

**Subject: Design and
Development of a streaming
web application.**

List of figures

Figure 1 : Dynamic context diagram	17
Figure 2: Use case diagram	18
Figure 3 : Sequence diagram of crud operations on the API	19
Figure 4 : authentication with JWT Tokens.....	20
Figure 5 : Authentication with JWT tokens (2)	21
Figure 6 : Class diagram	22
Figure 7 : Database Schema.....	23
Figure 8 : Angular & Spring	28
Figure 9 : Physical Architecture of the system.....	29
Figure 10 : Spring package design.....	30
Figure 11 : Notification sent via email to the user	31
Figure 12 : Webscraping Methodology.....	31
Figure 13 : Comparaision between POO and POA+POO	33
Figure 14 : Email confirmation sent to the user	35
Figure 15 : Example of the email sent and the link	36
Figure 16 : Register page of the application	36
Figure 17 : Home page of the application.....	39
Figure 18 : Anime list display	42
Figure 19 : TV shows and series display	42
Figure 20 : List of scraped Movies.....	43
Figure 21 : List of Our own movies.....	43
Figure 22 : Movie Details when not logged in.....	44
Figure 23 : Movie details when logged In	44
Figure 24 : Watch Movies	45
Figure 25 : Watch anime or tv shows with episodes.....	45
Figure 26 : Commenting on movies/ episodes	46
Figure 27 : Blog page and posts Tile.....	47
Figure 28 : Form to post on the blog	47
Figure 29 : Viewing Post details and comments related to it.....	48
Figure 30 : User profile details with his own posts displayed	48
Figure 31 : User watchlist display and delete button.....	49
Figure 32 : User watch history.....	50
Figure 33 : Admin section : delete comments/ ban users	50

Figure 34 : Admin section : delete Posts / movies	51
Figure 35 : Form for the admin to fill if he wants to post a movie	52

List of Tables

Table 1 : Target audience	9
Table 2 : Content of the application.....	10
Table 3 : Functional Requirements.....	11
Table 4 : Existing solutions	14
Table 5 : Benchmarking of our application vs existing	16
Table 6 : Used technologies and frameworks.....	25
Table 7 : List of libraries and dependencies used.....	26

Table of content

List of figures	2
List of Tables	3
Table of content	4
ABSTRACT	6
INTRODUCTION	7
Chapter I: General Context	8
1. Project's Global Objective:	8
2. Problematic:	8
3. Specifications:	8
a. Target Audience:	9
b. Content:	10
c. Functional Requirements:	11
d. Non-Functional Requirements:	12
Chapter 2: Preliminary study	14
1. Quick Overview:	14
2. Existing solutions analysis:	14
3. Critique of the Existing:	15
4. Benchmarking:	16
Chapter 3: Analysis and design	17
1. Identification of Actors:	17
2. Use Case Diagram:	18
3. Sequence Diagrams:	19
4. Class Diagram:	22
5. Database Schema:	23
Chapter 4: Realization	24
1. Working Methodology	24
a. Version Control:	24
b. Branching Strategy:	24
c. Pull Requests and Code Review:	24

d.	Issue Tracking:	25
e.	Continuous Integration:	25
2.	Technical study:	25
a.	Technologies and tools used:	25
b.	General Architecture:	28
c.	Web Scraping:	31
d.	Concepts and patterns used:.....	32
3.	Presentation of the application:	34
General Conclusion		53
Bibliography		54

ABSTRACT

The project involves the development of a streaming app specifically designed for movies and anime series. The app incorporates various features to enhance user experience and foster community engagement.

Users can access a wide range of content, including movies and anime series, through a user-friendly interface.

The app utilizes web scraping techniques to gather information from external websites, ensuring an extensive collection of titles and metadata. In addition to the scraped content, users have the ability to contribute their own movies to the platform, diversifying the available options.

The app facilitates user interaction by allowing comments on specific episodes of anime series, promoting discussions and sharing of opinions. Furthermore, users can create and participate in discussion posts related to movies and anime, encouraging a vibrant community.

The app prioritizes security and moderation to maintain a safe environment for users. Personalized recommendations based on user preferences and viewing history enhance the overall experience.

The app is compatible with multiple platforms, providing accessibility to a wide range of devices. Overall, the project aims to provide a comprehensive and engaging streaming experience for movies and anime enthusiasts while fostering a sense of community and interaction among users.

INTRODUCTION

The rapid growth of streaming services has revolutionized the way we consume entertainment, providing convenient access to a vast array of movies and TV shows. As avid enthusiasts of both movies and anime series, our team embarked on a project to develop a streaming app tailored specifically to cater to these genres. This report outlines the conceptualization, development, and implementation of our streaming app, which aims to provide users with a seamless and immersive experience in exploring and enjoying their favorite movies and anime series.

The primary objective of our project was to create a user-friendly platform that aggregates content from various sources, including external websites, and allows users to interact with the content through features such as episode comments and discussion posts. Leveraging web scraping techniques, we ensured that our app offers an extensive collection of movies and anime series, automatically updating its database to include the latest releases and relevant metadata. Additionally, we empowered users to contribute their own movies to the platform, enhancing the diversity and richness of available content.

Recognizing the importance of community engagement and interaction, our app provides a platform for users to express their thoughts, opinions, and analysis through episode comments. This feature fosters an environment where viewers can connect with one another, share their perspectives, and delve deeper into the intricacies of their favorite anime series. Furthermore, we incorporated a discussion post feature, allowing users to create and participate in conversations related to movies and anime, sparking insightful debates and enabling recommendations among community members.

In order to ensure a secure and enjoyable user experience, we implemented robust security measures and moderation mechanisms to prevent malicious activities and maintain a respectful environment. We understand the significance of personalized recommendations, and therefore, our app employs advanced algorithms that analyze user preferences and viewing history to offer tailored suggestions, aiding users in discovering new movies and anime series aligned with their tastes.

Throughout this report, we will delve into the technical details of our streaming app, highlighting the key features, methodologies employed, challenges faced, and the overall outcomes achieved. We believe that our app holds immense potential in providing movie and anime enthusiasts with a comprehensive, interactive, and engaging streaming experience while fostering a vibrant community of like-minded individuals.

By presenting our project's journey, we aim to offer insights into the development of a streaming app for movies and anime series, contributing to the growing landscape of digital entertainment platforms.

Chapter I: General Context

1. Project's Global Objective:

The global objective of our streaming app is to provide users with a comprehensive platform to stream and explore a wide range of movies and anime series. The app aims to create a vibrant community where users can engage in discussions, share their thoughts, and interact with content and other users in a seamless and intuitive manner.

2. Problematic:

The current landscape of online streaming platforms lacks a comprehensive solution that effectively caters to the preferences of movie, series, and anime enthusiasts. Existing platforms may fall short in providing seamless navigation, interactive community features, and a diverse content library. Users often face limitations in discovering new content, engaging with fellow enthusiasts, and accessing personalized recommendations.

To address this problem, we aimed to develop a streaming and blogging application for movie reviews that offers a user-friendly and engaging platform. The challenge was to create an intuitive interface that seamlessly integrates streaming capabilities, community interactions, and personalized content recommendations. We needed to ensure that users could easily navigate through a diverse content library, participate in discussions, and receive timely updates on new releases and community activities.

3. Specifications:

The web application we are developing aims to reach a broad audience of movie, series, anime, documentary, and anime enthusiasts. Our target audience consists of individuals who are passionate about audiovisual entertainment and are seeking a user-friendly and diverse streaming platform to fulfill their viewing needs. Here are the key characteristics of our target audience:

a. Target Audience:

Table 1 : Target audience

Audience	Description
Movie and Series Enthusiasts	<ul style="list-style-type: none">✓ Our target audience consists of movie and series enthusiasts who are eager to discover new content, explore different genres, and stay up-to-date with the latest releases.✓ They seek a seamless viewing experience that allows them to immerse themselves in captivating stories and emotionally connect with characters.
Anime Enthusiasts	<ul style="list-style-type: none">✓ A significant portion of our target audience comprises anime enthusiasts who have a deep appreciation for Japanese animation and storytelling.✓ They are dedicated fans who actively seek out new anime series, engage in discussions, and stay connected with the anime community.✓ These enthusiasts have specific preferences for anime genres, art styles, and cultural elements.
Adventurous Users	<ul style="list-style-type: none">✓ Our target audience consists of curious individuals who enjoy exploring new content and are open to personalized recommendations to broaden their viewing horizons.✓ They are eager to discover movies, series, documentaries, and anime that align with their specific interests and surprise them with their quality and originality.
Documentary Enthusiasts	<ul style="list-style-type: none">✓ Another portion of our target audience comprises documentary enthusiasts who appreciate inspiring narratives, real-world explorations, and in-depth investigations on a variety of subjects.✓ They seek high-quality documentaries that are both informative and entertaining, enriching their knowledge and understanding of the world around them.

By understanding the characteristics of our target audience, including anime enthusiasts, we will be able to shape the user experience, provide relevant recommendations, and create compelling content that meets the expectations and preferences of our diverse audience.

b. Content:

The web application we are developing will offer a wide range of audiovisual content to cater to the interests and preferences of our target audience. Here are the main content categories that we plan to include:

Table 2 : Content of the application

Content	Description
Movies	<ul style="list-style-type: none">✓ Our movie library will feature titles from various genres such as action, drama, comedy, thriller, science fiction, animation, romance, documentary, adventure, etc.✓ The movies offered will cover different time periods, ranging from cinema classics to the latest releases, providing a diverse selection for all tastes.
Series	<ul style="list-style-type: none">✓ We will provide a diverse selection of television series, including original series, popular series, and hit series from various countries and cultures.✓ The series will be grouped by genre, season, and popularity, allowing users to discover new series and follow their favorites.
Anime	<ul style="list-style-type: none">✓ We recognize the significance of anime and will include a dedicated section for this highly appreciated form of Japanese animation by many fans.✓ Our application will offer a wide selection of popular anime, including classic series, new seasons, movies, and OVA (Original Video Animation).✓ Users will be able to explore different genres of anime, such as action, adventure, romance, fantasy, science fiction, drama, and more.✓ We will ensure to include subtitles or dubbing in different languages to enable an international audience to fully enjoy this form of entertainment.
Ever-Expanding Library	<ul style="list-style-type: none">✓ We are committed to maintaining an ever-expanding content library by regularly adding new movies, series, and documentaries to provide a constantly fresh and exciting viewing experience.✓ We will consider user feedback, market trends, and recommendations to continue enriching our collection.

The addition of anime to our content enriches the application experience by offering a variety of choices to fans of Japanese animation. We aim to meet the growing demand for anime content by providing a diverse and constantly updated collection that fulfills the expectations of anime enthusiasts within our target audience

c. Functional Requirements:

Table 3 : Functional Requirements

Requirement	Description
User Registration and Authentication	<ul style="list-style-type: none">• The system should allow users to register an account by providing necessary information, such as username, email, and password.• Registered users should be able to log in to their accounts using their credentials.
Content Browsing and Filtering	<ul style="list-style-type: none">• The system should provide users with the ability to browse movies and anime series.• Users should be able to search for specific content by entering keywords or using filters such as genre, release date, or rating.• The search functionality should retrieve relevant results based on user queries and provide accurate and meaningful search suggestions.
Content Details and Playback	<ul style="list-style-type: none">• The system should display detailed information about movies and anime series, including title, description, genre, cast, and rating.• Users should be able to select and play individual episodes of anime series.• Playback controls, such as pause, play, and volume adjustment, should be available during content playback.
Commenting on Episodes	<ul style="list-style-type: none">• Registered users should have the ability to leave comments on specific episodes of anime series.• Users should be able to view and read comments posted by other users for a particular episode.
Discussion Posts	<ul style="list-style-type: none">• Registered users should be able to create discussion posts related to movies and anime series.• Users should be able to view and participate in discussions by commenting on existing posts.• The system should display the latest posts and allow users to sort posts based on popularity or recency.
Like and Dislike Functionality	<ul style="list-style-type: none">• Registered users should be able to like or dislike discussion posts to express their preferences.• Users should have the ability to like or dislike comments posted by other users.
Admin Privileges	<ul style="list-style-type: none">• The system should provide administrative privileges to designated admin users.• Admin users should be able to delete inappropriate comments and posts that violate guidelines or community standards.
Content Upload	<ul style="list-style-type: none">• Admin users should have the ability to upload new movies to the app's MongoDB database.• The system should validate and process uploaded content to ensure it meets the required specifications.

User Profile	<ul style="list-style-type: none"> Registered users should have a personal profile where they can view and manage their account information. Users should be able to update their profile details, including username, email, and password.
Recommendations	<ul style="list-style-type: none"> The system should provide personalized recommendations to users based on their viewing history, liked content, and preferences. Recommended movies and anime series should be displayed to users on their homepage or in a dedicated recommendations section.

These functional requirements outline the core functionalities of our streaming app, including user registration and authentication, content browsing and filtering, episode comments, discussion posts, like and dislike functionality, admin privileges, content upload, user profiles, recommendations, and user feedback and reporting. The search functionality allows users to find specific content they wish to watch, enhancing the overall user experience and facilitating content discovery.

d. Non-Functional Requirements:

1. Performance:

- The system should provide fast response times, ensuring a smooth and lag-free streaming experience for users.
- Content loading and buffering should be optimized to minimize wait times.
- The system should handle concurrent user requests efficiently, maintaining high performance even during peak usage periods.

2. Scalability:

- The system should be designed to handle an increasing number of users and content without significant degradation in performance.
- Scalability measures should be implemented to accommodate future growth and increasing user demand.

3. User Experience:

- The user interface should be visually appealing, intuitive, and easy to navigate, providing a seamless user experience.
- Responsive design should be implemented to ensure compatibility across various devices and screen sizes.
- The system should provide clear and concise error messages to guide users in case of any issues or invalid inputs.

4. Security:

- The system should implement secure communication protocols (such as HTTPS) to protect user data during transmission.

- User authentication and authorization mechanisms should be robust and resistant to common security vulnerabilities.
- Sensitive user information, including passwords, should be stored securely using encryption techniques.

5. Reliability:

- The system should have high availability, ensuring minimal downtime and providing uninterrupted access to content for users.
- Data backup and recovery mechanisms should be in place to prevent data loss and ensure data integrity.

6. Compatibility:

- The system should be compatible with major web browsers, ensuring consistent functionality and user experience across different platforms.
- Mobile responsiveness should be prioritized, allowing users to access and use the app seamlessly on mobile devices.

7. Accessibility:

- The system should comply with accessibility standards, making it accessible to users with disabilities.
- Accessibility features such as screen reader compatibility, keyboard navigation, and color contrast should be considered and implemented.

8. Maintainability:

- The system's codebase should be well-structured, modular, and maintainable, facilitating future enhancements and updates.
- Documentation should be comprehensive, providing clear instructions for system maintenance and future development.

9. Compliance:

- The system should adhere to relevant laws and regulations regarding data protection, privacy, and content distribution.
- Intellectual property rights of movies and anime series should be respected, and proper licensing agreements should be in place.

These non-functional requirements address the performance, scalability, user experience, security, reliability, compatibility, accessibility, maintainability, and compliance aspects of our streaming app.

By fulfilling these requirements, we aim to create a high-quality and reliable streaming platform that meets user expectations and provides an enjoyable and secure streaming experience.

Chapter 2: Preliminary study

1. Quick Overview:

A- Definition of Streaming Apps:

Streaming apps are digital platforms that allow users to stream and watch audiovisual content continuously on their connected devices, such as computers, smartphones, tablets, or smart TVs. These apps provide instant access to a vast library of movies, series, documentaries, and anime, enabling users to select and view content according to their personal preferences. Streaming apps have become increasingly popular in recent years, offering convenient and on-demand entertainment experiences for users worldwide.

B- Blogging and Reviews:


In addition to the streaming of content, many streaming apps incorporate features that encourage user engagement, such as blogging and reviews. These features allow users to express their opinions, share insights, and interact with other users within the streaming community. Blogging enables users to create and publish articles or posts related to movies, series, or anime, discussing various aspects such as plot analysis, character development, or thematic elements. Reviews allow users to rate and provide feedback on specific content, helping others make informed decisions about what to watch.



The inclusion of blogging and reviews within streaming apps fosters a sense of community, encourages dialogue, and enhances the overall user experience.

2. Existing solutions analysis:

The streaming industry has witnessed the rise of several prominent platforms that have transformed the way we consume audiovisual content. Three notable examples of existing streaming apps are Netflix, Amazon Prime Video, and Disney+. These platforms have gained immense popularity and have revolutionized the entertainment landscape.

Table 4 : Existing solutions

Solution name	Description
 Netflix	Netflix is a leading streaming platform that offers a wide range of movies, series, documentaries, and original content. It provides a personalized user experience with features like recommended content based on viewing history, user profiles, and the ability to create playlists. Netflix has gained a reputation for its vast library, original productions, and seamless streaming experience.

 <p>Amazon Prime</p>	<p>Amazon Prime Video is part of the Amazon Prime membership package and provides access to a diverse collection of movies, series, and exclusive Amazon Originals. In addition to streaming, it offers additional benefits such as free and fast shipping on Amazon purchases. Prime Video stands out for its integration with the Amazon ecosystem and its ability to offer a wide range of content genres.</p>
 <p>Disney+</p>	<p>Disney+ is a streaming service that focuses on content from the Disney, Pixar, Marvel, Star Wars, and National Geographic franchises. It offers a rich collection of family-friendly movies, beloved classics, and new releases. Disney+ is known for its strong brand presence, high-quality content, and dedicated fan base.</p>

3. Critique of the Existing:

The critique of existing solutions highlights certain aspects and limitations observed in current streaming apps.

1. Content Limitation:

While Netflix, Amazon Prime Video, and Disney+ offer a vast selection of content, they may not always cater to all user's tastes and preferences. Some users may find gaps in certain content categories or geographical restrictions that limit access to specific works.

2. Content Fragmentation:

Each streaming platform has its own exclusives and distribution agreements, leading to content fragmentation. Users may need to subscribe to multiple services to access the full range of movies, series, and documentaries they wish to watch, which can be costly and inconvenient.

3. User Interface and Experience:

While existing streaming apps generally provide a smooth user experience, there are still possible improvements in terms of the user interface. Some users may find content organization, navigation, or search functionalities to be lacking in certain areas.

4. Geographical Availability:











The availability of streaming apps can vary by country and region. Certain services may be limited or unavailable in specific parts of the world, restricting users' access to specific content.

By identifying these aspects, we aim to design a solution that addresses the existing gaps and offers an enhanced streaming experience by providing diverse content, a user-friendly interface, and global accessibility.

4. Benchmarking:

When comparing our streaming app solution with the existing platforms like Netflix, Amazon Prime Video, and Disney+, we aim to highlight the unique features and advantages that our solution brings to the market.

Table 5 : Benchmarking of our application vs existing

Features	Existing Solutions	Our solution	How
Content Diversity			By catering to the preferences of a diverse target audience, including anime enthusiasts, we provide a more inclusive and specialized content library.
Flexibility and Personalization			By integrating scraping capabilities from external websites, users can access a wider range of movies and series, including lesser-known or niche titles. This expands the content options and provides a more personalized streaming experience.
User-Focused Recommendations			While existing platforms provide recommendations based on viewing history, our solution enhances this feature by incorporating user-generated content.
Openness to User Contributions			Through comments, posts, and interactions, our recommendation system can better understand individual preferences and deliver more accurate and tailored content suggestions.
Blogging			By fostering a sense of community and enabling users to share their thoughts and engage in conversations, we create a more immersive and engaging user experience.

While existing platforms have established themselves as industry leaders, our solution brings innovative features, user engagement, personalized recommendations, and a diverse content library that caters to a wide range of interests. By combining these strengths, we aim to provide a unique and compelling streaming experience that differentiates us from the competition.

Chapter 3: Analysis and design

1. Identification of Actors:

1. User: The User represents individuals who visit the streaming app. They have access to the app's content and can navigate through the website, watch episodes, and explore different sections. Users have the option to register and become Registered Users.

2. Registered User: Registered Users are individuals who have created an account on the streaming app. In addition to the capabilities of regular Users, Registered Users can log in to their accounts, leave comments on episodes, create and participate in discussion posts, as well as like or dislike posts. They have the ability to engage more actively with the app's content and community.

3. Admin: The Admin is a special role assigned to specific individuals who have administrative privileges within the streaming app. The Admin has the authority to manage and moderate user-generated content. They can delete comments and posts that they find inappropriate or violate the app's guidelines. Furthermore, the Admin has the capability to upload movies to the app's MongoDB database, expanding the available content.

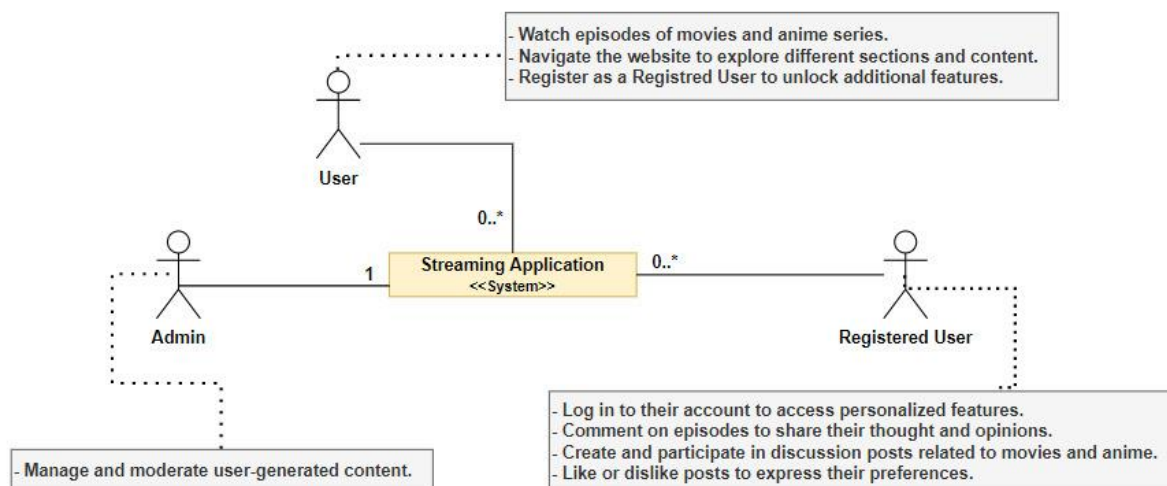


Figure 1 : Dynamic context diagram

By identifying the key actors and their associated functionalities, we have established a solid foundation for the development of our streaming app. These requirements serve as a guide to ensure that the app meets the expectations and needs of different user roles, creating a dynamic and engaging platform for movie and anime enthusiasts.

2. Use Case Diagram:

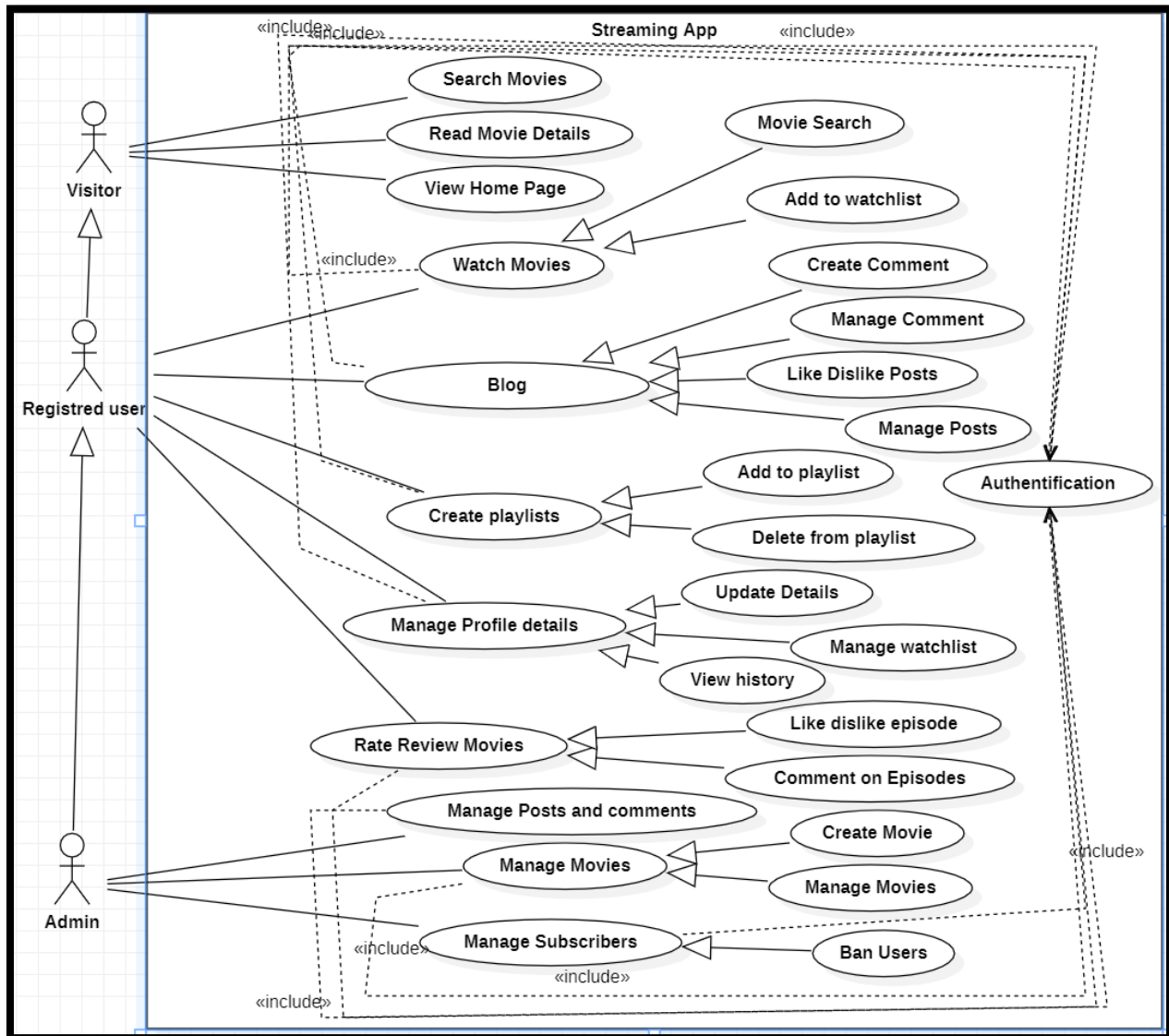


Figure 2: Use case diagram

Based on the actors, the functional and the nonfunctional requirements, we can come with the use case diagram above which represents all the interactions between the actors and the systems, as we can notice most of them require the user to have an account for better experience.

3. Sequence Diagrams:

The desired overall behavior of our API is modeled using a system sequence diagram, illustrated in the sequence diagram below. This diagram represents the external invocation of a CRUD functionality on a resource exposed by the API.

The functionalities of the API will cover all the actions, requests that need to be executed in the backend for both the streaming and the blogging services.

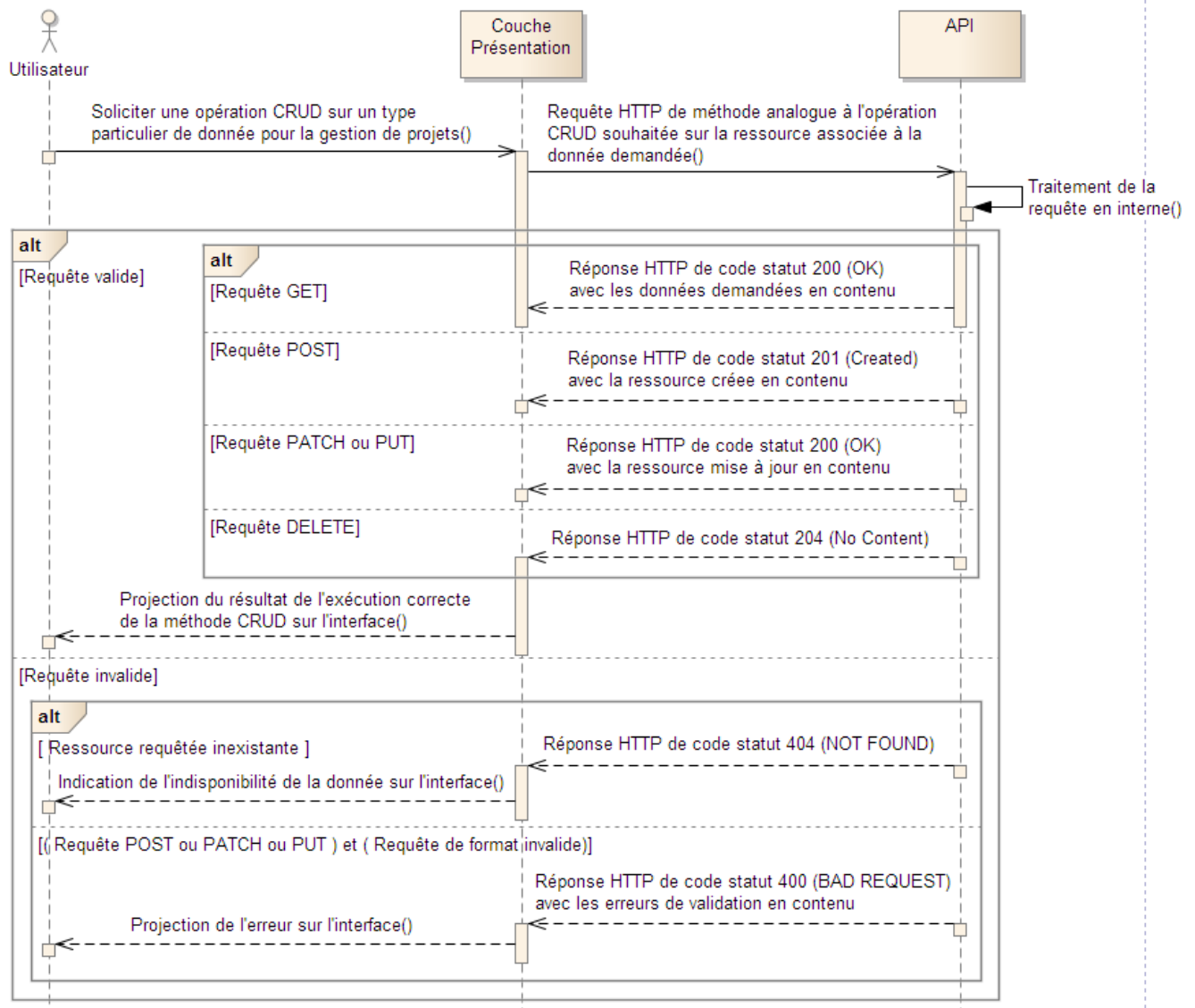


Figure 3 : Sequence diagram of crud operations on the API

Throughout the development process, security considerations are of utmost importance. Measures such as data encryption, secure authentication protocols, and role-based access control are implemented to protect user information and ensure a secure environment.

Upon acquiring the application, there is only one account initially: the default administrator account. It should be noted that there can be multiple administrator accounts, primarily to avoid potential rigidity during the operation of the application.

```

sequenceDiagram
    actor Utilisateur
    participant Front-end
    participant Serveur as Serveur d'application
    participant Conteneur as Conteneur Spring
    participant Interceptor as TenantLoginInterceptor
    participant Context as TenantContext
    participant Controller as Authentication Controller
    participant Repository as UsersRepository
    participant Base as Base de données

    Utilisateur->>Front-end: Afficher la page de login()
    Front-end->>Front-end: vérifier la session de l'utilisateur en local()
    alt [connecté]
        Front-end->>Front-end: rediriger vers la page d'accueil()
    else [déconnecté]
        Front-end->>Front-end: affichage de la page de login()
        Utilisateur->>Front-end: soumettre formulaire d'authentification()
        alt soumission formulaire
            [formulaire invalide]
                Front-end->>Front-end: affichage d'erreurs de validation sur le formulaire()
            opt [corriger le formulaire et resoumettre]
                Utilisateur->>Front-end: soumettre le formulaire d'authentification()
            alt soumission formulaire
            end
        else [formulaire valide]
            Front-end->>Serveur: requête POST sur /tenants/<pseudo>/login (identifiants utilisateur)
            Serveur->>Conteneur: relayer la requête (identifiants utilisateur)
            Conteneur->>Interceptor: relayer la requête (identifiants utilisateur)
            Interceptor->>Context: relayer la requête (identifiants utilisateur)
            Context->>Controller: relayer la requête (identifiants utilisateur)
            Controller->>Base: récupérer le tenant (client) à partir de «pseudo»
            Base-->>Controller: tenant() .Tenant
            Controller->>Context: setCurrentTenant(tenant)
            Context->>Interceptor: relayer la requête (identifiants utilisateur)
            Interceptor->>Conteneur: relayer la requête (identifiants utilisateur)
            Conteneur->>Serveur: traitement interne()
            Serveur->>Front-end: relayer la réponse()
            alt [utilisateur inexistant (null)]
                Front-end->>Front-end: affichage d'erreur login erroné sur le formulaire()
            else [utilisateur existant]
                Front-end->>Front-end: rediriger vers la page d'accueil()
            end
        else [soumission de formulaire]
        end
    end
    
```

20

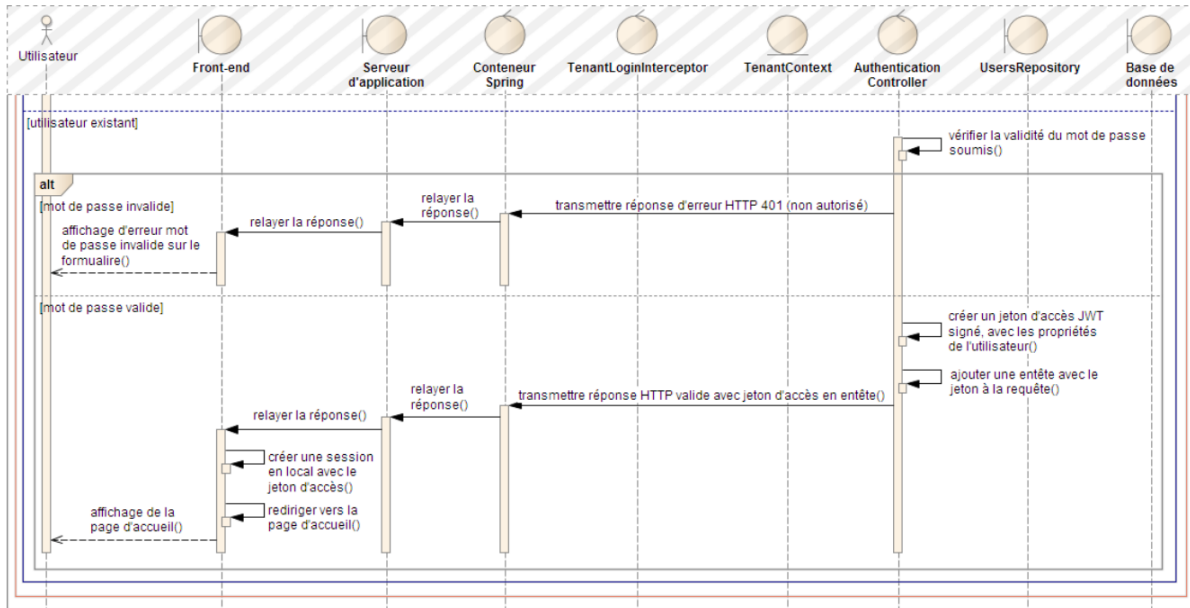


Figure 5 : Authentication with JWT tokens (2)

This effectively shifts the session management, traditionally handled on the server side, to the client side. The server only reads the trusted content to determine the session state, modifies it, and then transmits the updated state with a new token in the response. This approach allows our application to maintain its statelessness.

We take this opportunity to specifically present the user management case, which is newly introduced. Firstly, we notice the introduction of new elements compared to the existing ones, namely Spring Security Aspects. The remaining elements are part of the previously established execution flow. Regardless of the user requesting a resource from our API, the behavior remains quite similar.

Firstly, it is now the responsibility of the TenantHeaderInterceptor interceptor to specify the global client to serve before allowing the system to focus on the specific user. The client's identifier is provided by the access token sent with the request from the front-end. This is still the case because the front-end would automatically redirect a user to the login page if they attempted to access another page without being authenticated.

The second step is to authorize access to the requested resource for the specific user. The management of this aspect is delegated to Spring Security through the use of Programming Oriented Aspects to enhance the behavior of our API without modifying the existing infrastructure. In this particular case, the only user who would be denied access is the subscriber account type, as they have no Admin roles.

The final part of the authorization mechanism is responsible for filtering the content to be returned.

4. Class Diagram:

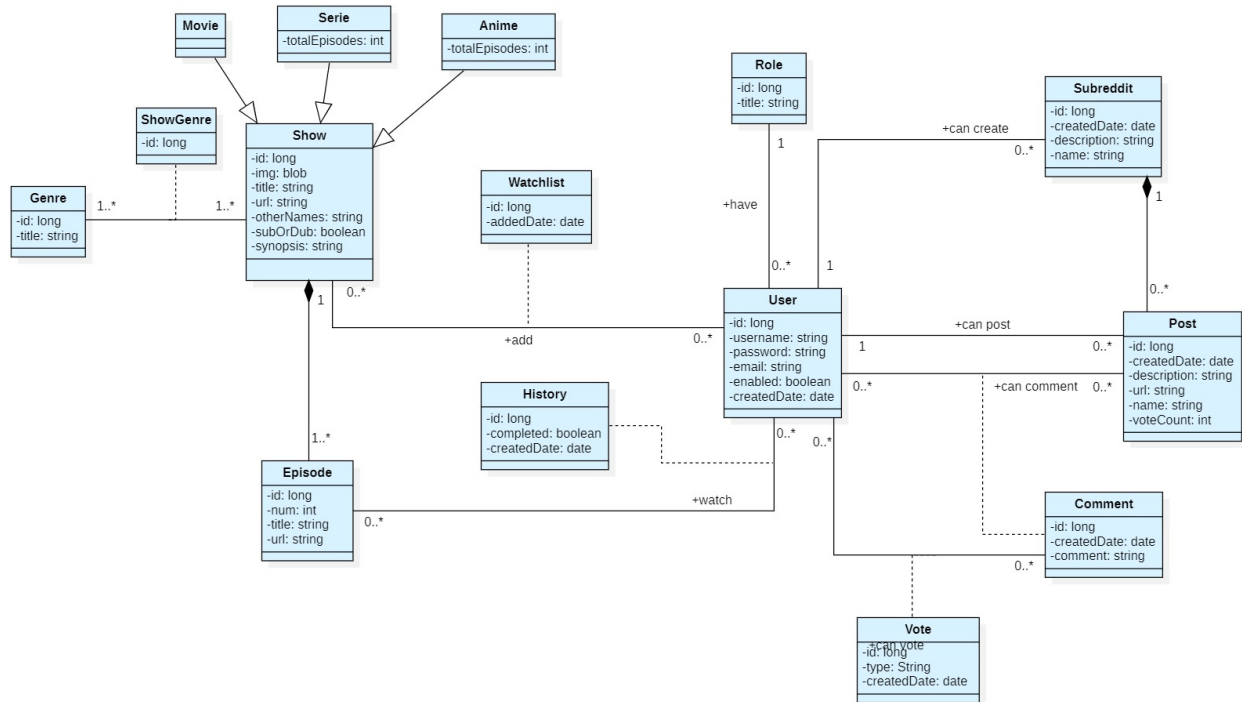


Figure 6 : Class diagram

To design a REST API, it is essential to determine the data model beforehand in order to achieve the object-resource mapping characteristic of a RESTful model. The established domain model is illustrated in the class diagram.

In this Class diagram we can see that we have an anime/ Movie or series that contain Episodes, these episodes only have the links to the playable videos, the anime class contains details of the Movie or episode.

An episode has no or many comments that are submitted by the subscribed users.

The users have the right to post on the blog, share the posts, comment on them and vote.

Every post belongs to a sub-category named subreddits, that can be common subjects, groups, anime, movies or series or just some discussions.

Every subreddit has many posts that are related to it.

A user can add many anime, episodes, movies to his watchlist, and these can be added to every user's watchlist.

Every time a user starts watching an episode either a movie or anime, it's added to his history where he can view them anytime.

The history contains some data about the watched episode.

5. Database Schema:

For our data to be well structured and stored, we will need to have the right structure for this type of project, that will save all the important details and events that can be keys to solve some of the problems or help improve the experience:

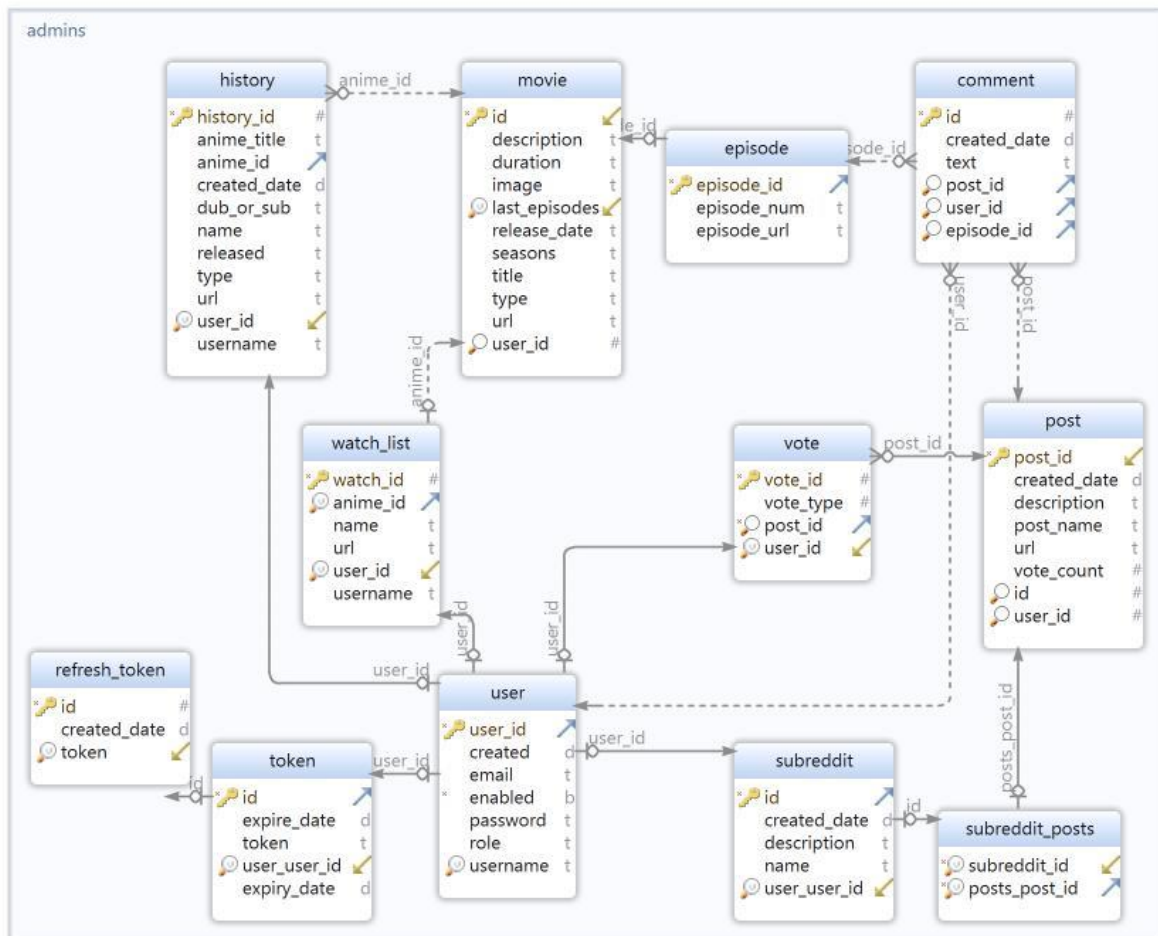


Figure 7 : Database Schema

Chapter 4: Realization

1. Working Methodology

To ensure efficient collaboration and version control during the development of our streaming app, we adopted a methodology that centered around using GitHub as our primary collaborative platform. GitHub provides a robust set of tools for code management, version control, issue tracking, and collaboration, making it an ideal choice for team-based software development projects.

Here is an overview of the methodology we followed:



a. Version Control:

- We utilized Git, a distributed version control system, to manage and track changes to our codebase effectively.
- GitHub served as our central repository, where all team members could push their code changes, branches, and collaborate on different features.

b. Branching Strategy:

- We followed a branching strategy that allowed for parallel development and collaboration without interfering with the main codebase.
- The "master" branch served as the stable production-ready version of the application, while feature branches were created for each new feature or bug fix.
- Collaborative work was done in feature branches, and once completed, changes were merged back into the master branch through pull requests.

c. Pull Requests and Code Review:

- Pull requests were created to propose and review changes before merging them into the main branch.
- This process ensured that code quality, best practices, and any potential issues were thoroughly reviewed and addressed by team members.
- Code reviews played a vital role in maintaining code consistency, identifying bugs or vulnerabilities, and fostering knowledge sharing within the team.

d. Issue Tracking:

- GitHub's issue tracking system was used to manage and track tasks, bugs, and enhancements throughout the project.
- Issues were categorized, assigned to team members, and labeled with relevant tags to provide clarity and organization.
- Regular communication through issue comments allowed for efficient collaboration and ensured that tasks were completed within the project's timeline.

e. Continuous Integration:

- We employed continuous integration practices to automate the build and testing process of our application.
- Whenever changes were pushed to the repository, automated tests were triggered to ensure code integrity and catch any potential issues early on.



By leveraging GitHub and following this methodology, we were able to streamline our development process, maintain a collaborative workflow, and ensure the stability and quality of our streaming app. The combination of version control, branching strategies, pull requests, code reviews, issue tracking, and continuous integration contributed to effective teamwork and successful project delivery.






2. Technical study:

a. Technologies and tools used:

For this project we used the following frameworks along side with their dependencies and libraries:

Table 6 : Used technologies and frameworks

Framework	Description
 ANGULAR	Angular is a widely used open-source web application framework maintained by Google. It enables the development of dynamic and interactive single-page applications (SPAs). Angular offers a component-based architecture, a powerful template system, and a rich set of features for building responsive user interfaces and managing application state.
 Spring	Spring Boot is a Java-based framework that simplifies the development of stand-alone, production-grade Spring applications. It provides a comprehensive set of libraries and features that facilitate rapid application development, dependency management, and configuration. Spring Boot allows developers to create robust, scalable, and easily deployable backend services.

	Mailtrap is a service that simulates an SMTP server for testing email functionality in development environments. It allows developers to capture, view, and debug outgoing emails without sending them to real recipients.
	JWT is a compact and self-contained mechanism for securely transmitting authentication and authorization data between parties. It consists of digitally signed tokens that contain encoded user information, enabling secure and stateless authentication in web applications. JWT provides a standardized method for implementing secure user authentication and authorization in our streaming app.
	MySQL is a widely used open-source relational database management system (RDBMS) known for its scalability, performance, and reliability. It offers robust ACID-compliant transaction support and a wide range of features for efficient data storage and retrieval.
	MongoDB, on the other hand, is a popular NoSQL document database that provides high flexibility and scalability. It stores data in JSON-like documents, allowing for dynamic schema structures and seamless integration with object-oriented programming paradigms.
	Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

a. Backend stack:

The backend of our streaming app utilizes various libraries and dependencies to enhance its functionality, security, and performance. Here are some of the important libraries and dependencies used in our project:

Table 7 : List of libraries and dependencies used

Libraries/Dependencies	Description
Spring Boot Starter Data JPA	Enables seamless integration with the Java Persistence API (JPA) for efficient data access and management.
Spring Boot Starter Web	Facilitates the development of RESTful APIs and web services, handling HTTP requests and responses.
Spring Boot Starter Validation	Provides support for data validation and ensures the integrity of user inputs.
Lombok	Simplifies the development process by reducing boilerplate code, such as getters, setters, and constructors.
Spring Boot Starter Test	Provides testing support for unit tests and integration tests, ensuring the reliability and functionality of the backend.
Jsoup	A Java library used for web scraping, allowing the extraction of anime and movie information from external websites.

Spring Boot Starter Data MongoDB	Enables integration with MongoDB, a NoSQL database, for efficient storage and retrieval of data.
Spring Boot Starter Security	Provides comprehensive security features, including authentication and authorization, to protect the application and user data.
Spring Boot Starter Mail	Enables email functionality, allowing the application to send emails for various purposes, such as user notifications.
Spring Boot Starter Actuator	Provides production-ready features for monitoring and managing the application, including health checks and metrics.
Spring Security Test	A testing library that assists in testing security configurations and authorization rules.
Spring Boot Starter OAuth2 Resource Server	Allows the backend to act as a resource server for OAuth2 authentication and authorization.
MapStruct	A Java annotation processor for generating mapping code between Java bean types, enhancing data transformation and mapping processes.

b. Frontend Stack:

The frontend stack of our Angular project consists of several essential dependencies that contribute to the development and functionality of our application. Here are the important frontend dependencies listed in the "package.json" file:

1. Angular Dependencies:

- @angular/animations: Provides support for animations within the Angular framework.
- @angular/cdk: Offers a set of UI components and utilities for building Angular applications.
- @angular/common, @angular/compiler, @angular/core, @angular/forms, @angular/platform-browser, @angular/platform-browser-dynamic, @angular/router: These packages are core Angular modules that provide the fundamental building blocks and functionality for the frontend.

2. UI and Styling:

- @fortawesome/angular-fontawesome, @fortawesome/free-solid-svg-icons: Enables the usage of Font Awesome icons in Angular applications.
- @ng-bootstrap/ng-bootstrap, bootstrap, ng-bootstrap: Provides Bootstrap components and utilities for Angular applications.
- primeng: Offers a rich set of UI components for Angular applications.
- ngx-useful-swiper, swiper: Enables the integration of Swiper, a popular and customizable carousel/slider library, into Angular applications.

3. Additional Functionality:

- @tinymce/tinymce-angular: Allows the integration of the TinyMCE WYSIWYG editor into Angular applications.
- ngx-toastr: Provides toast notifications for Angular applications.
- ngx-webstorage: Offers an Angular wrapper for the Web Storage API, enabling easy storage and retrieval of data in the browser.

- zone.js: Provides a mechanism for tracking and propagating changes within Angular's execution context.

These dependencies, and technologies formed the foundation of our streaming app, enabling us to develop a robust and scalable backend using Spring Boot, an interactive and dynamic frontend with Angular, reliable email functionality with Mailtrap, secure authentication with JWT, and efficient data storage and retrieval along with the core Angular packages, facilitate the development of a dynamic and responsive user interface, incorporating UI components, icons, styling, and additional functionality.

b. General Architecture:

a. Architecture

Before starting the development phase, it is necessary to step back from the requirements and establish an overview of the system from an architectural point of view. In our particular case, the SaaS aspect of the application requires a thorough study of the underlying infrastructure to be implemented and the technologies to be used.

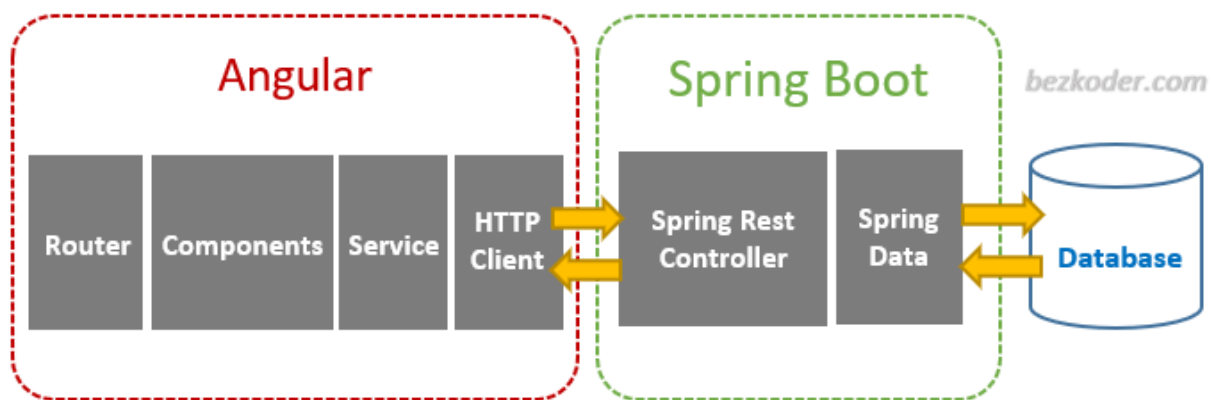


Figure 8 : Angular & Spring

An overview of the architecture highlights the different layers of the system to be developed. Considering the web nature of the application, we initially leaned towards a traditional 3-tier architecture, but eventually opted for a 4-tier architecture, which includes a separate static server for the front-end retrieval, distinct from the application server. The diagram illustrates the relationship between the various components of this architecture.

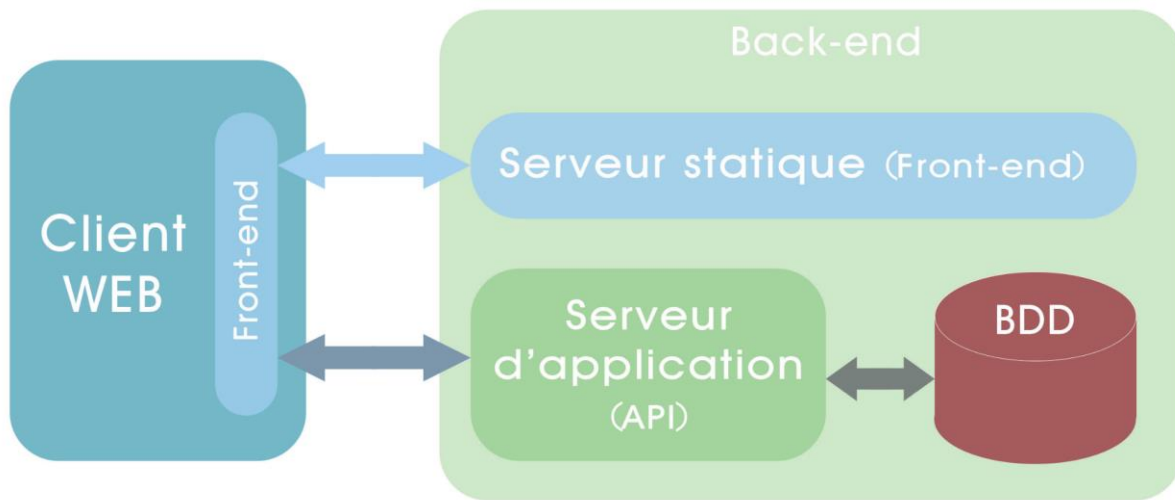


Figure 9 : Physical Architecture of the system

Our system, modeled by the Back-end, each tier represents one of its layers. The logical architecture of the system is divided into three layers:

- The data access layer: This layer is responsible for managing the system's data (storage, persistence, access management, etc.). It is represented by our database.
- The business layer: This is the functional part of the system. It implements the business operational logic and relies on the data provided by the lower layer. It offers services to the presentation layer. It is represented by our application server, which provides these services in the form of APIs.
- The presentation layer: It encapsulates the user interface and corresponds to the visible part of the application. This layer relies on the services offered by the business layer to process user requests and handles the rendering and formatting of retrieved information. It is represented by our static server, which provides the front-end application to the client.

In our context, this layered architecture is the most appropriate. The traditional 3-tier architecture offers the following benefits:

- Clear separation of responsibilities between the layers
- Loose coupling between the different levels
- Simplicity in testing (business logic is isolated from the other layers)
- Increased security due to the separation between the client and the data
- Ability to use lightweight clients (suitable for mobile devices)
- High scalability resulting from the independence of the layers

From there, the next logical step for a more refined separation of concerns and easier maintenance was to move towards the 4-tier architecture. Through its adoption, we have clearly separated the Front-end application from the application responsible for exposing our API. Managing these two applications as separate projects also allows us to benefit from:

The API client code is organized into a number of Java packages. The mains ones are shown in the diagram below, along with an indication of their purpose and the dependencies between them:

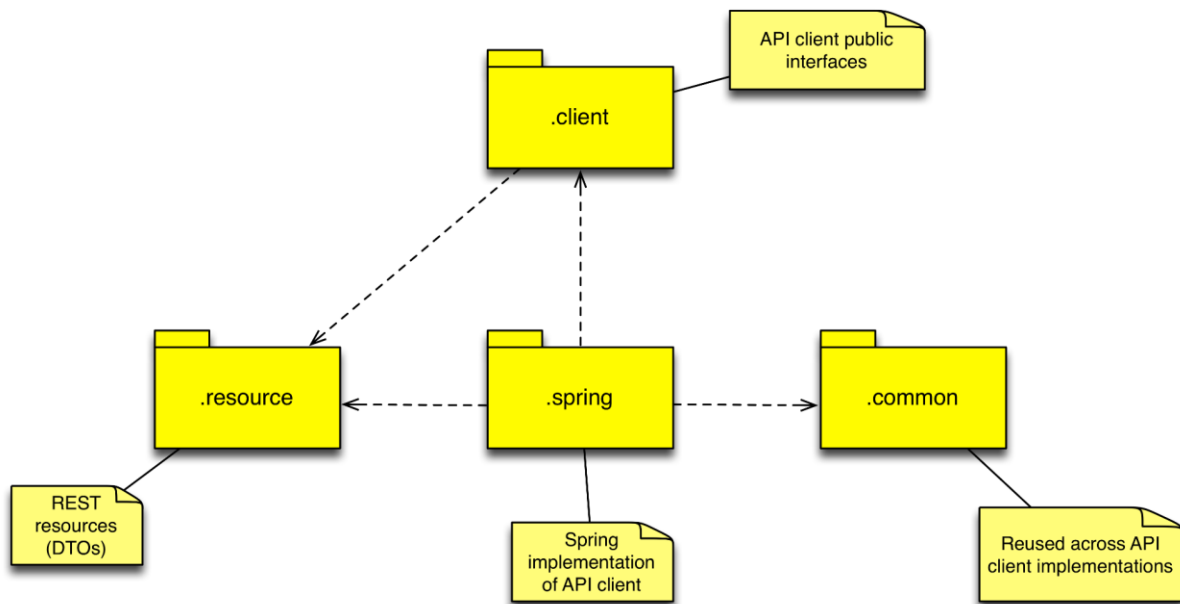


Figure 10 : Spring package design

The reasons behind this package design include:

- The 'client' package encapsulates the public interfaces of the API client. External services which use the API client need only utilise these classes, to minimise coupling.
- Classes which are specific to a particular implementation of the API client, such as the out-of-the-box Spring implementation, are contained in a separate package. Other implementations could be added in the future

b. Notification system:

We implemented a notification system in our application to enhance the user experience and provide timely information to our users. The system is designed to send confirmation emails to new subscribers or when a password change is requested. This ensures that users have a secure and seamless onboarding process.

Additionally, the notification system notifies users when someone likes or comments on their posts. This fosters engagement and encourages active participation within the community. Users receive real-time updates and can quickly respond to interactions, creating a more interactive and dynamic platform.

Abdelghafour12 posted a comment on your post.

Figure 11 : Notification sent via email to the user

Furthermore, the notification system can be extended to notify users when new movies are added. This feature ensures that users stay up-to-date with the latest content additions and allows them to explore new movies as soon as they are available.

During development and testing, we utilized a tool called Mailtrap. Mailtrap is an email testing service that helps simulate the sending and receiving of emails in a safe and controlled environment. By using Mailtrap, we ensured that the notification system functioned correctly without sending actual emails to real users during the testing phase. This allowed us to validate the functionality and troubleshoot any issues before deploying the application to production.

Overall, the notification system enhances user engagement, provides important updates, and ensures a smooth user experience within our application.

These notifications will also be used in the front end to add a more enhanced experience.

c. Web Scraping:

For this project we used existing data sources from other websites, this is called web scraping. Web scraping refers to the process of extracting data from websites. It involves using automated tools or scripts to access and retrieve information from web pages, typically in an organized and structured format. Web scraping can be useful for various purposes, such as gathering data for research, monitoring prices, aggregating content, or analyzing trends.

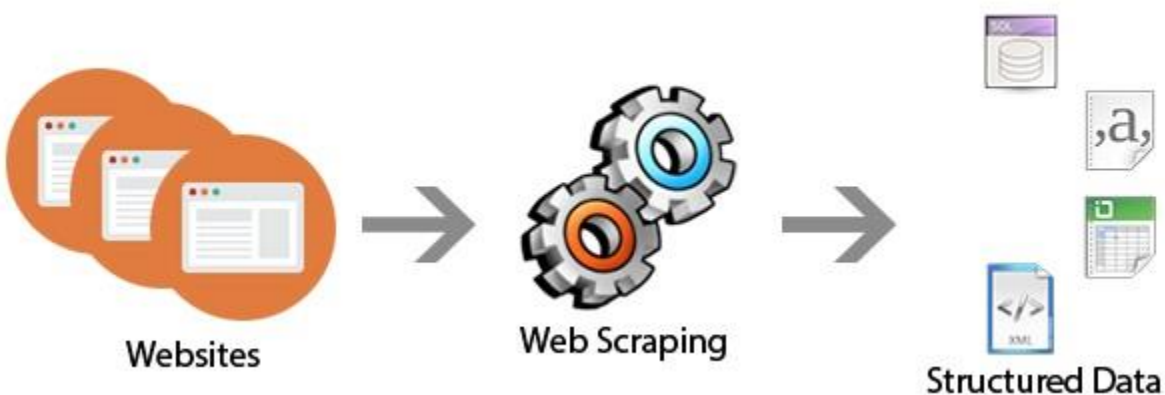


Figure 12 : Webscraping Methodology

Our API will be using multiple java tools and libraries to extract the data that we need from the website in real-time. For that we used JSoup Library and some External API's so that on every Call on our API, it will refresh the data and stream it directly without even storing it.

The raw data is then structured to be ready to use by our front-end client so that he can display it as we want it to be.

All the data will be displayed in a JSON format.

d. Concepts and patterns used:

By incorporating Spring Security into our API, we can enforce access control rules, handle user authentication and authorization, and implement secure communication protocols, such as HTTPS, to safeguard sensitive data.

Thus, we primarily rely on aspect-oriented programming (AOP) and the Repository pattern for the implementation of our API. We will also heavily utilize dependency injection (DI) provided by Spring.

Let's briefly explain these concepts:

Repository Pattern

The Repository pattern is one of the patterns defined by Martin Fowler in his book on enterprise application architecture patterns [39]. It involves setting up a mediator between the domain model and the data access layer using an interface that abstracts the access to domain objects in the form of collections.

Dependency Injection

Dependency injection is a mechanism that implements the principle of inversion of control. This principle, common in frameworks, advocates transferring the control of the software's execution flow from the application itself to the framework (or underlying software layer). Dependency injection aims to reduce explicit dependencies between software components. Thus, dependencies between software components are no longer statically expressed in the code but dynamically determined at runtime.

Aspect-Oriented Programming (AOP)

Aspect-Oriented Programming is a programming paradigm that introduces a new way of structuring and programming applications. It can be applied to both procedural and object-oriented programming languages. AOP applied to Object-Oriented Programming (OOP) aims to address two main limitations of individual OOP usage: the intertwining of business logic with cross-cutting functionality logic and code scattering.

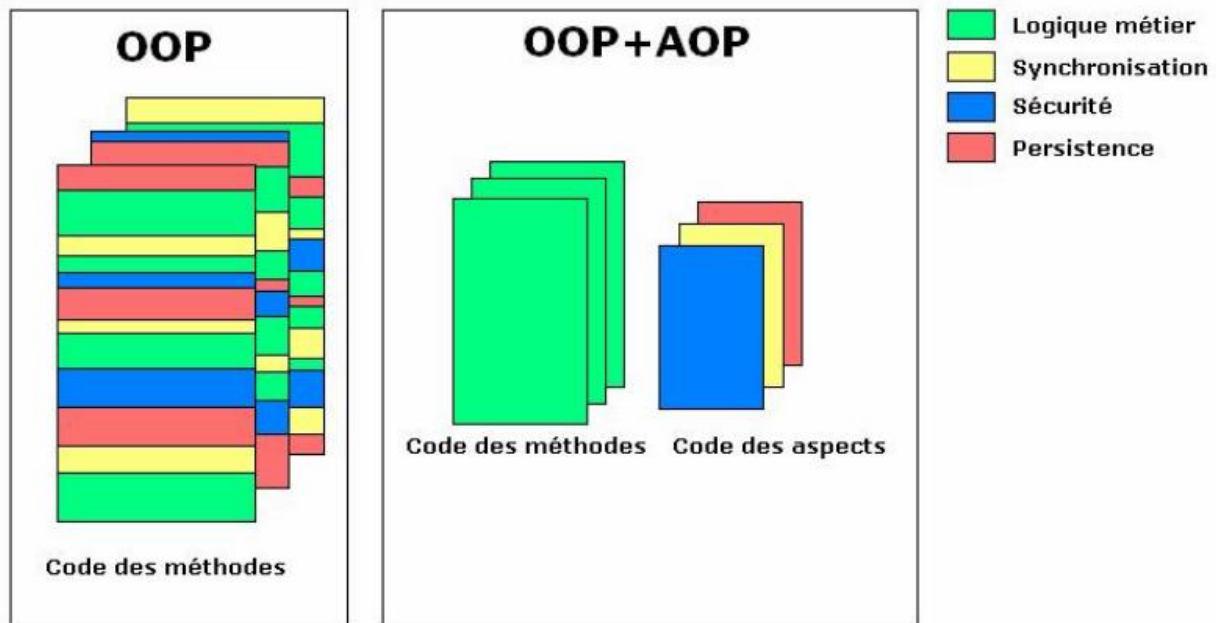


Figure 13 : Comparaison between POO and POA+POO

AOP provides a mechanism to modularize cross-cutting concerns, such as logging, caching, security, and transaction management, which would otherwise be scattered throughout the application. It achieves this by separating the core business logic from these cross-cutting concerns and encapsulating them in separate units called aspects.

Aspects define reusable modules that can be applied to multiple points in the application's codebase. This approach promotes code reuse, modularity, and separation of concerns. By utilizing AOP, developers can enhance the maintainability, extensibility, and readability of their applications.

ORM (Object-Relational Mapping)

For interacting with our data access layer, we chose to use Object-Relational Mapping (ORM) through the usage of interfaces provided by JPA (Java Persistence API). Once defined on our model, the mapping between the object world and the relational world helps us stay focused on the business logic without worrying about the internal workings of the database.

Spring Boot assists us in this aspect by default, as it offers the option to use the Hibernate ORM tool, which is an implementation of JPA. We will not be using any framework-specific features (i.e., outside of JPA) for the development of this part.

Using ORM simplifies database operations by allowing us to work with objects and their relationships, while the ORM framework takes care of translating these operations into corresponding database queries. This abstraction enables us to work at a higher level of abstraction and promotes maintainability and flexibility in our application.

Application Server

Choosing an application server is essential to expose our application over the web. Once again, Spring Boot comes to our aid by default, offering the capability to run our application within an embedded Tomcat server. Tomcat is a free, open-source JEE web container developed by the Apache Foundation. Its default integration means that an instance of this server is launched concurrently with each execution of our application.

This default setup proves particularly useful in our case, as we are implementing a SaaS (Software as a Service) application with a stateless architecture. Simply launching a new instance of our application allows us to scale this tier effortlessly (in addition to trivial routing between these instances).

By leveraging the embedded Tomcat server provided by Spring Boot, we can simplify the deployment and hosting of our application, ensuring seamless web accessibility and efficient resource utilization.

3. Presentation of the application:

The development model in AngularJS relies on the dependency injection pattern. The framework distinguishes between two types of injectable components:

- A controller is responsible for communicating with the user interface.
- A directive extends a basic HTML component into a component with dynamic behavior.
- A filter applies pre-defined logic to format input data.
- An animation dynamically associates a specific style with an HTML component.

During our development, we will only focus on designing controllers. For the rest, we will rely on the specialized objects provided by the framework by default.

Backend: We relied on the existing infrastructure to integrate the authentication and authorization mechanisms into the existing components. This was achieved through the use of Spring and its implementation of the AOP (Aspect-Oriented Programming) paradigm. To further simplify the process, we utilized the integration of the Spring Security module from the Spring suite. This module provides a convenient way to manage authentication and authorization aspects using the `PreAuthorize` and `PreFilter` annotations, which respectively allow or deny access to a method (in this case, responsible for one of our resources) and filter the result returned by a method (in relation to the specific resource). These annotations enhance the already established repositories that are exposed as resources for our API through Spring Data REST. Once authenticated, the user is constantly accessible through the shared global context with Spring Security, allowing us and Spring Security to consider the user during subsequent processing.

To adhere to security best practices and protect our user accounts, we made sure not to persist any passwords in our database. Instead, we rely entirely on the hashing and salting of passwords (using the BCrypt utility class from Spring Security) during user creation or when validating their credentials during authentication. This process is only executed during these two mentioned steps because for subsequent access, the user communicates with a trusted access token signed by us. Here, we ensured compliance with the JWT (JSON Web Token) standard for token generation and transmission.

Frontend: On one hand, we added a login page and a profile management page to the existing pages. On the other hand, we realized the user account management page in the global menu by reusing the generic resource management page model. This model represents user accounts as entries organized in a table, accompanied by a dropdown form for editing them. In particular, this form includes a password field that is not displayed in the table (it can only be updated). This page differs for different user types in terms of the ability to assign projects to accounts and the types of users managed within themselves.

The session is stored locally by persisting the received JWT token in the localStorage (HTML5 technology), which unlike cookies, is not automatically sent with each server request. This requires us to explicitly manage its sending in our code but protects us from CSRF (Cross-Site Request Forgery) attacks because only an AJAX request sent from a script executed on our domain can access it (a restriction implemented by popular browsers). Therefore, the logout process simply involves removing the session token locally. Since the token content is accessible, this local session state allows us to further customize our user interfaces by adjusting their details based on each user's context.

After registering, an email with a confirmation link containing a token validation is sent to the user:

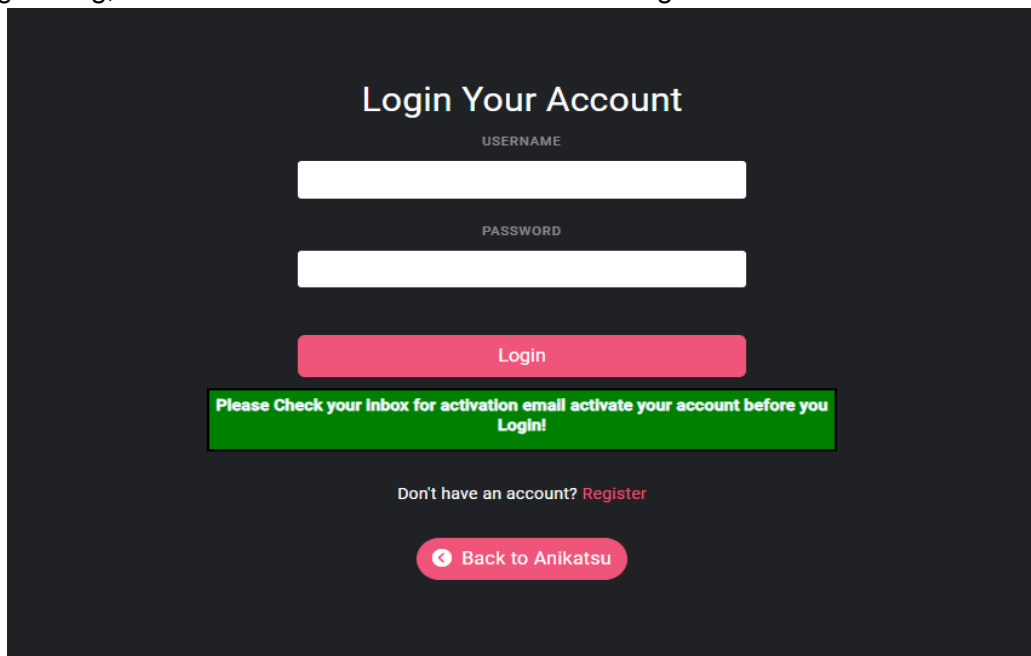
A login form titled "Login Your Account" on a dark background. It features two white input fields for "USERNAME" and "PASSWORD". Below these is a red "Login" button. A green banner with white text reads: "Please Check your inbox for activation email activate your account before you Login!". At the bottom, there is a link "Don't have an account? Register" and a red button with a left arrow and the text "Back to Anikatsu".

Figure 14 : Email confirmation sent to the user

The user then receives the mail looking like this (this could be optimized in the future:

```
Date: Wed, 12 Jul 2023 18:03:26 +0200 (CEST)
From: springreddit@email.com
To: abdel345@gmail.com
Message-ID: <899645991.0.1689177806096@DESKTOP-5696E1A.mshome.net>
Subject: Please Activate your Account
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 7bit

Thank you for signing up to Spring Reddit, please click on the below url to activate your account :
http://localhost:8080/api/auth/accountVerification/866fd595-7b14-4b5e-99f6-68bedf2ad67f
```

Figure 15 : Example of the email sent and the link

Upon visiting the link, the user then activates his account :

Account Activated Successfully

To validate the functional specifications related to this streaming app, we will present and test the interface rendering under different contexts:

Register

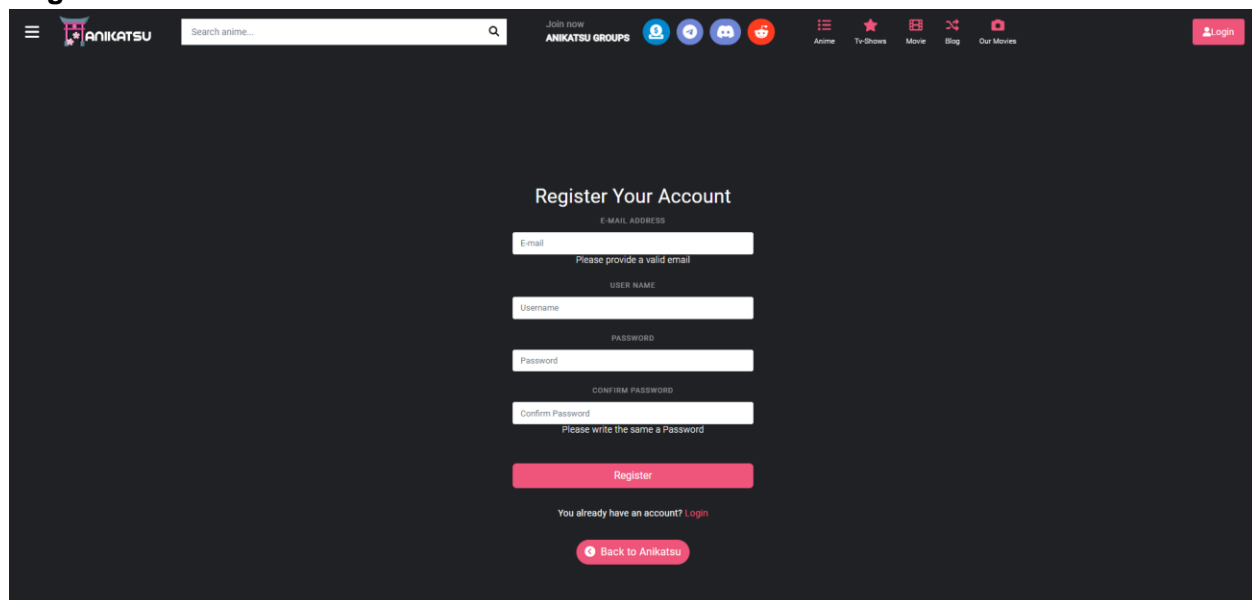
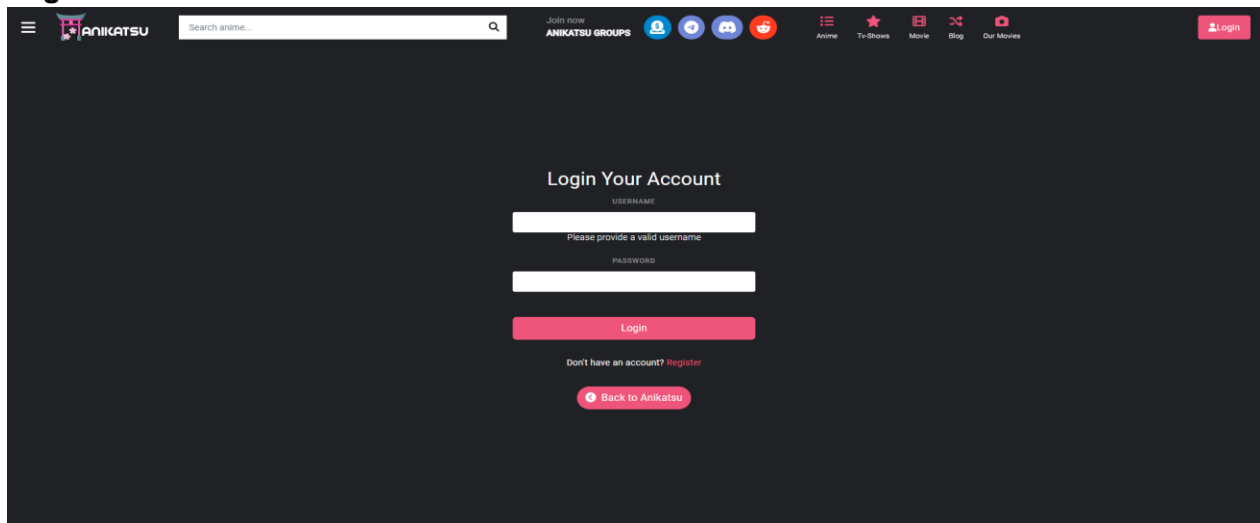


Figure 16 : Register page of the application

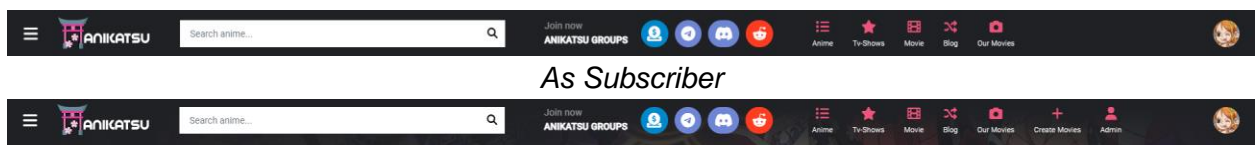
The register page is a crucial component of the streaming app's user registration process. On this page, users are prompted to enter the required information as specified in the corresponding fields. Once the user fills in the necessary details, they proceed to register their account. As a part of the registration flow, an email verification step is implemented to ensure the validity of the provided email address. After the user submits their registration, a verification link is sent to their Mailtrap account. Upon clicking the verification link, the user's account becomes enabled, granting them access to the full functionality of the streaming app. In case the user fails to verify their email address by clicking the link, their account remains disabled until the verification process is completed. This email verification mechanism enhances security and helps maintain the integrity of the user accounts within the streaming app.

Login :



The login page serves as the gateway for users to access their accounts within the streaming app. Users are required to enter their username and password in the designated fields. Upon submission, the entered credentials are verified for correctness. If the provided username and password match the records in the system, the user is successfully authenticated and redirected to the home page. However, if the entered credentials are incorrect, an error message is displayed, informing the user that their login details are invalid. This feedback helps users identify and rectify any mistakes in their credentials. By providing a seamless login experience, the streaming app ensures secure access to user accounts while maintaining a user-friendly interface.

Header:



As Admin

The header of the streaming app prominently features a navigation bar that provides users with access to various functionalities of the application. Located within the header is a search bar, allowing users to easily search for specific content they are looking for. Adjacent to the search bar, there are links to the app's social media platforms, including Telegram, Discord, and Reddit, providing users with additional channels to engage and connect with the streaming app's community. Additionally, there is a donation link to support the app's maintenance and development efforts, enabling users to contribute to the sustainability of the platform.

To cater to users' preferences, the header also includes icons that allow them to navigate to different sections of the app. For instance, there is an icon leading to the anime page, which aggregates data scraped from various websites dedicated to anime content. Another icon directs users to the movies page, which displays a collection of movies sourced from a dedicated movies

API. Additionally, there is a separate movies section that features content uploaded directly by the app's administrators, providing users with a curated selection of movies.

Located in the top right corner of the header, there is a login button that provides a convenient way for users to access their accounts. Clicking on the login button redirects users to the login page, where they can enter their credentials to authenticate and gain full access to the streaming app's features and personalized content. The inclusion of the login button in the header ensures easy and quick access to the login functionality, enhancing the user experience and streamlining the login process.

Home:

The home page of our streaming app offers a captivating experience for users with its well-curated sections. At the top, users will find a spotlight section that showcases the most popular and influential animes, ensuring they don't miss out on the latest trends and highly acclaimed series.

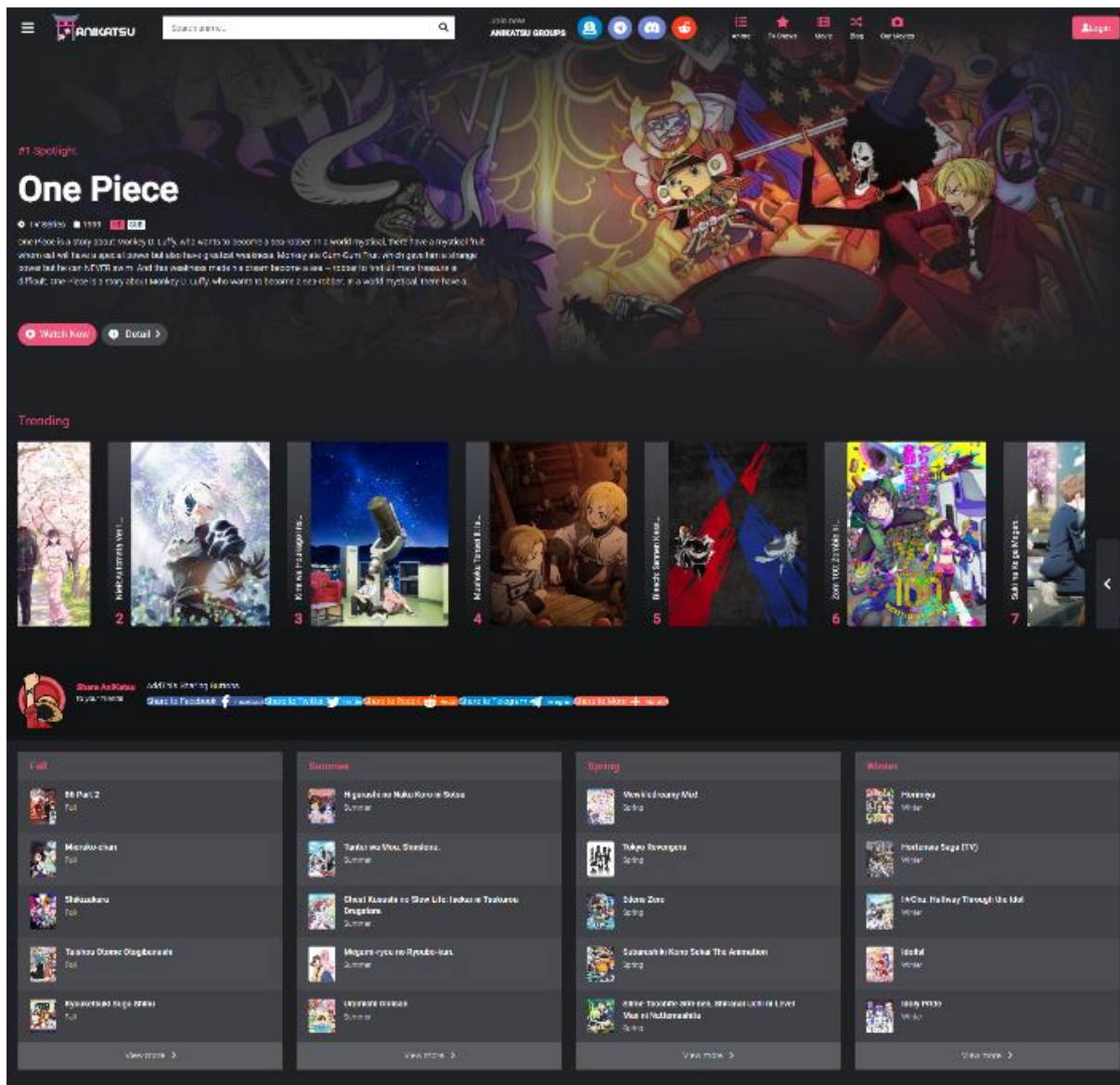
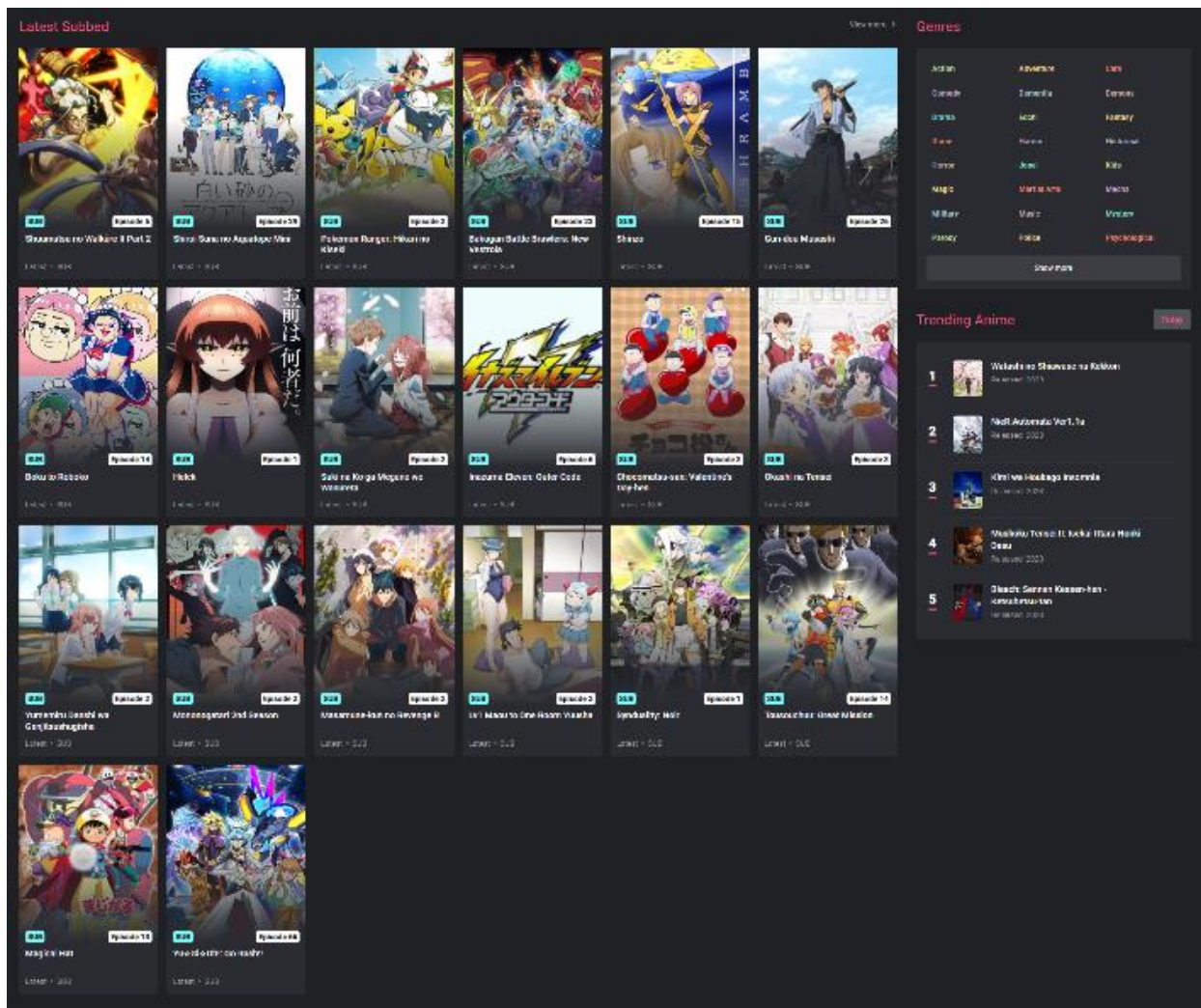
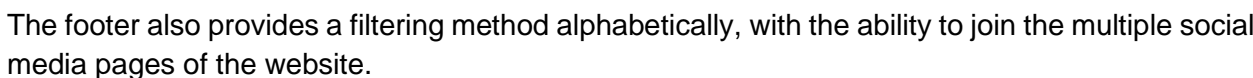


Figure 17 : Home page of the application

Additionally, the home page features a dedicated section for trending animes, providing users with a quick glimpse into the most talked-about and highly anticipated shows. For users who prefer subtitled content, there is a section exclusively dedicated to subbed anime, offering a vast selection of series with subtitles in various languages.



Likewise, users seeking dubbed content can explore the dedicated section for dubbed anime, providing a diverse range of anime with professionally dubbed audio tracks.



List of Anime:

On this page, users can conveniently browse through an extensive collection of animes, organized alphabetically or categorized based on specific genres. The user-friendly interface allows for effortless navigation and exploration of the vast range of anime titles available

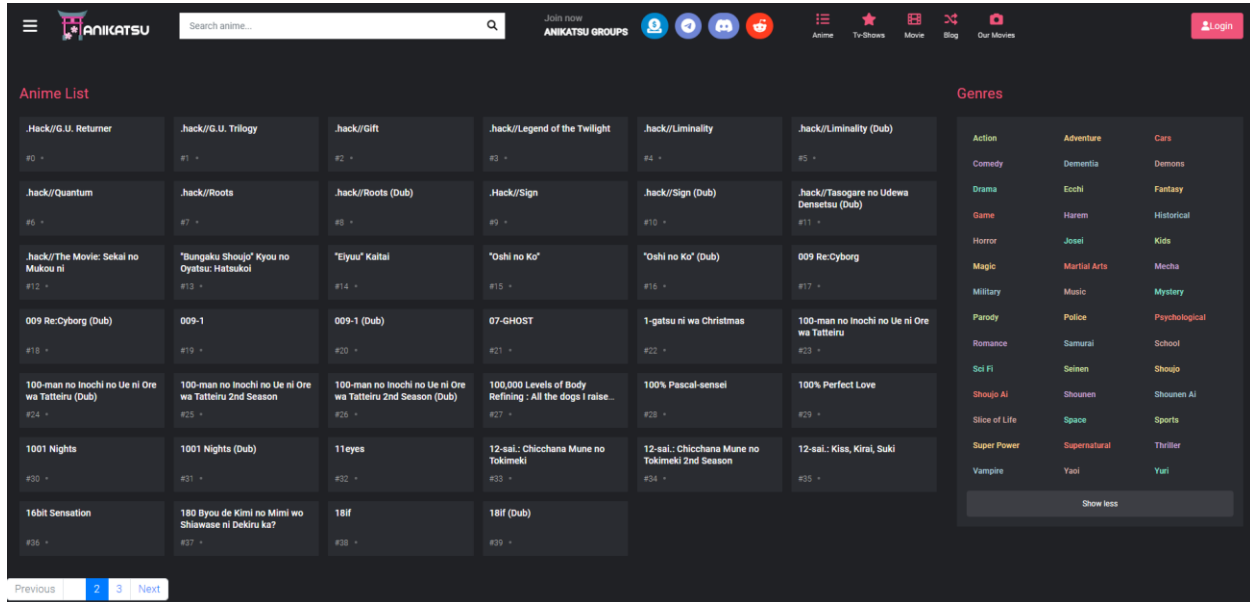


Figure 18 : Anime list display

List of TV shows:

In this version of the list, we not only provide the titles but also display the covers, seasons, and episode numbers, all organized based on their release dates. Moreover, users have the flexibility to filter the list by genre or utilize the convenient search bar at the top to find specific content they are looking for.

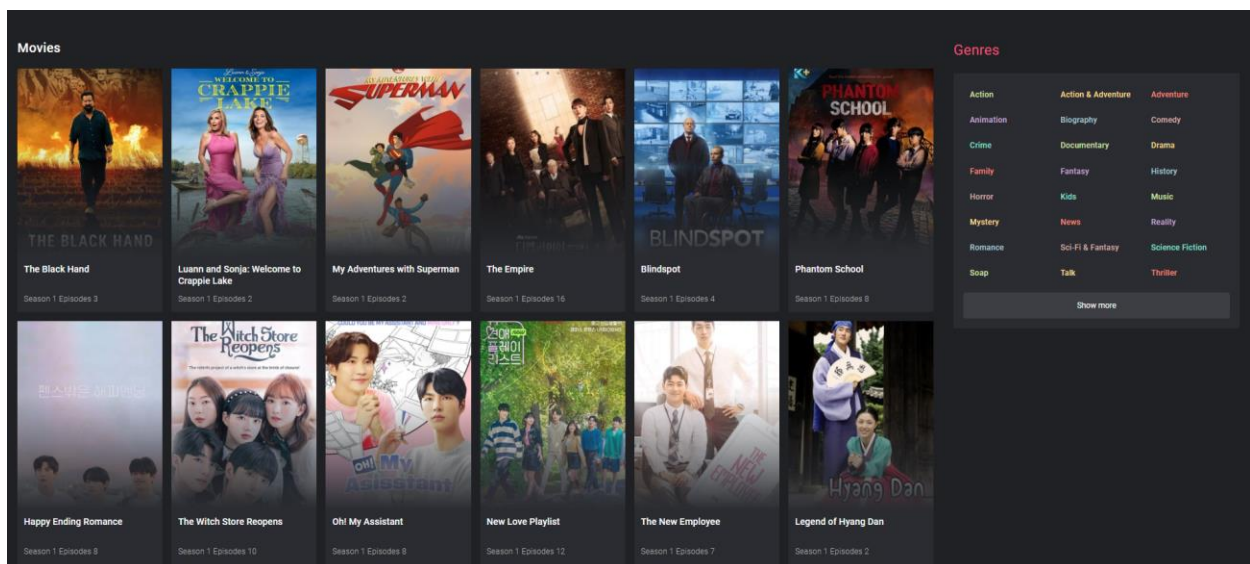


Figure 19 : TV shows and series display

List of Scraped Movies:

Here we are displaying the movies with their covers too, their titles and their release dates.

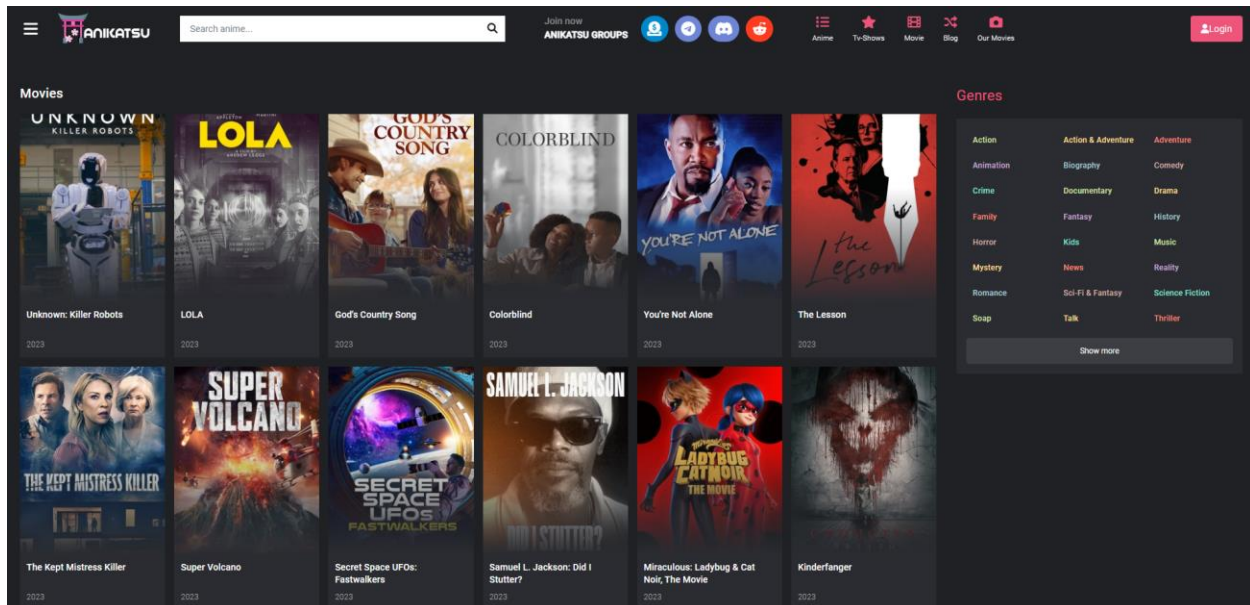


Figure 20 : List of scraped Movies

List of Our Movies:

Alongside with the movies that we are scraping, there will be a special area where we will display the movies that the website has published, these movies can be either short videos, films, conferences or anything artistic or cultural.

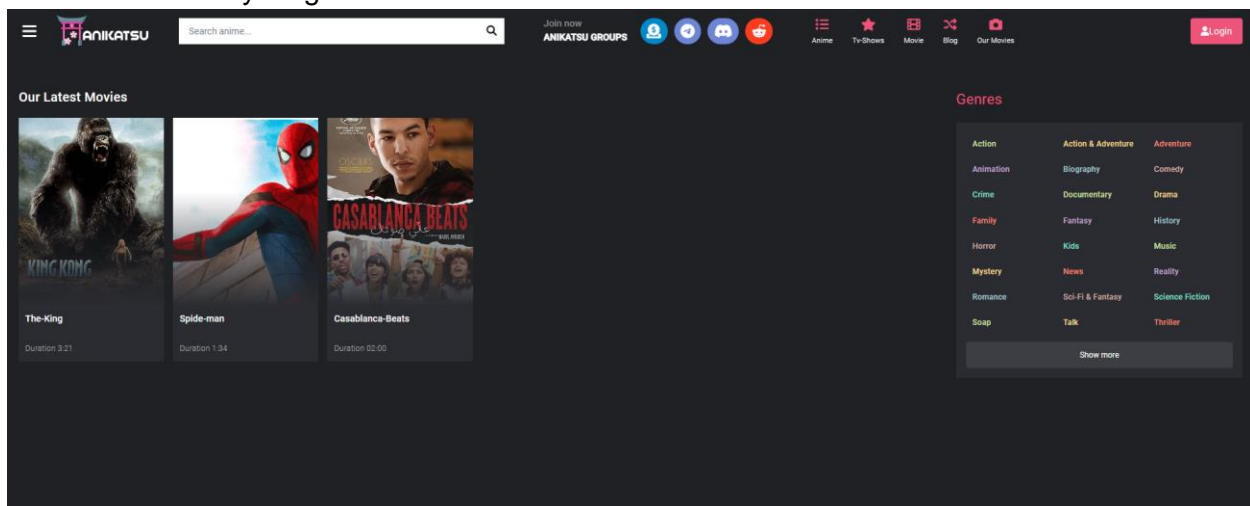


Figure 21 : List of Our own movies

Upon clicking on one of these cards, the user will be redirected towards a details page, where he will be able to read about the movie, view its votes and start watching it.

Details:

In addition, the user will have the option to interact with the content by clicking on a designated button, enabling them to initiate immediate playback. Alternatively, they can choose to add the content to their personalized watchlist, allowing them to conveniently access and enjoy it at a later time.

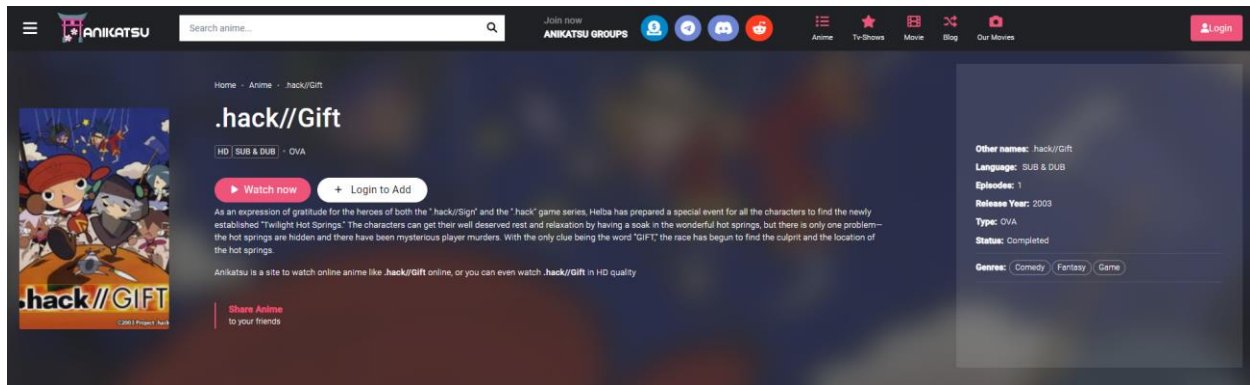


Figure 22 : Movie Details when not logged in

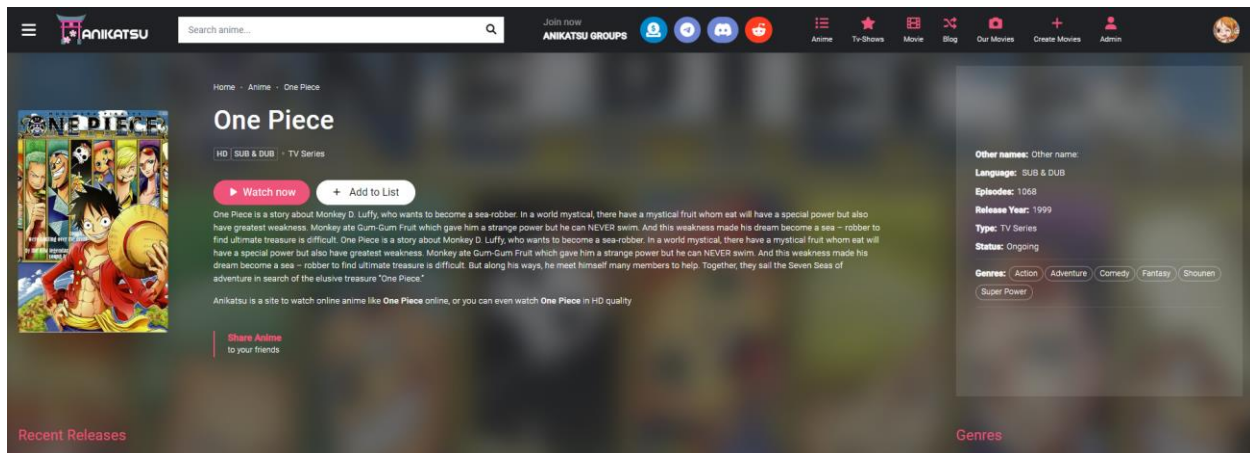


Figure 23 : Movie details when logged In

Watching a Movie :

Once the user clicks on the watch button, they will be seamlessly redirected to a dedicated page specifically designed for an immersive viewing experience. On this page, the user will not only have the opportunity to view comprehensive details about the movie on the right side but also gain access to a highly customizable player. The customizable player provides an array of options and features, allowing the user to tailor their viewing experience according to their preferences, whether it's adjusting the playback quality, enabling subtitles, or controlling playback speed.

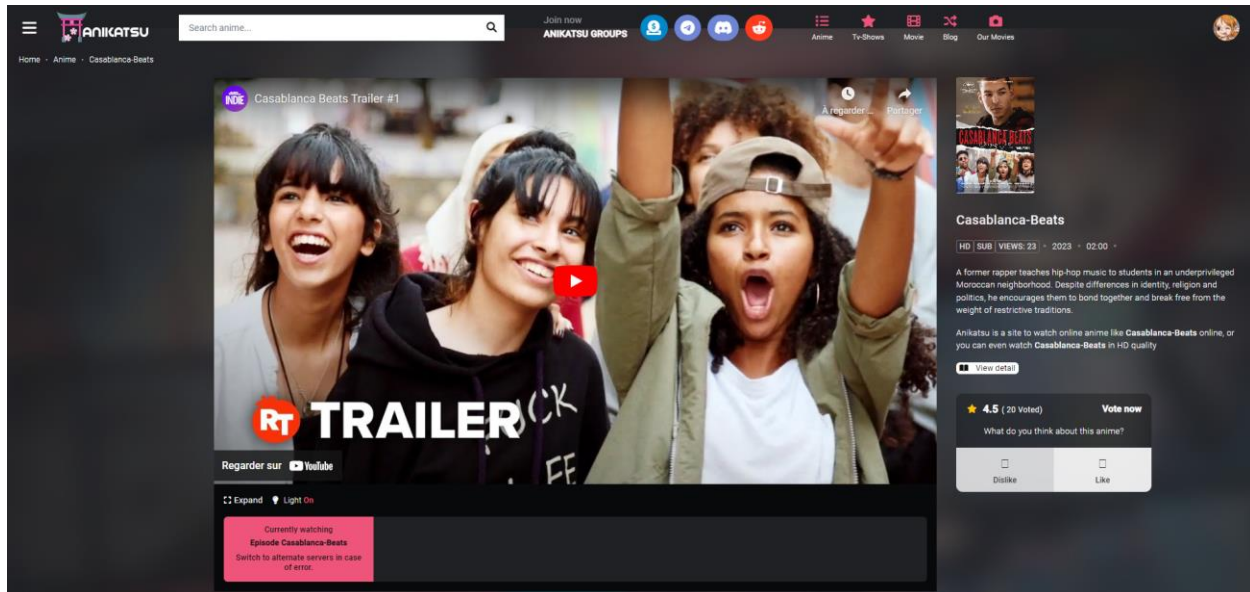


Figure 24 : Watch Movies

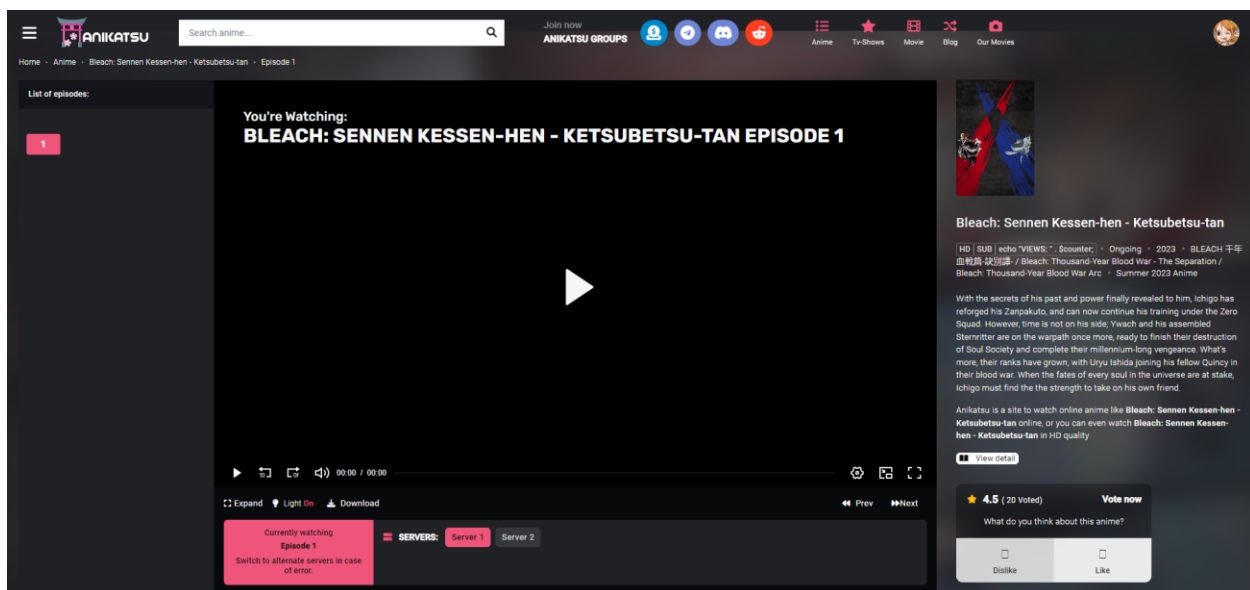


Figure 25 : Watch anime or tv shows with episodes

Expressing opinions on the spot :

At the bottom of the player, you will find a comment section powered by Tiny, a feature-rich platform that enhances the customization of your comments. This section serves as a valuable platform for users to express their instant opinions and share their thoughts about the movie or episode they have just watched. With the help of Tiny, users can personalize their comments by adding formatting, emojis, and even multimedia elements, making their feedback more engaging and expressive. This interactive comment section fosters a sense of community engagement, enabling users to connect with others and participate in dynamic discussions related to the content they have experienced.

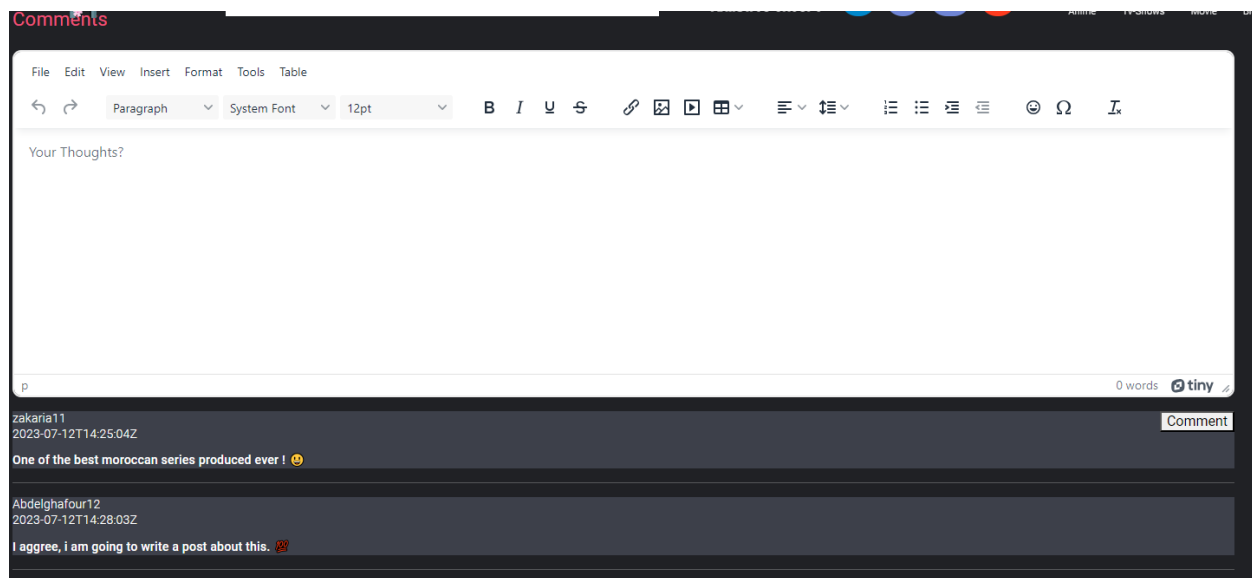


Figure 26 : Commenting on movies/ episodes

The blog:

The connected user is granted access to the blog, where a wealth of information awaits. Upon entering the blog, they will be greeted with a visually appealing layout featuring post tiles, providing a concise preview of the diverse range of content shared by people. Furthermore, the user will have the opportunity to explore all the subreddits, which represent the main subjects being discussed within the blog's community.

In this immersive environment, the user gains the ability to actively engage with the content. They can create their own posts, generating discussions and sharing their perspectives on various topics. Additionally, they have the privilege to create and manage their own subreddits, tailoring the content to their specific interests and fostering a dedicated community.

Moreover, the user can read through posts, vote on them to express their preference or agreement, and leave valuable comments to initiate meaningful conversations. This two-way interaction empowers the user to contribute their insights, exchange ideas, and connect with

fellow blog participants. By providing a platform for content creation and curation, the blog ensures an engaging and interactive experience for the connected user

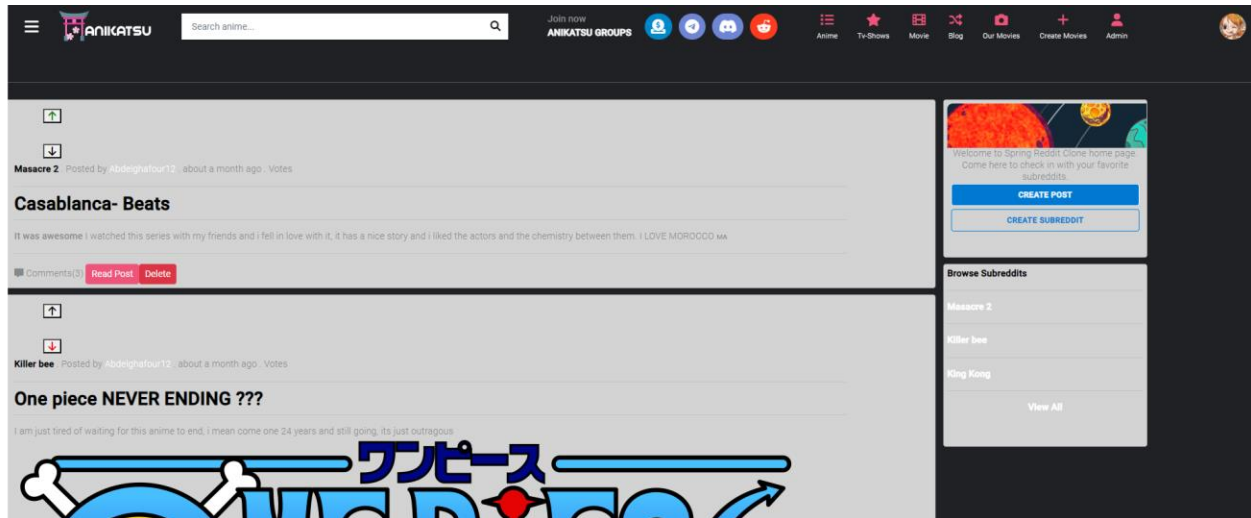


Figure 27 : Blog page and posts Tile

Create post :

Filling up this forms allows the user to post on the blog under the category he chose where others can view it and interact with it, he will have the possibility to manage it anytime.

The image shows a 'Create Post' form. It has a title input field, a URL input field, and a 'Select Subreddit' dropdown menu. Below these is a rich text editor with a toolbar containing icons for undo, redo, paragraph, bold, italic, bulleted list, numbered list, link, and unlink. The text area is empty. At the bottom right of the text area is a 'tiny' logo. At the bottom of the form are two buttons: 'DISCARD' and 'POST'.

Figure 28 : Form to post on the blog

Read Posts and Comment on them:

On the post details page, the user can view all the comments related to that post, he can add his own too and customize it.

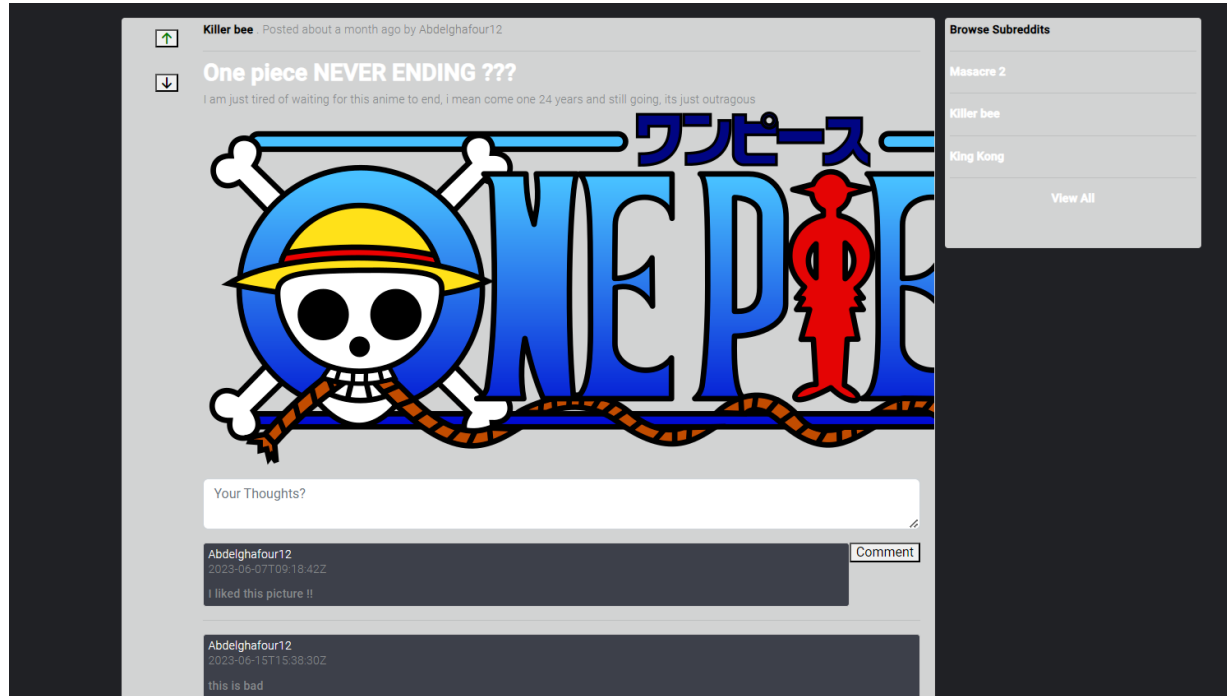


Figure 29 : Viewing Post details and comments related to it

User Profile:

The connected user can access his profile where he can view his details and his posts that he can manage.

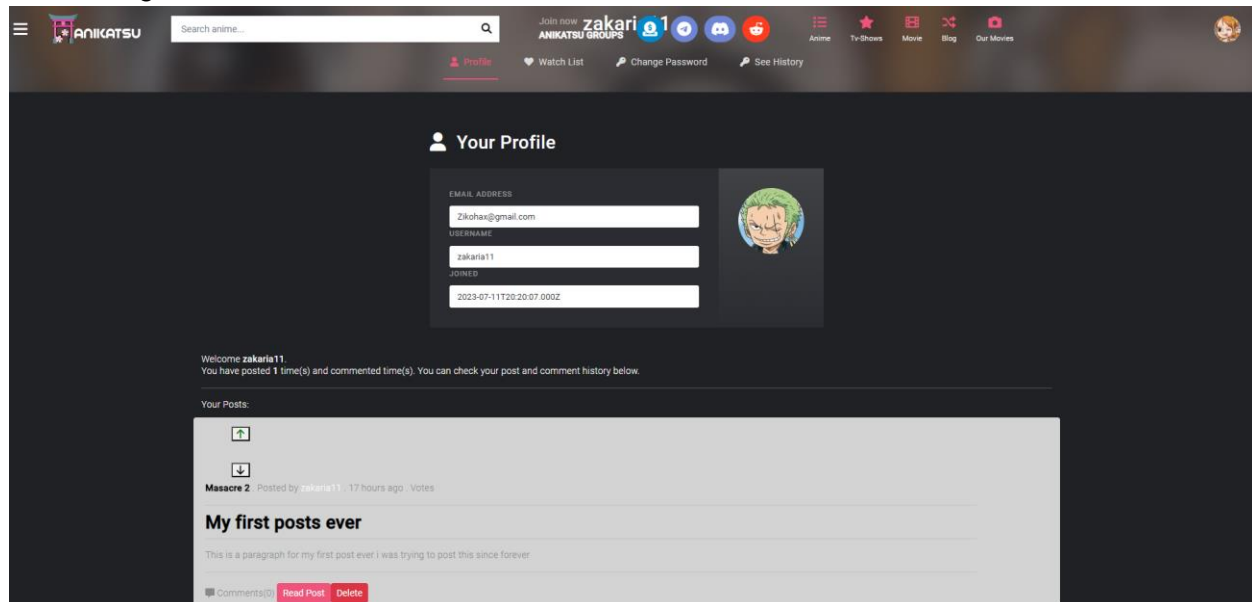


Figure 30 : User profile details with his own posts displayed

The connected user is provided with seamless access to their personalized profile, which serves as a centralized hub for managing their account and activities within the platform. Within the profile section, the user can conveniently view and update their personal details, ensuring that their information remains accurate and up to date.

Furthermore, the user can effortlessly navigate to their dedicated posts section within the profile, granting them full control over their own contributions. Here, they can manage their posts with ease, including editing, deleting, or organizing them as desired. This level of autonomy empowers the user to curate their content and maintain an organized presence within the platform.

By offering a dedicated profile area, the platform ensures that the connected user has a comprehensive overview of their account details and retains full autonomy over their own posts, facilitating a personalized and streamlined user experience.

User watchlist:

On the menu, the user can also view his own watchlist where he was saving his movies, in there he can remove some of them when he wants to.

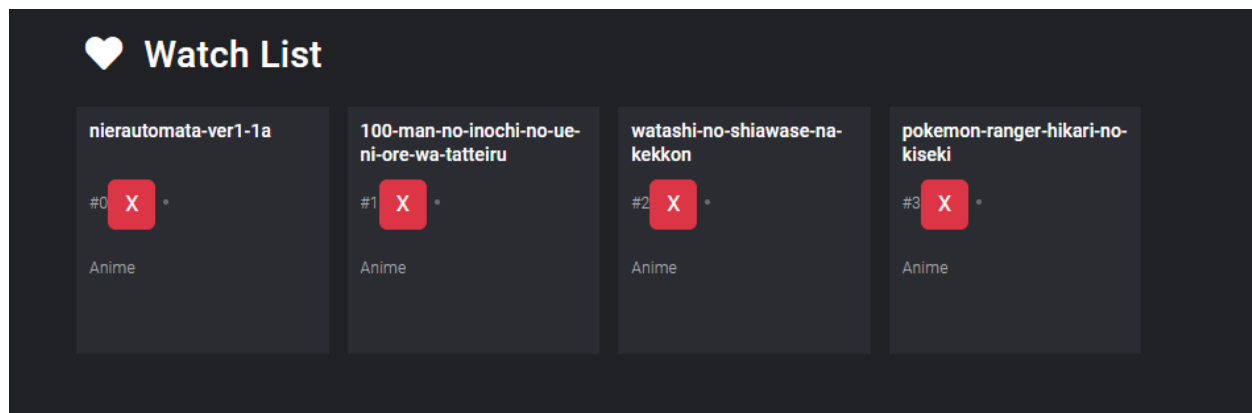


Figure 31 : User watchlist display and delete button

In addition to the aforementioned features, the user will find a dedicated section in the menu where they can access their personalized watchlist. This watchlist serves as a curated collection of movies that the user has saved for future viewing. It provides a convenient and centralized location for the user to keep track of their desired movies.

Within the watchlist, the user has the flexibility to manage their saved movies according to their preferences. They can easily remove movies from the watchlist whenever they choose to do so, allowing for effortless organization and customization of their viewing choices.

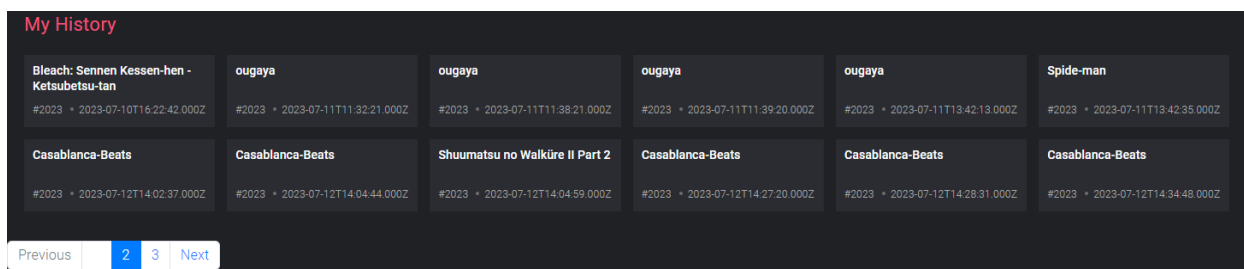
This functionality empowers the user to maintain a personalized and tailored watchlist, ensuring that they have quick access to their favorite movies and the ability to curate their viewing experiences based on their changing interests and preferences.

User History:

The user is provided with convenient access to their viewing history, allowing them to revisit and review their recently watched movies. Within the platform, they can easily navigate to the history section, which provides a comprehensive record of their viewing activity.

By accessing their history, the user can conveniently track and reference the movies they have recently watched. This feature enables them to keep a log of their viewing habits, helping them remember past choices and potentially revisit movies they enjoyed or wish to explore further.

With the ability to access and review their viewing history, the user can effortlessly stay informed about their past movie selections and enhance their overall viewing experience within the platform.



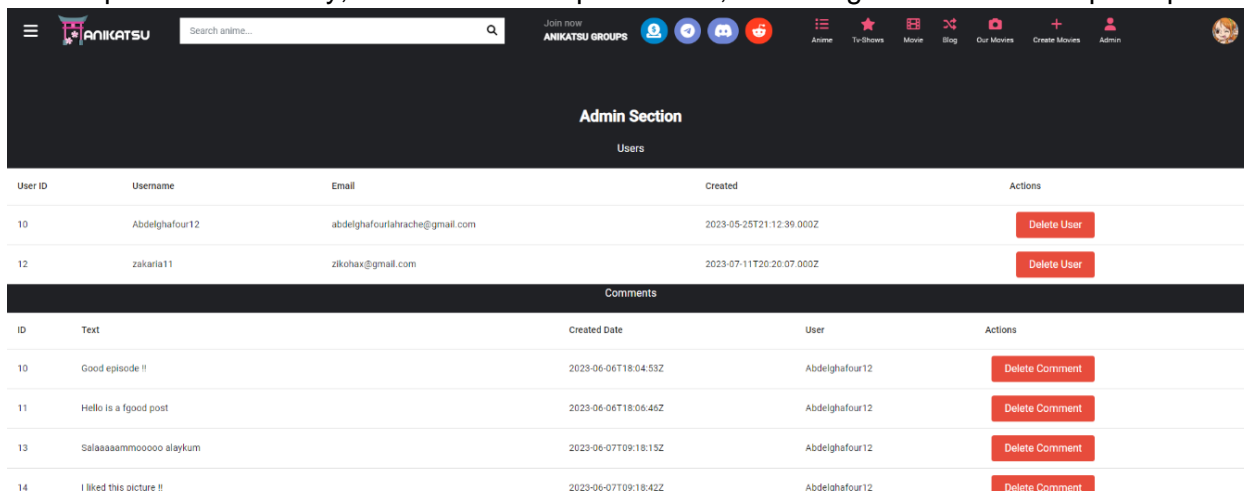
My History					
Bleach: Sennen Kessen-hen - Ketsubetsu-tan #2023 • 2023-07-10T16:22:42.000Z	ougaya #2023 • 2023-07-11T11:32:21.000Z	ougaya #2023 • 2023-07-11T11:38:21.000Z	ougaya #2023 • 2023-07-11T11:39:20.000Z	ougaya #2023 • 2023-07-11T13:42:13.000Z	Spide-man #2023 • 2023-07-11T13:42:35.000Z
Casablanca-Beats #2023 • 2023-07-12T14:02:37.000Z	Casablanca-Beats #2023 • 2023-07-12T14:04:44.000Z	Shuumatsu no Walküre II Part 2 #2023 • 2023-07-12T14:04:59.000Z	Casablanca-Beats #2023 • 2023-07-12T14:27:20.000Z	Casablanca-Beats #2023 • 2023-07-12T14:28:31.000Z	Casablanca-Beats #2023 • 2023-07-12T14:34:48.000Z
Previous	2	3	Next		

Figure 32 : User watch history

Admin Page :

As an admin, the user is granted additional privileges and responsibilities within the platform.

With the power to ban users, the admin can enforce community guidelines and ensure a safe and respectful environment for all users. In cases where a user's behavior violates platform policies or disrupts the community, the admin can impose a ban, restricting their access and participation.



Admin Section				
Users				
User ID	Username	Email	Created	Actions
10	Abdelghafour12	abdelghafourlahache@gmail.com	2023-05-25T21:12:39.000Z	Delete User
12	zakaria11	zikohax@gmail.com	2023-07-11T20:20:07.000Z	Delete User

Comments				
ID	Text	Created Date	User	Actions
10	Good episode !!	2023-06-06T18:04:53Z	Abdelghafour12	Delete Comment
11	Hello is a fgood post	2023-06-06T18:06:46Z	Abdelghafour12	Delete Comment
13	Salaassammooooo alaykum	2023-06-07T09:18:15Z	Abdelghafour12	Delete Comment
14	I liked this picture !!	2023-06-07T09:18:42Z	Abdelghafour12	Delete Comment

Figure 33 : Admin section : delete comments/ ban users

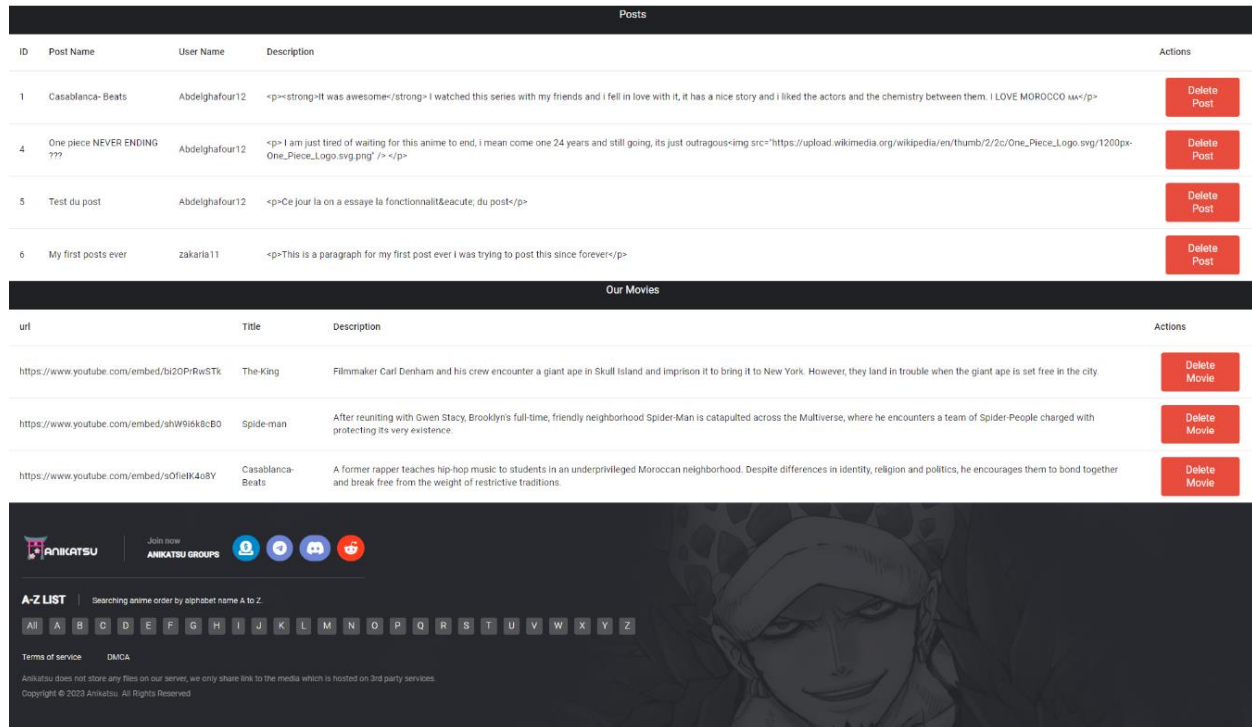


Figure 34 : Admin section : delete Posts / movies

Furthermore, the admin has the ability to remove posts that violate content guidelines or are deemed inappropriate. This proactive approach helps maintain the platform's integrity and ensures that the content aligns with the desired standards and objectives.

Additionally, the admin can address and remove bad comments that are considered inappropriate or violate community guidelines. By actively moderating the comments section, the admin can foster a positive and constructive discussion space for users.

These admin privileges empower them to safeguard the platform's reputation, maintain a respectful community, and create a welcoming environment for all users.

Create Movies :

As an admin, they hold the authority to create and add movies to the platform. This allows them to expand the content library and offer a diverse range of movies for users to explore and enjoy.

The admin can utilize their privileged access to add new movies to the platform's database. They can input relevant details such as the movie title, genre, release date, cast, and synopsis, ensuring that the movie information is comprehensive and accurate.

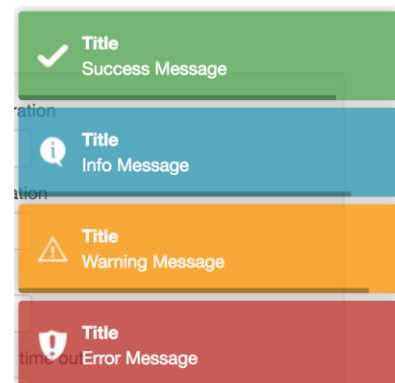
Figure 35 : Form for the admin to fill if he wants to post a movie

By creating and adding movies, the admin enhances the platform's content offerings, keeping it up-to-date with the latest releases and catering to the evolving preferences of the user base. This feature allows for a dynamic and engaging user experience, providing a wide selection of movies for users to discover and enjoy.

Notification system:

We implemented the notification system in our Angular application using the Toastr library. Toastr is a popular notification library for displaying non-intrusive and customizable alerts to users.

With Toastr, we were able to easily show notification messages for various events and actions within the application. For example, when a user successfully subscribes or changes their password, a success notification is displayed. Likewise, when someone likes or comments on a user's post, a notification is shown to inform the user about the interaction.



General Conclusion

In conclusion, the realization of this project, which involved the development of a streaming and blogging application for movie reviews, has provided a valuable solution for users who seek a platform to discover, watch, and discuss movies. The need for such an application arises from the increasing popularity of streaming services and the growing demand for user-generated content and discussions related to movies.

To address this need, we adopted the Spring-Angular approach, which leverages the Spring framework on the backend and AngularJS on the frontend. This choice allowed us to benefit from the robustness and flexibility of the Spring framework for building a secure and scalable backend API. At the same time, AngularJS provided a powerful and user-friendly frontend framework for creating interactive and dynamic interfaces.

Throughout the project, we followed a structured development process, starting with the design phase where we conceptualized the application's features and functionalities. We then proceeded to implement the backend and frontend components, ensuring seamless integration between them. Thorough testing and validation were conducted to verify the correct functioning of the application, and necessary adjustments were made based on the test results.

The resulting application is now ready to be deployed and offers a comprehensive platform for users to explore a wide range of movies, read and contribute to movie reviews and discussions, and manage their own profiles. It provides a seamless and engaging user experience, thanks to its intuitive interface, customizable player, and interactive comment section.

Overall, this project has successfully demonstrated the effectiveness of the Spring-Angular approach in developing a feature-rich streaming and blogging application. It has showcased the importance of a well-defined design phase, followed by meticulous development, testing, and validation processes, resulting in a reliable and user-centric application ready for deployment.

Bibliography

<https://nodejs.org>. [En ligne ; consulté le 21-06-2023].

<https://angular-ui.github.io>. [En ligne ; consulté le 20-06-2023].

<https://projects.spring.io/spring-security>. [En ligne ; consulté le 26-06-2023].

<https://httpd.apache.org/>. [En ligne ; consulté le 21-05-2023].

<https://www.w3.org/TR/webstorage/#the-localstorage-attribute>. [En ligne ; consulté le 26-05-2023].

<https://angular.io/guide/what-is-angular>. [En ligne ; consulté le 26-05-2023].

<https://docs.spring.io/spring-framework/reference/>. [En ligne ; consulté le 13-05-2023].

<https://getbootstrap.com/>. [En ligne ; consulté le 21-05-2023].

<https://www.getpostman.com>. [En ligne ; consulté le 21-05-2023].

<https://martinfowler.com/articles/injection.html>. [En ligne ; consulté le 21-05-2023].

<https://docs.spring.io/spring-data/rest/docs/current/reference/html>. [En ligne ; consulté le 13-05-2023].

<https://git-scm.com/>. [En ligne ; consulté le 15-05-2023].