Assigned: 20 September

Homework #3 – League Player API

EE 547: Fall 2022

Due: Tuesday, 27 September at 12:00. Late penalty: 10% per 24-hours before 29 September at 23:59. Submission instructions will follow separately on canvas.

Create a Node.js web application to manage players in a small sports league. Your application should handle one entity: `Player`. Use a JSON file to persist changes.

A. Technical

- Bind web server to port 3000.
- Use file `./data/player.json` to persist player data. See file format specification (see **E.**)

B. Schema

Assume that each player is identified by a unique `player_id` and has the following attributes:

```
fname:       string
lname:       string
is_active:   Boolean
balance_usd: Decimal
handed:      Enum("L", "R", "Ambi")
```

Currency: Represent all currencies as USD without a dollar sign (*i.e.,* `xxx.xx`). Use DECIMAL numbers with exactly two precision digits (*i.e.,* dollars and cents). Numbers or strings with more than two digits are invalid. Append zeros as necessary if less than two digits, *e.g.,* `1.332` is invalid (too many digits) but `1.33`, `1.3`, `1.0`, and `1` are all valid (*i.e.,* , append zeros).

Boolean: Accept (case insensitive) `1`, `true`, and `t` as true Boolean values. Assume all other non-empty or null values are false.

Player Name: Use "`fname lname`" as player name. If `lname` is empty or `null` use only `fname`. The name must not include any trailing spaces.

C. API Specification

- `GET /ping`

  Return: [Empty]

  Response code: 204

- `GET /player`

Return: Array of all **active** `Players`. Sort by player name `ASC` (i.e., "A to Z").

Response code: 200

- `GET /player/[pid]`

  Return: `Player[pid]`

  Response code:

  - 200 if exist.
  - 404 if not exist.

- `DELETE /player/[pid]`

  Response code:

  - 303 redirect on success to `GET /player`.
  - 404 if not exist.

- `POST /player?fname=&lname=&handed=[enum]&initial_balance_usd=[currency]`

  Add a new active player. The query string may contain none, some, or all the parameters. `fname` and `lname` may contain only letters. `handed` should be one of (case-insensitive): `left`, `right`, or `ambi`. INSERT the player only if it satisfies the data schema. Else fail.

  Response code:

  - 303 redirect on success to `GET /player/[pid]`.
  - 422 error on failure: body must be string that includes all invalid field names, *e.g.,* "`invalid fields: initial_balance_usd`" or "`invalid fields: handed, fname, initial_balance_usd`", etc. You do not need to report the reason.

- `POST /player/[pid]?active=[bool]&lname=`

  Update `Player[pid]`. The query string may contain none, some, or all the parameters. UPDATE the player only if it satisfies the schema.

  Response code:

  - 303 redirect on success to `GET /player/[pid]`.
  - 422 error on failure. You do not need to report the reason.

- `POST /deposit/player/[pid]?amount_usd=[currency]`

  Add **positive** currency to `Player[pid]` balance.

  Return: `PlayerBalance[pid]`

Response code:

- 200 on success.
- 404 if player does not exist.
- 400 if invalid amount.

D. Response Syntax

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

`Player[pid]`

```
{
pid:             int      (player id)
name:            string   "fname lname", no trailing spaces
handed:          string   (left|right|ambi)
is_active:       boolean
balance_usd:     string   (currency string)
}
```

`PlayerBalance[pid]`

```
{
old_balance_usd: string   (currency string)
new_balance_usd: string   (currency string)
}
```

E. JSON Database File Format

Use file `./data/player.json` to persist player data. Assume an empty league if the file does not exist. Create the file as needed. Immediately write any player data change to the file (create, delete, update, etc.). Overwrite the existing file instead of attempting random access updates. Use the following format.

```
{
players:         [Player] (array of players)
update_at:       string   (ISO-8601)
create_at:       string   (ISO-8601)
version:         string   "1.0"
}
```

`Player`

```
{
  pid:              int      (player id)
  fname:            string
  lname:            string
  handed:           string   (L|R|A)
  is_active:        boolean
  balance_usd:      string   (currency string)
}
```

F. Hints and points to investigate

- HTTP response code 303 redirect / location
- checkExist, mkdir (do not create until write)
- express routing with regular expressions and variable parameters
- Implement a DataSource interface class to abstract read and write operations. This will be helpful for HW 4.

```
class PlayerSourceJson {
  constructor(file) {}
  getPlayer(pid) {}
  createPlayer({...})
  updatePlayer(pid, {...}) {}
  deletePlayer(pid) {}
  getBalance(pid) {}
  getPlayers(...) {}
}
```