Backend Project

Today you will use the Tate Modern art data set to create an API that allows the user to create users, view art data and create a comment for each art entry.

Endpoints:

- /api/art - GET, view the entire art data set
- /api/art/ID - GET, view art data by ID
- /api/art/ID/comments - POST, add a comment for an art data entry
- /api/users - POST, create user
- /api/users - GET, see all users

You can use any language or web frameworks. But please remember to add a README with instructions on running project.

Import data

Import the Tate collection csv into your preferred relational database and create 3 tables: art, comments, users. You'll need to define the schema of each table based on the information below.

/api/art

Return JSON object example:

```
[{
        id: 10000,
        title: “Poppies”,
        artist: “Monet”,
        year: 1873,
        comments: []
},
{
        id: 10001,
        title: “Woman with the parasol”,
        artist: “Monet”,
        year: 1875,
        comments: [
                {
                        id: 10000,
                        name: “John”,
                        content: “This is rad”
                },
                {
                        id: 10001,
                        content: “This is super cool”,
                        name: “Allison Johnson”,
                        userID: 10000
                },
        ]
}]
```

/api/art/ID

Return JSON object example:

```
{
        id: 10000,
        title: “Poppies”,
        artist: “Monet”,
        year: 1873,
        comments: []
}
```

/api/art/ID/comments

Data to send when creating a new comment

- userID: STRING - optional
- name: STRING - required if there no user ID is sent,
- content: STRING, required

Logic

- Each art entry can only have one comment by a non-user of that name. For example, name “John” can only leave one comment per art entry.
- However, if a user ID is present and verified, the user can add as many comments as they want per art entry.

/api/users

Data to send when creating a new user

- name: STRING, required
- age: INTEGER, required
- location: STRING, required

Return JSON object example:

```
[{
        id: 101,
        name: “Ahren”,
        age: 24,
        location: “San Francisco”
},
{
        id: 102,
        name: “John”,
        age: 28,
        location: “San Francisco”
}]
```

Error handling

Each endpoint should have either a success or failed http code, and the necessary message as the response.