FINAL REPORT

CMPSC 431W Database Application

DECEMBER 8, 2024

RAMY MOHAMMED, JOSIAH KUTAI

Table of Contents

Final Report

Introduction

Our One-Piece Database Project is a robust application developed over two months by [repent-for-your-sintax][1] as part of CMPSC 431W's required coursework. Its purpose was to create an interactive and highly detailed database system inspired by the expansive world of *One Piece*. This project allowed us to apply concepts from database management and other computer science principles we learned, while also engaging with the lore our beloved anime and manga series.

Project Objectives

1. Built a comprehensive database to store and manage information about *One Piece* characters, Devil Fruits, swords, and organizations.
2. Provided a user-friendly GUI for inserting, updating, deleting, querying data, and other functions we promised.
3. Implemented advanced features such as transactional operations with rollback and analytical reporting.
4. Ensured data integrity and consistency and maintained user engagement through a visually appealing and intuitive interface.

Team Contract and Effectiveness

At the start of the project, we created a team contract to outline our responsibilities and expectations. This contract was pivotal in guiding our collaboration and ensuring that we met our goals efficiently.

Contract Highlights

1. **Behavioral Expectations**:
    - Both of us prioritized punctuality and ensured all work was submitted on time.
    - We also both emphasized organization and kept track of the team's progress throughout the project.
2. **Skill Strengths**:

[1] Ramy Mohammed and Josiah Kutai

- **Josiah Kutai**: Leveraged SQL expertise from an internship to design and optimize database queries.
- **Ramy Mohammed**: Focused on drafting technical documentation and testing the application for functionality.

3. **Participation and Communication**:
   - Both team members agreed to contribute equally by dividing responsibilities into brainstorming, coding, testing, and documentation.
   - Frequent text communication and daily in-person meetings ensured seamless progress.
4. **Conflict Resolution**:
   - We agreed to respect each other's approaches and resolve conflicts collaboratively.
5. **Progress and Assessment**:
   - Regular review dates helped us evaluate milestones and make adjustments as needed.
   - Both members worked on all aspects of the project together, reflecting equal and shared effort.

Effectiveness of the Contract

The contract proved to be highly effective in maintaining clarity and focus throughout the project. While one member contributed more to the coding and database management, the other took on a significant role in brainstorming, testing, and drafting essential documents. This balance ensured the project was completed successfully and on time, showcasing the total effort by both team members.

Code Review and System Overview

The project comprises several key components, each fulfilling specific roles within the application. Here's how they come together:

1. app.py

The main script integrates all components of the application into a Kivy/Tkinter-based GUI. It provides functionality for:

- **Data Management:** Insert, update, and delete operations across multiple database tables.
- **Data Navigation:** Pagination and sorting options for large datasets.
- **Reports:** Accessible tabs to generate and display detailed analytical reports.

2. **build tables.txt**

This SQL file establishes the schema for the database, defining tables such as:

- *Person*: Stores basic character information.
- *Devil_Fruit*: Details about Devil Fruits and their attributes.
- *Groups* and *Membership*: Manages affiliations among characters, crews, and marine ranks.
- *Artifact*s: Tracks legendary swords and possessions.

The relationships among these tables were carefully designed to allow for complex queries and ensure referential integrity.

3. **conn.py**

Handles database connectivity using the *psycopg2* library. By abstracting connection logic, it ensures seamless integration with the database across all application modules. This does require user's IP address to be permitted for security reasons.

4. **df.py**

Focused on Devil Fruits, this module:

- Allows users to view a comprehensive list of Devil Fruits.
- Generates reports detailing their type, awakening status, known users, and abilities.
- Integrates image handling for visual engagement.

5. **person_table.py**

Enables users to:

- View the Person table with options for pagination and column sorting.
- Update individual cells directly from the GUI.
- Navigate large datasets efficiently and easy to update.

6. **report.py**

Provides detailed character reports that combine data from multiple tables. For example:

- A report on a character includes their Devil Fruit, affiliations, swords, and possessions.
- Combines database queries and user-friendly presentation for comprehensive insights.

Accomplishments

1. **Database Structure and Functionality**:
    - Normalized schema, ensuring data consistency and reducing redundancy.
    - Successfully implemented foreign key relationships between tables like *Devil_Fruit* and *Person*, allowing for detailed association reports.
2. **Graphical User Interface**:
    - Built an intuitive interface using Kivy and Tkinter.
    - Integrated features like dropdown menus, input validation, and dynamic report generation.
3. **Advanced SQL Queries**:
    - Designed complex queries to retrieve data across five or more tables, such as generating reports on pirate crews or Devil Fruit users.
    - Optimized queries for performance, even with large datasets.
4. **Transactional Integrity**:
    - Implemented multi-step transactions with rollback capabilities.
5. **Inspirational Context**:
    - Leveraged extensive research into *One Piece* lore using resources like the One Piece Wiki and ListFist for realistic and engaging data for the database.

Challenges

1. **Complex Relationships**:
    - Designing a schema to represent intricate relationships between pirates, Devil Fruits, and organizations required extensive testing and refinement.
2. **Data Volume**:

    - Handling large datasets in the GUI while maintaining performance was challenging, especially for sorting and pagination. Not to mention, actually populating the database with the data.

3. **GUI Development**:
    - Developing a visually appealing and user-friendly interface with Kivy demanded significant time and iterations. This came easy with our GUI tool.

Outcome

This project successfully achieved its objectives, resulting in a comprehensive and engaging application. Users can explore *One Piece* lore interactively, gaining insights into characters, Devil Fruits, and legendary artifacts. The integration of transactional features, detailed reports, and a polished GUI makes this project a standout example of combining database theory with practical application. However, this wasn't without obstacles. We had many problems addressing the look of our application, exception handling, preventing injections, GUI code generation, and hashing the password. If we had more time this could have been polished more, but this is our final product.

Acknowledgments

- **Research Sources**:
    - One Piece Wiki
    - List of Devil Fruits on Reddit
    - Cross Guild Information
- **Tools**:
    - Labdeck's Python Designer for generating our GUI (saves us time coding this out).
- **CMPSC 431W Course**:
    - For providing the foundational knowledge of database systems.

Conclusion

The One-Piece Database project is a testament to the power of collaboration and dedication. It bridges the gap between theoretical database concepts and real-world application, offering a tool that is as informative as it is entertaining.