# Actor Hunt

PROJECT REPORT


MOHAMMED SHEHZAD - 4SO19CS090

NEHA L K - 4SO19CS096

PRAGNYA NAGURE - 4SO19CS110

PRAVIN KUMAR - 4SO19CS119

# Acknowledgement

We are grateful to our respectable teacher, Dr. Melwyn D'Souza , whose insightful leadership and knowledge benefited us to complete this project successfully. Thank you so much for your continuous support and presence whenever needed.

# Contents

# Introduction

## Problem Statement

Film Making industry has always been a part of our culture. Based on the massive movie information, it would be interesting to understand the actor/actress and the number of movies they've starred in along with their biography.

The Internet Movie Database (IMDb) is one of the world's most popular sources for movie, TV and celebrity content with more than 100 million unique visitors per month.

In this project, we take IMDb actor details as a response variable and display it to the users.

# Project Description

The application opens a webpage where the users can search for the actor/actress they need information on. The search then returns the users with actor details and two download options which include a word document with their biography and an excel sheet with the movies or shows they've starred in.

Our application ActorHunt provides a database that includes various details of the actors/actress'. The idea of our project is to get data by web scraping the details of the actor/actress from the IMDB website and use a database as a cache for the previously searched actors.

The functionalities of the project are as follows.

## DATABASE

ActorHunt contains a database with two tables. One stores the details of the actor like name, date of birth, profession, photo and their description and the other table stores the movie or show the actor has starred in and their corresponding year of release.

The web scraping process is done only when the actor is not found in the database and a copy of the scraped information is stored in the database as a search cache, so the next time the user searches for a previously searched actor, the retrieval time would be reduced.

## WEB SCRAPING

To get the details of the actor/actress, we make a request to the IMDB website and scrape the necessary information using the BeautifulSoup Library.

## WEB APIs (RESTful Web Services)

The exchange of data between client and server is possible through REST API i.e a POST request to the URL `/search` . We use a POST request to send the searched actor name from the client and receive it in the server. Necessary actions are performed on this request data and finally the response is sent back to the client.

## JSON

The exchange of data on the web takes place using JSON, so in our web application the data is parsed into JSON format and communicated between client and server.

The actor name is sent as a POST request in JSON format.

The scraped data of the actor is JSONified and sent as a response to the POST request.

## EXCEL SPREADSHEETS

When a user searches for a particular actor/actress , the application performs the necessary actions needed to retrieve the movies and their corresponding year of release, of the searched actor, which is then written into the Excel Spreadsheet that can be downloaded and viewed by the user.

## WORD DOCUMENT

The detailed biographical information and the picture of the searched actor is written into the word document and made available for the user to download and view.

# Source Code

## actorHunt.py

```
from flask import (
    Flask,
    json,
    render_template,
    request,
    make_response,
)
import writingInfo
import json

# Defining a FLASK web app
app = Flask(__name__)

# Route to the main page and rendering main page html
@app.route("/", methods=["POST", "GET"])
def mainPage():
    return render_template("index.html", content="")



@app.route("/search", methods=["POST", "GET"])
def getJson():
    if request.method == "POST":
```

```
        req = request.get_json() # getting the request data from client


        actorName, actorBD, actorJobs, actorPhotoPath, _ =
writingInfo.DoIt(req["name"])


        res = make_response(
            json.dumps( # Converting the data received into an JSON Format
                {
                    "name": actorName,
                    "job": actorJobs[1:-1],
                    "dob": actorBD[5:],
                    "pic": actorPhotoPath,
                }
            ),
            200,
        )
        return res # returning the JSON data to client with STATUS 200 OK
    return "get request received, something wrong"









if __name__ == "__main__":
    app.run(debug=True) # debug is set True to refresh the server whenever
the changes are made
```

## writingInfo.py

```
import requests, bs4
import openpyxl
import docx, os
import sqlite3



def DoIt(actor):

    try:
        # Establish a connection with the sqlite database file
        conn = sqlite3.connect("database.sqlite")
        cur = conn.cursor()

        # Creating the table if it doesn't exist in the database
        cur.execute("CREATE TABLE IF NOT EXISTS ACTORHUNT (NAME TEXT
PRIMARY KEY, DOB TEXT, JOB TEXT, PICTURE TEXT, INFO TEXT)")
        cur.execute("CREATE TABLE IF NOT EXISTS MOVIES (ACTORNAME TEXT,
TITLE TEXT, YEAR TEXT, FOREIGN KEY (ACTORNAME) REFERENCES ACTORHUNT (NAME)
)")

        # SQLite query to search for an actor in the table ACTORHUNT of
database
        cur.execute("SELECT * FROM ACTORHUNT WHERE NAME LIKE
?",('%'+actor.lower()+'%',))
        data0 = cur.fetchone() # retrieving all details from the returned
search cursor
```

```
        # SQLite query to search for all the movies of the actor in the
table MOVIES of database
        cur.execute("SELECT * FROM MOVIES WHERE ACTORNAME LIKE
?",('%'+actor.lower()+'%',))
        data1 = cur.fetchall()
    except sqlite3.Error as e:
        print("database error:\n" ,e.args)


    try:
        # If the data is found in the database then close connection with
database
        # write the data into respective Doc and Excel file
        # return the required data to the client
        if data0 is not None:
            cur.close()
            conn.close()
            writeDoc(data0)
            writeXL(data1)
            return data0
        else:
            # If the data not found in the database then close connection
with database
            # call `fetchAndSave(actor)` function to scrape web for the
details of the actor
            cur.close()
            conn.close()
            return fetchAndSave(actor)
    except UnboundLocalError as e:
        print("Error in database so \n" ,e.args)
```

```
def fetchAndSave(actor):
    try:
        # making request to the imdb website with the search query of the
actor
        res = requests.get("https://www.imdb.com/find?q=" +
actor.replace("", "+"))
        res.raise_for_status()
    except requests.exceptions.RequestException as e:
        print(e)

    # parsing html with BeautifulSoup
    soup = bs4.BeautifulSoup(res.text, "html.parser")
    soupelem = soup.select(".result_text a")

    try:
        actorpage = requests.get("https://www.imdb.com" +
soupelem[0].get("href"))
        actorpage.raise_for_status()
    except IndexError as e:
        print(e)
        return ['NOT FOUND', 'nil', 'nil', 'nil', 'nil']

    # actor page
    try:
        soup = bs4.BeautifulSoup(actorpage.text, "html.parser")
```

```
        # actor info
        actorName = soup.select(".header .itemprop")[0].text
        actorJobs = list(
            map(lambda name: name.text[1:],
soup.select("#name-job-categories a span"))
        )
        actorPhoto = soup.select("#name-poster")[0].get("src")

        actorBDsoup = soup.select("#name-born-info")[0]
        actorBD = " ".join(
            list(map(lambda text: text.strip(),
actorBDsoup.text.split("\n")))
        )

        # all the movies of the actor
        soupelem = soup.select("#filmo-head-actor +
.filmo-category-section .filmo-row")
        if len(soupelem) == 0:
            soupelem = soup.select("#filmo-head-actress +
.filmo-category-section .filmo-row")
        fullBioLink = soup.find("span", {"class":"see-more inline
nobr-only"}).a.get("href")
    except IndexError as e:
        print("actor not found, try again")
        return ['NOT FOUND', 'nil', 'nil', 'nil', 'nil']

    try:
        bio = requests.get("https://www.imdb.com"+fullBioLink)
        bio.raise_for_status()
```

```
    except requests.exceptions.RequestException as e:
        print(e)

    bioSoup = bs4.BeautifulSoup(bio.text, "html.parser")
    actorData = bioSoup.find("div", {"class":"soda odd"}).p

# ----------------------SAVING TO THE DATABASE----------------------
    try:
        conn = sqlite3.connect("database.sqlite")
        cur = conn.cursor()

        print("writing database")

        cur.execute("INSERT INTO ACTORHUNT (NAME, DOB, JOB, PICTURE, INFO)
VALUES (?, ?, ?, ?, ?)",(str(actorName).lower(), str(actorBD),
str(actorJobs), str(actorPhoto), str(actorData.text),))
        conn.commit()

        for row in range(2, len(soupelem)+2):
            movie = soupelem[row-2]
            title = movie.select("b a")[0].text
            year = movie.select(".year_column")[0].text[:6]
            cur.execute("INSERT INTO MOVIES (ACTORNAME, TITLE, YEAR)
VALUES (?, ?, ?)",(str(actorName).lower(), str(title), str(year)))
            conn.commit()

        cur.execute("SELECT * FROM ACTORHUNT WHERE NAME =
?",(actorName.lower(),))
```

```
        data0 = cur.fetchone()


        cur.execute("SELECT * FROM MOVIES WHERE ACTORNAME =
?",(actorName.lower(),))
        data1 = cur.fetchall()


        cur.close()
        conn.close()
    except sqlite3.Error as e:
        print("database key constraint error:\n" , e.args)
        # if key constraint occurs during insertion then the client is
prompted to check the spelling of the search query
        return ['check spelling', 'nil', 'nil', 'nil', 'nil']
#
---------------------------------------------------------------------------


    try:
        writeDoc(data0)
    except Exception as e:
        print(e)


    try:
        writeXL(data1)
    except Exception as e:
        print(e)


    try:
        return data0
```

```
    except UnboundLocalError as e:
        print("database error so \n" ,e.args)
        return ["NOT FOUND", 'NIL', 'NIL', 'NIL', 'NIL']



# function to write the actor movies into the excel sheet with extension
.xlsx
def writeXL(data1):
    wb = openpyxl.Workbook()
    sheet = wb.get_sheet_by_name("Sheet")
    sheet["A1"].value = "Title"
    sheet["B1"].value = "Year"

    for row in range(len(data1)):
        sheet["A" + str(row+2)] = data1[row][1]
        sheet["B" + str(row+2)] = data1[row][2]

    wb.save("static/actorMovies.xlsx")
    print("xlsx success")

# function to write the actor data into the Word document with extension
.docx
def writeDoc(data0):
    os.makedirs("photos", exist_ok=True)
    try:
        res = requests.get(data0[3])
    except requests.exceptions.ConnectionError as e:
        print(e)
```

```
    imageFile = open(os.path.join("photos", os.path.basename(data0[3])),
"wb")
    actorPhotoPath = data0[3].split("/")[-1]



    for chunk in res.iter_content(100000):
        imageFile.write(chunk)
    imageFile.close()

    doc = docx.Document()
    doc.add_paragraph(data0[0].capitalize(), "Title")
    doc.add_picture("./photos/" + actorPhotoPath, width=docx.shared.Cm(5))
    doc.add_paragraph("Actor Jobs: " + data0[2])
    doc.add_paragraph(data0[1])
    doc.add_paragraph(data0[4])

    doc.save("static/actordesc.docx")
    print("docx success!")
```

## Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <link rel="stylesheet"
href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
integrity="sha384-AYmEC3Yw5cVb3ZcuHtOA93w35dYTsvhLPVnYs9eStHfGJvOvKxVfELGr
oGkvsg+p" crossorigin="anonymous"/>
    <link rel="stylesheet" href="../static/style.css">
    <title>Actor Search</title>
</head>
{% block content %}
<body>
    <section class="main">
        <!-- <h1 class="icon"><i class="fas fa-film"></i></h1> -->
        <h1 class="heading">Actor <br><i class="fas fa-film"></i>unt</h1>

            <div class="searchbar">
                <input class="search" type="text" name="actor" id="actor"
required autocomplete="off"/>
                <span id="underline"></span>
            </div>
            <div class='btn-div'>
                <button class='btn' onclick="search();">Submit</button>
            </div>

        <span></span>

    </section>
    <section id="actor-info">
        <img id='spinner' src="../static/Spinner.gif" alt="">
        <div class="info">
```

```
            <h1 class="actorName"></h1>
            <img class="actor-img" src="" alt="">
            <div>
                <b>Occupation: </b><span class="actorOccupation"></span>
            </div>
            <div>
                <b>Born: </b><span class="actorBday"></span>
            </div>
            <div id="btn-download">
                <a class="btn" href="/static/actordesc.docx" download
>Download bio</a>
                <a class="btn" href="/static/actorMovies.xlsx" download
>Download movies</a>
            </div>
        </div>
    </section>
    <!-- <script src="./script.js"></script> -->
</body>
{% endblock %}
</html>


{% block script %}
<script>
    spinner = document.querySelector('#spinner');
    actorNameInput = document.querySelector('.search');
    actorNameheading = document.querySelector('.actorName');
    actorImg = document.querySelector('.actor-img')
    actorOccupation = document.querySelector('.actorOccupation');
```

```
actorBday = document.querySelector('.actorBday');
actorInfo = document.querySelector('#actor-info');
info = document.querySelector('.info');

actorNameInput.addEventListener('focus',(event)=>{
    document.querySelector('.fa-film').style.color='blueviolet';

})
actorNameInput.addEventListener('blur',(event)=>{
    document.querySelector('.fa-film').style.color='black';

})

//async function to make a post request
async            function postreq(){
res = await fetch('/search',{
    method:'POST',
    headers:{
        'content-type':'application/json'
    },
    body:JSON.stringify({
        name: actorNameInput.value
    })
})
data = await res.json()
return data
}
```

```
    async function search(){
        info.style.display='none';
        actorInfo.style.display='block';
        spinner.style.display='block';
        data = await postreq();
        spinner.style.display='none';
        info.style.display='block';
        actorNameheading.innerText=data.name;
        actorImg.src = data.pic;
        actorOccupation.innerText=data.job;
        actorBday.innerText=data.dob;
    }

</script>

{% endblock %}
```

## Style.css

```
@import
url("https://fonts.googleapis.com/css2?family=Montserrat:wght@200;300;400;
600;700&display=swap");
:root {
  --box-shadow: 0 0.5rem 1.5rem rgba(0, 0, 0, 0.1);
}

* {
  padding: 0;
```

```
  margin: 0;
  font-family: "Montserrat", sans-serif;
  box-sizing: border-box;
  text-decoration: none;
  outline: none;
  border: none;
  text-transform: capitalize;
  transition: all 0.2s linear;
}

section {
  background-color: white;
  max-width: fit-content;
  /* border: 1px solid black; */
  padding: 2em 3em;
  margin: 2rem auto;
  box-shadow: rgba(0, 0, 0, 0.24) 0px 3px 8px;
}

/* .main {
  position: absolute;
  top: 40%;
  left: 50%;
  transform: translate(-50%, -50%);
} */

.icon {
```

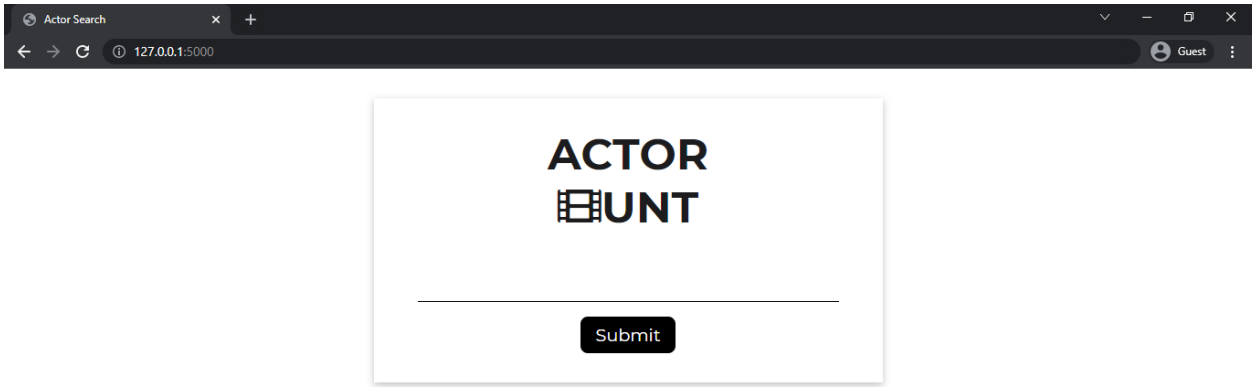
```
  text-align: center;
  font-size: 5rem;
}

.heading {
  width: 100%;
  text-align: center;
  color: #1c1c1e;
  font-size: 3rem;
  padding-bottom: 2rem;
  text-transform: uppercase;
}

.search {
  text-align: center;
  padding: 0.05em 0.5em;
  font-weight: 600;
  font-size: 2rem;
  border-bottom: 1px solid black;
  width: 100%;
}

.searchbar {
  position: relative;
}
.searchbar #underline::after {
  content: "";
```

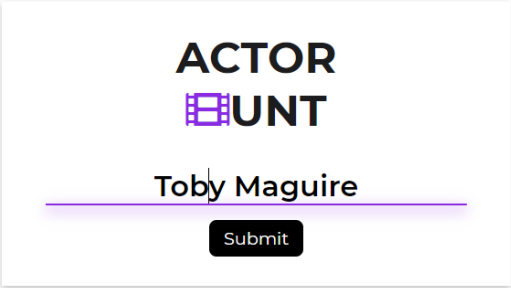
```
  position: absolute;
  z-index: 1;
  left: 0;
  bottom: 0;
  width: 100%;
  height: 2px;
  background-color: blueviolet;
  box-shadow: blueviolet 0px 5px 15px 0.1px;
  transform: scaleX(0);
  transition: all 0.2s linear;
}

.search:focus + #underline::after,
.search:valid + #underline::after {
  transform: scaleX(1);
}

.btn-div {
  display: flex;
  justify-content: center;
}

.btn {
  margin-top: 1rem;
  display: inline-block;
  font-size: 1.2rem;
  color: #fff;
```

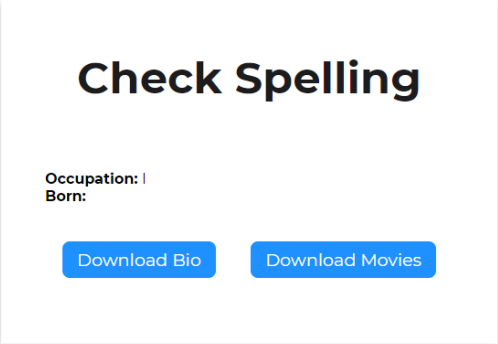
```
  background-color: black;
  border-radius: 0.5rem;
  cursor: pointer;
  padding: 0.5rem 1rem;
  transition: all 0.2s linear;
}

.btn:hover {
  transform: translateY(-0.2em);
  background-color: blueviolet;
  box-shadow: rgba(100, 100, 111, 0.2) 0px 7px 29px;
  /* box-shadow: violet 0px 5px 15px; */
}

/* actor display */
#actor-info {
  display: none;
}
.actorName {
  margin: 0.5em 0;
  text-align: center;
  color: #1c1c1e;
  font-size: 3rem;
}
.actor-img {
  display: block;
  margin: 0 auto;
```

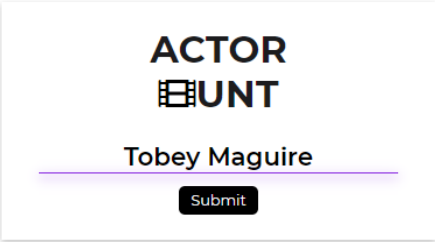
```
  margin-bottom: 3em;
}

#btn-download {
  margin: 5em 0;
  font-size: 0.5rem;
  display: flex;
  justify-content: space-between;
}

#btn-download .btn {
  margin: 0 1em;
  background-color: #1e90ff;
}

#spinner {
  margin: 0 auto;
  display: none;
}

.info {
  display: none;
}
```

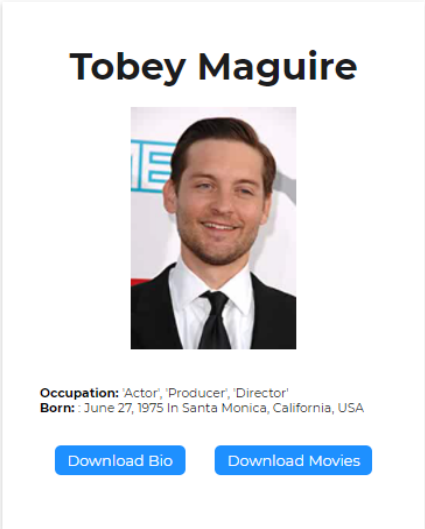
# Output

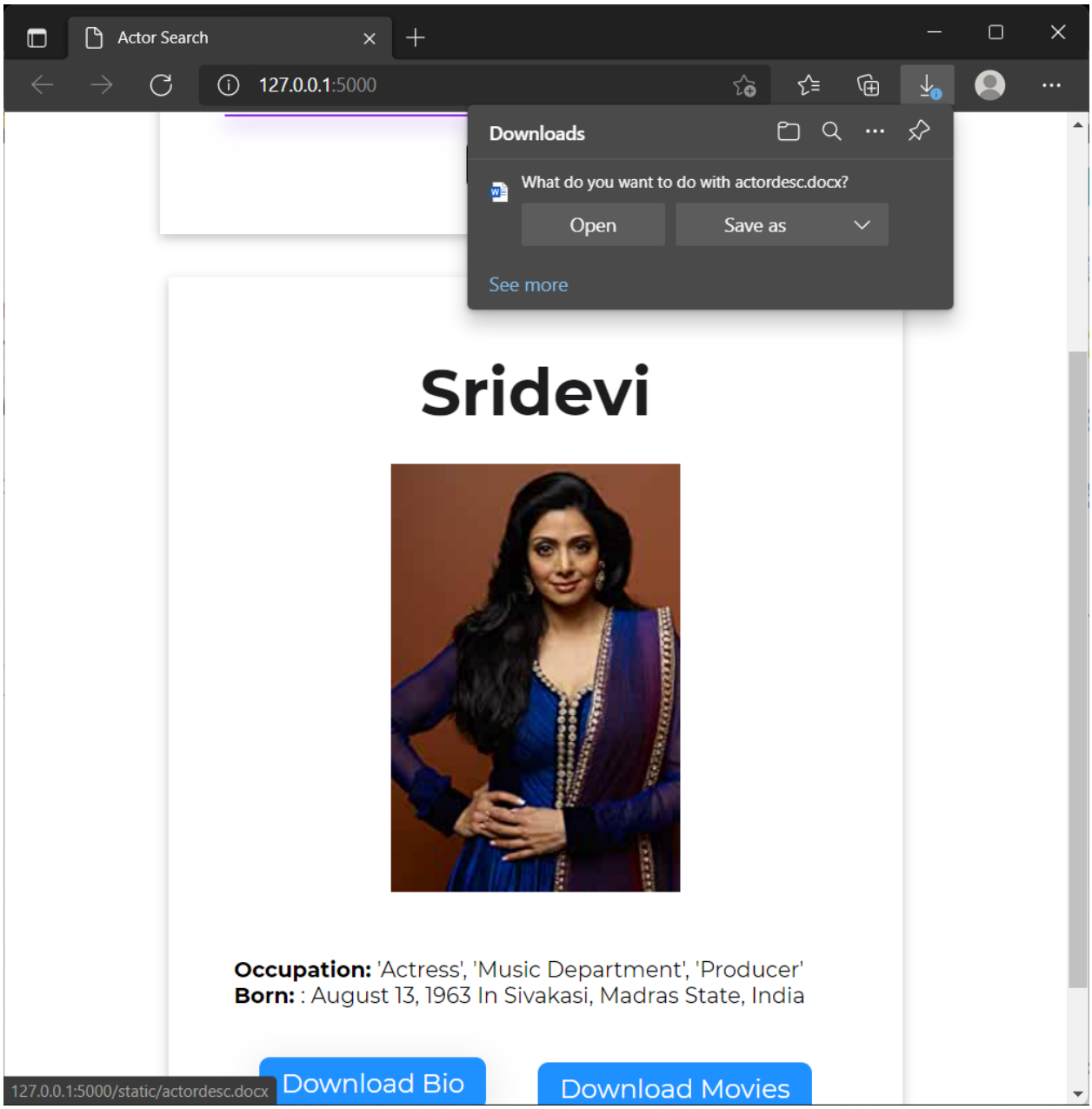


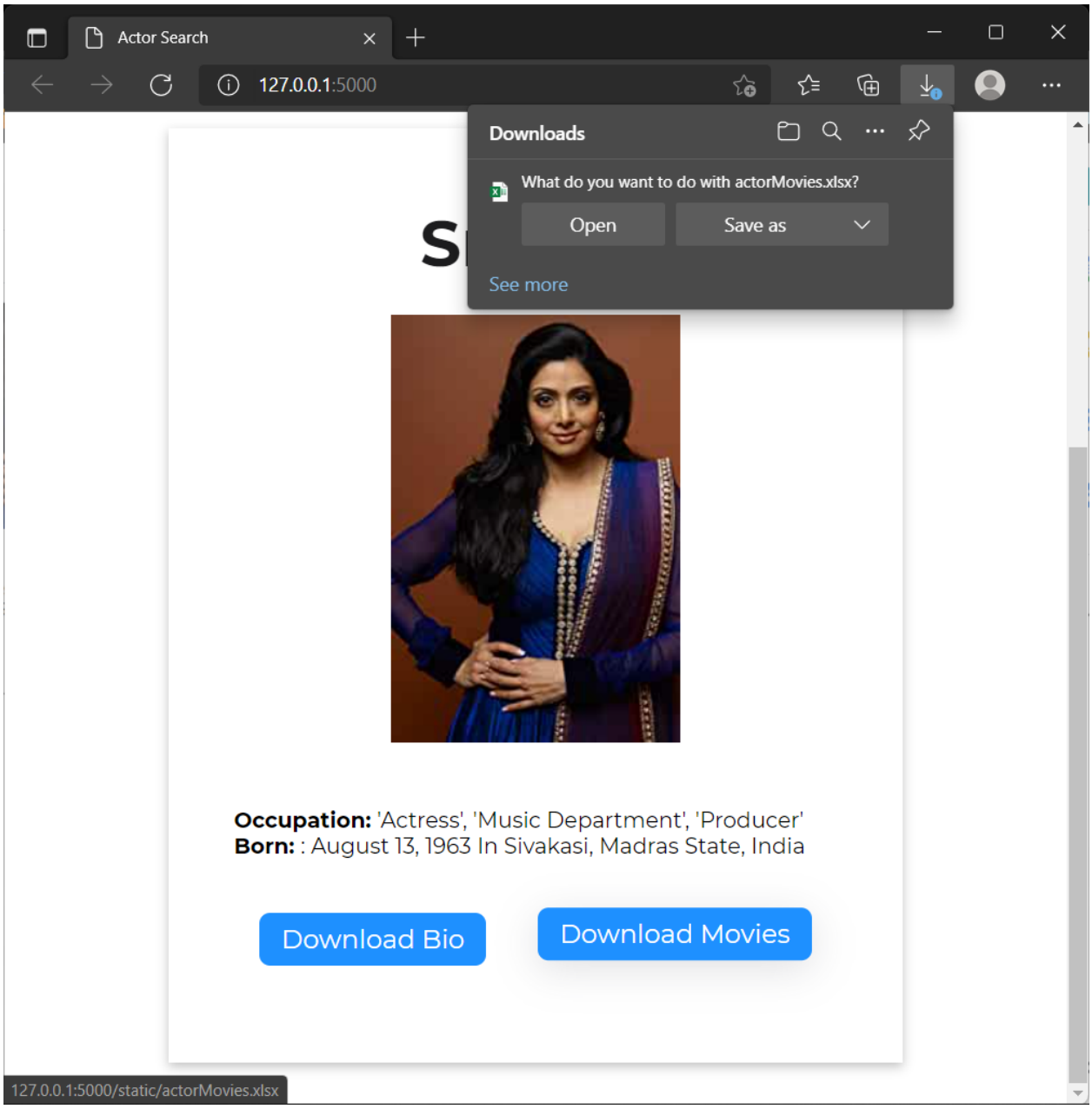

ACTOR
UNT
Toby Maguire
Submit


Check Spelling
Occupation: |
Born:
Download Bio
Download Movies


ACTOR
UNT
Tobey Maguire
Submit


Tobey Maguire
Occupation: 'Actor', 'Producer', 'Director'
Born: : June 27, 1975 In Santa Monica, California, USA
Download Bio
Download Movies


Actor Search
127.0.0.1:5000
Downloads
What do you want to do with actordesc.docx?
Open
Save as
See more
Sridevi
Occupation: 'Actress', 'Music Department', 'Producer'
Born: : August 13, 1963 In Sivakasi, Madras State, India
127.0.0.1:5000/static/actordesc.docx
Download Bio
Download Movies


Actor Search
127.0.0.1:5000
Downloads
What do you want to do with actorMovies.xlsx?
Open
Save as
See more
Occupation: 'Actress', 'Music Department', 'Producer'
Born: : August 13, 1963 In Sivakasi, Madras State, India
Download Bio
Download Movies
127.0.0.1:5000/static/actorMovies.xlsx

actordesc (8).docx (Protected View) - Word

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

# Sridevi

Actor Jobs: ['Actress', 'Music Department', 'Producer']

Born: August 13, 1963 in Sivakasi, Madras State, India

Sridevi was born on August 13, 1963 in her father's hometown of Sivakasi, Tamil Nadu, India. Her mother was from Tirupati, Andhra Pradesh. So Sridevi grew up speaking Telugu and Tamil. She has a sister named Srilatha and a stepbrother named Satish. Her dad passed away during the year 1991, while her mom died during 1997.She started her career at a very early age in 1967 as a child artiste in a Tamil movie 'Kandhan Karunai'. She also starred as a child actress in a Telugu movie 'Bangarakka' in 1977, and in a Malyalam movie 'Kumara Sambhavan' in 1969.She made her foray on the Bollywood tinsel screen in 1975 as a child actor with the smash-hit 'Julie' in which she played the younger sister of the lead actress. Thereafter she started to act in adult roles in Hindi only from 1979. She appeared in 63 movies in Hindi; 62 in Telugu; 58 in Tamil; and 21 in Malyalam - in a career that has

Page 1 of 2 421 words

actorMovies (2).xlsx [Protected View] - Excel

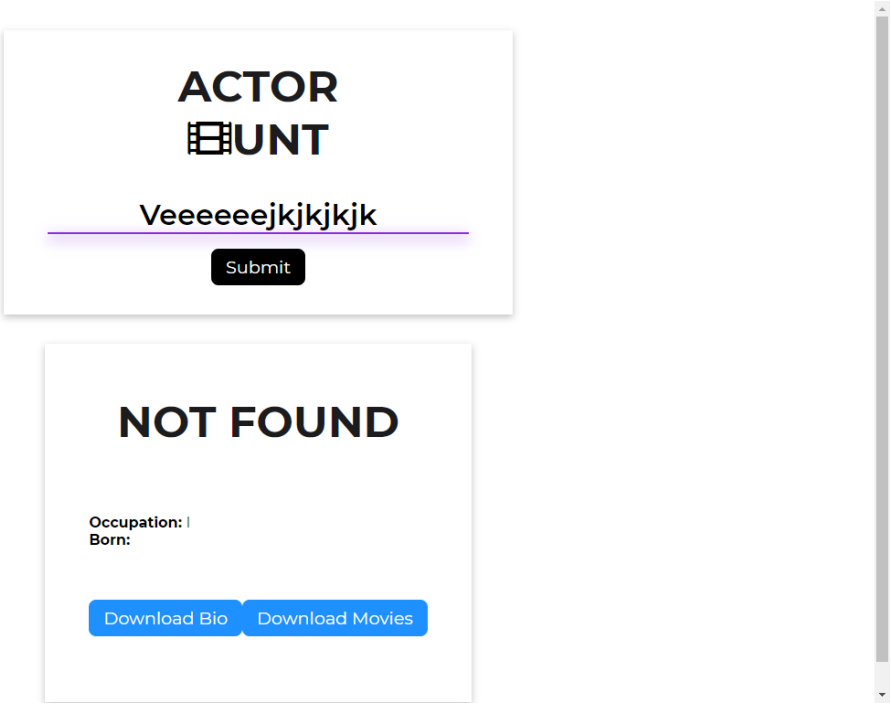
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

A1 Title

| | A | B |
|---|---|---|
| 1 | Title | Year |
| 2 | Mom | 2017 |
| 3 | Mr And Mrs Rangegowda | 2015 |
| 4 | Puli | 2015 |
| 5 | Vicente Ferrer | 2013 |
| 6 | English Vinglish | 2012 |
| 7 | Malini Iyer | 2004 |
| 8 | Meri Biwi Ka Jawab Nahin | 2004 |
| 9 | Kaun Sachcha Kaun Jhootha | 1997 |
| 10 | Judaai | 1997 |
| 11 | Devaraagam | 1996 |
| 12 | Mr. Bechara | 1996 |
| 13 | Army | 1996 |
| 14 | Gopikalyana | 1994 |
| 15 | Chaand Kaa Tukdaa | 1994 |
| 16 | S. P. Parasuram | 1994 |
| 17 | Laadla | 1994 |
| 18 | Chandra Mukhi | 1993 |
| 19 | Gurudev | 1993 |
| 20 | Gumrah | 1993 |
| 21 | Govindha Govindha | 1993 |
| 22 | Roop Ki Rani Choron Ka Raja | 1993 |
| 23 | Laila Majnu | 1992 |
| 24 | Aasmaan Se Gira | 1992 |
| 25 | Heer Ranjha | 1992 |
| 26 | Khuda Gawah | 1992 |
| 27 | Ennavo Nadikkaradhu | 1991 |
| 28 | Garajna | 1991 |

Sheet

Ready




Github repository:

https://github.com/M-Shehzad/adp-miniproject

Here is the demo video of the working of the project :

https://github.com/M-Shehzad/adp-miniproject/blob/main/Actor%20Search.mp4

# References


- www.stackoverflow.com
- www.youtube.com
- https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- https://www.sqlite.org/docs.html
- https://fontawesome.com/