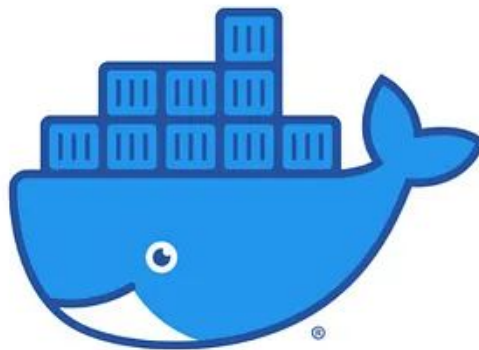


TMDB Poster Download Project

TEAM 7



Dockerized Flask + MongoDB Application



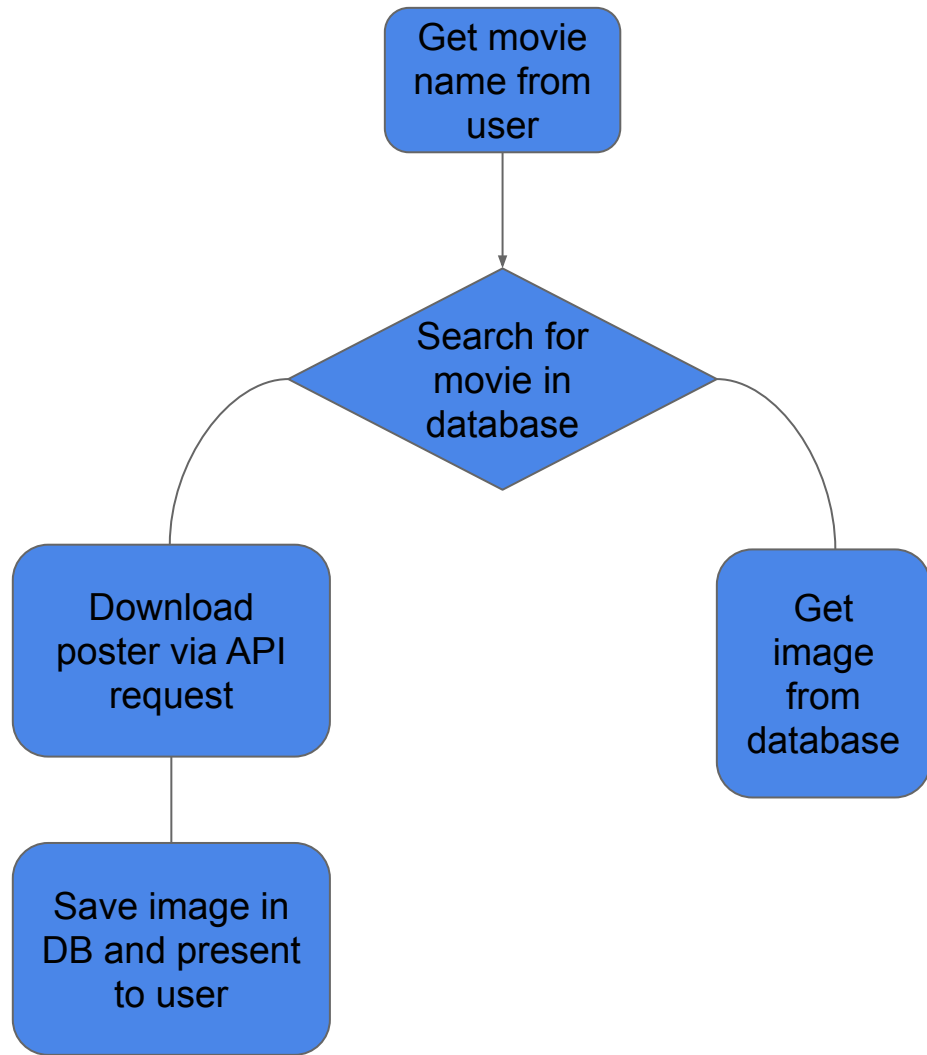
Team Members

?

Project Objectives

- **Get Movie Name**
- **Search Movie poster in TMDB Database**
- **Download poster and poster meta data and save it in MongoDB**
- **Construct a full Database API**
- **Write a simple web app to present the posters**
- **Dockerize the application**
- **Deploy application on AWS EC2 instance**

Algorithm Design



Teamwork and responsibilities

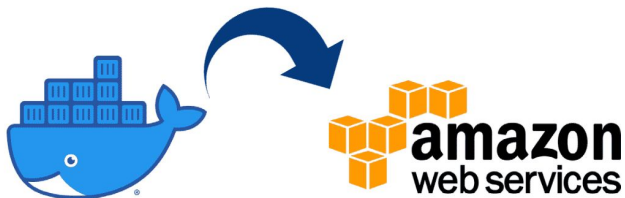


Software Design



- OOP approach - Allows scalable code and services autonomy.
- **TMDB_Downloader.py** module - holds **TMDBDownloader** class and handles all the logic of downloading posters
- **MongoDBAPI.py** module - holds **MongoAPI** class and handles all the API management logic including reading records, writing records, updating records and deleting records
- **mongo_tmdb_logic.py** module - holds the "Business logic", integration between the two classes - if movie poster is not found in DB then download it from TMDB website and insert to DB.
- **api.py** module - hold web application logic using Flask, currently handles only presenting posters feature.

Dockerize the App and deploy on AWS



- Write Dockerfile
- Write docker-compose file
- Write user_data script to deploy the App on an EC2 instance

```
#!/bin/bash
sudo yum update -y
sudo amazon-linux-extras install docker -y
sudo systemctl start docker
sudo usermod -a -G docker ec2-user
sudo systemctl enable docker
sudo curl -SL https://github.com/docker/compose/releases/download/v2.4.1/docker-compose-linux-x86_64 -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
sudo yum install git -y
sudo git clone https://github.com/AmirKatorza/AWS_reStart_Docker-Flask-Mongo_App.git
# wget https://github.com/AmirKatorza/AWS_reStart_Docker-Flask-Mongo_App/archive/refs/heads/master.zip
# sudo yum install unzip -y
# sudo unzip master.zip
sudo cd ./AWS_reStart_Docker-Flask-Mongo_App/
sudo docker-compose build
sudo docker-compose up
```

Reference:

<https://gist.github.com/npearce/6f3c7826c7499587f00957fee62f8ee9>

GET requests screenshots – Postman



POSTMAN

The screenshot displays the Postman application window. The top navigation bar includes 'Home', 'Workspaces', 'API Network', and 'Explore'. The main workspace is titled 'My Workspace' and shows a 'New' button and an 'Import' button. A sidebar on the left contains navigation options: 'Collections', 'APIs', 'Environments', 'Mock Servers', 'Monitors', 'Flows', and 'History'. The central area shows a 'Your collection' dialog with a 'Type' dropdown set to 'API key'. Below this, a message reads: 'Create a collection for your requests. A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.' A 'Create Collection' button is present, along with an 'or Use a Template' link.

The right pane shows a GET request to 'http://127.0.0.1:5001/search'. The 'Body' tab is selected, showing a table with the following data:

Key	Value	Description
<input checked="" type="checkbox"/> movie_name	superman	
Key	Value	Description

Below the table, the 'Body' tab shows the JSON response in 'Pretty' format:

```
1 {
2   "_id": {
3     "$oid": "643a7f25e877e3085794baba"
4   },
5   "Status": "Found in DB",
6   "file_name": "superman.jpeg"
7 }
```

The status bar at the bottom indicates '200 OK', '19 ms', and '265 B'. It also includes buttons for 'Save as Example', 'Cookies', 'Capture requests', 'Runner', 'Trash', and 'Console'.

GET requests screenshots – Postman



POSTMAN

The screenshot displays the Postman application interface. The top navigation bar includes links for Home, Workspaces, API Network, and Explore, along with a search bar and utility buttons like 'Invite', 'Upgrade', and window controls. The left sidebar shows the 'My Workspace' section with a 'Collections' tab active, displaying a tree view of 'My first collection' with two folders. Below this is a prompt to 'Create a collection for your requests' with a 'Create Collection' button and a 'Use a Template' link. At the bottom left, a 'Start working with APIs' progress bar shows 67% completion with a 'Show me' link.

The main workspace shows a GET request to `http://127.0.0.1:5001/search`. The 'Body' tab is selected, showing a table with one parameter:

Key	Value	Description
<input checked="" type="checkbox"/> movie_name	black panther	

Below the table, the 'Body' tab shows the response in 'Pretty' format:

```
1 {
2   "_id": {
3     "$oid": "643a903e02bdade075f4d93e"
4   },
5   "Status": "Added to DB",
6   "file_name": "black panther.jpeg"
7 }
```

The status bar at the bottom indicates a 200 OK response, 2.15s duration, and 270 B body size. It also includes links for 'Cookies', 'Capture requests', 'Runner', 'Trash', and a help icon.

What can be improved?

- Improved UI (User Interface) Design - currently a simple HTML form with only search feature.
- Full API coverage by the web application including updating a records and deleting records.
- Scale UP - currently downloading only the first movie from search results and only the first poster image.

We want to save all movies from search results and all the posters related.

Challenges

- Learning new technologies in a very short amount of time.
- Learn how to integrate and orchestrate all the technologies to work together.
- Not enough ManPower to handle a project in that scale.

Thank You
For Listening!

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.