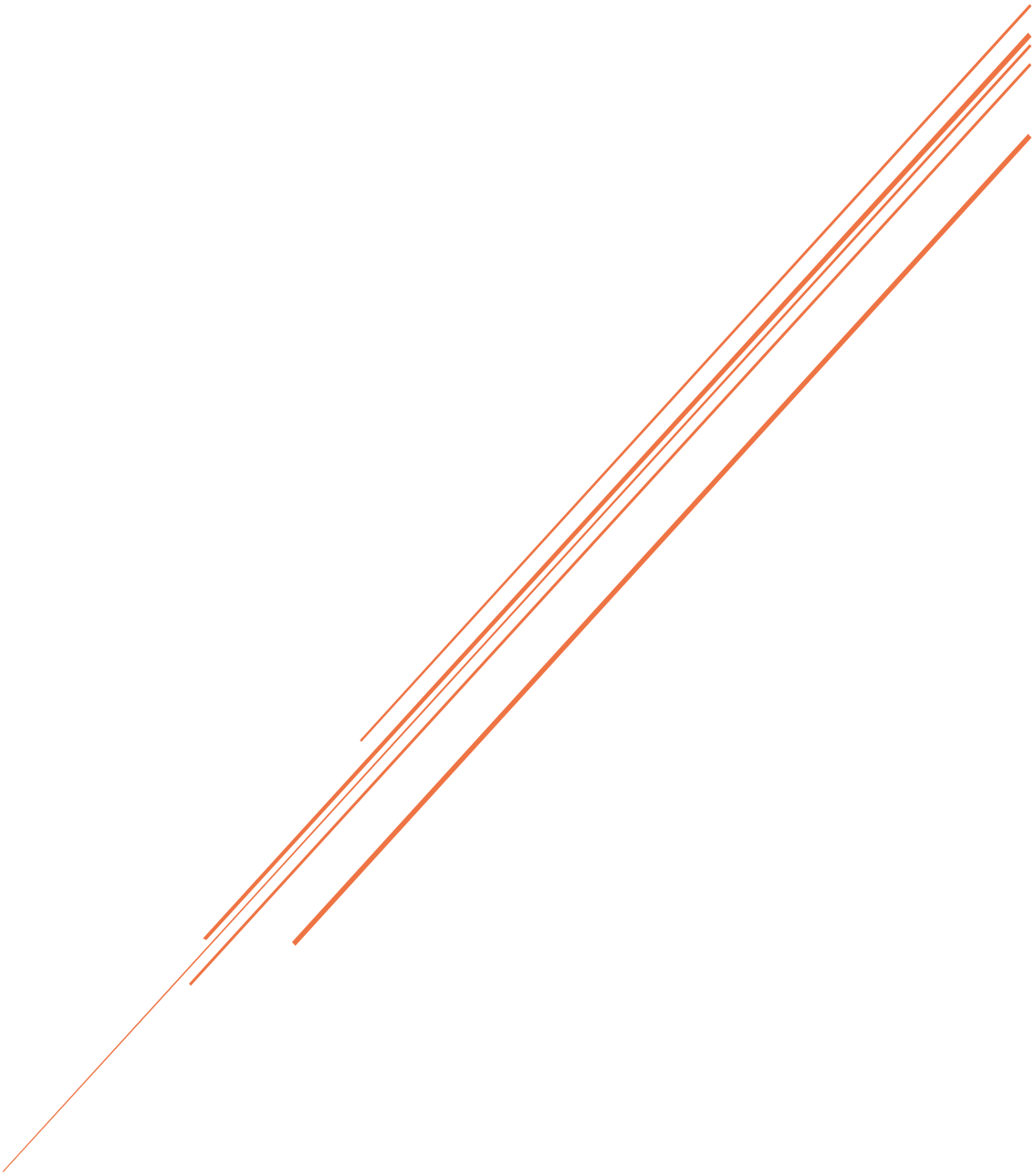
MTCG Protocol

BIF/3B

Alen Asanov



SWEN

1) Design:

Database Design:
I designed the PostgreSQL database to store user information, cards, decks and trading deals.

REST API Design:
I implemented RESTful endpoints for user authentication, card management, trading, battling and many more.

Security Measures:
Ensured token-based authentication to secure user actions on the server-side. Used HTTPS to encrypt data during transmission.

Game Mechanics:
Implemented the game mechanics as described, including card types, battles and trading deals.

Scalability:
Ensured that the design allows for future scalability, such as adding new features or extending existing functionalities.

2) Lesson learned:

Challenges with Game Logic:
Implementing the game logic, especially handling different card types and their interactions, posed challenges.

Database Persistence:
Ensured proper persistence of data in the PostgreSQL database. Learned to manage relational data efficiently.

HTTP/REST Implementation:
Developed a server using HTTP/REST without relying on helper frameworks, which required a solid understanding of HTTP methods and status codes.

Integration Testing:
Implemented integration tests using the provided curl script and Postman to ensure the correct functioning of the entire system.

Unit Testing:
Created comprehensive unit tests to validate individual components and functions, ensuring the reliability of the codebase.

3) Unit Test Design:

Overview:
The unit test design for the Monster Trading Cards Game project was driven by the need to ensure the correctness and reliability of various components, including the database operations, game logic and utility functions. The tests were structured to cover a wide range of scenarios, from basic functionality to edge cases.

Testing Framework:
The unit tests were implemented using the NUnit testing framework, chosen for its simplicity, versatility and compatibility with C# projects. NUnit provides a clean syntax for writing tests and offers features such as assertions and setup/teardown methods, facilitating effective testing.

Test Organization:
Tests are organized into a dedicated Tests class, with individual methods covering specific functionalities or components. The structure of the tests aligns with the modular architecture of the application, allowing for easy identification and isolation of test cases.

Test Setup:
The [SetUp] attribute is utilized to define a setup method that initializes the database, users and cards required for testing. This ensures a consistent and controlled environment for each test.

4) Test Cases:

Authentication Tests:

- Verify user registration and login functionalities.
- Test the system's response to empty or null user credentials.

Database Operations Tests:

- Validate database retrieval operations for users and cards.
- Test the addition and updating of user and card information.
- Verify trade-related database operations.

Game Logic Tests:

- Test the battle logic and ensure correct player Elo updates.
- Validate the response format of the battle log.
- Test the utility function for extracting card types from names.

Utility Function Tests:

- Validate the utility function for checking if a card is a spell.
- Test the function that extracts card types from names.

Coverage:
The unit test suite aims to achieve comprehensive coverage by testing various scenarios, including edge cases and unexpected inputs. Regular reviews and updates to the test suite are planned to maintain coverage as the codebase evolves.

Conclusion:
The unit test design is structured to provide thorough coverage of critical components, ensuring the reliability and correctness of the Monster Trading Cards Game project.

5) Time Spent:

The time spent on the Monster Trading Cards Game project was carefully tracked throughout its development. The breakdown of time allocation is as follows:

Design and Planning: 20 hours

Included architectural decisions, database schema design and outlining the overall project structure.

Implementation: 70 hours

Actual coding and development of the HTTP/REST server, game mechanics and database interactions.

Testing and Debugging: 20 hours

Dedicated time to write unit tests, perform integration testing with the provided curl script and Postman and debug issues.

Documentation: 5 hours

Time spent on creating this protocol and other necessary documentation.

Total Time: 115 hours ~ 55 Git Commits (1 Commit = ~1 - 3h)

6) Link to Git Repository:

The Git repository for the Monster Trading Cards Game project is available at the following link: https://github.com/Alino22Git/MonsterTradingCards

Note: The Git history serves as a valuable part of the documentation, capturing the evolution of the codebase over time. It is not duplicated within this protocol and reviewers are encouraged to refer to the Git repository for a detailed history of commits and changes.