**Q:** **Online game - Design a class to manage an online game, including character creation, game mechanics, and player interactions.**

**Approach to solve:**

1.Understand the problem statement clearly and find out the all the end points need to be created .

2.Figure out the proper datasturctures to manipulate the data in the api

3.Figure out the database design with proper list of tables and appropriate constraints to solve the problem.

**Pseudocode:**

**CREATE\_CHARACTER (HTTP POST method):**

Step-1: Extracts the required data from the JSON payload.

Step-2: If the data format is not correct or any required field is missing, a message saying "Invalid input data" is returned with status code 400.

Step-3: Database connection is created using the set\_connection() method. If it is created successfully, the extracted data is inserted into the characters table, and changes are saved.

Step-4:If any exception is raised, the transaction is rolled back, and a message is returned saying "Error creating character" with status code 500.

Step-5:If the character is created successfully, a message saying "Character created successfully" is returned with status code 201.

**GET\_CHARACTER (HTTP GET method):** retrieves the character with the given name from the database.

Step-1: Extracts the name of the character from the URL.

Step-2: Database connection is created using the set\_connection() method. If it is created successfully, the character with the given name is retrieved from the characters table.

Step-3:If the character does not exist, a message saying "Character not found" is returned with status code 404.

Step-4:If the character is found, the character information (name, strength, agility, and intelligence) is returned as a JSON object with status code 200.

Step-5:If any exception is raised, a message saying "Error getting character" is returned with status code 500.

**ATTACK (HTTP POST method):** simulates an attack between two characters in the database.

Step-1: Extracts the required data (attacker and defender names) from the JSON payload.

Step-2: If the data format is not correct or any required field is missing, a message saying "Invalid input data" is returned with status code 400.

Step-3: Database connection is created using the set\_connection() method. If it is created successfully, the attacker and defender data are retrieved from the characters table. If any of the characters do not exist, a message saying "Attacker/Defender not found in the database" is returned with status code 404.

Step-4: The damage is calculated by subtracting the defender's intelligence from the attacker's agility. If the damage is negative, it is set to 0.

Step-5: The damage is applied to the defender's intelligence in the database. If the defender's intelligence becomes zero or less, the defender is removed from the characters table. If the defender is removed, a message saying "Attack successful" with damage, defender name, and status ("dead") is returned with status code 200.

Step-6:If the defender is not removed, a message saying "Attack successful" with damage, defender name, and status ("alive") is returned with status code 200. If any exception is raised, a message saying "Error attacking" is returned with status code 500.