Software Requirements Specification(SRS)

FastFlix

**Movie Recommendation System**

**Version 1.0**

Prepared by**: Bilal Ali, Ehaab Tariq,Abdur Rehman**

Instructed by**: Dr. Syed Muazzam Ali Shah**

**Date:December 3,2023**

**Table of Contents**

**Table of Contents ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. External Interface Requirements 3**

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

**4. System Features 4**

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

**5. Other Nonfunctional Requirements 4**

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

**6. System Architecture**

6.1 Use Case Diagram

6.2 Class Diagram

6.3 SequenceDiagram

6.4 Communication Diagram

6.5 Activity Diagram

6.6 Deployment Diagram

6.7 Component Diagram

**7. Other Requirements**

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 5**

**Appendix C: To Be Determined List 6**

# Introduction

A movie recommendation system has become an indispensable systems component in various entertainment applications. So, welcome to the Software Requirements Specification (SRS) for "FastFlix," an innovative movie recommendation application designed to transform the way users explore, discover, and enjoy cinematic experiences. This comprehensive document serves as a blueprint for the development team, stakeholders, and any interested parties involved in the creation of this cutting-edge movie recommendation system.

## Purpose

Our goal is to create a one-stop shop for everything to do with movies, from their name and details to their rating and full-text reviews by various users. Users will be able to view extensive details about the film according to various criteria such as genre, director, release date and much more. Users will also be able to rate movies, review them and see other users’ scores as well. They will be able to create their own custom profiles and categorize their favorite movies in collections of their choosing. Thus, our website will be a great way for anyone to find movies based on large criteria and review those movies themselves.

## Document Conventions

Heading Font: Times

Heading style: Bold

Heading size: 18

Sub-Heading Font: Times

Sub-Heading style: Bold

Sub-Heading size: 14

Rest of the text Font: Times

Rest of the text style: Simple

Rest of the text size: 12

## Intended Audience and Reading Suggestions

This Software Requirements Specification (SRS) is crafted to cater to a diverse audience, including developers, project managers, marketing staff, users, testers, and documentation writers. Developers will find detailed technical specifications and system architecture information essential for implementation. Project managers can utilize the document for project planning and resource allocation. Marketing staff will gain insights into the app's features for effective promotion. Users will discover details on how the app enhances their movie-watching experience. Testers will find information crucial for creating comprehensive test plans. Documentation writers will benefit from understanding the system's functionality for crafting user manuals.

## Product Scope

We are planning to use a dataset of 1000 movies. These movies will be categorized in numerous ways, allowing users to filter movies that suit their precise preferences. Users will also be able to add their own entries, rate them and review them while admins will be able to update details about existing movies in the database. We are planning to use the React JS, Express JS and Node JS to handle our databases.

## References

*Title : IMDB, Date : 3-December-2023,*

*Author : Col Nedhaam, Source :* [*https://m.imdb.com/*](https://m.imdb.com/)

*Video followed (channel: Lama Dev):* [*https://youtu.be/fPuLnzSjPLE?si=tJmmpIUWM9i9MACB*](https://youtu.be/fPuLnzSjPLE?si=tJmmpIUWM9i9MACB)

*Another video followed(channel: PedroTech):* [*https://www.youtube.com/watch?v=W-sZo6Gtx\_E*](https://www.youtube.com/watch?v=W-sZo6Gtx_E)

*<https://www.researchgate.net/figure/Flowchart-of-the-methods-used-for-movie-recommendation_fig2_299593674>*

*<https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/>*

*<https://data-flair.training/blogs/data-science-r-movie-recommendation/>*

# Overall Description

## Product Perspective

This FastFlix Movie Recommendation System is an extension of an existing family of movie recommendation products. It is also designed to enhance and build upon the functionality and practicality of its predecessors, providing users with an improved, easy to use and user-friendly experience. As a follow-on member, it inherits the core principles and features of the established movie recommendation system while introducing new elements and features to increase the user's interaction. This system operates as a self-contained entity with its unique user interface, allowing individuals to seamlessly explore recommended movies based on their preferences. While part of a larger family, it stands independently, offering a fresh and refined approach to movie recommendations. The system's components are intricately connected to ensure smooth interactions, and external interfaces are optimized for user convenience. The product perspective is represented in the overall context of an evolving movie recommendation ecosystem.

## Product Functions

The major functions are:

* The user has to login the system, this indicates User authentication and authorization.
* The user's data storage and retrieval is secure.
* User to user communication in form of reviews.
* The user gets the movie recommendations based on which movies are popular at the time and are highly rated.
* Secondly, the app also recommends the movies based on the movie genres that the user prefers.
* Data consistency is ensured.

## User Classes and Characteristics

In our FastFlix Movie Recommendation System, we have a primary user class: Regular Users. Regular Users are individuals seeking an engaging movie-watching experience. They vary in technical expertise, educational backgrounds, and movie preferences. Regular Users will interact frequently with the system, utilizing features such as exploring recommendations based on their preferred genres, maintaining watchlists, and receiving personal review suggestions.Regular Users form the core user base Distinct requirements cater to the user class, ensuring a tailored experience for each user

## Operating Environment

Operating environment for the movie recommendation system is as listed below.

* Distributed database
* client/server system
* Operating system: Windows.
* database: Mysql database
* Platform: Chrome

## Design and Implementation Constraints

* Hardware limitations would include having a computer/laptop with more than 4 GB RAM and internal storage to be more than 500 GB( if you plan to put movies as well).

The system's performance may be subject to hardware limitations, especially on older devices with limited processing power and memory.

* If space is unavailable, our software links to an external website as well
* SQL commands for queries/applications are connected to backend code via Xampp and DataBase uploaded on Mysql workbench,

The system relies on MySQL as the primary database management system

* Node JS for the implementation of movie recommendation system Backend.
* HTML, CSS, JavaScript, ReactJS for the development of Ul.

## User Documentation

*An understanding of how normal websites work is enough for every user to operate our software.*

## Assumptions and Dependencies

Assumptions could be that as mentioned above that if someone plans to use the software and implement it on external online APIs then their code fetching the online APIs would be different than when data is fetched from SQL database.

# External Interface Requirements

## User Interfaces

FastFlix caters to both user roles: regular users with access to a diverse movie collection and the ability to like and review films, and administrators with the authority to modify or delete movie details. The user interface includes a main homepage and specialized pages for Most Popular, Newest, and Recommended movies, featuring a clean and intuitive design with a card-based layout and a hover-over effect for movie descriptions. The system is designed to be responsive to various screen sizes, and standard buttons for liking and reviewing movies are available. Navigation is intuitive, and error messages leverage existing software features for troubleshooting. Help functionality includes tooltips for users and comprehensive documentation, while accessibility considerations prioritize adherence to web accessibility standards for inclusivity. User authentication is seamlessly integrated, ensuring secure access to user-specific functionalities and personalization.

## Hardware Interfaces

FastFlix establishes seamless connections with its hardware components to orchestrate efficient data management and user interactions. Employing React as the frontend powerhouse, the system adeptly interfaces with the foundational database, acting as the robust repository for detailed movie information. Capitalizing on React's renowned optimization capabilities, the frontend harmoniously integrates with hardware interfaces. These interfaces play a pivotal role in executing error handling, logging, and ensuring the fluidity of data storage and retrieval operations. This harmonious integration results in a highly responsive and dependable user experience, where the hardware layer contributes significantly to the overall system performance enhancement. User authentication mechanisms are seamlessly incorporated, enhancing security and ensuring that access control is maintained at the hardware level.

## Software Interfaces

FastFlix is built on a tech stack comprising React for the frontend, Node.js for the backend, and a MySQL database. The React frontend, leveraging libraries such as Axios for communication and React Router DOM for page navigation, enables users to browse, like, and review movies. The Node.js backend processes user requests, communicates with the MySQL database using the MySQL library, and integrates tools like Cors for cross-origin resource sharing and Nodemon for efficient development. User authentication is seamlessly integrated into the Node.js backend to ensure that user-specific functionalities are securely managed. External APIs enrich the database, and the system operates on the Windows OS. Global data sharing within the backend ensures consistency, and detailed API protocols are documented separately to guide developers in understanding data exchange and authentication procedures.

## Communications Interfaces

FastFlix seamlessly integrates various communication interfaces to deliver a responsive and secure user experience. User interactions, facilitated through web browsers, leverage HTTP/HTTPS protocols for efficient communication with the React frontend. The Axios library orchestrates smooth data exchange between the frontend and the Node.js backend, employing standardized JSON formatting. The backend communicates with the MySQL database using SQL queries, ensuring reliable data storage and retrieval. External APIs, accessed via HTTP/HTTPS, enrich the movie database. Security is paramount, with HTTPS employed for encrypted data exchange, safeguarding user information. The system optimizes data transfer rates for responsiveness, and synchronization mechanisms within the backend maintain consistency during concurrent user interactions and database updates. User authentication is seamlessly integrated into the communication process, providing secure access to user-specific data and functionalities. Embracing established communication standards, such as HTTP/HTTPS and SQL, ensures compatibility across diverse technologies and platforms.

# System Features

## Feature 1: Browse Movie Categories

## 4.1.1 Description and Priority:

## Allow users to explore movie categories such as Most Popular, Newest, and Recommended. This is of High priority, providing diverse options for users to discover movies.

## 4.1.2 Stimulus/Response Sequences:

## User clicks on the "Most Popular" feature.

## System responds by loading a new page displaying a list of the most popular movies.

## 4.1.3 Functional Requirements:

## REQ-1: Implement a navigation system for users to select categories.

## REQ-2: Load a new page with a list of movies corresponding to the selected category.

## REQ-3: Ensure the system dynamically updates the movie list as categories change.

* REQ-4: Authenticate User: Ensure that user authentication is enforced to allow access to personalized features based on user roles.

## Feature 2: Movie Details Page

## 4.2.1 Description and Priority:

## Provide a detailed page for each movie, including information like title, genre, release date, and a review button. This is of Medium priority, enhancing user engagement with individual movies.

## 4.2.2 Stimulus/Response Sequences:

## User clicks on a specific movie from the list.

## System responds by reloading the page with detailed information about the selected movie.

## 4.2.3 Functional Requirements:

## Data Loading: Large data sets should load in less than 5 seconds to avoid delays.Response Time: The system should respond to user actions within 2 seconds to ensure a smooth experience.

## Data Loading: Large data sets should load in less than 5 seconds to avoid delays.REQ-4: Display comprehensive movie details on the dedicated page.

## REQ-5: Include features such as title, genre, release date, and a button to submit user reviews.

## REQ-6: Ensure the page reloads efficiently to present information for different movies.

* **REQ-7**: Authenticate User: Ensure that user authentication is enforced to allow access to personalized features, such as submitting reviews.

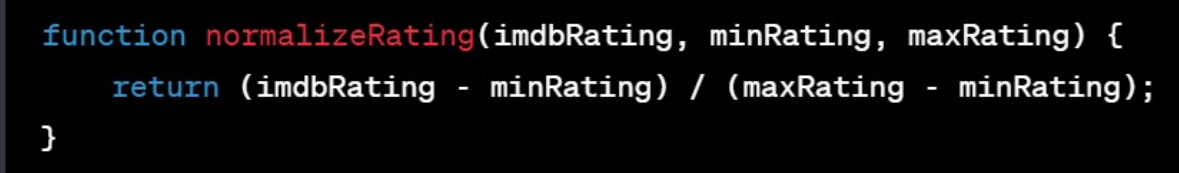
### Feature 3: Recommendation using Cosine Similarity

**4.3.1 Description and Priority:**

Utilize cosine similarity 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.



We have made the Genre attribute into a binary vector, which sets the corresponding position to 1 if the movie belongs to that genre and 0 otherwise and that we have normalized IMDB Ratings using the normalization formula.



**4.3.2 Stimulus/Response Sequences:**

* User interacts with the "Recommendations" feature.
* System responds by providing personalized movie recommendations based on the user's historical preferences and similarity metrics.

**4.3.3 Functional Requirements:**

* REQ-8: Implement a recommendation system using cosine similarity for personalized movie suggestions.
* REQ-9: Convert movie genres into binary vectors, setting positions to 1 if the movie belongs to that genre and 0 otherwise.
* REQ-10: Normalize IMDB ratings using the normalization formula to maintain consistency in rating scales.
* REQ-11: Calculate cosine similarity between movies using their feature vectors.
* REQ-12: Provide user-specific recommendations based on their historical preferences and similarity metrics.

**Performance Requirements:**

Data Loading: Large data sets should load in less than 5 seconds to avoid delays.

* By adding the authentication requirements (REQ-4 for Feature 1 and REQ-7 for Feature 2), we ensure that user authentication is an integral part of accessing personalized features and maintaining user security.

# 

# Other Nonfunctional Requirements

## Performance Requirements

* Response Time: The system should respond to user actions within 2 seconds to ensure a smooth experience.
* Data Loading: Large data sets should load in less than 5 seconds to avoid delays.

as for the their rationales:

* User Experience: Quick response times enhance user satisfaction.
* Efficiency: Faster data loading ensures users can work efficiently.
* Scalability: Supporting a high number of concurrent users caters to potential usage spikes.

for timing-relationships

* Real-Time Systems: In real-time scenarios, ensure critical actions occur within specific time frames for effective operation.

## Safety

We can set a password to our database to safely secure usernames, passwords, watchlists and other information. Moreover, we can encrypt our passwords to add an additional layer of security.

## Security Requirements

Database should be reached securely and the data should not be lost or inconsistent. It also should not change except if the administration updates it. Moreover, since our data-set contain some information about the user such as its login id and password, its watchlist, preferences, for these reasons security design is important in the web service, so one must store the database carefully or choose their database partner carefully.

## Software Quality Attributes

1.Availability: The site should be available on the specified date and specified time as many users are doing their search.

2.Adaptability:

* Requirement: The system should easily adapt to changes in user needs or business requirements.
* Rationale: This ensures the software remains useful even as things evolve.

3.Correctness: The site will give proper recommendations based on one's view.

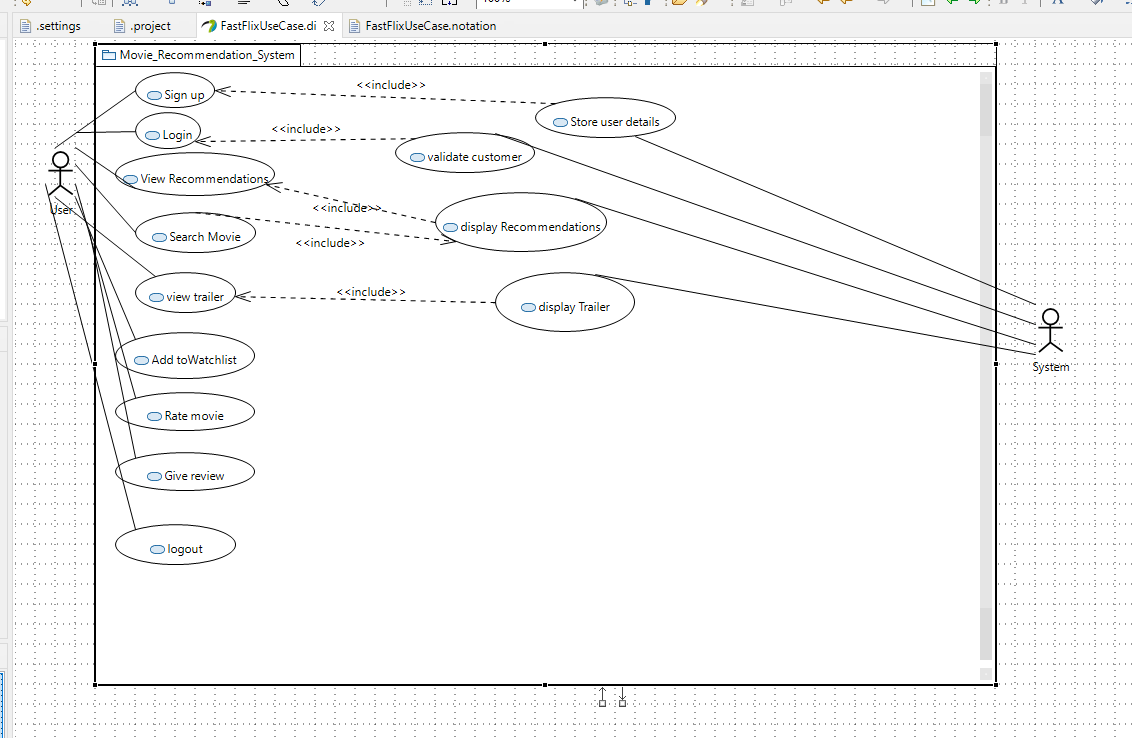
4.Maintainability: The administrators and back-end developers should maintain correct output of recommendations.

5.Usability: The site should satisfy a maximum number of users' needs.

# System Architecture

Here the whole architecture of FastFlix is shown in the form use case diagram, class diagram, activity diagram, sequence diagram, communication diagram and deployment diagram which would give a mind map of the movie recommendation app and enhance the ideas of developer to make further improvements using these diagrams

## 6.1 Use Case Diagram:

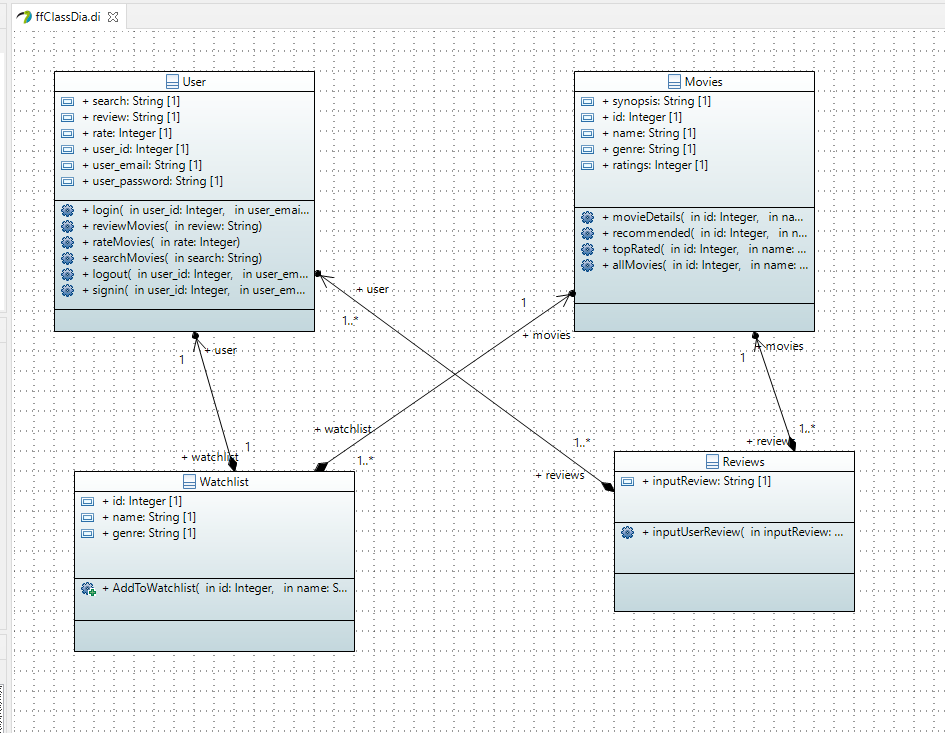


## 

This diagrams shows the functionality of FastFlix movie recommendations system. In this all the sub-functions describes each functions functionality.

* The use case diagram is named movie recommendations system.
* It it has 2 actors, of which “user” is the primary actor and admin and system is the secondary actor.
* The user can sign in, 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, validate the details when entered by user during login and gives recommendations.

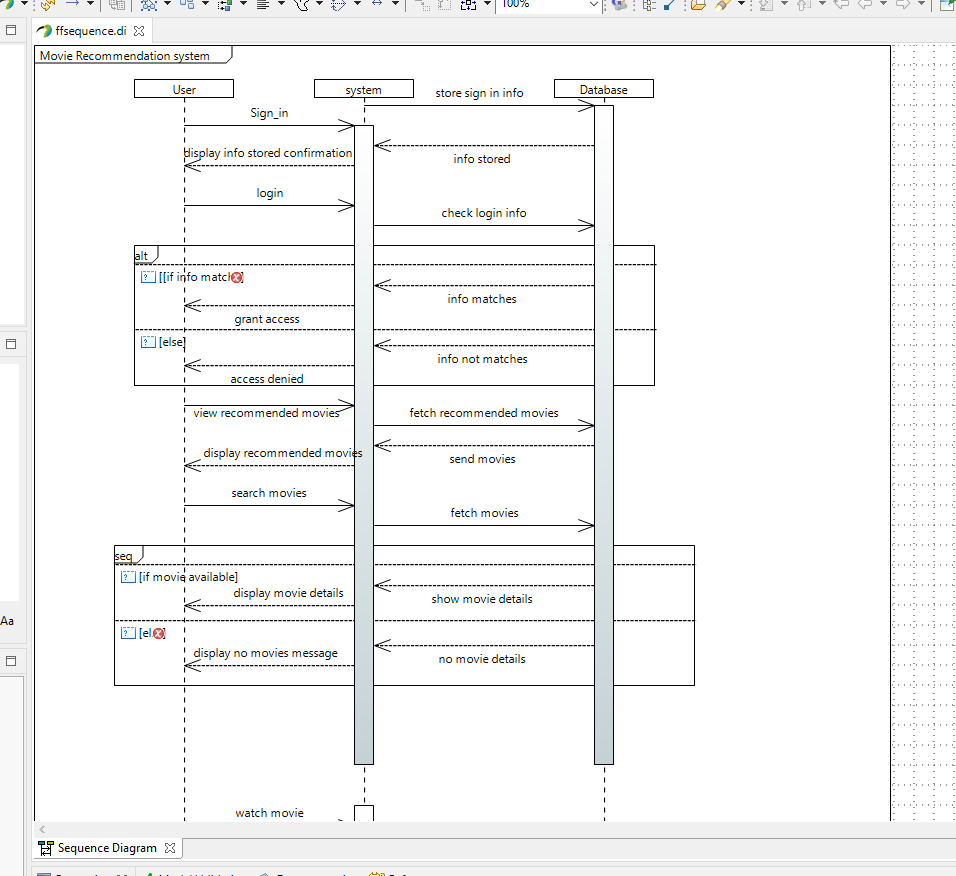
## 6.2 Class Diagram:

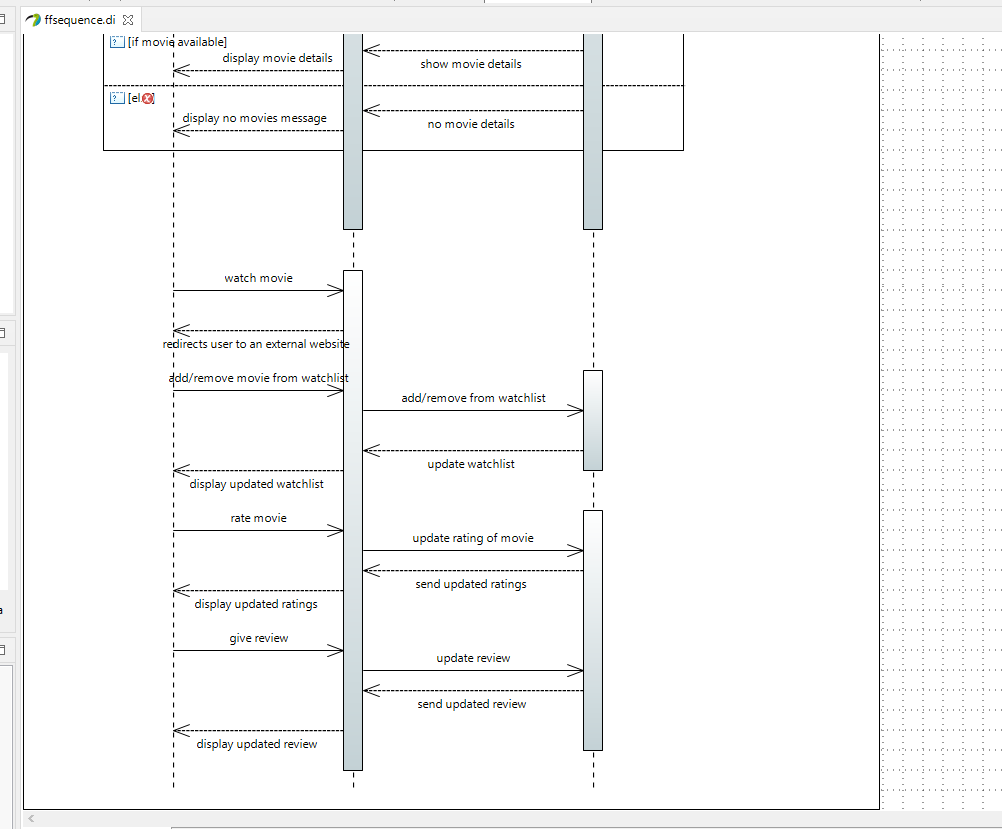


The class diagram shows the detailed system design for the system.

* Here there are 4 classes i.e. user, watchlist, movies and reviews..
* The user class also has its id, email, password, its review function, search function and rate function
* Watchlist contains id, genre and name of movies.
* Review class contains the input review and delete review.
* The movie class contains movies id, name ,genre,ratings and synopsis.

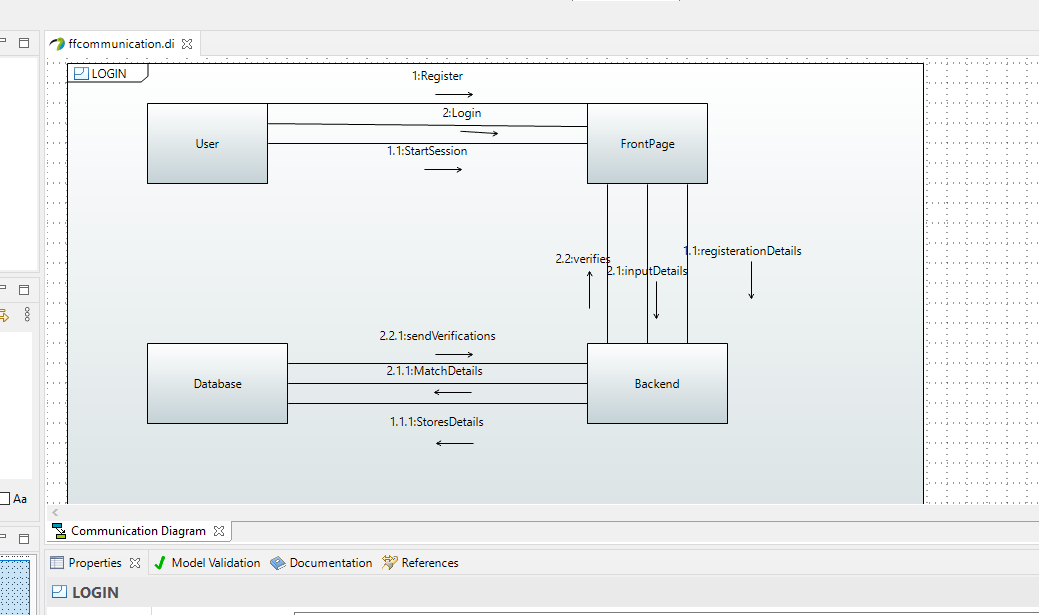
## 6.3 Sequence Diagram:



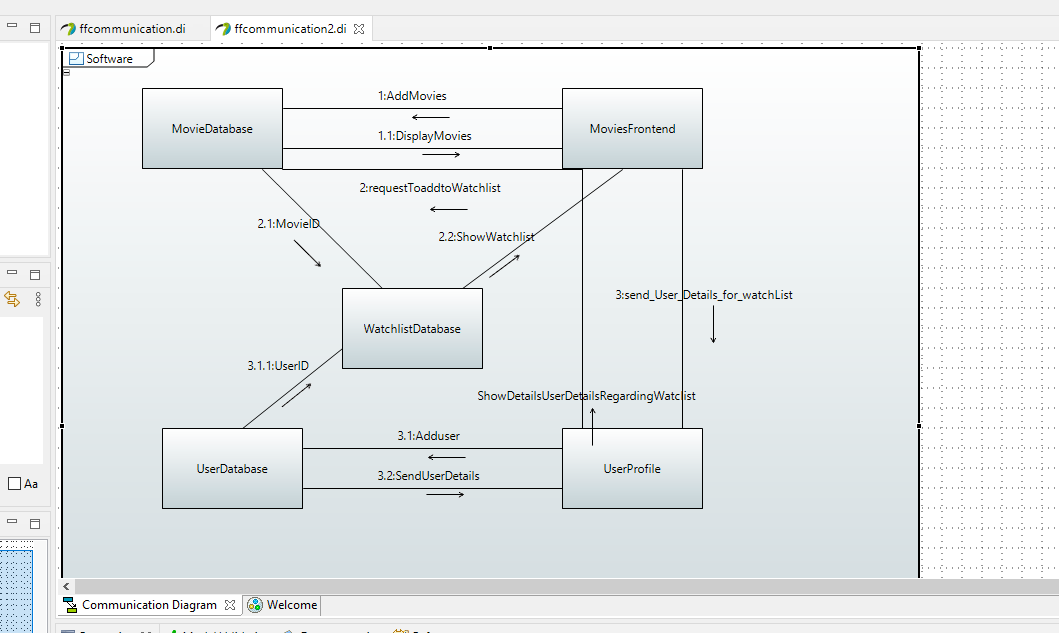


## 6.4 Communication Diagram:

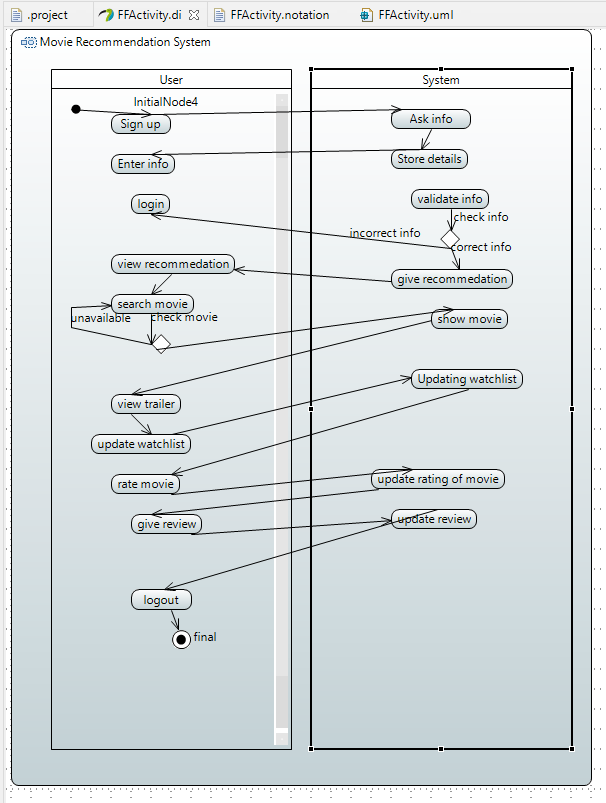
For LOGIN:



For Software:



## 6.5 Activity Diagram:

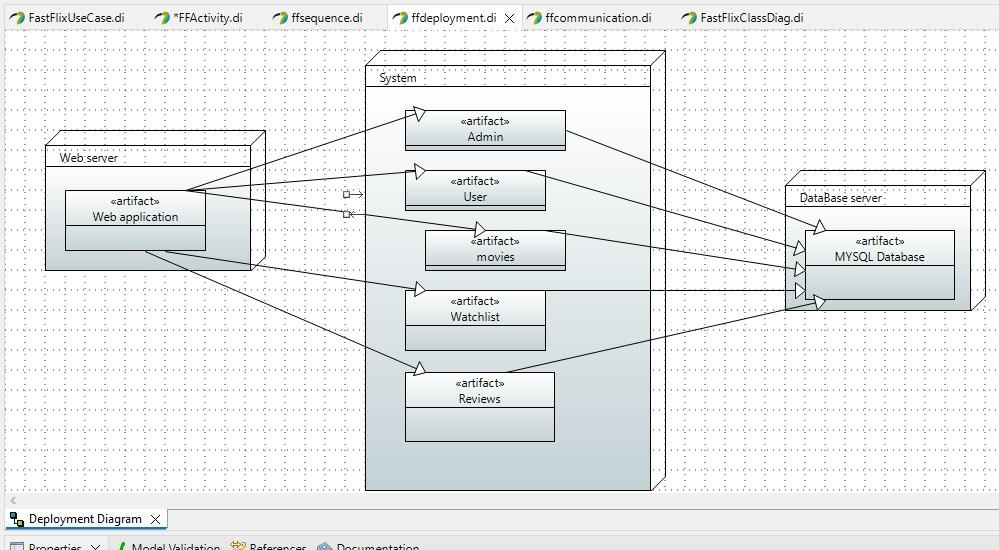


In the activity diagram for the FastFlix movie recommendation system

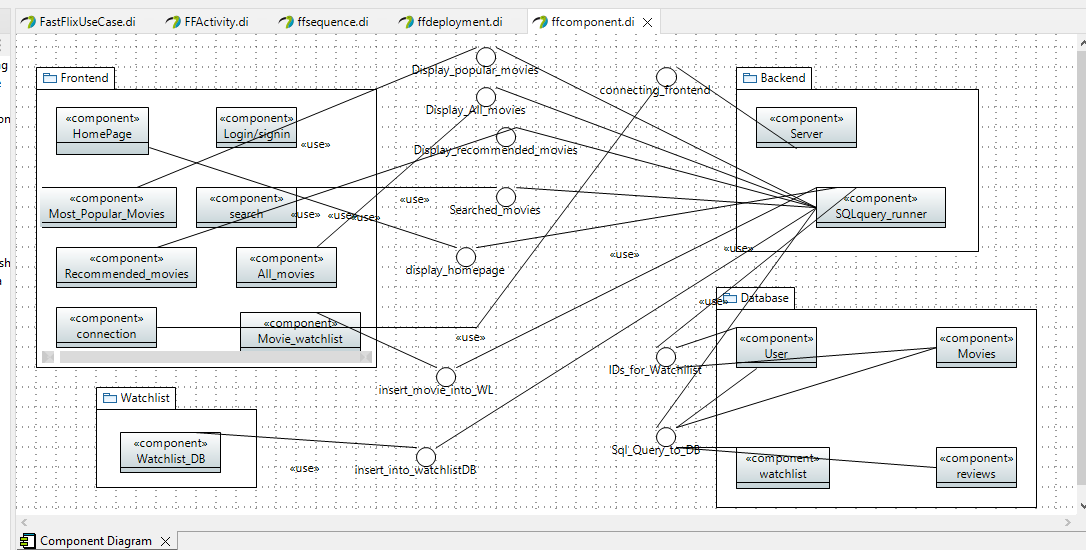
* The "user” and "system" are the 2 activity partitions.
* The user engages in activities such as signing in, logging in, viewing recommendations, watching trailers, adding movies to the watchlist, rating movies, and providing reviews.
* The system, in turn, stores user details, validates login information, and generates personalized recommendations. .

This diagram visually represents the dynamic flow of actions within the FastFlix movie recommendation system, illustrating how users interact with the system, and how administrators manage content based on user feedback.

## 6.6 Deployment Diagram:



## 6.7 Component Diagram:

****

**Appendix A: Glossary**

*All terms are self-explanatory*