Methodology Chapter

Bilal Ali 21K-3153

Arham Mehmood 21K-3201

Abdur Rehman 21K-4529

Huzaifa Ahmed 21K-3403

Ehaab Tariq 21K-3372

The project we have chosen to write our methodology report on is the **Movie Recommendation System** we designed in our Database Course.

**2.Methodology:**

This section outlines the requirement analysis, the frontend, backend, and database design and also the implementation of said design for our Movie Recommendation System.

*2.1 Requirement Analysis:*

The basic requirements for our project are for each user to have their own account, browse from a large selection of movies, add movies they like or have seen to their watchlist and have our personalized recommendation system recommend movies it thinks the user will like based on their watchlist.

These requirements are expanded further and split into functional and non-functional requirements in the following section:

*2.1.1 Functional Requirements:*

1. **Login Facility:**

* Users should be able to create their own private accounts
* Users should be able to log in to their accounts using their credentials.

1. **Large Selection of Movies:**

* Our system should provide a large number of movies for users to browse through
* Users should be able to search for movies based on their genre, release date, rating and more

1. **Rating and Commenting on Movies**

* Users should be able to rate movies from 1-10
* Users should be able to provide feedback on movies through comments and see what other users have commented as well.

1. **Adding Movies to Watchlist:**

* Users should be able to add the movies they have seen to their watchlist
* Users should be able to edit their watchlists, adding or removing movies as they see fit

1. **Robust Recommendation System:**

* The recommendation system should provide personalized movie recommendations based on the user’s watchlist

*2.1.2 Non Functional Requirements:*

1. **Security:**

* The login system should be secure, using encryption for passwords and proper authentication methods.
* Ratings and comments should be stored securely to prevent tampering.

1. **Scalability:**

* The system should be able to handle a large number of users and a large number of movies efficiently.

1. **Performance:**

* The system should be quick and responsive for each user in interactions such as browsing, searching and adding movies to watchlist.
* The recommendation system should not take too long in its computations and return results quickly.

1. **Usability:**

* The system should be easy to use and the UI should be intuitive. No user should have difficulties in operating the system itself.
* The system should run on multiple devices and be functional on multiple screen sizes to ensure a consistent user experience.

1. **Reliability:**

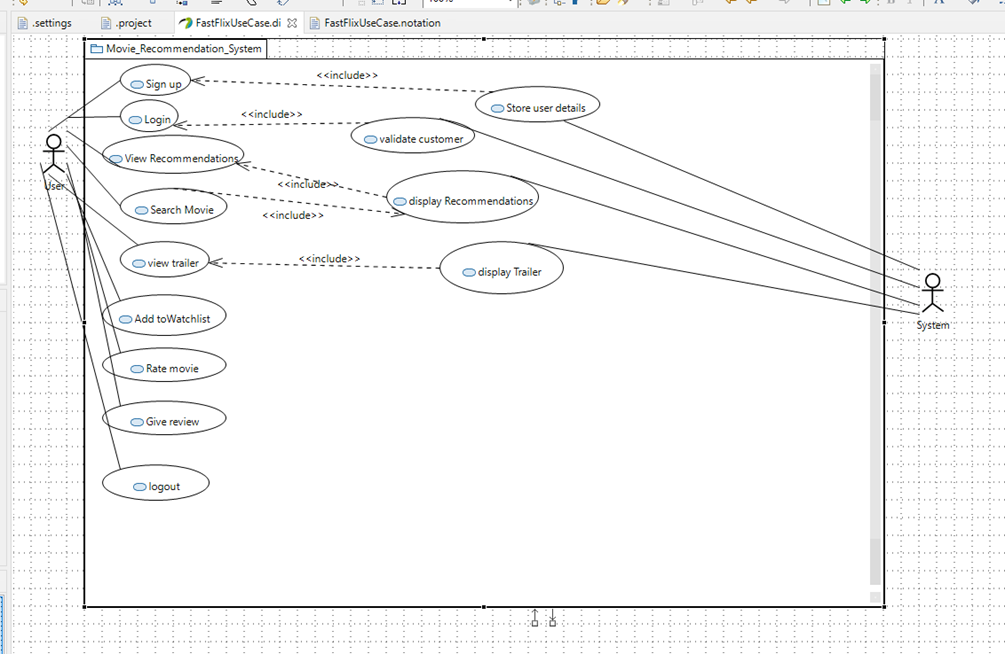
* The system should be reliable, users should not experience sudden loss of their watchlist, their recommendations or their credentials.
* Downtime should be minimal and fixed as soon as possible to prevent inconveniences.

*2.2 Design:*

This section outlines our use case diagram, sequence diagram and the frontend and backend design of our Movie Recommendation System.

*2.2.1 Use Case Diagram:*

The Use Case diagram depicts the functionality and actors of the Movie Recommendation System



**Fig 1. Use case diagram**

This diagram shows the functionality of the Movie Recommendation System. In this all the sub-functions describe each function's functionality:

● The use case diagram is named Movie Recommendations System.

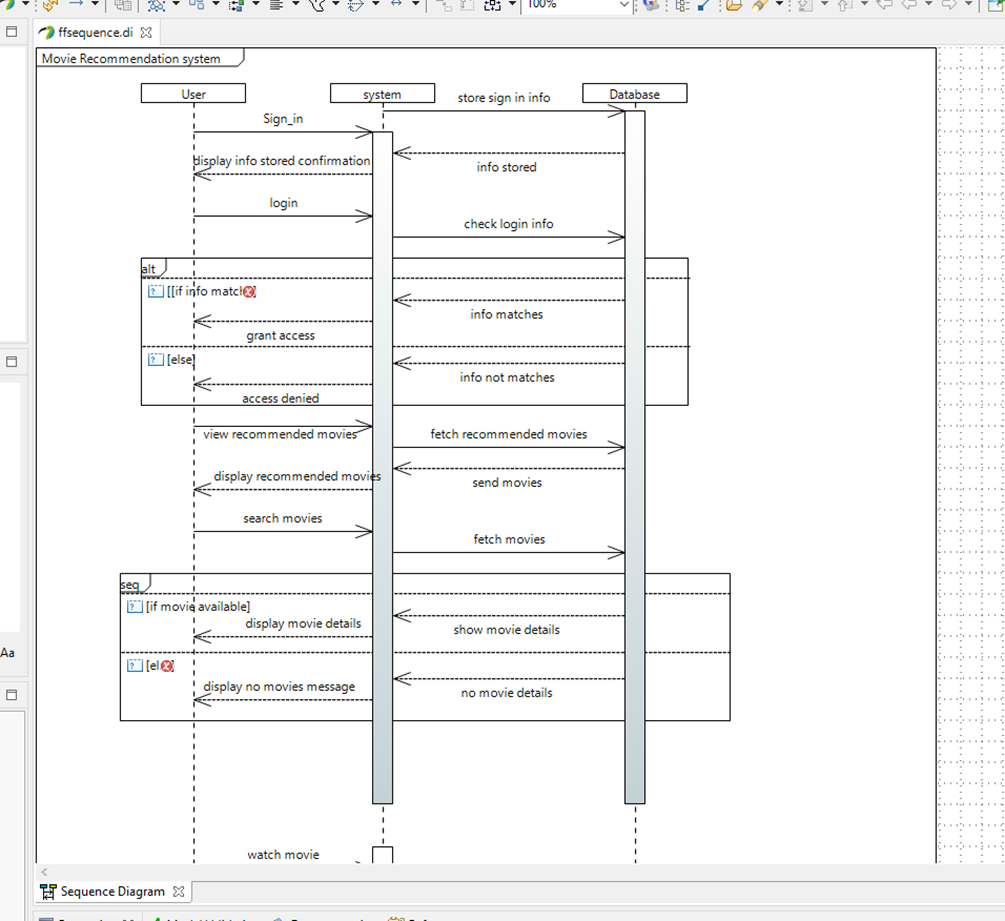
● It has 2 actors, of which “user” is the primary actor and admin and system is the secondary actor.

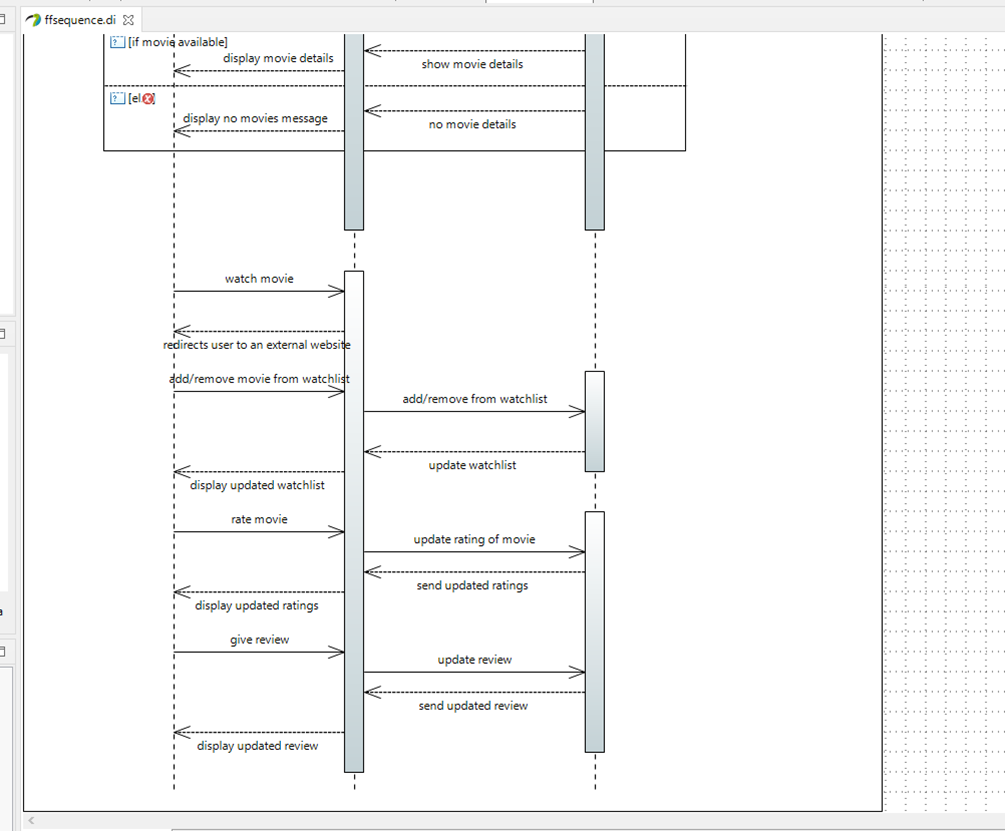
● The user can sign up, login, view recommendations, view trailers, add movies to the watchlist, rate the movies, and give reviews of the movies.

● The system stores the details of the user, validates the details when entered by the user during login. The system also analyzes the watchlist of the user to give personalized movie recommendations.

*2.2.2 Sequence Diagram:*

The sequence diagram depicts the entities that interact with each other. The diagram shows in detail the sequence of events occurring during the implementation of the Movie Recommendation System.

**

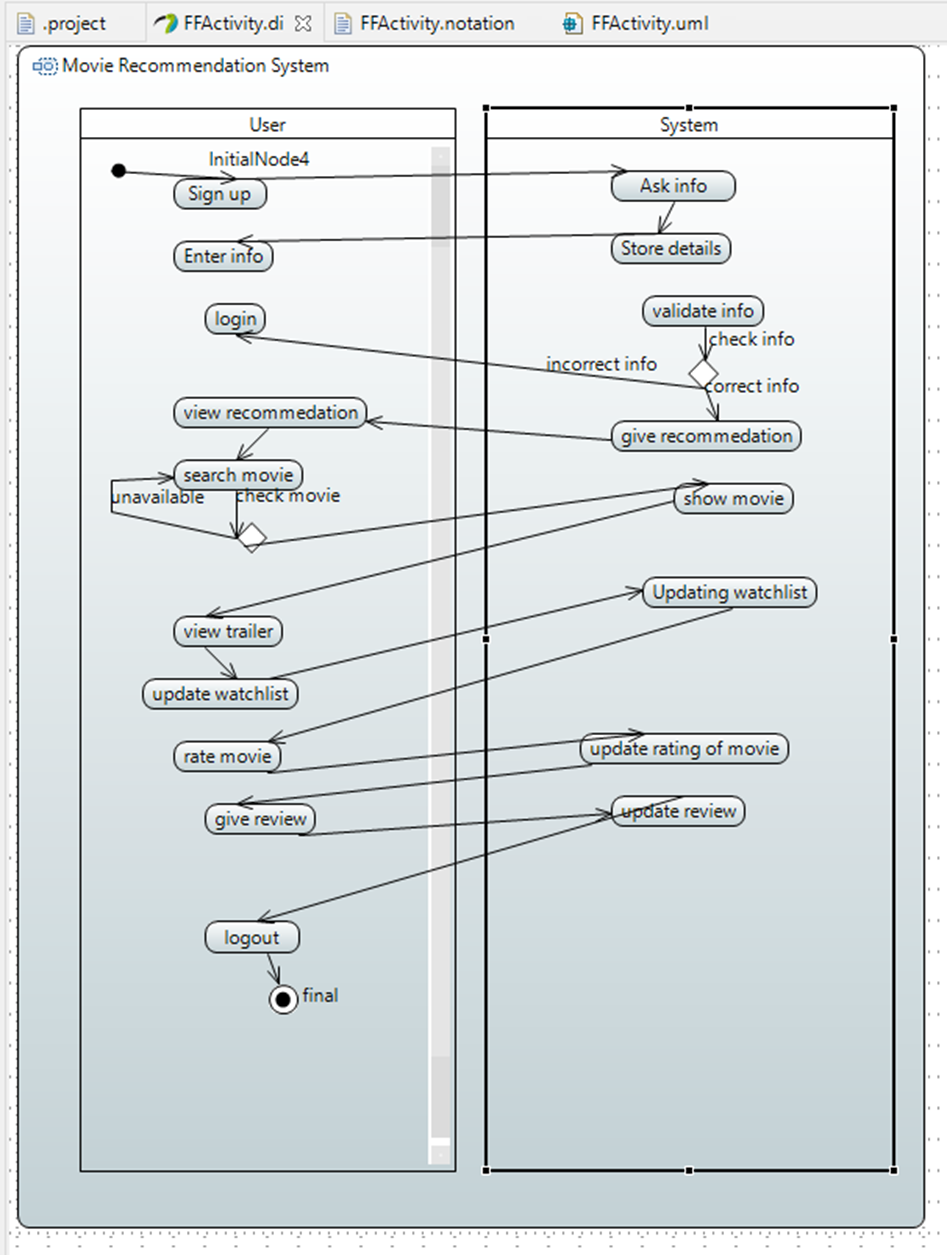
**

**Fig 2. Sequence Diagram**

The sequence diagram illustrates the flow of interactions within the Movie Recommendation System. It begins when the user logs in to the system, upon which their credentials are checked against stored credentials in the database. Once verified, the user is allowed access to the system, where they can browse movies, leave ratings and comments, watch movies and add movies to their watchlist. Moreover, upon visiting recommendations, the recommendation algorithm analyzes the user’s watchlist to provide personalized movie recommendations.

*2.2.3 Activity Diagram:*

An activity diagram provides a visual representation of the flow of activities within a system. The nodes represent activities and the arrows represent the flow of control between those activities.



**Fig 3. Activity diagram**

In the activity diagram for the Movie Recommendation System

● The "user” and "system" are the 2 activity partitions.

● The user engages in activities such as signing in, logging in, viewing recommendations, watching trailers, adding movies to the watchlist, rating movies, and providing reviews.

● The system, in turn, stores user details, validates login information, and generates personalized recommendations.

*2.2.4 Recommendation System:*

To implement the core functionality of our system, which are the “recommendations”, the Cosine Similarity Algorithm was chosen.

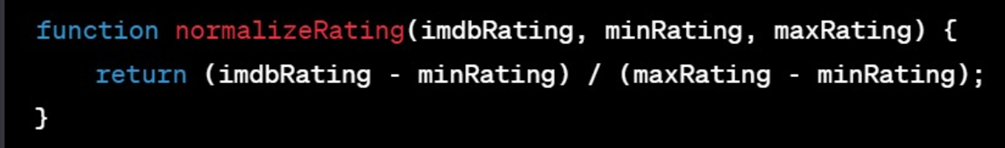
Cosine similarity was utilized to recommend movies based on the similarity between their feature vectors. The system employs binary genre vectors and normalized IMDB ratings to measure the similarity between items, enabling personalized recommendations for users.



**Fig 4. Cosine Similarity formula**

The Genre attribute will be turned into a binary vector, which sets the corresponding position to 1 if the movie belongs to that genre and 0 otherwise.

The IMDB Ratings of each movie will be normalized using the normalization formula.



**Fig 6. Normalization Formula**

*2.2.5 Database Design:*

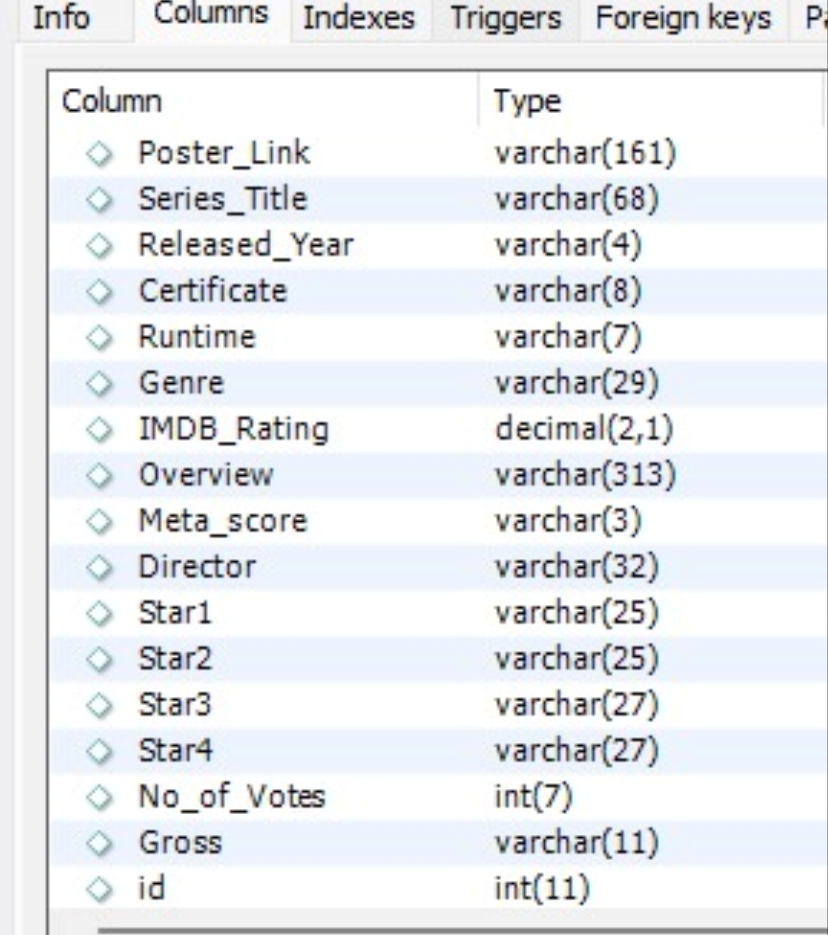
To store all the data of the Recommendation System, such as the vast library of movies, user credentials, ratings, reviews and scores, a Database is used.

A database has the following implicit properties:

* A database represents some aspect of the real world. Changes in the real world have to be reflected in the database. Thus, changes in the real world, such as a new user discovering our system, should be reflected in the database as well, such as a new tuple in our “Users” table.
* A database is a logically coherent collection of data with some inherent meaning. Movies are organized in the Movies table with fields of “Movie Name”, “Genre”, and so on.

To construct and manage databases, a **Database Management System (DBMS)** is utilized. Database Management System (DBMS) is a collection of programs that enables users to create and maintain a database. Defining a database involves specifying the data types, structures, and constraints of the data to be stored in the database.

Take the most important set of data, **Movies**. To define movies,a “Table” is created and multiple fields are made in that table, each corresponding to a specific attribute of a movie. Fields such as Title, Year Released, Director, IMDB Rating and Genre are included. Appropriate data types are assigned to each field. The Title field should only have words and numbers, thus it is assigned the “varchar” attribute, while the IMDB Rating should only have decimal or whole numbers, which is why it is assigned the “decimal” attribute.



**Fig 7. Database Design of Movies Table**

Our “Users”, “Reviews” and “Watchlist” are constructed in a similar manner.

Once a database is created, it can be **manipulated** to achieve specific functions, such as fetching details of movies, updating the tables when a new user joins the system and so on.

To access the database, an **application program** is used as it can request data from the database. A query typically causes some data to be retrieved; a transaction may cause some data to be read and some data to be written into the database.

Now that diagrams have been constructed of the planned design and an approach to database design has been thought up, the implementation phase can begin.

*2.3 Implementation:*

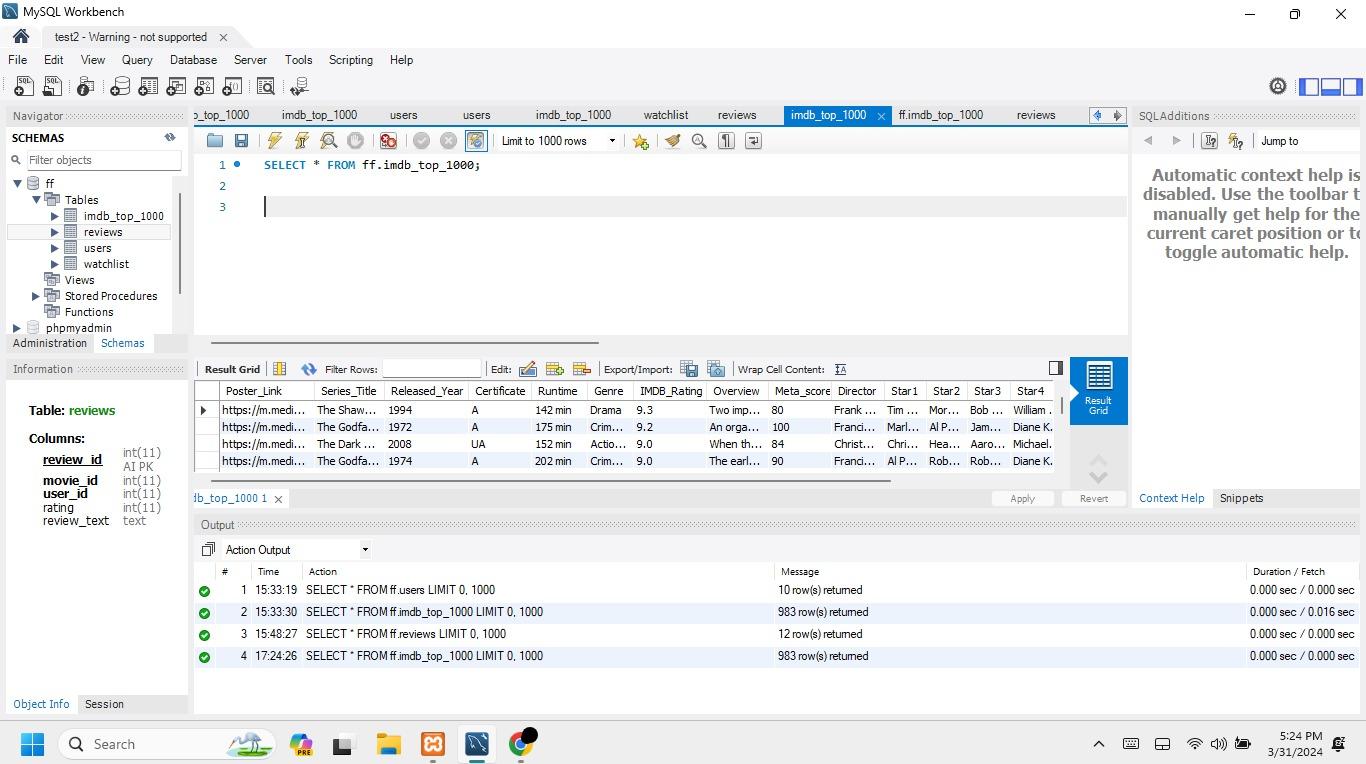
*2.3.1 Languages and technologies used:*

**SQL**:

To access and manipulate our databases, which contain Movies, Users, Reviews and Ratings, the SQL language is used. SQL stands for **Standard Query Language** and can execute queries against a database, write and retrieve data, update records, delete records, create new records and much more.

To make use of the full functionalities of SQL, a graphical user interface tool called **MySQL Workbench** is used to provide a helpful, visual interface in designing, modifying and developing databases.

In utilizing MySQL Workbench alongside the SQL query language, the databases can be efficiently designed, managed and maintained. The user-friendly interface provided by MySQL Workbench allows easy maintenance while the strong SQL language allows for easy updates and retrievals.



**Fig 8. Overview of MySQL Workbench**

****

****

**Fig 9. Sample SQL queries**

**NodeJS:**

JavaScript is considered to be one of the most famous scripting languages of all time. JavaScript, by definition, is a Scripting Language of the World Wide Web.

Node.js is an open-source, JavaScript runtime environment that allows developers to build scalable and high-performance network applications, like the **Movie Recommendation System.**

Its unique feature is the event-driven, non-blocking I/O model, which makes it particularly suitable for building real-time applications. It is also particularly suited to this system, where multiple users can operate it simultaneously. Node.js enables developers to use JavaScript for both client-side and server-side development, meaning it can be used for both frontend and backend. This feature was particularly useful in the design of the frontend, made using ReactJS.

In this project, NodeJS is used to build the backend. As NodeJS allows the import of existing modules for functionality, modules like MySQL (for databases), CORS (for network functionality) and Express (for building websites) are imported to further extend the functionality of the system.

Node.js, along with the mysql module, facilitates interaction with a MySQL database. SQL queries are executed to perform operations such as retrieving movie data, user authentication, registering users, managing watchlists, and storing user reviews.

NodeJS with Express is used to define various HTTP routes (e.g., GET, POST) that handle incoming requests from clients. Each website page, i.e movie/page is defined to handle specific functionalities like retrieving data from a database, user registration, login, and managing user reviews.

As mentioned above, NodeJS allows Javascript to be used in both frontend and backend development, leading to the frontend being developed in ReactJS.



**Fig 10. NodeJS GET and POST sample**

**ReactJS:**

ReactJS, often referred to simply as React, is an open-source JavaScript library used for building user interfaces (UIs) for web applications. React allows developers to create interactive and dynamic UI components that efficiently update in response to changes in data. This is why it is excellent for building responsive elements of our frontend.

React.js is a client-side library, meaning it runs in the user's browser and is primarily responsible for rendering UI components.

ReactJS is used to specify how our system will look to the average user. Each page on the website is broken down into modules and each module has been made in ReactJS.

ReactJS breaks down complex UIs into modules and those modules are built using HTML, CSS and Javascript, with a few added React functionalities.

* **HTML:** HTML is a hypertext markup language which is in reality the backbone of any website. HTML defines the structure and semantics of the webpage in the form of HTML elements. HTML elements represent different types of content such as text, images, links, forms, and multimedia.
* **CSS:** As HTML creates the structure and semantics of a webpage, CSS (Cascading Style Sheet) is responsible for styling those elements. They format the layout of Web pages and give them a visual flair, whether it be the size of text or the color of a background.
* **JavaScript:** The main implementation of Javascript is to implement various web functionalities, such as browser detections, cookie creation and so on.For frontend development, Javascript’s lightweightedness allows it to be integrated directly into the HTML code and specify what each HTML element does.



**Fig 11. Sample ReactJS showing all possible pages in the website**

Combining everything together, namely MySQL Workbench to visually manage databases, NodeJS for backend development and ReactJS for frontend development, coupled with the Cosine Similarity algorithm for movie recommendations, our Movie Recommendation System has been implemented.

All of these technologies are used together to build our Recommendation System. The databases are created, maintained and manipulated using the SQL query language with MySQL Workbench providing a helpful visual interface for administrators.

NodeJS utilizes JavaScript and provides a robust backend to facilitate interaction with MySQL Workbench and the databases hosted within.

ReactJS makes use of NodeJS, HTML and CSS to facilitate construction of a responsive frontend that allows users to browse their favorite movies and add them to their watchlist. Coupled with the Cosine Similarity Algorithm, personalized movie recommendations are provided to the user as well.