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.