5-2 Milestone Four: Enhancement Three: Databases – Narrative

CS-499

Southern New Hampshire University

Steve Edmund

**Artifact Description:**

The Grazioso Salvare project was created in May and June of 2022. It is an international rescue-animal training company that collaborates with the Austin Animal Center, a non-profit agency operating multiple animal shelters around Austin, Texas. Together, they have unveiled a web application designed to aggregate and sustain data from the Austin Animal Center. This digital platform provides Grazioso Salvare with a user-friendly interface, facilitating the identification of potential rescue animals that fit specific criteria. The initiative, championed by Global Rain, accentuates rescue animals' paramount role in various life-saving scenarios, ensuring efficient and targeted selection of breeds and ages suitable for different rescue missions.

The underlying architecture of this application harnesses MongoDB for data management, empowered by the Pymongo driver, to seamlessly execute CRUD (Create, Read, Update, Delete) operations. Dash, a dynamic Python library, infuses the dashboard with its interactive capabilities, offering a responsive data visualization experience. The Python library Pandas further enhances data manipulation, laying the groundwork for robust data frame structures integral to the application's performance.

**Inclusion of Artifact in E-Portfolio:**

This artifact was added to the E-Portfolio for multiple reasons. A primary concern is the security of hardcoded data in an HTML file, which presents a significant risk of data leaks or breaches. Using a database ensures better data security. Additionally, the Django framework simplifies data retrieval from the database by auto-generating queries, eliminating the need for developers to write extra code for data loading and access. Storing data in a database also facilitates access, especially if other applications on the same server need the same data. The PostgreSQL database, a client-server system, can serve multiple clients on a single server. This structure offers the flexibility to introduce more applications to the server, allowing them to either share an existing database or create new ones based on specific needs.

To integrate the PostgreSQL database, the Django application was first set up to connect to the database using the appropriate port and login credentials. Typically, PostgreSQL requires schema and table creation configuration, then establishing a connection with the Django app and running migration commands to transfer the data model. However, using the Replit app streamlined this process by providing access to a pre-configured database and facilitating auto-connection to the Django app. Further, the frontend of the Django Admin dashboard was enhanced with a button for CSV file uploads, enabling users to transfer data directly to the database with a single click.

Introducing PostgreSQL significantly enhanced the application, making it simpler to incorporate new features based on the available data and its presentation. The database addition boosted the app's portability and scalability and opened doors for expanding the existing server or refining the current application with PostgreSQL. User accessibility, a vital aspect of GUI-based application development, was addressed by customizing the Django Administration dashboard to support large data file uploads directly to the database. This direct data upload capability paved the way for new feature additions, like filtering options and a map display that visualizes animal locations based on longitude and latitude. All these features rely on data directly sourced from the database.

**Meeting Course Objectives:**

This artifact fulfills the requirements of the fourth milestone, which focuses on integrating a database into the Django application. To achieve this, the Django application was refined through data structuring and database configuration at both the application and database levels, ensuring a smooth data flow between the two. We opted for the PostgreSQL Database due to its inherent scalability advantages. The improvements brought about by this artifact align with the objectives and outcomes of the following courses:

1. DAD 220: Introduction to SQL
2. DAT 220: Fundamentals of Data Mining
3. CS 340: Advanced Programming Concepts

**Enhancing the Artifact:**

Upgrading and adapting the artifact by incorporating a PostgreSQL database into the Django application offered invaluable insights and posed several challenges. A primary lesson was the criticality of selecting an appropriate database management system (DBMS) tailored to a project's specific demands. While Django's default SQLite database suffices for smaller applications, shifting to PostgreSQL became imperative for scalability and bolstering data security. This choice illuminated the long-term ramifications inherent in database selections during software development.

The meticulous configuration and organization of the PostgreSQL database were pivotal for fluid data exchange between the frontend and backend. Grasping database schema intricacies and crafting tables that mirrored the application's data blueprint presented a learning trajectory. It highlighted the essence of prudent planning and vision in database structuring to brace for future evolutions and data augmentation.

Another enlightening aspect was customizing the Django Admin dashboard to encompass a CSV file upload function. While this adjustment notably amplified user accessibility and expedited data entry, it necessitated a thorough understanding of Django's administrative customization potential. This served as a testament to the paramountcy of user experience in software crafting, reiterating that even nuanced modifications can profoundly uplift usability.

A more profound appreciation for data security nuances was also gleaned. Opting for database storage over embedding data in HTML files curtailed data leakage or breach threats. This accentuated the importance of championing data safeguarding in software ventures, especially when handling delicate data. Moreover, database incorporation facilitates convenient data retrieval, fostering the seamless integration of novel application features to enhance user-friendliness.

Encountering challenges was part and parcel of the journey. For instance, orchestrating the PostgreSQL database and ensuring its harmonious liaison with the Django application proved testing. Although leveraging a preconfigured database via Replit eased this dimension, it spotlighted the intricate challenges potentially lurking in database initialization and oversight.

The artifact's enhancement underscored the pivotal nature of judicious database choice, schema formulation, user accessibility, and data safeguarding in software creation. It highlighted the indispensable nature of flexibility and problem-solving acumen when integrating fresh functionalities and systems into a pre-existing endeavor.