Coding Assignment - API Server

Description

You've accepted a freelance project to write an HTTP API server for Contrived Example University's admission system. The university has different colleges, each with different admittance criteria. Your API will include endpoints to submit applications and list applications in different formats. There will also be a maintenance endpoint to backup the application information to the local disk of the server running the API.

Running the server

Assuming we're in the project directory, the command to start the server should be

```
./contrived backup.json 3210
```

where the first argument is the relative or absolute path to the backup file and the second argument is the port number the server will listen on.

If we need to run some commands to install dependencies (etc.), please provide us with instructions.

Route specification

All requests & responses are assumed to have an `application/json` content type.

Responses may be unformatted or formatted in whatever way you'd like. Example responses here are formatted for clarity.

`POST /applications`

Submits an application to the university.

This route expects a JSON object with the following properties:

- `college` - String representing the name of the college (ex. "CompSci", "Business")

- `name` - String representing the name of the applicant (ex. "Alice", "Bob")
- `score` - Integer between 0-100 (inclusive)

All properties are required. If the object provided doesn't match the schema above, an HTTP 400 response should be returned with an informational message stating what was wrong with the request.

Applicants can apply to more than one college, but cannot apply to a college more than once. If an application has already been submitted for a given `college` / `name` pair, an HTTP 400 response should be returned with the following properties:

- `statusCode - 400`
- `error - "Bad request"`
- `message - "Application already submitted for this college/name pair"`

If the request was success, an HTTP 200 response should be returned with the following properties:

- `statusCode - 200`
- `message - "Application submitted successfully"`

Example application:

```
{
    "college": "CompSci",
    "name": "Alice",
    "score": 100
}
```

`GET /applicants`

Returns an object where the applicants' names are the keys, and the values are arrays of applications submitted by that applicant.

The arrays of applications should exclude the `name` field, and should be sorted by descending `score` value.

Example response:

```
{
  "Alice": [
    {
      "college": "CompSci",
      "score": 100
    },
    {
      "college": "Business",
      "score": 90
    }
  ],
  "Bob": [
    {
      "college": "CompSci",
      "score": 95
    },
  ]
}
```

`GET /applicants/{name}`

Returns an object with the following properties:

- `name` - Name of the applicant as provided
- `applications` - Array of applications for that applicant excluding the `name` field

The `applications` array should be sorted by descending `score` value.

If there are no applications for the given name, an HTTP 404 response should be returned with the following properties:

- `statusCode - 404`
- `error - "Not found"`
- `message - "No applications exist for that name"`

Example successful response:

```
  {
    "name": "Alice",
    "applications": [
      {
        "college": "CompSci",
        "score": 100
      },
      {
        "college": "Business",
        "score": 90
      }
    ]
  }
```

`GET /colleges`

Returns an object where the colleges' names are the keys, and the values are arrays of applications submitted to that college.

The arrays of applications should exclude the `college` field, and should be sorted by descending `score` value.

Example response:

```
{
  "CompSci": [
    {
      "Name": "Alice",
      "score": 100
    },
    {
      "Name": "Bob",
      "score": 95
    }
  ],
  "Business": [
    {
      "name": "Alice",
      "score": 90
    }
  ]
}
```

`GET /colleges/{name}`

Returns an object with the following properties:

- `college` - Name of the college as provided
- `applications` - Array of applications for that college excluding the `college` field

The `applications` array should be sorted by descending `score` value.

If there are no applications for the given college, an HTTP 404 response should be returned with the following properties:

- `statusCode - 404`
- `error - "Not found"`
- `message - "No applications exist for that college"`

Example successful response:

```
{
  "college": "Business",
  "applications": [
    {
      "name" : "Alice",
      "score": 90
    }
  ]
}
```

`POST /backup`

Write a JSON file to disk using the path provided when starting the server.

The format should be the same as the response from `GET /colleges` .

On success, an HTTP 200 response should be sent with the following properties:

- `statusCode - 200`
- `message - Backup successful`

How we'll evaluate your solution

- Can we run your server?
- When we run the `tests.sh` command included with this file, does it return the expected results?
- Is the code clean and idiomatic?
- Is there clear separation of concerns?

Other rules & hints

- Solutions in node.js are strongly preferred, then Python, but you may use whatever platform you'd like as long as you provide detailed instructions on how to run your server.
  - Extra credit is awarded for using https://github.com/hapijs/hapi
- You don't have to code everything from scratch; feel free to use whatever libraries you'd like.
  - If you're using hapi, check out hapijs/joi and hapijs/boom

- You don't need to optimize for performance, cleaner code is better.

Good luck!