SRI KRISHNA COLLEGE OF TECHNOLOGY
An Autonomous Institution | Accredited by NAAC with 'A' Grade
Affiliated to Anna University | Approved by AICTE
KOVAIPUDUR, COIMBATORE 641042

ANIME WORLD

A PROJECT REPORT

Submitted by

CHITHARANJAN.T **727821TUAD010**

In partial fulfilment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

JULY 2023

BONAFIDE CERTIFICATE

Certified that this project report **“ANIME WORLD”** is the Bonafide workof, **CHITHARANJAN.T** who carried out the project work under my supervision.

SIGNATURE	SIGNATURE
Mr. Manoj Kumar.R	**Dr. C.P. MAHESHWARAN**
SKILL DEVELOPMENT ENGINEER IAMNEO Coimbatore-42.	**HEAD OF THE DEPARTMENT** Professor Department of Artificial Intelligence and Data Science Sri Krishna College of Technology Kovaipudur Coimbatore-42.

Certified that the candidates were examined by us in the Project Work viva-voce examination held on .... at Sri Krishna College of Technology, Coimbatore -641 042.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost, we thank the **Almighty** for being our light and for showering his gracious blessings throughout the course of this project.

We are grateful to our beloved Principal **Dr.M.G. Sumithra M.E., Ph.D.** for her tireless and relentless support.

We extend our sincere thanks to our Head of the Department **- Dr.C.P. MaheshWaran** for his encouragement and inspiration.

We are greatly indebted to our Industry Mentor **Mr. Manoj Kumar.R** for his valuable guidance and suggestions in all aspects that aided us to ameliorate our skills.

We are thankful to all those who have directly and indirectly extended their help to us in completing this project work successfully.

ABSTRACT

The App Anime world is an representation of online video streaming platform that exists in use these days. YouTube was the foremost introduction of social media that is in use. By these day there come a lot of services out there and are being enjoyed by peoples. Is there any real use for these Streaming Service App and the answer is Yes, The Streaming service app is developed to provide users with a seamless and convenient way to access and enjoy a wide range of multimedia content, including movies, TV shows, music, and more.

Now this application has features such as looking through all Anime Episode uploaded, Search for a Specific Episode, Maintaining login and logout states, also registering new users. Out of these application as a project I would like to represent my knowledge about Spring Boot which is a backend framework for java, ReactJS which is an front-end frame work for JavaScript and integrating both these front-end and back-end application and creating and maintaining an Fullstack application.

Key Words – Login, Signup, Settings, Home and etc.

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIVE	ABBREVIATION
API	Application Programming Interface
CRUD	Create Read Update Delete
MVC	Model View Controller
REST	Representational State Transfer
URL	Uniform Resource Locator

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

Streaming apps such as YouTube, Netflix and etc.., Are doing good, Where users have instant access to a vast library of multimedia content at their fingertips. They can enjoy their favorite movies, TV shows, music, and more anytime, anywhere, and on any device. Whether it's a smartphone, tablet, smart TV, or streaming device, users can seamlessly stream content and create their personalized entertainment experience. The ability to pause, resume, and pick up where they left off across different devices adds to the convenience. Moreover, streaming apps often provide personalized recommendations based on user preferences, making it easier for users to discover new content that aligns with their interests

Streaming service apps can encounter several challenges that impact user satisfaction. These include limited content selection, making it difficult for users to find diverse and appealing options; content discovery issues, where ineffective algorithms or poor search functionality hinder the ability to explore and discover new content; inconsistent streaming quality, leading to buffering, interruptions, and low-resolution playback; complex user interfaces that confuse and frustrate users and platform fragmentation, requiring users to switch between multiple apps or platforms to access different types of content.

1.2 OVERVIEW

I have developed an Online video Streaming application which has the similar work flow of any other Streaming application. Most of the applications are similar in their working where the application that what has been built my me have also implemented those features.

My streaming service app is a cutting-edge platform that revolutionizes the way users access and enjoy multimedia content. With a user-friendly interface and seamless streaming capabilities, our app provides an unparalleled entertainment experience.

1.3 OBJECTIVES

The ideology of my project is that was to create an app similar to a famous app called ANIMIXPLAY, Which is a famous online streaming platform streams Animated video alias “ANIME” which are produced and directed by Japanese Studios such as “MAPPA”, “STUDIO PIERROT”, “TOEI ANIMATION” and etc.. . Then the app was Shut-down due some technical reason.

Now the term “ANIME” was becoming famous all-over the world, And my aim to is recreate application similar to the previous one, And to offer a vast and diverse content library, ensuring that users have access to a wide range of anime content. This will cater to different tastes and preferences, increasing user satisfaction and retention.

CHAPTER 2

LITERATURE SURVEY

Building a Streaming application using Spring Boot and ReactJS is an exciting endeavor that requires a comprehensive literature survey to gather insights and best practices. The literature survey begins by exploring the landscape of Streaming applications, their features, and their societal impact. It then delves into the advantages of using the Spring Boot framework for web application development, emphasizing its scalability, robustness, and efficiency. Concurrently, the survey examines the significance of ReactJS in creating interactive user interfaces and explores its integration with Spring Boot for a full-stack application.

An streaming app built using Spring Boot and ReactJS involves exploring existing research, articles, and publications related to the development of streaming apps using these technologies. The survey aims to gather insights into best practices, frameworks, and design patterns specific to Spring Boot and ReactJS that can be applied to the development of an anime streaming app.

The survey will cover literature on Spring Boot, focusing on topics such as building RESTful APIs, handling authentication and authorization, database integration, and performance optimization. It will explore the latest trends and advancements in Spring Boot development, including popular libraries, tools, and techniques that can enhance the app's functionality, security, and scalability.

CHAPTER 3

SYSTEM SPECIFICATIONS

3.1 BACKEND DEVELOPMENT

3.1.1 SpringBoot

Spring Boot is an open source Java-based framework used to create amicro Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications. This chapter will give youan introduction to Spring Boot and familiarizes you with its basic concepts.

Micro Service is an architecture that allows the developers to developand deploy services independently. Each service running has its own process and this achieves the lightweight model to support business applications.

3.1.2 Features of SpringBoot

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' dependencies to simplify your build configuration
- Automatically configure Spring and 3rd party libraries whenever possible
- Provide production-ready features such as metrics, health checks, and externalized configuration
- Absolutely no code generation and no requirement for XML configuration

3.1.3 SpringBoot Library

Spring Boot starter libraries are some of the main building blocksof Spring Boot applications. As Spring Boot developers, we've all grown accustomed to some of our favorite starter libraries, such as spring-boot-starter-web, spring-boot-starter-data-jpa, spring-boot-starter-actuator , etc.

3.1.4 History of SpringBoot

In October 2002, Rod Johnson wrote a book titled Expert One-on-One J2EE Design and Development. Published by Wrox, this book covered the state of Java enterprise application development at the time and pointed out a number of major deficiencies with Java EE and EJB component framework.In the book he proposed a simpler solution based on ordinary java classes (POJO - plain old java objects) and dependency injection.

Over years spring framework has grown substantially. Almost all infrastructural software components required by a java enterprise application is now available in spring framework. However collecting all the required spring components together and configuring them in a new application requires some effort. This involves setting up library dependencies in gradle/maven and then configuring the required spring beans using xml,annotations or java code.

3.1.5 SpringBoot Tools

- **Spring Boot Starter:** Spring Boot provides a set of starter dependencies that make it easy to include commonly used libraries andframeworks in your project. For example, "spring-boot-starter-web" includes everything you need to build web applications.

- **Spring Boot Autoconfiguration:** Spring Boot automatically configures various components based on the dependencies present in your project. It reduces the amount of manual configuration required, making development faster and more straightforward.

- **Spring Boot Actuator:** Actuator provides production-ready features to monitor and manage your application. It includes endpoints for health checks, metrics, monitoring, and management tasks. You can use it to gain insights into the internals of your application during runtime.

- **Spring Boot CLI:** The Spring Boot Command-Line Interface (CLI) is a tool for quickly creating and running Spring Boot applications from the command line. It allows you to write Groovy scripts that can bootstrap your application, run it, and perform various tasks.

- **Spring Boot Dev-Tools:** DevTools is a set of utilities that improve the development experience. It provides features like automatic application restarts, live reloading of static resources, and enhanced error reporting.

- **Spring Boot Testing:** Spring Boot provides testing support through the Spring Test framework. It offers utilities for writing unit tests, integration tests, and end-to-end tests. It also includes features like mocking and dependency injection for testing components.

- **Spring Data JPA:** Spring Data JPA is a part of the Spring Data project that simplifies database access using the Java Persistence API (JPA). It provides a powerful and convenient way to interact with databases by eliminating the need for boilerplate code.

- **Spring Security:** Spring Security is a robust framework for implementing security in Spring applications. It provides features like authentication, authorization, and various security mechanisms to protect your application.

3.2 FRONTEND DEVELOPMENT

React is a JavaScript library created by Facebook for building user interfaces It is a component-based, declarative, and highly efficient library that is used to develop interactive UIs (user interfaces) for single page web applications. React uses a virtual DOM (Document Object Model) that makes it faster and easier to manipulate the DOM elements. It also provides declarative components that allow developers to write code that is easy to read and maintain. React also offers an extensive library of tools and components that make it easier to develop complex user interface.

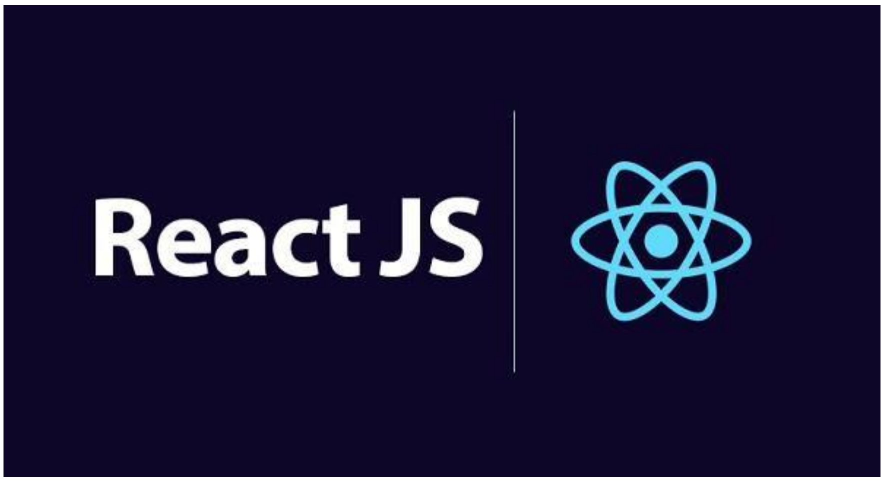
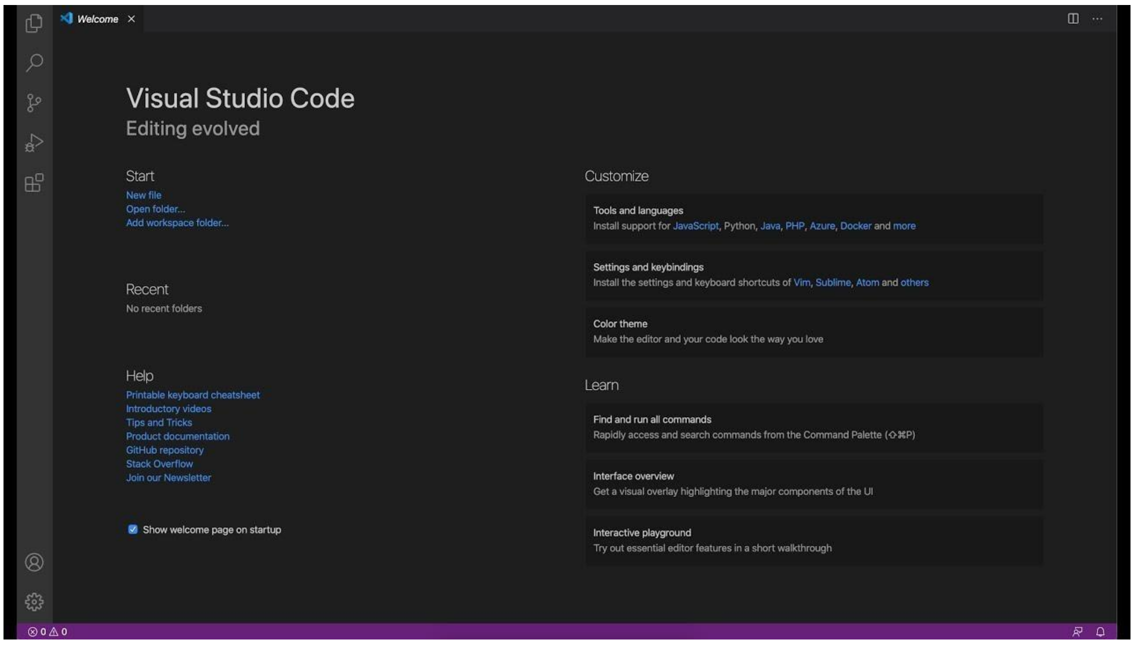


Fig 3.1 REACTJS

3.3 VS CODE

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux, and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is also customizable, so users can change the editor’s theme, keyboard shortcuts, and preferences.

VS Code is an excellent code editor for React projects. It is lightweight, customizable, and has a wide range of features that make it ideal for React development. It has built-in support for JavaScript, JSX, and TypeScript, and enables developers to quickly move between files and view detailed type definitions. It also has a built-in terminal for running tasks. Additionally, VS Code has an extensive library of extensions that allow developers to quickly add features like code snippets, debugging tools, and linting support to their projects.



Welcome

Visual Studio Code
Editing evolved

Start
New file
Open folder...
Add workspace folder...

Recent
No recent folders

Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow
Join our Newsletter

Show welcome page on startup

Customize

Tools and languages
Install support for JavaScript, Python, Java, PHP, Azure, Docker and more

Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime, Atom and others

Color theme
Make the editor and your code look the way you love

Learn

Find and run all commands
Rapidly access and search commands from the Command Palette (⇧⌘P)

Interface overview
Get a visual overlay highlighting the major components of the UI

Interactive playground
Try out essential editor features in a short walkthrough

⊗ 0 ⚠ 0

Fig 3.2 VS CODE

CHAPTER 4

PROPOSED SYSTEM

This chapter gives a small description about the proposed idea behind the development of our website.

4.1 PROPOSED SYSTEM

The Streaming app aims to provide a seamless and engaging streaming experience for anime enthusiasts. The app will be built using a combination of Spring Boot for back-end development and ReactJS for frontend development, leveraging their respective strengths to create a robust and interactive platform. The proposed system will feature a comprehensive content library with a wide selection of anime shows and movies, catering to diverse user preferences. Users will have the ability to browse and search for specific titles, as well as discover new anime through personalized recommendations based on their viewing history and preferences.

The app will provide a user-friendly interface with intuitive navigation, allowing users to easily explore and access their favorite anime. It will incorporate responsive design principles to ensure optimal viewing experiences across various devices, including smartphones, tablets, and desktops. To enhance the streaming experience, the proposed system will implement advanced video streaming techniques, leveraging the capabilities of Spring Boot to handle efficient video transcoding, adaptive streaming, and seamless playback. This will ensure smooth streaming without buffering and provide high-quality video and audio to users.

Overall, the proposed system aims to create a feature-rich and user-centric streaming app. By leveraging the capabilities of Spring Boot and ReactJS, it will deliver a seamless streaming experience, personalized recommendations, interactive features, and a visually appealing interface. The system will cater to the needs and preferences of anime enthusiasts, fostering engagement and satisfaction among users.

4.2 ADVANTAGES

A Streaming app offers numerous advantages that cater specifically to anime enthusiasts. Firstly, it provides a vast and diverse collection of anime shows and movies, offering users a wide range of content to explore and enjoy. This eliminates the need for physical media or searching for individual episodes, as users can access their favorite anime instantly with just a few clicks.

Secondly, an anime streaming app allows users to watch anime anytime, anywhere, and on any device. Whether it's a smartphone, tablet, smart TV, or desktop computer, users have the flexibility to enjoy their favorite anime on the go or in the comfort of their own homes. This convenience and accessibility add to the overall viewing experience.

Streaming apps support legal and authorized access to anime content, ensuring that users can enjoy their favorite shows while supporting the industry. By licensing content and adhering to regional restrictions, these apps provide a reliable and legitimate platform for fans to immerse themselves in the world of anime.

In conclusion, an streaming app offers advantages such as a vast content library, convenience and accessibility, personalized recommendations, language options, social engagement, and legal access to anime content. These features combine to provide an immersive, enjoyable, and community-driven anime-watching experience for fans worldwide.

CHAPTER 5

METHODOLOGIES

This chapter gives a small description about how our system works.

The Flowchart portrayed below just explains the administrators, how the project is working based upon the set of inputs/actions/gestures given by the user.

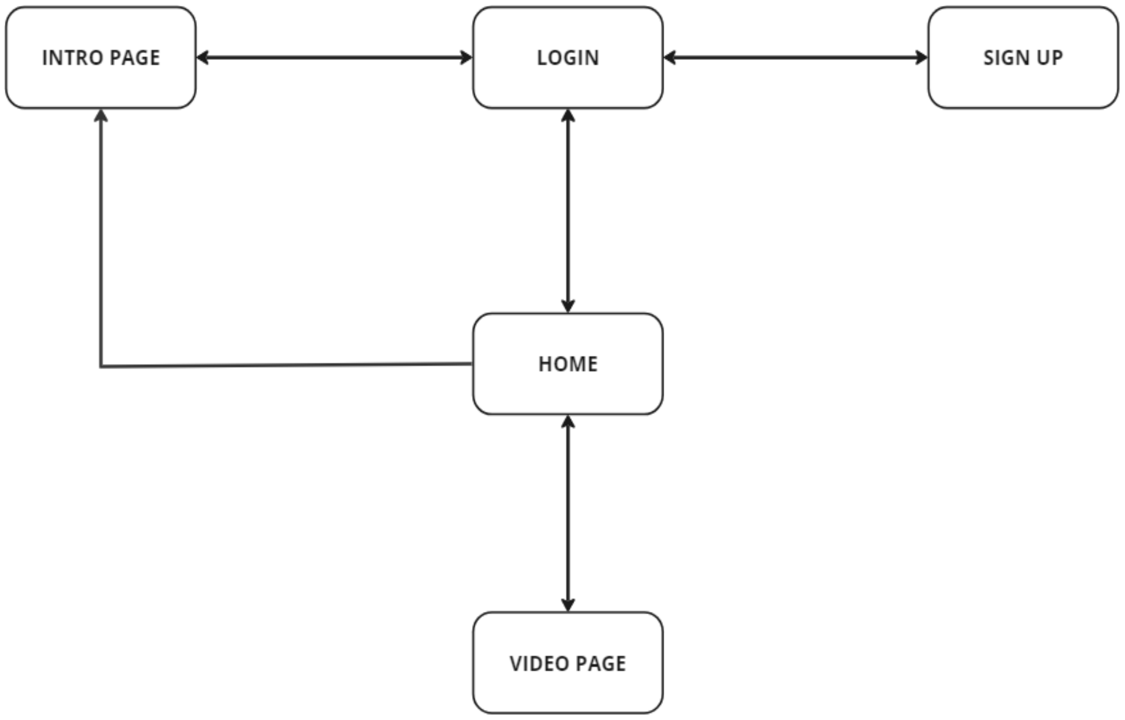


Fig 5.1 PROCESS FLOW DIAGRAM

5.1 LOGIN PAGE

In this page we will be asking about the username and password of the user. Firstly the website validates the user inputs. It verifies the username andpassword by checking it with the usernames and passwords stored in the localstorage when the user creates an account in the website.

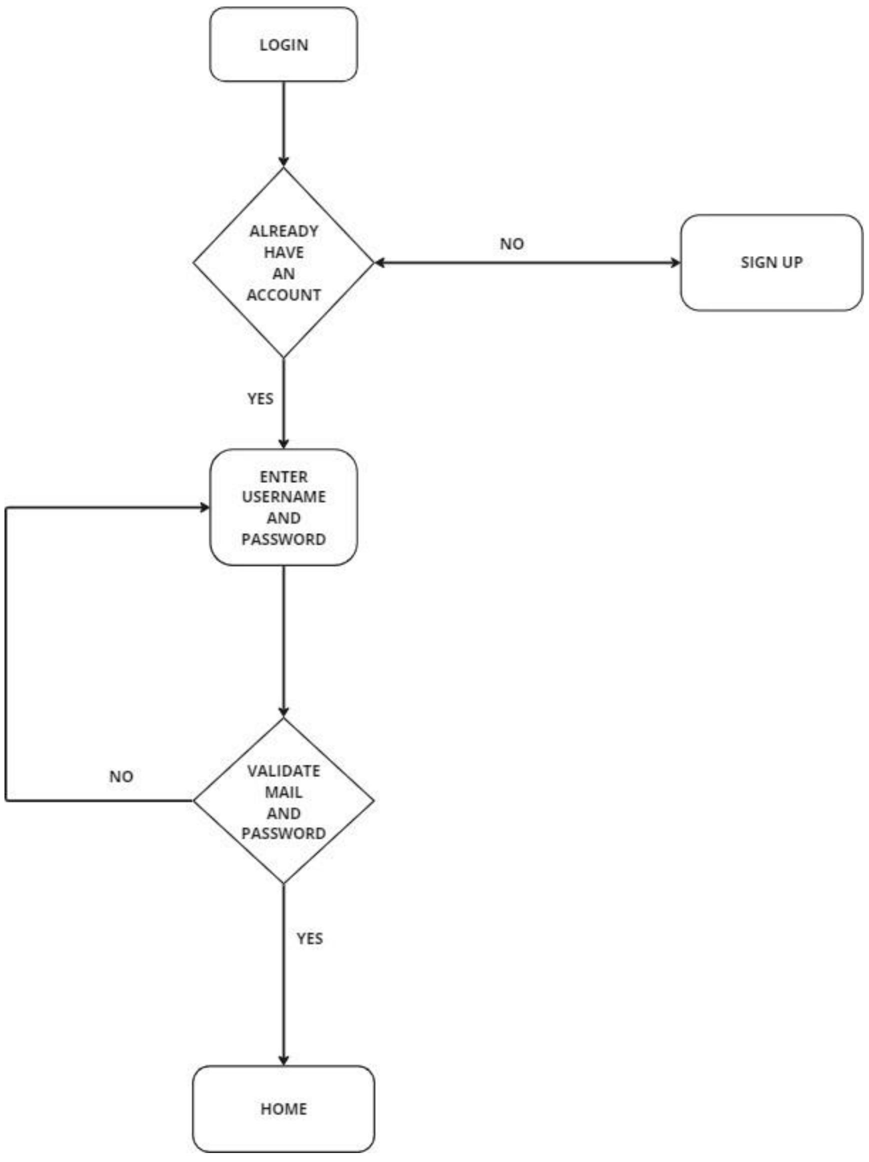


Fig 5.2 LOGIN PAGE FLOWCHART

5.2 SIGN-IN

This page asks users about the basic details of the user to create an account. This page asks for details like username, password, email id. After the user enters the details, these details are then validated by ourcode . If all details are correct then the user is then directed to the login page.

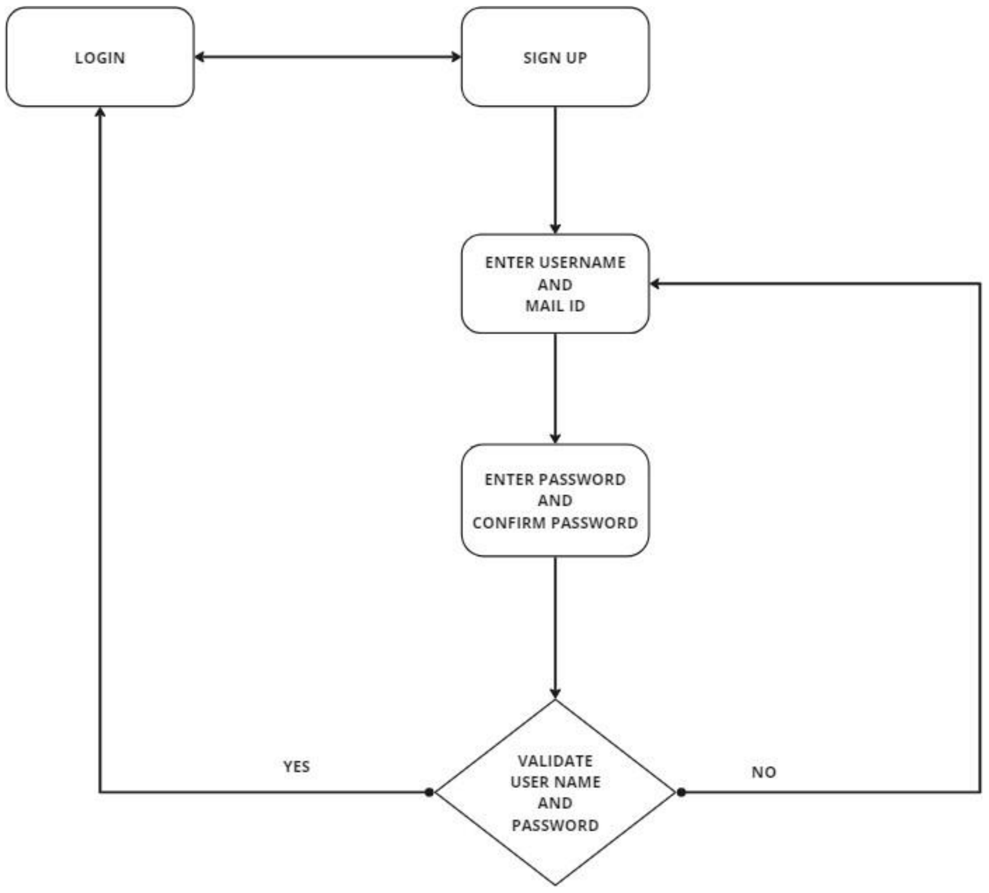

LOGIN
SIGN UP
ENTER USERNAME AND MAIL ID
ENTER PASSWORD AND CONFIRM PASSWORD
VALIDATE USER NAME AND PASSWORD
YES
NO

Fig: 5.3 SIGNUP PAGE FLOWCHART

CHAPTER 6

IMPLEMENTATION AND RESULTS

This chapter gives a description about the output that we produced by developing the website of our idea.

6.1 ANIME WORLD

When the user clicks the URL Link, It will be Re-Directed to the page Anime World, Where they can view all details about our application and There are links to Visit Login Page, Sign Up Page, Home Page.

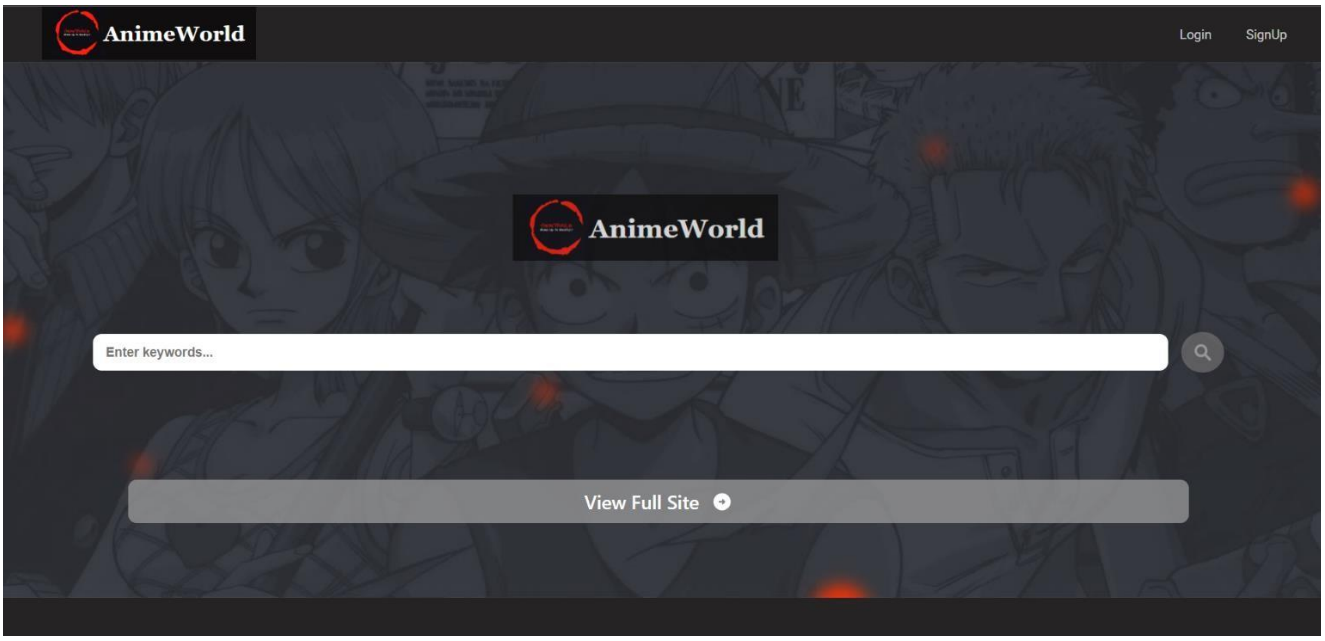


Fig: 6.1 ANIME WORLD PAGE

6.1.1 REACT CODE

```
import './defaultPage.css';
import logo2 from './logo.png'
import logo from './logo-name.png'
import search from './icons8-search.svg'
import { Link } from 'react-router-dom';
import linkArrow from './arrow_circle_right_white_24dp.svg'

export default function DefaultPage({ loginPage, setLoginPage, signUpPage,
setsignUpPage }) {



  return (
    <>
      <div className="defaultPage">
        <div className="dpage-home">
          <div className="dpage-home-body">
            <div className="dpage-home-link">
            <a href='www.google.com'>
            <img className='home-link-logo' src={logo}></img>
            </a>
            </div>

            <div className="dpage-input-div">
              <input className='dpage-input-box' type="text" placeholder='Enter
keywords...'/>
              <div className="dpage-search-div">
                <img src={search} className='dpage-search-icon' />
              </div>
            </div>
              <Link to="/home" className='dpge-home-vlink-a'>
                <div className="dpge-home-vlink">
                  View Full Site
                  <img src={linkArrow} className='dpge-home-vlink-icon'/>
                </div>
              </Link>

          </div>

          <div className='dpage-container'>
            <div className="dpage-details">
              <h1 className="dpage-details-h1">
              Anime World - The best site to watch anime online for Free
              </h1>

              <p className="dpage-details-p">
              Do you know that according to Google, the monthly search volume for anime related topics is up to over 1 Billion times? Anime is famous worldwide and it is no wonder we've seen a sharp rise in the number of free anime streaming sites.
              </p>

              <p className="dpage-details-p">
              Just like free online movie streaming sites, anime watching sites are not created equally, some are better than the rest, so we've decided to build AnimeWorld to be one of the best free anime streaming site for all anime fans on the world.
                </p>

                <h1 className="dpage-details-h1">
                1/ What is Anime World?
              </h1>
```

```
      <p className="dpage-details-p">
            AnimeWorld is a free site to watch anime and you can even download subbed or dubbed anime in ultra HD quality without any registration or payment. By having No Ads in all kinds, we are trying to make it the safest site for free anime.
             </p>

            <h1 className="dpage-details-h1">
            2/ Is AnimeWorld safe?
            </h1>

            <p className="dpage-details-p">
            Yes we are, we do have only one Ads to cover the server cost and we keep scanning the ads 24/7 to make sure all are clean, If you find any ads that is suspicious, please forward us the info and we will remove it.
             </p>

            <h1 className="dpage-details-h1">
            3/ So what make AnimeWorld the best site to watch anime free online?
            </h1>

            <ul>
              <li className="dpage-details-li">
                <strong>Safety:</strong>
                We try our best to not having harmful ads on AnimeWorld.
              </li>
              <li className="dpage-details-li">
                <strong>Content library:</strong>
                Our main focus is anime. You can find here popular, classic, as well as current titles from all genres such as action, drama, kids, fantasy, horror, mystery, police, romance, school, comedy, music, game and many more. All these titles come with English subtitles or are dubbed in many languages.
```

```
    </li>
              <li className="dpage-details-li">
     <strong>Quality/Resolution:</strong>
    All titles are in excellent resolution, the best quality possible. AnimeWorld also has
a quality setting function to make sure our users can enjoy streaming no matter how fast
your Internet speed is. You can stream the anime at 360p if your Internet is being
ridiculous, Or if it is good, you can go with 720p or even 1080p anime.
              </li>
              <li className="dpage-details-li">
                <strong>Updates: </strong>
                We updates new titles as well as fulfill the requests on a daily basis so be
warned, you will never run out of what to watch on AnimeWorld.
              </li>
              <li  className="dpage-details-li">
                <strong>User interface: </strong>
                Our UI and UX makes it easy for anyone, no matter how old you are,
how long have you been on the Internet. Literally, you can figure out how to navigate our
site after a quick look. If you want to watch a specific title, search for it via the search box.
If you want to look for suggestions, you can use the site's categories or simply scroll down
for new releases.
              </li>
              <li className="dpage-details-li">
                <strong>Device compatibility:</strong>
                AnimeWorld works alright on both your mobile and desktop.
              </li>

             </ul>
```

```
          <p className="dpage-details-p">
                    So if you're looking for a trustworthy and safe site for your Anime
streaming, let's give AnimeWorld a try. And if you like us, please help us to spread the
words and do not forget to bookmark our site.
          </p>
          <h1 className="dpage-details-h1">
                    4/ I can not access AnimeWorld website, what should I do?
                    </h1>
                    <p className="dpage-details-p">
                    Your ISP (Internet Service Provider) may have put AnimeWorld domain
into the block list, you can now access AnimeWorld website via our Proxy sites network,
which can be found here: https://AnimeWorldanime.net. We will keep adding new domain
to the list so please bookmark AnimeWorldAnime.net website to stay updated.
                     </p>

                     <p className="dpage-details-p">
                     Thank you!
                     </p>
                 </div>
               </div>
             </div>
           </div>
         </>
       )
     }
```

6.2 LOGIN

When User enters our website he will be asked about his login details like user-name and password. The login details will be verified with the details given while the user creates an account.

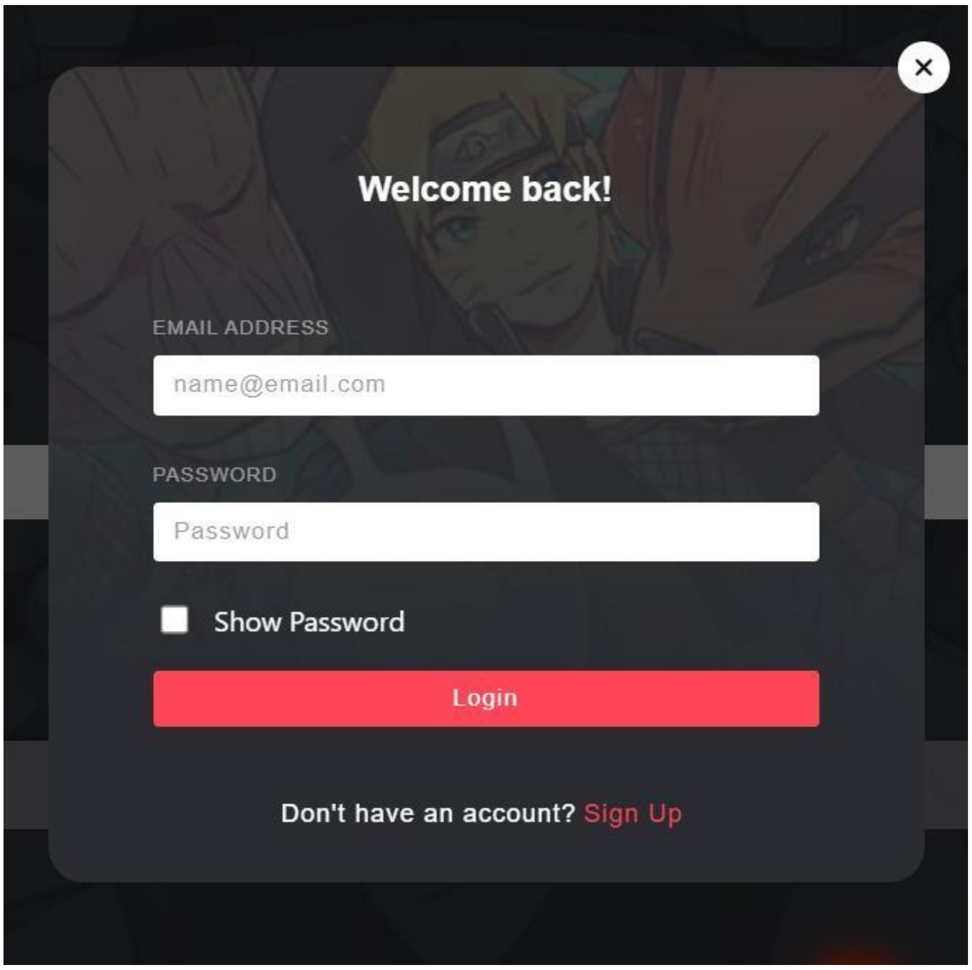


Fig: 6.2 LOGIN

6.2.1 REACT CODE

```
import './Login.css'

import React from 'react'

export default function Login({ setLoginPage, setsignUpPage }) {

    return (
        <>
            <div className={`login-container`}>
                <div className="login-tab">
                    <div onClick={() => setLoginPage(false) } className="login-close-button">
                        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 384 512">
                            <path d="M342.6 150.6c12.5-12.5 12.5-32.8 0-45.3s-32.8-12.5-45.3 0L192 
210.7 86.6 105.4c-12.5-12.5-32.8-12.5-45.3 0s-12.5 32.8 0 45.3L146.7 256 41.4 361.4c-12.5 
12.5-12.5 32.8 0 45.3s32.8 12.5 45.3 0L192 301.3 297.4 406.6c12.5 12.5 32.8 12.5 45.3 
0s12.5-32.8 0-45.3L237.3 256 342.6 150.6z"/>
                        </svg>
                    </div>
                    <h5 className="login-header-h">Welcome back!</h5>
                    <div className="login-content">
                        <form className='login-input-form' method='post' >
                            <div className="login-input-div">
                                <label htmlFor='email' className='login-input-label' >email 
address</label>
                                <input name='email' type="email" className="login-input" 
placeholder='name@email.com' />
                            </div>
                            <div className="login-input-div">
                                <label htmlFor='password' className='login-input-label' 
>password</label>
                                <input id='password' name='password' type="password" 
className="login-input" placeholder='Password' />
                            </div>
                            <div className="login-show-pswd">
                                <input onClick={ () => {
                                    const password = document.getElementById('password')
                                    password.type = (password.type === 'password') ? 'text' : 'password'
                                }} type="checkbox" name="show-password" id="show-pswd" />
                                <label htmlFor="show-password" onClick={ () => 
document.getElementById("show-pswd").click() } className="login-check-label">show 
password</label>
                            </div>
```

```
 <button onClick={ () => window.location.pathname = '/' } className='login-
button'>login</button>
                        </form>
                    </div>
                    <div className="login-redirect">
                        <span>Don't have an account?</span>
                        <a href="" onClick={ (e) => {
                            e.preventDefault()
                            setLoginPage(false)
                            setsignUpPage(true)
                        }} className="redirect">sign up</a>
                    </div>
                </div>
            </div>
        </>
    )
}
```

6.3 SIGNUP

If a user doesn't have an account on the website, User can use a component named create new account available in the login page. When the user clicks on that he will be redirected to the signup page. In sign up he should fill up his user-name, email id, password and confirm password. These inputs will be validated.

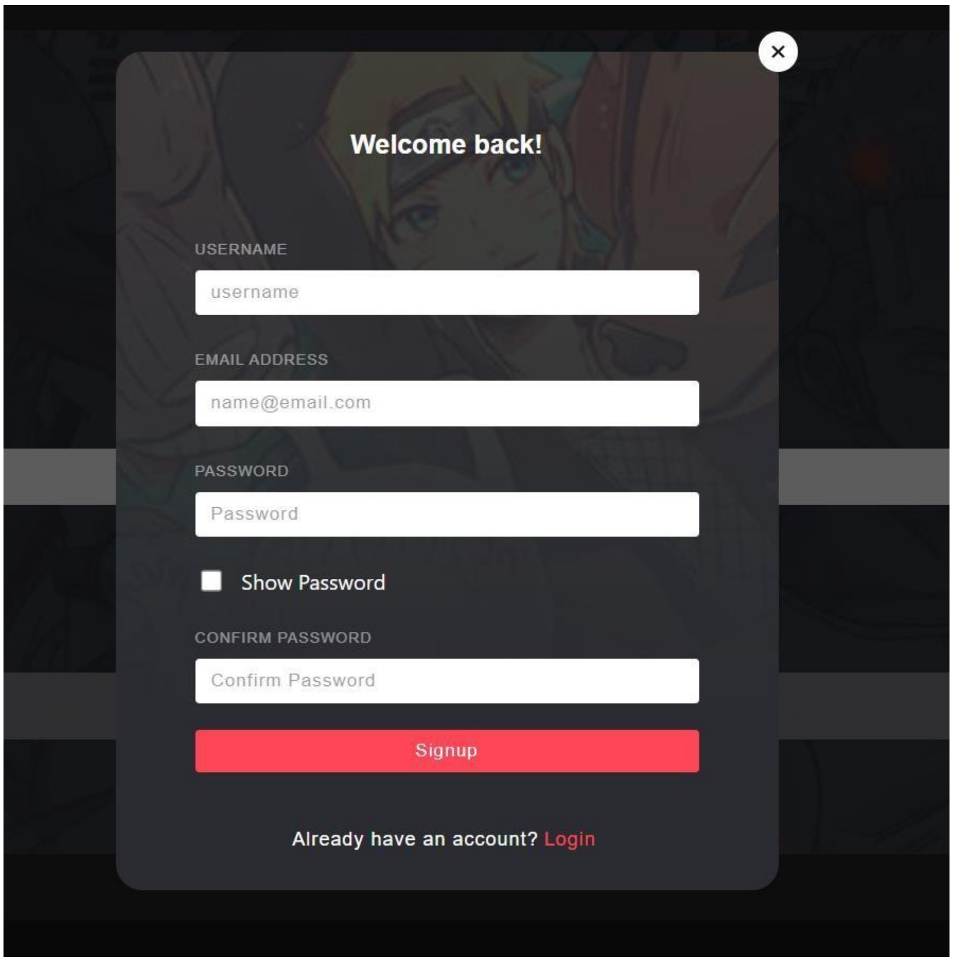


Fig: 6.3 SIGNUP

6.3.1 REACT CODE

```
import './SignUp.css'

export default function SignUp({ setsignUpPage,
setLoginPage }){
    return <>

        <div className="signup-container">
            <div className="signup-tab">
                <div onClick={() => setsignUpPage(false)}
className="signup-close-button">
                    <svg xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 384 512">
                        <path d="M342.6 150.6c12.5-12.5 12.5-32.8
0-45.3s-32.8-12.5-45.3 0L192 210.7 86.6 105.4c-12.5-12.5-
32.8-12.5-45.3 0s-12.5 32.8 0 45.3L146.7 256 41.4 361.4c-
12.5 12.5-12.5 32.8 0 45.3s32.8 12.5 45.3 0L192 301.3 297.4
406.6c12.5 12.5 32.8 12.5 45.3 0s12.5-32.8 0-45.3L237.3
256 342.6 150.6z"/>
                    </svg>
                </div>
                <h5 className="signup-header-h">Welcome
back!</h5>
                <div className="signup-content">
                    <form className='signup-input-form'
method='post' >
                        <div className="signup-input-div">
                            <label htmlFor='username'
className='signup-input-label' >username</label>
                            <input name='username' type="username"
className="signup-input" placeholder='username' />
                        </div>
                        <div className="signup-input-div">
                            <label htmlFor='email' className='signup-
input-label' >email address</label>
                            <input name='email' type="email"
className="signup-input" placeholder='name@email.com'
/>
```

```
</div>
                    <div className="signup-input-div">
                        <label htmlFor='password'
className='signup-input-label' >password</label>
                        <input id='password' name='password'
type="password" className="signup-input"
placeholder='Password' />
                    </div>
                    <div className="signup-show-pswd">
                        <input onClick={ () => {
                            const password =
document.getElementById('password')
                            password.type = (password.type ===
'password') ? 'text' : 'password'
                        }} type="checkbox" name="show-
password" id="show-pswd" />
                        <label htmlFor="show-password"
onClick={ () => document.getElementById("show-
pswd").click() } className="signup-check-label">show
password</label>
                    </div>
                    <div className="signup-input-div">
                        <label htmlFor='password'
className='signup-input-label' >confirm password</label>
                        <input name='password' type="password"
className="signup-input" placeholder='Confirm Password'
/>
                    </div>
                    <button onClick={(e) => {
                        e.preventDefault()
                        setsignUpPage(false)
                        setLoginPage(true)
                    }} className='signup-button'
type="submit">signup</button>
                </form>
```

```
                        </div>
                                <div className="signup-redirect">
                                    <span>Already have an account?</span>
                                    <a href="" onClick={(e) => {
                                        e.preventDefault()
                                        setsignUpPage(false)
                                        setLoginPage(true)
                                    }} className="redirect">login</a>
                                </div>
                            </div>
                        </div>
                    </>
}
```

6.4 NAV BAR

NavBar consist of our application logo , Login , sign up Profile. It plays an important role in our application.

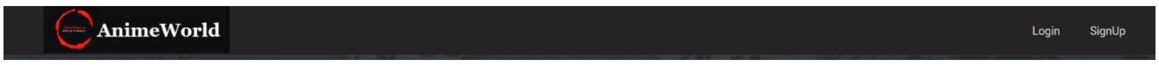


Fig: 6.4 NAVBAR

6.4.1 REACT CODE

```
import './NavBar.css'
import logo from './logo-name.png'
import React from 'react'

export default function NavBar({ setLoginPage,
setsignUpPage }) {

   const showLogin = (e) => {
      e.preventDefault()
      setLoginPage(true)
   }
   const showSignup = (e) => {
      e.preventDefault()
      setsignUpPage(true)
   }
return (
      <>
         <div className="dpage-home-head">
            <div className="home-head-logo">
               <a href='www.google.com'>
                  <img className='head-logo'
src={logo}></img>
               </a>
            </div>
```

```
                    <div className="home-head-links">
                              <a className='home-head-login'
                    onClick={showLogin} href=''>Login</a>
                              <a className='home-head-signup'
                    onClick={showSignup} href=''>SignUp</a>
                            </div>
                          </div>
                        </>
                      )
}
```

6.5 VIDEO PAGE

This is our applications main page , This plays Video of the anime that was Selected in home page and the side bar has the anime episodes that are currently available In than specific anime. The Video Player has all feature that other Streaming app has. You Can also download the episode and You can Increase the playback speed etc… NavBar consist of our application logo , Login , sign up and Profile.

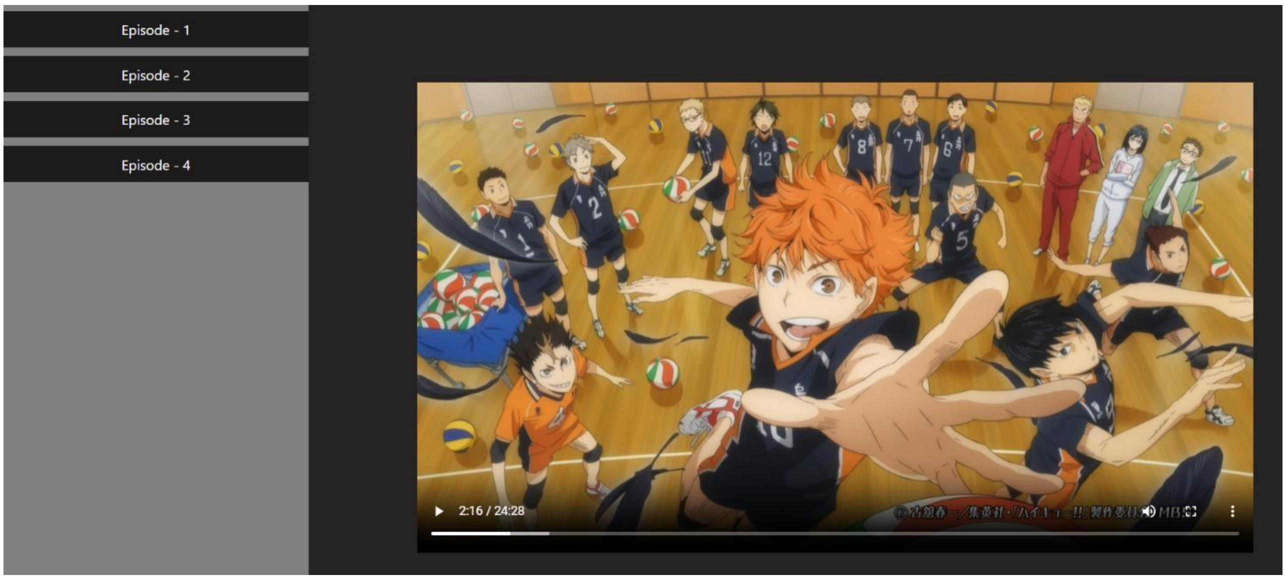


Fig: 6.5 VIDEO PLAYER

6.5.1 REACT CODE

```
import './videoPlayer.css'

import React from 'react'
import video from './videos/1 X 1.mkv'
import { useNavigate } from 'react-router-dom'

export default function VideoPlayer(props) {

    return (
        <>
            <div className="player-container">
                <div className="episode-container">
                    <div className="list-episodes">
                        <div className="episode" >Episode - 1</div>
                    </div>
                </div>
                <div className="video-player">
                    <video id='video' width="960" height="540" controls
autoPlay>
                        <source src={video} type="video/webm" />
                    </video>
                </div>
            </div>
        </>
    )
}
```

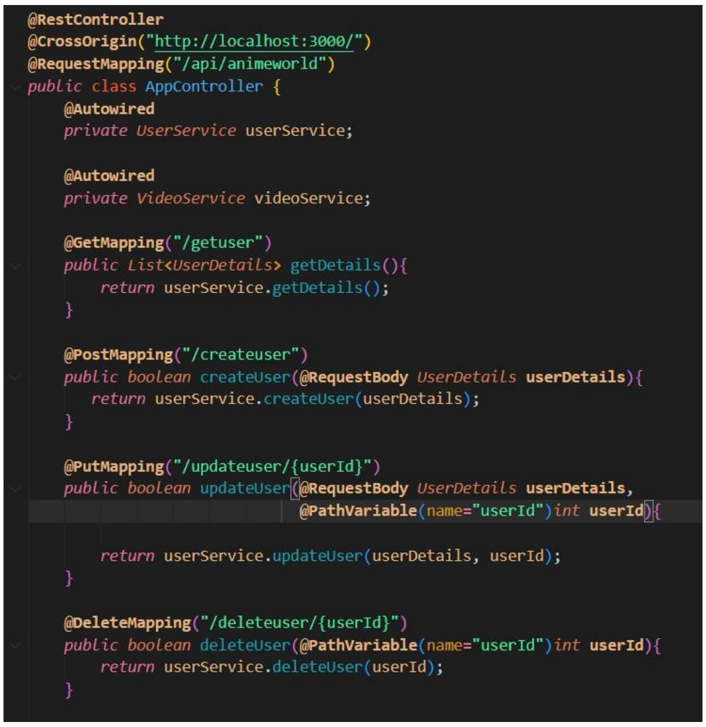
CHAPTER 7

BACKEND WITH SPRINGBOOT

7.1 CONTROLLER

In Spring Boot, a controller acts as a bridge between the user interface andthe application's business logic, handling incoming HTTP requests and returning appropriate responses. Controllers are a fundamental part of the MVC (Model-View-Controller) architecture, facilitating the separation of concerns and promoting clean code organization.A controller in Spring Boot is typically a Java class annotated with the @Controller or @RestController annotation. The @Controller annotation indicates thatthe class handles HTTP requests and returns views, while the @RestController annotation is specifically used for building RESTful APIs, where the returned data is serialized directly to the response body.Controllers define endpoints by using annotations such as @RequestMapping, @GetMapping, @PostMapping, @PutMapping, or @DeleteMapping. These annotations map the URLs and HTTP methodsto specific controller methods. For example, a method annotated with @GetMapping("/users") will handle GET requests to the "/users" URL path.Within a controller method, incoming request data can be accessed through method parameters. Spring Boot provides automatic data binding, allowing request parameters, path variables, and request bodies to bemapped to method parameters. This simplifies the extraction and processing of data from the request.Controllers can also make use of dependency injection to access other components, such as services or repositories, that handle business logic and data operations.

Through dependency injection, Spring Boot automatically manages the instantiation and wiring of these dependencies, ensuring loose coupling and modular design.Error handling is an essential aspect of controller development. Controllers can define exception handling methods using the @ExceptionHandler annotation or by creating custom exception handling logic. This allows for the graceful handling of exceptions that may occur during request processing, enabling the generation of appropriate error responses.Overall, controllers in Spring Boot provide a structured approach to handling HTTP requests and form a vital part of building web applications and RESTful APIs. They enable the separation of concerns, facilitate data binding, support dependency injection, and offer error handling capabilities, contributing to the robustness and maintainability of Spring Boot application.

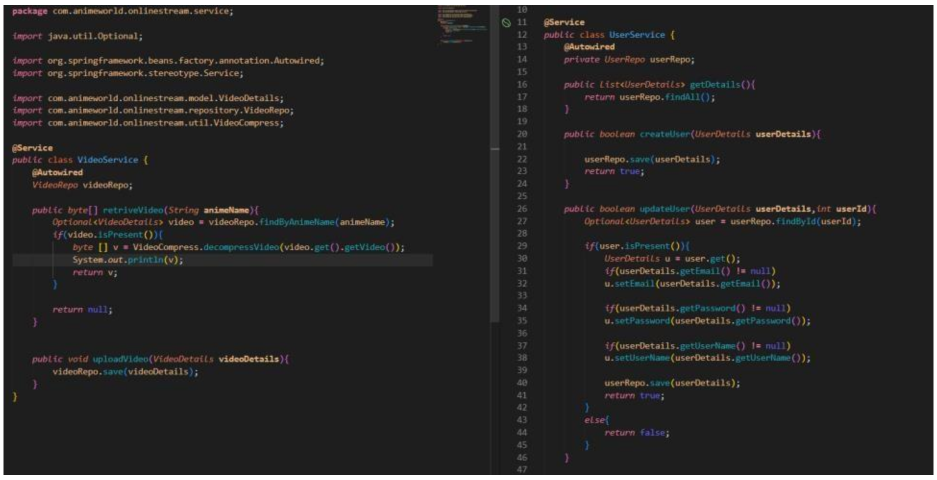


```java
@RestController
@CrossOrigin("http://localhost:3000/")
@RequestMapping("/api/animeworld")
public class AppController {
    @Autowired
    private UserService userService;

    @Autowired
    private VideoService videoService;

    @GetMapping("/getuser")
    public List<UserDetails> getDetails(){
        return userService.getDetails();
    }

    @PostMapping("/createuser")
    public boolean createUser(@RequestBody UserDetails userDetails){
       return userService.createUser(userDetails);
    }

    @PutMapping("/updateuser/{userId}")
    public boolean updateUser(@RequestBody UserDetails userDetails,
                              @PathVariable(name="userId")int userId){

        return userService.updateUser(userDetails, userId);
    }

    @DeleteMapping("/deleteuser/{userId}")
    public boolean deleteUser(@PathVariable(name="userId")int userId){
        return userService.deleteUser(userId);
    }
```

Fig 7.1 Controller Code

7.2 SERVICE

In Spring Boot, a service plays a crucial role in implementing the business logic and performing operations that are not directly related to handling HTTP requests. Services encapsulate the application's core functionalities, providing a modular and reusable layer between controllers and repositories. Here are some key points about services in Spring Boot:Services are typically implemented as Java classes annotated with the @Service annotation, which marks them as candidates for dependency injection. This allows the services to be automatically discovered and instantiated by the Spring framework.Services handlecomplex operations, such as data manipulation, validation, and interaction with repositories or external systems. They encapsulate the business logic and ensure that controllers remain lightweight and focused on request handling.Service methods are responsible for executing specific tasks, processing data, and coordinating interactions betweenvarious components of the application. They can be called by controllers or other services, promoting code reusability and modular design.Services often interact with repositories, which are responsible for persisting and retrieving data from databases or other data sources. By separating the concerns of business logic and data access, services enable better maintainability, testability, and flexibility in the application.



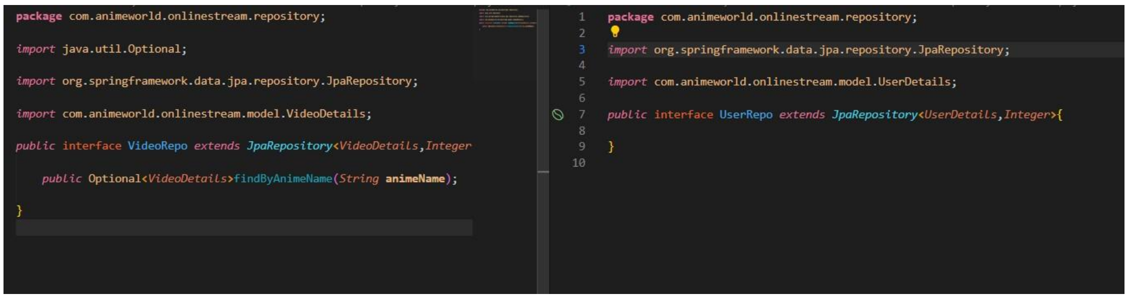
```java
package com.animeworld.onlinestream.service;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.animeworld.onlinestream.model.VideoDetails;
import com.animeworld.onlinestream.repository.VideoRepo;
import com.animeworld.onlinestream.util.VideoCompress;

@Service
public class VideoService {
    @Autowired
    VideoRepo videoRepo;

    public byte[] retriveVideo(String animeName){
        Optional<VideoDetails> video = videoRepo.findByAnimeName(animeName);
        if(video.isPresent()){
            byte [] v = VideoCompress.decompressVideo(video.get().getVideo());
            System.out.println(v);
            return v;
        }

        return null;
    }


    public void uploadVideo(VideoDetails videoDetails){
        videoRepo.save(videoDetails);
    }
}
```

```java

@Service
public class UserService {
    @Autowired
    private UserRepo userRepo;

    public List<UserDetails> getDetails(){
        return userRepo.findAll();
    }

    public boolean createUser(UserDetails userDetails){

        userRepo.save(userDetails);
        return true;
    }

    public boolean updateUser(UserDetails userDetails,int userId){
        Optional<UserDetails> user = userRepo.findById(userId);

        if(user.isPresent()){
            UserDetails u = user.get();
            if(userDetails.getEmail() != null)
            u.setEmail(userDetails.getEmail());

            if(userDetails.getPassword() != null)
            u.setPassword(userDetails.getPassword());

            if(userDetails.getUserName() != null)
            u.setUserName(userDetails.getUserName());

            userRepo.save(userDetails);
            return true;
        }
        else{
            return false;
        }
    }

```

7.2 Service code

7.3 REPOSITORY

In Spring Boot, a repository acts as an interface between the application and the underlying data storage system, typically a database. It provides a way to perform CRUD (Create, Read, Update, Delete) operations on the data, abstracting the complexities of data access and persistence. Here are some key points about repositories in Spring Boot:A repository is typically defined as an interface, annotated with the @Repository annotation, which marks it as a candidate for dependency injection and exception translation. Spring Boot provides various types of repositories, such as JpaRepository, CrudRepository, or MongoRepository, tailored to specific data storage systems.Repositories encapsulate data access logic, including querying, saving, updating, and deleting data. They define methods that correspond to specific data operations and provide an abstraction layer that shields the application from the underlying data storage details.Repositories leverage the power of Spring Data JPA, which simplifies database access by providing built-in implementation classes and automatically generating SQL queries based on method names. This reduces the amount of boilerplate code developers need to write, making data access more concise and efficient.By using repositories, developers can work with entities or domain objects instead of directly dealing with SQL statements or low-level database operations. Repositories automatically convert these objects into database- specific queries and handle the necessary data mapping, ensuring proper interaction with the data storage system.Repositories support various querying options, including method name queries, derived queries, or custom queries using the @Query annotation. These querying capabilities allow developers to retrieve data based on specific criteria, perform complex joins, or aggregate operations, ensuring flexibility in data retrieval.



```java
package com.animeworld.onlinestream.repository;

import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;

import com.animeworld.onlinestream.model.VideoDetails;

public interface VideoRepo extends JpaRepository<VideoDetails,Integer

    public Optional<VideoDetails>findByAnimeName(String animeName);

}
```

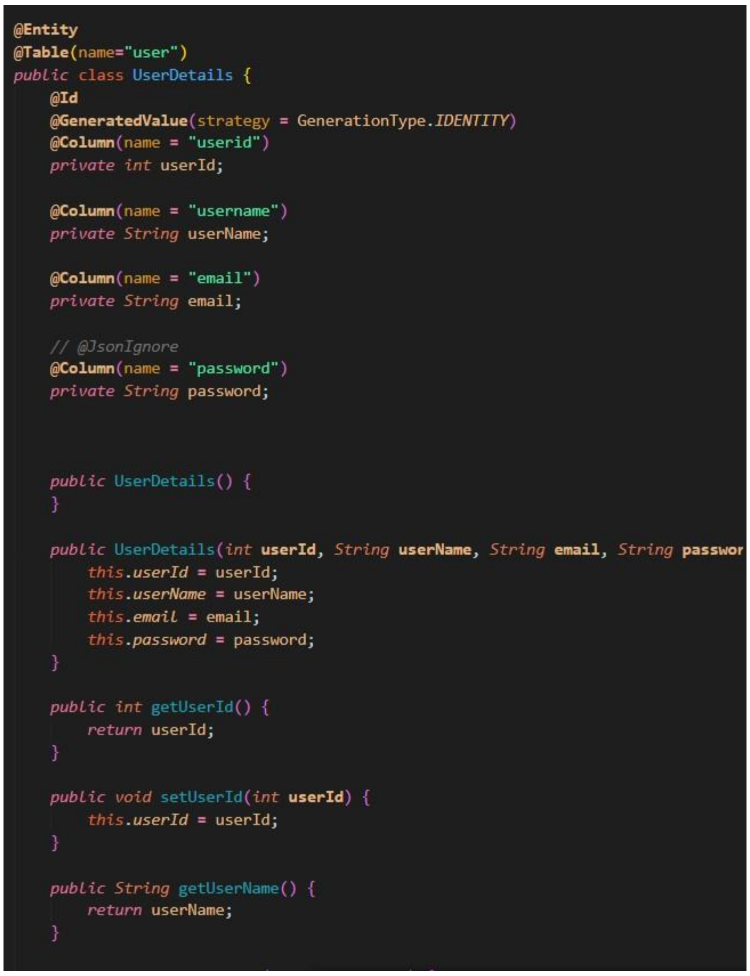
```java
package com.animeworld.onlinestream.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.animeworld.onlinestream.model.UserDetails;

public interface UserRepo extends JpaRepository<UserDetails,Integer>{

}
```

Fig 7.3 Repository Code

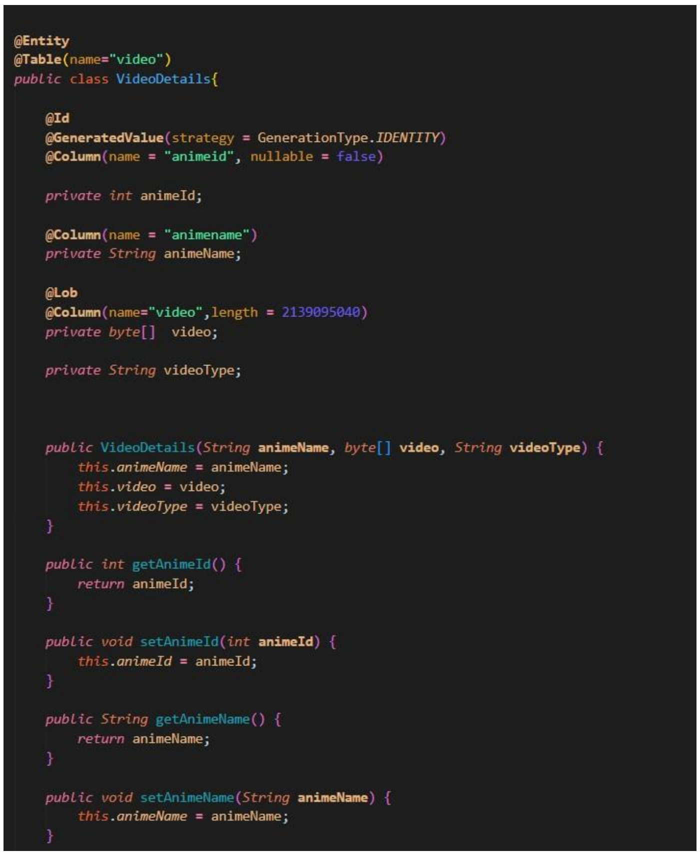
7.4 MODEL

In Spring Boot, a model represents the data and business entities of anapplication. It serves as a container for holding and manipulating data, encapsulating the application's state and providing a structured representation of the data entities. Here are some key points about models in Spring Boot:Models are typically implemented as POJOs (Plain Old Java Objects) that contain attributes or properties representing the data fields. These attributes can have different data types, such as strings, numbers, booleans, orcustom objects, depending on the requirements of the application.Modelsplay a vital role in implementing the domain or business logic of the application. They define the structure and behavior of the data entities, enabling developers to perform operations and enforce business rules on the data.Models are often annotated with additional metadata annotations, such as@Entity, @Table, or @Column, when using object-relational mapping (ORM) frameworks like Hibernate. These annotations provide mapping information, allowing the models to be persistently stored and retrieved from a database.Models can also define relationships between different entities, such as one-to-one, one-to-many, or many-to-many relationships. These relationships are typically established using annotations like @OneToOne, @OneToMany, or @ManyToMany, enabling the mapping and navigation between related data entities.Models can incorporate validation rules using annotations like @NotNull, @Size, or custom validation annotations.

These annotations help enforce data integrity and ensure that the data meets specific constraints or requirements.Models can be used as input or output objects in the application's controllers or services, allowing data to be transferred between different layers of the application. They serve as a standardized representation of the data, promoting consistency and clarity in communication.Models can also be used for data transformation or serialization purposes. Spring Boot provides support for converting models todifferent formats, such as JSON, XML, or HTML, allowing seamless integration with various presentation layers or APIs.



```
@Entity
@Table(name="user")
public class UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "userid")
    private int userId;

    @Column(name = "username")
    private String userName;

    @Column(name = "email")
    private String email;

    // @JsonIgnore
    @Column(name = "password")
    private String password;



    public UserDetails() {
    }

    public UserDetails(int userId, String userName, String email, String passwor
        this.userId = userId;
        this.userName = userName;
        this.email = email;
        this.password = password;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public String getUserName() {
        return userName;
    }
```

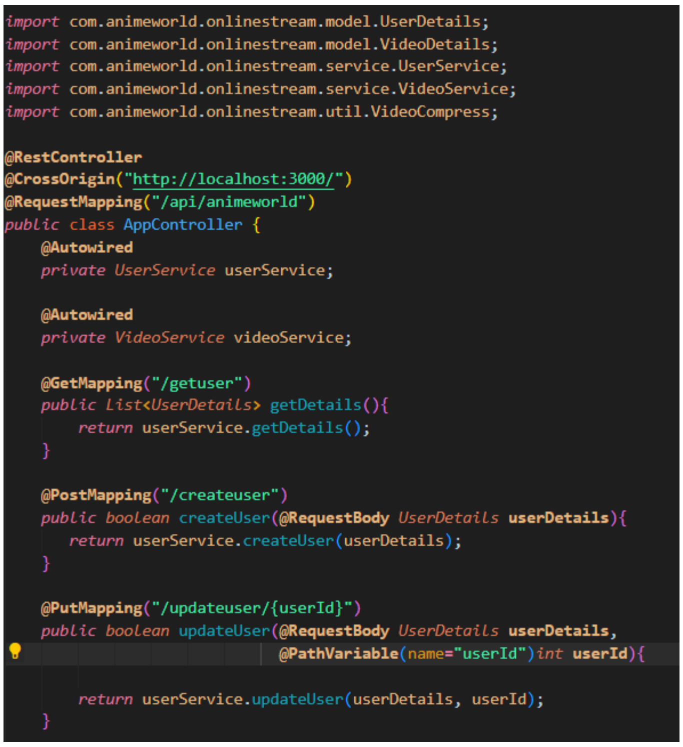


```java
@Entity
@Table(name="video")
public class VideoDetails{

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "animeid", nullable = false)

    private int animeId;

    @Column(name = "animename")
    private String animeName;

    @Lob
    @Column(name="video",length = 2139095040)
    private byte[]  video;

    private String videoType;



    public VideoDetails(String animeName, byte[] video, String videoType) {
        this.animeName = animeName;
        this.video = video;
        this.videoType = videoType;
    }

    public int getAnimeId() {
        return animeId;
    }

    public void setAnimeId(int animeId) {
        this.animeId = animeId;
    }

    public String getAnimeName() {
        return animeName;
    }

    public void setAnimeName(String animeName) {
        this.animeName = animeName;
    }
```

Fig 7.4 Model Codes

7.5 CONNECTION WITH FRONTEND (@CrossOrigin)

The @CrossOrigin annotation is used to enable Cross-Origin Resource Sharing (CORS) support for specific controller methods or the entire controller. CORS is a mechanism that allows web browsers to make requests to a different domain than the one serving the web page, which is essential for enabling cross-domain communication in web applications. Here are some key points about the @CrossOrigin annotation in Spring Boot:The @CrossOrigin annotation can be applied at the method or controller level. When applied at the method level, it overrides the CORS configuration specified at the controller level.By default, web browsers restrict cross-origin requests for security reasons. However, by adding the @CrossOrigin annotation, you can specify which origins are allowed to access the controller method. You can define specific origins, such as "http://example.com", or usewildcard characters like "*" to allow requests from any origin.The @CrossOrigin annotation allows you to configure additional CORS properties, such as allowed HTTP methods, headers, and whether to allow credentials (such as cookies or HTTP authentication) in the cross-origin requests. This flexibility allows you to fine-tune the CORS configuration based on your application's requirements.Enabling CORS with the @CrossOrigin annotation is crucial when building web applications that interact with APIs or services hosted on different domains. It helps prevent web browser security restrictions from blocking requests and ensures seamless communication between different domains. The @CrossOrigin annotation is particularly useful when developing Single Page Applications (SPAs) that consume APIs from separate backend servers or when building microservices architectures with multiple services running on different domains.It's important to note that enabling CORS should be done carefully and with consideration for security. You should only allow trusted originsand carefully configure allowed methods, headers, and credentials to prevent potential security vulnerabilities.



```java
import com.animeworld.onlinestream.model.UserDetails;
import com.animeworld.onlinestream.model.VideoDetails;
import com.animeworld.onlinestream.service.UserService;
import com.animeworld.onlinestream.service.VideoService;
import com.animeworld.onlinestream.util.VideoCompress;

@RestController
@CrossOrigin("http://localhost:3000/")
@RequestMapping("/api/animeworld")
public class AppController {
    @Autowired
    private UserService userService;

    @Autowired
    private VideoService videoService;

    @GetMapping("/getuser")
    public List<UserDetails> getDetails(){
        return userService.getDetails();
    }

    @PostMapping("/createuser")
    public boolean createUser(@RequestBody UserDetails userDetails){
       return userService.createUser(userDetails);
    }

    @PutMapping("/updateuser/{userId}")
    public boolean updateUser(@RequestBody UserDetails userDetails,
                              @PathVariable(name="userId")int userId){

        return userService.updateUser(userDetails, userId);
    }
```

Fig 7.5 @CrossOrigin Implementation

CHAPTER 8

CONCLUSION

In conclusion, the development and implementation of a streaming service app offer numerous benefits and opportunities for both users and businesses. The app provides a convenient and personalized platform for users to access a vast library of multimedia content, including movies, TV shows, and music, at their fingertips. With features like personalized recommendations, intuitive navigation, and high-quality streaming capabilities, the app enhances the user experience and keeps users engaged and entertained.

From a business perspective, a streaming service app opens up avenues for revenue generation through subscription models, advertising, or partnerships with content providers. The app allows businesses to reach a global audience, expand their user base, and capitalize on the growing demand for digital entertainment.

However, the success of a streaming app relies heavily on addressing certain challenges, such as licensing agreements, content availability, technical issues, and competition in the market. These challenges require continuous improvement, innovation, and adaptation to ensure the app remains competitive and meets the evolving needs of users.

In summary, a well-designed and user-centric streaming service app has the potential to revolutionize the way we consume and enjoy multimedia content. It provides convenience, personalization, and a seamless streaming experience, offering users a wide range of content options and keeping them engaged. With the right strategies and considerations, a streaming service app can become a valuable asset for businesses and a go-to platform for users seeking entertainment on-demand.

REFERENCES

Böhm, M., Körber, M., & Stohr, D. (2016). Streaming media systems: Challenges and directions. IEEE Transactions on Multimedia, 18(11), 2285-2290.

Mishra, R., & Verma, A. (2020). Design and implementation of online video streaming using adaptive algorithm. International Journal of Computer Science and Information Security, 18(1), 17-23.

Rana, R. R., & Pawar, P. S. (2020). Video streaming app using hybrid recommendation system. In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

Satapathy, M. K., Swain, A., & Khilar, P. M. (2021). A novel recommendation system for video streaming apps using hybrid content-based filtering. International Journal of Machine Learning and Networked Collaborative Engineering, 3(2), 44-52.

App References

SANJI.TO - https://sanji.to/

SERIES ONLINE - https://seriesonline.gg/