Design: Database

Brandon Schmidt

Maryville University

SWDV 691 Capstone

Joseph Gradecki

March 23, 2025

Design: Database

This is going to be the Database portion of my Bowling Site. It's mainly going to be built around the user creation and storing as well as all the portions consisting of score tracking and saving. I am choosing to use PostgreSQL for this as it should work a lot better than say MongoDB since there should be many to many relationships in this DB. The Db will consist of player information, game sessions, score entries, as well as some user authentication.

The web application will interact with PostgreSQL through a REST API, which will serve as the intermediary between the frontend and the database. The backend will be developed using ASP.NET Core with Entity Framework Core (EF Core) for ORM-based database interactions

Database Schema and Table Definitions

The application will consist of the following tables:

1. User Table

Stores information about users who access the system (players and admins).

```
CREATE TABLE user (
   user_id SERIAL PRIMARY KEY,
   full_name VARCHAR (100) NOT NULL,
   email VARCHAR (255) UNIQUE NOT NULL,
   password_hash TEXT NOT NULL,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

2. Game Table

Represents a single bowling session for a player. Each game is linked to a user.

```
CREATE TABLE game (
```

```
    game_id SERIAL PRIMARY KEY,
    user_id INT REFERENCES user(user_id) ON DELETE CASCADE,
    game_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    total_score INT DEFAULT 0
);
```

3. Frame Table

Each game consists of 10 frames. This table stores frame-wise scores.

```
CREATE TABLE frame (
    frame_id SERIAL PRIMARY KEY,
    game_id INT REFERENCES game(game_id) ON DELETE CASCADE,
    frame_number INT CHECK (frame_number BETWEEN 1 AND 10),
    first_roll INT CHECK (first_roll BETWEEN 0 AND 10),
    second_roll INT CHECK (second_roll BETWEEN 0 AND 10),
    bonus_roll INT CHECK (bonus_roll BETWEEN 0 AND 10),
    frame_score INT DEFAULT 0
);
```

4. Score Table

Stores cumulative scores for each frame within a game.

```
CREATE TABLE score (
    score_id SERIAL PRIMARY KEY,
    game_id INT REFERENCES game(game_id) ON DELETE CASCADE,
    frame_id INT REFERENCES frame(frame_id) ON DELETE CASCADE,
    cumulative_score INT NOT NULL
);
```

5. Setting Table

Allows users to customize settings like game mode, difficulty, etc.

```
CREATE TABLE setting (
   setting_id SERIAL PRIMARY KEY,
   user_id INT REFERENCES user(user_id) ON DELETE CASCADE,
   theme VARCHAR (50) DEFAULT 'light',
   sound_enabled BOOLEAN DEFAULT TRUE
);
```

This is going to be the relationships of the above tables

- **User to Game** (One-to-Many) – A user can have multiple game records.
- **Game to Frame** (One-to-Many) – A game consists of 10 frames.
- **Frame to Score** (One-to-One) – Each frame has an associated cumulative score.
- **User to Setting** (One-to-One) – Each user has personalized settings.

For the interactions for the above relationships this is how it should be.

- A Player signs up and starts a new game and enters the scores
    - That shows linking the user to a game
    - Game to Frame as well as Frame to Score
- There is also going to be some Calculations with the Score.
    - Where a Strike is 10 + the next 2 rolls
    - Spare is 10 + Next roll
    - Open frame = Sum of the 2 rolls
- There will also be some sort of admin abilities
    - Activity logs as well as ability to reset some data

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