Building a Flask API with Authentication, Movie Details, and Movie List

Objective:

The objective of this assignment is to create a Flask API that includes authentication using an API key. The API should allow users to retrieve movie details by sending a movie name to the API endpoint and also provide a list of all available movies. To accomplish this, you will utilise a free open-source movie data source.

Codes: I've used OMBD API to fulfil the project

Config.py:

```
#this is the API key to authenticate the header given in the post man.

API_KEY = "a2828bd2"
```

Auth.py:

```
#to import we require these libraries.

# The wraps function from the functools module is imported,
# the metadata of the original function is preserved using the wraps function.
from functools import wraps

#request is used to access the data sent with the request.
from flask import request, jsonify

#imports the API_KEY variable from the Config module.
from Config import API_KEY

#This function acts as a decorator, preventing access to the wrapped function
func until a valid API key is detected.
def require_api_key(func):

    #This decorator line uses the wraps decorator to wrap the decorated
function.
    @wraps(func)

    #The decorated inner function is defined. This function will replace old
require_api_key function.
    def decorated(*args, **kwargs):
```

```
        #Using the 'request.headers.get' method, this line extracts the value
of the "API-Key".
        api_key = request.headers.get("API-Key")

        #This line determines whether the api_key variable is empty or
different from the API_KEY.
        if not api_key or api_key != API_KEY:

            #When a requirement for an unauthorised request is satisfied,
issuing a JSON response along with an error message.
            return jsonify({"error": "Unauthorized"})

        #This line calls the original function if the API key is valid.
        return func(*args, **kwargs)

    #returned as the result of the require_api_key function.
    return decorated
```

App.py:

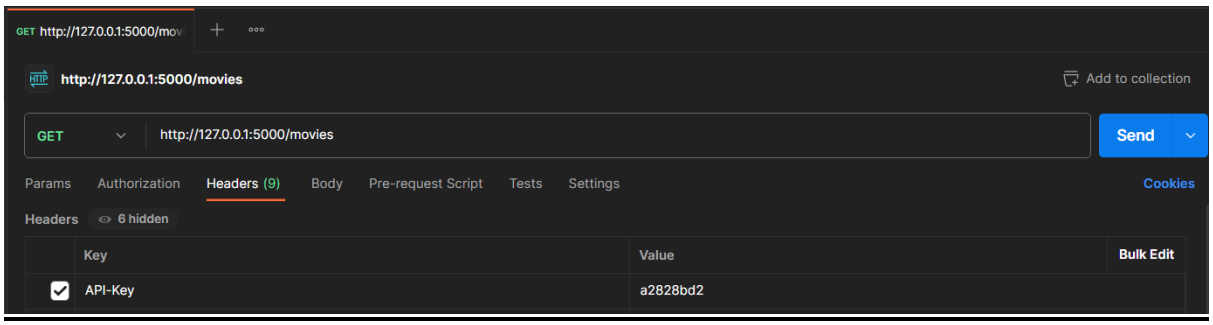
```
#flask used for flask application, jsonify is used for JSON format responses.
from flask import Flask, jsonify

#imports the require_api_key decorator from the Auth module.
from Auth import require_api_key

#used for making HTTP requests to external APIs.
import requests

#This will handle incoming requests and direct them to the appropriate
functions.
app = Flask(__name__)

#It indicates that HTTP GET requests will be handled by this route.
@app.route("/movies/<movie_name>", methods=["GET"])

#This decorator ensures that a valid API key must be provided in the request
headers in order to access the function.
@require_api_key

#the 'get_movie_details' function, takes 'movie_name' as a parameter.
def get_movie_details(movie_name):

    #URL to fetch movie details from the OMDb API.
```

```
    #It includes the movie name as a query parameter, API key required for
authentication.
    url = f"http://www.omdbapi.com/?apikey=a2828bd2&t={movie_name}"

    #The response is saved in the response variable after it makes an GET
request to the URL created in the previous step.
    response = requests.get(url)

    #This line takes the response object's JSON data and extracts it, storing
it in the data variable.
    data = response.json()

    #checks if the OMDb API response has the key "Response". If true, movie
details were found.
    if data.get("Response") == "True":

        #movie details
        movie_details = {
            "title": data["Title"],
            "release_year": data["Year"],
            "plot": data["Plot"],
            "cast": data["Actors"].split(", "),
            "rating": data["imdbRating"]
        }

        #This line sends a JSON response with the movie_details if the movie
details were found.
        return jsonify(movie_details)

    #This line delivers a JSON response with an error message if the movie
details are not found.
    return jsonify({"error": "Movie not found"})

#This line specifies an alternative path to the endpoint /movies.
@app.route("/movies", methods=["GET"])

##This decorator ensures that a valid API key must be provided in the request
headers in order to access the function.
@require_api_key

#Defines the 'get_movie_list' function, which does not take any parameters.
def get_movie_list():

    ##URL to fetch movie details from the OMDb API.
    #It includes the necessary API key for authentication and specifies
"movie" as the search type.
    url = f"http://www.omdbapi.com/?apikey=a2828bd2&s=popular&type=movie"
```

```
    #The response is saved in the response variable after it makes an GET
request to the URL created in the previous step.
    response = requests.get(url)

    ##create a list of movie dictionaries that include details like "title"
and "release_year."
    data = response.json()
    if data.get("Response") == "True":
        movie_list = []
        for movie in data["Search"]:
            movie_list.append({
                "title": movie["Title"],
                "release_year": movie["Year"]
            })

        #This line provides a JSON response with the movie_list if the movie
list is found.
        return jsonify(movie_list)

    # If the movie list is not found, this line returns a JSON response with
an error message.
    return jsonify({"error": "Movie list not found"})

#This line checks if the current module is the main script.
if __name__ == "__main__":

    #This line starts the Flask development server with debugging enabled if
the current module is the main script.
    app.run(debug=True)
```

POSTMAN:

The link returns the output in JSON format



GET http://127.0.0.1:5000/mov

http://127.0.0.1:5000/movies

Add to collection

GET http://127.0.0.1:5000/movies Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

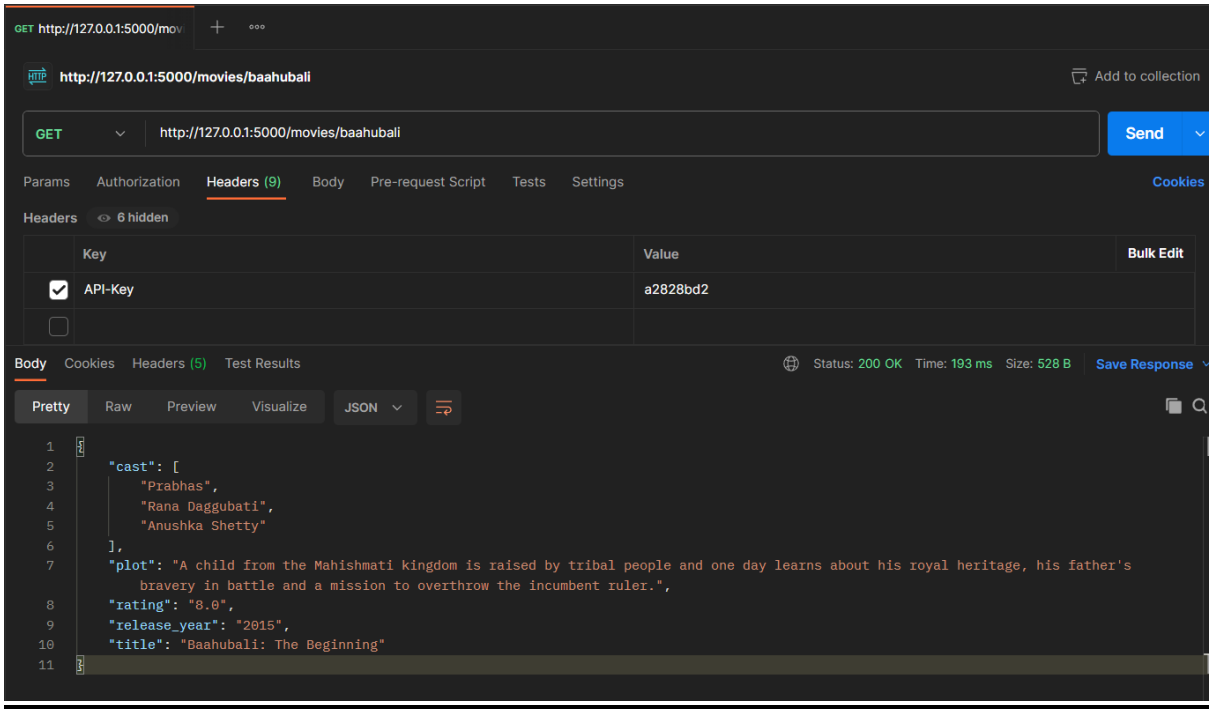
Body Cookies Headers (5) Test Results Status: 200 OK Time: 439 ms Size: 1.13 KB Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "release_year": "2011",
        "title": "Stone Cold Steve Austin: The Bottom Line on the Most Popular Superstar of All Time"
    },
    {
        "release_year": "1962",
        "title": "A Symposium on Popular Songs"
    },
    {
        "release_year": "1955",
        "title": "How to Be Very, Very Popular"
    },
    {
        "release_year": "1947",
        "title": "Are You Popular?"
    },
    {
        "release_year": "1978",
        "title": "Notes on the Popular Arts"
    },
    {
        "release_year": "1933",
        "title": "Popular Melodies"
    },
    {
        "release_year": "2018",
        "title": "Dollhouse: The Eradication of Female Subjectivity from American Popular Culture"
    },
    {
        "release_year": "2007",
        "title": "Paul Simon: The Library of Congress Gershwin Prize for Popular Song"
    },
    {
        "release_year": "2011",
        "title": "Popular"
    },
    {
        "release_year": "2010",
        "title": "The Library of Congress Gershwin Prize for Popular Song: In Performance at the White House - Paul McCartney"
    }
]
```

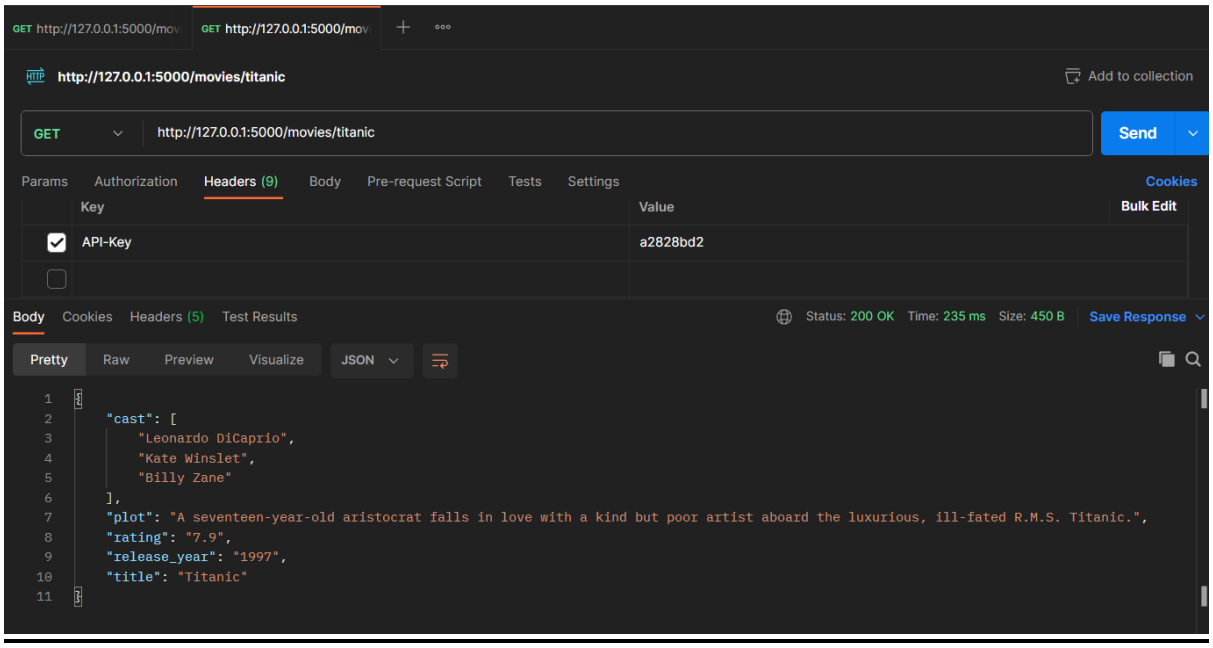

GET: http://127.0.0.1:5000/movies/baahubali

The link returns the cast, plot, rating, release year, title of the movie “Baahubali”.



GET: http://127.0.0.1:5000/movies/titanic

The link returns the cast, plot, rating, release year, title of the movie “Titanic”.



Sandhya Raghavi

AP20110010684

SRM University

2020 – 2024

**********THE END**********