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.

“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

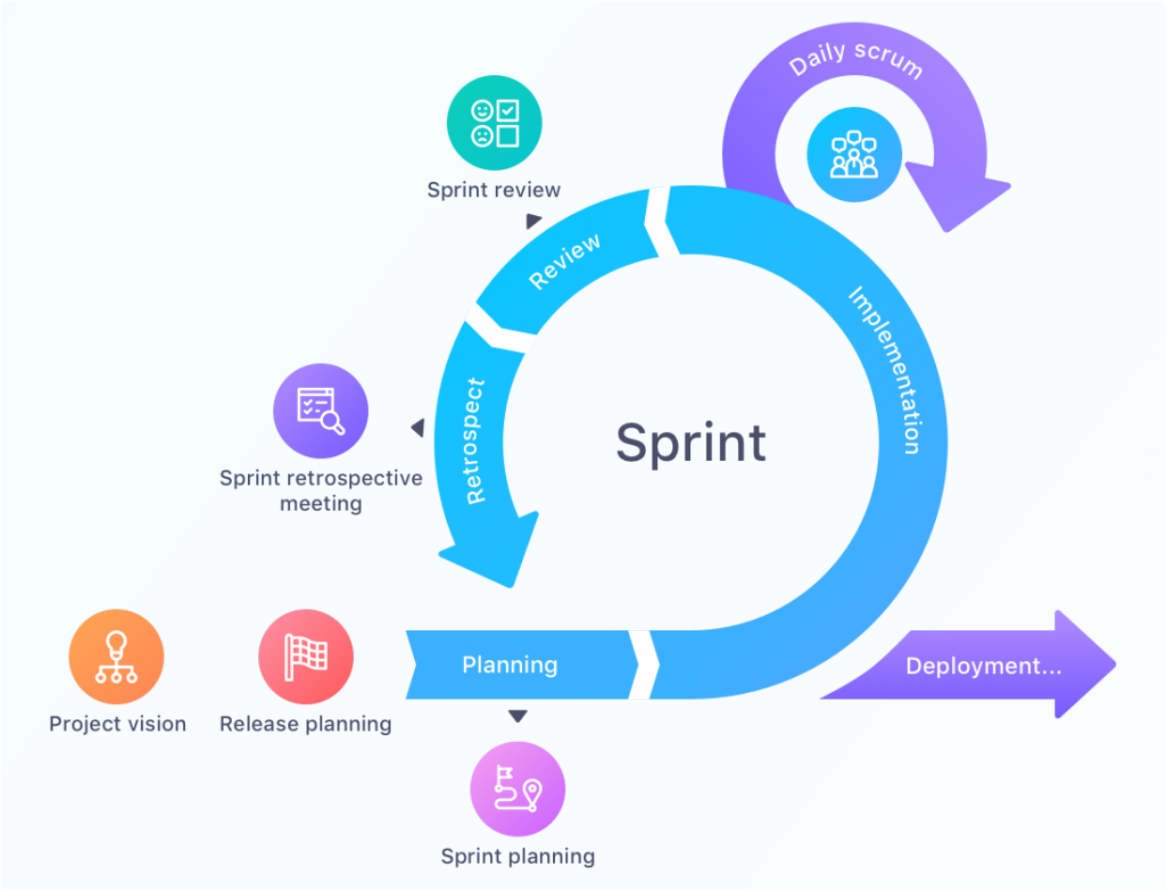
Product Owner: The Product Owner's main goal is to maximize the value of the product created by the Development Team. This involves managing the Product Backlog, conducting analyses, and maintaining a business-oriented focus. They also handle external interactions and act as the interface between stakeholders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

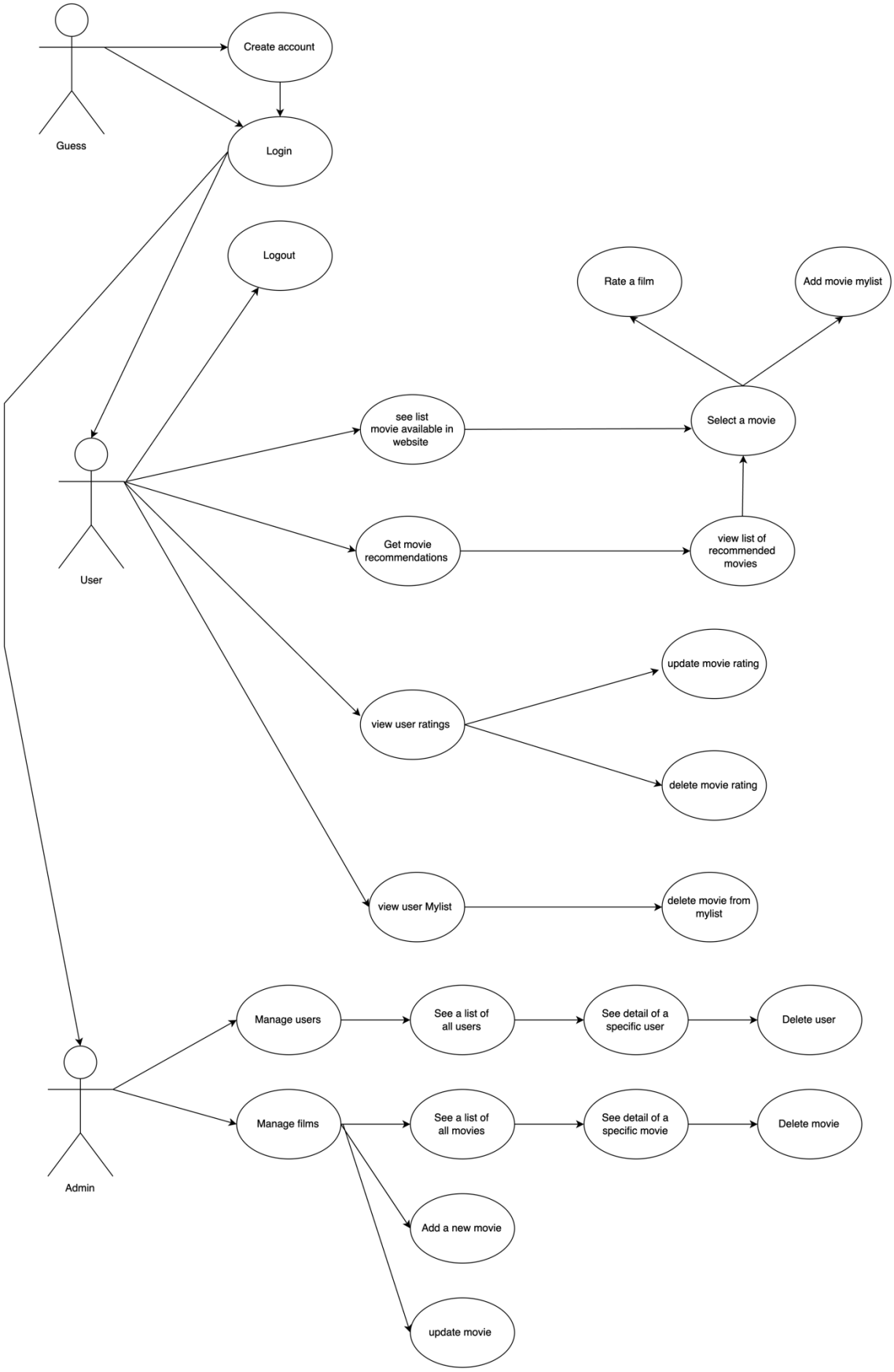
- Search for movie
- Get suggested movie
- Get a specific movie detail
- Create new rating
- Read existed rating
- Update existed rating
- Delete existed rating
- Add movie to user mylist to rate later
- Delete movie from user mylist

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:


Guess
Create account
Login
Logout
User
see list movie available in website
Rate a film
Add movie mylist
Select a movie
Get movie recommendations
view list of recommended movies
view user ratings
update movie rating
delete movie rating
view user Mylist
delete movie from mylist
Admin
Manage users
See a list of all users
See detail of a specific user
Delete user
Manage films
See a list of all movies
See detail of a specific movie
Delete movie
Add a new movie
update movie

5.3 Functional requirements

Functional requirements refer to the product features or functions that must be implemented bythe 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, functionalrequirements 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.

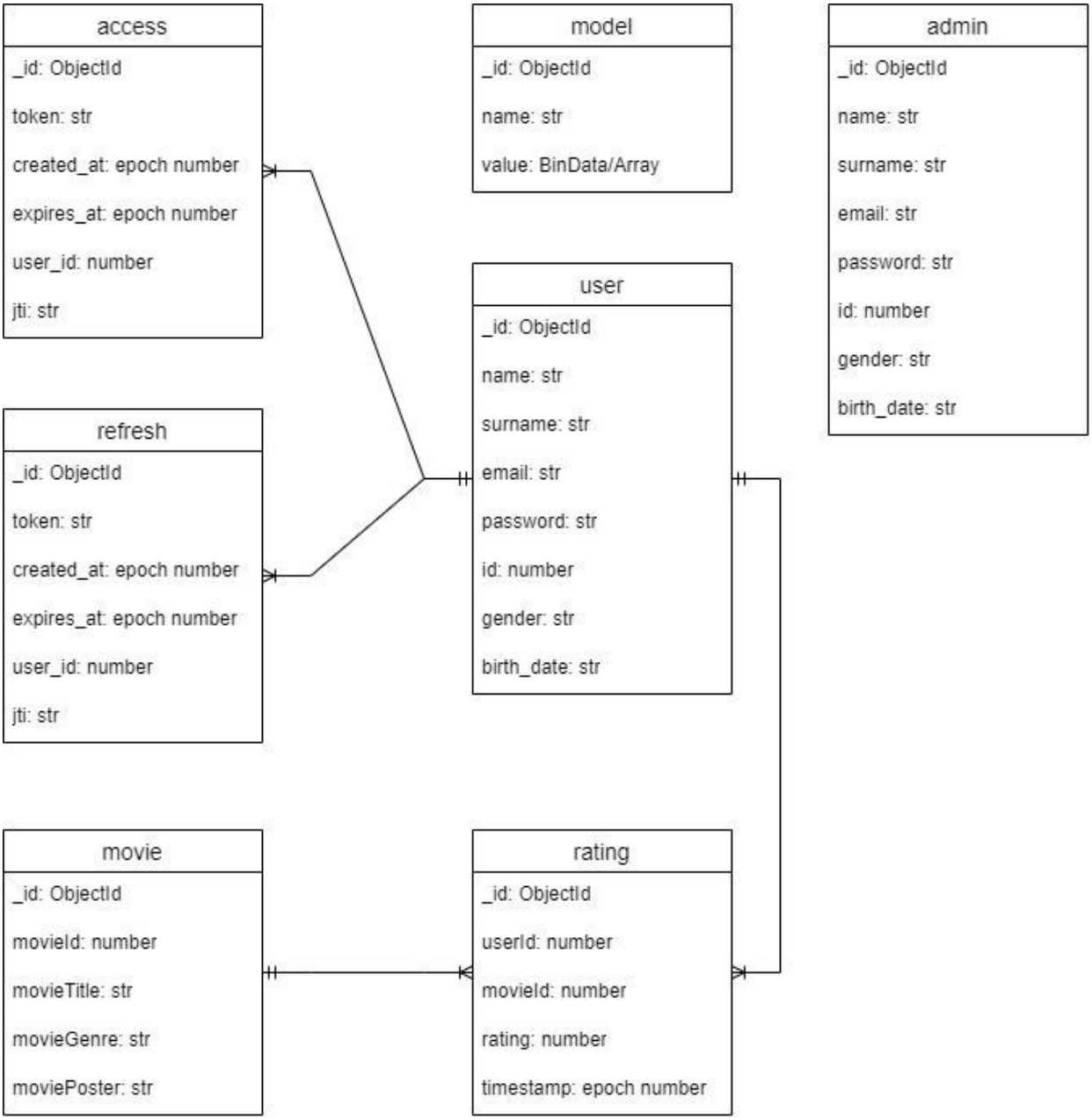
5.5 Domain requirements

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.

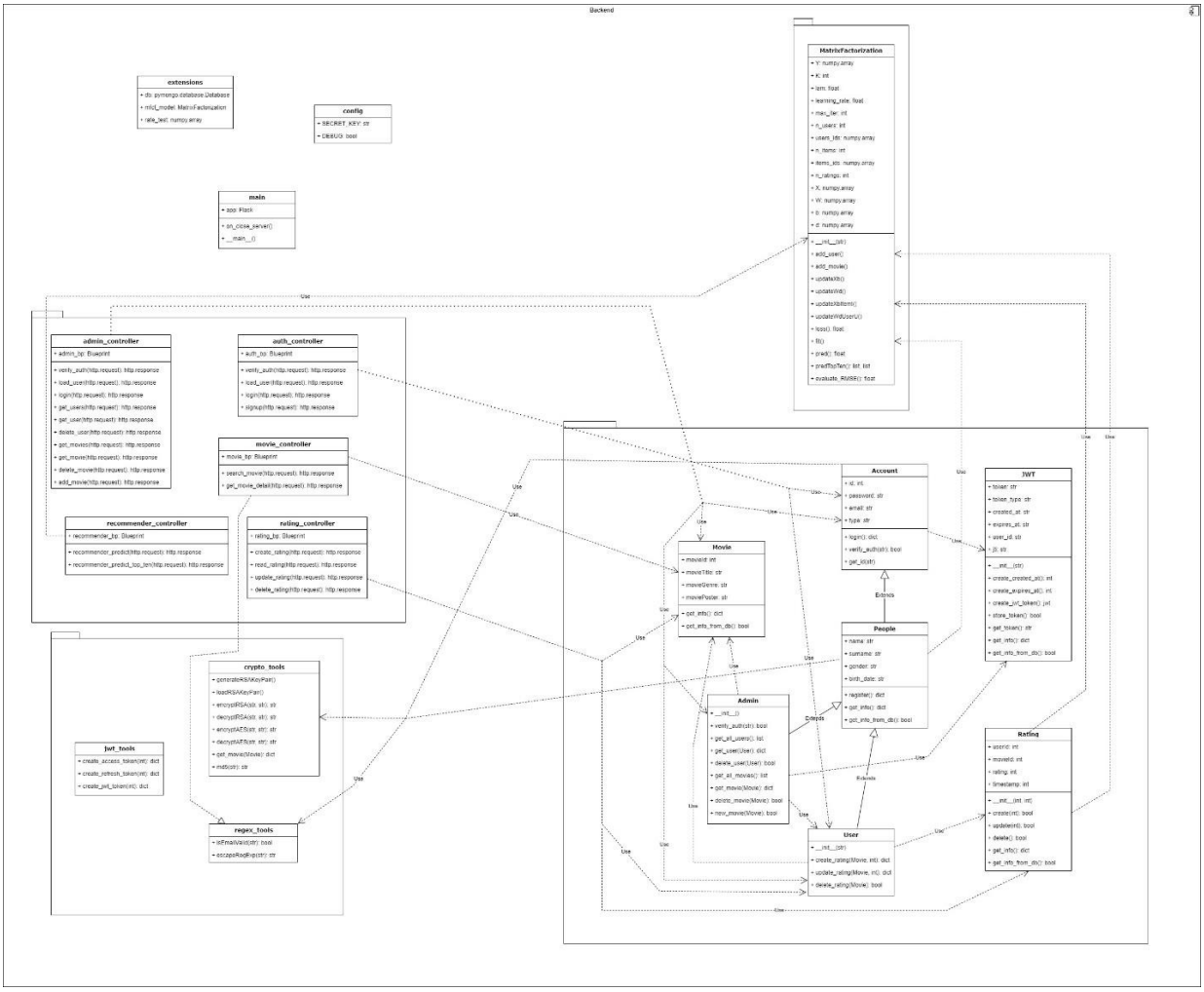
6. Architectural Design

6.1. Database Design

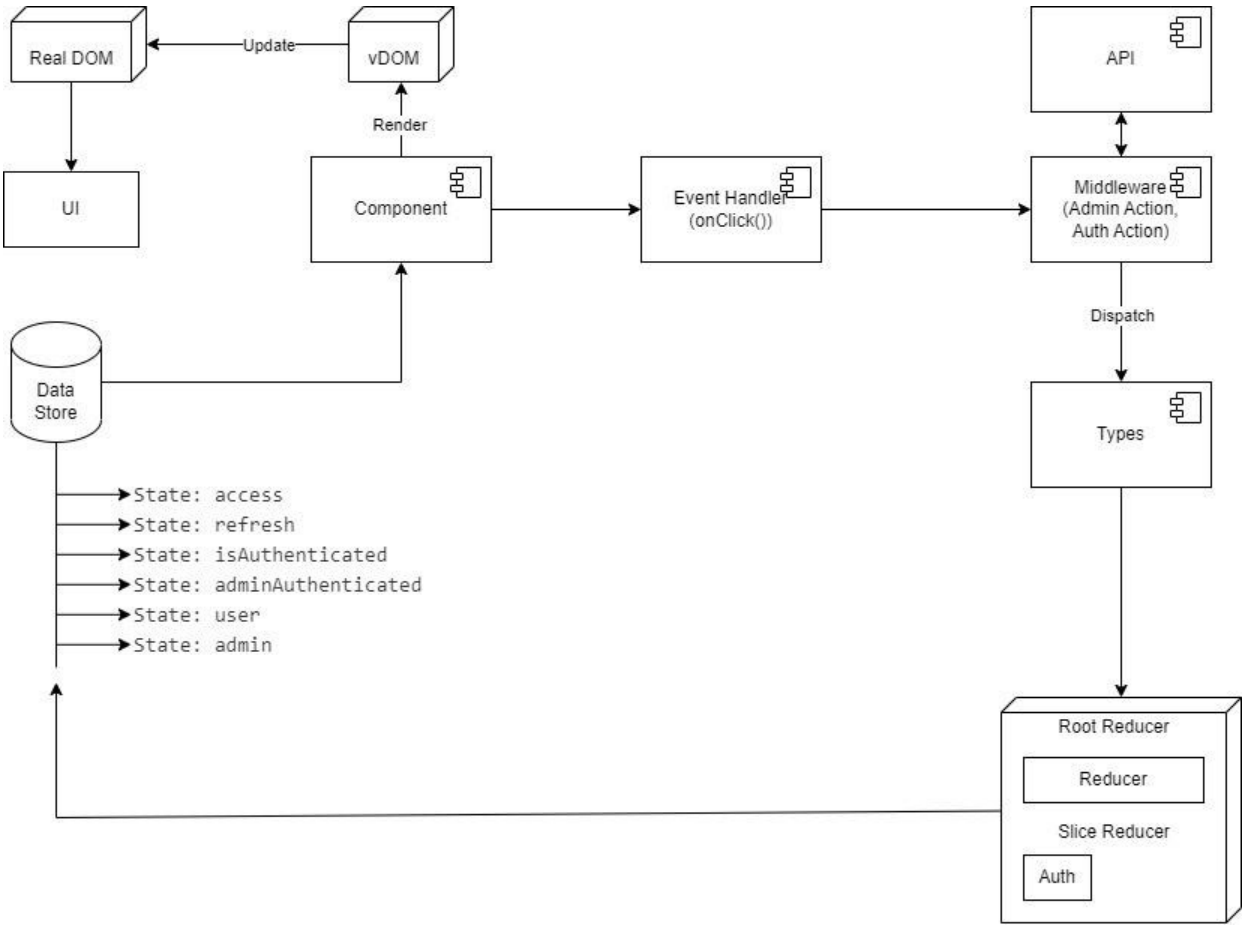


6.2. Backend Design

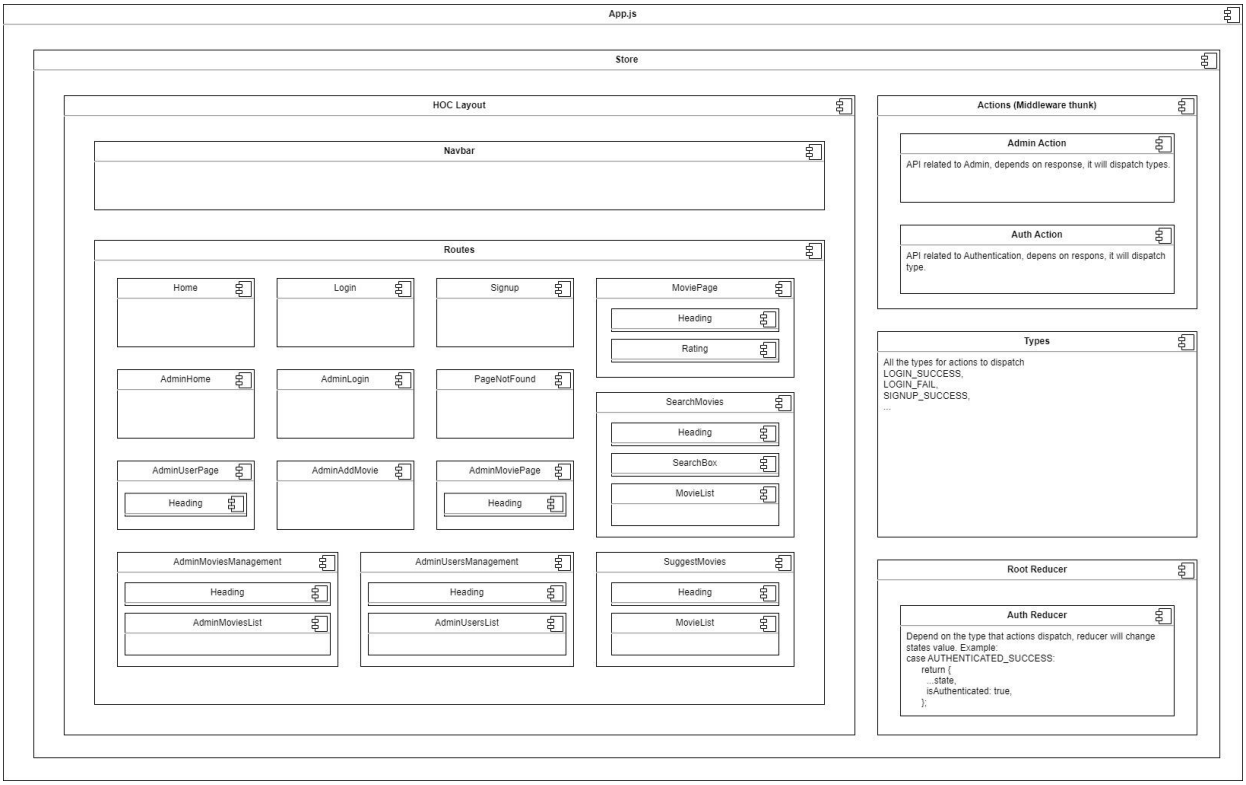
Class Diagram


Backend
extensions
+ db: pymongo.database.Database
+ mfcLmodel: MatrixFactorization
+ rate_test: numpy.array
config
+ SECRET_KEY: str
+ DEBUG: bool
main
+ app: Flask
+ on_close_server()
+ __main__()
MatrixFactorization
+ Y: numpy.array
+ K: int
+ lam: float
+ learning_rate: float
+ max_iter: int
+ n_users: int
+ users_ids: numpy.array
+ n_items: int
+ items_ids: numpy.array
+ n_ratings: int
+ X: numpy.array
+ W: numpy.array
+ b: numpy.array
+ d: numpy.array
+ __init__(y,k)
+ add_user()
+ add_movie()
+ updateAll()
+ updateW()
+ updateXBiases()
+ updateWDUserU()
+ loss(): float
+ fit()
+ pred(): float
+ predTopTen(): list, list
+ evaluate_RMSE(): float
admin_controller
+ admin_bp: Blueprint
+ verify_auth(http.request): http.response
+ load_user(http.request): http.response
+ login(http.request): http.response
+ get_users(http.request): http.response
+ get_user(http.request): http.response
+ delete_user(http.request): http.response
+ get_movies(http.request): http.response
+ get_movie(http.request): http.response
+ delete_movie(http.request): http.response
+ add_movie(http.request): http.response
auth_controller
+ auth_bp: Blueprint
+ verify_auth(http.request): http.response
+ load_user(http.request): http.response
+ login(http.request): http.response
+ signup(http.request): http.response
movie_controller
+ movie_bp: Blueprint
+ search_movie(http.request): http.response
+ get_movie_detail(http.request): http.response
recommender_controller
+ recommender_bp: Blueprint
+ recommender_predict(http.request): http.response
+ recommender_predict_top_ten(http.request): http.response
rating_controller
+ rating_bp: Blueprint
+ create_rating(http.request): http.response
+ read_rating(http.request): http.response
+ update_rating(http.request): http.response
+ delete_rating(http.request): http.response
crypto_tools
+ generateRSAKeyPair()
+ loadRSAKeyPair()
+ encryptRSA(str): str
+ decryptRSA(str): str
+ encryptAES(str): str
+ decryptAES(str): str
+ get_movie(Movie): dict
+ md5(str): str
jwt_tools
+ create_access_token(int): dict
+ create_refresh_token(int): dict
+ create_jwt_token(int): dict
regex_tools
+ isEmailValid(str): bool
+ escapeRoadRep(str): str
Account
+ id: int
+ password: str
+ email: str
+ type: str
+ login(): dict
+ verify_auth(str): bool
+ get_id(str)
JWT
+ token: str
+ token_type: str
+ created_at: str
+ expires_at: str
+ user_id: str
+ jti: str
+ __init__(str)
+ create_created_at(): int
+ create_expires_at(): int
+ create_uri_token(): int
+ store_token(): bool
+ get_token(): str
+ get_info(): dict
+ get_info_from_db(): bool
Movie
+ movieId: int
+ movieTitle: str
+ movieGenre: str
+ moviePoster: str
+ get_info(): dict
+ get_info_from_db(): bool
People
+ name: str
+ surname: str
+ gender: str
+ birth_date: str
+ register(): dict
+ get_info(): dict
+ get_info_from_db(): bool
Admin
+ __init__()
+ verify_auth(str): bool
+ get_all_users(): list
+ get_user(User): dict
+ delete_user(User): bool
+ get_all_movies(): list
+ get_movie(Movie): dict
+ delete_movie(Movie): bool
+ new_movie(Movie): bool
Rating
+ userId: int
+ movieId: int
+ rating: int
+ timestamp: int
+ __init__(int, int)
+ create(int): bool
+ update(int): bool
+ delete(): bool
+ get_info(): dict
+ get_info_from_db(): bool
User
+ __init__(str)
+ create_rating(Movie, int): dict
+ update_rating(Movie, int): dict
+ delete_rating(Movie): bool
Use
Extends

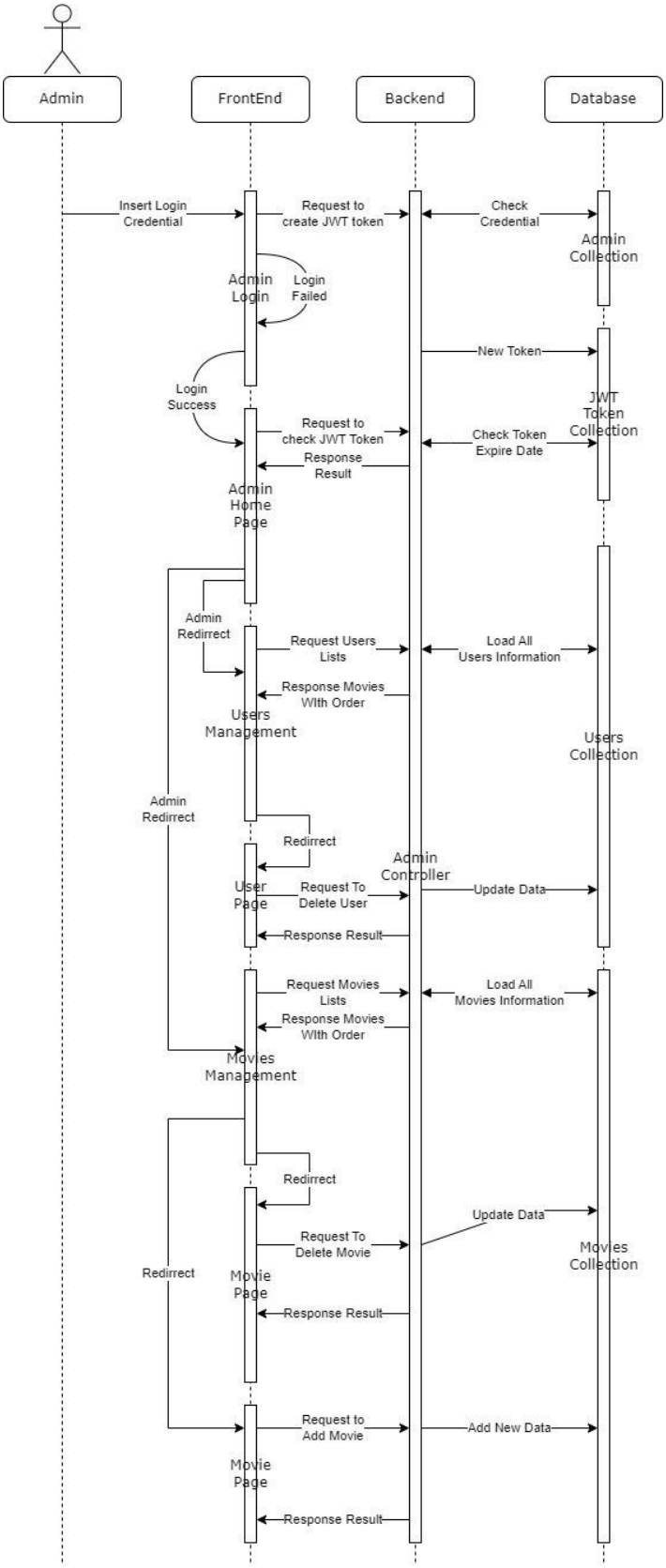
6.3. Frontend Design

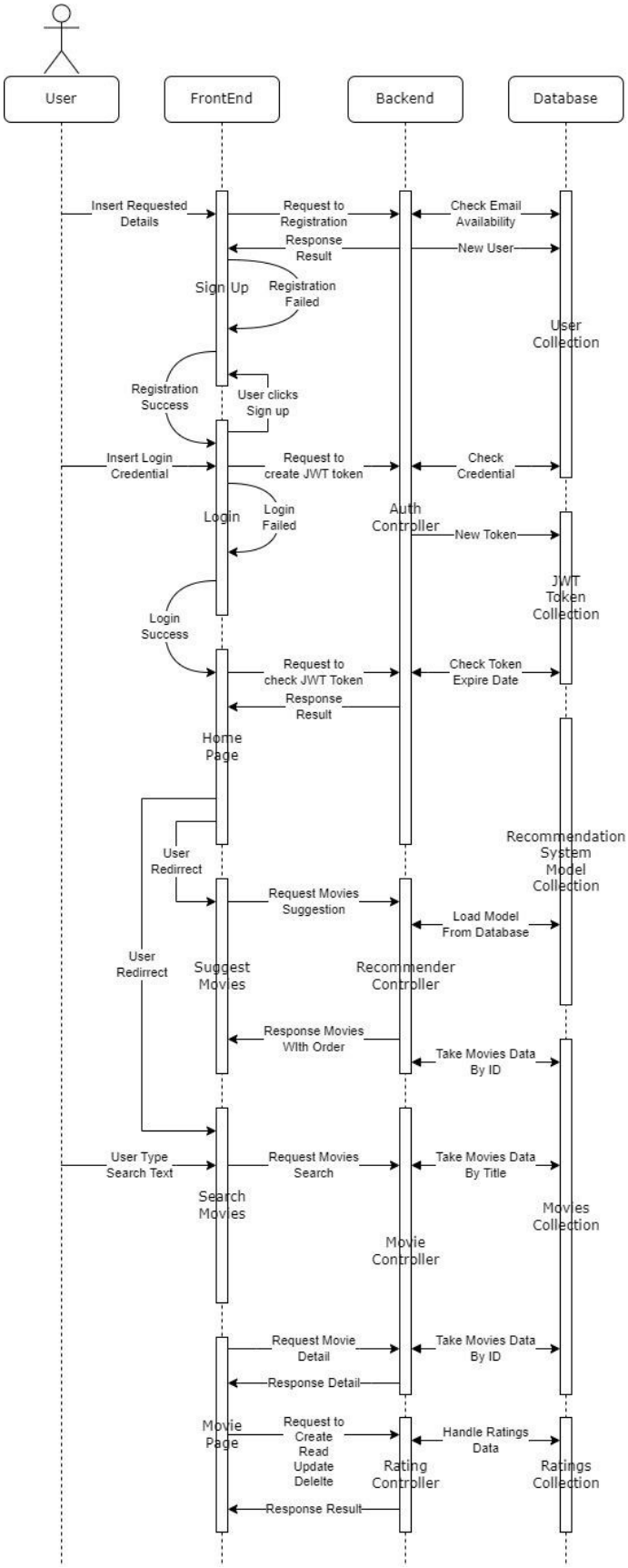


6.4. Ui Layer Design Diagram

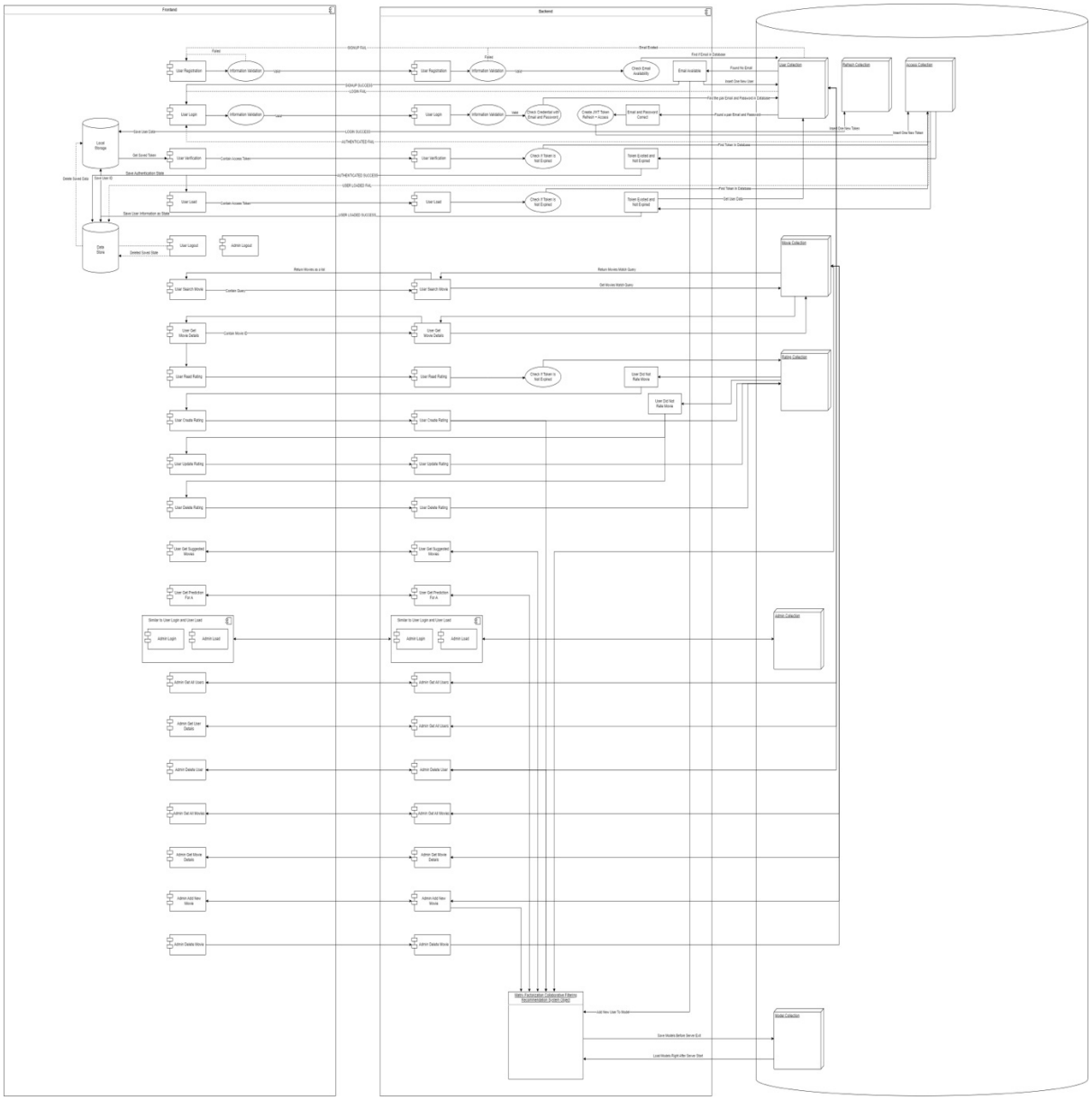


6.5. Sequence Diagram



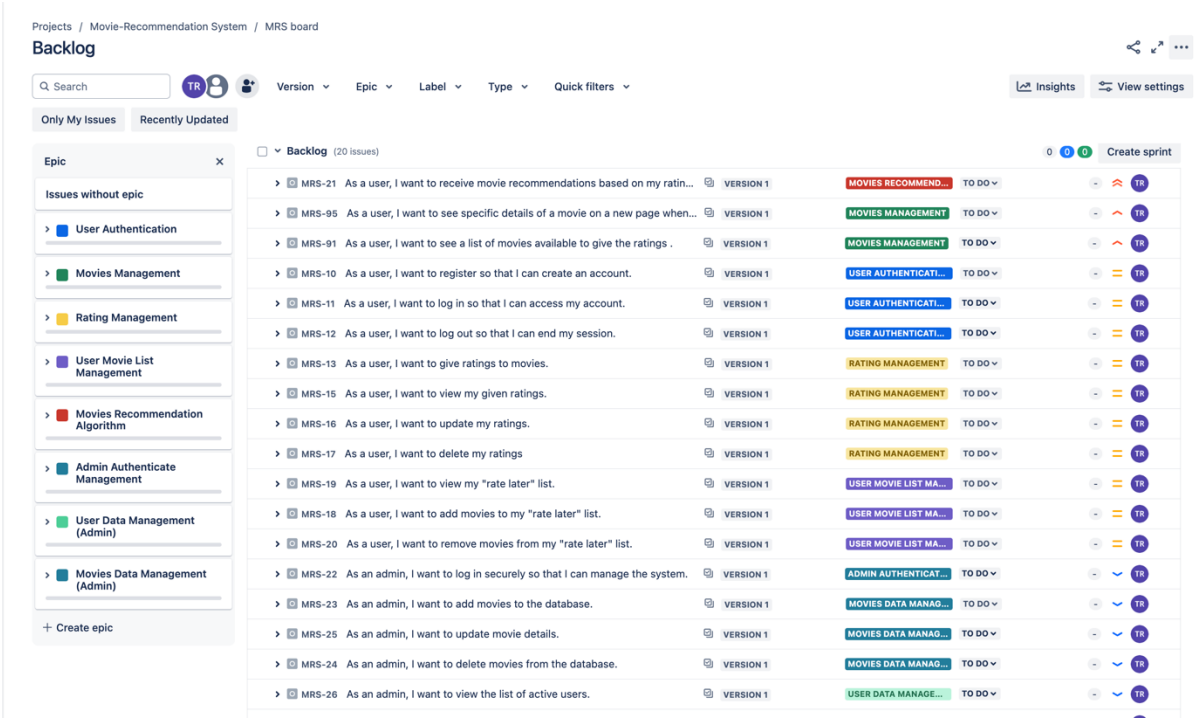

User
FrontEnd
Backend
Database
Insert Requested Details
Request to Registration
Check Email Availability
Response Result
New User
Sign Up
Registration Failed
User Collection
Registration Success
User clicks Sign up
Insert Login Credential
Request to create JWT token
Check Credential
Login
Login Failed
Auth Controller
New Token
JWT Token Collection
Login Success
Request to check JWT Token
Check Token Expire Date
Response Result
Home Page
User Redirrect
Recommendation System Model Collection
Request Movies Suggestion
Load Model From Database
User Redirrect
Suggest Movies
Recommender Controller
Response Movies With Order
Take Movies Data By ID
User Type Search Text
Request Movies Search
Take Movies Data By Title
Search Movies
Movies Collection
Movie Controller
Request Movie Detail
Take Movies Data By ID
Response Detail
Movie Page
Request to Create Read Update Delelte
Handle Ratings Data
Rating Controller
Ratings Collection
Response Result

6.6. Components Diagram



7. Backlog refinement

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



7.1 Epic

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

Projects / Movie-Recommendation System / MRS board

Backlog

Search | 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

0 of 0 issues visible | Estimate: 0 of 0

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

MRS-10	As a user, I want to register so that I can create an account.	VERSION 1	USER AUTHENTICATI...	TO DO
MRS-36	Create a registration form in React.	VERSION 1		TO DO
MRS-31	Validate registration data on the frontend.	VERSION 1		TO DO
MRS-33	Develop API endpoint in Django to handle user registration.	VERSION 1		TO DO
MRS-34	Store user information securely in the database.	VERSION 1		TO DO
MRS-11	As a user, I want to log in so that I can access my account.	VERSION 1	USER AUTHENTICATI...	TO DO
MRS-37	Create a login form in React.	VERSION 1		TO DO
MRS-38	Validate login credentials on the frontend.	VERSION 1		TO DO
MRS-39	Develop API endpoint in Django to handle user login.	VERSION 1		TO DO
MRS-40	Implement JWT token authentication.	VERSION 1		TO DO
MRS-12	As a user, I want to log out so that I can end my session.	VERSION 1	USER AUTHENTICATI...	TO DO
MRS-29	Create a logout button in React.	VERSION 1		TO DO
MRS-30	Invalidate the JWT token on the frontend.	VERSION 1		TO DO

+ Create issue

MRS Sprint 2 Add dates (0 issues) — 0 0 0

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

+ Create issue

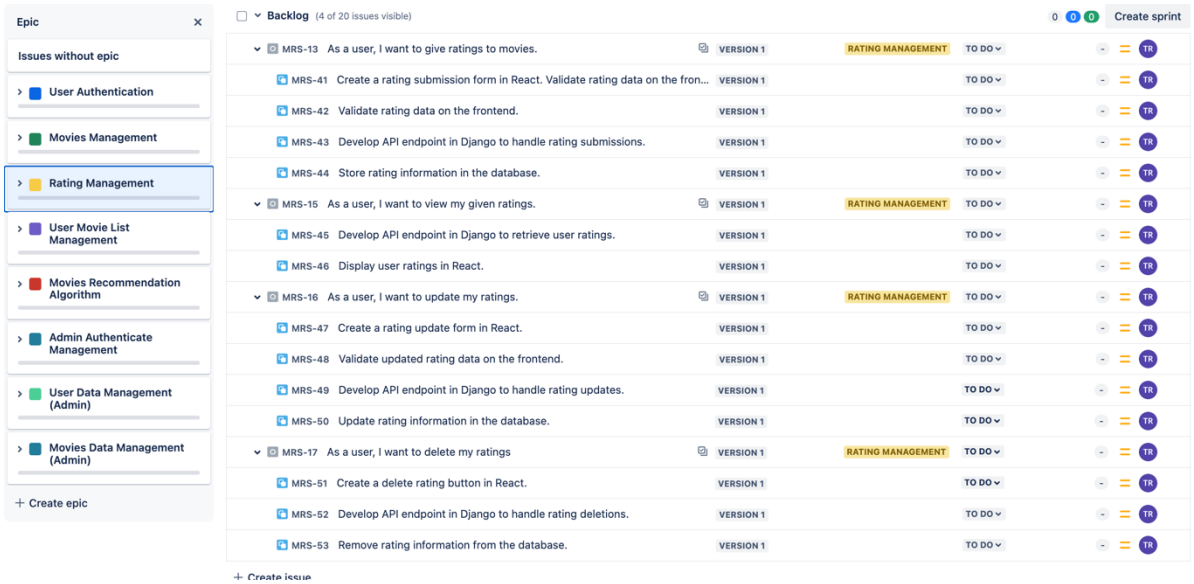
- User Authentication
- Movies Management
- Rating Management
- User Movie List Management
- Movies Recommendation Algorithm
- Admin Authenticate Management
- User Data Management (Admin)
- Movies Data Management (Admin)

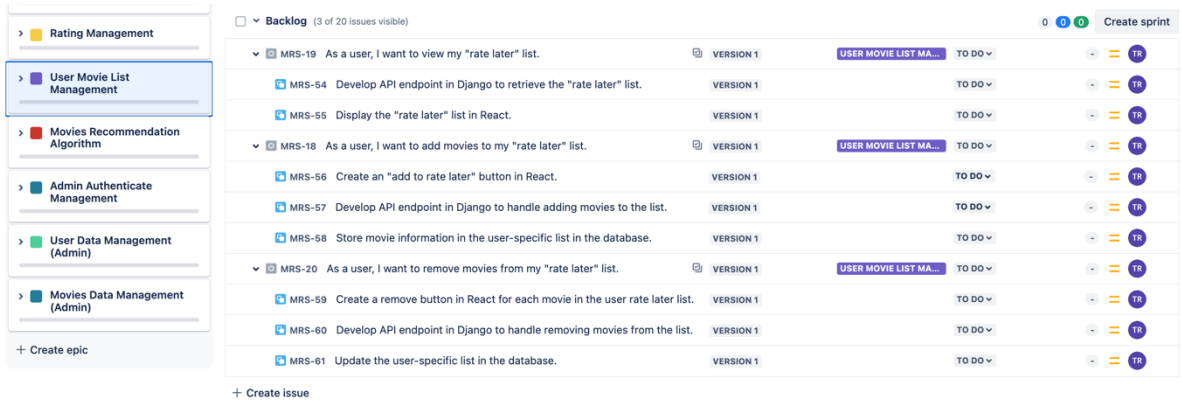
+ Create epic

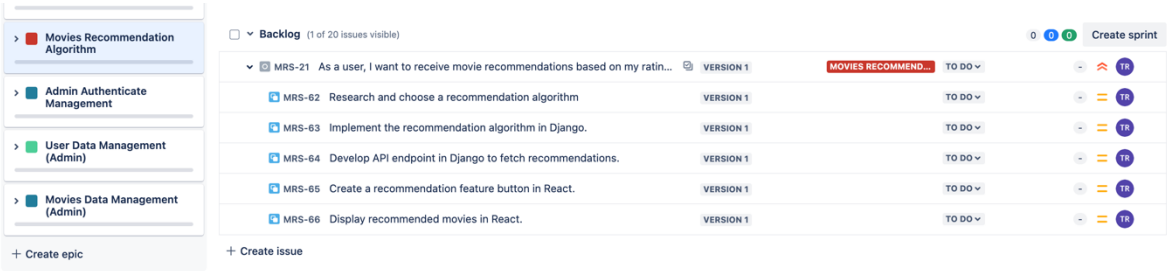
0 of 0 issues visible | Estimate: 0 of 0

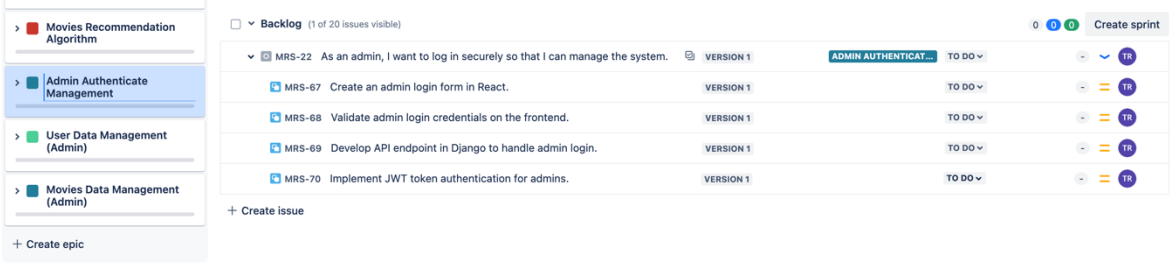
Backlog (2 of 20 issues visible) — 0 0 0 — Create sprint

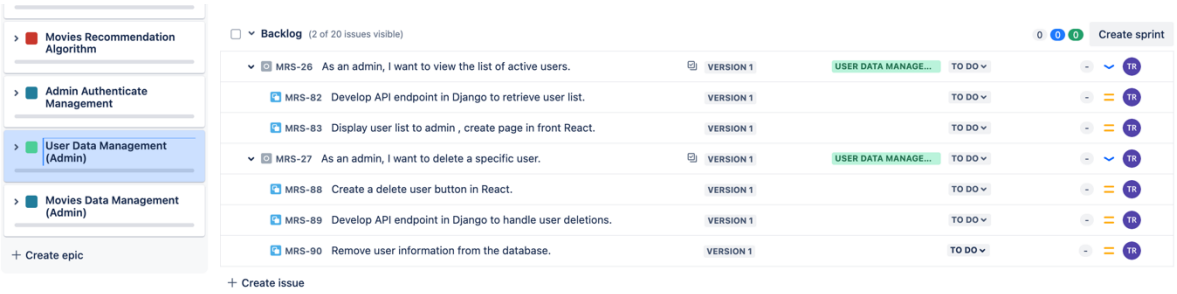
MRS-95	As a user, I want to see specific details of a movie on a new page when...	VERSION 1	MOVIES MANAGEMENT	TO DO
MRS-96	Develop API endpoint in Django to retrieve movie details.	VERSION 1		TO DO
MRS-97	Create a movie details UI in React.	VERSION 1		TO DO
MRS-98	Implement navigation to the movie details page when a movie is clicked	VERSION 1		TO DO
MRS-99	Fetch and display the specific details of the selected movie on the movi...	VERSION 1		TO DO
MRS-91	As a user, I want to see a list of movies available to give the ratings .	VERSION 1	MOVIES MANAGEMENT	TO DO
MRS-92	Develop API endpoint in Django to retrieve the list of movies.	VERSION 1		TO DO
MRS-93	Create a movie list UI in React.	VERSION 1		TO DO
MRS-94	Fetch and display the list of movies in the movie list UI.	VERSION 1		TO DO

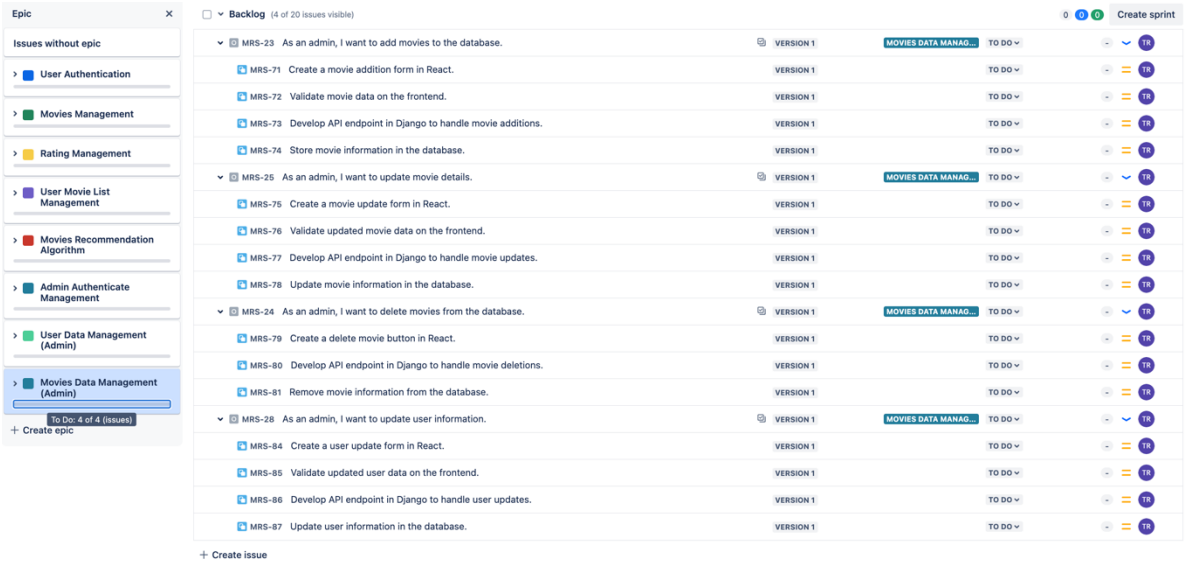

Epic
Issues without epic
User Authentication
Movies Management
Rating Management
User Movie List Management
Movies Recommendation Algorithm
Admin Authenticate Management
User Data Management (Admin)
Movies Data Management (Admin)
Create epic
Backlog (4 of 20 issues visible)
Create sprint
MRS-13 As a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


Rating Management
User Movie List Management
Movies Recommendation Algorithm
Admin Authenticate Management
User Data Management (Admin)
Movies Data Management (Admin)
Create epic
Backlog (3 of 20 issues visible)
Create sprint
MRS-19 As a user, I want to view my "rate later" list. VERSION 1 USER MOVIE LIST MA... TO DO
MRS-54 Develop API endpoint in Django to retrieve the "rate later" list. VERSION 1 TO DO
MRS-55 Display the "rate later" list in React. VERSION 1 TO DO
MRS-18 As a user, I want to add movies to my "rate later" list. VERSION 1 USER MOVIE LIST MA... TO DO
MRS-56 Create an "add to rate later" button in React. VERSION 1 TO DO
MRS-57 Develop API endpoint in Django to handle adding movies to the list. VERSION 1 TO DO
MRS-58 Store movie information in the user-specific list in the database. VERSION 1 TO DO
MRS-20 As a user, I want to remove movies from my "rate later" list. VERSION 1 USER MOVIE LIST MA... TO DO
MRS-59 Create a remove button in React for each movie in the user rate later list. VERSION 1 TO DO
MRS-60 Develop API endpoint in Django to handle removing movies from the list. VERSION 1 TO DO
MRS-61 Update the user-specific list in the database. VERSION 1 TO DO
Create issue


Movies Recommendation Algorithm
Admin Authenticate Management
User Data Management (Admin)
Movies Data Management (Admin)
Create epic
Backlog (1 of 20 issues visible)
Create sprint
MRS-21 As a user, I want to receive movie recommendations based on my ratin... VERSION 1 MOVIES RECOMMEND... TO DO
MRS-62 Research and choose a recommendation algorithm VERSION 1 TO DO
MRS-63 Implement the recommendation algorithm in Django. VERSION 1 TO DO
MRS-64 Develop API endpoint in Django to fetch recommendations. VERSION 1 TO DO
MRS-65 Create a recommendation feature button in React. VERSION 1 TO DO
MRS-66 Display recommended movies in React. VERSION 1 TO DO
Create issue


Movies Recommendation Algorithm
Admin Authenticate Management
User Data Management (Admin)
Movies Data Management (Admin)
Create epic
Backlog (1 of 20 issues visible)
Create sprint
MRS-22 As an admin, I want to log in securely so that I can manage the system. VERSION 1 ADMIN AUTHENTICAT... TO DO
MRS-67 Create an admin login form in React. VERSION 1 TO DO
MRS-68 Validate admin login credentials on the frontend. VERSION 1 TO DO
MRS-69 Develop API endpoint in Django to handle admin login. VERSION 1 TO DO
MRS-70 Implement JWT token authentication for admins. VERSION 1 TO DO
Create issue





Epics 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.

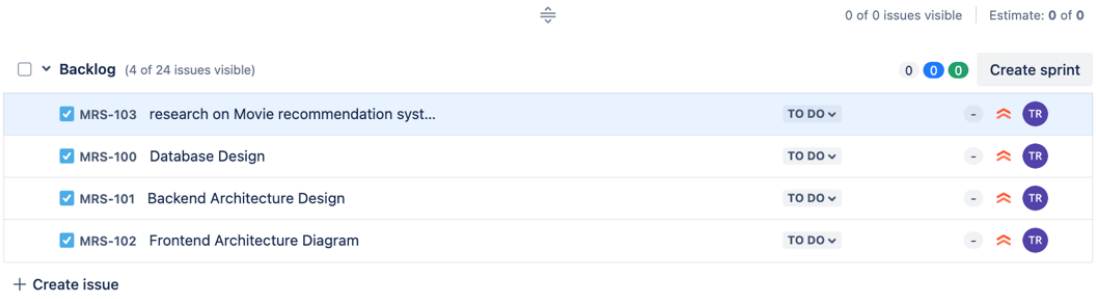
8. Epic: User Management (Admin)

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.



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:

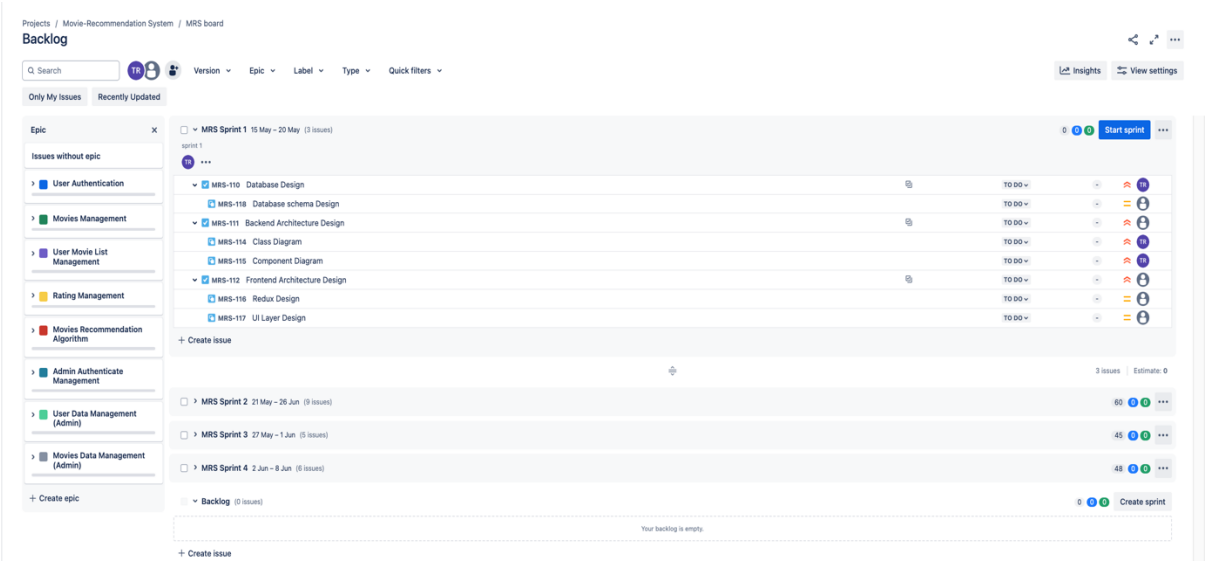
1. Sprint 1 runs from 15/May/2024 to 19/May/2024. (5 working days).
2. Sprint 2 runs from 20/ May/2024 to 24/ May/2024. (5 working days)
3. Sprint 3 runs from 26/ May/2024 to 30/May/2024. (5 working days).
4. Sprint 4 runs from 01/ June/2024 to 05/June/2024. (5 working days) .

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:

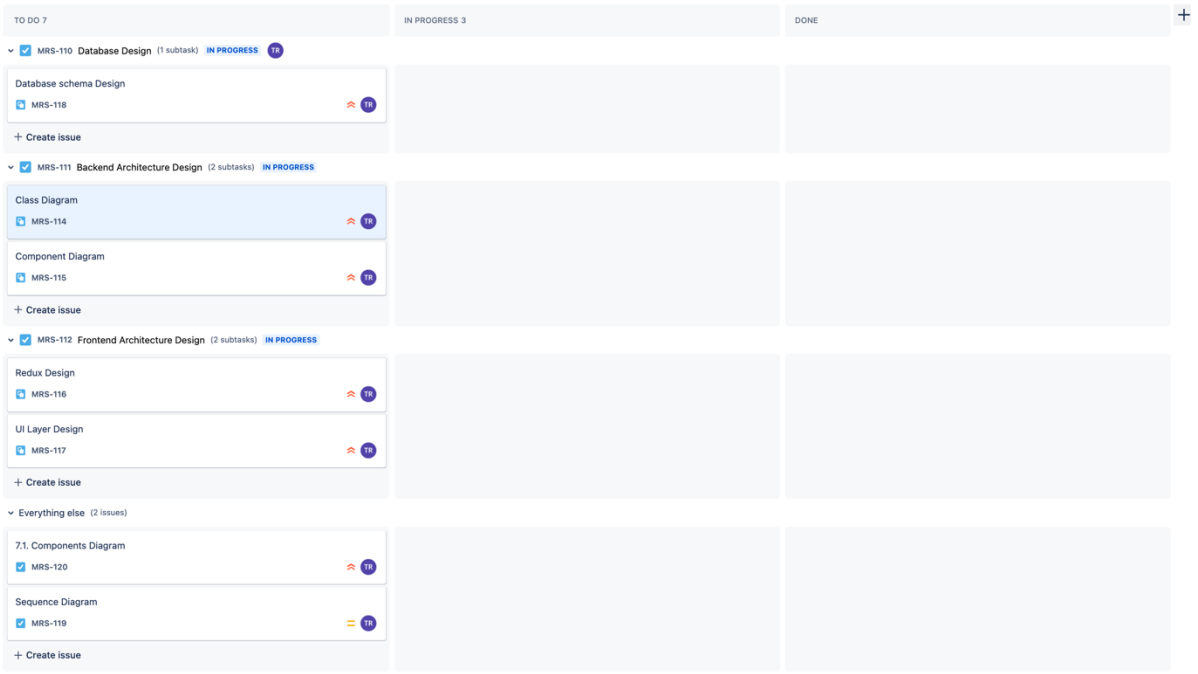


Above requirements are Delivered in this sprint .

1.1 Planning

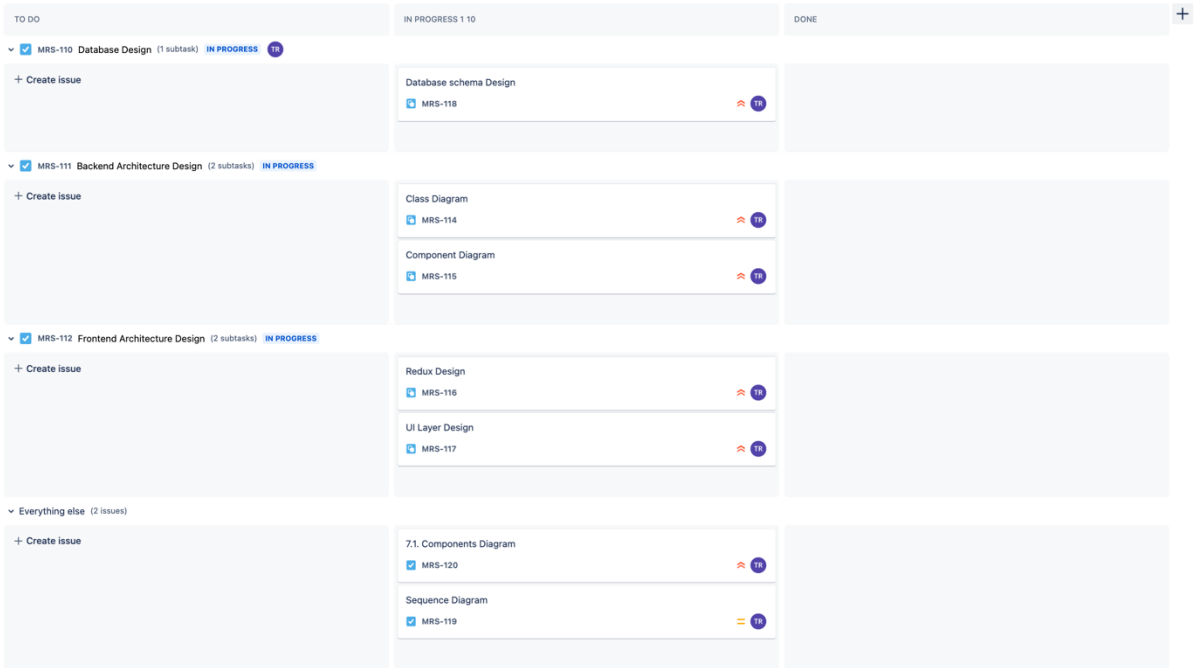
In Scrum Planning, we will do task identification. The team identifies the tasks that need to be complete to deliver the product backlog items that are selected for the sprint.

In Sprint 1, tasks will be as below:



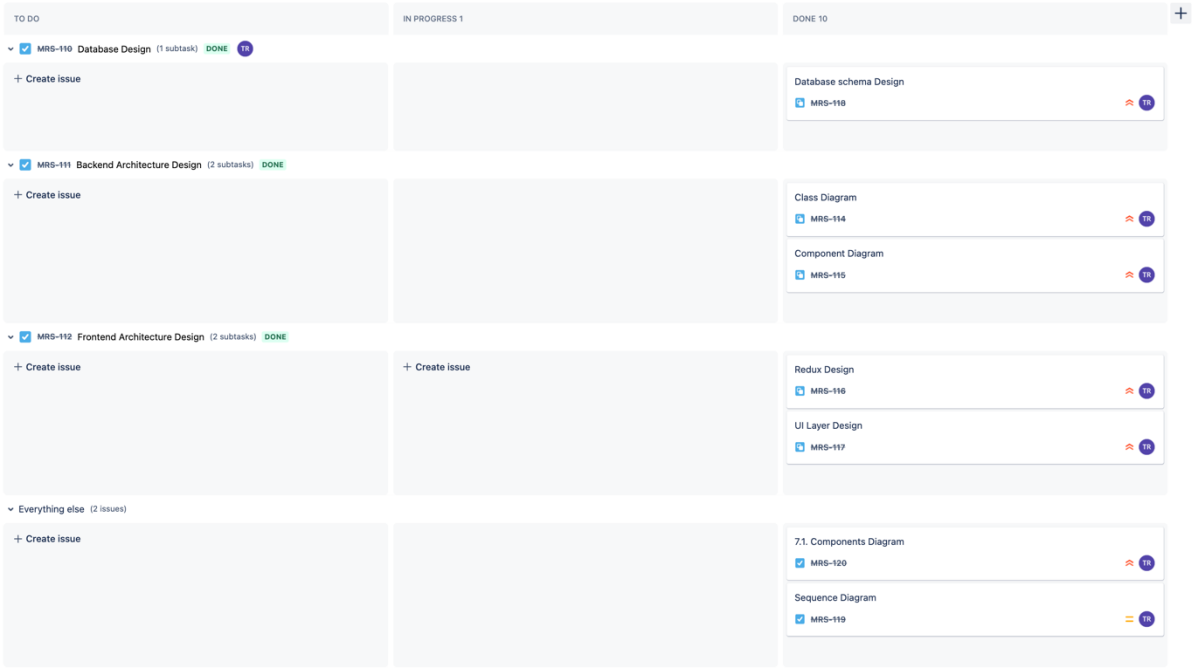
1.2 Implementation

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



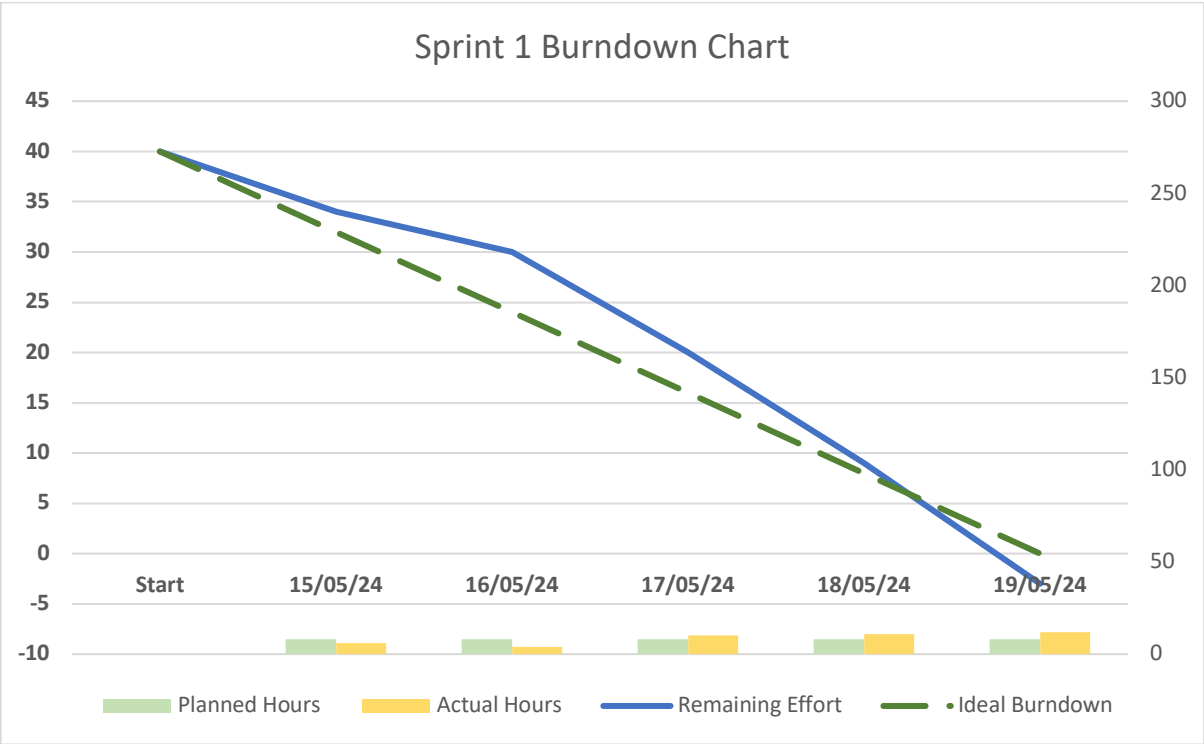
1.3 Review

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

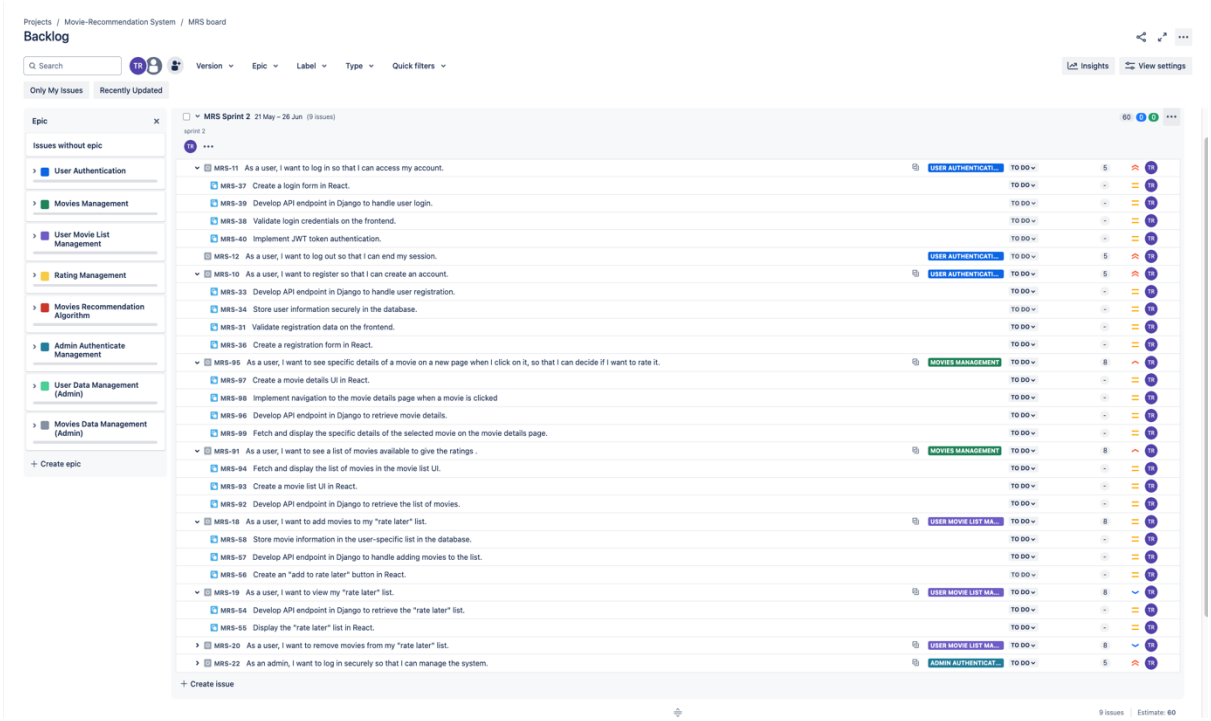
Release Burndown Chart for Sprint 1.



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:



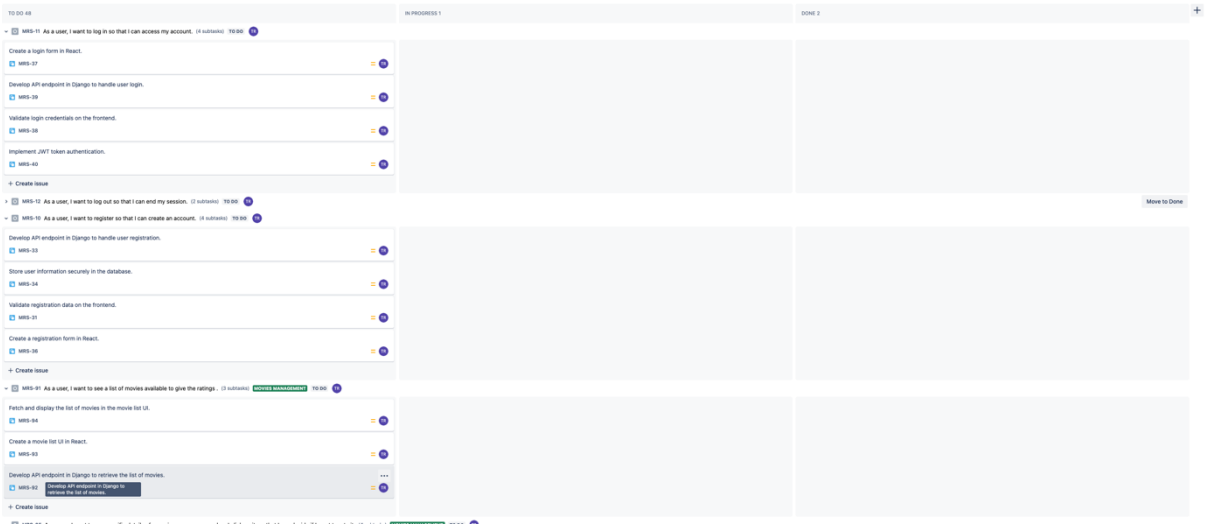
Above requirements are Delivered in this sprint2

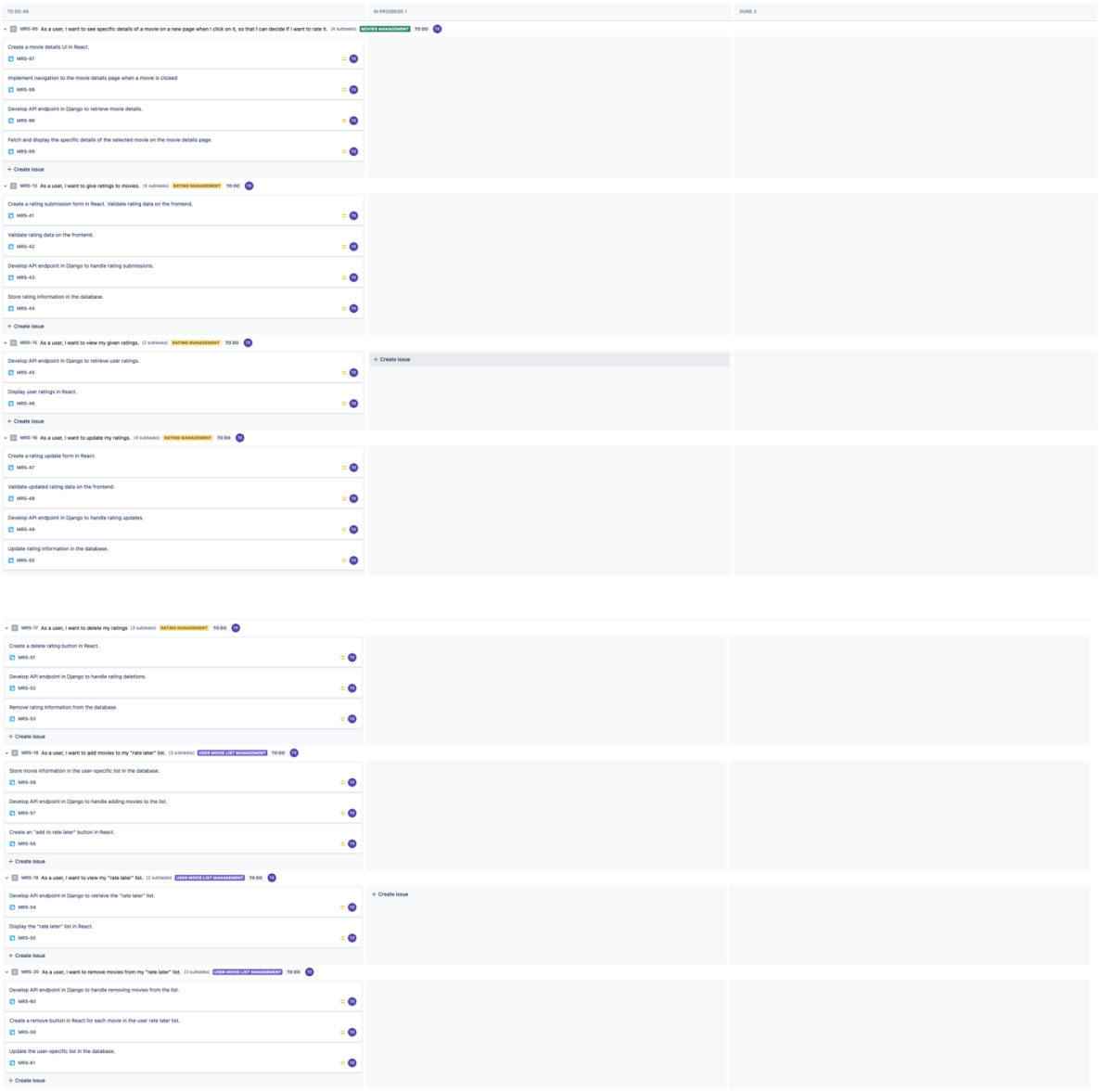
2.1. 2.1 Planning

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:



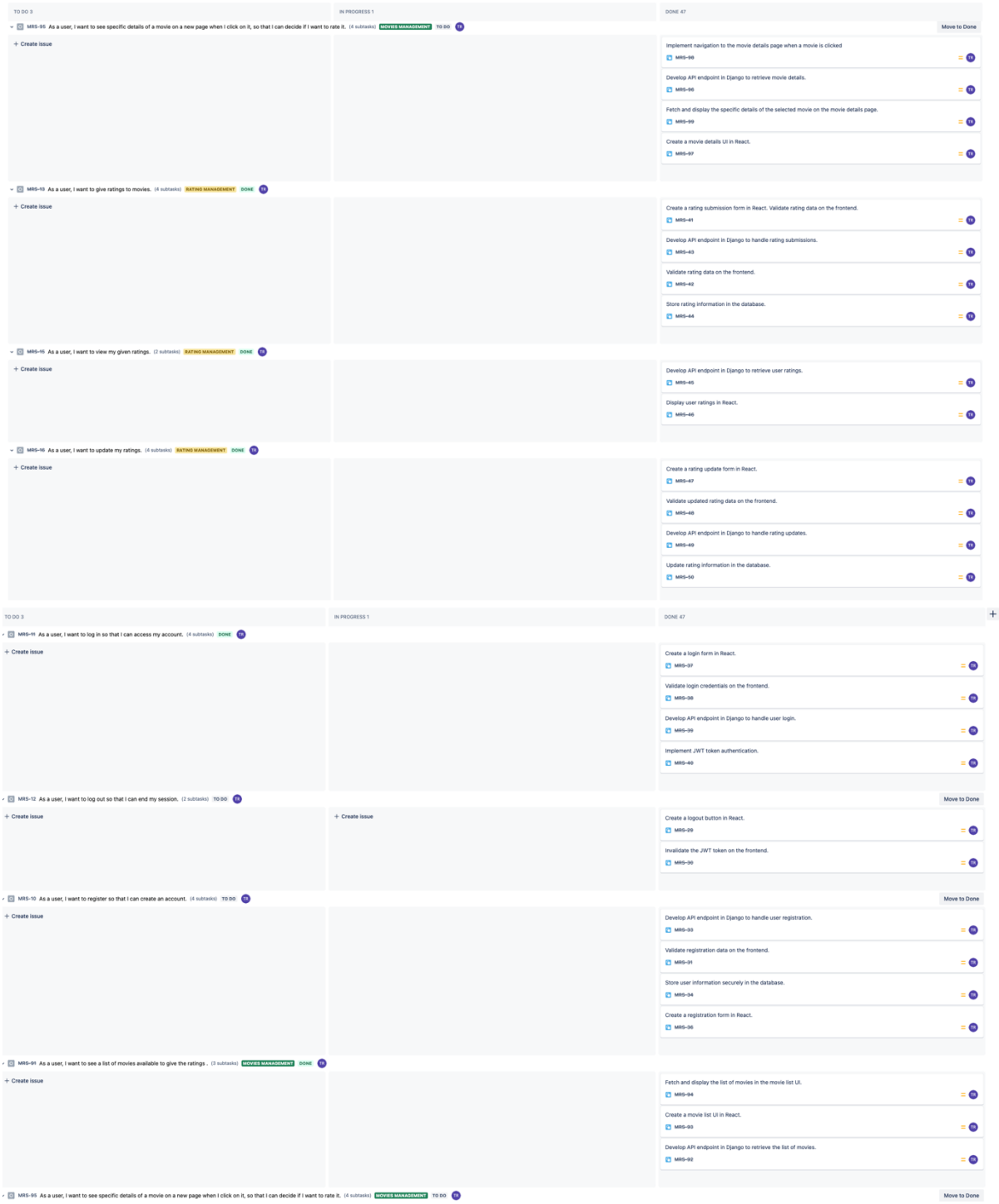


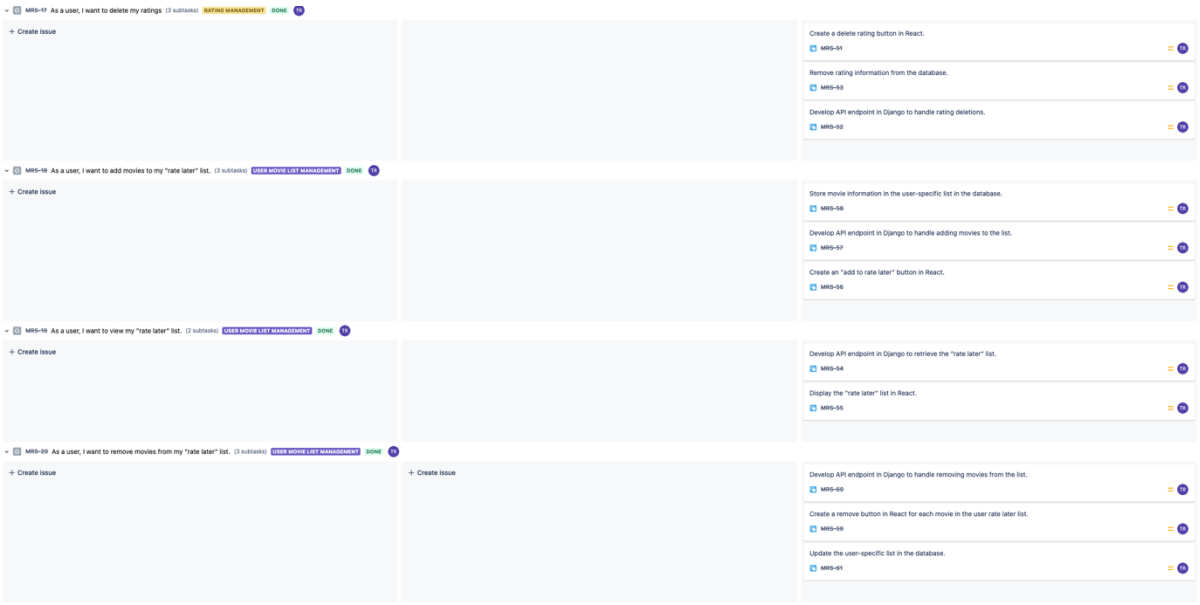
2.2. Implementation

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

2.3 Review

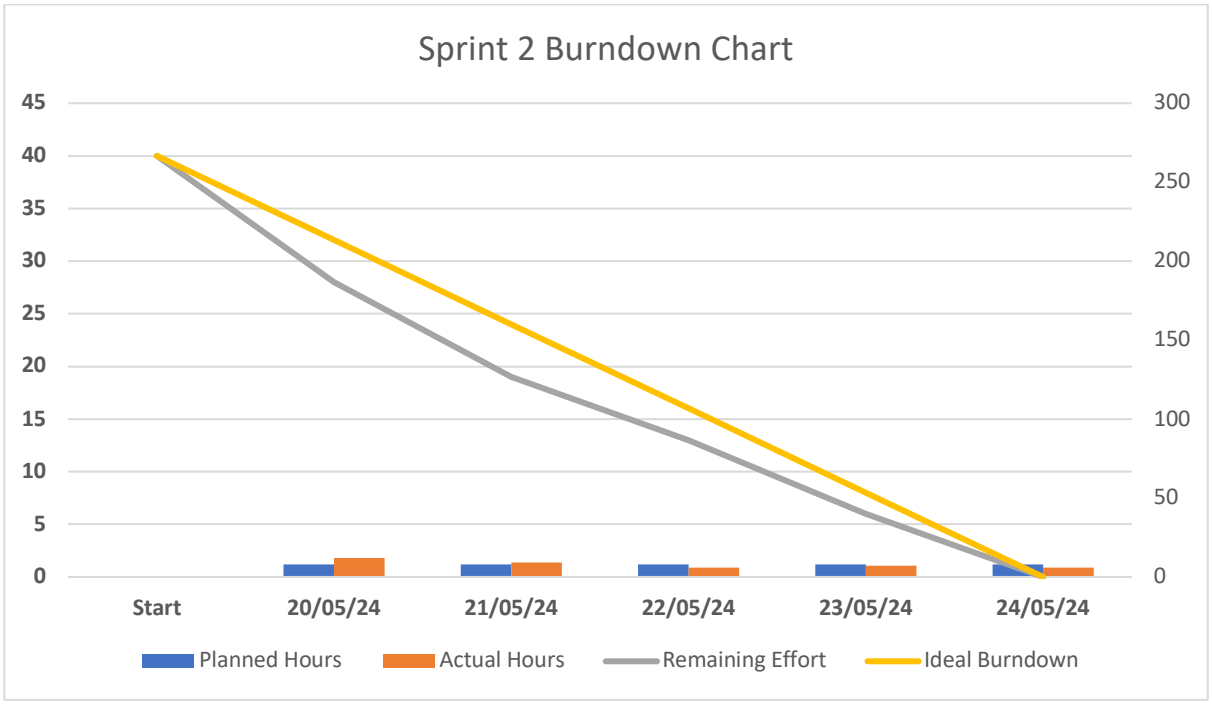
Thanks to the design of the system in Sprint 1, the implementation in Sprint 2 can run more smoothly, there are no backlogs remaining.


TO DO 3
IN PROGRESS 1
DONE 47
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 TA
Move to Done
+ Create issue
Implement navigation to the movie details page when a movie is clicked
MRS-98
Develop API endpoint in Django to retrieve movie details.
MRS-96
Fetch and display the specific details of the selected movie on the movie details page.
MRS-99
Create a movie details UI in React.
MRS-97
MRS-13 As a user, I want to give ratings to movies. (4 subtasks) RATING MANAGEMENT DONE TA
+ Create issue
Create a rating submission form in React. Validate rating data on the frontend.
MRS-41
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
MRS-15 As a user, I want to view my given ratings. (2 subtasks) RATING MANAGEMENT DONE TA
+ Create issue
Develop API endpoint in Django to retrieve user ratings.
MRS-45
Display user ratings in React.
MRS-46
MRS-16 As a user, I want to update my ratings. (4 subtasks) RATING MANAGEMENT DONE TA
+ Create issue
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-50
TO DO 3
IN PROGRESS 1
DONE 47
MRS-11 As a user, I want to log in so that I can access my account. (4 subtasks) DONE TA
+ Create issue
Create a login form in React.
MRS-37
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
MRS-12 As a user, I want to log out so that I can end my session. (2 subtasks) TO DO TA
Move to Done
+ Create issue
+ Create issue
Create a logout button in React.
MRS-29
Invalidate the JWT token on the frontend.
MRS-30
MRS-10 As a user, I want to register so that I can create an account. (4 subtasks) TO DO TA
Move to Done
+ Create issue
Develop API endpoint in Django to handle user registration.
MRS-33
Validate registration data on the frontend.
MRS-31
Store user information securely in the database.
MRS-34
Create a registration form in React.
MRS-36
MRS-91 As a user, I want to see a list of movies available to give the ratings . (3 subtasks) MOVIES MANAGEMENT DONE TA
+ Create issue
Fetch and display the list of movies in the movie list UI.
MRS-94
Create a movie list UI in React.
MRS-93
Develop API endpoint in Django to retrieve the list of movies.
MRS-92
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 TA
Move to Done



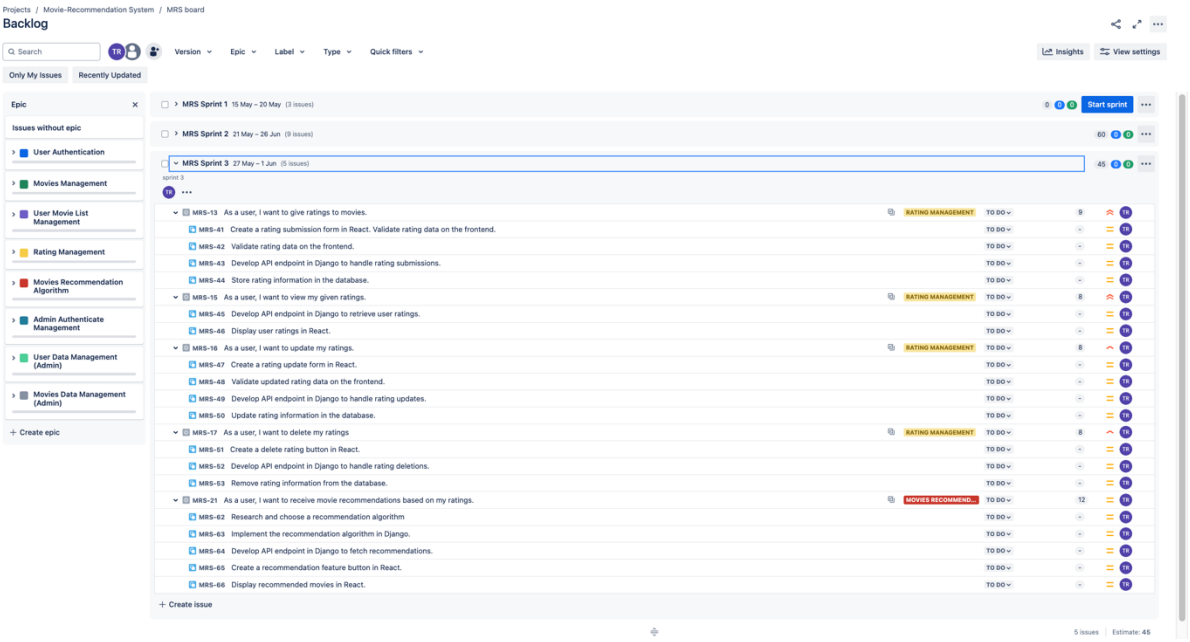
2.4 Retrospective.

Release Burndown Chart:



3. Sprint 3

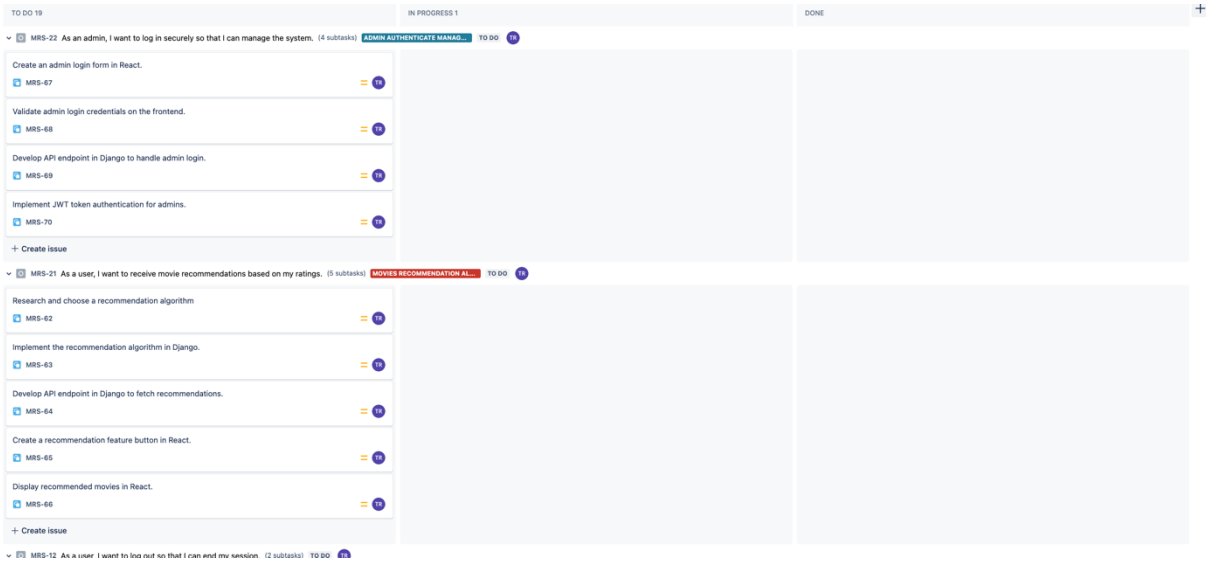
Sprint Backlog:

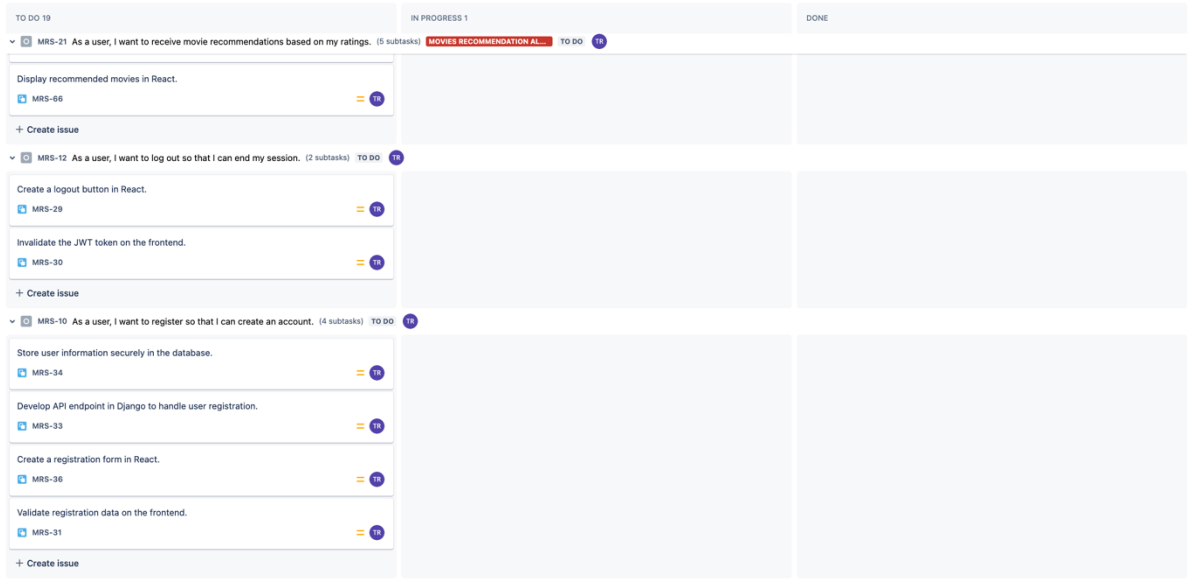


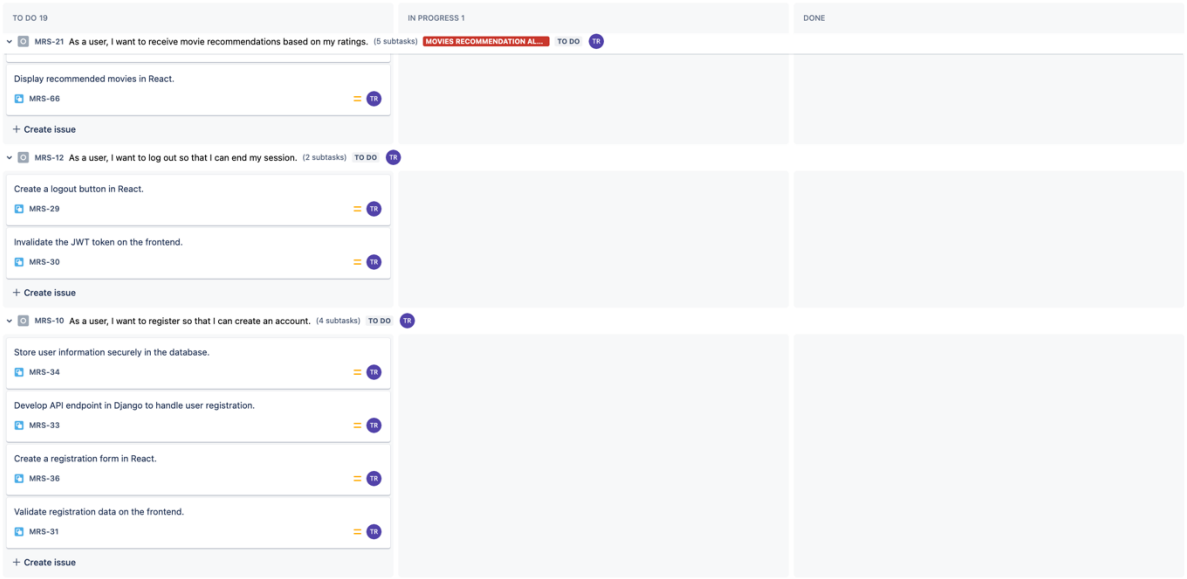
Above requirements are Delivered in this sprint3

3.1. Planning

Task identification for Sprint 3:





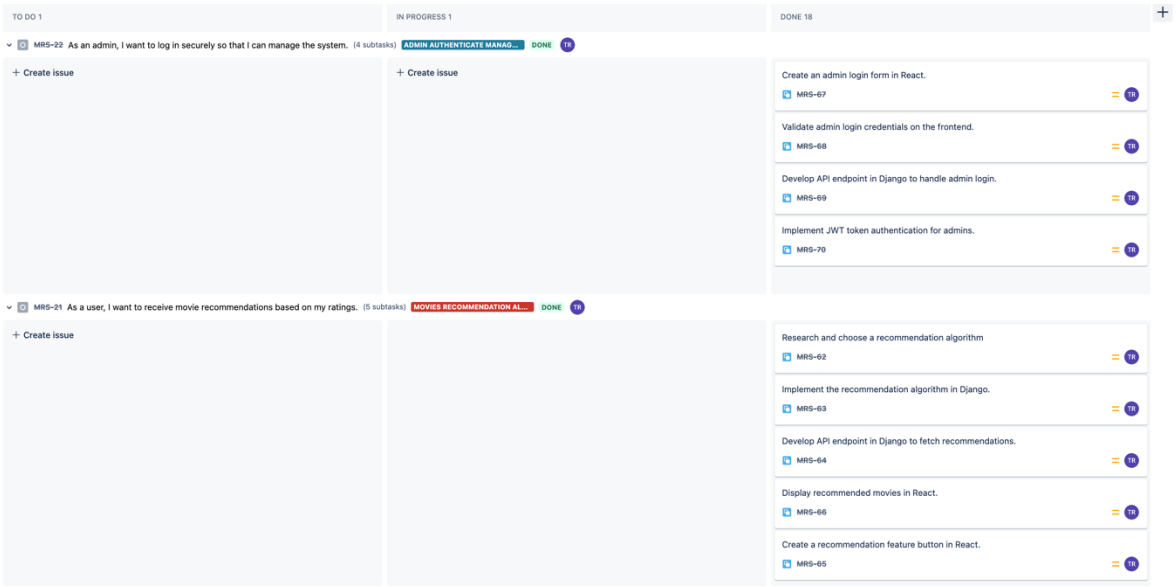


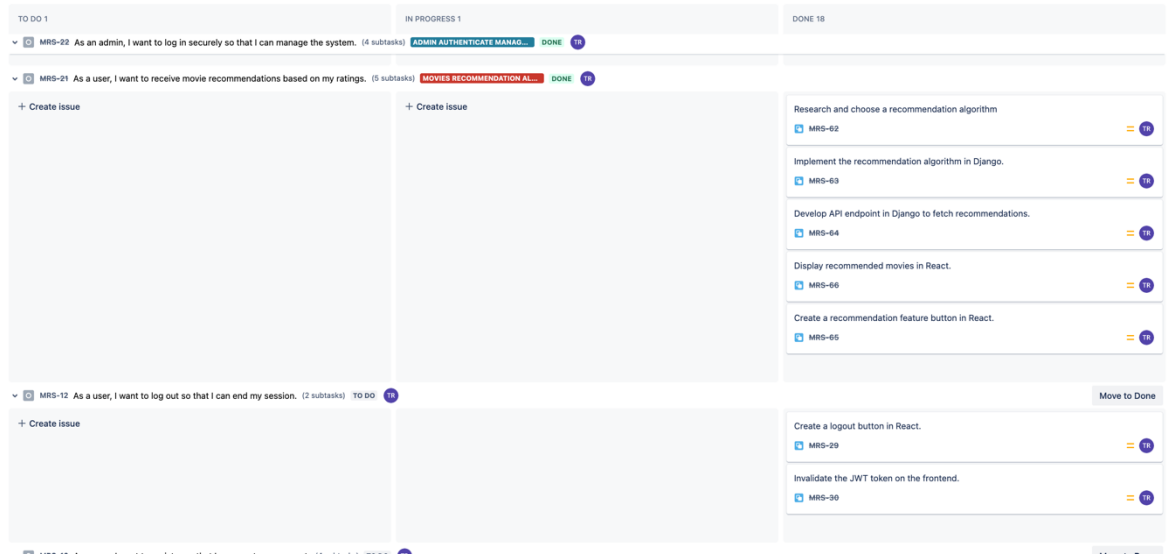
3.2. Implementation

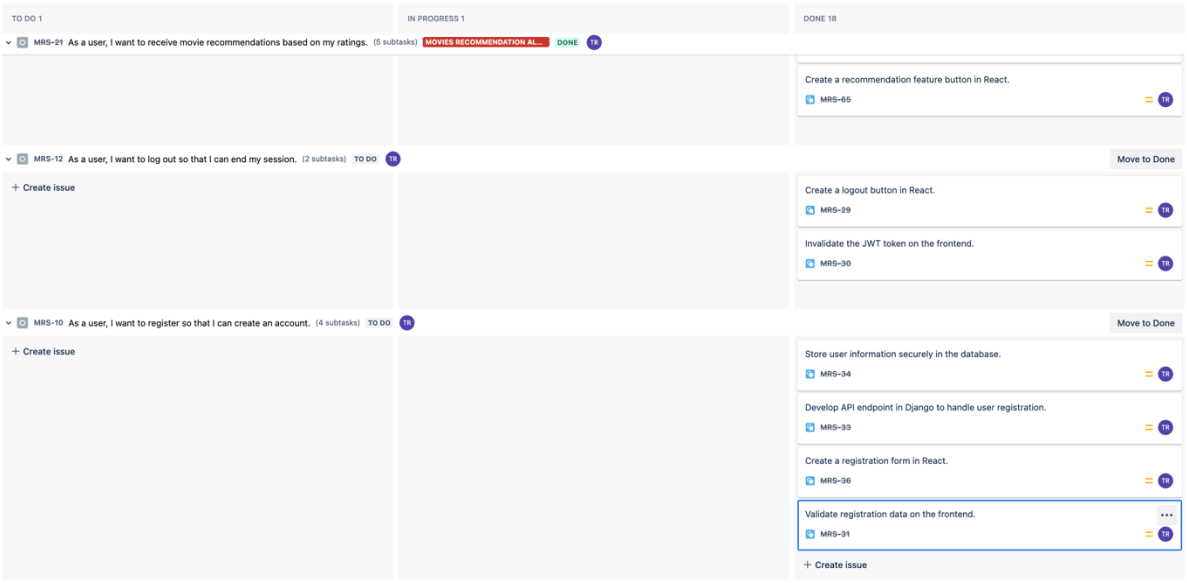
I reuse most of the parts that have been programmed from Scrum before, of course, it will have to be slightly modified due to changes in the Database and API structure in the Backend.

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.

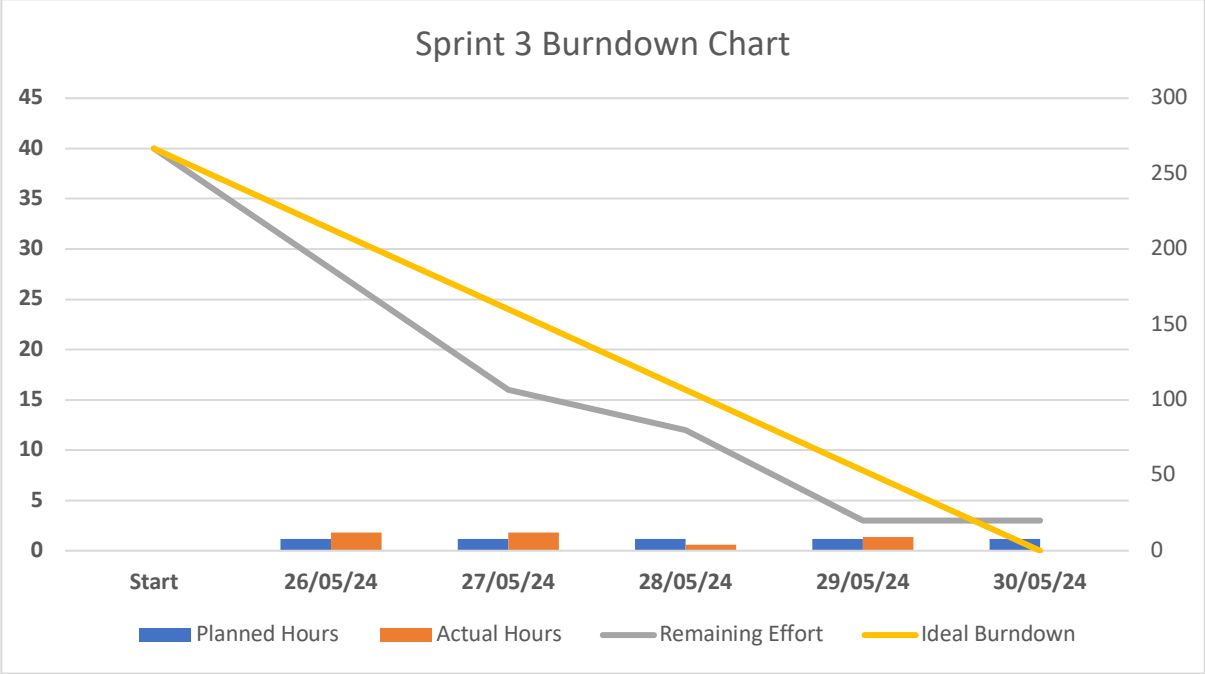






3.4 Retrospective

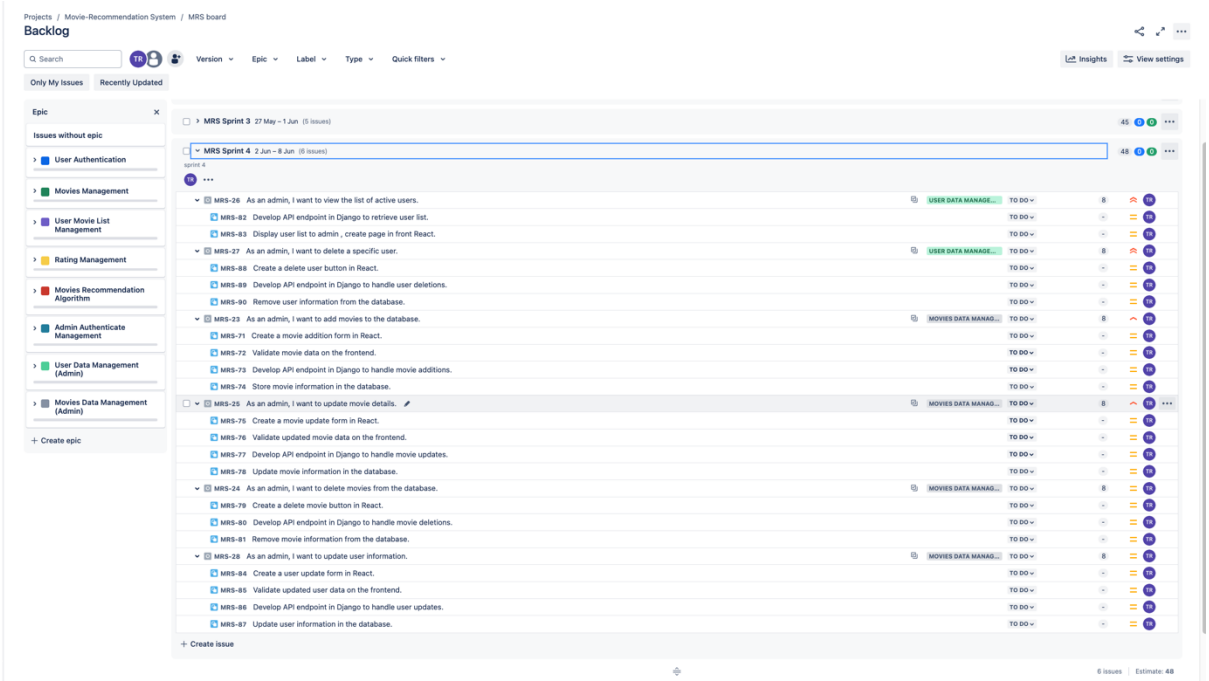
Release Burndown Chart:



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:

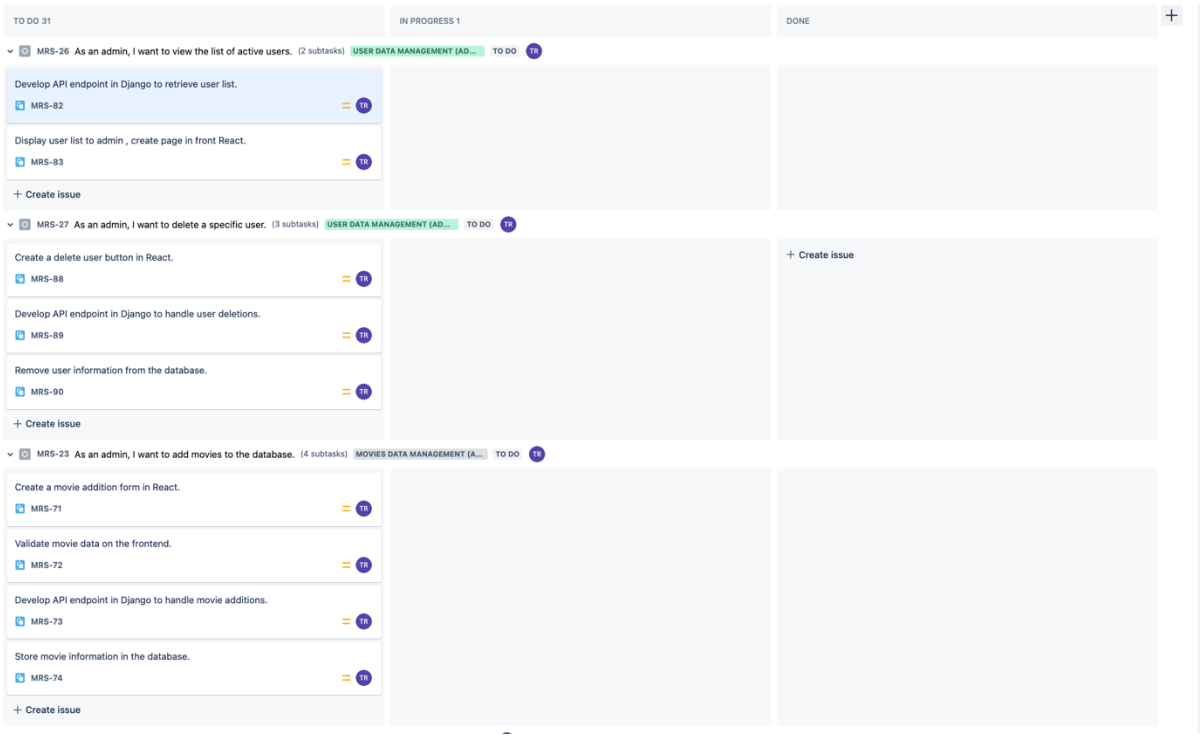


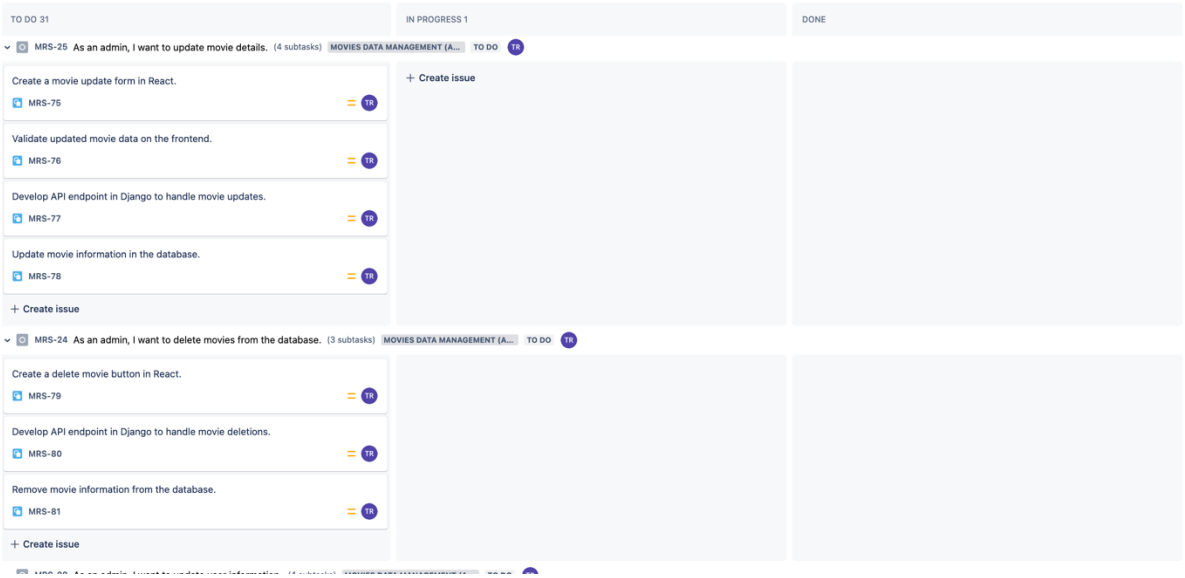
Above requirements are Delivered in this sprint4 .

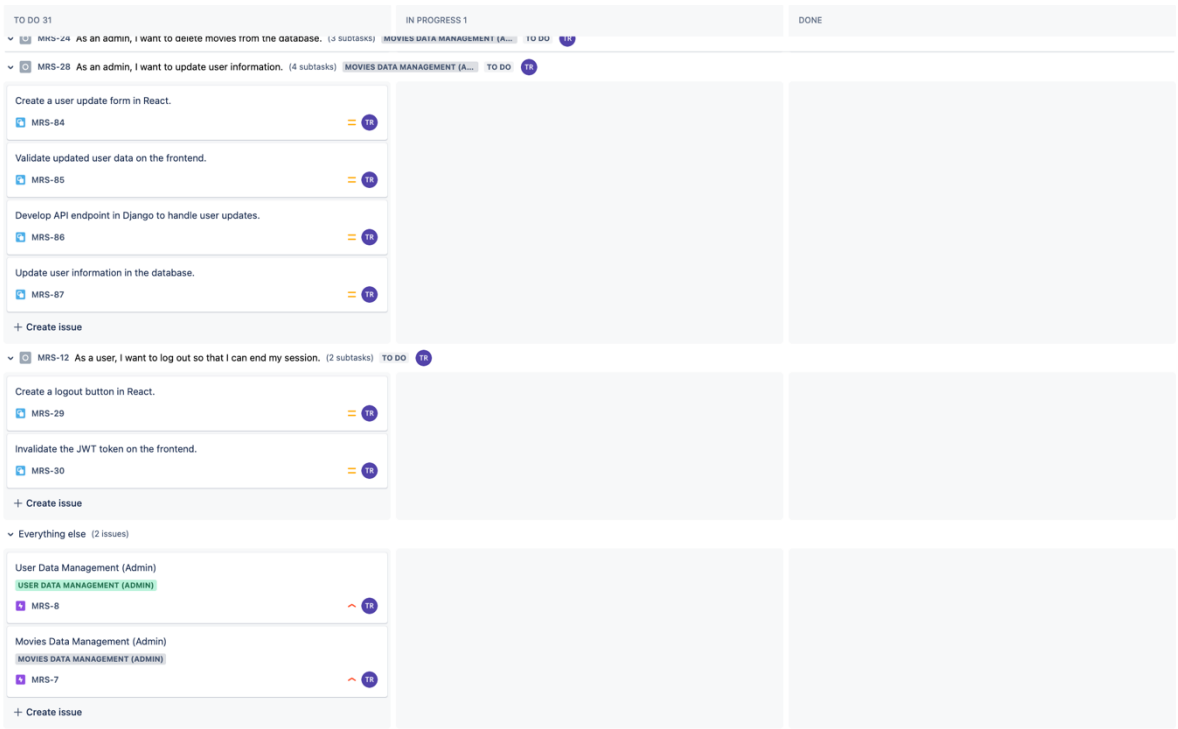
4.1.Planning

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.







4.2 Implementation

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

FR.6 : Admin Authentication

FR.7: Admin Users Data Management

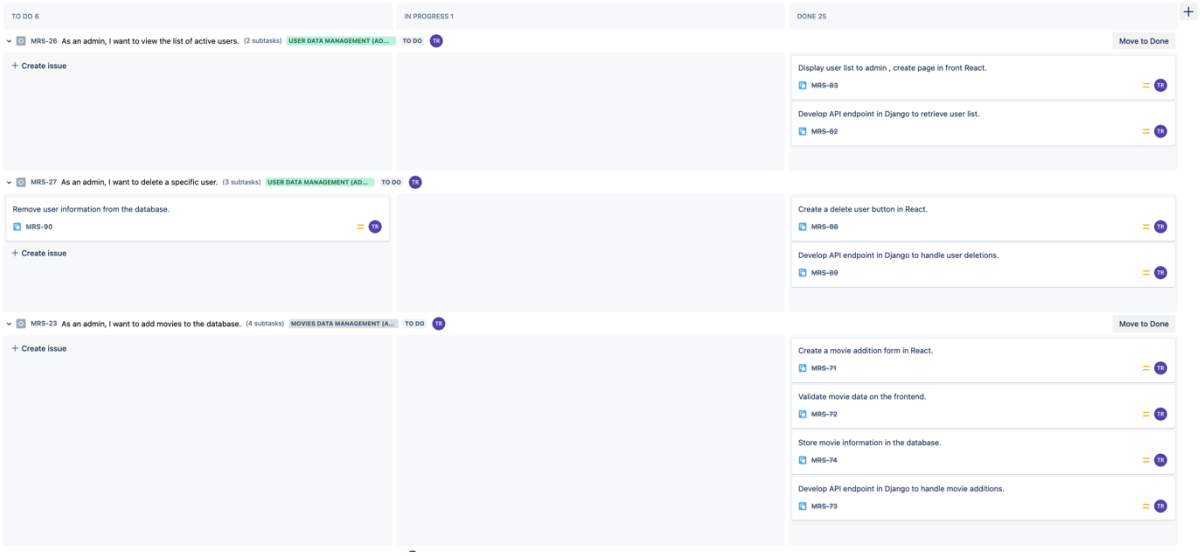
FR.8: Admin Movies Data management

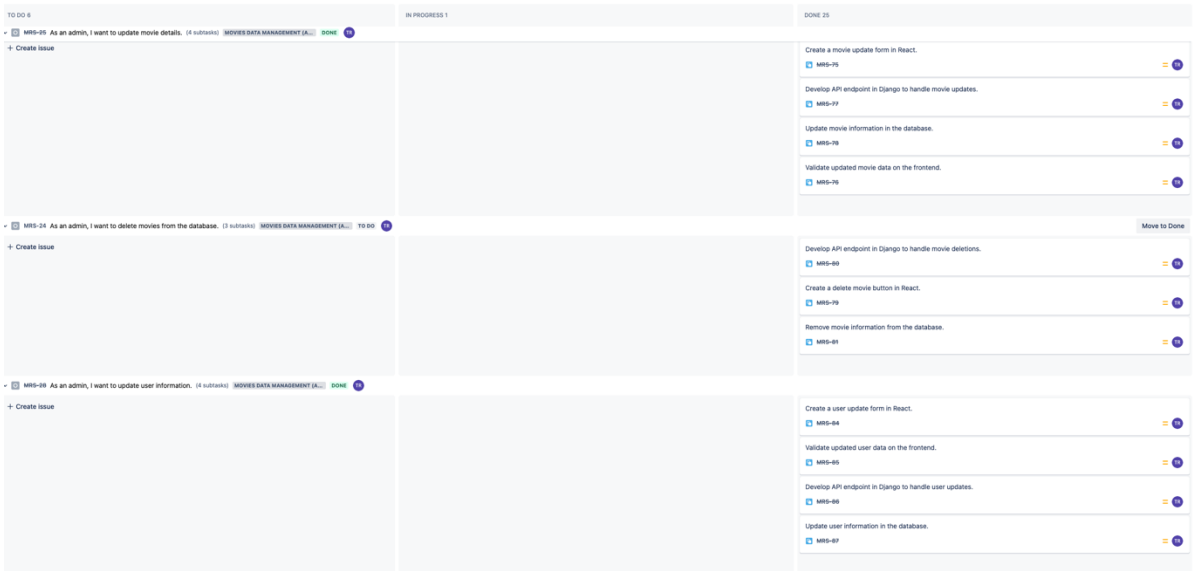
FR.6 is pretty similar to FR.1: user Authentication so it will not take long time toimplement.

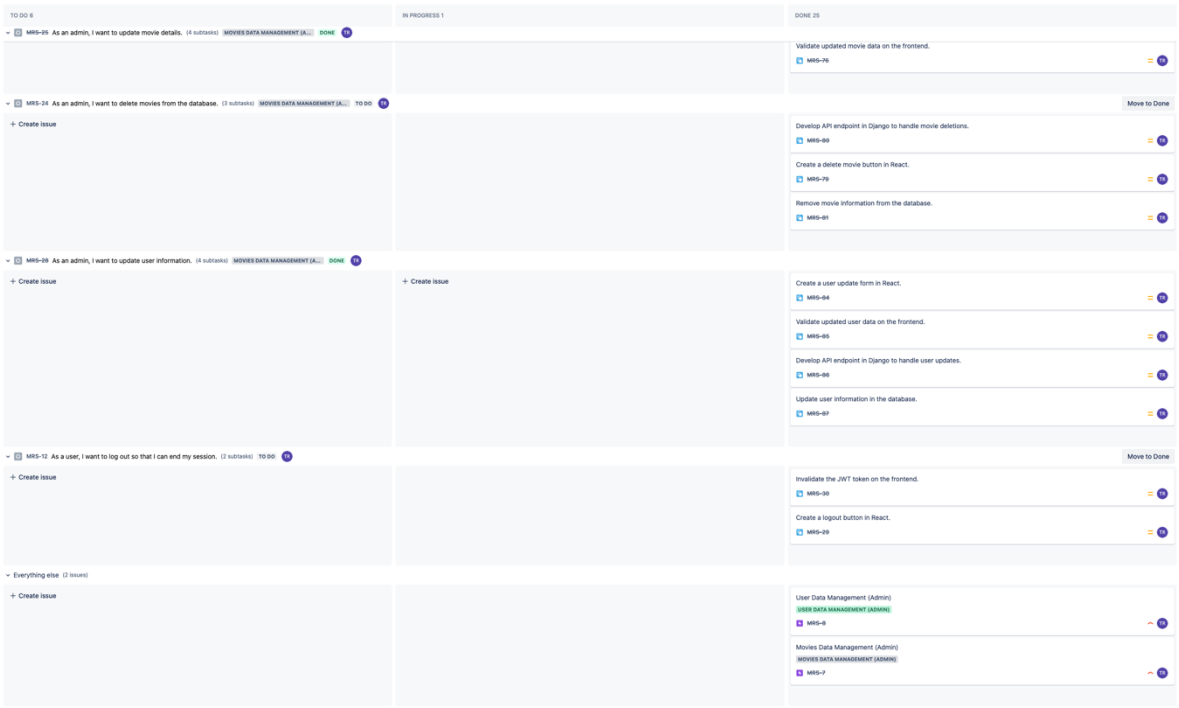
The other two FRs (7 and 8) also don't take too much time because there's no need to focus too much on UX when the UI just needs to be usable, not too beautiful.

4.2 Review

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

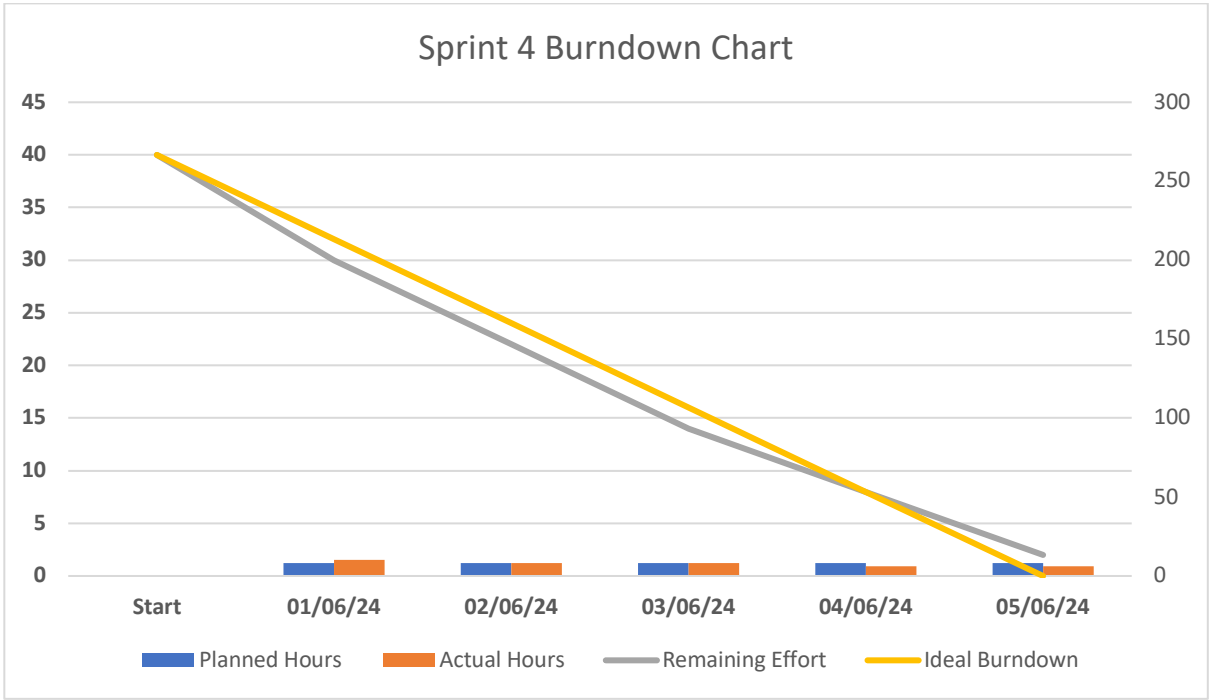






4.3 Retrospective

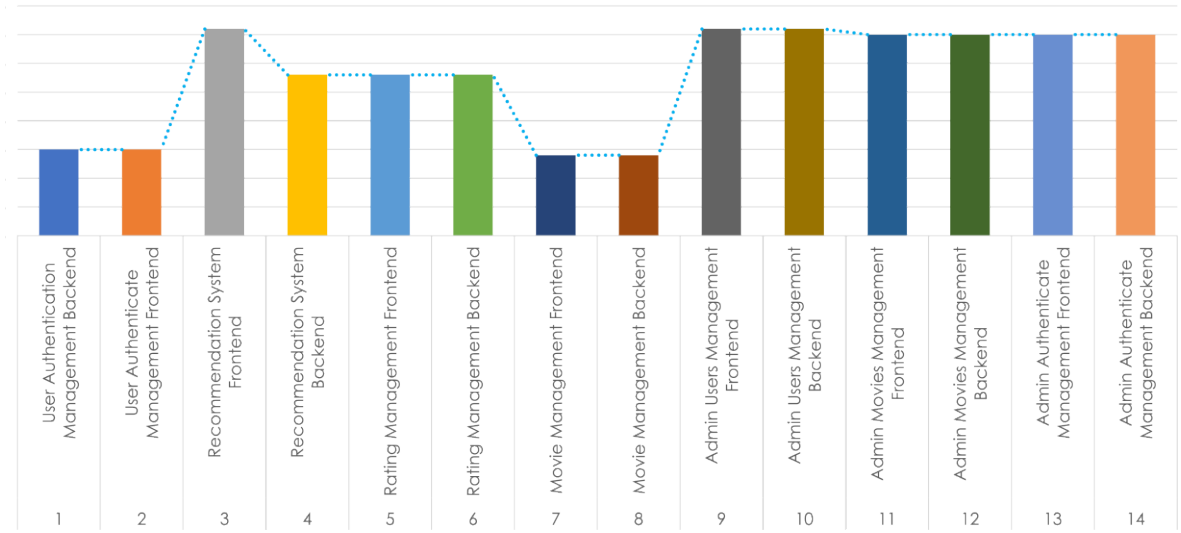
Release Burndown Chart



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

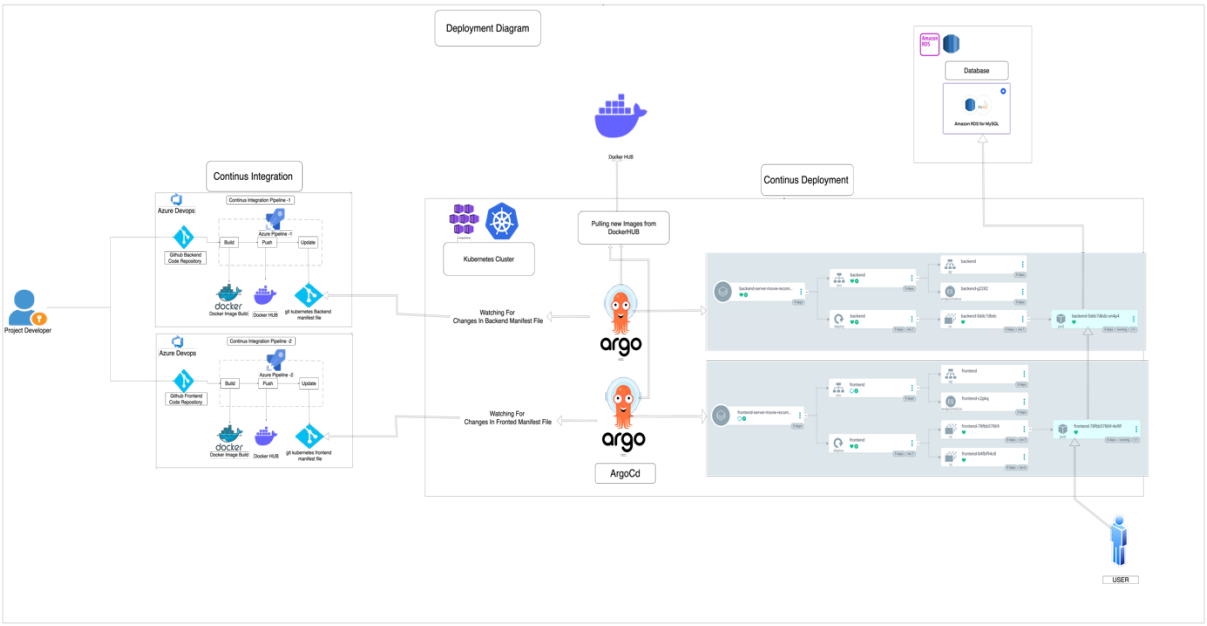
How does this application suggest movies to users? By a using machine learning model. Accepted.

To get more accurate suggestions, what does a user need to do? Give more ratings to moremovies, 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.

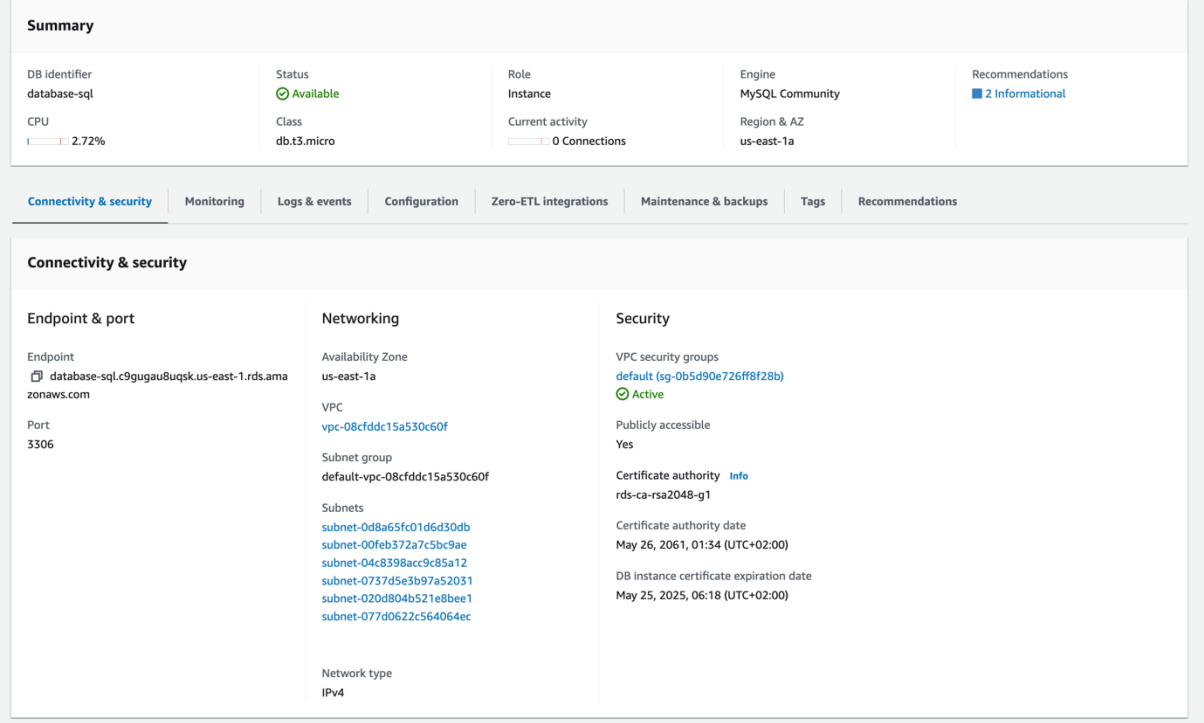
4. Database Deployment

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 .

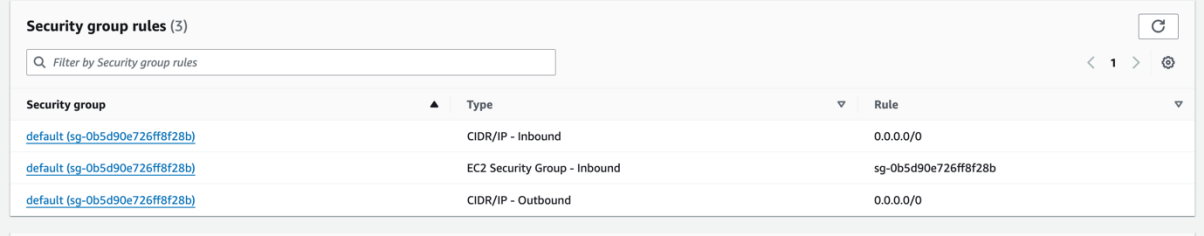
The region is us-east-1



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)

Filter by Security group rules

1

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,

```
mysql> SELECT * FROM ecomapp_movies;
+----+----------------------------+----------------------------+------------------------------------------------------------------------------------------------------------------------------+
| id | title                      | genre                      | image_url                                                                                                                    |
+----+----------------------------+----------------------------+------------------------------------------------------------------------------------------------------------------------------+
|  8 | 12 Angry Men               | Crime, Drama               | https://upload.wikimedia.org/wikipedia/commons/b/b5/12_Angry_Men_%281957_film_poster%29.jpg                                  |
|  9 | 12 Years a Slave           | Drama, History             | https://m.media-amazon.com/images/M/MV5BMjExMTEzODkyN15BMl5BanBnXkFtZTcwNTU4NTc4OQ@@._V1_.jpg                                |
| 10 | A Beautiful Mind           | Biography, Drama           | https://upload.wikimedia.org/wikipedia/en/b/b8/A_Beautiful_Mind_Poster.jpg                                                   |
| 11 | Avatar                     | Action, Adventure, Fantasy | https://lumiere-a.akamaihd.net/v1/images/avatar_800x1200_208c9665.jpeg?region=0%2C0%2C800%2C1200                             |
| 12 | Avengers: Infinity War     | Action, Adventure, Sci-Fi  | https://lumiere-a.akamaihd.net/v1/images/p_avengersinfinitywar_19871_cb171514.jpeg?region=0,0,540,810                        |
| 13 | Avengers: Endgame          | Action, Adventure, Sci-Fi" | https://www.ilcineocchio.it/cine/wp-content/uploads/2020/05/Avengers-Endgame.jpg                                             |
| 14 | Back to the Future         | Adventure, Comedy, Sci-Fi  | https://m.media-amazon.com/images/M/MV5BZmU0M2Y1OGUtZjIxNi00ZjBkLTg1MjgtOWIyNThiZWIwYjRiXkEyXkFqcGdeQXVyMTQxNzMzNDI@._V1_.jpg |
| 15 | Batman Begins              | Action, Adventure"         | https://m.media-amazon.com/images/I/619eBtaKX3L._AC_UF894,1000_QL80_DpWeblab_.jpg                                            |
| 16 | Casino Royale              | Action, Thriller           | https://m.media-amazon.com/images/M/MV5BYmI3MmMzMGMtNzc4Ni00YWQ4LWFkMDYtNjVlOWU3ZGZiNjY1XkEyXkFqcGdeQXVyNDQ2MTMzODA@._V1_.jpg |
| 17 | Coco                       | Animation, Adventure       | https://assets-in.bmscdn.com/iedb/movies/images/mobile/thumbnail/xlarge/coco-et00052365-15-10-2020-06-41-39.jpg              |
| 18 | The Dark Knight Rises      | Action, Adventure"         | https://m.media-amazon.com/images/M/MV5BMTk4ODQzNDY3Ml5BMl5BanBnXkFtZTcwODA0NTM4Nw@@._V1_FMjpg_UX1000_.jpg                   |
| 19 | Die Hard                   | Action, Thriller           | https://m.media-amazon.com/images/M/MV5BZjRlNDUxZjAtOGQ4OC00OTNlLTgxNmQtYTBmMDgwZmNmNjkxXkEyXkFqcGdeQXVyNzkwMjQ5NzM@._V1_.jpg |
| 20 | Django Unchained           | Drama                      | https://m.media-amazon.com/images/I/81IVfnsVtIL._AC_UF1000,1000_QL80_.jpg                                                    |
| 21 | Drive                      | Crime, Drama               | https://m.media-amazon.com/images/M/MV5BZjY5ZjQyMjMtMmEwOC00Nzc2LTllYTItMmU2MzJjNTg1NjY0XkEyXkFqcGdeQXVyNjQ1MTMzMDQ@._V1_.jpg |
| 22 | Fight Club"                | Drama"                     | https://m.media-amazon.com/images/M/MV5BMmEzNTkxYjQtZTc0MC00YTVjLTg5ZTEtZWMwOWVlYzY0NWIwXkEyXkFqcGdeQXVyNzkwMjQ5NzM@._V1_.jpg |
| 23 | Forrest Gump               | Drama, Romance             | https://upload.wikimedia.org/wikipedia/en/6/67/Forrest_Gump_poster.jpg                                                       |
| 24 | Frozen                     | Animation, Comedy          | https://lumiere-a.akamaihd.net/v1/images/p_frozen_18373_3131259c.jpeg?region=0%2C0%2C540%2C810                               |
| 25 | Gladiator                  | Action, Adventure,Drama    | https://m.media-amazon.com/images/I/51GA6V6VE1L._AC_UF894,1000_QL80_.jpg                                                     |
| 26 | Gone Girl                  | Mistery , Thriller         | https://m.media-amazon.com/images/I/41QQLt90QsL._AC_UF1000,1000_QL80_.jpg                                                    |
| 27 | Good Will Hunting          | Drama ,Romance             | https://m.media-amazon.com/images/M/MV5BOTI0MzcxMTYtZDVkMy00NjY1LTgyMTYtZmUxN2M3NmQ2NWJhXkEyXkFqcGdeQXVyMTQxNzMzNDI@._V1_.jpg |
| 28 | Gravity                    | sci-fi ,thriller           | https://m.media-amazon.com/images/M/MV5BNjE5MzYwMzYxMF5BMl5BanBnXkFtZTcwOTk4MTk0OQ@@._V1_.jpg                                |
| 29 | Guardians of the Galaxy    | Action, comedy             | https://m.media-amazon.com/images/M/MV5BNDIzMTk4NDYtMjg5OS00ZGI0LWJhZDYtMzdmZGY1YWU5ZGNkXkEyXkFqcGdeQXVyMTI5NzUyMTIz._V1_.jpg |
| 30 | Home Alone                 | comedy                     | https://lumiere-a.akamaihd.net/v1/images/image_ed8d7a4e.jpeg?region=0%2C0%2C800%2C1200                                       |
| 31 | Inception                  | Adventure, Sci-Fi          | https://m.media-amazon.com/images/M/MV5BMjAxMzY3NjcxNF5BMl5BanBnXkFtZTcwNTI5OTM0Mw@@._V1_.jpg                                |
| 32 | Inside out                 | Animation, Comedy          | https://m.media-amazon.com/images/M/MV5BOTgxMDQwMDk0OF5BMl5BanBnXkFtZTgwNjU5OTg2NDE@._V1_.jpg                                |
| 33 | Intersteller               | Adventure , Sci-Fi         | https://m.media-amazon.com/images/I/91vIHsL-zjL._AC_UF1000,1000_QL80_DpWeblab_.jpg                                           |
| 34 | Joker                      | Crime, Drama ,Thriller     | https://upload.wikimedia.org/wikipedia/en/e/e1/Joker_%282019_film%29_poster.jpg                                              |
| 35 | Memento                    | Mystry, Thriller           | https://m.media-amazon.com/images/M/MV5BZTcyNjk1MjgtOWI3Mi00YzQwLWI5MTktMzY4ZmI2NDAyNzYzXkEyXkFqcGdeQXVyNjU0OTQ0OTY@._V1_.jpg |
| 36 | Oldboy                     | Action,Drama               | https://m.media-amazon.com/images/M/MV5BMTI3NTQyMzU5M15BMl5BanBnXkFtZTcwMTM2MjgyMQ@@._V1_.jpg                                |
| 37 | Parasite                   | Drama, thriller            | https://pad.mymovies.it/filmclub/2019/03/022/locandina.jpg                                                                   |
| 38 | Pulp Fiction               | Crime,Drama                | https://m.media-amazon.com/images/M/MV5BNGNhMDIzZTUtNTBlZi00MTRlLWFjM2ItYzViMjE3YzI5MjljXkEyXkFqcGdeQXVyNzkwMjQ5NzM@._V1_.jpg |
| 39 | Schindler's List           | Biography, Drama           | https://m.media-amazon.com/images/M/MV5BNDE4OTMxMTctNmRhYy00NWE2LTg3YzItYTk3M2UwOTU5Njg4XkEyXkFqcGdeQXVyNjU0OTQ0OTY@._V1_.jpg |
| 40 | Shutter Island             | Mystry, Thriller           | https://m.media-amazon.com/images/M/MV5BYzhiNDkyNzktNTZmYS00ZTBkLTk2MDAtM2U0YjU1MzgxZjgzXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_.jpg |
| 41 | The Avengers"              | Action, Sci-Fi             | https://m.media-amazon.com/images/M/MV5BNDYxNjQyMjAtNTdiOS00NGYwLWFmNTAtNThmYjU5ZGI2YTI1XkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_.jpg |
| 42 | The Lion King              | Animation, Comedy          | https://lumiere-a.akamaihd.net/v1/images/p_thelionking_19752_1_0b9de87b.jpeg?region=0%2C0%2C540%2C810                        |
| 43 | The Pianist                | Biography, Drama           | https://m.media-amazon.com/images/I/61bvLx+opGL._AC_UF1000,1000_QL80_.jpg                                                    |
| 44 | The Shawshank Redemption   | Drama                      | https://m.media-amazon.com/images/M/MV5BNDE3ODcxYzMtY2YzZC00NmNlLWJiNDMtZDViZWM2MzIxZDYwXkEyXkFqcGdeQXVyNjAwNDUxODI@._V1_.jpg |
| 45 | The Godfather"             | Crime, Drama               | https://m.media-amazon.com/images/M/MV5BM2MyNjYxNmUtYTAwNi00MTYxLWJmNWYtYzZlODY3ZTk3OTFlXkEyXkFqcGdeQXVyNzkwMjQ5NzM@._V1_.jpg |
| 46 | The Truman Show            | comedy,Drama, Sci_fi       | https://m.media-amazon.com/images/I/71zEnyA+49L._AC_UF1000,1000_QL80_DpWeblab_.jpg                                           |
| 47 | The Wolf of Wall Street    | Crime, Drama ,Comedy       | https://m.media-amazon.com/images/M/MV5BMjIxMjgxNTk0MF5BMl5BanBnXkFtZTgwNjIyOTg2MDE@._V1_.jpg                                |
| 48 | Toy Story                  | Animation, Comedy          | https://m.media-amazon.com/images/M/MV5BMDU2ZWJlMjktMTRhMy00ZTA5LWEzNDgtYmNmZTEwZTViZWJkXkEyXkFqcGdeQXVyNDQ2OTk4MzI@._V1_.jpg |
+----+----------------------------+----------------------------+------------------------------------------------------------------------------------------------------------------------------+
41 rows in set (0.47 sec)

mysql>
```

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:

```
pyodbc==5.0.1
PyMySQL==1.1.1
mssql-django==1.5
mysqlclient==2.0.3
sqlparse==0.3.1
urllib3==1.25.9
whitenoise==5.1.0
djangorestframework==3.15.1
djangorestframework-simplejwt==5.3.1
django-cors-headers==3.7.0
boto3==1.14.1
botocore==1.17.1
django==3.2.11
django-storages==1.9.1
docutils==0.15.2
gunicorn==20.0.4
idna==2.9
jmespath==0.10.0
numpy
pandas
Pillow==7.1.2
python-dateutil==2.8.1
pytz==2020.1
recommend==0.2.2
requests==2.23.0
s3transfer==0.3.3
scipy
six
```

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

Pip will I install 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:

```
# Stage 1: Build stage
FROM python:3.8.18 as build

# Set the working directory to /movie_recommender
WORKDIR /movie_recommender

# Copy the requirements file to the working directory
COPY ./requirements.txt /movie_recommender/

# Install any needed packages specified in requirements.txt
RUN apt-get update && apt-get install -y gcc && \
    pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application files into the container at /movie_recommender
COPY . /movie_recommender

# Stage 2: Final stage
FROM python:3.8.18

# Set the working directory to /movie_recommender
WORKDIR /movie_recommender

# Copy only the installed dependencies from the build stage
COPY --from=build /usr/local/lib/python3.8/site-packages /usr/local/lib/python3.8/site-packages
COPY --from=build /usr/local/bin /usr/local/bin

# Copy the rest of the application files
COPY . /movie_recommender

# Define environment variable
ENV NAME World

# Expose port 5015 to the outside world
EXPOSE 5015

# Run app.py when the container launches
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 .

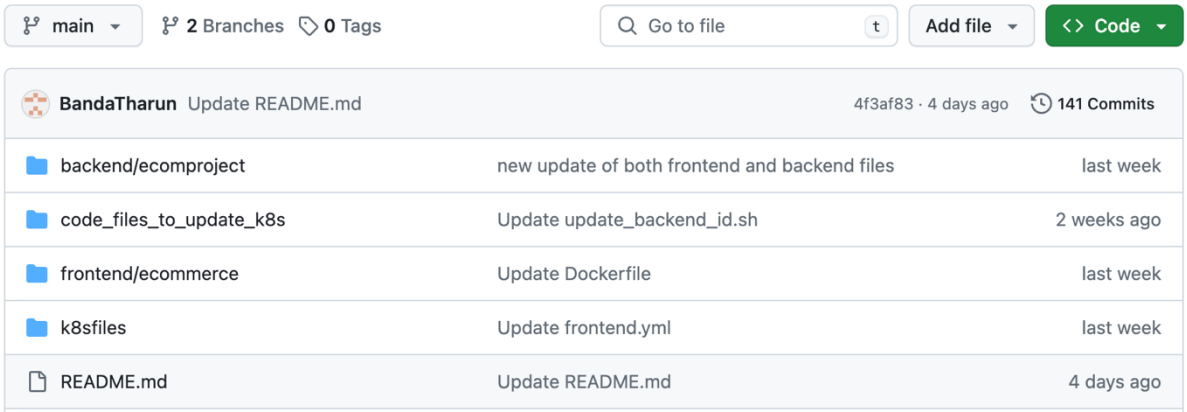
Configuring frontend server package.json file and including backend server address to establish connection,

```
"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

```
# Use a Node.js base image
FROM node:14-alpine

# Set working directory inside the container
WORKDIR /app

# Copy package.json and package-lock.json (if available)
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application
COPY . .

# Build the React app
RUN npm run build

# Expose port 3000
EXPOSE 3000

# Command to run the application
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.



Repository Structure

```
├── README.md
├── backend
│   └── ecomproject
│       ├── db.sqlite3
│       ├── dockerfile
│       ├── ecomapp
│       ├── ecomproject
│       ├── manage.py
│       ├── passsword
│       ├── requirements.txt
│       └── static
├── code_files_to_update_k8s
│   ├── update_backend_id.sh
│   └── update_frontend_id.sh
├── frontend
│   └── ecommerce
│       ├── Dockerfile
│       ├── README.md
│       ├── build
│       ├── node_modules
│       ├── package-lock.json
│       ├── package.json
│       ├── public
│       └── src
└── k8sfiles
    ├── k8sbackend
    │   └── backend.yml
    └── k8sfrontend
        └── frontend.yml
```

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.

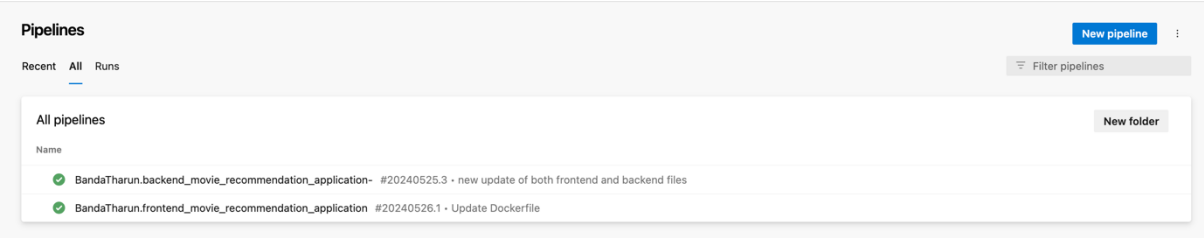
    Tasks of pipeline:
    1. Build Docker Image.
    2. Push Docker Image to Docker Hub.
    3. Update Docker Image Tag in Kubernetes Deployment Files.

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:

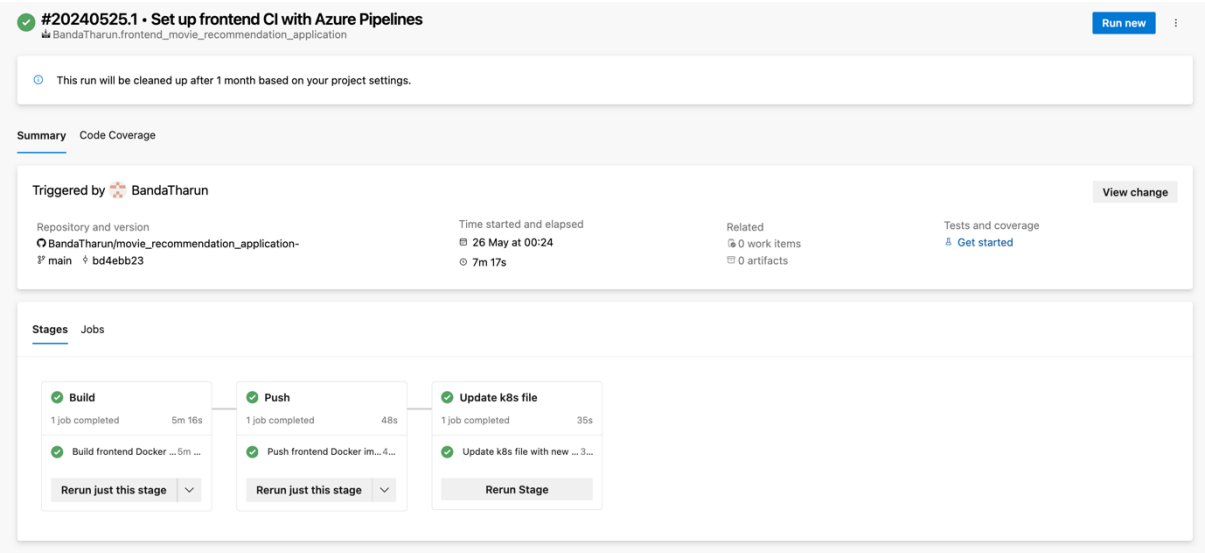
CI Pipeline Setup in Azure DevOps

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



Pipeline -1 : BandaTharun.frontend_movie_recommendation_application

Tasks of the Pipeline :

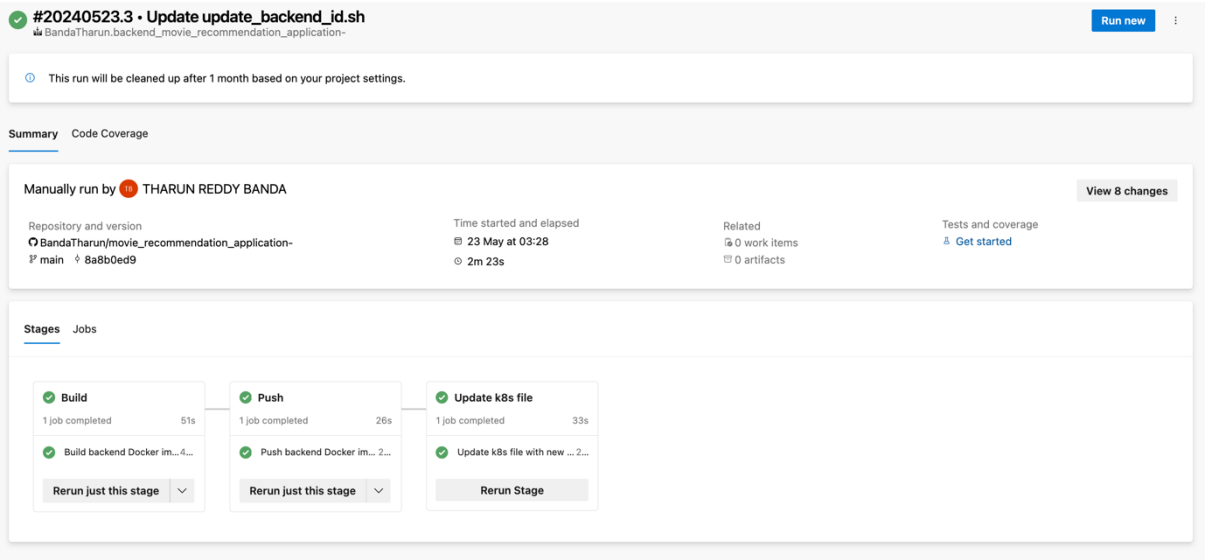


Pipeline-1.Yml configuration code file that does the above tasks,

```yaml


trigger:
  paths:
    include:
      - frontend/ecomproject/*

resources:
- repo: self

variables:
  tag: '$(Build.BuildId)'
  frontendImageName: 'frontend_image'

pool:
  name: 'agentpool-to-build-container'

stages:
- stage: Build
  displayName: Build
  jobs:
  - job: BuildFrontend
    displayName: Build frontend Docker image
    steps:
      Settings
    - task: Docker@2
      displayName: Login to Docker Hub
      inputs:
        command: login
        containerRegistry: 'dockerregestryconnection'

      Settings
    - task: Docker@2
      displayName: Build frontend Docker image
      inputs:
        containerRegistry: 'dockerregestryconnection'
        repository: 'tharun9705/movie_recommendation_applications_repository'
        command: 'build'
        Dockerfile: '$(Build.SourcesDirectory)/frontend/ecommerce/Dockerfile'
        tags: '$(frontendImageName)$(tag)'

- stage: Push
  displayName: Push
  jobs:
  - job: PushFrontend
    displayName: Push frontend Docker image
    steps:
      Settings
    - task: Docker@2
      displayName: Push Docker image
      inputs:
        containerRegistry: 'dockerregestryconnection'
        repository: 'tharun9705/movie_recommendation_applications_repository'
        command: 'push'
        tags: '$(frontendImageName)$(tag)'

- stage: UpdateK8sFile
  displayName: Update k8s file
  jobs:
  - job: UpdateK8s
    displayName: Update k8s file with new Docker ID
    steps:
      Settings
    - task: ShellScript@2
      displayName: Update Kubernetes Configuration
      inputs:
        scriptPath: 'code_files_to_update_k8s/update_frontend_id.sh'
        args: '$(frontendImageName)$(tag)'
```

Pipeline -2 : BandaTharun.backend_movie_recommendation_application-

Tasks of the Pipeline :



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

```yaml

# Docker
# Build a Docker image
# https://docs.microsoft.com/azure/devops/pipelines/languages/docker

trigger:
  paths:
    include:
      - backend/ecomproject/*

resources:
- repo: self

variables:
  tag: '$(Build.BuildId)'
  key: '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'
  backendImageName: 'backend_image'


pool:
  name: 'agentpool-to-build-container'

stages:
- stage: Build
  displayName: Build
  jobs:
  - job: BuildBackend
    displayName: Build backend Docker image
    steps:
    Settings
    - task: Docker@2
      displayName: Login to Docker Hub
      inputs:
        command: login
        containerRegistry: 'dockerregestryconnection'

    Settings
    - task: Docker@2
      displayName: Build backend Docker image
      inputs:
        containerRegistry: 'dockerregestryconnection'
        repository: 'tharun9705/movie_recommendation_applications_repository'
        command: 'build'
        Dockerfile: '$(Build.SourcesDirectory)/backend/ecomproject/dockerfile'
        tags: '$(backendImageName)$(tag)'

- stage: Push
  displayName: Push
  jobs:
  - job: PushBackend
    displayName: Push backend Docker image
    steps:
    Settings
    - task: Docker@2
      displayName: Push Docker image
      inputs:
        containerRegistry: 'dockerregestryconnection'
        repository: 'tharun9705/movie_recommendation_applications_repository'
        command: 'push'
        tags: '$(backendImageName)$(tag)'

- stage: UpdateK8sFile
  displayName: Update k8s file
  jobs:
  - job: UpdateK8s
    displayName: Update k8s file with new Docker ID
    steps:
    Settings
    - task: ShellScript@2
      displayName: Update Kubernetes Configuration
      inputs:
        scriptPath: 'code_files_to_update_k8s/update_backend_id.sh'
        args: '$(backendImageName)$(tag) $(key)'


```

CD Pipeline Setup AKS cluster :

The Continuous Deployment pipeline ensures that any changes 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 .

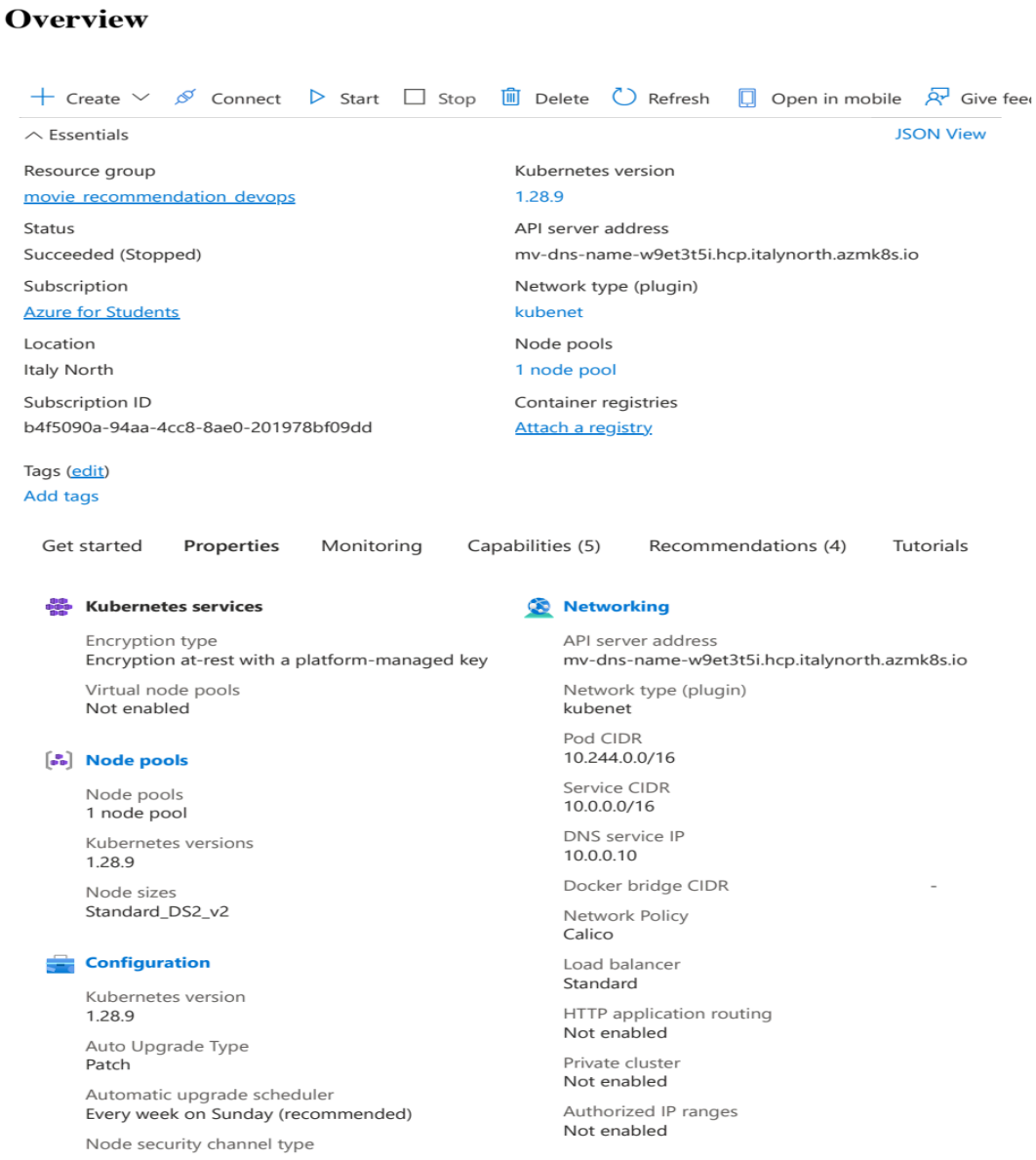
frontend.yml file

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: tharun9705/movie_recommendation_applications_repository:frontend_image49
        ports:
        - containerPort: 3000
---
apiVersion: v1
kind: Service
metadata:
  name: frontend
spec:
  selector:
    app: frontend
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
  type: LoadBalancer
```

Backend.yml file

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
      - name: backend
        image: tharun9705/movie_recommendation_applications_repository:backend_image47
        ports:
        - containerPort: 5015
---
apiVersion: v1
kind: Service
metadata:
  name: backend
spec:
  selector:
    app: backend
  ports:
    - protocol: TCP
      port: 5015
      targetPort: 5015
  type: ClusterIP
```

1. **Create AKS Cluster**:
    1. Setting up an AKS cluster in the Azure portal.



2. **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.

```yaml


apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: backend-app
  namespace: argocd
spec:
  project: default
  source:
    repoURL: 'https://github.com/BandaTharun/movie_recommendation_application-.git'
    path: k8sfiles/k8sbackend
    targetRevision: HEAD
  destination:
    server: 'https://kubernetes.default.svc'
    namespace: default
  syncPolicy:
    automated:
      prune: true
      selfHeal: true


```

    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.

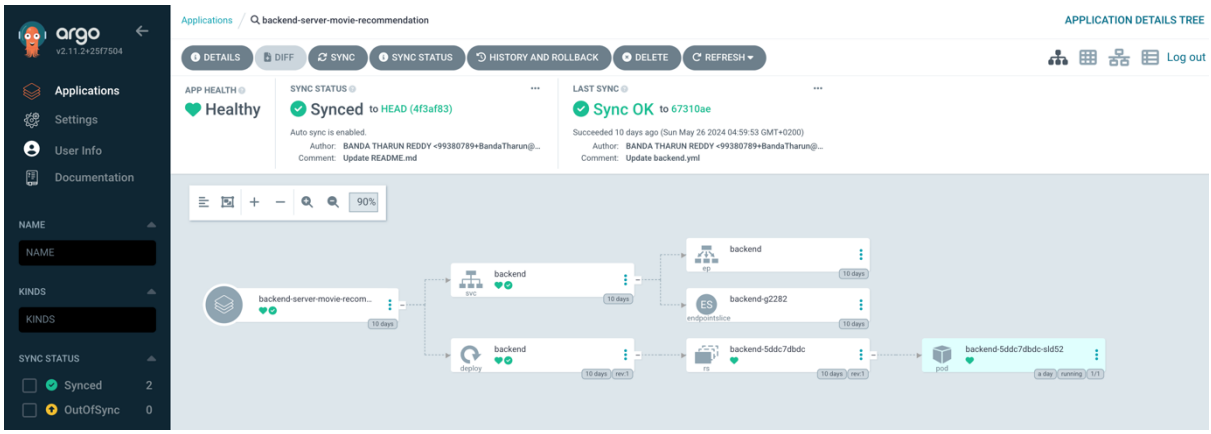
```yaml

apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: frontend-app
  namespace: argocd
spec:
  project: default
  source:
    repoURL: 'https://github.com/BandaTharun/movie_recommendation_application-.git'
    path: k8sfiles/k8sfrontend
    targetRevision: HEAD
  destination:
    server: 'https://kubernetes.default.svc'
    namespace: default
  syncPolicy:
    automated:
      prune: true
      selfHeal: true


```

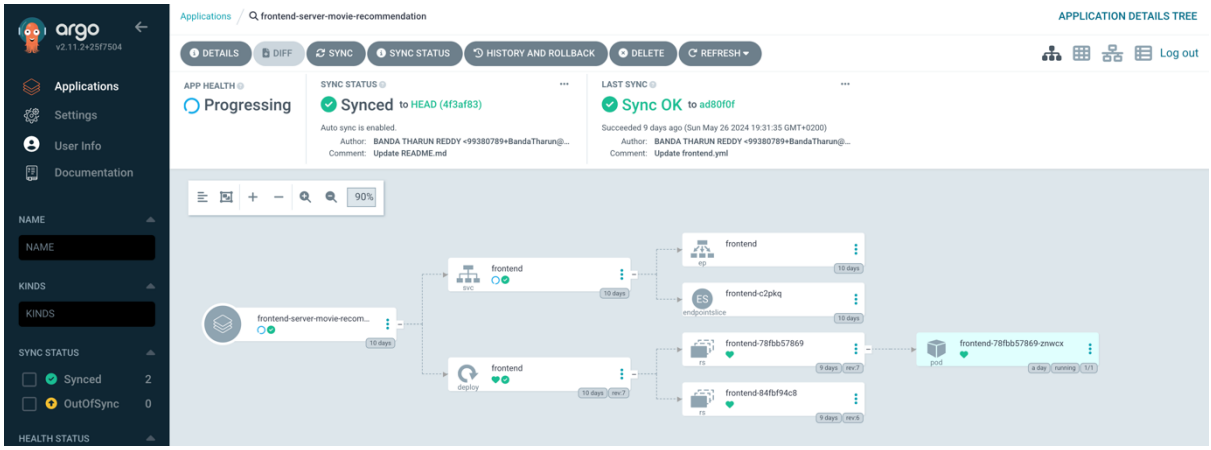
3. Apply the Application Definitions:

```
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:

2. 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.

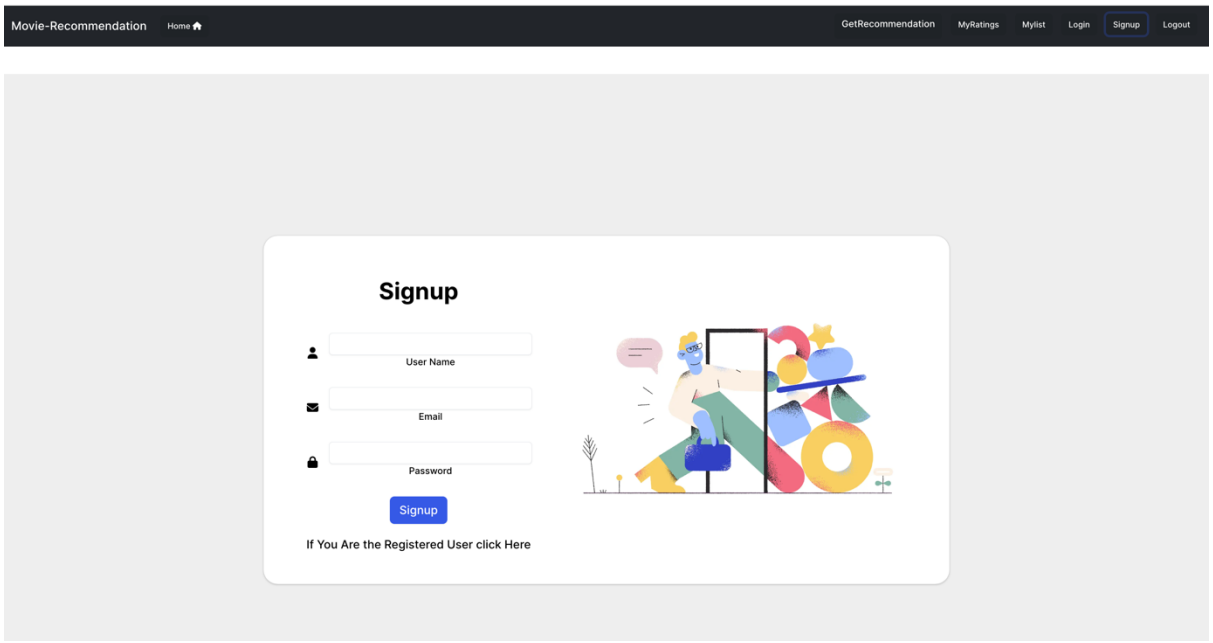
7. Recommendations Generation:

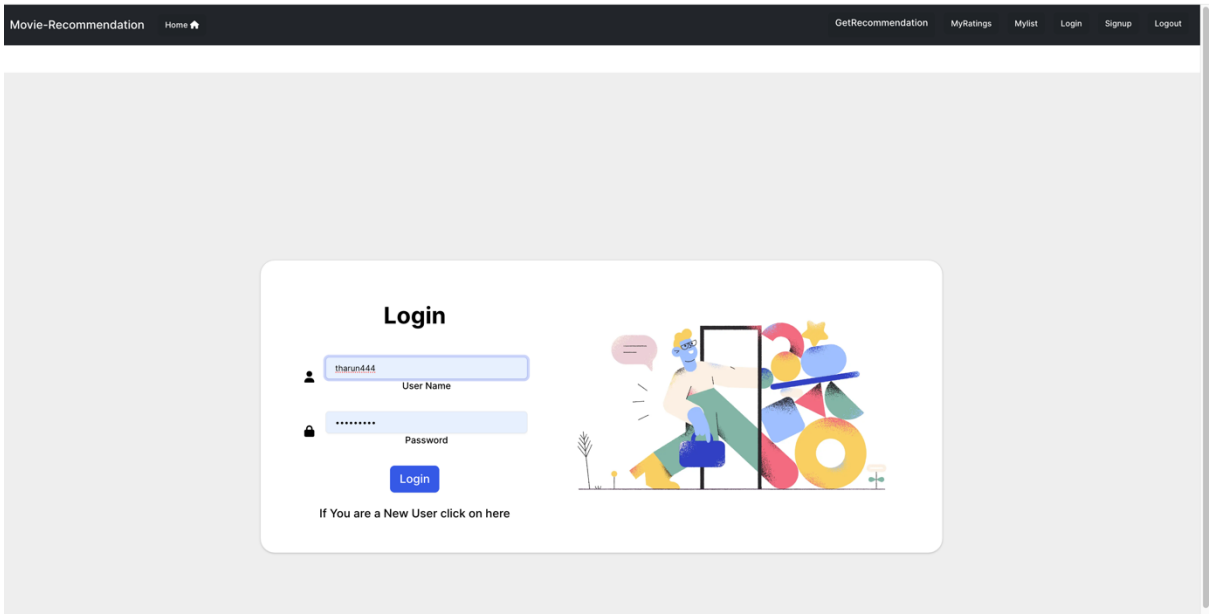
1. The movies are summed and sorted to find the most similar movies to generate recommendations. Movies the user has already watched are filtered out.

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


Movie-Recommendation
Home
GetRecommendation
MyRatings
Mylist
Login
Signup
Logout
Signup
User Name
Email
Password
Signup
If You Are the Registered User click Here


Movie-Recommendation
Home
GetRecommendation
MyRatings
Mylist
Login
Signup
Logout
Login
tharun444
User Name
Password
Login
If You are a New User click on here

Movie-Recommendation Home GetRecommendation MyRatings Mylist Login Signup Logout

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

Casino Royale

Coco

Coco

The Dark Knight Rises

The Dark Knight Rises

Die Hard

Die Hard

Django Unchained

Drive

Drive

Fight Club"

Fight Club"

Forrest Gump

Forrest Gump

Movie-Recommendation Home GetRecommendation MyRatings Mylist Login Signup Logout

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 Presonal Rating

★★★★★

Submit Rating

Add to My List

Movie-Recommendation Home GetRecommendation MyRatings Mylist Login Signup Logout

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 Presonal Rating

★★★★★

Submit Rating

Add to My List

Movie-Recommendation Home GetRecommendation MyRatings Mylist Login Signup Logout

MyRatings

The Dark Knight Rises

Click on the Checkbox To Delete the movie from MyRatings

★★★★★

Update Rating

Movie-Recommendation Home GetRecommendation MyRatings Mylist Login Signup Logout

Recommendations

Back to the Future

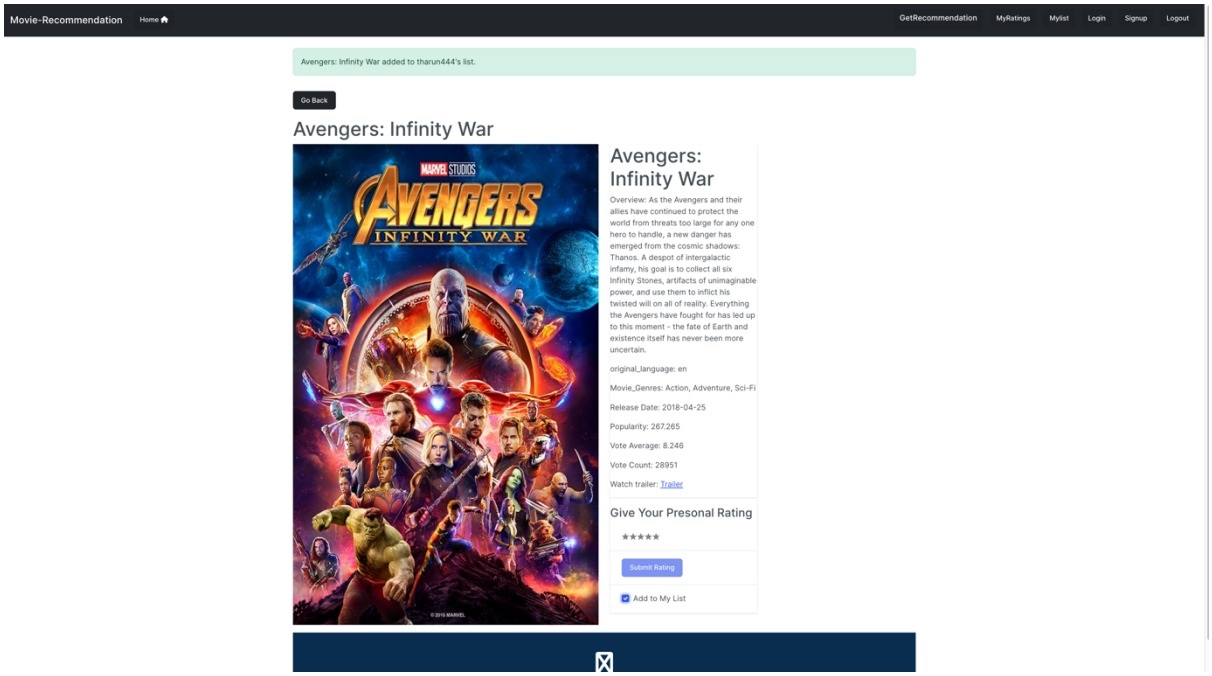
Coco

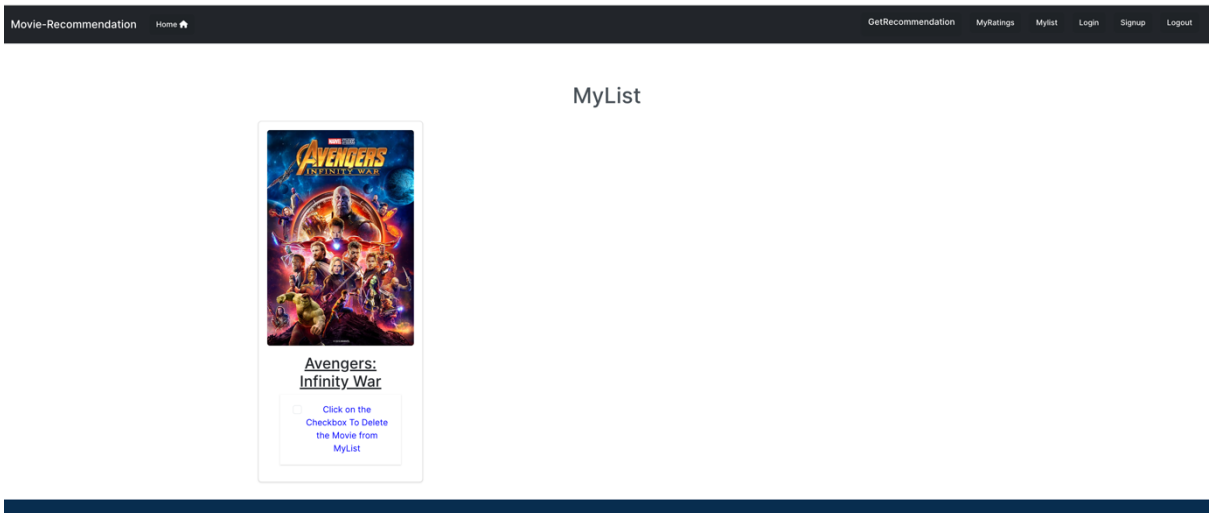
Intersteller

Memento

Shutter Island

The Lion King


Movie-Recommendation
Home
GetRecommendation
MyRatings
Mylist
Login
Signup
Logout
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 from threats too large for any one hero to handle, a new danger has emerged from the cosmic shadows: Thanos. A despot of intergalactic infamy, his goal is to collect all six Infinity Stones, artifacts of unimaginable power, and use them to inflict his twisted will on all of reality. Everything the Avengers have fought for has led up to this moment - the fate of Earth and existence itself has never been more uncertain.
original_language: en
Movie_Genres: Action, Adventure, Sci-Fi
Release Date: 2018-04-25
Popularity: 267.265
Vote Average: 8.246
Vote Count: 28951
Watch trailer: Trailer
Give Your Presonal Rating
★★★★★
Submit 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

Admin Name

Admin Password

Login

Movie-Recommendation Admin Page Main Page

Manage Movies Manage Users Home Website Page Logout

WELCOME TO ADMIN PAGE

- Manage Movies
- Manage Users

© 2020 Movie-Recommendation

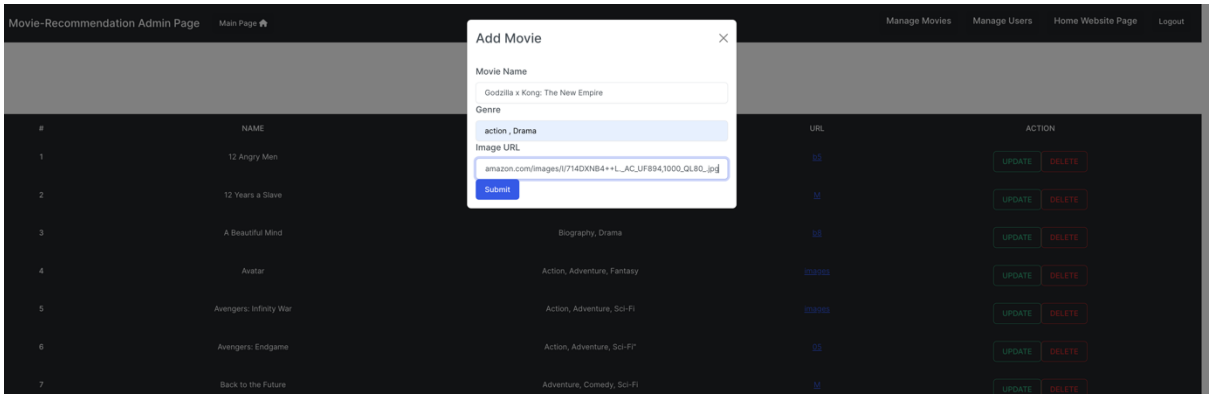
Movie-Recommendation Admin Page Main Page

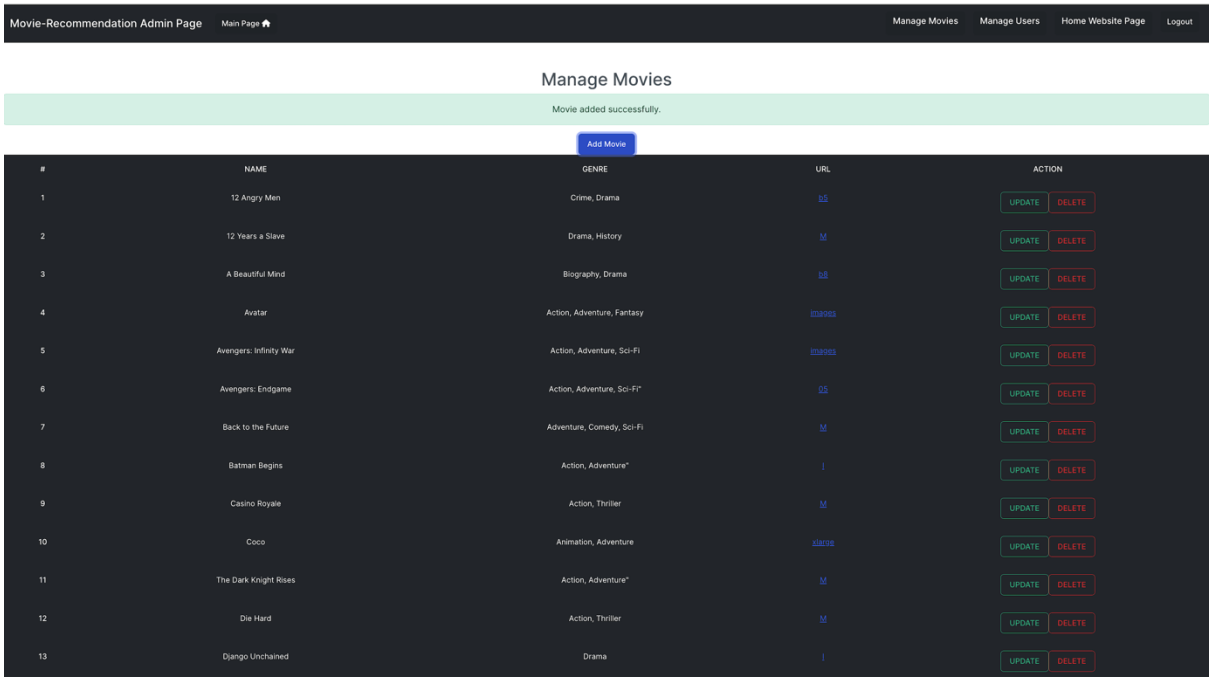
Manage Movies Manage Users Home Website Page Logout

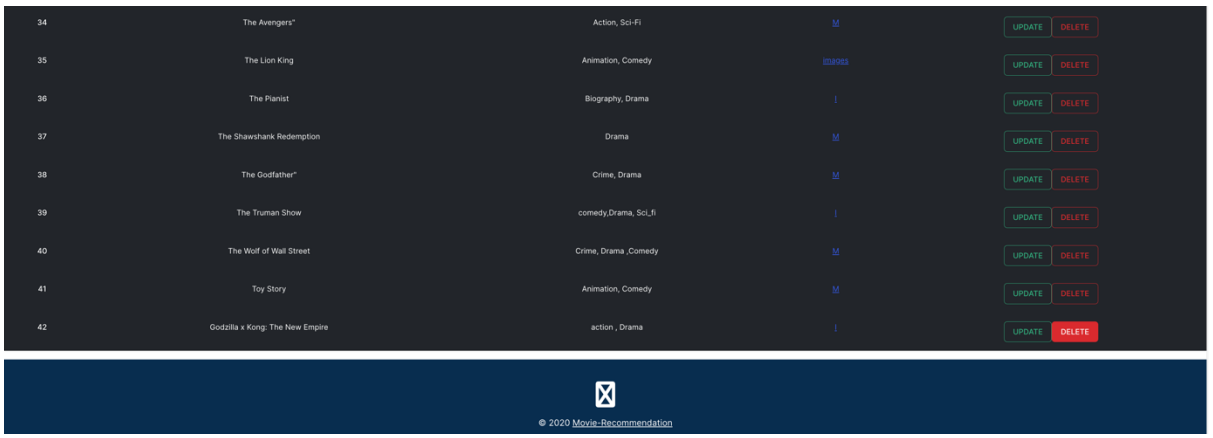
Manage Movies

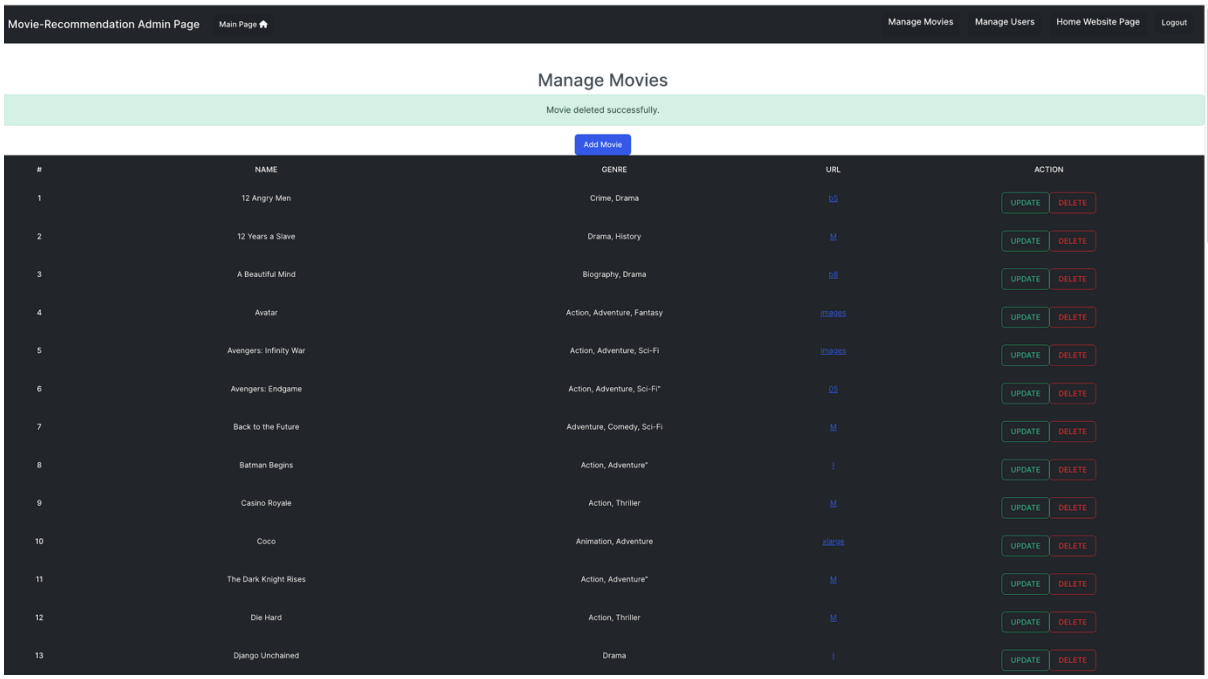
Add Movie

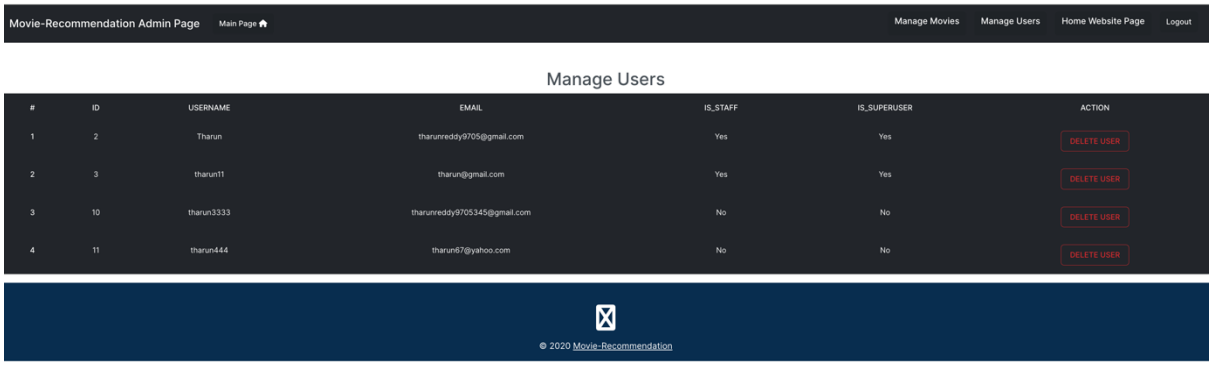
#	NAME	GENRE	URL	ACTION
1	12 Angry Men	Crime, Drama	b5	UPDATE DELETE
2	12 Years a Slave	Drama, History	M	UPDATE DELETE
3	A Beautiful Mind	Biography, Drama	b8	UPDATE DELETE
4	Avatar	Action, Adventure, Fantasy	images	UPDATE DELETE
5	Avengers: Infinity War	Action, Adventure, Sci-Fi	images	UPDATE DELETE
6	Avengers: Endgame	Action, Adventure, Sci-Fi"	05	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

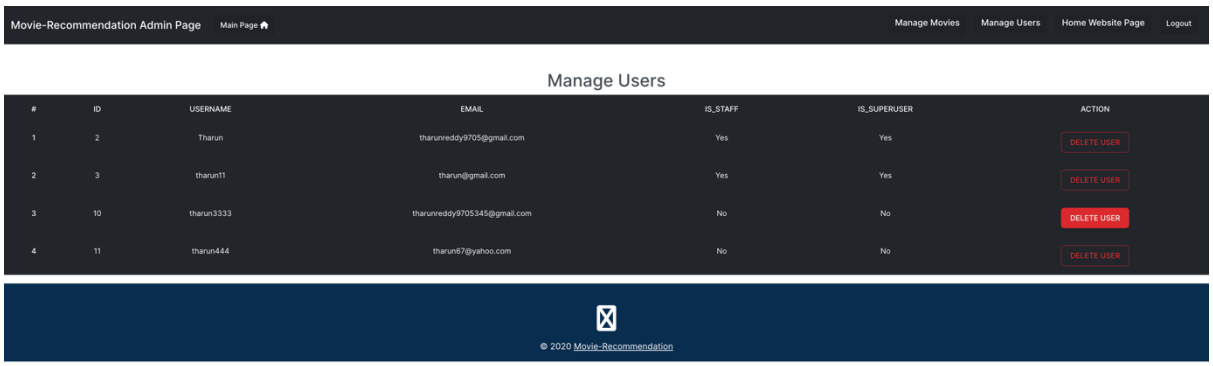

Movie-Recommendation Admin Page
Main Page
Manage Movies
Manage Users
Home Website Page
Logout
Add Movie
Movie Name
Godzilla x Kong: The New Empire
Genre
action , Drama
Image URL
amazon.com/images/I/714DXNB4++L._AC_UF894,1000_QL80_.jpg
Submit
NAME URL ACTION
1 12 Angry Men UPDATE DELETE
2 12 Years a Slave UPDATE DELETE
3 A Beautiful Mind Biography, Drama UPDATE DELETE
4 Avatar Action, Adventure, Fantasy UPDATE DELETE
5 Avengers: Infinity War Action, Adventure, Sci-Fi UPDATE DELETE
6 Avengers: Endgame Action, Adventure, Sci-Fi" UPDATE DELETE
7 Back to the Future Adventure, Comedy, Sci-Fi 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 UPDATE DELETE
2 12 Years a Slave Drama, History UPDATE DELETE
3 A Beautiful Mind Biography, Drama UPDATE DELETE
4 Avatar Action, Adventure, Fantasy UPDATE DELETE
5 Avengers: Infinity War Action, Adventure, Sci-Fi UPDATE DELETE
6 Avengers: Endgame Action, Adventure, Sci-Fi" UPDATE DELETE
7 Back to the Future Adventure, Comedy, Sci-Fi UPDATE DELETE
8 Batman Begins Action, Adventure" UPDATE DELETE
9 Casino Royale Action, Thriller UPDATE DELETE
10 Coco Animation, Adventure UPDATE DELETE
11 The Dark Knight Rises Action, Adventure" UPDATE DELETE
12 Die Hard Action, Thriller UPDATE DELETE
13 Django Unchained Drama UPDATE DELETE


34 The Avengers" Action, Sci-Fi UPDATE DELETE
35 The Lion King Animation, Comedy UPDATE DELETE
36 The Pianist Biography, Drama UPDATE DELETE
37 The Shawshank Redemption Drama UPDATE DELETE
38 The Godfather" Crime, Drama UPDATE DELETE
39 The Truman Show comedy,Drama, Sci_fi UPDATE DELETE
40 The Wolf of Wall Street Crime, Drama ,Comedy UPDATE DELETE
41 Toy Story Animation, Comedy UPDATE DELETE
42 Godzilla x Kong: The New Empire action , Drama UPDATE DELETE
© 2020 Movie-Recommendation


Movie-Recommendation Admin Page
Main Page
Manage Movies
Manage Users
Home Website Page
Logout
Manage Movies
Movie deleted successfully.
Add Movie
NAME GENRE URL ACTION
1 12 Angry Men Crime, Drama bS 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


Movie-Recommendation Admin Page
Main Page
Manage Movies
Manage Users
Home Website Page
Logout
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
© 2020 Movie-Recommendation


Movie-Recommendation Admin Page
Main Page
Manage Movies
Manage Users
Home Website Page
Logout
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
© 2020 Movie-Recommendation

Movie-Recommendation Admin Page | Main Page | Manage Movies | Manage Users | Home Website Page | Logout

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	tharun67@yahoo.com	No	No	DELETE USER