Service Layer Re-Design Document

Andrew Dillard

07/21/2024

Overview

The backend service for my web application will also be built using the Express library from Node.js as it provides a REST-ful API for interacting with game session data. This service layer ensures separation between the front-end UI and the database.

Specifications

Game Route

Given the arcade-inspired design of the game, a GET method would really only be used to fetch game session data and scores from the backend before the game starts. While a POST method here would be used to submit player names and scores at the end of game play.

Get High Scores

- Method: GET
- URL: /api/user_scores
- Purpose: Fetches the top 5 usernames and high scores from the database.
- Example Request: GET /api/high-scores
- Success Response:

```
{
    "success": true,
    "highScores": [
      { "username": "Alice", "score": 180 },
      { "username": "Billy", "score": 160 },
      { "username": "Charlie", "score": 150 }
    ]
  }
```

- Error Response:

```
{
    "success": false,
    "error": "Database error"
  }
```

Get Words

- Method: GET
- URL: /api/words
- Purpose: Fetches from the list of words and their hints from the database for use in the game.
- Example Request: GET /api/words

- Success Response:

```
{
    "success": true,
    "words": [
      { "word": "VENUS", "hint": "The second planet from the sun" }
    ]
  }
```

- Error Response:

```
{
    "success": false,
    "error": "Failed to load words."
  }
```

Post-Game

- Method: POST
- URL: /api/user_score
- Purpose: Saves the player's name and score to the database.

- Example Request:

```
POST /api/user_score
Content-Type: application/json

{
  "username": "Alice",
  "score": 150
}
```

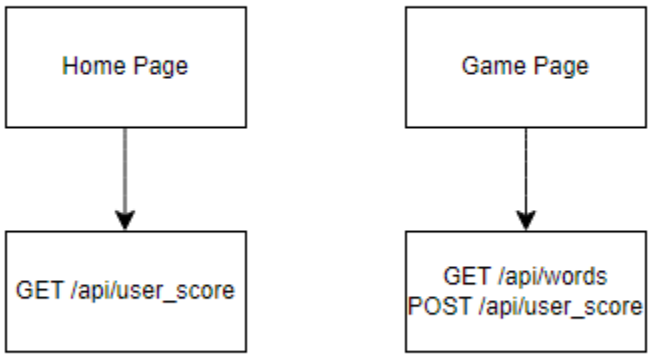
- Error Response:

```
{
    "success": false,
    "error": "Database error"
  }
```

Given the nature of the gameplay, the PUT and DELETE methods may not align with this game as those methods are primarily used to *update* or *delete* an existing resource (i.e. player name, scores) – which, in this case, the game will not need to do.

Finally, since the game doesn't require user profiles, logins, or passwords, the User Route is simplified to handle only the submission of a player name.

Diagram of User Interface Pages and Service Endpoints



Endpoint Interaction with Pages

Home Page

- Endpoint Interaction:
    - GET /api/user_score: To retrieve all previous high scores for previous players to display on the home page.

Game Page

- Endpoint Interaction:
    - GET /api/words: To retrieve a word from the list of the words used for the game.
    - POST /api/user_score: Submit the game data with the player's name and score at the end of the game.