Outline of RESTful API for COMP 2406

R. Routly & Z. Ali
March 14th 2021, Project Check In Element

Described below are the routes/urls that our API supports, for each route we will discuss the points below, routes accessing the same resource are grouped together for clarity

- URL naming scheme
- Type of request (GET, POST, PUT)
- Description of the request and what it will accomplish
- Data required from the request and the format that it should have (if applicable)
- MIME type supported for this route (application/json, text/html, or both)
- What failure response codes will be supported.

Resources supported are Movies, Users, People, and Reviews

Movies

/movies/{movieID}

- Type of Request: GET
- Retrieves movie of the given ID with a parameterized URL of an integer movie id
- Supports response types application/json and text/html
    - application/json requests are responded with an object representing data for the movie, valid request status code is 200
    - text/html requests are responded with an html page generated from a pug template with the data representing the movie, valid request status code is 200
- Failure response is status code 404, indicating the resource doesn't exist

/movies

- Type of Request: GET
- Route used to search with query parameters
    - Supported Query Parameters are title, actor name, genre keyword, num items per page, current page number
- Supports response types application/json and text/html
    - application/json requests are responded with an array of movie ids of the movies matching the query parameters, valid request status code is 200
    - text/html requests are responded with an html page generated from a pug template with the search results, valid request status code is 200

- Failure response is a status code 400, indicating an unsupported query parameter

/movies

- Type of Request: POST
- Route used to add a movie to the database
- Data is expected to come from the client in a JSON string, must contain at minimum a title, release year, runtime, plot, genre keyword, and at least one writer, director and actor
- Supports response types application/json
    - application/json requests are responded with the created object in json format
- Failure responses
    - Status Code 400 indicating there is an issue with the data provided by the user not allowing it to be updated
    - Status Code 401 meaning the current user is not a contributing user and thus cannot add movies to the database

/movies/{movieID}/reviews

- Type of Request: GET
- Retrieves the array of reviews about a particular movie
- Supports response types application/json
    - application/json requests are responded with an array of the ids of reviews about the movie, valid request status code is 200
- Failure responses
    - Status Code 404 indicating the film cannot be found

Users

/users/{username}

- Type of Request: GET
- Retrieves user of the given username with a parameterized URL of the string username
    - If this is the currently logged in user it will direct to the userprofile view, if not it will direct to the page for other users
- Supports response types application/json and text/html
    - application/json requests are responded with an object representing data for the user, valid request status code is 200
    - text/html requests are responded with an html page generated from a pug template with the data representing the user, valid request status code is 200
- Failure response is status code 404, indicating the resource doesn't exist or can't be found
-

/users/signup

- Type of Request: POST
- Adds a new user to the server, account type is set to basic user by default
- Expected data is a JSON string representing the information needed to add a user to the database
  {"username": username, "password": password, "accountType": False}
- Redirects the user to their profile after signing the user up successfully
- Supports response types application/json
    - application/json requests are responded with an object representing data for the newly created user, valid request status code is 200
- Failure responses
    - Status Code 400 indicates that the data could not be added to the server

/users/login

- Type of Request: POST
- Authenticates a user, being sent as a post request since it is insecure to pass login information through a GET request
- Redirects the user to their profile after logging in successfully.
- Expected data is a JSON string representing the authentication information need to authorize a user
  {"username": username, "password": password}
- Supports response types application/json

        - application/json requests are responded with an object representing data for the user, valid request status code is 200
    - Failure responses
        - Status Code 401 indicates the authentication is wrong

/users/{username}/logout

- Type of Request: GET
- Verifies that the currently logged in user is the one making the request, then removes the status and authentication of the user
- Redirects the user to the homepage.
- Supports response types application/json
    - application/json requests are responded with an object {“loggedOut”: True}, valid request status code is 200
- Failure responses
    - Status Code 401 indicates the authentication is wrong (not the same user logging out)

/users/{username}/accountType

- Type of Request: GET
- Retrieves the boolean value of accountType of the user if logged in, True indicates contributing account, False indicates basic account
- Supports response types application/json
    - application/json requests are responded with an object representing the username, password, and account type for the logged in user, valid request status code is 200
- Failure responses
    - Status Code 401 meaning this user is not logged in and cannot see their information since they are not authorized
    - Status Code 404 indicating the user cannot be found

/users/{username}/accountType

- Type of Request: PUT
- Changes account type of a user, only available for the currently logged in user
- Data expected in a JSON string of the pattern { “accountType”: boolean value of account type}
- Supports response types application/json
    - application/json requests are responded with an object representing updated account type for the logged in user, valid request status code is 200
- Failure responses

        - Status Code 400 indicates that the data could not be added to the server
        - Status Code 401 meaning this user is not logged in and cannot see their information since they are not authorized
        - Status Code 404 indicating the user cannot be found

/users/{username}/peoplefollowing

- Type of Request: GET
- Retrieves the list of people that a user is following if requested about their own account
- Supports response types application/json
    - application/json requests are responded with an array of the names of people the user is following, valid request status code is 200
- Failure responses
    - Status Code 401 meaning this user is not logged in and cannot cannot receive information about their account
    - Status Code 404 indicating the user cannot be found

/users/{username}/peoplefollowing

- Type of Request: PUT
- Changes the array of people the user is following to the new array passed in the request, will be validated on the server side
- Data expected in a JSON string of the pattern {“people following”: [array of people’s names (string)]}
- Supports response types application/json
    - application/json requests are responded with an object representing the updated array of people following for the logged in user, valid request status code is 200
- Failure responses
    - Status Code 400 indicates that the data could not be added to the server
    - Status Code 401 meaning this user is not logged in and cannot see their information since they are not authorized
    - Status Code 404 indicating the user cannot be found

/users/{username}/usersfollowing

- Type of Request: GET
- Retrieves the list of other users that a user is following if requested about their own account
- Supports response types application/json
    - application/json requests are responded with an array of the usernames of other users the user is following, valid request status code is 200

- Failure responses
    - Status Code 401 meaning this user is not logged in and cannot cannot receive information about their account
    - Status Code 404 indicating the user cannot be found

/users/{username}/usersfollowing

- Type of Request: PUT
- Changes the array of other users the user is following to the new array passed in the request, will be validated on the server side
- Data expected in a JSON string of the pattern {“users following”: [array of usernames (string)]}
- Supports response types application/json
    - application/json requests are responded with an object representing the updated array of other users that the logged in user is following, valid request status code is 200
- Failure responses
    - Status Code 400 indicates that the data could not be added to the server
    - Status Code 401 meaning this user is not logged in and cannot see their information since they are not authorized
    - Status Code 404 indicating the user cannot be found

/users/{username}/watchlist

- Type of Request: GET
- Retrieves the list of movies that a user is has on their watchlist if requested about their own account
- Supports response types application/json
    - application/json requests are responded with an array of the ids of movies the user is has on their watchlist, valid request status code is 200
- Failure responses
    - Status Code 401 meaning this user is not logged in and cannot cannot receive information about their account
    - Status Code 404 indicating the user cannot be found

/users/{username}/watchlist

- Type of Request: PUT
- Changes the array of films the user has on their watchlist to the new array passed in the request, will be validated on the server side
- Data expected in a JSON string of the pattern {“watchlist”: [array of movie ids (int)]}
- Supports response types application/json

        - application/json requests are responded with an object representing the updated array of films on the users watchlist, valid request status code is 200
    - Failure responses
        - Status Code 400 indicates that the data could not be added to the server
        - Status Code 401 meaning this user is not logged in and cannot see their information since they are not authorized
        - Status Code 404 indicating the user cannot be found

/users/{username}/reviews

- Type of Request: GET
- Retrieves the array of reviews that a user has written if requested about their own account
- Supports response types application/json
    - application/json requests are responded with an array of the ids of reviews the user is has written, valid request status code is 200
- Failure responses
    - Status Code 401 meaning this user is not logged in and cannot cannot receive information about their account
    - Status Code 404 indicating the user cannot be found

People

/people/{personName}

- Type of Request: GET
- Retrieves information stored on the user with the specified name with a parameterized URL of an string name
- Supports response types application/json and text/html
    - application/json requests are responded with an object representing data for the person, valid request status code is 200
    - text/html requests are responded with an html page generated from a pug template with the data representing the person, valid request status code is 200
- Failure response is status code 404, indicating the resource doesn't exist

/people

- Type of Request: POST
- Route used to add a person to the database
- Data is expected to come from the client in a JSON string, must contain at minimum a name
- Supports response types application/json
    - application/json requests are responded with the created object in json format
- Failure responses
    - Status Code 400 indicating there is an issue with the data provided by the user (duplicate name etc.)
    - Status Code 401 meaning the current user is not a contributing user and thus cannot add movies to the database

Reviews

We made the decision to include a reviews resource in order to reference it from the user and movie arrays connected to each review. This saves server space and avoids duplicate records.

/reviews/{reviewID}

- Type of Request: GET
- Retrieves review of the given ID with a parameterized URL of the review id stored as an integer
- Supports response types application/json and text/html
    - application/json requests are responded with an object representing data for the movie, valid request status code is 200
- Failure response is status code 404, indicating the resource doesn't exist

/reviews

- Type of Request: POST
- Adds a new review
- Data expected in a JSON string of the pattern, the brief and full review strings are optional but it must include the other keys, will be validated on the server side {"associated user": username, "associated movie": movie id, "rating": int, "reviewBrief": brief summary (string), "reviewFull": full review (string)}
- Supports response types application/json
    - application/json requests are responded with an object representing the id of the added review, valid request status code is 200
- Failure responses
    - Status Code 400 indicates that the data could not be added to the server