Name :- Nallapuneni Vamsi Krishna(AP20110010735)
Github link:-
Demo video link:-https://drive.google.com/file/d/12qa0wdJDpzJBZw8uLPLsS4G_BUpdHUQL/view?usp=sharing

Assignment: 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.

CODE :-

```
import requests
from flask import Flask, request, jsonify
from functools import wraps

app = Flask(__name__)

# API key configuration
API_KEY = 'dbcbb9be92919b0e4e12a446a11216c1'

def require_api_key(view_func):
   @wraps(view_func)
   def decorated_func(*args, **kwargs):
       provided_key = request.headers.get('X-API-Key')

       # Check if the provided API key matches the configured API key
       if not provided_key or provided_key != API_KEY:
           # Return an "Unauthorized" error if the API key is missing or
incorrect
           return jsonify({'error': 'Unauthorized'}), 401

       # If the API key is valid, proceed to the decorated view function
       return view_func(*args, **kwargs)

   return decorated_func

@app.route('/movies/<movie_name>', methods=['GET'])
@require_api_key
def get_movie_details(movie_name):
   # Make a request to the TMDb API to search for the movie
```

```
    api_url =
f'https://api.themoviedb.org/3/search/movie?api_key={API_KEY}&query={movie_name
}'
    response = requests.get(api_url)

    # Check if the request was successful
    if response.status_code == 200:
        # Extract the movie details from the response
        data = response.json()

        # Check if any movie matches the search query
        if data['total_results'] > 0:
            # Get the first movie from the search results
            movie = data['results'][0]

            # Extract the relevant details from the movie
            movie_id = movie['id']
            cast_api_url =
f'https://api.themoviedb.org/3/movie/{movie_id}/credits?api_key={API_KEY}'
            cast_response = requests.get(cast_api_url)
            cast_data = cast_response.json()
            cast = [actor['name'] for actor in cast_data['cast']]

            # Prepare the movie details to be returned as JSON
            movie_details = {
                'title': movie['title'],
                'release_year': movie['release_date'][:4],
                'plot': movie['overview'],
                'cast': cast,
                'rating': movie['vote_average']
            }

            return jsonify(movie_details)

    # If no movie is found or there was an error, return an error message
    return jsonify({'error': 'Movie not found'}), 404

@app.route('/movies', methods=['GET'])
@require_api_key
def get_movie_list():
    # Get query parameters from the request
    year = request.args.get('year')
    genre = request.args.get('genre')
    rating = request.args.get('rating')

    # Build the query parameters for the API request
    query_params = {
        'api_key': API_KEY,
        'year': year,
```

```
        'with_genres': genre,
        'vote_average.gte': rating
    }

    # Make a request to the TMDb API to retrieve the movie list
    api_url = f'https://api.themoviedb.org/3/discover/movie'
    response = requests.get(api_url, params=query_params)

    # Check if the request was successful
    if response.status_code == 200:
        # Extract the movie list from the response
        data = response.json()

        # Extract the relevant details from each movie
        movie_list = []
        for movie in data['results']:
            # Fetch cast details for each movie
            cast_api_url =
f'https://api.themoviedb.org/3/movie/{movie["id"]}/credits?api_key={API_KEY}'
            cast_response = requests.get(cast_api_url)
            cast_data = cast_response.json()
            cast = [actor['name'] for actor in cast_data['cast']]

            # Check if the release_date key exists in the movie data
            release_year = movie.get('release_date', '')[:4]

            # Prepare the movie details to be included in the movie list
            movie_details = {
                'title': movie['title'],
                'release_year': release_year,
                'plot': movie['overview'],
                'cast': cast,
                'rating': movie['vote_average']
            }
            movie_list.append(movie_details)

        return jsonify(movie_list)

    # If there was an error, return an error message
    return jsonify({'error': 'Failed to retrieve movie list'}), 500

if __name__ == '__main__':
    # Run the Flask app in debug mode
    app.run(debug=True)
```

Explanation about the code : -

This code is a Python Flask application that serves as an API for retrieving movie details from The Movie Database (TMDb) API. Let's break it down step by step:

1. The necessary imports are made, including the `requests` library for making HTTP requests, `Flask` for creating the web application, `request` for handling HTTP requests, `jsonify` for creating JSON responses, and `wraps` from `functools` for creating decorators.

2. An instance of the Flask application is created.

3. An API key is defined. This key is used to authenticate and authorize access to the TMDb API.

4. The `require_api_key` function is defined as a decorator. This decorator will be applied to routes that require an API key for access. It checks if the API key provided in the request headers matches the configured API key. If the API key is missing or incorrect, it returns an "Unauthorized" error response. If the API key is valid, it allows access to the decorated view function.

5. The `get_movie_details` route is defined with the route pattern `/movies/<movie_name>`. It is decorated with `@require_api_key` to require an API key for access. This route handles GET requests and retrieves details about a specific movie. It makes a request to the TMDb API to search for the movie using the provided `movie_name`. If a matching movie is found, it extracts relevant details such as title, release year, plot, cast, and rating. These details are returned as a JSON response. If no movie is found or there is an error, an error message is returned.

6. The `get_movie_list` route is defined with the route pattern `/movies`. It is also decorated with `@require_api_key` to require an API key for access. This route handles GET requests and retrieves a list of movies based on query parameters such as year, genre, and rating. It constructs the query parameters for the TMDb API request and makes the request to retrieve the movie list. It then extracts the relevant details from each movie, including the cast, and returns the list of movie details as a JSON response. If there is an error, an error message is returned.

7. The main entry point of the application is checked using `if __name__ == '__main__'`. If the script is executed directly, the Flask application is run in debug mode using `app.run(debug=True)`.

In summary, this code sets up a Flask API that requires an API key for accessing movie details. It provides two routes, one for retrieving details about a specific movie and another for retrieving a list of movies based on certain criteria.
8. We get the link on the output we post it on postman and send get request we get these outputs

We run this code on any editor i used pycharm

```python
import requests
from flask import Flask, request, jsonify
from functools import wraps

app = Flask(__name__)

# API key configuration
API_KEY = 'dbcbb9be92919b0e4e12a446a11216c1'

def require_api_key(view_func):
    @wraps(view_func)
    def decorated_func(*args, **kwargs):
        provided_key = request.headers.get('X-API-Key')

        # Check if the provided API key matches the configured API key
        if not provided_key or provided_key != API_KEY:
            # Return an "Unauthorized" error if the API key is missing or incorrect
            return jsonify({'error': 'Unauthorized'}), 401
```

```
C:\Users\INS\anaconda3\python.exe C:\Users\INS\Desktop\Flaskapp\main.py
 * Serving Flask app "main" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 279-991-330
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [16/Jul/2023 21:14:58] "GET /movies HTTP/1.1" 200 -
```

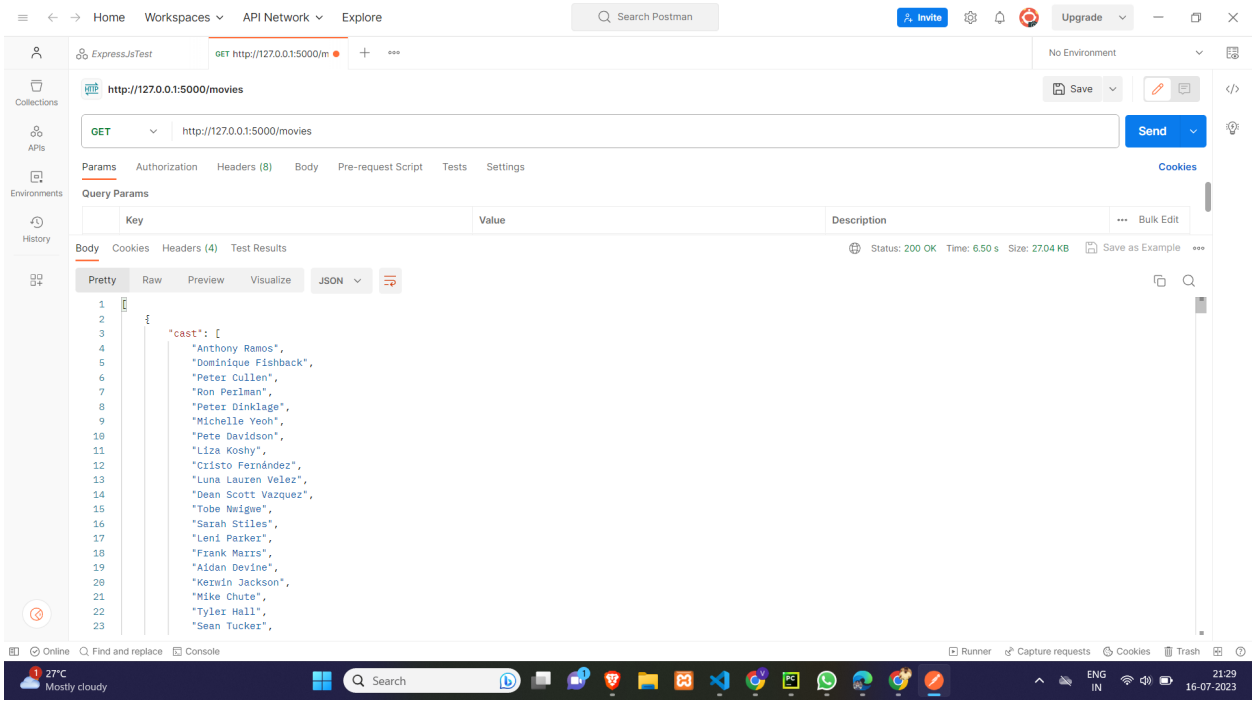
We are having screen shots of two requests on postman they are

http://127.0.0.1:5000/movies **And http://127.0.0.1:5000/movies/knights of zodiac**

GET http://127.0.0.1:5000/movies/Knights of the Zodiac

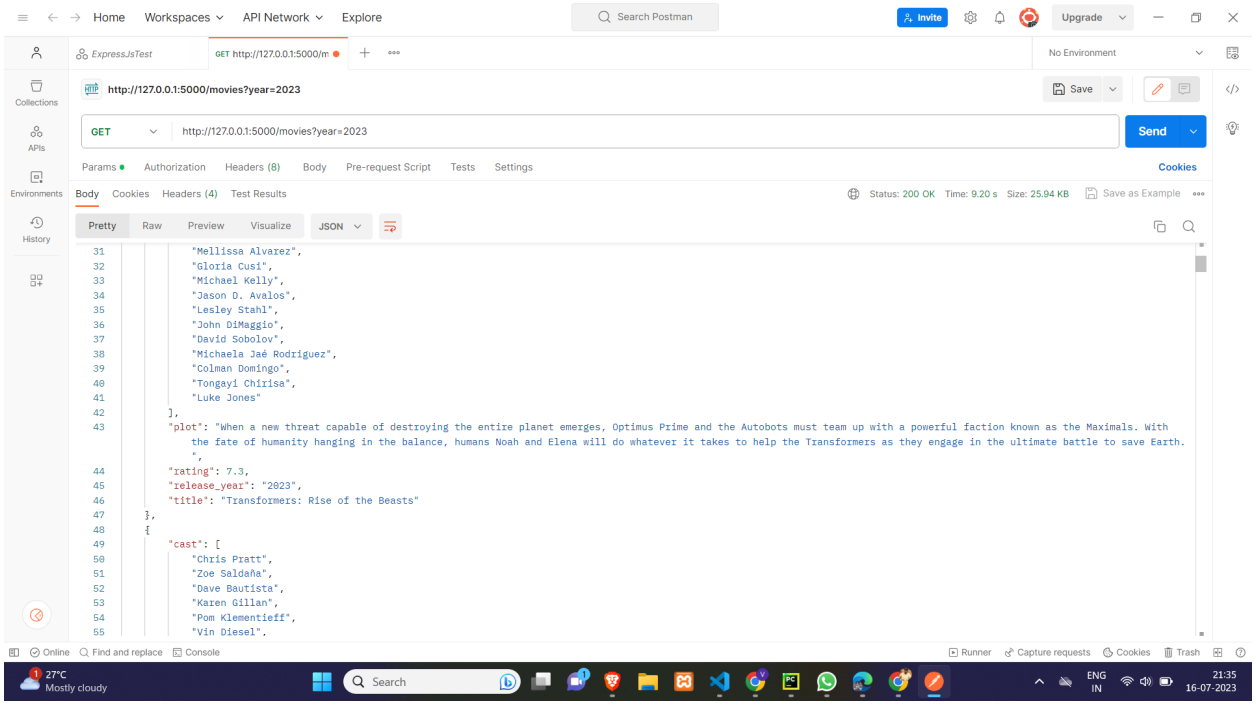
Status: 200 OK Time: 1038 ms Size: 887 B

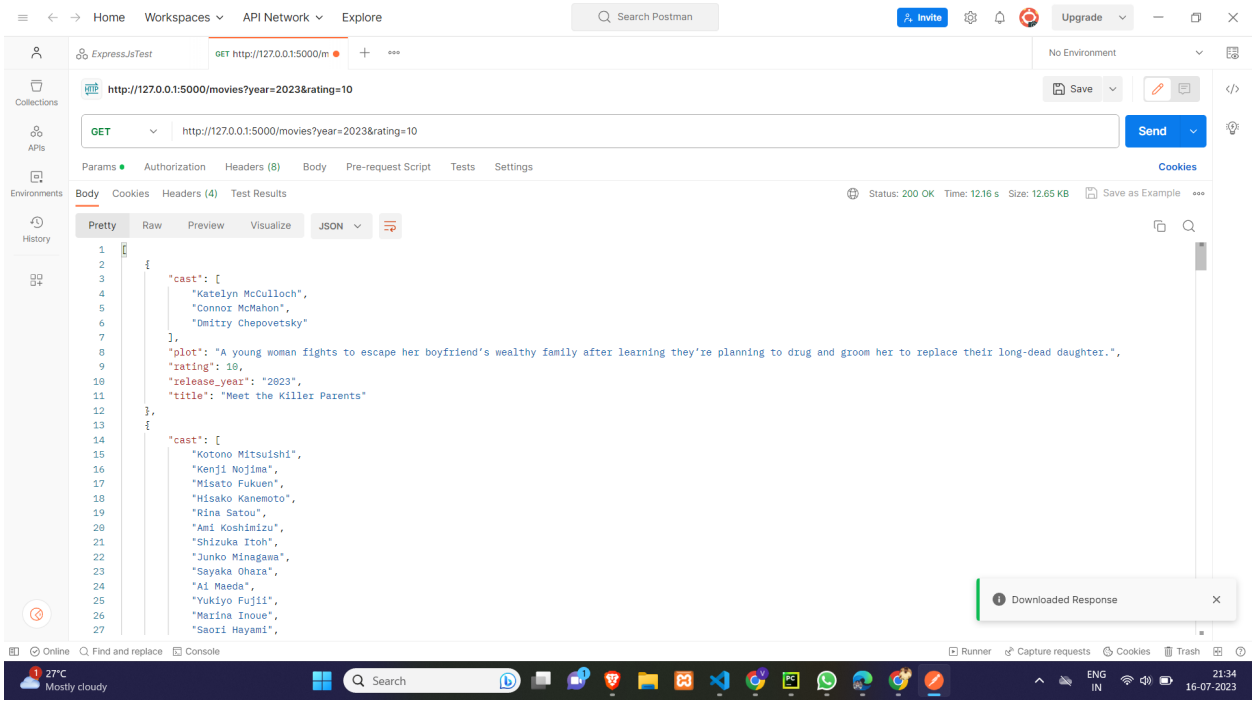

```
{
    "cast": [
        "Mackenyu",
        "Madison Iseman",
        "Diego Tinoco",
        "Mark Dacascos",
        "Nick Stahl",
        "Famke Janssen",
        "Sean Bean",
        "Caitlin M Hutson",
        "Katie Moy",
        "Kaylan Teague",
        "Ryusei Iwata",
        "T.J. Storm",
        "David Torok",
        "Todd Williams",
        "Zoltán Duzkó"
    ],
    "plot": "When a headstrong street orphan, Seiya, in search of his abducted sister unwittingly taps into hidden powers, he discovers he might be the only person alive who can protect a reincarnated goddess, sent to watch over humanity. Can he let his past go and embrace his destiny to become a Knight of the Zodiac?",
    "rating": 6.6,
    "release_year": "2023",
    "title": "Knights of the Zodiac"
}
```



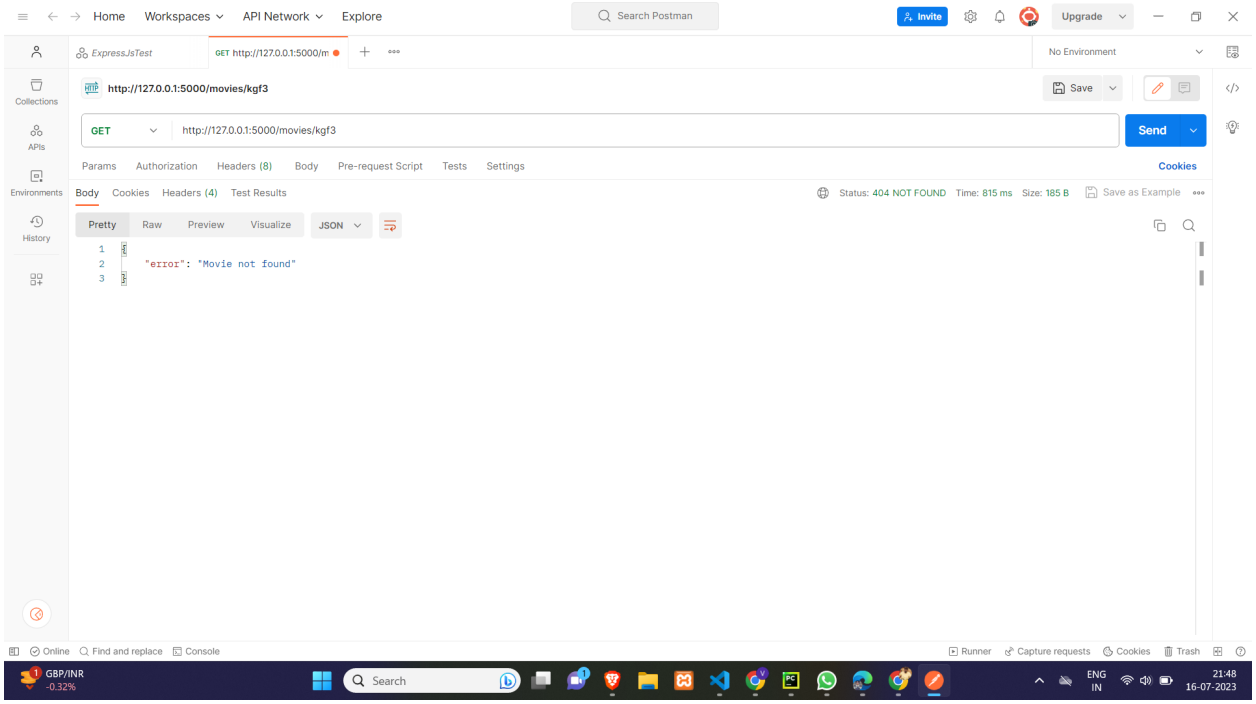
We now having screen shots of two requests on post man they are

http://127.0.0.1:5000/movies?year=2023
And
http://127.0.0.1:5000/movies?year=2023&rating=10





if we send some unknown request of not having movies we get output like this



we get the api key by sign up and login and send request to the tmdb

TMDB

← Back

Your Account

New Discussion

Actions

Stop Watching

Make Public

Re-Open Discussion

View Activity

Email Notifications

Unsubscribe

Users In This Discussion

V

Address Postal Code: 521190

Like Quote

1 reply (on page 1 of 1) • Jump to last post

Reply by **Travis Bell** STAFF
on July 16, 2023 at 8:06 AM

Hi vamsi_nallapuneni,

Your request for an API key has been approved. You can start using this key *immediately*.

API Key: dbcbb9be92919b0e4e12a446a11216c1

Here's an example API request:

```
https://api.themoviedb.org/3/movie/550?api_key=dbcbb9be92919b0e4e12a446a11216c1
```

Useful Links

1. Documentation
2. Support forum
3. Wrappers & libraries
4. Service status

If you have any questions, feel free to create a new discussion in our support forum.

--
The Movie Database Team
https://www.themoviedb.org

Like Quote Report Ignore