Assigned: 11 October

Homework #5 – League Match API

EE 547: Fall 2022

Due: Sunday, 23 October at 23:59. Late penalty: 10% per 24-hours before 25 October at 23:59. Submission instructions will follow separately on canvas.

Refactor and extend your HW #4 Node.js web application to improve currency handling and handle a new entity type.

1. **Convert fixed-digit currency strings to integers.**

   Your League-API currently use fixed-digit strings to represent currency amounts. Fixed-digit strings removed potential floating-point "rounding errors". But it complicated arithmetic operations on currency amounts and necessitated a "read and update" anti-pattern to update database values.

   Replace all fixed-digit string currency amounts with integer *centi-* values. Append `_cents` to all existing currency fields and change the datatype from string to integer (not integer string). For instance, {`total_prize_usd: "5.43"`} becomes {`total_prize_usd_cents: 543`}. Refactor database currency operations to use atomic increment and decrement in place of "read and update". Update the following:

   API Response fields
   - `Player.`{`balance_usd|total_prize_usd`}
   - `Match.`{`entry_fee_usd|prize_usd`}

   Mongo document attributes
   - `Player.`{`balance_usd`}
   - `Match.`{`entry_fee_usd|prize_usd`}

2. Extend your HW #4 Node.js web application to manage matches between players in a small sports league. Your application should handle two entities: `Player` and `Match`.

   Introduction

   Matches are contests between two players. There can be many simultaneous active matches. An active match is a match that has not ended. Players may participate in any number of matches but can be in only one active match at a time. Players must pay the entry fee to participate in a match. Players with insufficient balance to pay the entry fee cannot join a match. Players may deposit funds to increase their balance.

   The player with the most points when the match ends is the winner. The winner receives the match *prize*. Matches cannot end if both players have the same number of points (*i.e.,* no ties allowed). A match ends immediately if one player is disqualified (DQ) even if the score is tied. The other (non-disqualified) player is the winner regardless of score and receives the prize.

   A. B. API Specification

- All previous endpoints.
- `GET /match`

  Return: Array of `Matches`. Return all active matches sorted by `prize_usd_cents DESC` (*i.e.*, "largest first") followed by the four most recently ended inactive matches sorted by `end_at DESC` ("newest first").

  Response code: 200
- `GET /match/[mid]`

  Return: `Match[mid]`.

  Response code:

  - 200 if exist.
  - 404 if not exist.
- `POST /match?pid1=&pid2=&entry_fee_usd_cents=[currency]&prize_usd_cents=[currency]`

  Start a new Match. `Pid1` and `Pid2` must exist, have balance sufficient to cover the entry fee, and not be in an active match already.

  Response code:

  - 303 redirect on success to `GET /match/[mid]`.
  - 404 if player1 or player2 does not exist.
  - 409 if player1 or player2 already in an active match.
  - 402 if insufficient account balance (either player).
  - 400 else.
- `POST /match/[mid]/award/[pid]?points=[int]`

  Points must be positive integer. Player must be in the match and match must be active.

  Return: `Match[mid]`.

  Response code:

  - 200 if success.
  - 404 if player or match does not exist.
  - 409 if match not active.
  - 400 else.

- `POST /match/[mid]/end`

  End an active match. Match must exist and be active. One player points must be higher than the other player points.

  Return: `Match[mid]`.

  Response code:

  - 200 if success.
  - 404 if match does not exist.
  - 409 if match not active or points tied.

- `/match/[mid]/disqualify/[pid]`

  Disqualify a player from match and end the match. Match must be active and player must be in the match.

  Return: `Match[mid]`.

  Response code:

  - 200 if success.
  - 404 if player or match does not exist.
  - 409 if match not active.
  - 400 else.

C. Response Syntax

Use JSON for all (non-empty) responses. Use the following syntax for entity response.

```
Player[pid]
```

```
{
  pid:              string       player id
  name:             string       "fname lname" - no trailing spaces
  handed:           string       left|right|ambi
  is_active         boolean
  num_join          int          number of matches
  num_won           int          number of matches won
  num_dq            int          number of disqualifications
  balance_usd_cents int          currency
  total_points      int          total number of points, all matches
  total_prize_usd_cents  int     total prize for player (currency)
  efficiency        float        frac *completed* matches won (0-1)
  in_active_match   string|null  mid of active match, else null
}
```

Match[mid]

```
{
  mid                     string        match id
  entry_fee_usd_cents     int           currency
  p1_id                   string        player 1 id
  p1_name                 string        (see Player.name)
  p1_points               int
  p2_id                   string        player 2 id
  p2_name                 string        (see Player.name)
  p2_points               int
  winner_pid              string|null   null if active
  is_dq                   boolean       true if end in dq
  is_active               boolean
  prize_usd_cents         int           currency
  age                     int           seconds since create
  ended_at                string        ISO-8601 (date+time)
}
```

D. Mongo Document "Base" Schema

You may extend the following base document schema. You may add fields or collections to maintain additional state within the same database but DO NOT MODIFY ANY ATTRIBUTES BELOW. Ensure your script accepts any document that satisfies the base schema as a valid document and infer reasonable defaults. Use a collection called `match` to store Match documents.

```
match
```

```
{
  _id                    ObjectId      match id
  created_at             Date
  ended_at               Date|null
  entry_fee_usd_cents    int           currency
  is_dq                  boolean
  p1_id                  ObjectId      player 1 id
  p1_points              int
  p2_id                  ObjectId      player 2 id
  p2_points              int
  prize_usd_cents        int           currency
}
```