



Università degli Studi di Messina

"Università degli Studi di Messina"

Department of Mathematical, Computer, Physical, and Earth Sciences
Bachelor's Degree Program in Data Analysis.

Software Engineering Project:

“Movie Recommendation system “

Developed by
Tharun Reddy Banda

Guided by: Professor Distefano Salvatore

I. Overview.

This is a project report for Software Engineering at the University of Messina taught by Professor Distefano Salvatore. The purpose of the project is to apply a process to create a complete software system product.

1.1 Project idea

The idea of Movie Recommendation System, came from research to develop a content-based movie recommendation system capable of providing personalized movie suggestions to users based on their past interactions and content preferences. Unlike traditional recommendation systems, which rely on user behaviour and collaborative filtering, our approach utilizes the inherent characteristics of movies, such as genres, directors, actors, and plot summaries, to make tailored recommendations.

II. Software Process

1. Chosen process: Scrum

Scrum is one of the most popular frameworks for Agile implementation. Scrum focuses on managing iterative development rather than specific Agile practices.

2. Why Scrum?

Scrum, an Agile framework, excels in managing complex projects with swiftly changing requirements, a typical scenario in software development. For this project, where requirements are likely to shift based on ongoing research into recommendation systems, Scrum or Agile methodologies are particularly suitable. Scrum's iterative nature and frequent feedback loops can significantly enhance the alignment of the final product with stakeholders' needs.

Given that this project involves only one participant, traditional Agile methodologies may fall short as they depend on team members sharing all roles and responsibilities. Scrum, however, provides a structured division of roles into three key positions: Product Owner, Scrum Master, and Development Team. This clear separation allows for a more accurate simulation of a team environment, helping to understand the distinct challenges and duties associated with each role.

Furthermore, Scrum ensures that the project remains on track by facilitating early detection and resolution of potential issues. Its emphasis on regular communication and collaboration helps keep all team members—or the single participant—informed about any emerging problems and fosters a proactive approach to solving them.

Additionally, Scrum promotes continuous improvement through its sprint reviews and retrospectives. These practices not only ensure that the project is progressing as planned but also provide opportunities to refine processes and enhance the quality of the work being done.

Overall, choosing Scrum can help deliver a product that meets stakeholders' expectations, is completed on schedule, and upholds high standards of quality. The structured roles and processes inherent in Scrum provide a robust framework for managing complex projects effectively, even when managed by a single individual.

3. Three phases in Scrum:

Initial Phase: This phase focuses on setting the project's overall goals and designing the software architecture. It includes several essential steps such as creating the project vision, identifying stakeholders, appointing the Scrum Master and Product Owner, forming the Scrum team, creating a prioritized product backlog, and conducting release planning.

Sprint Cycle Phase: This phase consists of a series of iterative sprint cycles, with each cycle working towards developing an increment of the system. Activities in this phase include creating deliverables, conducting daily stand-up meetings, grooming the prioritized product backlog, and delivering the increment.

Project Closure Phase: This final phase is dedicated to wrapping up the project, completing necessary documentation like system help frames and user manuals, and reflecting on the lessons learned throughout the project. Key activities include shipping the deliverables and conducting a project retrospective.

These phases collectively ensure that the Scrum framework is effectively implemented, guiding the project from inception to completion while continuously delivering value.

4. Three main roles in Scrum

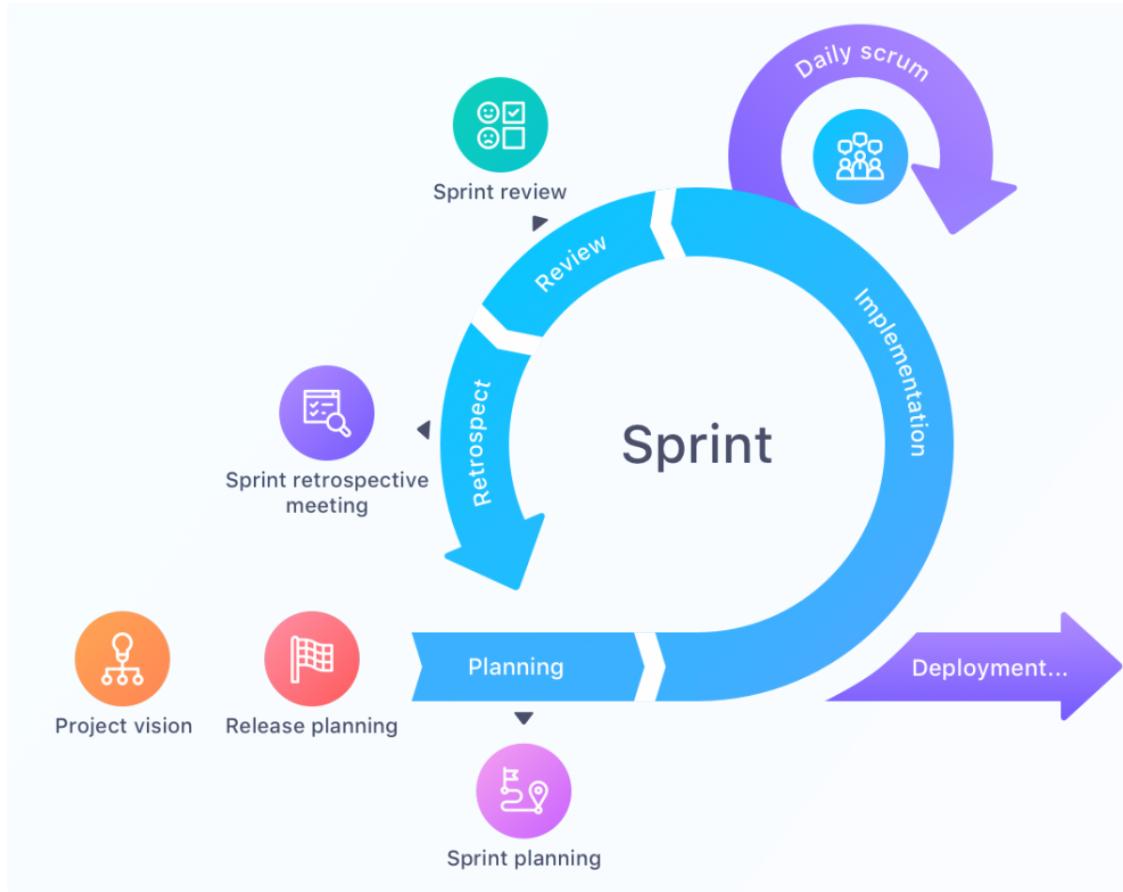
Product Owner: The Product Owner's main goal is to maximize the value of the product created by the Development Team. This involves managing the Product Backlog, conducting analyses, and maintaining a business-oriented focus. They also handle external interactions and act as the interface between stakeholders and the team. Their role ensures that the team is focused on building the right product.

Scrum Master: The Scrum Master is responsible for making sure that Scrum practices are understood and followed by all team members and stakeholders. Serving as a servant leader, the Scrum Master also acts as an advisor, coach, mentor, and facilitator. Their primary objective is to ensure that the team works efficiently and quickly, adhering to Scrum principles.

- Development Team: The Development Team is tasked with delivering a potentially releasable increment of the product at the end of each Sprint. They operate with a high degree of self-management and self-organization, ensuring they build the product correctly. The team works collaboratively to meet Sprint goals and maintain a high standard of quality.

These roles collectively ensure that the Scrum framework is implemented effectively, promoting efficient, high-quality product development that meets stakeholder needs.

5. Sprint cycle



Sprints are fixed durations, typically lasting 2-4 weeks. In this project, each sprint spans 5 working days. Generally, each sprint consists of four main phases:

Planning: Before each sprint begins, the team selects the tasks it commits to completing by the sprint's end, starting from the top of the product backlog. During the sprint, these commitments and the sprint end-date remain unchanged. If a significant issue arises, the Product Owner can instruct the team to terminate the sprint early and initiate a new one.

Implementation – Daily Scrum Meeting: Each day, the team holds a brief meeting to update one another on progress and identify any obstacles. The team stands up to keep the meeting brief, typically under 15 minutes. During this meeting, each member reports on three key points: what was completed since yesterday, what will be done by tomorrow, and any blocks encountered. Utilizing a Burndown Chart is crucial for the team's self-management.

Review: At the sprint's conclusion, the Product Owner, Team, Scrum Master, and Stakeholders gather to review and demo the completed work. The Product Owner collects feedback on how to enhance what has been built. This session also includes a review of the

timeline, budget, potential capabilities, and marketplace conditions for the next anticipated releases of functionality.

Retrospective: The Team, Product Owner, and Scrum Master meet at the end of each sprint to reflect on their working process and improve effectiveness. Typically, this meeting lasts around 3 hours for a one-month sprint. This retrospective is an opportunity for the Scrum Team to discuss what went well, what could be improved, and what specific actions will be taken to enhance the next sprint. All team members are encouraged to participate fully in these discussions.

These phases ensure that each sprint is productive and that the team continuously improves their workflow and product quality.

III. Initial phase :

1. Project members

- Product Owner: Tharun Reddy Banda
- SCRUM Master: Tharun Reddy Banda
- Member in Team: Tharun Reddy Banda

Taking all three Scrum roles by oneself is generally not advised. Scrum, like other software development processes, is designed to address issues related to independent interpersonal collaboration. However, due to the challenge of finding a suitable partner at the right time, I am undertaking this project solo. Despite this, I will strive to perform each role independently to gain a comprehensive understanding of their contributions to a project.

While it is unconventional to combine the roles of Product Owner, Scrum Master, and Development Team into one person, this approach will offer valuable insights into the specific responsibilities and impacts of each role. Although Scrum and other software methodologies emphasize the importance of teamwork and role separation for optimal functionality, this solo will provide a unique perspective on the inner workings and interdependencies of these roles. This experience will enhance my appreciation for the distinct contributions each role makes in a typical Scrum team, despite the inherent challenges of managing all roles single-handedly.

2. Project ideal

The project's goal is to create a website that suggests movies to users based on their preferences. Users can rate the movies they've watched on a 5-point scale, which helps the system learn their tastes. A machine learning algorithm will analyse this data to provide personalized movie recommendations. The more ratings users provide, the more precise the movie suggestions will become.

3. User Story

To ensure a robust system architecture, we propose dividing the system into three distinct components: Database, Backend, and Frontend.

The Backend component will establish a connection with the Database using a designated driver library. Additionally, communication between Backend and Frontend will occur through HTML Requests.

The Database, stringent measures will be implemented to safeguard the security of the data. The Backend development will prioritize transparency, culminating in the provision of comprehensive designs to stakeholders elucidating the underlying algorithms and mechanisms.

For the Frontend, user-centric features will be pivotal, encompassing functionalities such as Account Creation, Login, and Logout to facilitate Authentication. Subsequently, authenticated users will have access to the following features:

- Movie Search:

Users will be presented with an input form enabling them to retrieve a list of movies corresponding to their query.

- Movie Suggestions:

An intuitive recommendation system will furnish users with top movie recommendations tailored to their preferences.

Upon selecting a movie from the movies list, users will be redirected to a dedicated page showcasing detailed information about the movie. Here, users can engage in activities such as adding movies to user mylist to rate later, removing movies from user mylist , rating creation, updating rating, and deleting rating.

4. Technologies used

- Django Framework for Backend.
- ReactJS library for Frontend.
- MySQL of for Database.
- Git for Source control.
- Draw.io for Designing.
- Postman for Unit test.
- Azure Dev for Scrum management and CICD.
- Microsoft Azure for Deployment.

5. Requirements Analysis

5.1 User Functional requirements

The software will allow the user to perform the various functions:

- Create Account
- Login
- Logout

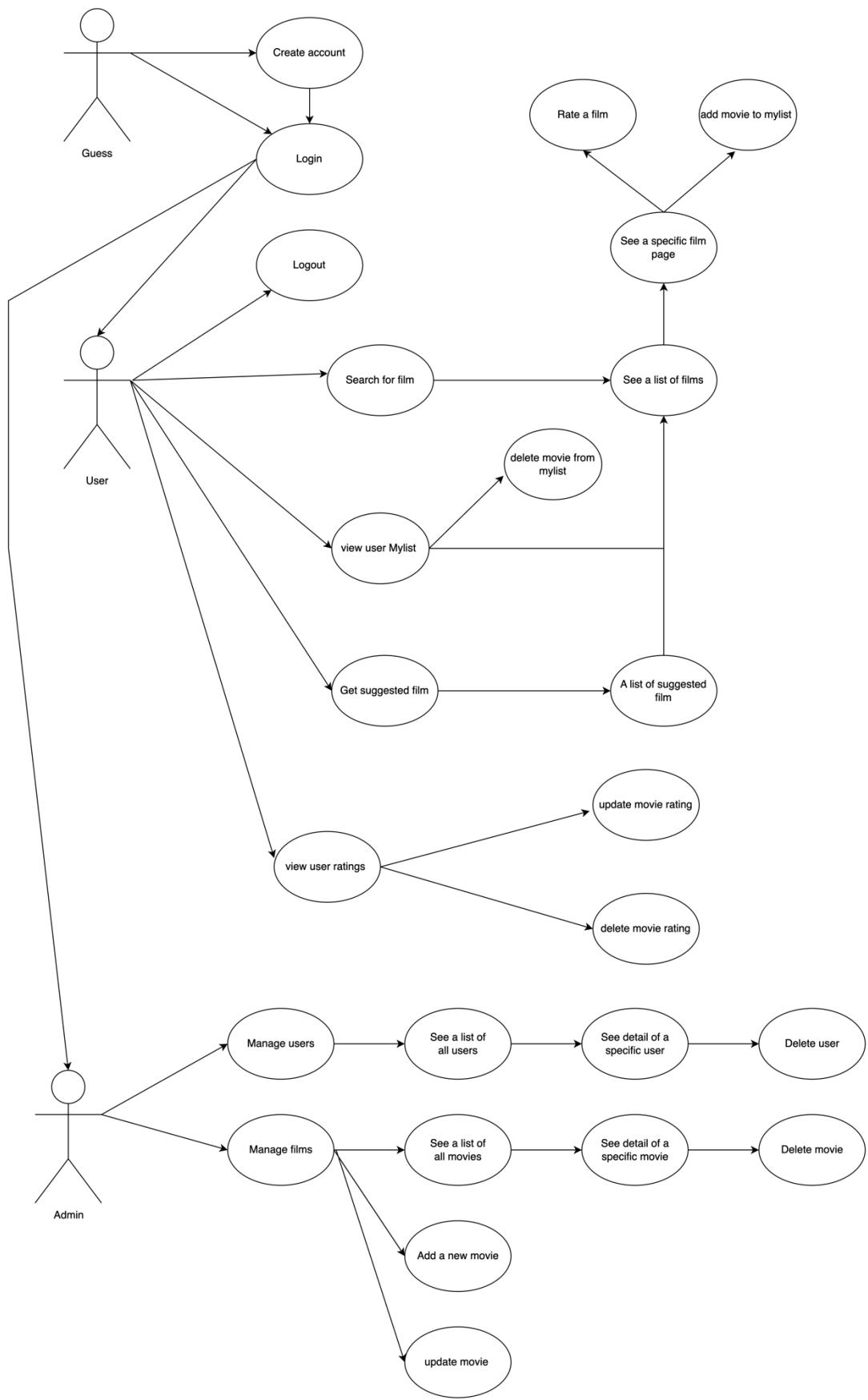
- Search for movie
- Get suggested movie
- Get a specific movie detail
- Create new rating
- Read existed rating
- Update existed rating
- Delete existed rating
- Add movie to user mylist to rate later
- Delete movie from user mylist

If the logged in user is an Admin, they will be able to perform the various functions:

- Login
- Logout
- Get a list of all users
- Get a specific user detail
- Delete a specific user
- Get a list of all movies
- Get a specific movie detail
- Delete a specific movie
- Add a new movie
- Update a movie

5.2 Use case diagram

From the Functional Requirements analysed in the previous section, I have come up with a usecase diagram as follows:



5.3 Functional requirements

Functional requirements refer to the product features or functions that must be implemented by the development team to enable users to carry out their tasks. It is important to make these requirements clear for both the development team and stakeholders. Typically, functional requirements describe the system's behavior in specific circumstances. Below are functional requirements for software:

FR.1: User Authenticate Management:

- FR.1.1: User Registration
- FR.1.2: User Login
- FR.1.3: User Verification
- FR.1.4: User Logout
- FR.1.5: User Load

FR.2: Movie Management:

- FR.2.1: User Search Movie
- FR.2.2: User Get Movie Details

FR.3: Rating Management:

- FR.3.1: User Create Rating
- FR.3.2: User Read Rating
- FR.3.3: User Update Rating
- FR.3.4: User Delete Rating

FR.4: Recommendation System:

- FR.4.1: User Get Suggested Movies
- FR.4.2: User Get Prediction for A Movie

FR.5: Admin Authenticate Management:

- FR.5.1: Admin Login
- FR.5.2: Admin Logout
- FR.5.3: Admin Load

FR.6: Admin Users Management:

- FR.6.1: Admin Get All Users
 - FR.6.2: Admin Get User Details
 - FR.6.3: Admin Delete User
- FR.7: Admin Movies management:
- FR.7.1: Admin Get All Movies
 - FR.7.2: Admin Get Movie Details
 - FR.7.3: Admin Add New Movie
 - FR.7.4: Admin Delete Movie

- FR.7: Admin Movies management:
- FR.7.1: Admin Get All Movies
- FR.7.2: Admin Get Movie Details
- FR.7.3: Admin Add New Movie
- FR.7.4: Admin Delete Movie

FR.8: User Mylist Management:

- FR.8.1: User add movie to user mylist
- FR.8.2: User get movies in user mylist
- FR.8.3: User Delete movies in mylist

5.4 Non-functional requirements

Nonfunctional requirements, not related to the system functionality, rather define how the system should perform. Below are non-functional requirements for software:

NFR.1: Friendly Graphic User Interface.

NFR.2: Multiplatform compatible.

NFR.3: Fast response to user.

NFR.4: Movie cover should be available

NFR.5: User passwords should be encrypted.

5.5 Domain requirements

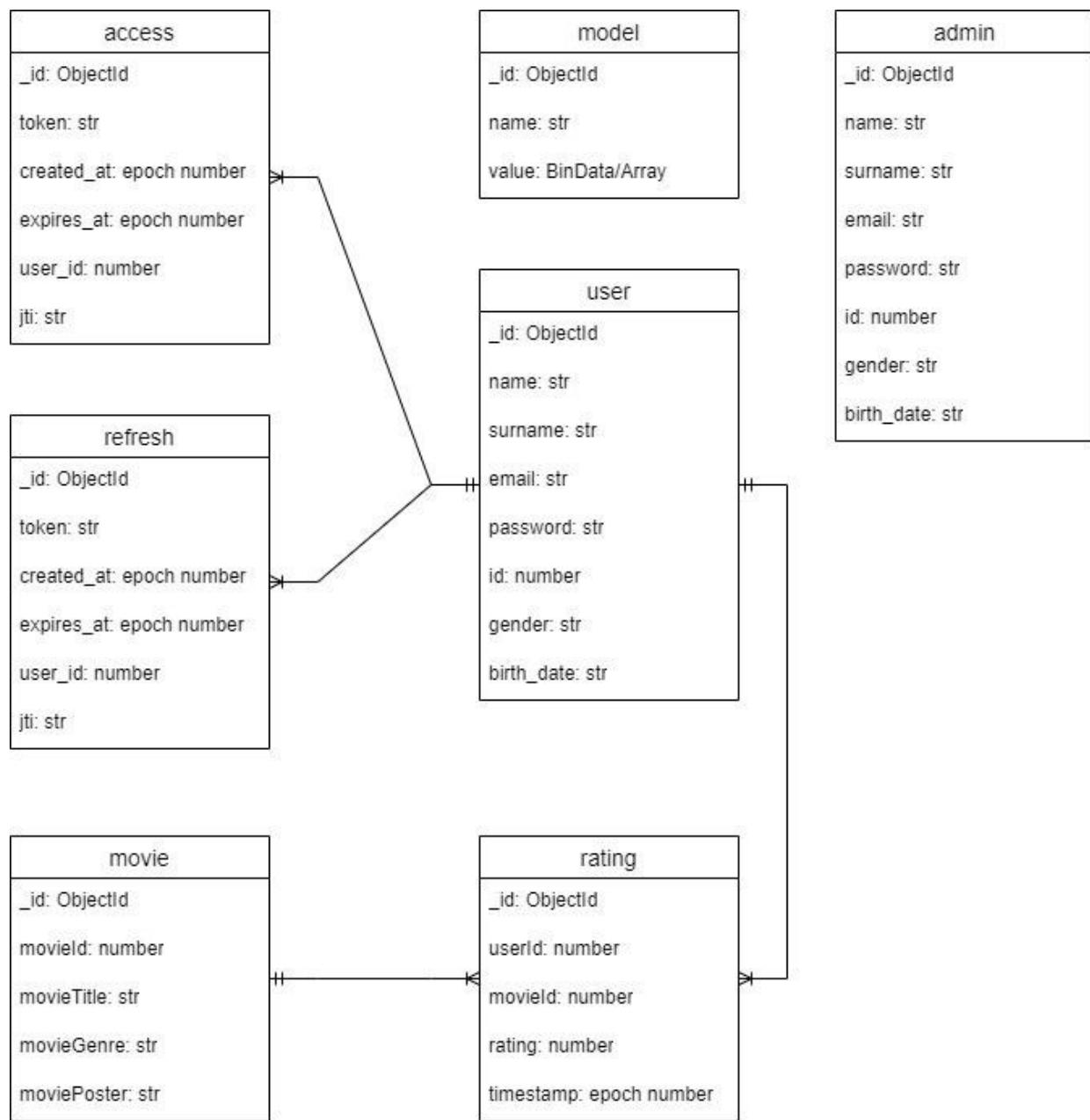
Since the production will be a web application, domain requirements can include expectations related to performance, scalability and usability. In this project, the domain requirements are:

DR.1: Must have internet connection.

DR.2: Access through a Browser.

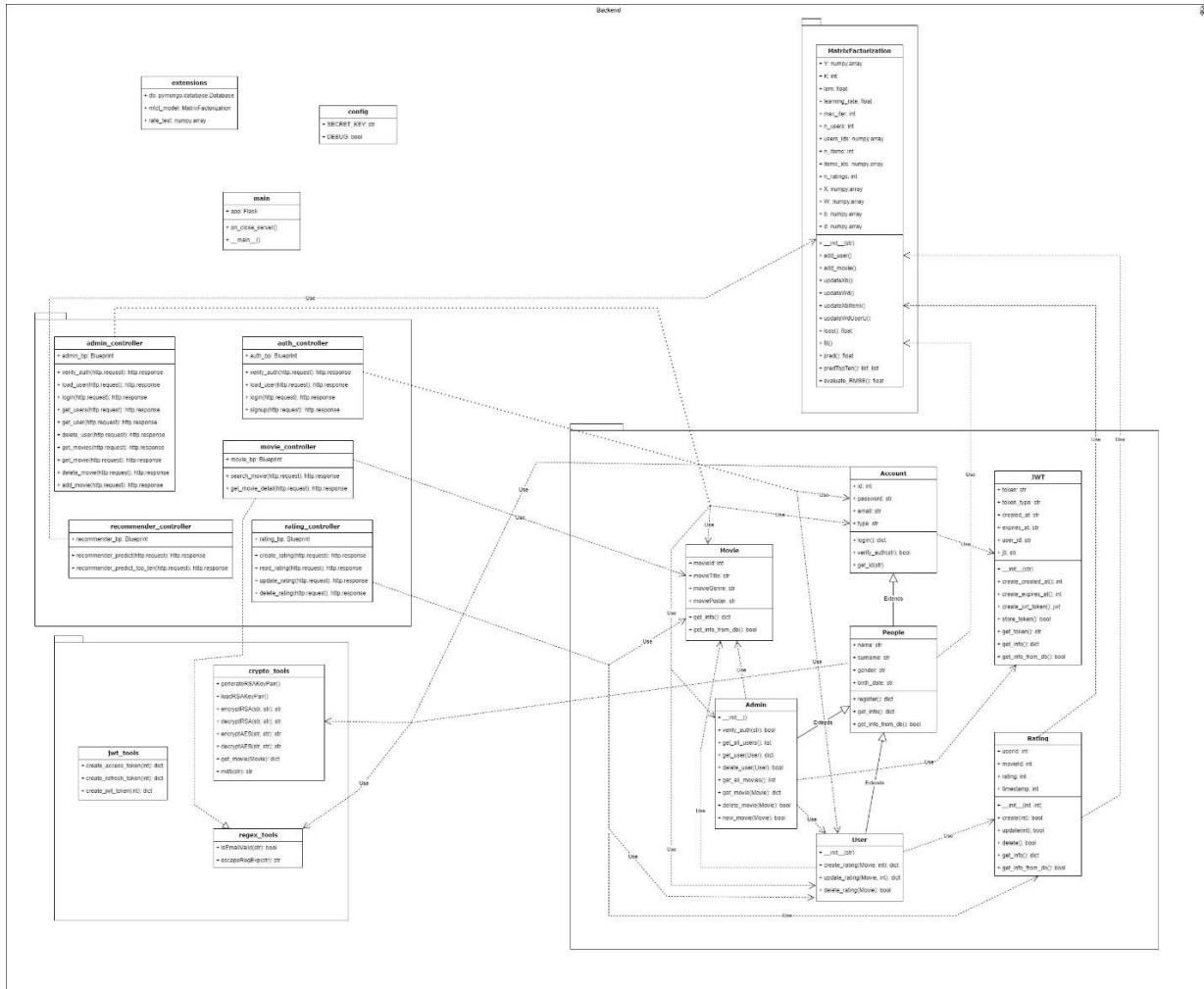
6. Architectural Design

6.1. Database Design

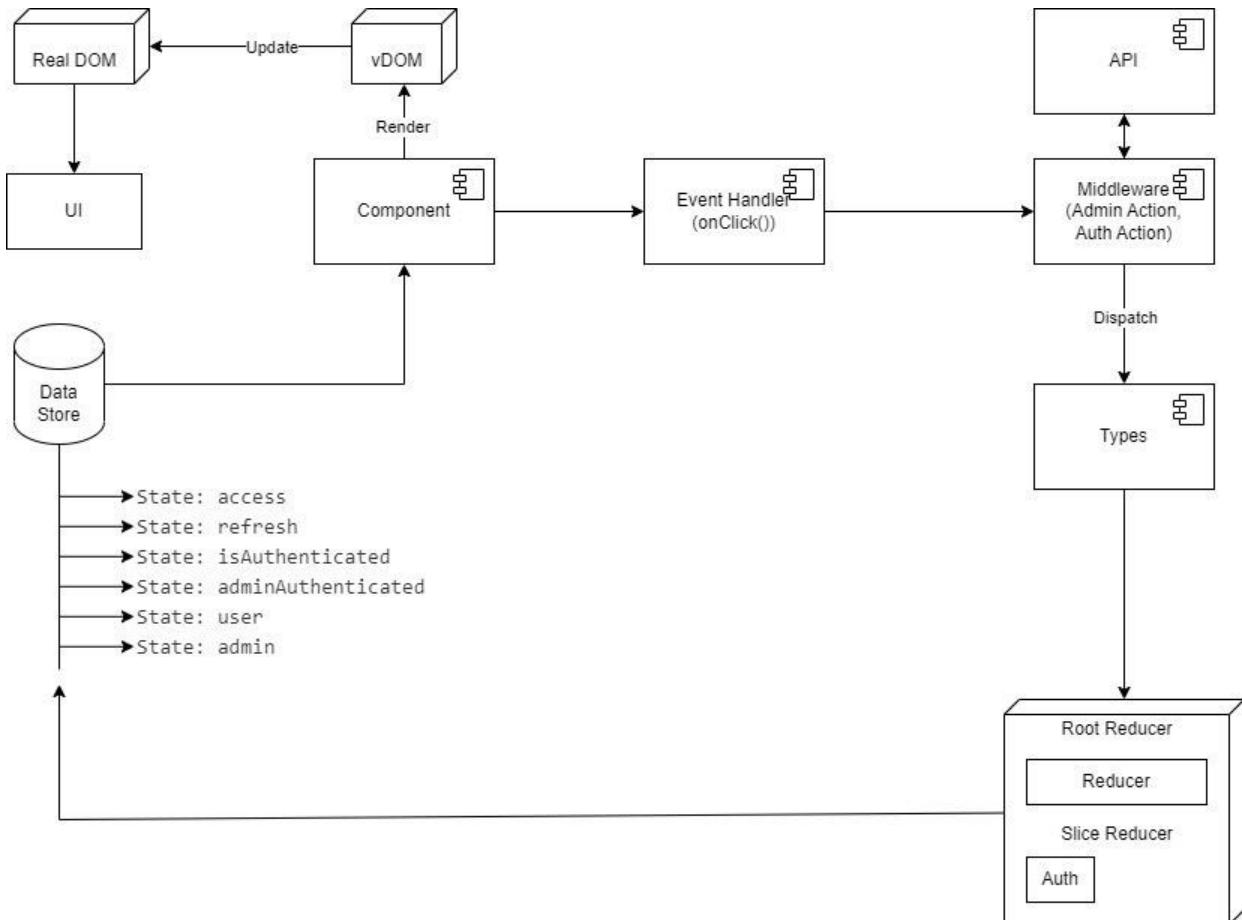


6.2. Backend Design

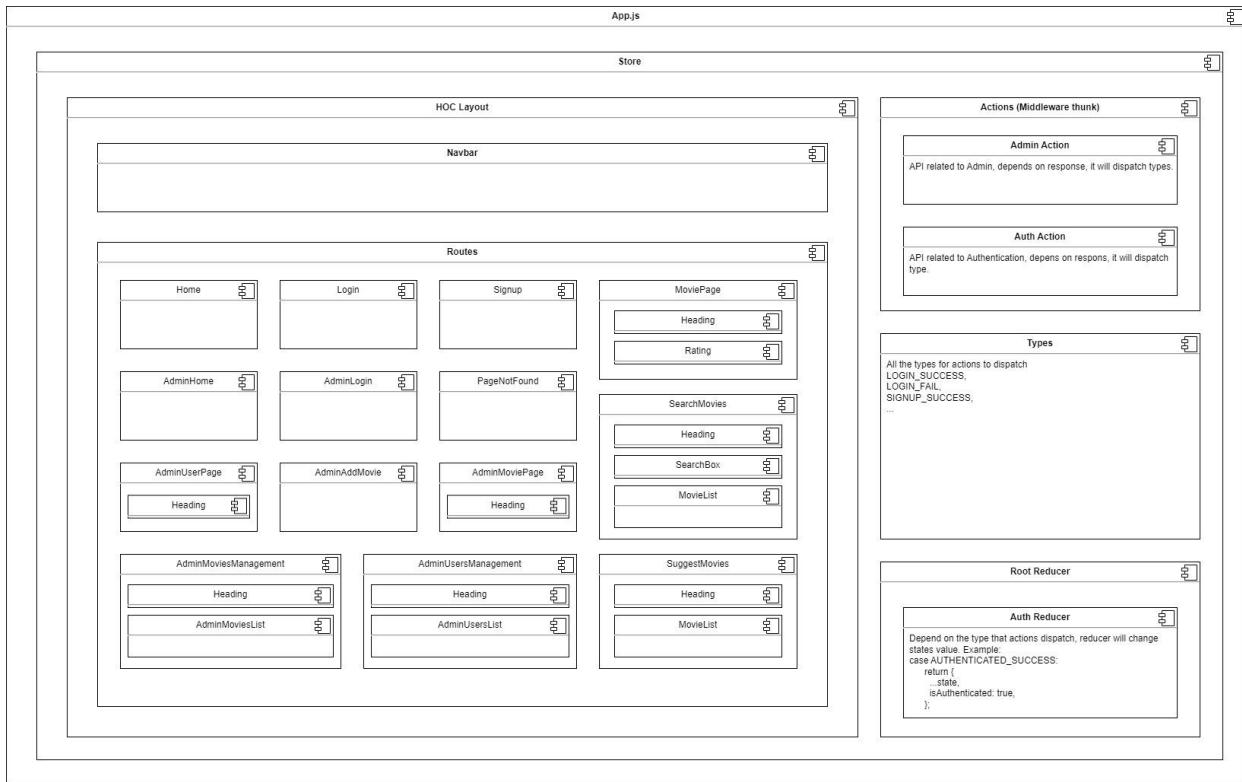
Class Diagram



6.3. Frontend Design

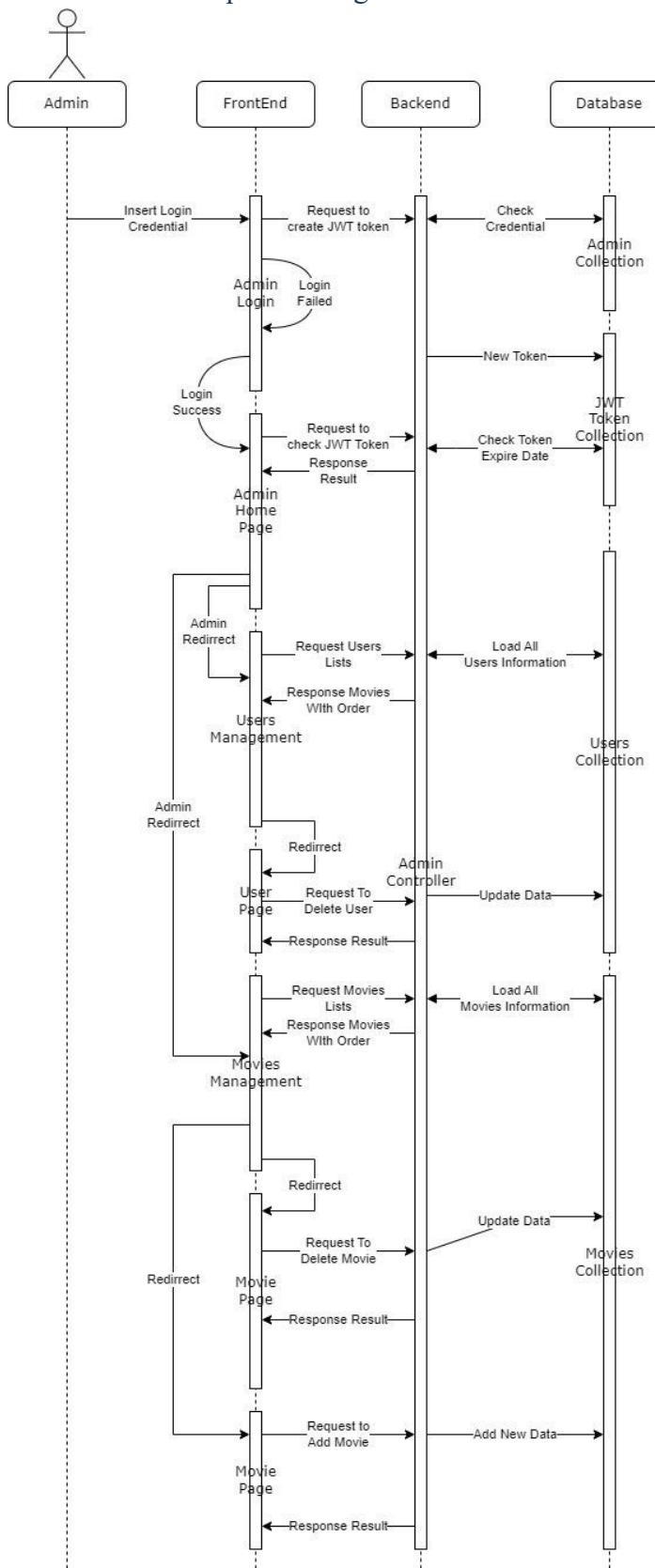


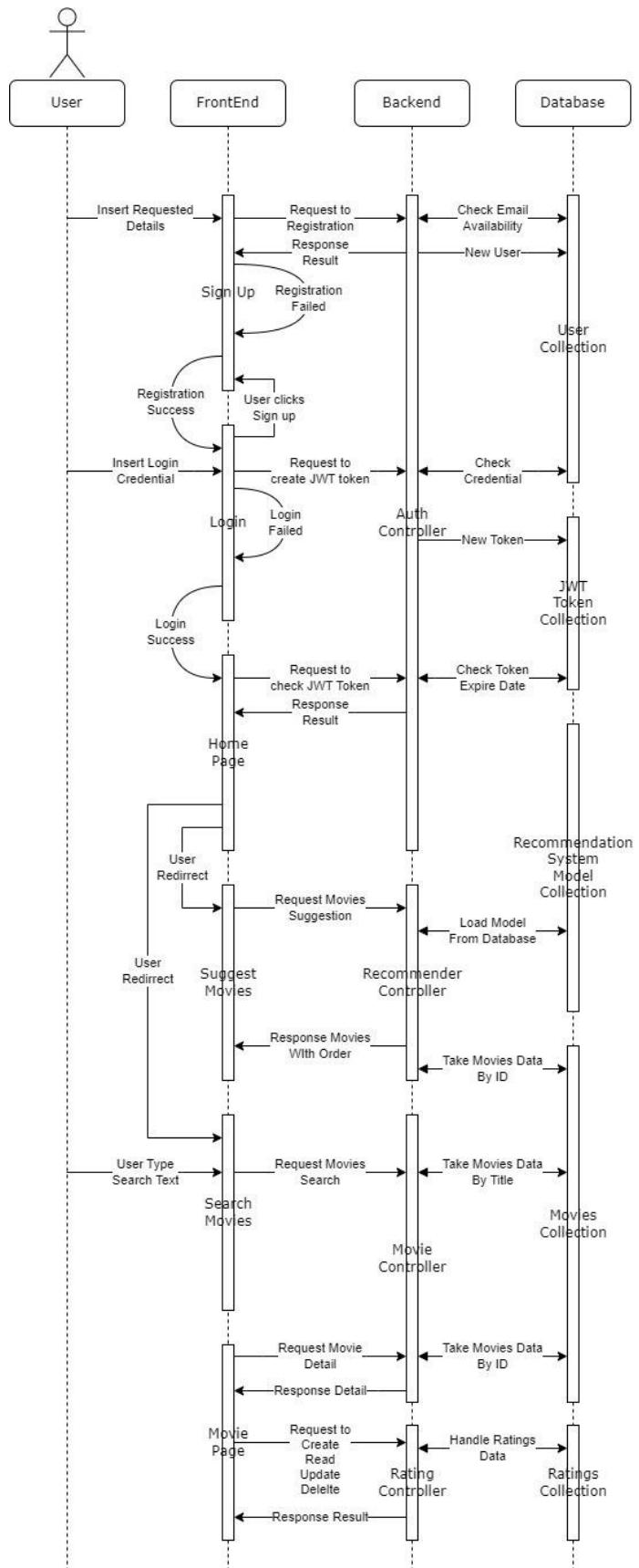
6.4. Ui Layer Design Diagram



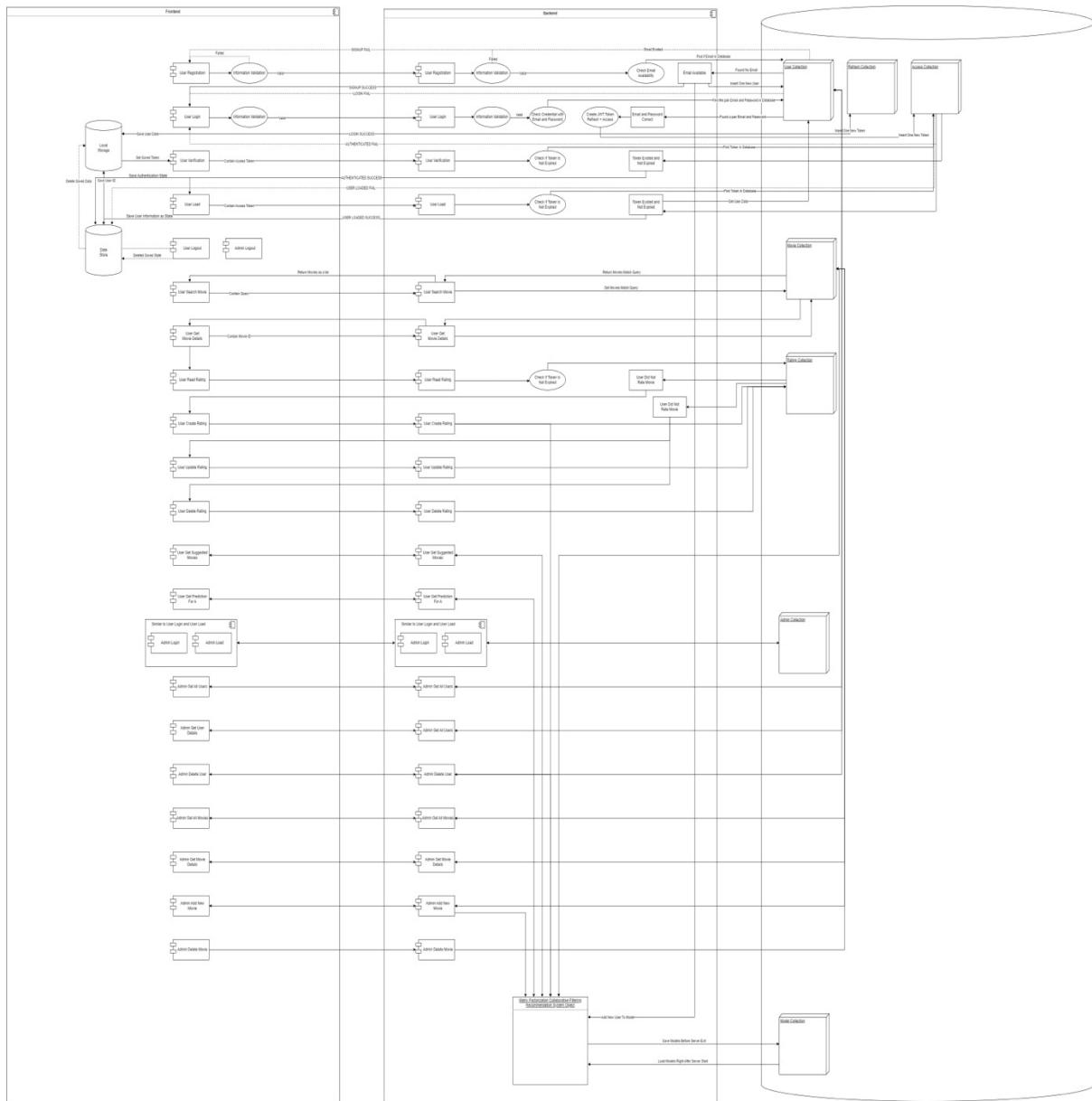
6.5.

Sequence Diagram





6.6. Components Diagram



7. Backlog refinement

From the Requirements Analysis, as a Product Owner, I have refined the backlog as follows.

Projects / Movie-Recommendation System / MRS board

Backlog

Q Search Version Epic Label Type Quick filters Insights View settings

Only My Issues Recently Updated

Epic

Issues without epic

> User Authentication

> Movies Management

> Rating Management

> User Movie List Management

> Movies Recommendation Algorithm

> Admin Authenticate Management

> User Data Management (Admin)

> Movies Data Management (Admin)

+ Create epic

Backlog (20 issues)

Issue	Version	Type	Labels	Progress	Created	Last updated	Actions	
MRS-21 As a user, I want to receive movie recommendations based on my rating...	VERSION 1	MOVIES RECOMMEND...	TO DO	0%	TR	TR	0%	0%
MRS-95 As a user, I want to see specific details of a movie on a new page when...	VERSION 1	MOVIES MANAGEMENT	TO DO	0%	TR	TR	0%	0%
MRS-91 As a user, I want to see a list of movies available to give the ratings .	VERSION 1	MOVIES MANAGEMENT	TO DO	0%	TR	TR	0%	0%
MRS-10 As a user, I want to register so that I can create an account.	VERSION 1	USER AUTHENTICAT...	TO DO	0%	TR	TR	0%	0%
MRS-11 As a user, I want to log in so that I can access my account.	VERSION 1	USER AUTHENTICAT...	TO DO	0%	TR	TR	0%	0%
MRS-12 As a user, I want to log out so that I can end my session.	VERSION 1	USER AUTHENTICAT...	TO DO	0%	TR	TR	0%	0%
MRS-13 As a user, I want to give ratings to movies.	VERSION 1	RATING MANAGEMENT	TO DO	0%	TR	TR	0%	0%
MRS-15 As a user, I want to view my given ratings.	VERSION 1	RATING MANAGEMENT	TO DO	0%	TR	TR	0%	0%
MRS-16 As a user, I want to update my ratings.	VERSION 1	RATING MANAGEMENT	TO DO	0%	TR	TR	0%	0%
MRS-17 As a user, I want to delete my ratings	VERSION 1	RATING MANAGEMENT	TO DO	0%	TR	TR	0%	0%
MRS-19 As a user, I want to view my "rate later" list.	VERSION 1	USER MOVIE LIST MA...	TO DO	0%	TR	TR	0%	0%
MRS-18 As a user, I want to add movies to my "rate later" list.	VERSION 1	USER MOVIE LIST MA...	TO DO	0%	TR	TR	0%	0%
MRS-20 As a user, I want to remove movies from my "rate later" list.	VERSION 1	USER MOVIE LIST MA...	TO DO	0%	TR	TR	0%	0%
MRS-22 As an admin, I want to log in securely so that I can manage the system.	VERSION 1	ADMIN AUTHENTICAT...	TO DO	0%	TR	TR	0%	0%
MRS-23 As an admin, I want to add movies to the database.	VERSION 1	MOVIES DATA MANAG...	TO DO	0%	TR	TR	0%	0%
MRS-25 As an admin, I want to update movie details.	VERSION 1	MOVIES DATA MANAG...	TO DO	0%	TR	TR	0%	0%
MRS-24 As an admin, I want to delete movies from the database.	VERSION 1	MOVIES DATA MANAG...	TO DO	0%	TR	TR	0%	0%
MRS-26 As an admin, I want to view the list of active users.	VERSION 1	USER DATA MANAGE...	TO DO	0%	TR	TR	0%	0%

0 0 0 Create sprint

7.1 Epic

From 8 Functional Requirements we analyzed 8 epics as stakeholders require in User Story, I created 27 corresponding user stories. Each user story has a Requirement of 3 for building the system Backend, Frontend and database.

Backlog

Search TR User Version Epic 1 Label Type Quick filters Clear filters Insights View settings

Only My Issues Recently Updated

Epic

- Issues without epic
- > User Authentication
- > Movies Management
- > Rating Management
- > User Movie List Management
- > Movies Recommendation Algorithm
- > Admin Authenticate Management
- > User Data Management (Admin)
- > Movies Data Management (Admin)
- + Create epic

+ Create issue

Backlog (3 of 20 issues visible) 0 0 0 Create sprint

MRS-10	VERSION 1	USER AUTHENTICATION	TO DO	TR
MRS-36 Create a registration form in React.	VERSION 1	TO DO	TR	
MRS-31 Validate registration data on the frontend.	VERSION 1	TO DO	TR	
MRS-33 Develop API endpoint in Django to handle user registration.	VERSION 1	TO DO	TR	
MRS-34 Store user information securely in the database.	VERSION 1	TO DO	TR	
MRS-11	VERSION 1	USER AUTHENTICATION	TO DO	TR
MRS-37 Create a login form in React.	VERSION 1	TO DO	TR	
MRS-38 Validate login credentials on the frontend.	VERSION 1	TO DO	TR	
MRS-39 Develop API endpoint in Django to handle user login.	VERSION 1	TO DO	TR	
MRS-40 Implement JWT token authentication.	VERSION 1	TO DO	TR	
MRS-12	VERSION 1	USER AUTHENTICATION	TO DO	TR
MRS-29 Create a logout button in React.	VERSION 1	TO DO	TR	
MRS-30 Invalidate the JWT token on the frontend.	VERSION 1	TO DO	TR	

+ Create issue

MRS Sprint 2 Add dates (0 issues) 0 0 0 Create sprint

Plan a sprint by dragging the sprint footer down below some issues, or by dragging issues here.

+ Create issue

Backlog (2 of 20 issues visible) 0 0 0 Create sprint

MRS-95	VERSION 1	MOVIES MANAGEMENT	TO DO	TR
MRS-96 Develop API endpoint in Django to retrieve movie details.	VERSION 1	TO DO	TR	
MRS-97 Create a movie details UI in React.	VERSION 1	TO DO	TR	
MRS-98 Implement navigation to the movie details page when a movie is clicked	VERSION 1	TO DO	TR	
MRS-99 Fetch and display the specific details of the selected movie on the movie list UI.	VERSION 1	TO DO	TR	
MRS-91	VERSION 1	MOVIES MANAGEMENT	TO DO	TR
MRS-92 Develop API endpoint in Django to retrieve the list of movies.	VERSION 1	TO DO	TR	
MRS-93 Create a movie list UI in React.	VERSION 1	TO DO	TR	
MRS-94 Fetch and display the list of movies in the movie list UI.	VERSION 1	TO DO	TR	

Epic	Issues without epic	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
Backlog (4 of 20 issues visible)					
> User Authentication	MRS-13 As a user, I want to give ratings to movies.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
> Movies Management	MRS-41 Create a rating submission form in React. Validate rating data on the front...	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
> Rating Management	MRS-42 Validate rating data on the frontend.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
> User Movie List Management	MRS-43 Develop API endpoint in Django to handle rating submissions.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
> Movies Recommendation Algorithm	MRS-44 Store rating information in the database.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
> Admin Authenticate Management	MRS-15 As a user, I want to view my given ratings.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
> User Data Management (Admin)	MRS-45 Develop API endpoint in Django to retrieve user ratings.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
> Movies Data Management (Admin)	MRS-46 Display user ratings in React.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
+ Create epic	MRS-16 As a user, I want to update my ratings.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
	MRS-47 Create a rating update form in React.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
	MRS-48 Validate updated rating data on the frontend.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
	MRS-49 Develop API endpoint in Django to handle rating updates.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
	MRS-50 Update rating information in the database.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
	MRS-17 As a user, I want to delete my ratings	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
	MRS-51 Create a delete rating button in React.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
	MRS-52 Develop API endpoint in Django to handle rating deletions.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint
	MRS-53 Remove rating information from the database.	VERSION 1	RATING MANAGEMENT	TO DO	0 0 0 Create sprint

+ Create issue

Epic	Issues without epic	VERSION 1	USER MOVIE LIST MA...	TO DO	0 0 0 Create sprint
Backlog (3 of 20 issues visible)					
> Rating Management	MRS-19 As a user, I want to view my "rate later" list.	VERSION 1	USER MOVIE LIST MA...	TO DO	0 0 0 Create sprint
> User Movie List Management	MRS-54 Develop API endpoint in Django to retrieve the "rate later" list.	VERSION 1	USER MOVIE LIST MA...	TO DO	0 0 0 Create sprint
> Movies Recommendation Algorithm	MRS-55 Display the "rate later" list in React.	VERSION 1	USER MOVIE LIST MA...	TO DO	0 0 0 Create sprint
> Admin Authenticate Management	MRS-18 As a user, I want to add movies to my "rate later" list.	VERSION 1	USER MOVIE LIST MA...	TO DO	0 0 0 Create sprint
> User Data Management (Admin)	MRS-56 Create an "add to rate later" button in React.	VERSION 1	USER MOVIE LIST MA...	TO DO	0 0 0 Create sprint
> Movies Data Management (Admin)	MRS-57 Develop API endpoint in Django to handle adding movies to the list.	VERSION 1	USER MOVIE LIST MA...	TO DO	0 0 0 Create sprint
+ Create epic	MRS-58 Store movie information in the user-specific list in the database.	VERSION 1	USER MOVIE LIST MA...	TO DO	0 0 0 Create sprint
	MRS-20 As a user, I want to remove movies from my "rate later" list.	VERSION 1	USER MOVIE LIST MA...	TO DO	0 0 0 Create sprint
	MRS-59 Create a remove button in React for each movie in the user rate later list.	VERSION 1	USER MOVIE LIST MA...	TO DO	0 0 0 Create sprint
	MRS-60 Develop API endpoint in Django to handle removing movies from the list.	VERSION 1	USER MOVIE LIST MA...	TO DO	0 0 0 Create sprint
	MRS-61 Update the user-specific list in the database.	VERSION 1	USER MOVIE LIST MA...	TO DO	0 0 0 Create sprint

+ Create issue

Epic	Issues without epic	VERSION 1	MOVIES RECOMMEND...	TO DO	0 0 0 Create sprint
Backlog (1 of 20 issues visible)					
> Movies Recommendation Algorithm	MRS-21 As a user, I want to receive movie recommendations based on my rating...	VERSION 1	MOVIES RECOMMEND...	TO DO	0 0 0 Create sprint
> Admin Authenticate Management	MRS-62 Research and choose a recommendation algorithm	VERSION 1	MOVIES RECOMMEND...	TO DO	0 0 0 Create sprint
> User Data Management (Admin)	MRS-63 Implement the recommendation algorithm in Django.	VERSION 1	MOVIES RECOMMEND...	TO DO	0 0 0 Create sprint
> Movies Data Management (Admin)	MRS-64 Develop API endpoint in Django to fetch recommendations.	VERSION 1	MOVIES RECOMMEND...	TO DO	0 0 0 Create sprint
+ Create epic	MRS-65 Create a recommendation feature button in React.	VERSION 1	MOVIES RECOMMEND...	TO DO	0 0 0 Create sprint
	MRS-66 Display recommended movies in React.	VERSION 1	MOVIES RECOMMEND...	TO DO	0 0 0 Create sprint

+ Create issue

Epic	Issues without epic	VERSION 1	ADMIN AUTHENTICAT...	TO DO	0 0 0 Create sprint
Backlog (1 of 20 issues visible)					
> Admin Authenticate Management	MRS-22 As an admin, I want to log in securely so that I can manage the system.	VERSION 1	ADMIN AUTHENTICAT...	TO DO	0 0 0 Create sprint
> User Data Management (Admin)	MRS-67 Create an admin login form in React.	VERSION 1	ADMIN AUTHENTICAT...	TO DO	0 0 0 Create sprint
> Movies Data Management (Admin)	MRS-68 Validate admin login credentials on the frontend.	VERSION 1	ADMIN AUTHENTICAT...	TO DO	0 0 0 Create sprint
+ Create epic	MRS-69 Develop API endpoint in Django to handle admin login.	VERSION 1	ADMIN AUTHENTICAT...	TO DO	0 0 0 Create sprint
	MRS-70 Implement JWT token authentication for admins.	VERSION 1	ADMIN AUTHENTICAT...	TO DO	0 0 0 Create sprint

+ Create issue

The screenshot shows the Jira backlog for the 'Movies Recommendation Algorithm' epic. The sidebar on the left lists epics: 'Movies Recommendation Algorithm', 'Admin Authenticate Management', 'User Data Management (Admin)', 'Movies Data Management (Admin)', and '+ Create epic'. The main area displays a backlog of user stories under the 'Backlog' section, each with a summary, version, status, and priority.

User Story	Version	Status	Priority
MRS-26 As an admin, I want to view the list of active users.	VERSION 1	TO DO	TR
MRS-82 Develop API endpoint in Django to retrieve user list.	VERSION 1	TO DO	TR
MRS-83 Display user list to admin , create page in front React.	VERSION 1	TO DO	TR
MRS-27 As an admin, I want to delete a specific user.	VERSION 1	TO DO	TR
MRS-88 Create a delete user button in React.	VERSION 1	TO DO	TR
MRS-89 Develop API endpoint in Django to handle user deletions.	VERSION 1	TO DO	TR
MRS-90 Remove user information from the database.	VERSION 1	TO DO	TR

The screenshot shows the Jira backlog for the 'Movies Data Management (Admin)' epic. The sidebar on the left lists epics: 'Issues without epic', 'User Authentication', 'Movies Management', 'Rating Management', 'User Movie List Management', 'Movies Recommendation Algorithm', 'Admin Authenticate Management', 'User Data Management (Admin)', and '+ Create epic'. The main area displays a backlog of user stories under the 'Backlog' section, each with a summary, version, status, and priority.

User Story	Version	Status	Priority
MRS-23 As an admin, I want to add movies to the database.	VERSION 1	TO DO	TR
MRS-71 Create a movie addition form in React.	VERSION 1	TO DO	TR
MRS-72 Validate movie data on the frontend.	VERSION 1	TO DO	TR
MRS-73 Develop API endpoint in Django to handle movie additions.	VERSION 1	TO DO	TR
MRS-74 Store movie information in the database.	VERSION 1	TO DO	TR
MRS-25 As an admin, I want to update movie details.	VERSION 1	TO DO	TR
MRS-75 Create a movie update form in React.	VERSION 1	TO DO	TR
MRS-76 Validate updated movie data on the frontend.	VERSION 1	TO DO	TR
MRS-77 Develop API endpoint in Django to handle movie updates.	VERSION 1	TO DO	TR
MRS-78 Update movie information in the database.	VERSION 1	TO DO	TR
MRS-24 As an admin, I want to delete movies from the database.	VERSION 1	TO DO	TR
MRS-79 Create a delete movie button in React.	VERSION 1	TO DO	TR
MRS-80 Develop API endpoint in Django to handle movie deletions.	VERSION 1	TO DO	TR
MRS-81 Remove movie information from the database.	VERSION 1	TO DO	TR
MRS-28 As an admin, I want to update user information.	VERSION 1	TO DO	TR
MRS-84 Create a user update form in React.	VERSION 1	TO DO	TR
MRS-85 Validate updated user data on the frontend.	VERSION 1	TO DO	TR
MRS-86 Develop API endpoint in Django to handle user updates.	VERSION 1	TO DO	TR
MRS-87 Update user information in the database.	VERSION 1	TO DO	TR

Epic and User Stories Breakdown

1. Epic: User Authentication

1. User Story: As a user, I want to register so that I can create an account.

- Task: Create a registration form in React.
- Task: Validate registration data on the frontend.
- Task: Develop API endpoint in Django to handle user registration.
- Task: Store user information securely in the database.

2. User Story: As a user, I want to log in so that I can access my account.

- Task: Create a login form in React.
- Task: Validate login credentials on the frontend.
- Task: Develop API endpoint in Django to handle user login.
- Task: Implement JWT token authentication.

3. User Story: As a user, I want to log out so that I can end my session.

- Task: Create a logout button in React.
- Task: Invalidate the JWT token on the frontend.

2. Epic: Movie Ratings Management

1. User Story: As a user, I want to see a list of movies so that I can select movies to rate.
 - Task: Develop API endpoint in Django to retrieve the list of movies
 - Task: Create a movie list UI in React.
 - Task: Fetch and display the list of movies in the movie list UI.
 - Task: Implement pagination or infinite scroll for the movie list (if necessary).

3. Epic: Movie Ratings Management

1. User Story: As a user, I want to give ratings to movies.
 - Task: Create a rating submission form in React.
 - Task: Validate rating data on the frontend.
 - Task: Develop API endpoint in Django to handle rating submissions.
 - Task: Store rating information in the database.
2. User Story: As a user, I want to view my given ratings.
 - Task: Develop API endpoint in Django to retrieve user ratings.
 - Task: Display user ratings in React.
3. User Story: As a user, I want to update my ratings.
 - Task: Create a rating update form in React.
 - Task: Validate updated rating data on the frontend.
 - Task: Develop API endpoint in Django to handle rating updates.
 - Task: Update rating information in the database.
4. User Story: As a user, I want to delete my ratings.
 - Task: Create a delete rating button in React.
 - Task: Develop API endpoint in Django to handle rating deletions.
 - Task: Remove rating information from the database.

4. Epic: User Movie List Management

1. User Story: As a user, I want to add movies to my "rate later" list.
 - Task: Create an "add to rate later" button in React.
 - Task: Develop API endpoint in Django to handle adding movies to the list.
 - Task: Store movie information in the user-specific list in the database.
2. User Story: As a user, I want to view my "rate later" list.
 - Task: Develop API endpoint in Django to retrieve the "rate later" list.
 - Task: Display the "rate later" list in React.
3. User Story: As a user, I want to remove movies from my "rate later" list.
 - Task: Create a remove button in React for each movie in the list.

- Task: Develop API endpoint in Django to handle removing movies from the list.
- Task: Update the user-specific list in the database.

5. Epic: Movie Recommendations

1. User Story: As a user, I want to receive movie recommendations based on my ratings.
 - Task: Research and choose a recommendation algorithm.
 - Task: Implement the recommendation algorithm in Django.
 - Task: Develop API endpoint in Django to fetch recommendations.
 - Task: Create a recommendation feature button in React.
 - Task: Display recommended movies in React.

6. Epic: Admin Authentication

1. User Story: As an admin, I want to log in securely so that I can manage the system.
 - Task: Create an admin login form in React.
 - Task: Validate admin login credentials on the frontend.
 - Task: Develop API endpoint in Django to handle admin login.
 - Task: Implement JWT token authentication for admins.

7. Epic: Movie Management (Admin)

1. User Story: As an admin, I want to add movies to the database.
 - Task: Create a movie addition form in React.
 - Task: Validate movie data on the frontend.
 - Task: Develop API endpoint in Django to handle movie additions.
 - Task: Store movie information in the database.
2. User Story: As an admin, I want to delete movies from the database.
 - Task: Create a delete movie button in React.
 - Task: Develop API endpoint in Django to handle movie deletions.
 - Task: Remove movie information from the database.
3. User Story: As an admin, I want to update movie details.
 - Task: Create a movie update form in React.
 - Task: Validate updated movie data on the frontend.
 - Task: Develop API endpoint in Django to handle movie updates.
 - Task: Update movie information in the database.

8. Epic: User Management (Admin)

1. User Story: As an admin, I want to view the list of active users.
 - Task: Develop API endpoint in Django to retrieve user list.
 - Task: Display user list in React.

2. User Story: As an admin, I want to delete a specific user.
 - Task: Create a delete user button in React.
 - Task: Develop API endpoint in Django to handle user deletions.
 - Task: Remove user information from the database.

3. User Story: As an admin, I want to update user information.
 - Task: Create a user update form in React.
 - Task: Validate updated user data on the frontend.
 - Task: Develop API endpoint in Django to handle user updates.
 - Task: Update user information in the database.

Apart from 8 Functional Requirements, there are 3 Design tasks that must done before. Database Design, Backend Architecture Design, Frontend Architecture Design.

The screenshot shows a Jira backlog with the following items:

- MRS-103 research on Movie recommendation syst... (Status: TO DO, TR)
- MRS-100 Database Design (Status: TO DO, TR)
- MRS-101 Backend Architecture Design (Status: TO DO, TR)
- MRS-102 Frontend Architecture Diagram (Status: TO DO, TR)

At the top right, it says "0 of 0 issues visible" and "Estimate: 0 of 0". There are buttons for "Create sprint" and "Create issue".

7.2 Priority and Story points assignment

There are 4 priority levels in this project.

1. Must do
2. High Priority
3. Second-level Priority
4. Low Priority

All product backlogs related to Design will have priority Must do.

1. Epic: User Authentication

Priority: Must Do

- Story Points: 20

1. User Story: As a user, I want to register so that I can create an account.
 - Priority: Must Do
 - Story Points: 8

2. User Story: As a user, I want to log in so that I can access my account.

- Priority: High Priority
- Story Points: 8

3. User Story: As a user, I want to log out so that I can end my session.

- Priority: Second-level Priority
- Story Points: 4

2. Epic: Movie Management

Epic Priority: Must Do

- Story Points: 20

1. User Story: As a user, I want to see a list of movies available to give the ratings.

- Priority: Must Do
- Story Points: 8

2. User Story: As a user, I want to see specific details of a movie on a new page when I click on it, so that I can decide if I want watch the movie and rate it as a user.

- Priority: High Priority
- Story Points: 6

3. Epic: Movie Ratings Management

Epic Priority: Must Do

- Story Points: 40

1. User Story: As a user, I want to see a list of movies so that I can select movies to rate.

- Priority: Must Do
- Story Points: 10

2. User Story: As a user, I want to give ratings to movies.

- Priority: Must Do
- Story Points: 8

3. User Story: As a user, I want to view my given ratings.

- Priority: High Priority
- Story Points: 6

4. User Story: As a user, I want to update my ratings.

- Priority: High Priority
- Story Points: 8

5. User Story: As a user, I want to delete my ratings.

- Priority: High Priority
- Story Points: 8

4 . Epic: User Movie List Management

Epic Priority: High Priority

- Story Points: 18

1. User Story: As a user, I want to add movies to my "rate later" list.

- Priority: Must Do
- Story Points: 6

2. User Story: As a user, I want to view my "rate later" list.

- Priority: High Priority
- Story Points: 6

3. User Story: As a user, I want to remove movies from my "rate later" list.

- Priority: Second-level Priority
- Story Points: 6

5 . Epic: Movie Recommendations

Epic Priority: High Priority

- Story Points: 13

1. User Story: As a user, I want to receive movie recommendations based on my ratings.

- Priority: Must Do
- Story Points: 13

6. Epic: Admin Authentication (Admin)

Epic Priority: Second-level Priority

- Story Points: 8

1. User Story: As an admin, I want to log in securely so that I can manage the system.

- Priority: Second-level Priority
- Story Points: 8

7. Epic: Movie Management (Admin)

Epic Priority: Second-level Priority

- Story Points: 20

1. User Story: As an admin, I want to add movies to the database.

- Priority: Low Priority
- Story Points: 8

2. User Story: As an admin, I want to delete movies from the database.

- Priority: Low Priority

- Story Points: 6
3. User Story: As an admin, I want to update movie details.
- Priority: Low Priority
 - Story Points: 6

8. Epic: User Management (Admin)

Epic Priority: Second-level Priority

- Story Points: 20

1. User Story: As an admin, I want to view the list of active users.
 - Priority: Low Priority
 - Story Points: 8
2. User Story: As an admin, I want to delete a specific user.
 - Priority: Low Priority
 - Story Points: 6
3. User Story: As an admin, I want to update user information.
 - Priority: Low Priority
 - Story Points: 6 .

7.3 Project estimation (time)

In this project, 1 Sprint is 5 working day long.

There are 4 Sprints:

1. Sprint 1 runs from 15/May/2024 to 19/May/2024. (5 working days).
2. Sprint 2 runs from 20/ May/2024 to 24/ May/2024. (5 working days)
3. Sprint 3 runs from 26/ May/2024 to 30/May/2024. (5 working days).
4. Sprint 4 runs from 01/ June/2024 to 05/June/2024. (5 working days) .

All the Architecture Design will be released in sprint 1 .

All the Epics with Priority: [Must Do] will be released in sprint 2 .

All the Epics with Priority: [High Priority] will be released in sprint 3 .

All the Epics with Priority: [Second-level Priority] will be released in sprint 4..

VII. Sprint cycles

1. Sprint 1

Sprint Backlog:

The screenshot shows a Jira backlog board for the 'Movie-Recommendation System' project. The left sidebar lists various epics: 'User Authentication', 'Movies Management', 'User Movie List Management', 'Rating Management', 'Movies Recommendation Algorithm', 'Admin Authenticate Management', 'User Data Management (Admin)', and 'Movies Data Management (Admin)'. The main area displays the 'MRS Sprint 1' backlog for the period from May 15 to May 20, containing 3 issues. These issues are: 'MRS-110 Database Design' (with sub-tasks 'MRS-118 Database schema Design', 'MRS-114 Class Diagram', and 'MRS-115 Component Diagram'), 'MRS-111 Backend Architecture Design' (with sub-tasks 'MRS-116 Redux Design' and 'MRS-117 UI Layer Design'), and 'MRS-112 Frontend Architecture Design'. Each issue has a status column showing 'TO DO' and a progress bar indicating completion. Below the sprint backlog, there are sections for 'MRS Sprint 2' (9 issues), 'MRS Sprint 3' (5 issues), 'MRS Sprint 4' (6 issues), and a 'Backlog' section which is currently empty.

Above requirements are Delivered in this sprint

1.1 Planning

In Scrum Planning, we will do task identification. The team identifies the tasks that need to be complete to deliver the product backlog items that are selected for the sprint.

In Sprint 1, tasks will be as below:

The board has three columns: TO DO, IN PROGRESS, and DONE.

- TO DO:**
 - MRS-110 Database Design (1 subtask) **IN PROGRESS**
 - Database schema Design
MRS-118
 - + Create issue
 - MRS-111 Backend Architecture Design (2 subtasks) **IN PROGRESS**
 - Class Diagram
MRS-114
 - Component Diagram
MRS-115
 - + Create issue
 - MRS-112 Frontend Architecture Design (2 subtasks) **IN PROGRESS**
 - Redux Design
MRS-116
 - UI Layer Design
MRS-117
 - + Create issue
 - Everything else (2 issues)
 - 7.1. Components Diagram
MRS-120
 - Sequence Diagram
MRS-119
 - + Create issue
- IN PROGRESS:**
 - 1 item: Database schema Design (MRS-118)
 - 1 item: Class Diagram (MRS-114)
 - 1 item: Component Diagram (MRS-115)
 - 1 item: Redux Design (MRS-116)
 - 1 item: UI Layer Design (MRS-117)
 - 1 item: 7.1. Components Diagram (MRS-120)
 - 1 item: Sequence Diagram (MRS-119)
- DONE:**
 - 0 items

1.2 Implementation

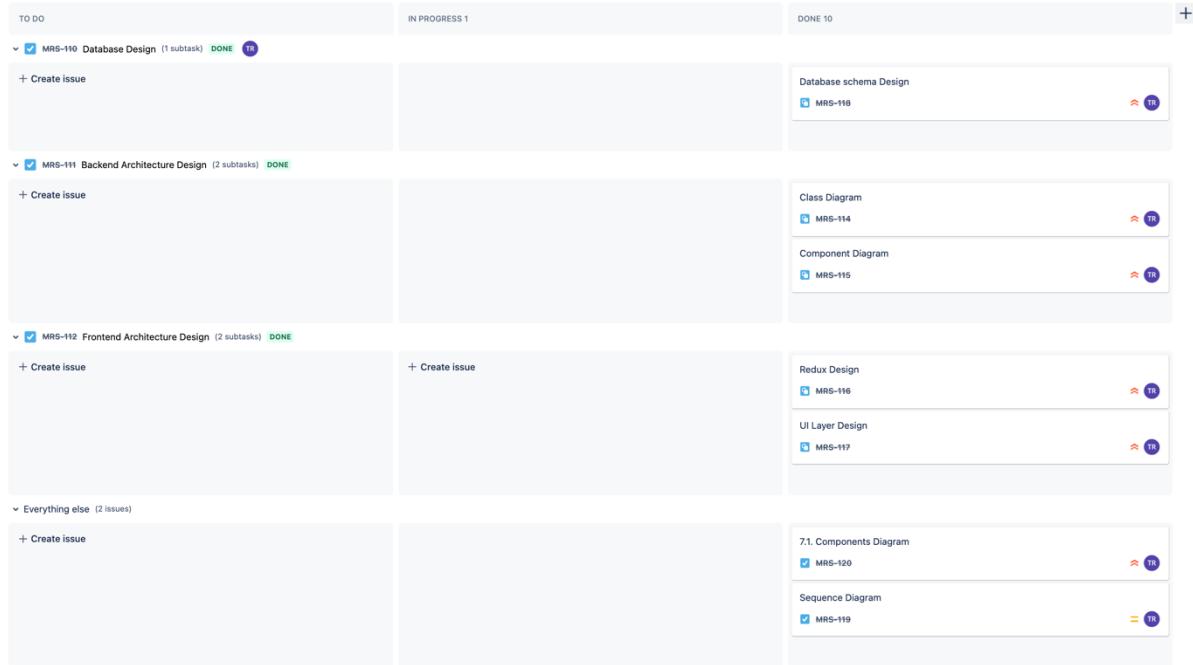
All the designs are located in section 6. Architectural Design.

The board has three columns: TO DO, IN PROGRESS, and DONE.

- TO DO:**
 - MRS-110 Database Design (1 subtask) **IN PROGRESS**
 - + Create issue
 - MRS-111 Backend Architecture Design (2 subtasks) **IN PROGRESS**
 - + Create issue
 - MRS-112 Frontend Architecture Design (2 subtasks) **IN PROGRESS**
 - + Create issue
 - Everything else (2 issues)
 - + Create issue
- IN PROGRESS:**
 - 1 item: Database schema Design (MRS-118)
 - 1 item: Class Diagram (MRS-114)
 - 1 item: Component Diagram (MRS-115)
 - 1 item: Redux Design (MRS-116)
 - 1 item: UI Layer Design (MRS-117)
 - 1 item: 7.1. Components Diagram (MRS-120)
 - 1 item: Sequence Diagram (MRS-119)
- DONE:**
 - 0 items

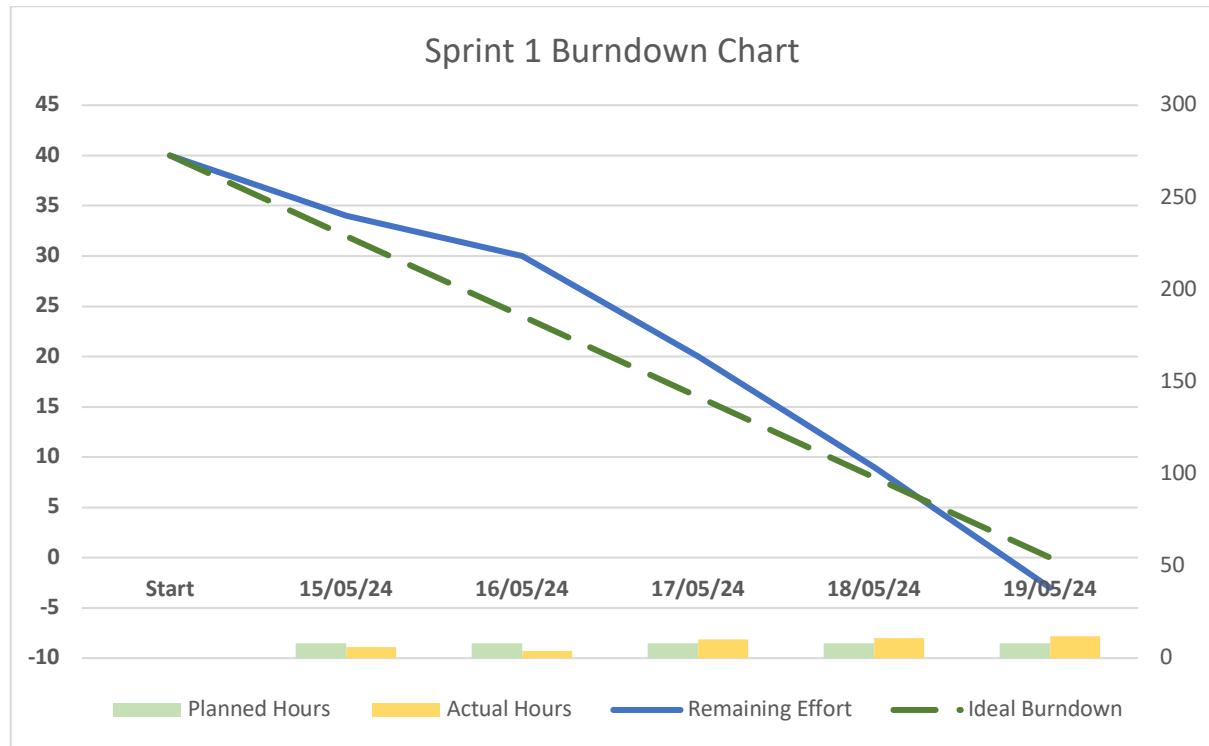
1.3 Review

The best practice is to have a good system design. With a good enough design after the Requirement Analysis process, the Development Team can accelerate quickly and complete the Functional Requirements on time and perform as expected.



1.4 Retrospective

Release Burndown Chart for Sprint 1.



Design takes more time than how long the team expected, at Day 1 and Day 2, the team couldn't find the right process to release design diagram. After Day 2, All the remaining days of the week, the team has to work overtime to release diagrams on time.

2. Sprint 2

Sprint Backlog:

Above requirements are Delivered in this sprint2

2.1. 2.1 Planning

In the Sprint 1, because the team did not manage time good enough, so they have to work overtime at some point while there are days that they did not take all 8 hours for works.

In Sprint 2, the team should take more attention for time management.

Task identified for Sprint 2:

TO DO	IN PROGRESS	DONE
- [] MRS-48 As a user, I want to see specific details of a movie on a new page when I click on it, so that I can decide if I want to rate it. (4 subtasks) RATING MANAGEMENT TO DO		
Create movie details UI in React. [] MRS-97	[] MRS-98	
Implement navigation to the movie details page when a movie is clicked. [] MRS-99	[] MRS-100	
Develop API endpoint in Django to retrieve movie details. [] MRS-101	[] MRS-102	
Fetch and display the specific details of the selected movie on the movie details page. [] MRS-103	[] MRS-104	
+ Create Issue		
- [] MRS-105 As a user, I want to give ratings to movies. (4 subtasks) RATING MANAGEMENT TO DO		
Create a rating submission form in React. Validate rating data on the frontend. [] MRS-106	[] MRS-107	
Validate rating data on the frontend. [] MRS-108	[] MRS-109	
Develop API endpoint in Django to handle rating submissions. [] MRS-110	[] MRS-111	
Store rating information in the database. [] MRS-112	[] MRS-113	
+ Create Issue		
- [] MRS-114 As a user, I want to view my given ratings. (3 subtasks) RATING MANAGEMENT TO DO		
Develop API endpoint in Django to retrieve user ratings. [] MRS-115	[] MRS-116	
Display user ratings in React. [] MRS-117	[] MRS-118	
+ Create Issue		
- [] MRS-119 As a user, I want to update my ratings. (4 subtasks) RATING MANAGEMENT TO DO		
Create a rating update form in React. [] MRS-120	[] MRS-121	
Validate updated rating data on the frontend. [] MRS-122	[] MRS-123	
Develop API endpoint in Django to handle rating updates. [] MRS-124	[] MRS-125	
Update rating information in the database. [] MRS-126	[] MRS-127	
+ Create Issue		
- [] MRS-128 As a user, I want to delete my ratings. (3 subtasks) RATING MANAGEMENT TO DO		
Create a delete rating button in React. [] MRS-129	[] MRS-130	
Develop API endpoint in Django to handle rating deletions. [] MRS-131	[] MRS-132	
Remove rating information from the database. [] MRS-133	[] MRS-134	
+ Create Issue		
- [] MRS-135 As a user, I want to add movies to my "rate later" list. (3 subtasks) USER MOVIE LIST MANAGEMENT TO DO		
Store movie information in the user-specific list in the database. [] MRS-136	[] MRS-137	
Develop API endpoint in Django to handle adding movies to the list. [] MRS-138	[] MRS-139	
Create an "add to rate later" button in React. [] MRS-140	[] MRS-141	
+ Create Issue		
- [] MRS-142 As a user, I want to view my "rate later" list. (2 subtasks) USER MOVIE LIST MANAGEMENT TO DO		
Develop API endpoint in Django to retrieve the "rate later" list. [] MRS-143	[] MRS-144	
Display the "rate later" list in React. [] MRS-145	[] MRS-146	
+ Create Issue		
- [] MRS-147 As a user, I want to remove movies from my "rate later" list. (3 subtasks) USER MOVIE LIST MANAGEMENT TO DO		
Develop API endpoint in Django to handle removing movies from the list. [] MRS-148	[] MRS-149	
Create a remove button in React for each movie in the user rate later list. [] MRS-150	[] MRS-151	
Update the user-specific list in the database. [] MRS-152	[] MRS-153	
+ Create Issue		

2.2. Implementation

During the implementation, the team also did Unit Test using Postman after completing a feature. We will demonstrate testing in the following system testing section.

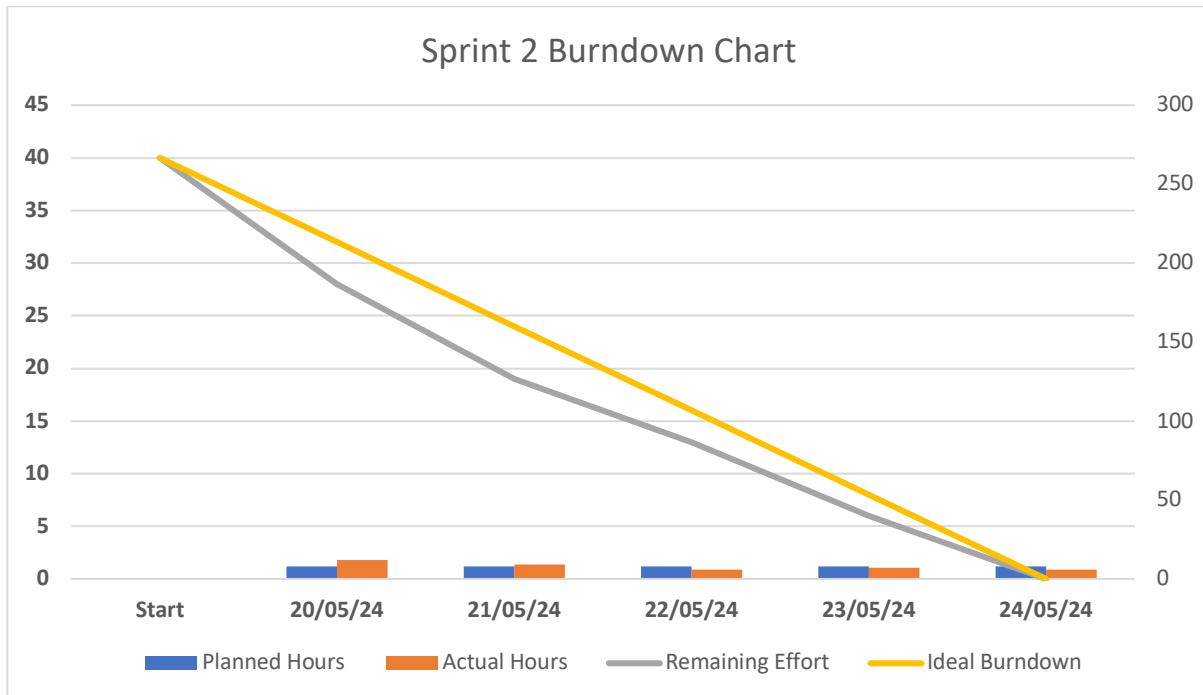
2.3 Review

Thanks to the design of the system in Sprint 1, the implementation in Sprint 2 can run more smoothly, there are no backlogs remaining.

TO DO 3	IN PROGRESS 1	DONE 47
<ul style="list-style-type: none"> + <input checked="" type="checkbox"/> MRS-95 As a user, I want to see specific details of a movie on a new page when I click on it, so that I can decide if I want to rate it. (4 subtasks) MOVIES MANAGEMENT TO DO <p>+ Create Issue</p>		<p>Move to Done</p> <ul style="list-style-type: none"> Implement navigation to the movie details page when a movie is clicked. MRS-48 = Develop API endpoint in Django to retrieve movie details. MRS-46 = Fetch and display the specific details of the selected movie on the movie details page. MRS-49 = Create a movie details UI in React. MRS-47 =
<ul style="list-style-type: none"> + <input checked="" type="checkbox"/> MRS-96 As a user, I want to give ratings to movies. (4 subtasks) RATING MANAGEMENT DONE <p>+ Create Issue</p>		<ul style="list-style-type: none"> Create a rating submission form in React. Validate rating data on the frontend. MRS-45 = Develop API endpoint in Django to handle rating submissions. MRS-43 = Validate rating data on the frontend. MRS-42 = Store rating information in the database. MRS-44 =
<ul style="list-style-type: none"> + <input checked="" type="checkbox"/> MRS-97 As a user, I want to view my given ratings. (2 subtasks) RATING MANAGEMENT DONE <p>+ Create Issue</p>		<ul style="list-style-type: none"> Develop API endpoint in Django to retrieve user ratings. MRS-45 = Display user ratings in React. MRS-46 =
<ul style="list-style-type: none"> + <input checked="" type="checkbox"/> MRS-98 As a user, I want to update my ratings. (4 subtasks) RATING MANAGEMENT DONE <p>+ Create Issue</p>		<ul style="list-style-type: none"> Create a rating update form in React. MRS-47 = Validate updated rating data on the frontend. MRS-48 = Develop API endpoint in Django to handle rating updates. MRS-49 = Update rating information in the database. MRS-46 =
<p>TO DO 3</p> <ul style="list-style-type: none"> - <input checked="" type="checkbox"/> MRS-99 As a user, I want to log in so that I can access my account. (4 subtasks) DONE <p>+ Create Issue</p>		<p>Move to Done</p> <ul style="list-style-type: none"> Create a login form in React. MRS-39 = Validate login credentials on the frontend. MRS-38 = Develop API endpoint in Django to handle user login. MRS-39 = Implement JWT token authentication. MRS-40 =
<ul style="list-style-type: none"> - <input checked="" type="checkbox"/> MRS-100 As a user, I want to log out so that I can end my session. (2 subtasks) TO DO <p>+ Create Issue</p>		<p>Move to Done</p> <ul style="list-style-type: none"> Create a logout button in React. MRS-39 = Invalidate the JWT token on the frontend. MRS-38 =
<ul style="list-style-type: none"> - <input checked="" type="checkbox"/> MRS-101 As a user, I want to register so that I can create an account. (4 subtasks) TO DO <p>+ Create Issue</p>		<p>Move to Done</p> <ul style="list-style-type: none"> Develop API endpoint in Django to handle user registration. MRS-39 = Validate registration data on the frontend. MRS-41 = Store user information securely in the database. MRS-34 = Create a registration form in React. MRS-36 =
<ul style="list-style-type: none"> - <input checked="" type="checkbox"/> MRS-102 As a user, I want to see a list of movies available to give the ratings. (3 subtasks) MOVIES MANAGEMENT DONE <p>+ Create Issue</p>		<p>Move to Done</p> <ul style="list-style-type: none"> Fetch and display the list of movies in the movie list UI. MRS-44 = Create a movie list UI in React. MRS-45 = Develop API endpoint in Django to retrieve the list of movies. MRS-42 =
<ul style="list-style-type: none"> - <input checked="" type="checkbox"/> MRS-103 As a user, I want to see specific details of a movie on a new page when I click on it, so that I can decide if I want to rate it. (4 subtasks) MOVIES MANAGEMENT TO DO 		<p>Move to Done</p>

<p>v [] MRS-17 As a user, I want to delete my ratings (3 subtasks) Rating Management DONE 11</p> <p>+ Create issue</p>	<p>Create a delete rating button in React.</p> <p>[] MRS-51 = 1</p> <p>Remove rating information from the database.</p> <p>[] MRS-53 = 1</p> <p>Develop API endpoint in Django to handle rating deletions.</p> <p>[] MRS-64 = 1</p>
<p>v [] MRS-19 As a user, I want to add movies to my "rate later" list. (3 subtasks) User Movie List Management DONE 11</p> <p>+ Create issue</p>	<p>Store movie information in the user-specific list in the database.</p> <p>[] MRS-58 = 1</p> <p>Develop API endpoint in Django to handle adding movies to the list.</p> <p>[] MRS-59 = 1</p> <p>Create an "add to rate later" button in React.</p> <p>[] MRS-66 = 1</p>
<p>v [] MRS-19 As a user, I want to view my "rate later" list. (2 subtasks) User Movie List Management DONE 11</p> <p>+ Create issue</p>	<p>Develop API endpoint in Django to retrieve the "rate later" list.</p> <p>[] MRS-64 = 1</p> <p>Display the "rate later" list in React.</p> <p>[] MRS-65 = 1</p>
<p>v [] MRS-20 As a user, I want to remove movies from my "rate later" list. (3 subtasks) User Movie List Management DONE 11</p> <p>+ Create issue</p>	<p>Develop API endpoint in Django to handle removing movies from the list.</p> <p>[] MRS-69 = 1</p> <p>Create a remove button in React for each movie in the user rate later list.</p> <p>[] MRS-69 = 1</p> <p>Update the user-specific list in the database.</p> <p>[] MRS-61 = 1</p>

2.4 Retrospective. Release Burndown Chart:



3. Sprint 3

Sprint Backlog:

Projects / Movie-Recommendation System / MRS board

Backlog

Only My Issues Recently Updated

Epic

- Issues without epic
- > User Authentication
- > Movies Management
- > User Movie List Management
- > Rating Management
- > Movies Recommendation Algorithm
- > Admin Authenticate Management
- > User Data Management (Admin)
- > Movies Data Management (Admin)
- + Create epic

MRS Sprint 1 15 May - 20 May (3 issues)

MRS Sprint 2 21 May - 26 Jun (9 issues)

MRS Sprint 3 27 May - 1 Jun (5 issues)

MRS-13 As a user, I want to give ratings to movies.

MRS-41 Create a rating submission form in React. Validate rating data on the frontend.

MRS-42 Validate rating data on the frontend.

MRS-43 Develop API endpoint in Django to handle rating submissions.

MRS-44 Store rating information in the database.

MRS-15 As a user, I want to view my given ratings.

MRS-45 Develop API endpoint in Django to retrieve user ratings.

MRS-46 Display user ratings in React.

MRS-16 As a user, I want to update my ratings.

MRS-47 Create a rating update form in React.

MRS-48 Validate updated rating data on the frontend.

MRS-49 Develop API endpoint in Django to handle rating updates.

MRS-50 Update rating information in the database.

MRS-17 As a user, I want to delete my ratings.

MRS-61 Create a delete rating button in React.

MRS-62 Develop API endpoint in Django to handle rating deletions.

MRS-63 Remove rating information from the database.

MRS-21 As a user, I want to receive movie recommendations based on my ratings.

MRS-62 Research and choose a recommendation algorithm

MRS-63 Implement the recommendation algorithm in Django.

MRS-64 Develop API endpoint in Django to fetch recommendations.

MRS-65 Create a recommendation feature button in React.

MRS-66 Display recommended movies in React.

RATING MANAGEMENT (5 issues)

MOVIES RECOMMENDATION (12 issues)

5 issues | Estimate: 45

Above requirements are Delivered in this sprint3

3.1. Planning

Task identification for Sprint 3:

TO DO 19	IN PROGRESS 1	DONE
<p>✓ MRS-22 As an admin, I want to log in securely so that I can manage the system. (4 subtasks) ADMIN AUTHENTICATE MANAG... TO DO 19</p> <p>Create an admin login form in React. MRS-67</p> <p>Validate admin login credentials on the frontend. MRS-68</p> <p>Develop API endpoint in Django to handle admin login. MRS-69</p> <p>Implement JWT token authentication for admins. MRS-70</p> <p>+ Create issue</p>		
<p>✓ MRS-21 As a user, I want to receive movie recommendations based on my ratings. (5 subtasks) MOVIES RECOMMENDATION AL... TO DO 19</p> <p>Research and choose a recommendation algorithm MRS-62</p> <p>Implement the recommendation algorithm in Django. MRS-63</p> <p>Develop API endpoint in Django to fetch recommendations. MRS-64</p> <p>Create a recommendation feature button in React. MRS-65</p> <p>Display recommended movies in React. MRS-66</p> <p>+ Create issue</p>		
<p>✓ MRS-12 As a user, I want to log out so that I can end my session. (2 subtasks) TO DO 19</p>		

TO DO 19	IN PROGRESS 1	DONE
<p>✓ MRS-21 As a user, I want to receive movie recommendations based on my ratings. (5 subtasks) MOVIES RECOMMENDATION AL... TO DO 19</p> <p>Display recommended movies in React. MRS-66</p> <p>+ Create issue</p>		
<p>✓ MRS-12 As a user, I want to log out so that I can end my session. (2 subtasks) TO DO 19</p> <p>Create a logout button in React. MRS-29</p> <p>Invalidate the JWT token on the frontend. MRS-30</p> <p>+ Create issue</p>		
<p>✓ MRS-10 As a user, I want to register so that I can create an account. (4 subtasks) TO DO 19</p> <p>Store user information securely in the database. MRS-34</p> <p>Develop API endpoint in Django to handle user registration. MRS-33</p> <p>Create a registration form in React. MRS-36</p> <p>Validate registration data on the frontend. MRS-31</p> <p>+ Create issue</p>		

TO DO 19	IN PROGRESS 1	DONE
<p>✓ MRS-21 As a user, I want to receive movie recommendations based on my ratings. (5 subtasks) MOVIES RECOMMENDATION AL... TO DO 19</p> <p>Display recommended movies in React. MRS-66 = 19</p> <p>+ Create issue</p>		
<p>✓ MRS-12 As a user, I want to log out so that I can end my session. (2 subtasks) TO DO 19</p> <p>Create a logout button in React. MRS-29 = 19</p> <p>Invalidate the JWT token on the frontend. MRS-30 = 19</p> <p>+ Create issue</p>		
<p>✓ MRS-10 As a user, I want to register so that I can create an account. (4 subtasks) TO DO 19</p> <p>Store user information securely in the database. MRS-34 = 19</p> <p>Develop API endpoint in Django to handle user registration. MRS-33 = 19</p> <p>Create a registration form in React. MRS-36 = 19</p> <p>Validate registration data on the frontend. MRS-31 = 19</p> <p>+ Create issue</p>		

3.2. Implementation

I reuse most of the parts that have been programmed from Scrum before, of course, it will have to be slightly modified due to changes in the Database and API structure in the Backend.

3.3. Review

Implementation in Sprint 3 is quite smooth because most of the backlogs in Sprint 3 can be reused from the previous Scrum. However, there is still an unfinished backlog.

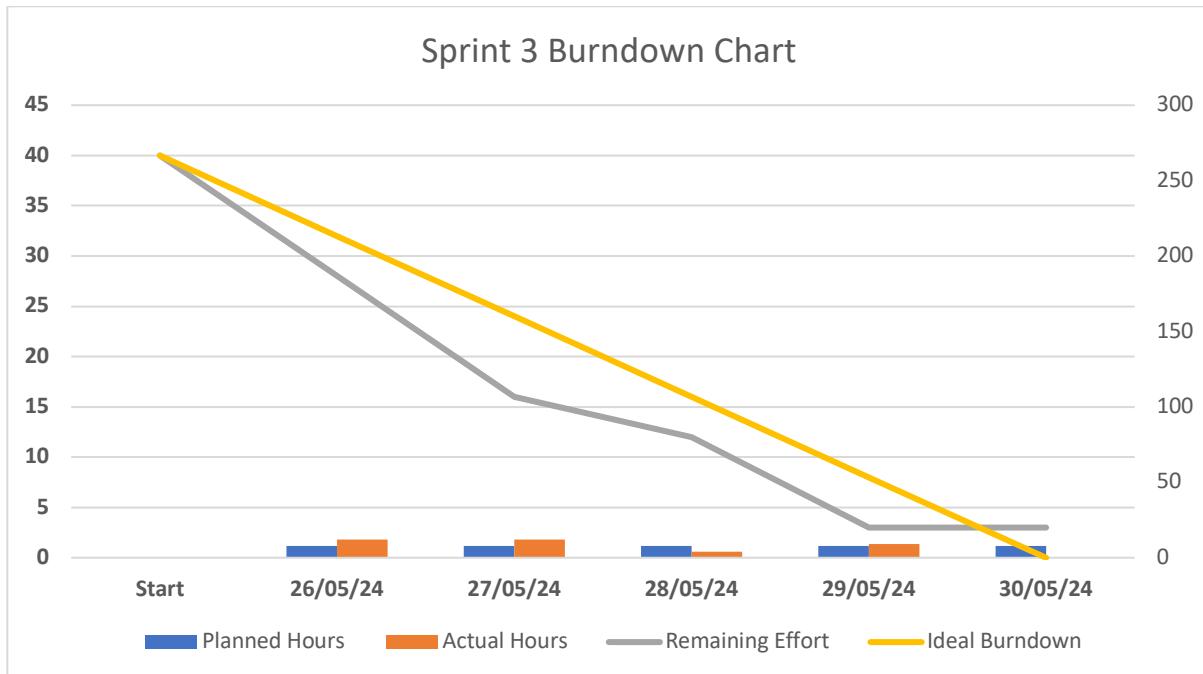
TO DO 1	IN PROGRESS 1	DONE 18
<p>✓ MRS-22 As an admin, I want to log in securely so that I can manage the system. (4 subtasks) ADMIN AUTHENTICATE MANA... DONE 19</p> <p>+ Create issue</p>	<p>+ Create issue</p>	<p>Create an admin login form in React. MRS-67 = 19</p> <p>Validate admin login credentials on the frontend. MRS-68 = 19</p> <p>Develop API endpoint in Django to handle admin login. MRS-69 = 19</p> <p>Implement JWT token authentication for admins. MRS-79 = 19</p>
<p>✓ MRS-21 As a user, I want to receive movie recommendations based on my ratings. (5 subtasks) MOVIES RECOMMENDATION AL... DONE 19</p> <p>+ Create issue</p>		<p>Research and choose a recommendation algorithm MRS-62 = 19</p> <p>Implement the recommendation algorithm in Django. MRS-63 = 19</p> <p>Develop API endpoint in Django to fetch recommendations. MRS-64 = 19</p> <p>Display recommended movies in React. MRS-66 = 19</p> <p>Create a recommendation feature button in React. MRS-65 = 19</p>

TO DO	IN PROGRESS	DONE
✓ MRS-22 As an admin, I want to log in securely so that I can manage the system. (4 subtasks)	ADMIN AUTHENTICATE MANAG... IN PROGRESS 1	DONE 18
✓ MRS-21 As a user, I want to receive movie recommendations based on my ratings. (5 subtasks)	MOVIES RECOMMENDATION AL... IN PROGRESS 1	DONE 18
+ Create issue	+ Create issue	Research and choose a recommendation algorithm. MRS-62 TP
		Implement the recommendation algorithm in Django. MRS-63 TP
		Develop API endpoint in Django to fetch recommendations. MRS-64 TP
		Display recommended movies in React. MRS-66 TP
		Create a recommendation feature button in React. MRS-65 TP
✓ MRS-12 As a user, I want to log out so that I can end my session. (2 subtasks)	TO DO 1	Move to Done
+ Create issue		Create a logout button in React. MRS-29 TP
		Invalidate the JWT token on the frontend. MRS-30 TP

TO DO 1	IN PROGRESS 1	DONE 18
<p>✓ MRS-21 As a user, I want to receive movie recommendations based on my ratings. (5 subtasks) MOVIES RECOMMENDATION AL... DONE 18</p> <p>+ Create issue</p>		Create a recommendation feature button in React. MRS-65
<p>✓ MRS-12 As a user, I want to log out so that I can end my session. (2 subtasks) TO DO 18</p> <p>+ Create issue</p>		<p>Move to Done</p> <p>Create a logout button in React. MRS-29</p> <p>Invalidate the JWT token on the frontend. MRS-30</p> <p>Move to Done</p>
<p>✓ MRS-10 As a user, I want to register so that I can create an account. (4 subtasks) TO DO 18</p> <p>+ Create issue</p>		<p>Store user information securely in the database. MRS-34</p> <p>Develop API endpoint in Django to handle user registration. MRS-33</p> <p>Create a registration form in React. MRS-36</p> <p>Validate registration data on the frontend. MRS-91</p> <p>+ Create issue</p>

3.4 Retrospective

Release Burndown Chart:



Looking at the Burndown Chart, we can see that the work was completed earlier than planned quite early in the beginning of the Sprint but by the end did not complete 100% of the Backlogs. Is there a problem with the team. This will be discussed in the Planning section of Sprint 4.

4. Sprint 4

Sprint Backlog:

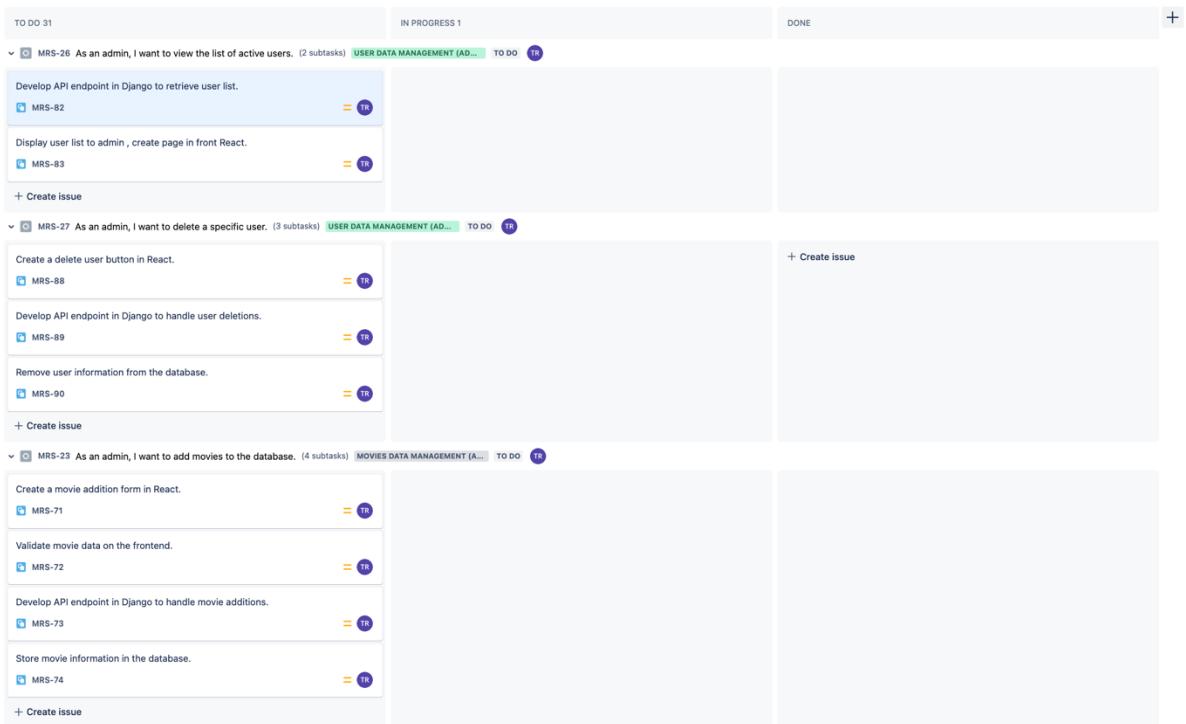
The screenshot shows the Jira backlog for Sprint 4. The left sidebar lists epics: User Authentication, Movies Management, User Movie List Management, Rating Management, Movies Recommendation Algorithm, Admin Authenticate Management, User Data Management (Admin), and Movies Data Management (Admin). The main area shows the backlog for MRS Sprint 4 (2 Jun - 8 Jun) with 48 issues. Issues are categorized by epic and priority. A grid on the right shows the status of each issue across columns: To Do, In Progress, Testing, and Done. The total estimate for the sprint is 48.

Above requirements are Delivered in this sprint4 .

4.1.Planning

Sprint 4 is the place with the most Backlogs in all 4 Sprints, but we still have to add Backlog not completed in Sprint 3. This backlog has Priority 4 so we'll still do it last if Sprint 4 doesn't have enough time.

Task identified for Sprint 4 as below.



TO DO 31

- MRS-25 As an admin, I want to update movie details. (4 subtasks) MOVIES DATA MANAGEMENT (ADMIN) TO DO TR
- Create a movie update form in React. MRS-75 = TR
- Validate updated movie data on the frontend. MRS-76 = TR
- Develop API endpoint in Django to handle movie updates. MRS-77 = TR
- Update movie information in the database. MRS-78 = TR
- + Create issue

IN PROGRESS 1

- + Create issue

DONE

TO DO 24

- MRS-24 As an admin, I want to delete movies from the database. (3 subtasks) MOVIES DATA MANAGEMENT (ADMIN) TO DO TR
- Create a delete movie button in React. MRS-79 = TR
- Develop API endpoint in Django to handle movie deletions. MRS-80 = TR
- Remove movie information from the database. MRS-81 = TR
- + Create issue

TO DO 31

- MRS-24 As an admin, I want to delete movies from the database. (3 subtasks) MOVIES DATA MANAGEMENT (ADMIN) TO DO TR
- MRS-28 As an admin, I want to update user information. (4 subtasks) MOVIES DATA MANAGEMENT (ADMIN) TO DO TR
- Create a user update form in React. MRS-84 = TR
- Validate updated user data on the frontend. MRS-85 = TR
- Develop API endpoint in Django to handle user updates. MRS-86 = TR
- Update user information in the database. MRS-87 = TR
- + Create issue

IN PROGRESS 1

- + Create issue

DONE

TO DO 12

- MRS-12 As a user, I want to log out so that I can end my session. (2 subtasks) TO DO TR
- Create a logout button in React. MRS-29 = TR
- Invalidate the JWT token on the frontend. MRS-30 = TR
- + Create issue

IN PROGRESS 0

DONE

Everything else (2 issues)

- User Data Management (Admin) USER DATA MANAGEMENT (ADMIN) MRS-8 = TR
- Movies Data Management (Admin) MOVIES DATA MANAGEMENT (ADMIN) MRS-7 = TR

+ Create issue

4.2 Implementation

In Sprint 4, there are 3 Functional Requirements implemented are:

FR.6 : Admin Authentication

FR.7: Admin Users Data Management

FR.8: Admin Movies Data management

FR.6 is pretty similar to FR.1: user Authentication so it will not take long time to implement.

The other two FRs (7 and 8) also don't take too much time because there's no need to focus too much on UX when the UI just needs to be usable, not too beautiful.

4.2 Review

While all the Backlogs planned for Sprint 4 have been completed, the Backlogs from Sprint 3 have not yet been completed.

The image displays two separate Kanban boards, each consisting of three columns: To Do, In Progress, and Done.

User Data Management (Top Board):

- To Do:** MRS-26 As an admin, I want to view the list of active users. (2 subtasks) [USER DATA MANAGEMENT (ADM)] TO DO 19
- In Progress:** None
- Done:** Display user list to admin , create page in front React. (MRS-83) = 19
Develop API endpoint in Django to retrieve user list. (MRS-82) = 19

MRS-27 As an admin, I want to delete a specific user. (3 subtasks) [USER DATA MANAGEMENT (ADM)] TO DO 19

- To Do:** Remove user information from the database. (MRS-99) = 19
- In Progress:** None
- Done:** Create a delete user button in React. (MRS-88) = 19
Develop API endpoint in Django to handle user deletions. (MRS-89) = 19

MRS-23 As an admin, I want to add movies to the database. (4 subtasks) [MOVIES DATA MANAGEMENT (ADM)] TO DO 19

- To Do:** None
- In Progress:** None
- Done:** Create a movie addition form in React. (MRS-71) = 19
Validate movie data on the frontend. (MRS-72) = 19
Store movie information in the database. (MRS-74) = 19
Develop API endpoint in Django to handle movie additions. (MRS-73) = 19

Movies Data Management (Bottom Board):

- To Do:** MRS-95 As an admin, I want to update movie details. (4 subtasks) [MOVIES DATA MANAGEMENT (ADM)] DONE 19
- In Progress:** None
- Done:** Create a movie update form in React. (MRS-75) = 19
Develop API endpoint in Django to handle movie updates. (MRS-79) = 19
Update movie information in the database. (MRS-78) = 19
Validate updated movie data on the frontend. (MRS-79) = 19

MRS-24 As an admin, I want to delete movies from the database. (2 subtasks) [MOVIES DATA MANAGEMENT (ADM)] TO DO 19

- To Do:** None
- In Progress:** None
- Done:** Develop API endpoint in Django to handle movie deletions. (MRS-89) = 19
Create a delete movie button in React. (MRS-79) = 19
Remove movie information from the database. (MRS-81) = 19

MRS-98 As an admin, I want to update user information. (4 subtasks) [MOVIES DATA MANAGEMENT (ADM)] DONE 19

- To Do:** None
- In Progress:** None
- Done:** Create a user update form in React. (MRS-84) = 19
Validate updated user data on the frontend. (MRS-89) = 19
Develop API endpoint in Django to handle user updates. (MRS-86) = 19
Update user information in the database. (MRS-87) = 19

TO DO 6

- MRS-25 As an admin, I want to update movie details. (1 subtask) MOVIES DATA MANAGEMENT [0] **TO DO** 1
- MRS-24 As an admin, I want to delete movies from the database. (0 subtasks) MOVIES DATA MANAGEMENT [0] **TO DO** 0
- + Create issue

IN PROGRESS 1

- MRS-25 As an admin, I want to update movie details. (1 subtask) MOVIES DATA MANAGEMENT [0] **IN PROGRESS** 1
- MRS-24 As an admin, I want to delete movies from the database. (0 subtasks) MOVIES DATA MANAGEMENT [0] **TO DO** 0
- + Create issue

DONE 25

- Validate updated movie data on the frontend. MRS-76 **DONE**
- Develop API endpoint in Django to handle movie deletions. MRS-98 **DONE**
- Create a delete movie button in React. MRS-79 **DONE**
- Remove movie information from the database. MRS-81 **DONE**

+ Create issue

Move to Done

MRS-25 As an admin, I want to update user information. (4 subtasks) MOVIES DATA MANAGEMENT [0] **DONE** 0

- + Create issue

Create a user update form in React. MRS-84 **DONE**

Validate updated user data on the frontend. MRS-85 **DONE**

Develop API endpoint in Django to handle user updates. MRS-86 **DONE**

Update user information in the database. MRS-87 **DONE**

+ Create issue

Move to Done

MRS-12 As a user, I want to log out so that I can end my session. (2 subtasks) TO DO 0

- + Create issue

Invalidate the JWT token on the frontend. MRS-38 **TO DO**

Create a logout button in React. MRS-29 **TO DO**

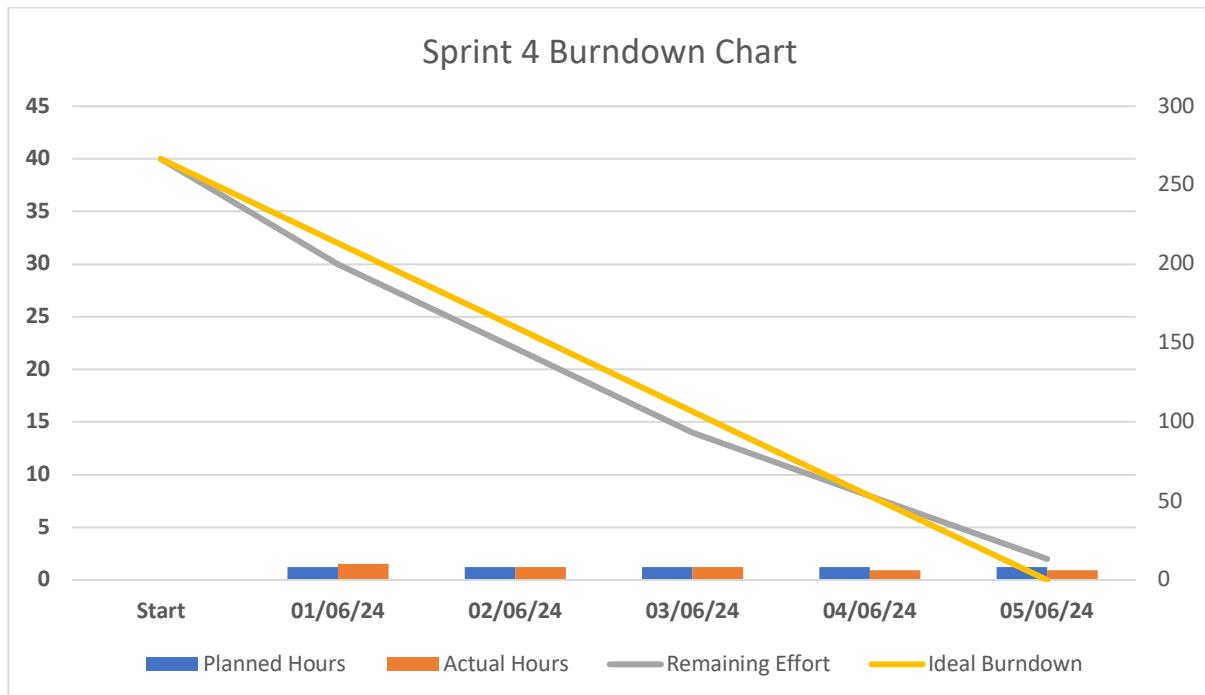
+ Create issue

Everything else (2 issues)

User Data Management (Admin) MRS-8 **TO DO**

Movies Data Management (Admin) MRS-7 **TO DO**

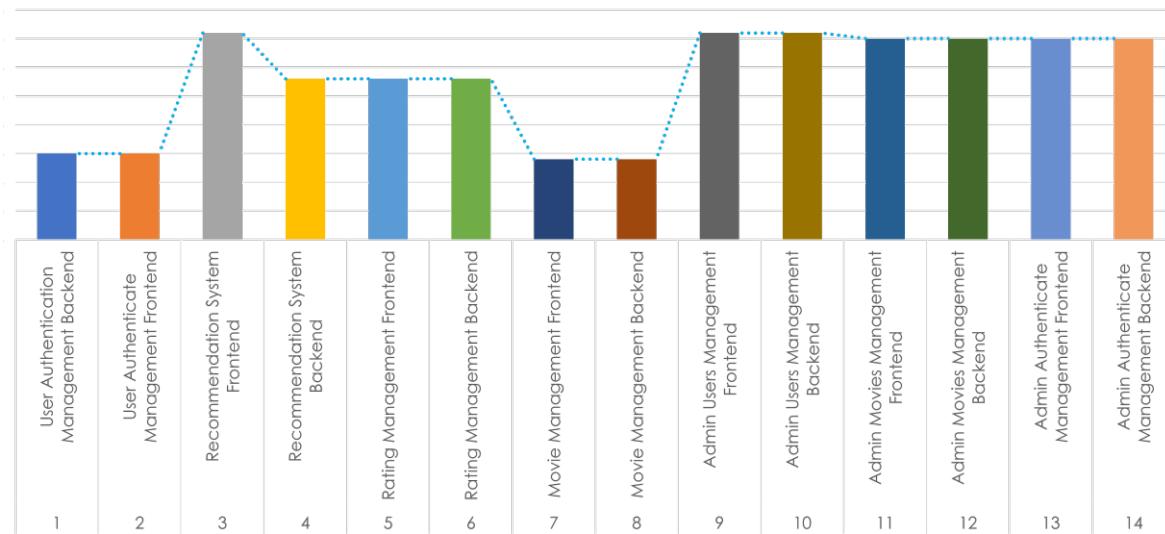
4.3 Retrospective Release Burndown Chart



Looking at the Burndown Chart, we can see that the team has found the right pace and process for the project. The Backlogs are completed quite close to the time, not too late or too early like the previous times. Finishing too early is also not so good as finishing too late.

VIII. Project closure

1. Release Burn Up Chart



From the Burn Up Chart, we can see that most of the product's features are completed in the last Sprint. There are 4 features completed in the early stage, 3 features completed in the middle stage, and 7 features completed in the final stage.

2. Verification test

Questions need answers in verification test:

- Are we building the product right?
- Does the software meet the specifications?

FR.1: User Authenticate Management:

FR.1.1: User Registration

Passed: Verify that all required fields are present, such as email, password, confirm password, ...

Passed: Verify that user can successfully register with valid information.

Passed: Checking if a user can register with valid data only in mandatory fields

Passed: A user get registered should be allowed to login to the application.

Passed: A user tries to register an existing email then an error message should get displayed.

Passed: Verify if the registration form has all the fields required by the requirement document.

Passed: Verify that a user can navigate to Login if they have an account.

FR.1.2: User Login

Passed: Verify that a user can log in with valid credentials.

Passed: Verify that a user cannot log in with invalid credentials.

Passed: Verify that a user cannot log in with invalid email and password.

Passed: Verify that Browser will save login session at local storage.

Passed: Verify that a user can navigate to Register if they don't have an account.

FR.1.3: User Verification

Passed: Verify that the user can successfully access the application with access token.

Passed: Verify that a logged in user doesn't need to login again if they refresh the web page.

Passed: Verify that a logged in user can navigate to another function of application.

FR.1.4: User Logout

Passed: Verify that at any function of the application, user can press logout.

Passed: Verify that after user press logout, all the information of user and token will be deleted.

FR.1.5: User Load

Passed: Verify that after user get verified, user information will be loaded.

Passed: Verify that user information will not be in local storage.

Passed: Verify that user information will be in Data Store as a state.

FR.2: Movie Management:

FR.2.1: User Search Movie

Passed: User cannot search for movie with less than 3 characters.

Passed: User can search for movie with more than 3 characters.

Passed: User will get fast response when call multiple searches at a short time.

Passed: User will get a list of movies with covers for a clear view.

FR.2.2: User Get Movie Details

Passed: User can get all information about a movie such as: Title, Genre.

Passed: User can see movie cover.

FR.3: Rating Management:

FR.3.1: User Create Rating

Passed: User will be allowed to create a new rating when they see a movie page.

Passed: After user rated a movie, their W, d in machine learning model get trained.

FR.3.2: User Read Rating

Passed: When a user sees a movie page, they can see their existing rating.

Passed: If user didn't rate that movie, it will display as blank star.

FR.3.3: User Update Rating

Passed: Verify if update rating function will only appear with existing rating.

Passed: Verify if update rating will change their W, d in machine learning model.

Passed: User should see updated rating value after server confirm.

FR.3.4: User Delete Rating

Passed: Verify if delete rating function will only appear with existing rating.

Passed: User should see blank rating star after server confirm rating is deleted.

FR.4: Recommendation System:

FR.4.1: User Get Suggested Movies

Passed: Verify if user can get a list of suggested movies.

Passed: Verify there is no rated movie in the list.

Passed: Evaluate model with training set, testing set.

FR.4.2: User Get Prediction for A Movie

Passed: API call for this function work.

FR.5: Admin Authenticate Management:

FR.5.1: Admin Login

Passed: Same as User Login

Passed: User credentials cannot be used to be login as admin.

Passed: Admin credentials cannot be used to be login as user.

FR.5.2: Admin Logout

Passed: When an admin press logout, there is no information left in the local storage.

Passed: When an admin press logout, there is no information left in the data store.

FR.6: Admin Users Management:

FR.6.1: Admin Get All Users

Passed: Admin can see all users in the database.

Passed: Click on the list will direct admin to see a specific user information.

FR.6.2: Admin Get User Details

Passed: Admin can see all information about user.

Passed: Admin cannot know user password nor see user encrypted password.

FR.6.3: Admin Delete User

Passed: Admin can only delete existing user.

Passed: Admin will access this feature in the specific user details.

FR.7: Admin Movies management:

FR.7.1: Admin Get All Movies

Passed: Admin can see all movies in the database.

Passed: Click on the list will direct admin to see a specific movie information.

FR.7.2: Admin Get Movie Details

Passed: Admin can see all information about movie.**FR.7.3:**

Admin Add New Movie

Passed: Admin can access this feature through FR.7.1.

Passed: After pressing a button, admin will turn into a page where they can fill the new movie.

FR.7.4: Admin Delete Movie

Passed: Admin can only delete existing movie.

Passed: Admin will access this feature in the specific movie details.

3. Validation test

Questions need answers in validation test:

- Are we building the right product?
- Does the software meet the user requirements?

Can this application suggest movies to the user? Accepted

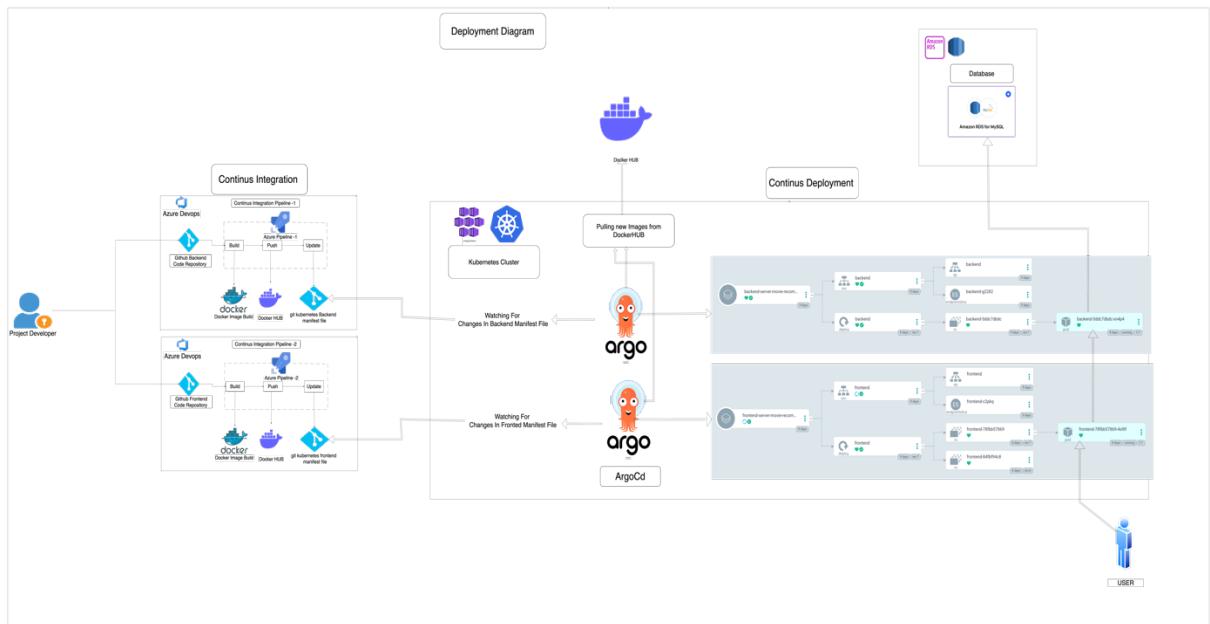
How does this application suggest movies to users? By a using machine learning model. Accepted.

To get more accurate suggestions, what does a user need to do? Give more ratings to more movies, update their rating or delete existing ratings.

To rate a movie, what does a user need to do? Create an account, login and access a movie page. How can a user access a movie page? Through Movie Search or Movie Suggestion.

Basically, all the requirements of the stakeholders for the application have been met and the validation test passed.

4 Operation plan and Deployment order



This deployment Diagram includes the implementation of a Continuous Integration and Continuous Deployment (CI/CD) pipelines using Azure DevOps and ArgoCD. The pipeline is designed for this movie recommendation application separates frontend and backend components. The system ensures that any updates to specific files of Frontend server folder and Backend server folder in the GitHub repository will automatically trigger the appropriate Azure DevOps pipeline, which will build docker image, push Docker images to Docker Hub, and update Kubernetes deployment configuration file with the new Docker image tag. ArgoCD then handles the continuous deployment which redeploy services in Azure Kubernetes Service (AKS) cluster Whenever there is change detected in frontend.yml and backend.yml files, which are the Kubernetes deployment configuration files in the GitHub Repository.

Target cost: free plan.

There are 3 components in this project:

- Database
- Backend
- Frontend

Frontend will only work when it gets response from Backend.

Backend will only work when it gets data from database to process. The deployment order will be as follows:

1. Database
2. Backend
3. Frontend

The cost plan will aim to free, details will be in the next sections.

4. Database Deployment

For this specific project i have used Aws service, which offer free RDS data storage for the student account.

The cost plan is free.

The provider is aws .

The region is us-east-1

Summary					
DB identifier	Status	Role	Engine	Recommendations	
database-sql	Available	Instance	MySQL Community	2 Informational	
CPU	Class db.t3.micro	Current activity 0 Connections	Region & AZ us-east-1a		
Connectivity & security					
Connectivity & security					
Endpoint & port	Networking	Security			
Endpoint database-sql.c9gugau8uqsk.us-east-1.rds.amazonaws.com	Availability Zone us-east-1a	VPC security groups default (sg-0b5d90e726ff8f28b) Active			
Port 3306	VPC vpc-08cfddc15a530c60f	Publicly accessible Yes			
	Subnet group default-vpc-08cfddc15a530c60f	Certificate authority Info rds-ca-rsa2048-g1			
	Subnets subnet-0da65fc01dd30db subnet-00feb372a7c5bc9ae subnet-04c8398acc9c85a12 subnet-0737d5e3b97a52031 subnet-020d804b521e8bee1 subnet-077dd0622c564064ec	Certificate authority date May 26, 2061, 01:34 (UTC+02:00) DB instance certificate expiration date May 25, 2025, 06:18 (UTC+02:00)			
	Network type IPv4				

i have set username and password as a authentication method to connect with the database ,

using the MySQL library and the database credentials the backend will connect with the with the database.

configured rules to connect to the database ,

Security group rules (3)		
Security group	Type	Rule
default (sg-0b5d90e726ff8f28b)	CIDR/IP - Inbound	0.0.0.0/0
default (sg-0b5d90e726ff8f28b)	EC2 Security Group - Inbound	sg-0b5d90e726ff8f28b
default (sg-0b5d90e726ff8f28b)	CIDR/IP - Outbound	0.0.0.0/0

The database has been set up with all necessary configurations, and all migrations have been successfully applied from the backend server to establish the storage structure and transfer the data.

Storage structure in DB tables,

```
Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_mv |
+-----+
| auth_group
| auth_group_permissions
| auth_permission
| auth_user
| auth_user_groups
| auth_user_user_permissions
| authtoken_token
| django_admin_log
| django_content_type
| django_migrations
| django_session
| ecomapp_movies
| ecomapp_mylist
| ecomapp_myrating
| token_blacklist_blacklistedtoken
| token_blacklist_outstandingtoken |
+-----+
16 rows in set (0.18 sec)

mysql> █
```

Movies data list,

5. Backend and Frontend setup for CICD

First, for Backend since the application is developed using Django , we need to setup requirement.txt file, this file will contain all the names of the packages and their respective versions that are needed for backend application. My requirement.txt file is:

```
1 pyodbc==5.0.1
2 PyMySQL==1.1.1
3 mssql-django==1.5
4 mysqlclient==2.0.3
5 sqlparse==0.3.1
6 urllib3==1.25.9
7 whitemoise==5.1.0
8 djangorestframework==3.15.1
9 djangorestframework-simplejwt==5.3.1
10 django-cors-headers==3.7.0
11 boto3==1.14.1
12 botocore==1.17.1
13 django==3.2.11
14 django-storages==1.9.1
15 docutils==0.15.2
16 gunicorn==20.0.4
17 idna==2.9
18 jmespath==0.10.0
19 numpy
20 pandas
21 Pillow==7.1.2
22 python-dateutil==2.8.1
23 pytz==2020.1
24 recommend==0.2.2
25 requests==2.23.0
26 s3transfer==0.3.3
27 scipy
28 six
```

With this requirement.txt file, a package-management system written in Python called

Pip will be installed automatically when we set up docker container.

Next, we need to set up the Dockerfile to containerize the application, Backend server Dockerfile will be as follows:

```
1 # Stage 1: Build stage
2 FROM python:3.8.18 as build
3
4 # Set the working directory to /movie_recommender
5 WORKDIR /movie_recommender
6
7 # Copy the requirements file to the working directory
8 COPY ./requirements.txt /movie_recommender/
9
10 # Install any needed packages specified in requirements.txt
11 RUN apt-get update && apt-get install -y gcc && \
12     pip install --no-cache-dir -r requirements.txt
13
14 # Copy the rest of the application files into the container at /movie_recommender
15 COPY . /movie_recommender
16
17 # Stage 2: Final stage
18 FROM python:3.8.18
19
20 # Set the working directory to /movie_recommender
21 WORKDIR /movie_recommender
22
23 # Copy only the installed dependencies from the build stage
24 COPY --from=build /usr/local/lib/python3.8/site-packages /usr/local/lib/python3.8/site-packages
25 COPY --from=build /usr/local/bin /usr/local/bin
26
27 # Copy the rest of the application files
28 COPY . /movie_recommender
29
30 # Define environment variable
31 ENV NAME World
32
33 # Expose port 5015 to the outside world
34 EXPOSE 5015
35
36 # Run app.py when the container launches
37 CMD ["python", "manage.py", "runserver", "0.0.0.0:5015"]
```

Configuring backend server setting.py file to establish connection with the database,

```
✓ DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mv',
        'USER': 'admin',
        'PASSWORD': 'adminuser',
        'HOST': 'database-sql.c9gugau8uqsk.us-east-1.rds.amazonaws.com',
        'PORT': '3306',
    }
}
```

In the same way we also need to include the backend server address in frontend server to connect and also we need to write the Dockerfile for the frontend application which is build using React .

Configuring frontend server package.json file and including backend server address to establish connection,

```
"name": "ecommerce",
"proxy": "http://backend:5015/",
"version": "0.1.0",
"private": true,
"dependencies": {
  "@testing-library/jest-dom": "^5.17.0",
  "@testing-library/react": "^13.4.0",
  "@testing-library/user-event": "^13.5.0",
  "axios": "^1.6.5",
  "react": "^18.2.0",
```

Dockerfile

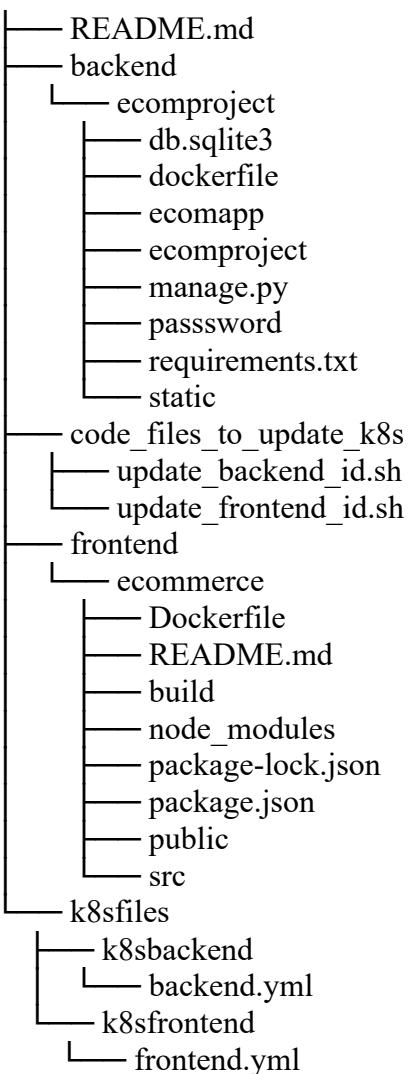
```
1  # Use a Node.js base image
2  FROM node:14-alpine
3
4  # Set working directory inside the container
5  WORKDIR /app
6
7  # Copy package.json and package-lock.json (if available)
8  COPY package*.json ./
9  |
10 # Install dependencies
11 RUN npm install
12
13 # Copy the rest of the application
14 COPY . .
15
16 # Build the React app
17 RUN npm run build
18
19 # Expose port 3000
20 EXPOSE 3000
21
22 # Command to run the application
23 CMD ["npm", "start"]
```

Now, after configuring the frontend and backend server files, the next step is to push to GitHub repository all the code files of frontend and backend. Additionally, including the Kubernetes folder, which contains the deployment configuration files (frontend.yml and backend.yml), as well as a custom code files folder that will update the Docker image tag in the Kubernetes deployment configuration file.

main ▾ 2 Branches 0 Tags Go to file Add file Code

 BandaTharun	Update README.md	4f3af83 · 4 days ago	141 Commits
 backend/ecomproject	new update of both frontend and backend files	last week	
 code_files_to_update_k8s	Update update_backend_id.sh	2 weeks ago	
 frontend/ecommerce	Update Dockerfile	last week	
 k8sfiles	Update frontend.yml	last week	
 README.md	Update README.md	4 days ago	

Repository Structure



The above repository can be accessible using this link:

https://github.com/BandaTharun/movie_recommendation_application-.git

Now, Next step is to setup (CICD) continues integration and continues deployment Pipelines.

IX. Deployment Plan.

1. Code Repository Structure:

1. The frontendserver and backendserver folders contain the source code for the frontend and backend applications respectively.
2. The Kubernetes folder contains Kubernetes deployment configuration files: frontend.yml and backend.yml.

2. CI Pipeline Configuration:

CI Pipeline Triggers:

1. Frontend pipeline triggers on updates to files in the frontendserver folder.
2. Backend pipeline triggers on updates to files in the backendserver folder.

Tasks of pipeline:

1. Build Docker Image.
2. Push Docker Image to Docker Hub.
3. Update Docker Image Tag in Kubernetes Deployment Files.

3. CD Pipeline Configuration:

1. ArgoCD is configured to watch the Kubernetes folder in the GitHub repository.
2. When changes are detected in frontend.yml or backend.yml, ArgoCD redeploys updated services to the AKS cluster.

Detailed Implementation plan:

CI Pipeline Setup in Azure DevOps

1. Creating the Pipelines:

1. Create two pipelines in Azure DevOps: frontend-pipeline and backend-pipeline.

The screenshot shows the Azure DevOps Pipelines interface. At the top, there are tabs for 'Recent', 'All', and 'Runs', with 'All' selected. On the right, there are buttons for 'New pipeline' and 'Filter pipelines'. Below the tabs, there's a 'New folder' button. The main area is titled 'All pipelines' and lists two pipelines:

- BandaTharun.backend_movie_recommendation_application - #20240525.3 • new update of both frontend and backend files
- BandaTharun.frontend_movie_recommendation_application #20240526.1 • Update Dockerfile

Pipeline -1 : BandaTharun.frontend_movie_recommendation_application

Tasks of the Pipeline :

#20240525.1 • Set up frontend CI with Azure Pipelines

Run new ⋮

This run will be cleaned up after 1 month based on your project settings.

Summary Code Coverage

Triggered by BandaTharun

Repository and version

Repository: BandaTharun/movie_recommendation_application

Branch: main ↗ bd4ebb23

Time started and elapsed

Started: 26 May at 00:24

Elapsed: 7m 17s

Related

0 work items

0 artifacts

Tests and coverage

Get started

View change

Stages Jobs

Build: 1 job completed 5m 16s

Push: 1 job completed 48s

Update k8s file: 1 job completed 35s

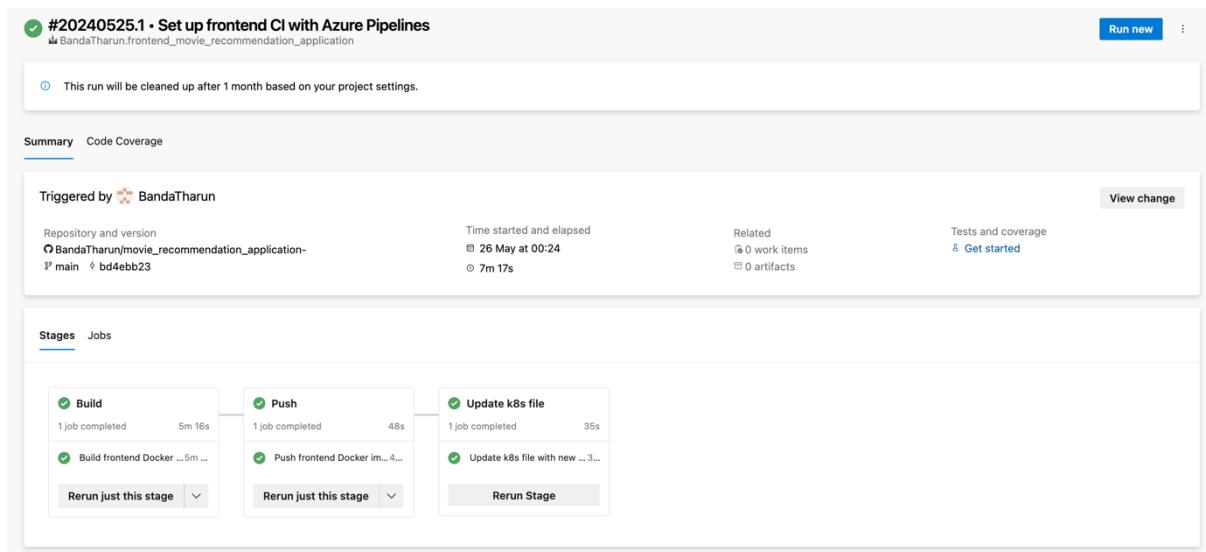
Build front-end Docker image 5m ...

Push front-end Docker image 4...

Update k8s file with new ... 3...

Rerun just this stage ⋮

Rerun Stage



Pipeline-1.Yml configuration code file that does the above tasks,

```

1
2
3
4     trigger:
5       paths:
6         include:
7           - frontend/ecomproject/*
8
9   resources:
10  - repo: self
11
12 variables:
13   tag: '$(Build.BuildId)'
14   frontendImageName: 'frontend_image'
15
16 pool:
17   name: 'agentpool-to-build-container'
18
19 stages:
20 - stage: Build
21   displayName: Build
22   jobs:
23     - job: BuildFrontend
24       displayName: Build frontend Docker image
25       steps:
26         Settings
27           - task: Docker@2
28             displayName: Login to Docker Hub
29             inputs:
30               command: login
31               containerRegistry: 'dockerregistryconnection'
32
33           - task: Docker@2
34             displayName: Build frontend Docker image
35             inputs:
36               containerRegistry: 'dockerregistryconnection'
37               repository: 'tharun9705/movie_recommendation_applications_repository'
38               command: build
39               Dockerfile: '$(Build.SourcesDirectory)/frontend/eCommerce/Dockerfile'
40               tags: '$(frontendImageName)$(tag)'
41
42 - stage: Push
43   displayName: Push
44   jobs:
45     - job: PushFrontend
46       displayName: Push frontend Docker image
47       steps:
48         Settings
49           - task: Docker@2
50             displayName: Push Docker image
51             inputs:
52               containerRegistry: 'dockerregistryconnection'
53               repository: 'tharun9705/movie_recommendation_applications_repository'
54               command: push
55               tags: '$(frontendImageName)$(tag)'
56
57 - stage: UpdateK8sFile
58   displayName: Update k8s file
59   jobs:
60     - job: UpdateK8s
61       displayName: Update k8s file with new Docker ID
62       steps:
63         Settings
64           - task: ShellScript@2
65             displayName: Update Kubernetes Configuration
66             inputs:
67               scriptPath: 'code_files_to_update_k8s/update_frontend_id.sh'
68               args: '$(frontendImageName)$(tag)'

```

Pipeline -2 : BandaTharun.backend_movie_recommendation_application-

Tasks of the Pipeline :

#20240523.3 • Update update_backend_id.sh

BandaTharun.backend_movie_recommendation_application-

This run will be cleaned up after 1 month based on your project settings.

Summary Code Coverage

Manually run by THARUN REDDY BANDA

Repository and version
BandaTharun/movie_recommendation_application-
main 8a8b0ed9

Time started and elapsed
23 May at 03:28
2m 23s

Related
0 work items
0 artifacts

Tests and coverage
Get started

View 8 changes

Stages Jobs

Build
1 job completed 51s
Build backend Docker im... 4...
Rerun just this stage

Push
1 job completed 26s
Push backend Docker im... 2...
Rerun just this stage

Update k8s file
1 job completed 33s
Update k8s file with new ... 2...
Rerun Stage

Pipeline-2.Yml configuration code file that does the above tasks ,

```

1
2
3 # Docker
4 # Build a Docker image
5 # https://docs.microsoft.com/azure/devops/pipelines/languages/docker
6
7 trigger:
8   paths:
9     include:
10    - backend/ecomproject/*
11
12 resources:
13   - repo: self
14
15 variables:
16   tag: '$(Build.BuildId)'
17   key: 'ghp_15Cep5CuGRN0h6Iule7L4qIwesigkN4T3STH'
18   backendImageName: 'backend_image'
19
20 pool:
21   name: 'agentpool-to-build-container'
22
23 stages:
24   - stage: Build
25     displayName: Build
26     jobs:
27       - job: BuildBackend
28         displayName: Build backend Docker image
29         steps:
30           Settings
31             - task: Docker@2
32               displayName: Login to Docker Hub
33               inputs:
34                 command: login
35                 containerRegistry: 'dockerregistryconnection'
36           Settings
37             - task: Docker@2
38               displayName: Build backend Docker image
39               inputs:
40                 containerRegistry: 'dockerregistryconnection'
41                 repository: 'tharun9705/movie_recommendation_applications_repository'
42                 command: 'build'
43                 Dockerfile: '$(Build.SourcesDirectory)/backend/ecomproject/dockerfile'
44                 tags: '$(backendImageName)$(tag)'
45
46   - stage: Push
47     displayName: Push
48     jobs:
49       - job: PushBackend
50         displayName: Push backend Docker image
51         steps:
52           Settings
53             - task: Docker@2
54               displayName: Push Docker image
55               inputs:
56                 containerRegistry: 'dockerregistryconnection'
57                 repository: 'tharun9705/movie_recommendation_applications_repository'
58                 command: 'push'
59                 tags: '$(backendImageName)$(tag)'
60
61   - stage: UpdateK8sFile
62     displayName: Update k8s file
63     jobs:
64       - job: UpdateK8s
65         displayName: Update k8s file with new Docker ID
66         steps:
67           Settings
68             - task: ShellScript@2
69               displayName: Update Kubernetes Configuration
70               inputs:
71                 scriptPath: 'code_files_to_update_k8s/update_backend_id.sh'
72                 args: '$(backendImageName)$(tag) $(key)'


```

CD Pipeline Setup AKS cluster :

The Continuous Deployment pipeline ensures that any changes to the Kubernetes folder in the **frontend.yml** or **backend.yml** files in the GitHub repository automatically trigger ArgoCD to deploy the latest versions of the frontend and backend applications to the AKS cluster.

Kubernetes deployment frontend.yml and backend.yml code file in K8's folder in GitHub .

frontend.yml file

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: frontend
5  spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        app: frontend
10   template:
11     metadata:
12       labels:
13         app: frontend
14     spec:
15       containers:
16         - name: frontend
17           image: tharun9705/movie_recommendation_applications_repository:frontend_image49
18         ports:
19           - containerPort: 3000
20   ---
21  apiVersion: v1
22  kind: Service
23  metadata:
24    name: frontend
25  spec:
26    selector:
27      app: frontend
28    ports:
29      - protocol: TCP
30        port: 3000
31        targetPort: 3000
32    type: LoadBalancer
```

Backend.yml file

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: backend
5  spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        app: backend
10   template:
11     metadata:
12       labels:
13         app: backend
14   spec:
15     containers:
16       - name: backend
17         image: tharun9705/movie_recommendation_applications_repository:backend_image47
18         ports:
19           - containerPort: 5015
20   ---
21  apiVersion: v1
22  kind: Service
23  metadata:
24    name: backend
25  spec:
26    selector:
27      app: backend
28    ports:
29      - protocol: TCP
30        port: 5015
31        targetPort: 5015
32    type: ClusterIP
```

1. Create AKS Cluster:

1. Setting up an AKS cluster in the Azure portal.

Overview

+ Create ▾ [Connect](#) [Start](#) [Stop](#) [Delete](#) [Refresh](#) [Open in mobile](#) [Give feedback](#) [JSON View](#)

Essentials

Resource group	Kubernetes version
movie_recommendation_devops	1.28.9
Status	API server address
Succeeded (Stopped)	mv-dns-name-w9et3t5i.hcp.italynorth.azmk8s.io
Subscription	Network type (plugin)
Azure for Students	kubenet
Location	Node pools
Italy North	1 node pool
Subscription ID	Container registries
b4f5090a-94aa-4cc8-8ae0-201978bf09dd	Attach a registry

[Tags \(edit\)](#)
[Add tags](#)

[Get started](#) [Properties](#) [Monitoring](#) [Capabilities \(5\)](#) [Recommendations \(4\)](#) [Tutorials](#)

Kubernetes services

Encryption type	API server address
Encryption at-rest with a platform-managed key	mv-dns-name-w9et3t5i.hcp.italynorth.azmk8s.io
Virtual node pools	Network type (plugin)
Not enabled	kubenet

Node pools

Node pools	Pod CIDR
1 node pool	10.244.0.0/16
Kubernetes versions	Service CIDR
1.28.9	10.0.0.0/16
Node sizes	DNS service IP
Standard_DS2_v2	10.0.0.10

Configuration

Kubernetes version	Docker bridge CIDR
1.28.9	-
Auto Upgrade Type	Network Policy
Patch	Calico
Automatic upgrade scheduler	Load balancer
Every week on Sunday (recommended)	Standard
Node security channel type	HTTP application routing
	Not enabled
	Private cluster
	Not enabled
	Authorized IP ranges
	Not enabled

Networking

API server address	Pod CIDR
mv-dns-name-w9et3t5i.hcp.italynorth.azmk8s.io	10.244.0.0/16
Network type (plugin)	Service CIDR
kubenet	10.0.0.0/16
Pod CIDR	DNS service IP
10.244.0.0/16	10.0.0.10
Service CIDR	Docker bridge CIDR
10.0.0.0/16	-
DNS service IP	Network Policy
10.0.0.10	Calico
Docker bridge CIDR	Load balancer
-	Standard
Network Policy	HTTP application routing
Calico	Not enabled
Load balancer	Private cluster
Standard	Not enabled
HTTP application routing	Authorized IP ranges
Not enabled	Not enabled

- Install ArgoCD in AKS: `kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml`

```
(base) THARUN@MacBook-Pro-M2 ~ % kubectl get deployments --all-namespaces=true
NAMESPACE      NAME                READY   UP-TO-DATE   AVAILABLE   AGE
argocd         argocd-applicationset-controller   1/1     1           1           9d
argocd         argocd-dex-server        1/1     1           1           9d
argocd         argocd-notifications-controller 1/1     1           1           9d
argocd         argocd-redis          1/1     1           1           9d
argocd         argocd-repo-server       1/1     1           1           9d
argocd         argocd-server          1/1     1           1           9d
calico-system  calico-kube-controllers 1/1     1           1          10d
calico-system  calico-typha          1/1     1           1          10d
default        backend              1/1     1           1           9d
default        frontend             1/1     1           1           9d
kube-system    coredns              2/2     2           2          10d
kube-system    coredns-autoscaler    1/1     1           1          10d
kube-system    konnectivity-agent    2/2     2           2          10d
kube-system    metrics-server       2/2     2           2          10d
tigera-operator tigera-operator     1/1     1           1          10d
(base) THARUN@MacBook-Pro-M2 ~ %
```

3. Configure ArgoCD to Monitor GitHub Repository the frontend.yml and backend.yml deployment files.

1. Create ArgoCD Application for Backend:

ArgoCD YAML configuration file `backend-app.yaml` that will Monitor k8's deployment file `backend.yml` in GitHub Repository.

```

1
2
3  apiVersion: argoproj.io/v1alpha1
4  kind: Application
5  metadata:
6    name: backend-app
7    namespace: argocd
8  spec:
9    project: default
10   source:
11     repoURL: 'https://github.com/BandaTharun/movie\_recommendation\_application-.git'
12     path: k8sfiles/k8sbackend
13     targetRevision: HEAD
14   destination:
15     server: 'https://kubernetes.default.svc'
16     namespace: default
17   syncPolicy:
18     automated:
19       prune: true
20       selfHeal: true
21
22
23 |
```

2. Create ArgoCD Application for Frontend:

ArgoCD YAML configuration file `frontend-app.yaml` that will Monitor k8's deployment file `frontend.yml` in GitHub Repository.

```

1
2  apiVersion: argoproj.io/v1alpha1
3  kind: Application
4  metadata:
5    name: frontend-app
6    namespace: argocd
7  spec:
8    project: default
9    source:
10      repoURL: 'https://github.com/BandaTharun/movie_recommendation_application-.git'
11      path: k8sfiles/k8sfrontend
12      targetRevision: HEAD
13    destination:
14      server: 'https://kubernetes.default.svc'
15      namespace: default
16    syncPolicy:
17      automated:
18        prune: true
19        selfHeal: true
20
21
22

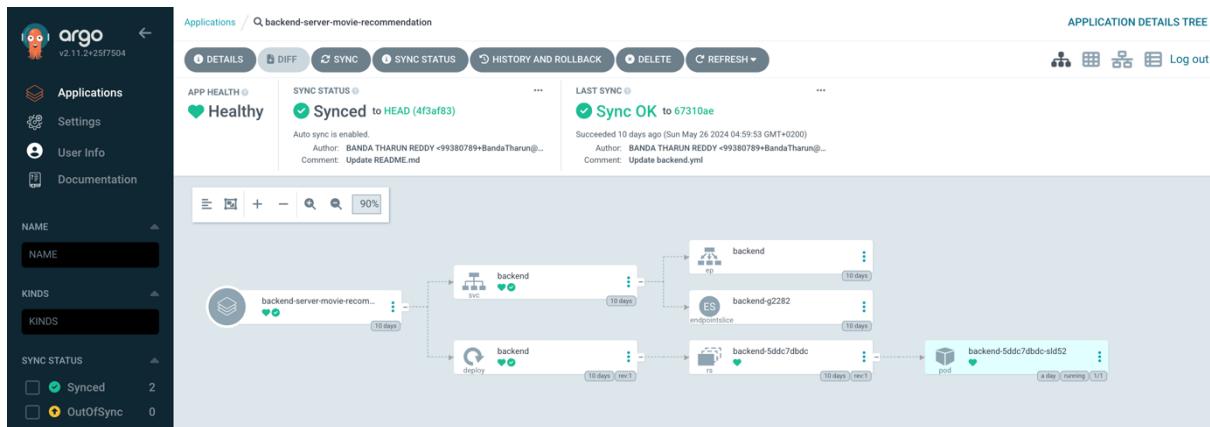
```

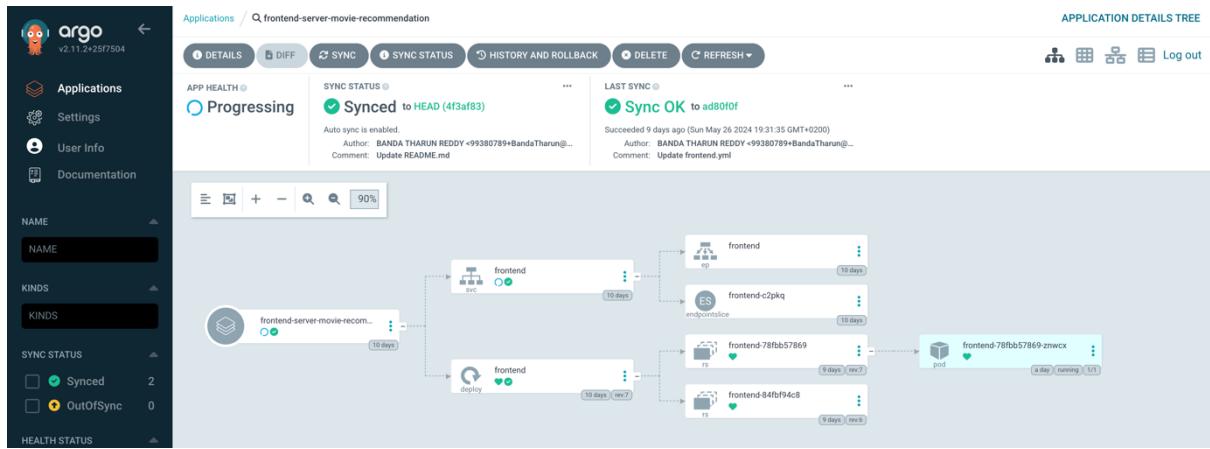
3. Apply the Application Definitions:

```
kubectl apply -f frontend-app.yaml
kubectl apply -f backend-app.yaml
```

After applying, The ArgoCD will continuously monitors **frontend.yml** and **backend.yml** in the GitHub repository, If any changes are detected, ArgoCD automatically pulls the updated deployment configuration file, Applies the new configuration to the AKS cluster and Redeploys the respective frontend or backend services with the new Docker images.

This is how Argocd Deploys, Monitors and Sync the new code changes to the cluster.





By following all the configuring setups and steps , The CI/CD pipeline using Azure DevOps and ArgoCD provides a robust solution for automated deployments of the movie recommendation application. By configuring the pipelines to trigger on specific folder updates, build and push Docker images, and update Kubernetes deployment files, we ensure efficient and reliable deployments to the AKS cluster. This setup reduces manual intervention, enhances deployment speed, and ensures consistency across the development and production environments.

X. Recommendation System Algorithm

Collaborative Filtering (Recommender Algorithm) have been used in this project .

Collaborative Filtering (CF) is a popular recommendation algorithm used in recommender systems to suggest items (such as movies, products, or music) to users based on the preferences and behaviors of other users. It operates under the assumption that if two users have had similar preferences in the past, they will have similar preferences in the future.

There are two main types of collaborative filtering:

user-based collaborative filtering **and** item-based collaborative filtering.

User-Based Collaborative Filtering: User-based collaborative filtering makes recommendations based on the similarity between users. The algorithm works as follows:

- Identify Similar Users: Find users who have similar tastes to the target user. This can be done using similarity measures such as Pearson correlation or cosine similarity
- Generate Recommendations: Recommend items that similar users have liked but the target user has not yet seen or rated.

Example:

If User A and User B have both rated a set of movies similarly, and User B liked a movie that User A hasn't seen yet, then that movie might be recommended to User A.

2. Item-Based Collaborative Filtering

Item-based collaborative filtering makes recommendations based on the similarity between items. The algorithm works as follows:

- Identify Similar Items: Find items that are similar to items the target user has liked or interacted with. This is done by looking at users' ratings and interactions with these items.
- Generate Recommendations: Recommend items that are similar to the items the user has liked in the past.

Example:

If a user likes a particular movie, the algorithm will recommend other movies that are similar to that one based on the ratings and interactions from other users.

Item-Based Collaborative Filtering algorithm have been implemented to generate movie recommendations.

Algorithm code:

```

# Recommendation Algorithm
def recommend(request):

    if not request.user.is_authenticated:
        return redirect("login")
    if not request.user.is_active:
        raise Http404

    movie_rating=pd.DataFrame(list(Myrating.objects.all().values()))

    new_user=movie_rating.user_id.unique().shape[0]
    current_user_id= request.user.id
    # if new user not rated any movie
    if current_user_id>new_user:
        movie=Movie.objects.get(id=19)
        q=Myrating(user=request.user,movie=movie,rating=0)
        q.save()

    userRatings = movie_rating.pivot_table(index=['user_id'],columns=['movie_id'],values='rating')
    userRatings = userRatings.fillna(0, axis=1)
    corrMatrix = userRatings.corr(method='pearson')

    user = pd.DataFrame(list(Myrating.objects.filter(user=request.user).values())).drop(['user_id','id'],axis=1)
    user_filtered = [tuple(x) for x in user.values]
    movie_id_watched = [each[0] for each in user_filtered]

    similar_movies = pd.DataFrame()
    for movie, rating in user_filtered:
        similar_movies = similar_movies.append(get_similar(movie, rating, corrMatrix), ignore_index = True)

    movies_id = list(similar_movies.sum().sort_values(ascending=False).index)
    movies_id_recommend = [each for each in movies_id if each not in movie_id_watched]
    preserved = Case(*[When(pk=pk, then=pos) for pos, pk in enumerate(movies_id_recommend)])
    movie_list=list(Movie.objects.filter(id__in = movies_id_recommend).order_by(preserved)[:10])

    context = {'movie_list': movie_list}
    return render(request, 'recommend/recommend.html', context)

```

Here's a breakdown of how the algorithm works in this context:

4. Data Preparation:

1. The ratings data is loaded into a DataFrame (movie rating), with each rating associated with a user and a movie.
2. The unique count of users and the current user's ID are determined. If the user is new and hasn't rated any movies, they are assigned a default rating for a specific movie.

3. Pivot Table Creation:

1. A pivot table (userRatings) is created, where the rows represent users, the columns represent movies, and the values are the ratings. Missing values are filled with 0.

4. Correlation Matrix Calculation:

1. The correlation matrix (corrMatrix) is calculated using the Pearson correlation method. This matrix shows the similarity between different movies based on user ratings.

5. User's Ratings Extraction:

1. The ratings of the current user are extracted into a DataFrame (user), and the list of movies the user has watched and rated is obtained.

6. Finding Similar Movies:

```
# To get similar movies based on user rating
def get_similar(movie_name,rating,corrMatrix):
    similar_ratings = corrMatrix[movie_name]*(rating-2.5)
    similar_ratings = similar_ratings.sort_values(ascending=False)
    return similar_ratings
```

For each movie the user has rated, a list of similar movies is found using the `get_similar` function. The similarities are aggregated into the `similar_movies` DataFrame.

7. Recommendations Generation:

1. The movies are summed and sorted to find the most similar movies to generate recommendations. Movies the user has already watched are filtered out.
2. The list of recommended movie IDs is used to query the Movie database, and the top 10 recommendations are retrieved and ordered.

Production Screenshots:

User

Signup

User Name

Email

Password

Signup



If You Are the Registered User click Here

Login

User Name

Password

Login



If You are a New User click on here

Movies

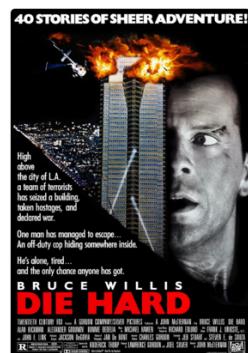
[12 Angry Men](#)[12 Years a Slave](#)[A Beautiful Mind](#)[Avatar](#)

12 Angry Men

12 Years a Slave

A Beautiful Mind

Avatar

[Avengers: Infinity War](#)[Avengers: Endgame](#)[Back to the Future](#)[Batman Begins](#)[Casino Royale](#)[Coco](#)[The Dark Knight Rises](#)[Die Hard](#)

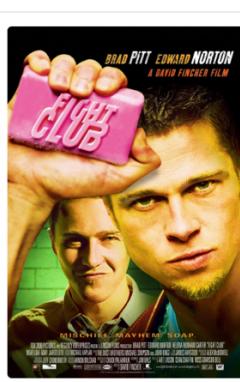
Casino Royale

Coco

The Dark Knight Rises

Die Hard

The Dark Knight Rises

[Django Unchained](#)[Drive](#)[Fight Club"](#)[Forrest Gump](#)

Django Unchained

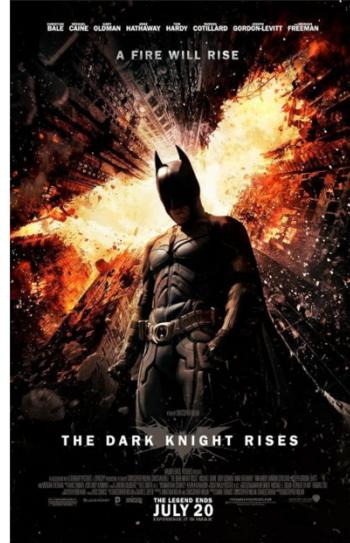
Drive

Fight Club"

Forrest Gump

[Go Back](#)

The Dark Knight Rises



The Dark Knight Rises

Overview: Following the death of District Attorney Harvey Dent, Batman assumes responsibility for Dent's crimes to protect the late attorney's reputation and is subsequently hunted by the Gotham City Police Department. Eight years later, Batman encounters the mysterious Selina Kyle and the villainous Bane, a new terrorist leader who overwhelms Gotham's finest. The Dark Knight resurfaces to protect a city that has branded him an enemy.

original_language: en

Movie_Genres: Action, Adventure"

Release Date: 2012-07-17

Popularity: 90.231

Vote Average: 7.78

Vote Count: 22161

Watch trailer: [Trailer](#)

Give Your Personal Rating

★★★★★

[Submit Rating](#)

Add to My List

Rating added successfully.

[Go Back](#)

The Dark Knight Rises



The Dark Knight Rises

Overview: Following the death of District Attorney Harvey Dent, Batman assumes responsibility for Dent's crimes to protect the late attorney's reputation and is subsequently hunted by the Gotham City Police Department. Eight years later, Batman encounters the mysterious Selina Kyle and the villainous Bane, a new terrorist leader who overwhelms Gotham's finest. The Dark Knight resurfaces to protect a city that has branded him an enemy.

original_language: en

Movie_Genres: Action, Adventure"

Release Date: 2012-07-17

Popularity: 90.231

Vote Average: 7.78

Vote Count: 22161

Watch trailer: [Trailer](#)

Give Your Personal Rating

★★★★★

[Submit Rating](#)

Add to My List

MyRatings



The Dark Knight Rises

Click on the
Checkbox To Delete
the movie from
MyRatings

★★★★★

[Update Rating](#)

Recommendations



Back to the
Future



Coco



Intersteller



Memento



Shutter Island



The Lion King

Movie-Recommendation Home

Avengers: Infinity War added to tharun444's list.

Go Back

Avengers: Infinity War



Avengers:
Infinity War

Overview: As the Avengers and their allies have continued to protect the world, the stakes have never been too large for any one hero to handle, a new threat has emerged from the cosmic shadows: Thanos. A despot of intergalactic infinity, his goal is to collect all six Infinity Stones, artifacts of unimaginable power, and use them to further his twisted plan of total annihilation. Everything the Avengers have fought for has led up to this moment - the fate of Earth and existence itself has never been more uncertain.

originalLanguage: en
Movie_Genres: Action, Adventure, Sci-Fi
Release Date: 2018-04-25
Popularity: 267265
Vote Average: 8.248
Vote Count: 28951
Watch trailer: [Trailer](#)

Give Your Personal Rating

★★★★★

Add to My List

Movie-Recommendation Home

GetRecommendation MyRatings MyList Login Signup Logout

MyList



Avengers:
Infinity War

Click on the Checkbox To Delete the Movie from MyList

Admin

Movie-Recommendation Admin Page [Main Page](#)

Manage Movies Manage Users Home Website Page Logout

Admin Login



Movie-Recommendation Admin Page [Main Page](#)

Manage Movies Manage Users Home Website Page Logout

WELCOME TO ADMIN PAGE

[Manage Movies](#)
[Manage Users](#)

X
© 2020 Movie-Recommendation

Movie-Recommendation Admin Page [Main Page](#)

Manage Movies Manage Users Home Website Page Logout

Manage Movies

[Add Movie](#)

#	NAME	GENRE	URL	ACTION
1	12 Angry Men	Crime, Drama	b5	UPDATE DELETE
2	12 Years a Slave	Drama, History	M	UPDATE DELETE
3	A Beautiful Mind	Biography, Drama	b8	UPDATE DELETE
4	Avatar	Action, Adventure, Fantasy	images	UPDATE DELETE
5	Avengers: Infinity War	Action, Adventure, Sci-Fi	images	UPDATE DELETE
6	Avengers: Endgame	Action, Adventure, Sci-Fi	OS	UPDATE DELETE
7	Back to the Future	Adventure, Comedy, Sci-Fi	M	UPDATE DELETE
8	Batman Begins	Action, Adventure*	I	UPDATE DELETE
9	Casino Royale	Action, Thriller	M	UPDATE DELETE
10	Coco	Animation, Adventure	xlarge	UPDATE DELETE
11	The Dark Knight Rises	Action, Adventure*	M	UPDATE DELETE
12	Die Hard	Action, Thriller	M	UPDATE DELETE
13	Django Unchained	Drama	I	UPDATE DELETE
14	Drive	Crime, Drama	M	UPDATE DELETE

Movie-Recommendation Admin Page [Main Page](#)

[Manage Movies](#) [Manage Users](#) [Home Website Page](#) [Logout](#)

#	NAME	GENRE	URL	ACTION
1	12 Angry Men	Action, Drama	12AngryMen	UPDATE DELETE
2	12 Years a Slave	Image URL amazon.com/images/I/714DXNB4++L_AC_UF8941000QL80_.jpg	12YearsASlave	UPDATE DELETE
3	A Beautiful Mind	Biography, Drama	ABeautifulMind	UPDATE DELETE
4	Avatar	Action, Adventure, Fantasy	Avatar	UPDATE DELETE
5	Avengers: Infinity War	Action, Adventure, Sci-Fi	AvengersInfinityWar	UPDATE DELETE
6	Avengers: Endgame	Action, Adventure, Sci-Fi*	AvengersEndgame	UPDATE DELETE
7	Back to the Future	Adventure, Comedy, Sci-Fi	BacktotheFuture	UPDATE DELETE

Movie-Recommendation Admin Page [Main Page](#)

[Manage Movies](#) [Manage Users](#) [Home Website Page](#) [Logout](#)

Manage Movies

Movie added successfully.

[Add Movie](#)

#	NAME	GENRE	URL	ACTION
1	12 Angry Men	Crime, Drama	12AngryMen	UPDATE DELETE
2	12 Years a Slave	Drama, History	12YearsASlave	UPDATE DELETE
3	A Beautiful Mind	Biography, Drama	ABeautifulMind	UPDATE DELETE
4	Avatar	Action, Adventure, Fantasy	Avatar	UPDATE DELETE
5	Avengers: Infinity War	Action, Adventure, Sci-Fi	AvengersInfinityWar	UPDATE DELETE
6	Avengers: Endgame	Action, Adventure, Sci-Fi*	AvengersEndgame	UPDATE DELETE
7	Back to the Future	Adventure, Comedy, Sci-Fi	BacktotheFuture	UPDATE DELETE
8	Batman Begins	Action, Adventure*	BatmanBegins	UPDATE DELETE
9	Casino Royale	Action, Thriller	CasinoRoyale	UPDATE DELETE
10	Coco	Animation, Adventure	Coco	UPDATE DELETE
11	The Dark Knight Rises	Action, Adventure*	TheDarkKnightRises	UPDATE DELETE
12	Die Hard	Action, Thriller	DieHard	UPDATE DELETE
13	Django Unchained	Drama	DjangoUnchained	UPDATE DELETE

34	The Avengers*	Action, Sci-Fi	TheAvengers	UPDATE DELETE
35	The Lion King	Animation, Comedy	TheLionKing	UPDATE DELETE
36	The Pianist	Biography, Drama	ThePianist	UPDATE DELETE
37	The Shawshank Redemption	Drama	TheShawshankRedemption	UPDATE DELETE
38	The Godfather*	Crime, Drama	TheGodfather	UPDATE DELETE
39	The Truman Show	comedy, Drama, Sci-Fi	TheTrumanShow	UPDATE DELETE
40	The Wolf of Wall Street	Crime, Drama, Comedy	TheWolfofWallStreet	UPDATE DELETE
41	Toy Story	Animation, Comedy	ToyStory	UPDATE DELETE
42	Godzilla x Kong: The New Empire	Action, Drama	GodzillaxKongTheNewEmpire	UPDATE DELETE



© 2020 Movie-Recommendation

Manage Movies					
Movie deleted successfully.					
#	NAME	GENRE	URL	ACTION	
1	12 Angry Men	Crime, Drama	12	UPDATE	DELETE
2	12 Years a Slave	Drama, History	M	UPDATE	DELETE
3	A Beautiful Mind	Biography, Drama	B	UPDATE	DELETE
4	Avatar	Action, Adventure, Fantasy	A	UPDATE	DELETE
5	Avengers: Infinity War	Action, Adventure, Sci-Fi	I	UPDATE	DELETE
6	Avengers: Endgame	Action, Adventure, Sci-Fi*	E	UPDATE	DELETE
7	Back to the Future	Adventure, Comedy, Sci-Fi	M	UPDATE	DELETE
8	Batman Begins	Action, Adventure*	B	UPDATE	DELETE
9	Casino Royale	Action, Thriller	M	UPDATE	DELETE
10	Coco	Animation, Adventure	A	UPDATE	DELETE
11	The Dark Knight Rises	Action, Adventure*	M	UPDATE	DELETE
12	Die Hard	Action, Thriller	D	UPDATE	DELETE
13	Django Unchained	Drama	J	UPDATE	DELETE

Manage Users							
#	ID	USERNAME	EMAIL	IS_STAFF	IS_SUPERUSER	ACTION	
1	2	Tharun	tharunreddy9705@gmail.com	Yes	Yes	DELETE USER	
2	3	tharun11	tharun@gmail.com	Yes	Yes	DELETE USER	
3	10	tharun3333	tharunreddy9705345@gmail.com	No	No	DELETE USER	
4	11	tharun444	tharun67@yahoo.com	No	No	DELETE USER	

Manage Users							
#	ID	USERNAME	EMAIL	IS_STAFF	IS_SUPERUSER	ACTION	
1	2	Tharun	tharunreddy9705@gmail.com	Yes	Yes	DELETE USER	
2	3	tharun11	tharun@gmail.com	Yes	Yes	DELETE USER	
3	10	tharun3333	tharunreddy9705345@gmail.com	No	No	DELETE USER	
4	11	tharun444	tharun67@yahoo.com	No	No	DELETE USER	

Manage Users

User deleted successfully.

#	ID	USERNAME	EMAIL	IS_STAFF	IS_SUPERUSER	ACTION
1	2	Tharun	tharunreddy9705@gmail.com	Yes	Yes	DELETE USER
2	3	tharun11	tharun@gmail.com	Yes	Yes	DELETE USER
3	11	tharun444	tharun5@yahoo.com	No	No	DELETE USER

