**API Documentation**

Introduction

Since the goal of the bowling exercise is to keep score for games of bowling and enable a front-end client to perform certain tasks, I decided to implement an API using the Rails framework that accepts HTTP requests and returns JSON responses.

A game of bowling consists of 10 frames and in each frame, a player is allowed up to 2 rolls to knock down 10 pins which would result in three different outcomes:

1. **Open** **Frame**: When a player knocks down less than 10 pins, the resulting score would be equivalent to the number of pins knocked down.
2. **Spare**: When a player knocks down the rest of the pins on the second roll, the resulting score is 10 plus the pins knocked down on the next roll.
3. **Strike**: When a player knocks down 10 pins on the first roll, the resulting score is 10 plus the pins knocked down on the next 2 rolls.

The tenth frame in a game has a special case due to the spare and strike rules, if a player scores a spare or a strike in the tenth frame, the player is allowed up to two more frames to determine the final score.

In addition to Rails, I used SQLite3 to maintain the table records. To generate the database models, I used scaffolding to help generate a set of a model, views and controller. I also used migrations to alter the database. In order to transfer data through HTTP requests, I used the command line tool curl. Descriptions of each column in the models generated are listed below, along with endpoints, sample requests and responses that comprise the API.

Models

* Player
  + **name**: A unique identifier for player
* Game
  + **description**: An additional identifier for games
* GamePlayer
  + **game**: A foreign key referencing game model
  + **player**: A foreign key referencing player model
  + **current\_frame**: Keeps track of the player’s current frame
  + **game\_status**: Current game status whether ongoing or finished
  + **cumulative\_score**: A player’s total score
* FrameScore
  + **game\_player**: A foreign key referencing GamePlayer model
  + **frame\_number**: Frame number
  + **roll\_one\_score**: Number of pins knocked down for the first roll
  + **roll\_two\_score**: Number of pins knocked down for the second roll
  + **rolls\_to\_add**: Bonus roll scores to add for Strike or Spare
  + **frame\_result**: A Char field for a Strike, Spare, Open frame or Foul
  + **frame\_score**: Total score obtained in a frame

Endpoints Implemented

* **games/start**: A POST request to start a game
  + Sample parameters
    - {"game": {"description": "Sat Night", "players": ["Tom", "Dick", "Harry"]}}
  + Sample request
    - curl -H "Content-Type: application/json" -d "{\"game\": {\"description\": \"Sat Night\", \"players\": [\"Tom\", \"Dick\", \"Harry\"]}}" http://localhost:3000/games/start
* **games/getwinner/:game**: A GET request to obtain either the winner or the leading player
  + Sample endpoint with parameter
    - <http://localhost:3000/games/getwinner/1>
  + Sample response
    - {"id":1,"game\_id":1,"player\_id":2,"current\_frame":1,"game\_status":"O","cumulative\_score":10,"created\_at":"2019-07-25T21:15:30.559Z","updated\_at":"2019-07-25T21:16:17.049Z"}
* **game\_players/getresult/:game/:player**: A GET request to return GamePlayer instance
  + Sample endpoint with parameters
    - <http://localhost:3000/game_players/getresult/1/1>
  + Sample response
    - {"id":1,"game\_id":1,"player\_id":1,"current\_frame":1,"game\_status":"O","cumulative\_score":5,"created\_at":"2019-07-25T21:15:30.559Z","updated\_at":"2019-07-25T21:16:17.049Z"}
* **frame\_scores/result**: A POST request that scores a player’s roll in a frame
  + Sample parameters
    - {"result": {"game": 1, "player": 1, "frame": 1, "roll": 1, "pins": 10}}
  + Sample request
    - curl -H "Content-Type: application/json" -d "{\"result\": {\"game\": 1, \"player\": 1, \"frame\": 1, \"roll\": 1, \"pins\": 10}}" http://localhost:3000/frame\_scores/result