Brandon Lo

CS496

OSUID: 932055012

3/18/18

Final Project Cloud Only REST Routes

**App Link:**

<https://oauth-194607.appspot.com>

**Postman Video:**

<https://media.oregonstate.edu/media/t/0_3niab00u>

**Model Entities**

ID’s were automatically created with ndb with urlsafe to allow for embedding into an URL, so ID’s are

omitted from the model entities

**movie**

* movie\_id (string)
* title (string)
* description (string)
* release\_date (string)
* rt\_score (string)

**userAccount**

* movie\_id (string)
* fname (string)
* lname (string)
* email (string)

**3RD PARTY API:**

Link: https://ghibliapi.herokuapp.com

Description: Is a catalog of all of the movies that the studio Ghibli creates. The API returns a response in JSON which is parsed through and a random film is assigned for the user to watch and complete. The Ghibli API has the title, description, producer, release date, rotten tomato score, people, species, locations, and vehicles. In the application, the app only grabbed ad associated the movie, description, release date, and the rotten tomato with the user account. I chose this API since I wanted to watch some of the Ghibli studios movies and keep track of what I watched. Additionally based off the rotten tomatoes scores of all the movies that the user has been associated with, the average is associated with the user.

**User Handler**

GET /users/{user\_id}

**Action**: Returns the user account information and movies connected to the user\_id

**Body:** N/A

**Response:** Will return 200 response if successful. Returns 400 for an error, such as if the user\_id is wrong.

**Movie Handler**

GET /users/{user\_id}/movieInfo

**Action**: Returns all the movies info connected to the user\_id along with the average rotten tomato score of all the movies

**Body:** N/A

**Response:** Will return 200 response if successful. Returns 400 for an error, such as if the user\_id is wrong.

GET /users/{user\_id}/movieInfo/{movie\_id}

**Action**: Returns the movies info based on the movie\_id

**Body:** N/A

**Response:** Will return 200 response if successful. Returns 400 for an error, such as if the movie\_id is wrong.

POST /users/{user\_id}/movieInfo

**Action**: Creates the new movie and attaches it to the user\_id. The movie title is matched to the API’s titles and creates a new entity based off the title.

**Body:** {“title”: string}

**Response:** Will return 200 response if successful. Returns 400 for an error, such as if the body is wrong. If the body is wrong it displays the correct format to input.

DELETE /users/{user\_id}/movieInfo/{movie\_id}

**Action**: Deletes the movie associated with the user\_id based on the movie\_id

**Body:** N/A

**Response:** Will return 200 response if successful. Returns 400 for an error, such as if the movie\_id is wrong or if the user\_id is wrong.

PATCH /users/{user\_id}/movieInfo/{movie\_id}

**Action**: Updates the movie based on the body by the movie\_id

**Body:** {"title": string} or {"description": string} or {"release\_date": string} or {“rt\_score": string}

**Response:** Will return 200 response if successful. Returns 400 for an error, such as if the movie\_id is wrong or if the body is incorrectly entered.

PUT /users/{user\_id}/movieInfo/{movie\_id}

**Action**: Updates all the movie by the movie\_id

**Body:** {"title": string, “description": string, "release\_date": string, “rt\_score": string}

**Response:** Will return 200 response if successful. Returns 400 for an error, such as if the movie\_id is wrong or if the body is incorrectly entered.

**User Account Handler**

GET /users/{user\_id}/accountinfo

**Action**: Returns all the account info based on the user\_id

**Body:** N/A

**Response:** Will return 200 response if successful. Returns 400 for an error, such as if the user\_id is wrong.

DELETE /users/{user\_id}/accountinfo

**Action**: Deletes the user and the movies associated with the user.

**Body:** N/A

**Response:** Will return 200 response if successful. Returns 400 for an error, such as if the user\_id is wrong.

PATCH /users/{user\_id}/accountinfo

**Action**: Updates the user based on the body by the user\_id

**Body:** {" fname ": string} or {" lname ": string} or {" email \_date": string}

**Response:** Will return 200 response if successful. Returns 400 for an error, such as if the user\_id is wrong or if the body is incorrectly entered.

PUT /users/{user\_id}/cryptoassets/{asset\_id}

**Action**: Updates all the user’s info by the user\_id

**Body:** {" fname ": string, " lname ": string, " email \_date": string}

**Response:** Will return 200 response if successful. Returns 400 for an error, such as if the user\_id is wrong or if the body is incorrectly entered.