code IF-ELIF Example*

As a document database, **MongoDB** offers robust capabilities for managing large volumes of data and simplifies the storage of both structured and unstructured data. By utilizing a JSON-like format to store documents, MongoDB seamlessly maps data to native objects in modern programming languages, eliminating the need for complex data normalization.

The web application's **CRUD functionality** is transparent and straightforward, but the development of the dashboard using the **Dash framework** proved more time intensive. Significant effort was invested in understanding and exploring the functionality of Dash core components, HTML elements, and callbacks to create an efficient and well-structured codebase. The framework's complexity and its actively evolving libraries require frequent updates and installations to ensure compatibility and stability.

Recreating the web application in a different environment involved updating and enhancing the application documentation. This updated guide provides clear and comprehensive instructions for reproducing the web app in either a Linux or Windows environment. The process of creating detailed, coherent, and technically sound documentation has further refined skills in crafting professional written and visual communication that is well-adapted to specific audiences and contexts.

**Reference:**

 **MongoDB Documentation**: MongoDB, Inc. (n.d.). MongoDB manual. Retrieved February 9, 2025, from <https://www.mongodb.com/docs/>

 **Dash Framework Documentation**: Plotly. (n.d.). Dash Python user guide & documentation. Retrieved February 9, 2025, from https://dash.plotly.com/

 **Software Development Best Practices**: McConnell, S. (2004). Code complete: A practical handbook of software construction (2nd ed.). Microsoft Press.

 **Python Programming Standards**: Van Rossum, G., & Drake, F. L. (2009). Python 3 reference manual. CreateSpace.