Design: Service Layer(s)

Brandon Schmidt

Maryville University

SWDV 691 Capstone

Joseph Gradecki

March 23, 2025

Design: Service Layer(s)

For this service design document, we are going to talk about the services for the bowling site. I'll be using Node.js, express frameworks, and rest API. Rest API should be able to do the handling of the requests of the front end and then as well as interacting with the DB talked about in the DB design doc. There will be Routes, Controllers, and services. Routes will deal with the frontend. Controllers should handle the http requests and the logic behind the processing in the service layer. And then finally Service will contain business logic that will communication with the DB to help with data manipulation.

API Endpoints

User Route

1. **User Registration**
    - **Method:** POST
    - **URL:** /api/users/register
    - **Purpose:** Allows new users to sign up by providing credentials and profile information.
    - **Request Example:**

      curl --request POST \

      --url http://localhost/api/users/register \

      --header 'Content-Type: application/json' \

      --data '{"username": "jon_snow", "email": "jon@example.com", "password": "securePizza123"}'
    - **Success Response:** { "message": "User registered successfully" }
    - **Error Response:** { "error": "Email already in use" }
2. **User Login**
    - **Method:** POST

- **URL:** /api/users/login
- **Purpose:** Authenticates a user and generates session.
- **Example:**

```
curl --request POST \
--url http://localhost:5000/api/users/login \
--header 'Content-Type: application/json' \
--data '{"email": "jon@example.com", "password": "securePizza123"}'
```

- **Success Response:** {“message": "Login successful" }
- **Error Response:** {“error": "Invalid credentials" }

3. **Get User Profile**
   - **Method:** GET
   - **URL:** /api/users/:id
   - **Purpose:** Retrieves user details.
   - **Example:**

```
curl --request GET \
--url http://localhost:5000/api/users/1 \
--header 'Cookie: session_id=your_session_cookie_here'
```

   - **Success Response:** { "id": 1, "username": "jon_snow", "email": "jon@example.com" }
   - **Error Response:** { "error": "User not found" }

Game Route

1. **Start New Game**
   - **Method:** POST
   - **URL:** /api/games/start

- **Purpose:** Creates a new game session for a user.

2. **Get Game Details**
    - **Method:** GET
    - **URL:** /api/games/:id
    - **Purpose:** Retrieves details of a specific bowling game.

Score Route

1. **Submit Score Entry**
    - **Method:** POST
    - **URL:** /api/scores/submit
    - **Purpose:** Allows users to record their bowling scores.
    - **Example:**

```
curl --request POST \
--url http://localhost:5000/api/scores/submit \
--header 'Content-Type: application/json' \
--header 'Cookie: session_id=your_session_cookie_here' \
--data '{"gameId": 101, "frame": 1, "score": 8}'
```

    - **Success Response:** { "message": "Score recorded" }
    - **Error Response:** { "error": "Invalid game ID" }
2. **Get Score Summary**
    - **Method:** GET
    - **URL:** /api/scores/user/:id
    - **Purpose:** Retrieves all scores associated with a user.

Leaderboard Route

1. **Get Top Players**

- **Method:** GET
- **URL:** /api/leaderboard
- **Purpose:** Returns a list of top-ranking players.

Alley Locator Route:

1. **Search for Alleys**

- **Method**: GET
- **URL**: /api/alleys?location={city}
- **Purpose**: Returns a list of bowling alleys based on a city or zip code.
- **Successful Response**:

{

"alleys": [

{ "id": 1, "name": "1st Street Bowling", "address": "123 1st St, City", "phone": "555-1234" },

{ "id": 2, "name": "Lannister Lanes", "address": "456 Kingsroad , Kings Landing", "phone": "213-675-4456" }

]

}

2. **Get alley Details:**

- **URL**: /api/alleys/:id
- **Purpose**: to be able to retrieve details about a specific bowling alley pulled up.

{

“id”: 1,

“name”: “Lannister Lanes” ,

“address”: “456 Kingsroad, Kings Landing”,

“phone”: “213-675-4456”

“open_hours”: “9am-11pm”

}

Database Interaction

So, with using PostgreSQL for the DB, the service should be able to interact with it in ORM. This should allow for optimal data management. The layer should take care of input validation, errors, and potentially security measures like authentications and authorizations.

UI Interaction gram

UI Interaction Diagram

Below is the Frontend and backend interactions:

- **Registration Page** → POST /api/users/register
- **Login Page** → POST /api/users/login
- **User Profile Page** → GET /api/users/:id
- **Game Start Page** → POST /api/games/start
- **Game Details Page** → GET /api/games/:id
- **Score Entry Page** → POST /api/scores/submit
- **Score Summary Page** → GET /api/users/:id/scores
- **Leaderboard Page** → GET /api/leaderboard

References

Node.js. (n.d.). *Node.js*. https://nodejs.org/en

PostgreSQL Global Development Group. (n.d.). *PostgreSQL*. https://www.postgresql.org

For additional information on APA Style formatting, please consult the APA Style Manual, 7th Edition.