

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590018



A DBMS Mini Project Report

on

" MOVIE BUZZ "

*Submitted in partial fulfillment of the requirements for the V semester
and award of the degree of Bachelor of Engineering in Computer Science
and Engineering of Visvesvaraya Technological University, Belagavi*

Submitted by:

Pavan Kumar M 1RN20CS191

Dheeraj M 1RN20CS041

Under the Guidance of:

Mr. K Sunil Kumar

Assistant Professor

Dept. of CSE



Department of Computer Science and Engineering

(Accredited by NBA upto 30-06-2025)

RNS Institute of Technology

Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560 098

2022-2023

RNS Institute of Technology

Channasandra, Dr. Vishnuvardhan Road, Bengaluru-560098

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

(Accredited by NBA upto 30-06-2025)



CERTIFICATE

Certified that the mini project work entitled “**MOVIE BUZZ**” has been successfully carried out by “**Pavan Kumar M**” bearing USN “**1RN20CS191**” and “**Dheeraj M**” bearing USN “**1RN20CS041**”, bonafide students of “**RNS Institute of Technology**” in partial fulfillment of the requirements for the 5th semester of “**Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University**”, Belagavi, during academic year 2022-2023. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the DBMS laboratory requirements of 5th semester BE, CSE.

Signature of the Guide
Mr. K Sunil Kumar
Assistant Professor
Dept. of CSE

Signature of the HoD
Dr. Kiran
Professor and HoD
Dept. of CSE

Signature of the Principal
Dr. M K Venkatesha
Principal

External Viva:

Name of the Examiners

Signature with Date

- 1.
- 2.

Acknowledgement

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project work. We would like to take this opportunity to thank them all.

We are grateful to Management and **Dr. M K Venkatesha**, Principal, RNSIT, Bangalore, for his support towards completing this mini project.

We would like to thank **Dr. Kiran P**, Professor & Head, Department of Computer Science & Engineering, RNSIT, Bangalore, for his valuable suggestions and expert advice.

We deeply express our sincere gratitude to our guide **Mr. K Sunil Kumar**, Assistant Professor, Department of CSE, RNSIT, Bangalore, for his able guidance, regular source of encouragement and assistance throughout this project.

We would like to thank all the teaching and non-teaching staff of Department of Computer Science & Engineering, RNSIT, Bengaluru-98 for their constant support and encouragement.

Abstract

The project “Movie Buzz” is to be used by movie viewer for streaming movies of their choice, Achieving this objective is difficult using a manual system as the information about movies are scattered throughout the web, can be redundant and collecting the relevant information may be very time-consuming. All these problems are solved using this project.

The project has been thought about from a viewer point of view and revolves around the usual requirements for a viewer about his/her choice of movies. The objective of this project is making an interactive user-friendly website which makes the registration of viewers, admins and their modification, deletion in an effective, easy way. Every entity can be easily searched according to their name. We have implemented the idea by making a web application that includes table visualization. Any admin using the website will be able to insert all the details, update and delete them when needed. The UI has been made very simple to provide ease of access for all types of users.

This project requires HTML, CSS in the frontend, Python Flask for backend and database connectivity and MySQL for database management. We seek to expand the project by having a fully real-life model of the department in the college.

Contents

Acknowledgement	i
Abstract	ii
List of Figures	v
1 Introduction To Database Management System	1
1.1 Introduction	1
1.2 History of DBMS	2
1.3 Characteristics of DBMS	3
1.3.1 Self-Describing Nature of a Database System	3
1.3.2 Insulation between Programs and Data, and Data Abstraction	4
1.3.3 Support of Multiple Views of the Data	4
1.3.4 Sharing of Data and Multiuser Transaction Processing	5
1.4 Applications of DBMS	5
2 Requirement Analysis	7
2.1 Hardware Requirements	7
2.2 Software Requirements	7
2.3 Functional Requirements	8
2.3.1 Major Entities	8
2.3.2 End User Requirements	8
3 Database Design	9
3.1 Entities, Attributes and Relationships	9
3.2 Identify Major entities, attributes and relationships	10
3.3 ER Schema	10
3.4 Schema Diagram	11

4	Description Of Tools And Technologies	12
4.1	HTML	12
4.2	Bootstrap - A CSS Framework	12
4.3	Javascript	13
4.4	Python	14
4.5	MYSQL	14
5	Flask-MYSQL Database Connectivity	15
5.1	Database Schema	15
5.2	5 Steps to connect to the database in flask	16
5.3	Triggers	16
6	Implementation	17
6.1	FlaskApp - Python	17
6.2	Backbone - HTML, CSS,Bootstrap,Jinja	18
6.3	Movie Structure - HTML,CSS	18
6.4	Database	19
7	Snapshots	20
8	Conclusion and Future Enhancements	23
8.1	Conclusion	23
8.2	Future Enhancements	24
	References	25

List of Figures

3.1	Movie Table	9
3.2	Entity Relationship Diagram	10
3.3	Relational Schema	11
6.1	Backend Code	17
6.2	Frontend	18
6.3	Movie Page	18
6.4	Movie Table	19
6.5	Admin Table	19
6.6	Cast Table	19
7.1	Login Form	20
7.2	Movie Details	21
7.3	Home Page	21
7.4	Cast Page	22
7.5	Movie Page	22

Chapter 1

Introduction To Database Management System

1.1 Introduction

Databases and database technology have a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used, including business, electronic commerce, engineering, medicine, genetics, law, education, and library science. The word database is so commonly used that we must begin by defining what a database is. Our initial definition is quite general. A database is a collection of related data. By data, we mean known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of the people you know. You may have recorded this data in an indexed address book or you may have stored it on a hard drive, using a personal computer and software such as Microsoft Access or Excel. This collection of related data with an implicit meaning is a database. The preceding definition of database is quite general; for example, we may consider the collection of words that make up this page of text to be related data and hence to constitute a database. However, the common use of the term database is usually more restricted. A database has the following implicit properties:

- A database represents some aspect of the real world, sometimes called the miniworld or the universe of discourse (UoD). Changes to the miniworld are reflected in the database.
- A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database

- A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications. Defining a database involves specifying the data types, structures, and constraints of the data to be stored in the database. The database definition or descriptive information is also stored by the DBMS in the form of a database catalog or dictionary; it is called meta-data. Constructing the database is the process of storing the data on some storage medium that is controlled by the DBMS. Manipulating a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data. Sharing a database allows multiple users and programs to access the database simultaneously.

1.2 History of DBMS

In 1959, the TX-2 computer was developed at MIT's Lincoln Laboratory. The TX-2 integrated a number of new man-machine interfaces. A light pen could be used to draw sketches on the computer using Ivan Sutherland's revolutionary Sketchpad software.[4] Using a light pen, Sketchpad allowed one to draw simple shapes on the computer screen, save them and even recall them later. The light pen itself had a small photoelectric cell in its tip. This cell emitted an electronic pulse whenever it was placed in front of a computer screen and the screen's electron gun fired directly at it. By simply timing the electronic pulse with the current location of the electron gun, it was easy to pinpoint exactly where the pen was on the screen at any given moment. Once that was determined, the computer could then draw a cursor at that location. Also in 1961 another student at MIT, Steve Russell, created the first video game, E. E. Zajac, a scientist at Bell Telephone Laboratory (BTL), created a film called "Simulation of a two-giro gravity attitude control system" in 1963. During 1970s, the first major advance in 3D computer graphics was created at UU by these early pioneers, the hidden-surface algorithm. In order to draw a representation of a 3D object on the screen, the computer must determine which surfaces are "behind" the object from the viewer's perspective, and thus should be "hidden" when the computer creates (or renders) the image. In the 1980s, artists and graphic designers began to see the personal computer, particularly the Commodore Amiga and Macintosh, as a serious design tool, one that could save time and draw more accurately than other methods. In the late 1980s, SGI computers were used to create some of the first fully computer-generated short films at Pixar.

1.3 Characteristics of DBMS

A number of characteristics distinguish the database approach from the much older approach of programming with files. In traditional file processing, each user defines and implements the files needed for a specific software application as part of programming the application. For example, one user, the grade reporting office, may keep files on students and their grades. Programs to print a student's transcript and to enter new grades are implemented as part of the application. A second user, the accounting office, may keep track of students' fees and their payments. Although both users are interested in data about students, each user maintains separate files—and programs to manipulate these files—because each requires some data not available from the other user's files. This redundancy in defining and storing data results in wasted storage space and in redundant efforts to maintain common up-to-date data. In the database approach, a single repository maintains data that is defined once and then accessed by various users. In file systems, each application is free to name data elements independently. In contrast, in a database, the names or labels of data are defined once, and used repeatedly by queries, transactions, and applications. The main characteristics of the database approach versus the file-processing approach are the following:

- Self-describing nature of a database system.
- Insulation between programs and data, and data abstraction.
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing.

1.3.1 Self-Describing Nature of a Database System

A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints. This definition is stored in the DBMS catalog, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data. The information stored in the catalog is called meta-data, and it describes the structure of the primary database. The catalog is used by the DBMS software and also by database users who need information about the database structure. A general-purpose DBMS software package is not written for a specific database application. Therefore, it must refer to the catalog to know the structure of the files in a specific database, such as the type and format of data it will access. The DBMS software must

work equally well with any number of database applications—for example, a university database, a banking database, or a company database—as long as the database definition is stored in the catalog. In traditional file processing, data definition is typically part of the application programs themselves. Hence, these programs are constrained to work with only one specific database, whose structure is declared in the application programs. For example, an application program written in C++ may have structure or class declarations, and a COBOL program has data division statements to define its files. Whereas file-processing software can access only specific databases, DBMS software can access diverse databases by extracting the database definitions from the catalog and using these definitions.

1.3.2 Insulation between Programs and Data, and Data Abstraction

In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access that file. By contrast, DBMS access programs do not require such changes in most cases. The structure of data files is stored in the DBMS catalog separately from the access programs. We call this property program-data independence. In some types of database systems, such as object-oriented and object-relational systems users can define operations on data as part of the database definitions. An operation (also called a function or method) is specified in two parts. The interface (or signature) of an operation includes the operation name and the data types of its arguments (or parameters). The implementation (or method) of the operation is specified separately and can be changed without affecting the interface. User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented. This may be termed program-operation independence. The characteristic that allows program-data independence and program-operation independence is called data abstraction. A DBMS provides users with a conceptual representation of data that does not include many of the details of how the data is stored or how the operations are implemented. Informally, a data model is a type of data abstraction that is used to provide this conceptual representation. The data model uses logical concepts, such as objects, their properties, and their interrelationships, that may be easier for most users to understand than computer storage concepts. Hence, the data model hides storage and implementation details that are not of interest to most database users.

1.3.3 Support of Multiple Views of the Data

A database typically has many users, each of whom may require a different perspective or view of the database. A view may be a subset of the database or it may contain virtual data that is derived

from the database files but is not explicitly stored. Some users may not need to be aware of whether the data they refer to is stored or derived. A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views.

1.3.4 Sharing of Data and Multiuser Transaction Processing

A multiuser DBMS, as its name implies, must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database. The DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct. For example, when several reservation agents try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one agent at a time for assignment to a passenger. These types of applications are generally called online transaction processing (OLTP) applications. A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly and efficiently. The concept of a transaction has become central to many database applications. A transaction is an executing program or process that includes one or more database accesses, such as reading or updating of database records. Each transaction is supposed to execute a logically correct database access if executed in its entirety without interference from other transactions. The DBMS must enforce several transaction properties. The isolation property ensures that each transaction appears to execute in isolation from other transactions, even though hundreds of transactions may be executing concurrently. The atomicity property ensures that either all the database operations in a transaction are executed or none are.

1.4 Applications of DBMS

Applications where we use Database Management Systems are:

- **Telecom:** There is a database to keep track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.
- **Industry:** Where it is a manufacturing unit, warehouse or distribution centre, each one needs a database to keep the records of ins and outs. For example distribution centre should keep a track of the product units that supplied into the centre as well as the products that got delivered out from the distribution centre on each day; this is where DBMS comes into picture.

- **Banking System:** For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.
- **Education Sector:** Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner.
- **Online Shopping:** You must be aware of the online shopping websites such as Amazon, Flip kart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

Chapter 2

Requirement Analysis

2.1 Hardware Requirements

The Hardware requirements are very minimal and the program can be run on most of the machines.

Processor : Intel Atom® processor or Intel® Core™ i3 processor

Processor Speed : 2.4 GHz

RAM : 1 GB

Storage Space : 40 GB

Monitor Resolution : 1024*768 or 1336*768 or 1280*1024

2.2 Software Requirements

Operating System - Windows 7 or later

Text Editor - Visual Studio Code

SQL - MySQL

Language - Python Flask

Browser - Any one which supports HTML and Python Flask

Additional tool - php My Admin

2.3 Functional Requirements

2.3.1 Major Entities

USER: To store or modify the user details in user entity or table.

ADMIN: We can store or modify the admin details in admin entity or table.

MOVIES: To store the details of the movie.

MOVIE CAST: To store the details of the casting details of a movie present in movie entity.

DIRECTOR: To store the details of a directed who have directed one or movies which are present in movie entity.

PRODUCTION: To store the details of a production houses who have produced one or movies which are present in movie entity.

RATING: To store the rating of the movie present in movie entity.

2.3.2 End User Requirements

1. Main Goals:

- Our motto is to provide user friendly interface for cinema lovers where they can view movies based on different categories.
- Hereby, our main objective is the ease of acces of important data considering how fast the data exchange takes place.

2. Ease of access:

- The details can be easily added, deleted or updated without any hassle.
- Our software will perform and fulfill all the tasks that any administrator would desire.

3. Saving Lookup Time:

- The person looking for the movie doesn't need to go through the whole database to do small operation.

Chapter 3

Database Design

3.1 Entities, Attributes and Relationships

The database, called moviebuzz, will have nine tables, user, admin, movies, cast, language, moviecast, director, production and rating. Each will hold information revolving around a movie only. The two tables will be linked through a foreign key. Since one director can direct many movies, We thought it only right to insert a foreign key director-id into the movie table. In addition, we have made the user account to have admin privilege(for some users). This will enable us to focus more on the programming than on particulars of the database.

The movie table has the following fields:

Figure 3.1: Movie Table

Field	Description
movie_id	unique id of the movie
title	stores movie name
year	stores release year of the movie
director-id	stores id of the director who has directed the movie
production	stores the name of production house
link	has link of the movie

3.2 Identify Major entities, attributes and relationships

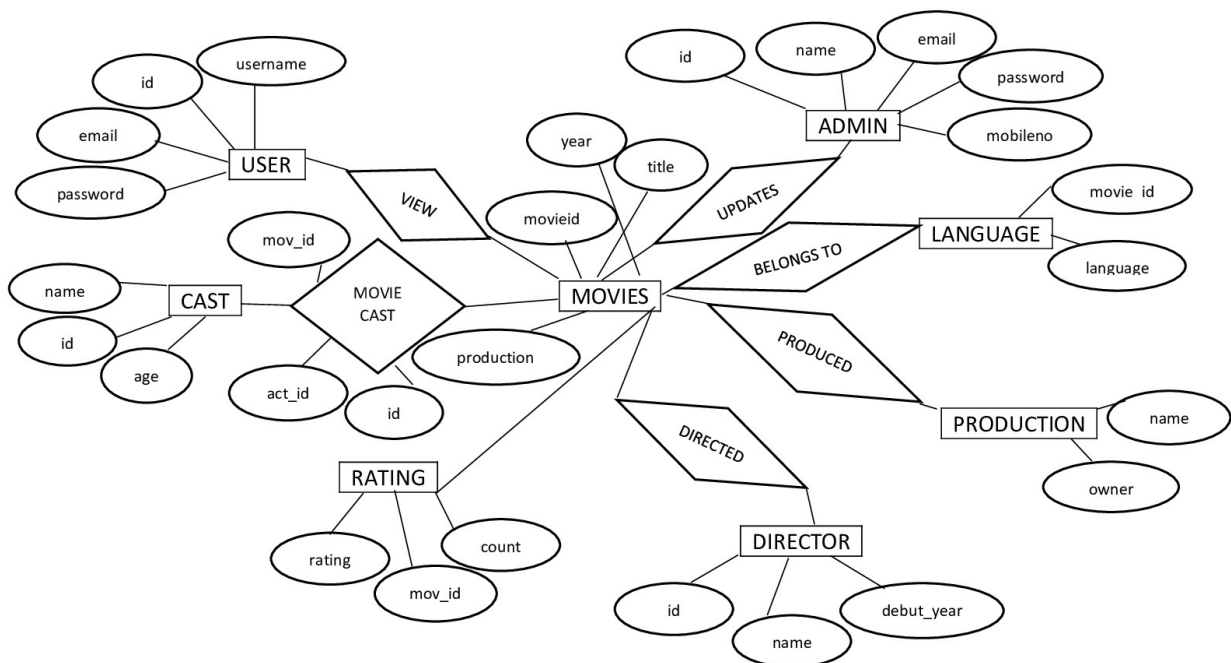
- Login page to give access to privileged users and normal users.
- Adding movie details by the user(admin).
- Admins can check all details of movies and update them.
- Admins can delete the entities in the tables.
- Easy search facilities to get the required information.

3.3 ER Schema

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

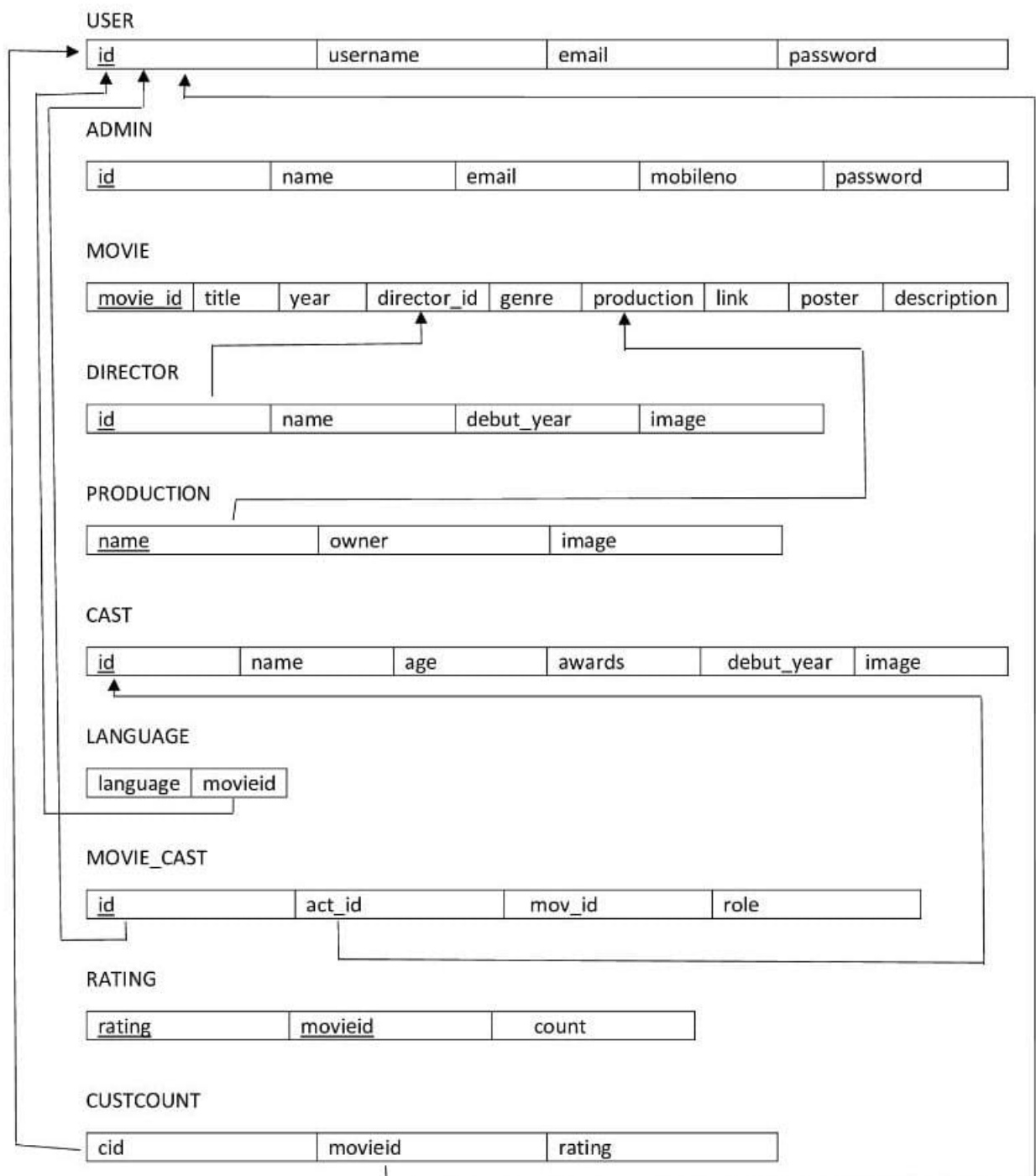
Figure 3.2: Entity Relationship Diagram



3.4 Schema Diagram

A relational schema is a set of relational tables and associated items that are related to one another. All of the base tables, views, indexes, domains, user roles, stored modules, and other items that a user creates to fulfill the data needs of a particular enterprise or set of applications belong to one schema.

Figure 3.3: Relational Schema



Chapter 4

Description Of Tools And Technologies

4.1 HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects, such as interactive forms, may be embedded into the rendered page. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as `` and `<input />` introduce content into the page directly. Others such as `<p>...</p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page. HTML can embed programs written in a scripting language such as JavaScript which affect the behavior and content of web pages. Inclusion of CSS defines the look and layout of content.

4.2 Bootstrap - A CSS Framework

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other

media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Bootstrap is a free and open-source front-end web framework used for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only.

Bootstrap is the second most-starred project on GitHub, with more than 111,600 stars and 51,500 forks.

4.3 Javascript

JavaScript, often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.

Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and this is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets.

Although there are strong outward similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in

design; JavaScript was influenced by programming languages such as Self and Scheme.

4.4 Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community after 30 years.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation.

4.5 MYSQL

MySQL is a relational database management system (RDBMS) developed by Oracle that is based on structured query language (SQL).

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or a place to hold the vast amounts of information in a corporate network. In particular, a relational database is a digital store collecting data and organizing it according to the relational model. In this model, tables consist of rows and columns, and relationships between data elements all follow a strict logical structure. An RDBMS is simply the set of software tools used to actually implement, manage, and query such a database.

MySQL is integral to many of the most popular software stacks for building and maintaining everything from customer-facing web applications to powerful, data-driven B2B services. Its open-source nature, stability, and rich feature set, paired with ongoing development and support from Oracle, have meant that internet-critical organizations such as Facebook, Flickr, Twitter, Wikipedia, and YouTube all employ MySQL backends.

Chapter 5

Flask-MYSQL Database Connectivity

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program. Flask is commonly used with MongoDB, which gives it more control over databases and history.

Applications that use the Flask framework include Pinterest, LinkedIn, and the community web page for Flask itself.

5.1 Database Schema

A database schema is the logical representation of a database, which shows how the data is stored logically in the entire database. It contains list of attributes and instruction that informs the database engine that how the data is organized and how the elements are related to each other.

A database schema contains schema objects that may include tables, fields, packages, views, relationships, primary key, foreign key.

In actual, the data is physically stored in files that may be in unstructured form, but to retrieve it and use it, we need to put it in a structured form. To do this, a database schema is used. It provides knowledge about how the data is organized in a database and how it is associated with other data.

5.2 5 Steps to connect to the database in flask

There are 5 steps to connect any flask application with the database in MYSQL. They are as follows:

- Import the dependencies
- Open a connection to an MYSQL database file
- Load the data from the dataframe into the database
- Make functions to execute different SQL queries
- Closing connection

5.3 Triggers

A trigger is a named database object that is associated with a table, and that activates when a particular event occurs for the table. Some uses for triggers are to perform checks of values to be inserted into a table or to perform calculations on values involved in an update.

A trigger is defined to activate when a statement inserts, updates, or deletes rows in the associated table. These row operations are trigger events. For example, rows can be inserted by INSERT or LOAD DATA statements, and an insert trigger activates for each inserted row. A trigger can be set to activate either before or after the trigger event. For example, you can have a trigger activate before each row that is inserted into a table or after each row that is updated.

MySQL triggers activate only for changes made to tables by SQL statements. This includes changes to base tables that underlie updatable views. Triggers do not activate for changes to tables made by APIs that do not transmit SQL statements to the MySQL Server. This means that triggers are not activated by updates made using the NDB API.

BEFORE triggers run the trigger action before the triggering statement is run. AFTER triggers run the trigger action after the triggering statement is run.

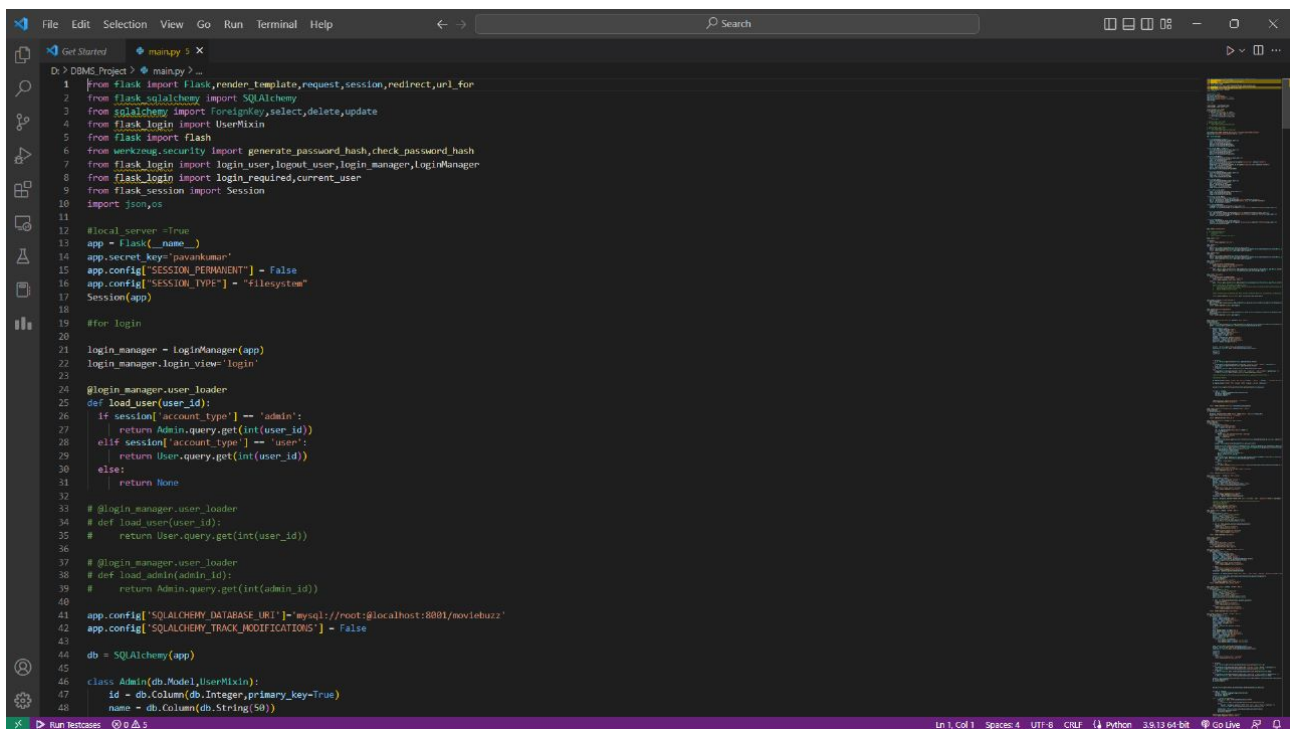
Example: CREATE TRIGGER 'Average' BEFORE INSERT ON 'custcount' FOR EACH ROW
BEGIN UPDATE rating set rating=((NEW.rating+(rating*COUNT))/(COUNT+1)), COUNT=COUNT+1
WHERE rating.movieid = NEW.movieid ;

Chapter 6

Implementation

6.1 FlaskApp - Python

Figure 6.1: Backend Code



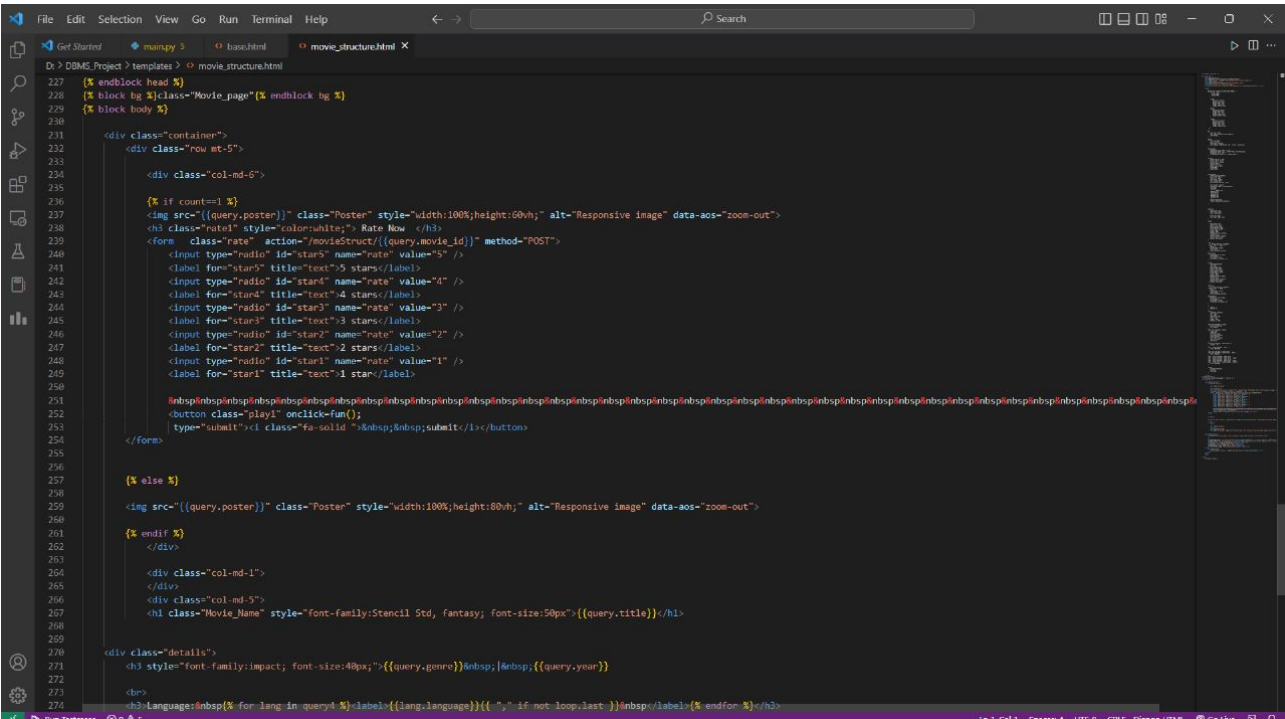
```
1 from flask import Flask, render_template, request, session, redirect, url_for
2 from flask_sqlalchemy import SQLAlchemy
3 from sqlalchemy import ForeignKey, select, delete, update
4 from flask_login import UserMixin
5 from flask import flash
6 from werkzeug.security import generate_password_hash, check_password_hash
7 from flask_login import login_user, logout_user, login_manager, LoginManager
8 from flask_login import login_required, current_user
9 from flask_session import Session
10 import json, os
11
12 #local server = True
13 app = Flask(__name__)
14 app.secret_key = 'pavankumar'
15 app.config['SESSION_PERMANENT'] = False
16 app.config['SESSION_TYPE'] = 'filesystem'
17 Session(app)
18
19 #for login
20
21 login_manager = LoginManager(app)
22 login_manager.login_view = 'login'
23
24 @login_manager.user_loader
25 def load_user(user_id):
26     if session['account_type'] == 'admin':
27         return Admin.query.get(int(user_id))
28     elif session['account_type'] == 'user':
29         return User.query.get(int(user_id))
30     else:
31         return None
32
33 # @login_manager.user_loader
34 # def load_user(user_id):
35 #     return User.query.get(int(user_id))
36
37 # @login_manager.user_loader
38 # def load_admin(admin_id):
39 #     return Admin.query.get(int(admin_id))
40
41 app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:localhost:8001/moviebuzz'
42 app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
43
44 db = SQLAlchemy(app)
45
46 class Admin(db.Model, UserMixin):
47     id = db.Column(db.Integer, primary_key=True)
48     name = db.Column(db.String(50))
```


6.2 Backbone - HTML, CSS,Bootstrap,Jinja

Figure 6.2: Frontend

6.3 Movie Structure - HTML,CSS

Figure 6.3: Movie Page



6.4 Database

Figure 6.4: Movie Table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 movie_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 title	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	3 year	year(4)			No	None			Change Drop More
<input type="checkbox"/>	4 director_id	int(11)			No	None			Change Drop More
<input type="checkbox"/>	5 genre	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	6 production	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	7 link	varchar(1000)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	8 poster	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	9 description	varchar(1000)	utf8mb4_general_ci		No	None			Change Drop More

Figure 6.5: Admin Table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 name	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	3 email	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	4 mobilen	bigint(11)			No	None			Change Drop More
<input type="checkbox"/>	5 password	varchar(1000)	utf8mb4_general_ci		No	None			Change Drop More

Figure 6.6: Cast Table

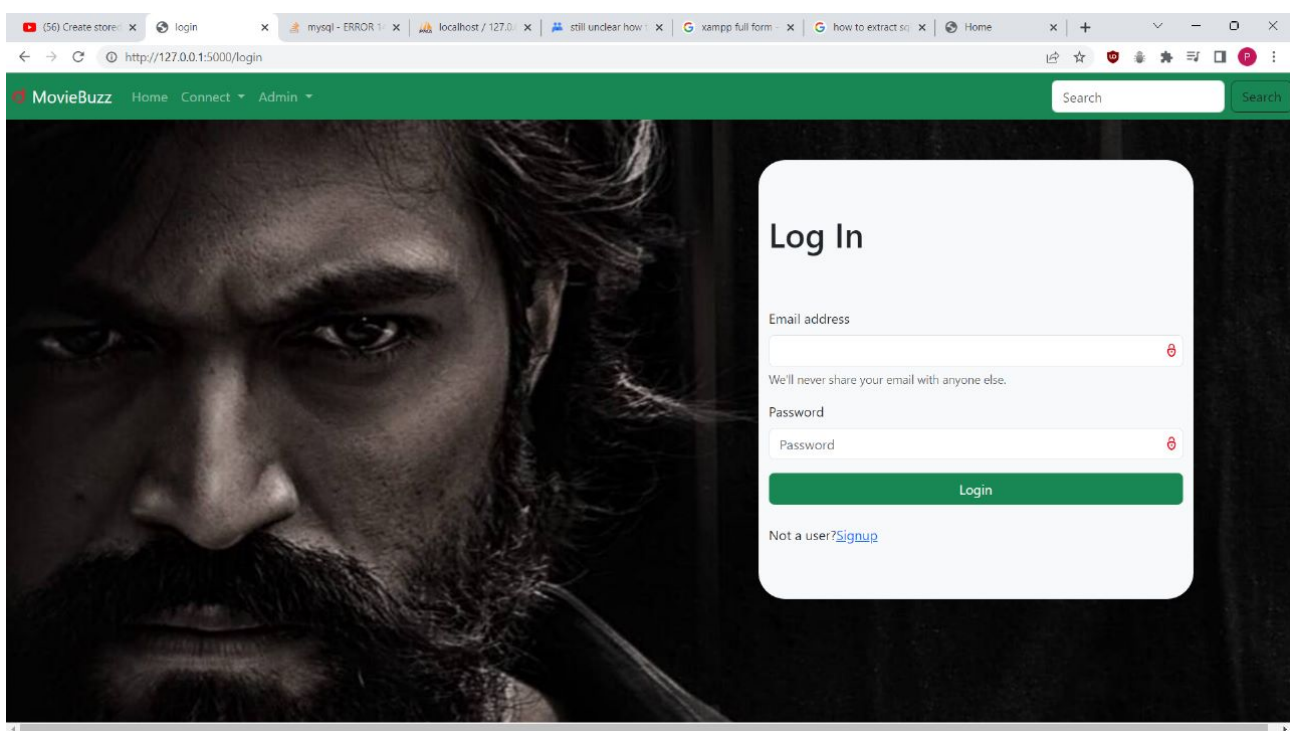
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 name	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	3 age	int(11)			No	None			Change Drop More
<input type="checkbox"/>	4 awards	varchar(200)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	5 debut_year	year(4)			No	None			Change Drop More
<input type="checkbox"/>	6 image	varchar(100)	utf8mb4_general_ci		Yes	../static/default.png			Change Drop More

Chapter 7

Snapshots

This page enables both user and admin login. User has to enter email-id ,password for authentication. Once the user is authenticated he is redirected to home page.

Figure 7.1: Login Form



The screenshot shows a web browser window with the URL `http://127.0.0.1:5000/login`. The page features a green header bar with the "MovieBuzz" logo and navigation links for "Home", "Connect", and "Admin". A search bar is located on the right side of the header. The main content area has a dark background with a close-up image of a man's face. Overlaid on this is a white "Log In" form. The form contains two input fields: "Email address" and "Password", both with red lock icons indicating they are required. Below the email field is a text line: "We'll never share your email with anyone else." Below the password field is a green "Login" button. At the bottom of the form, there is a link: "Not a user? [Signup](#)".

Figure 7.2: Movie Details

ID	Title	Year	Director	Genre	Production	Link	Poster	Description	Edit	Delete
21	K.G.F Chapter 2	2022	Prashanth Neel	Action	Hombale Films	https://youtu.be/Qah9sSIXJqk	../static/kgf.jpg	Rocky takes control of the Kolar Gold Fields and his newfound power makes the government as well as his enemies jittery. However, he still has to confront Ramika, Adheera and Inayat.	Edit	Delete
22	Kantara	2022	Rishab Shetty	Action	Hombale Films	https://youtu.be/ppYoloW73PI	../static/kantara.jpg	A fiery young man clashes with an unflinching forest officer in a south Indian village where spirituality, fate and folklore rule the lands.	Edit	Delete
23	James	2022	Chethan Kumar	Action	Kishore Productions	https://youtu.be/TPm8djlUtY	../static/james.jpg	The life of Santosh changes after Vijay, owner of a criminal organisation called Gayakwad Syndicate, enlists his help in his business dealings.	Edit	Delete
24	Kirik Party	2016	Rishab Shetty	Comedy	Paramvah Studios	https://youtu.be/lfvnBER_6sQ	../static/Kirik Party.jpg	Karna, a first-year engineering student, falls in love with Saanvi, a final-year pupil from his college. However, a tragic event changes his perception towards life and he mends his ways.	Edit	Delete
25	777 Charlie	2022	Kiranraj K	Comedy	Paramvah Studios	https://youtu.be/REqFOV2A7sl	../static/777 Charlie.jpg	Hope emerges when Charlie, an abused dog, stumbles upon Dharma, a jaded man. As their connection grows, horrific news sends the two friends on a journey together.	Edit	Delete
26	Tagaru	2018	Suri	Action	Venus Entertainers	https://youtu.be/oSy91wM2KVI	../static/Tagaru.jpg	Shiva, an encounter specialist, takes an oath to eradicate the wrongdoers from society. Meanwhile, Punarvasu, a young woman,	Edit	Delete

Figure 7.3: Home Page

Hi! PAVAN, Welcome to Movies page

K.G.F Chapter 2

Rocky takes control of the Kolar Gold Fields and his newfound power makes the government as well as his enemies jittery. However, he still has to confront Ramika, Adheera and Inayat.

[View Now](#)

Kantara

A fiery young man clashes with an unflinching forest officer in a south Indian village where spirituality, fate and folklore rule the lands.

[View Now](#)

James

The life of Santosh changes after Vijay, owner of a criminal organisation called Gayakwad Syndicate, enlists his help in his business dealings.

[View Now](#)

Kirik Party

Karna, a first-year engineering student, falls in love with Saanvi, a final-year pupil from his college. However, a tragic event changes his perception towards life and he mends his ways.

[View Now](#)

Figure 7.4: Cast Page



Figure 7.5: Movie Page



Chapter 8

Conclusion and Future Enhancements

8.1 Conclusion

The Movie Buzz has been designed to maintain the records of all the aspects of Movies with respect to viewers choices.

This project was designed to model the working of an entertainment website. It also contains the details about the technicians who have worked in movies, the ratings and cast information are also saved in the database. The database system project includes triggers that allow the user to get the average of rating and stars every time those values are modified. It also includes a stored procedure to search a movie from the database using its title.

The project provides simple retrieval techniques and easy updating and deletion operations thus helping inefficient maintenance of records.

8.2 Future Enhancements

Just like any other developer this project is the most basic website built using simple tools. We seek to increase the dynamic of the project by adding various other innovations to it. Such innovations would seem possible only with time, which we lack but regardless we strive to complete what we started.

We believe that apart from the present functionalities we can add :

“RECOMMENDATIONS” which will recommend movies for user based on his taste.

“EFFICIENT SEARCH” feature through user can search movies by cast name,director name,production name etc.

“SUBSCRIPTION SYSTEM” so that we provide exclusive access to subscribed viewers.

Apart from these changes we are open to various suggestions and hope to implement them soon so that this website can be used by the users for their needs.

References

- [1] Edward Angel, “*Fundamentals of Database Systems*”, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
- [2] F.S. Hill, “*Database management systems, Ramakrishnan, and Gehrke*”, 3rd Edition, 2014, McGraw Hill.
- [3] @online OpenGL Official ,<https://www.w3schools.com/css/default.asp>
- [4] @online OpenGL Overview , <https://www.flask.palletsprojects.com>
- [5] @online OpenGL Official ,<https://www.geeksforgeeks.org/css/>