A PROJECT REPORT

ON

Video Streaming Site Streamify

Submitted by

Ishika Mangla, Ishika Goyal,

Navya Gupta, Mehak Sahni

2010991725 , 2010991724,

2110991770,2110991762

Supervised By

Ashwani Dubey

Department of Computer Science and Engineering

DECLARATION

We hereby declare that the project work titled, Streamify submitted as part of Bachelor's degree in CSE, at Chitkara University, Punjab, is an authentic record of our own work carried out under the supervision of Ashwani Dubey.

Signature(s):

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to everyone who contributed to the successful completion of our project, "Streamify," undertaken as part of our academic curriculum at Chitkara University, Rajpura. This endeavor would not have been possible without the support, guidance, and encouragement of numerous individuals and organizations.

First and foremost, we extend our heartfelt thanks to our project supervisor Mr.Ashwani Dubey, for his invaluable guidance, constructive feedback, and unwavering support throughout the development of Food Junction. His expertise and insights greatly contributed to the refinement and success of our project.

We are also grateful to the faculty members of the Department of Computer Science and Engineering for providing us with a conducive learning environment and the necessary resources to undertake this project.

Finally, we would like to express our gratitude to Chitkara University for providing us with the platform and resources to pursue our academic and research endeavors.

Thank you to everyone who played a role, directly or indirectly, in the completion of this project. Your support has been instrumental, and we appreciate the collaborative effort that has gone into making Streamify a reality.

INTRODUCTION

In the ever-evolving digital landscape, the convenience of video streaming has become an integral part of our daily lives. The Streamify website stands out as a beacon, offering a seamless and delightful experience for users seeking to satisfy their dopamine hits. As we present our project , it is imperative to delve into a comprehensive report that explores the various facets of the Streamify website.

Genesis of Streamify: Movies and series Streaming site

Streamify was conceived with a vision to redefine the way people interact with and indulge the trend of latest shows.. This amalgamation of cutting-edge technologies empowers Food Junction to deliver a user-friendly, responsive, and efficient platform that has a range of movies and Tv shows for users.

Login and SignUp using Firebase Authentication

One of the standout features of Streamify is its privacy protection. Users can effortlessly enter their login credentials or sign up and all the data will be stored in the firebase. The React.js frontend ensures that the login adapts dynamically.

Fetching List of shows and users from MongoDb

Streamify leverages the power of Node.js to orchestrate a seamless streaming process. Customers can like shows and add them to my list and delete as well. All the changes will be instantly reflected on Database as well, without my effort.

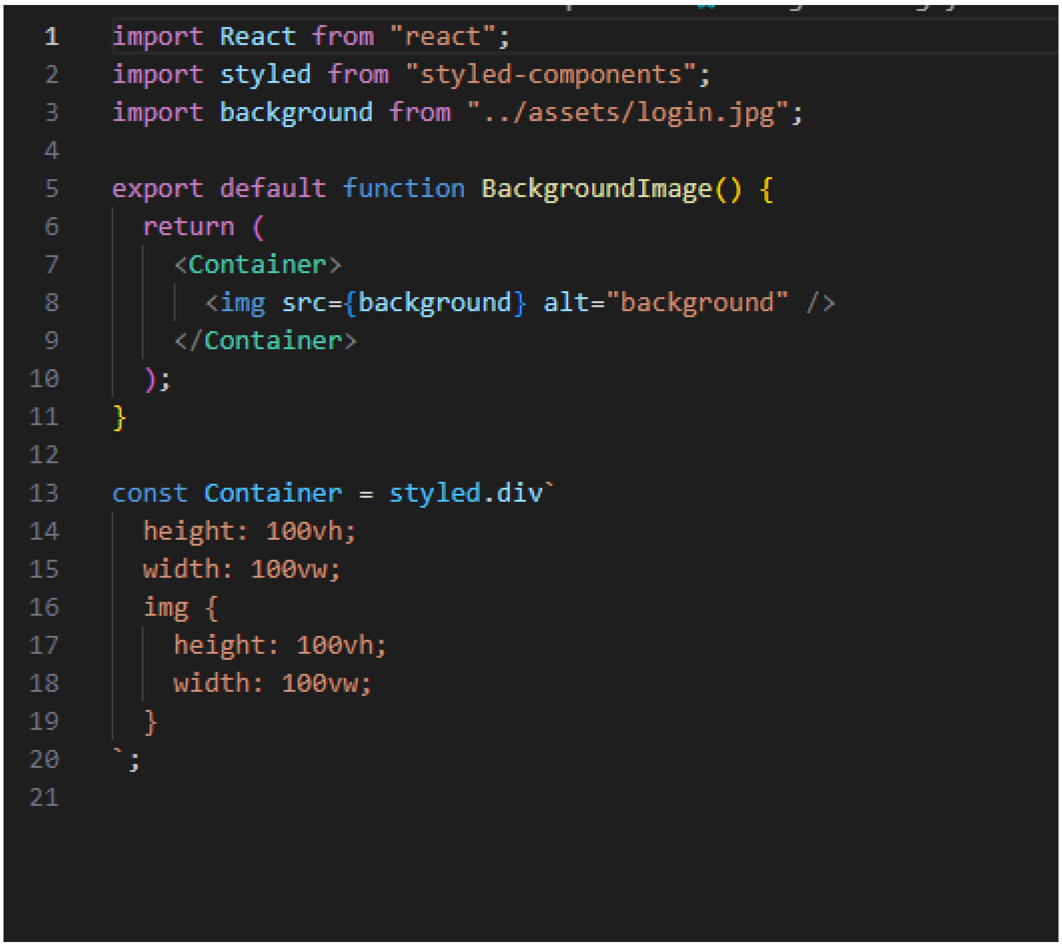
ABSTRACT

The "Streamify Website to strem your favourites movies and shows" report encapsulates a comprehensive examination of the technological marvel that is the Streamify online platform. This study delves into the genesis, evolution, and technological underpinnings of Food Junction, dissecting how it seamlessly integrates convenience. From its founding principles to the intricacies of its user-centric design, innovative features, and future technological roadmap, this report unveils the intricate tapestry that makes Streamify a leader in the intersection of gastronomy and technology. Through a meticulous exploration of its inception, architecture, and user experience, this abstract serves as a preview to the rich insights awaiting in the detailed report, showcasing how Streamify has crafted a digital haven.

Netflix UI Folder

Components used:

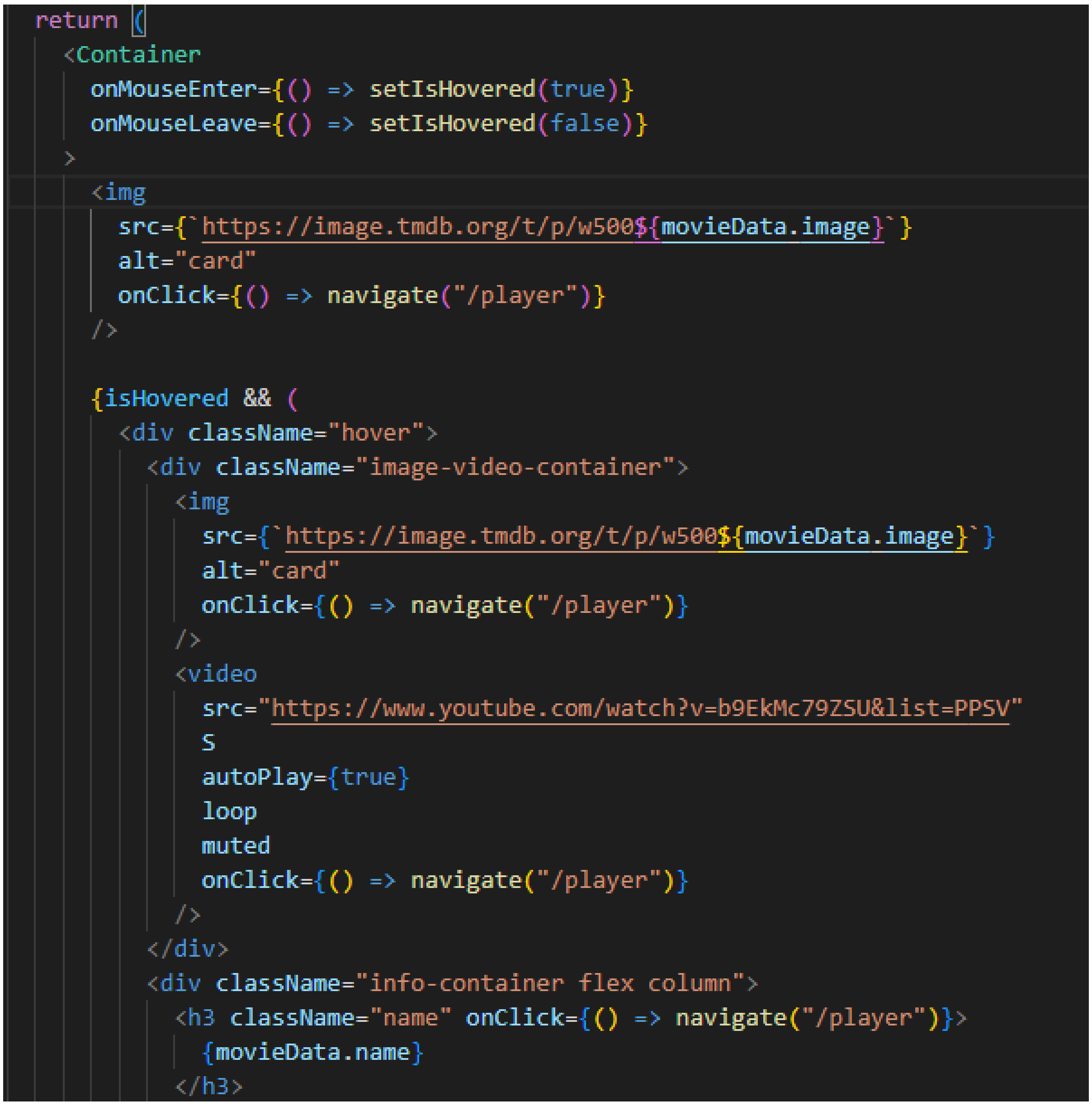
BackgroundImage.jsx



```jsx
import React from "react";
import styled from "styled-components";
import background from "../assets/login.jpg";

export default function BackgroundImage() {
  return (
    <Container>
      <img src={background} alt="background" />
    </Container>
  );
}

const Container = styled.div`
  height: 100vh;
  width: 100vw;
  img {
    height: 100vh;
    width: 100vw;
  }
`;
```

1. **This React code defines a functional component named** `BackgroundImage`**.**
2. **It uses the styled-components library to create a full-screen container (**`Container`**) with a background image.**
3. **The background image is sourced from "../assets/login.jpg".**
4. **The container has a height and width set to 100 viewport height and width.**
5. **The component exports the styled container and is intended for use as a background image in a React application.**

Card.jsx

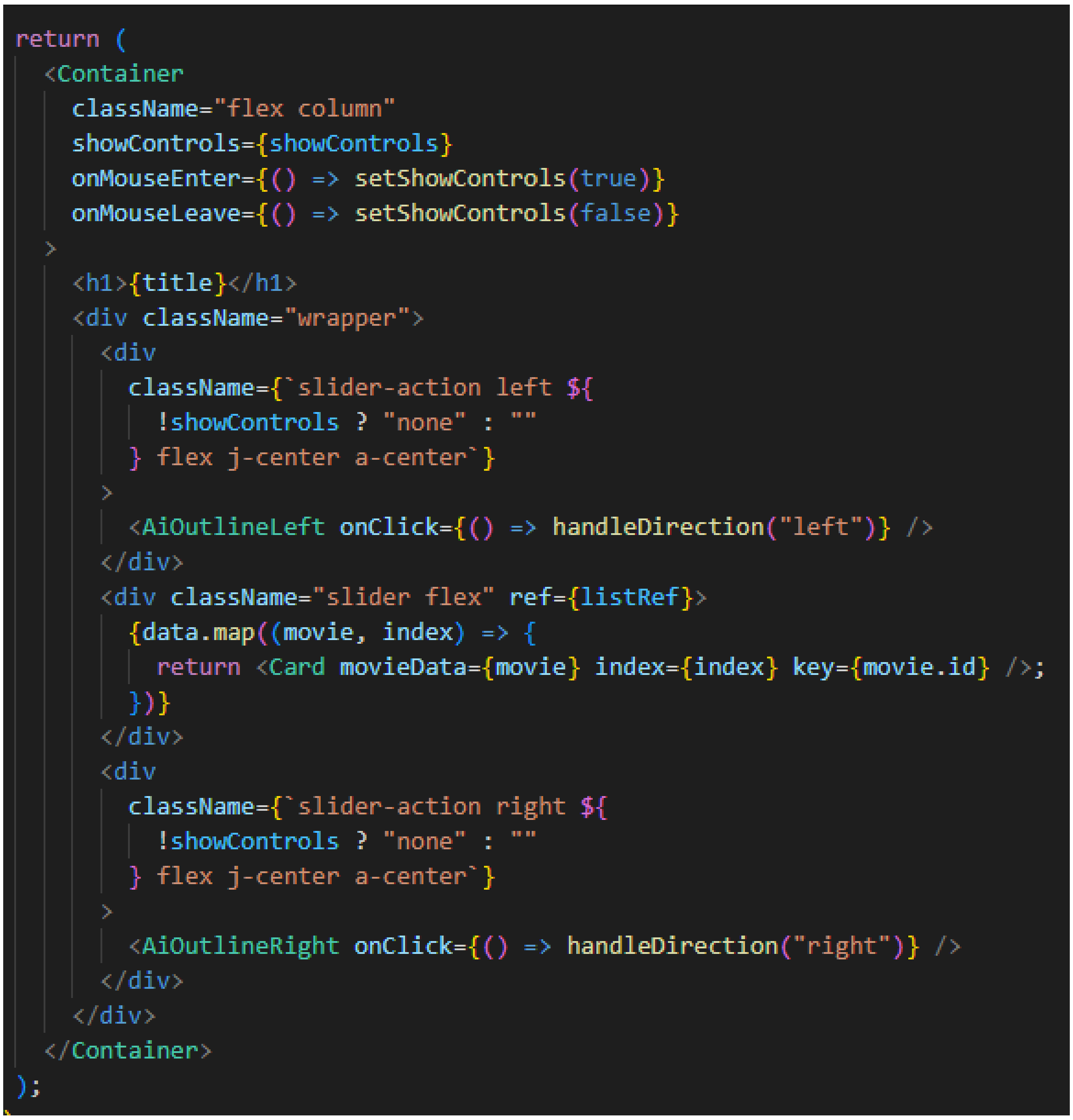


```
return (
  <Container
    onMouseEnter={() => setIsHovered(true)}
    onMouseLeave={() => setIsHovered(false)}
  >
    <img
      src={`https://image.tmdb.org/t/p/w500${movieData.image}`}
      alt="card"
      onClick={() => navigate("/player")}
    />

    {isHovered && (
      <div className="hover">
        <div className="image-video-container">
          <img
            src={`https://image.tmdb.org/t/p/w500${movieData.image}`}
            alt="card"
            onClick={() => navigate("/player")}
          />
          <video
            src="https://www.youtube.com/watch?v=b9EkMc79ZSU&list=PPSV"
            S
            autoPlay={true}
            loop
            muted
            onClick={() => navigate("/player")}
          />
        </div>
        <div className="info-container flex column">
          <h3 className="name" onClick={() => navigate("/player")}>
            {movieData.name}
          </h3>
```

1. **This React code defines a memoized functional component called `Card` that displays movie information.**
2. **It utilizes styled-components for styling and includes various icons from different libraries.**
3. **The component fetches and displays movie data, handles user interactions, and has an optional "like" feature.**
4. **On hovering over the card, additional information, controls, and genre details are displayed.**

5. **The styling is done using styled-components, creating a responsive and visually appealing card layout.**

CardSlider.jsx

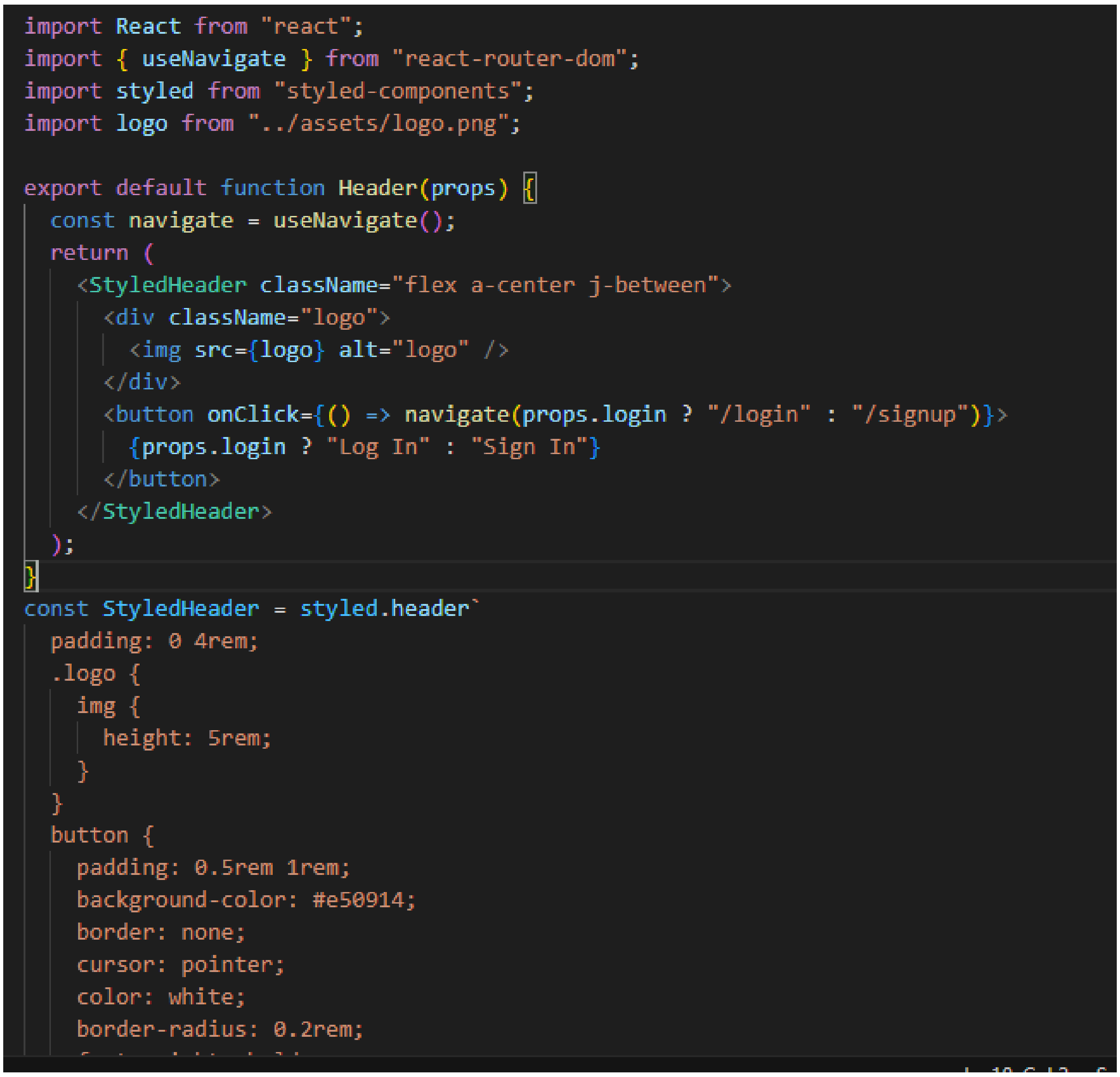


```
return (
  <Container
    className="flex column"
    showControls={showControls}
    onMouseEnter={() => setShowControls(true)}
    onMouseLeave={() => setShowControls(false)}
  >
    <h1>{title}</h1>
    <div className="wrapper">
      <div
        className={`slider-action left ${
          !showControls ? "none" : ""
        } flex j-center a-center`}
      >
        <AiOutlineLeft onClick={() => handleDirection("left")} />
      </div>
      <div className="slider flex" ref={listRef}>
        {data.map((movie, index) => {
          return <Card movieData={movie} index={index} key={movie.id} />;
        })}
      </div>
      <div
        className={`slider-action right ${
          !showControls ? "none" : ""
        } flex j-center a-center`}
      >
        <AiOutlineRight onClick={() => handleDirection("right")} />
      </div>
    </div>
  </Container>
);
```

1. The code defines a memoized React component called `CardSlider` for displaying a slider of movie cards with left and right navigation controls.
2. It uses `listRef` and state variables (`sliderPosition` and `showControls`) to handle slider movement and control visibility.
3. The `handleDirection` function adjusts the slider position based on the specified direction.

4. The slider displays movie cards using the `Card` component, and controls are shown on hover.
5. Styling is achieved using styled-components, providing a responsive and visually appealing slider layout with title and navigation controls.

Header.jsx

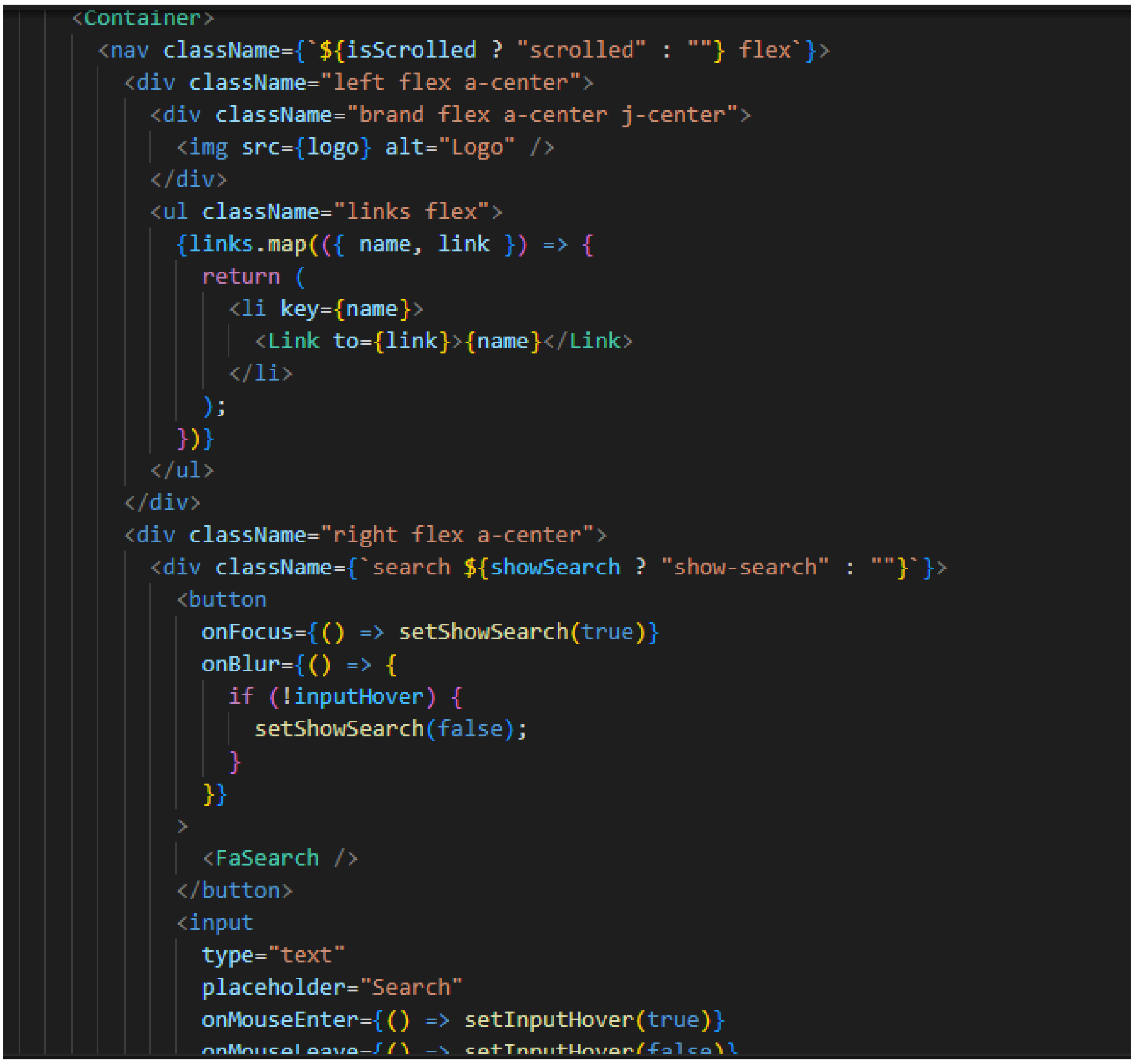


```
import React from "react";
import { useNavigate } from "react-router-dom";
import styled from "styled-components";
import logo from "../assets/logo.png";

export default function Header(props) {
  const navigate = useNavigate();
  return (
    <StyledHeader className="flex a-center j-between">
      <div className="logo">
        <img src={logo} alt="logo" />
      </div>
      <button onClick={() => navigate(props.login ? "/login" : "/signup")}>
        {props.login ? "Log In" : "Sign In"}
      </button>
    </StyledHeader>
  );
}
const StyledHeader = styled.header`
  padding: 0 4rem;
  .logo {
    img {
      height: 5rem;
    }
  }
  button {
    padding: 0.5rem 1rem;
    background-color: #e50914;
    border: none;
    cursor: pointer;
    color: white;
    border-radius: 0.2rem;
```

1. **The code defines a React functional component called `Header` for a navigation header.**
2. **It uses styled-components for styling and includes a logo and a button for login or signup.**

3. **The `useNavigate` hook from react-router-dom is employed to handle navigation.**
4. **The button's text and destination depend on the `login` prop passed to the component.**
5. **The styling includes padding, button appearance, and logo size adjustments using styled-components.**

Navbar.jsx

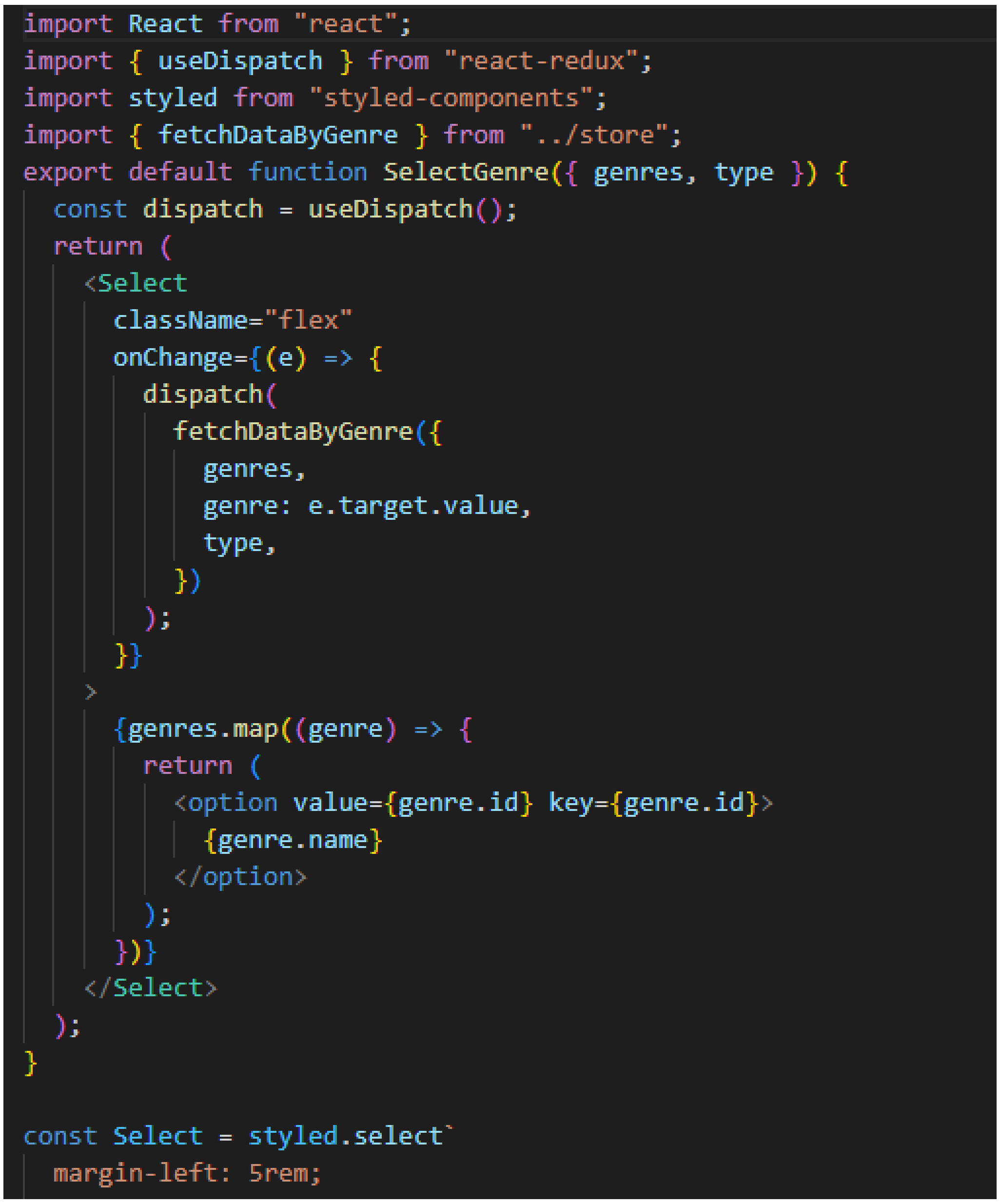


```jsx
<Container>
  <nav className={`${isScrolled ? "scrolled" : ""} flex`}>
    <div className="left flex a-center">
      <div className="brand flex a-center j-center">
        <img src={logo} alt="Logo" />
      </div>
      <ul className="links flex">
        {links.map(({ name, link }) => {
          return (
            <li key={name}>
              <Link to={link}>{name}</Link>
            </li>
          );
        })}
      </ul>
    </div>
    <div className="right flex a-center">
      <div className={`search ${showSearch ? "show-search" : ""}`}>
        <button
          onFocus={() => setShowSearch(true)}
          onBlur={() => {
            if (!inputHover) {
              setShowSearch(false);
            }
          }}
        >
          <FaSearch />
        </button>
        <input
          type="text"
          placeholder="Search"
          onMouseEnter={() => setInputHover(true)}
          onMouseLeave={() => setInputHover(false)}
```

1. **The code defines a React component named `Navbar` for a navigation bar with links, a search bar, and a sign-out button.**

2. **It uses styled-components for styling, and the navigation bar changes appearance on scroll (`isScrolled` prop).**
3. **The component includes a logo, navigation links, a search button, and a sign-out button powered by Firebase authentication.**
4. **The search bar expands on focus and contracts on blur, providing a visually appealing interaction.**
5. **Styling adjustments, such as color changes and transitions, are applied based on user interactions and scroll behavior.**

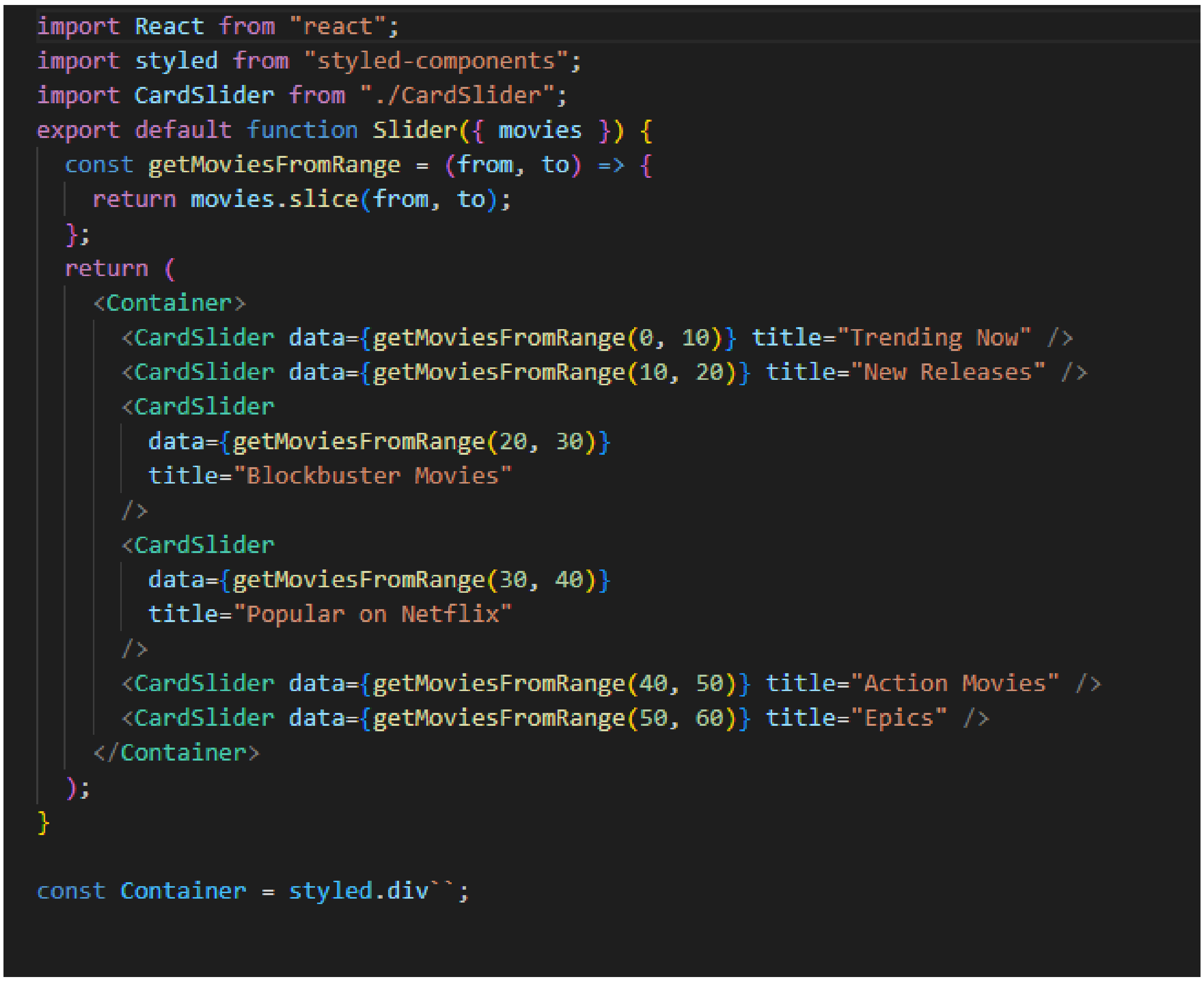
SelectGenre.jsx



```jsx
import React from "react";
import { useDispatch } from "react-redux";
import styled from "styled-components";
import { fetchDataByGenre } from "../store";
export default function SelectGenre({ genres, type }) {
  const dispatch = useDispatch();
  return (
    <Select
      className="flex"
      onChange={(e) => {
        dispatch(
          fetchDataByGenre({
            genres,
            genre: e.target.value,
            type,
          })
        );
      }}
    >
      {genres.map((genre) => {
        return (
          <option value={genre.id} key={genre.id}>
            {genre.name}
          </option>
        );
      })}
    </Select>
  );
}

const Select = styled.select`
  margin-left: 5rem;
```

1. The code defines a React component named `SelectGenre` for a dropdown menu to select movie genres.
2. It uses the `useDispatch` hook from react-redux to dispatch an action (`fetchDataByGenre`) based on the selected genre.
3. The component takes `genres` and `type` as props to populate the dropdown options and specify the data type.
4. The styled dropdown has a margin, cursor styling, and a semi-transparent background with white text.
5. On selection, the component dispatches an action with the chosen genre and type to fetch relevant data from the store.

Slider.jsx



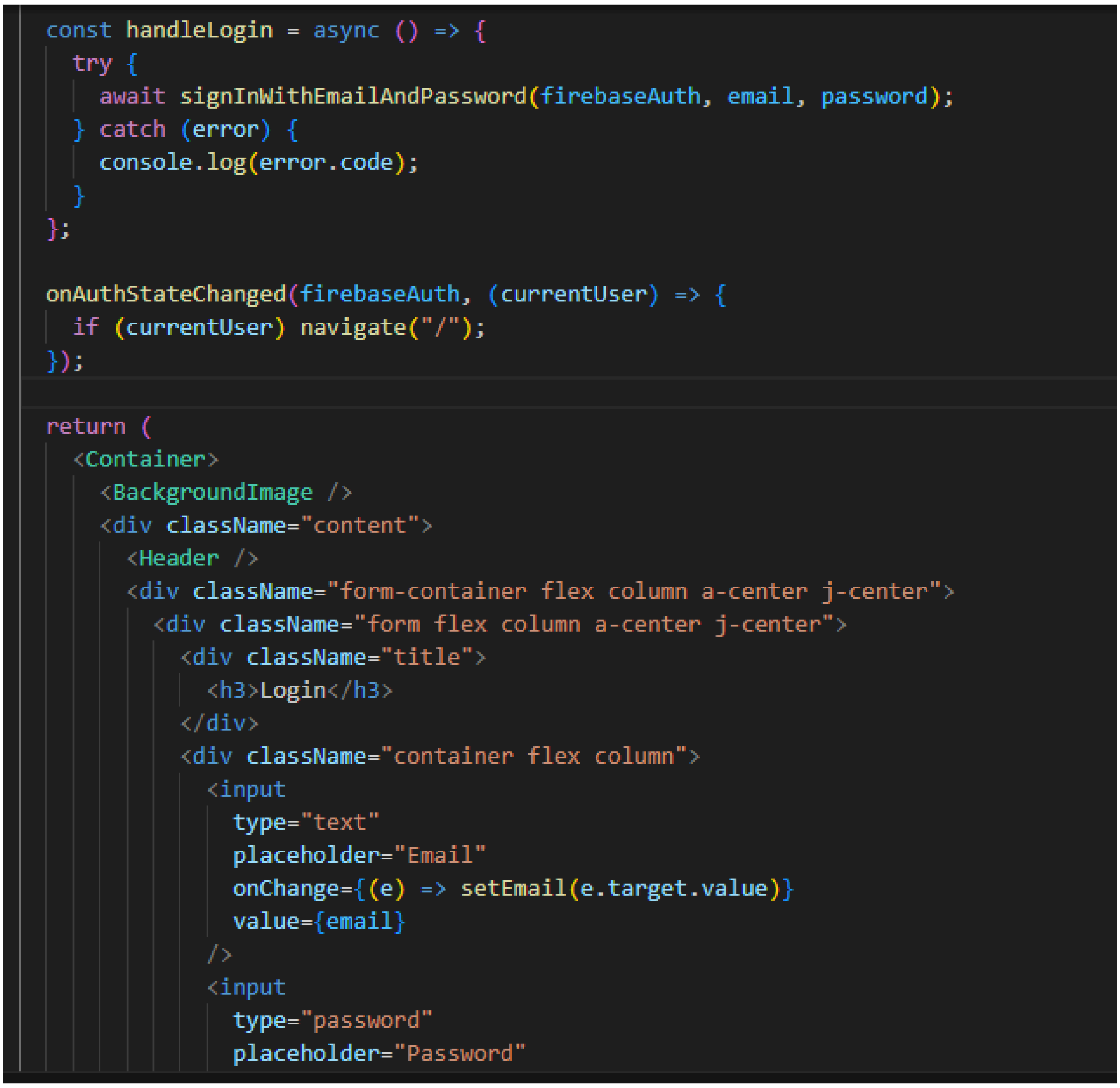
```
import React from "react";
import styled from "styled-components";
import CardSlider from "./CardSlider";
export default function Slider({ movies }) {
  const getMoviesFromRange = (from, to) => {
    return movies.slice(from, to);
  };
  return (
    <Container>
      <CardSlider data={getMoviesFromRange(0, 10)} title="Trending Now" />
      <CardSlider data={getMoviesFromRange(10, 20)} title="New Releases" />
      <CardSlider
        data={getMoviesFromRange(20, 30)}
        title="Blockbuster Movies"
      />
      <CardSlider
        data={getMoviesFromRange(30, 40)}
        title="Popular on Netflix"
      />
      <CardSlider data={getMoviesFromRange(40, 50)} title="Action Movies" />
      <CardSlider data={getMoviesFromRange(50, 60)} title="Epics" />
    </Container>
  );
}

const Container = styled.div``;
```

1. The code defines a React component named `Slider` that aggregates multiple `CardSlider` components for different movie categories.

2. **It utilizes the `getMoviesFromRange` function to extract a range of movies from the provided list.**
3. **The `CardSlider` components display movie cards with titles corresponding to different categories.**
4. **The component receives a prop (`movies`) containing the full list of movies.**
5. **Styled-components are used to create a container (`Container`) for the sliders.**

Pages used:

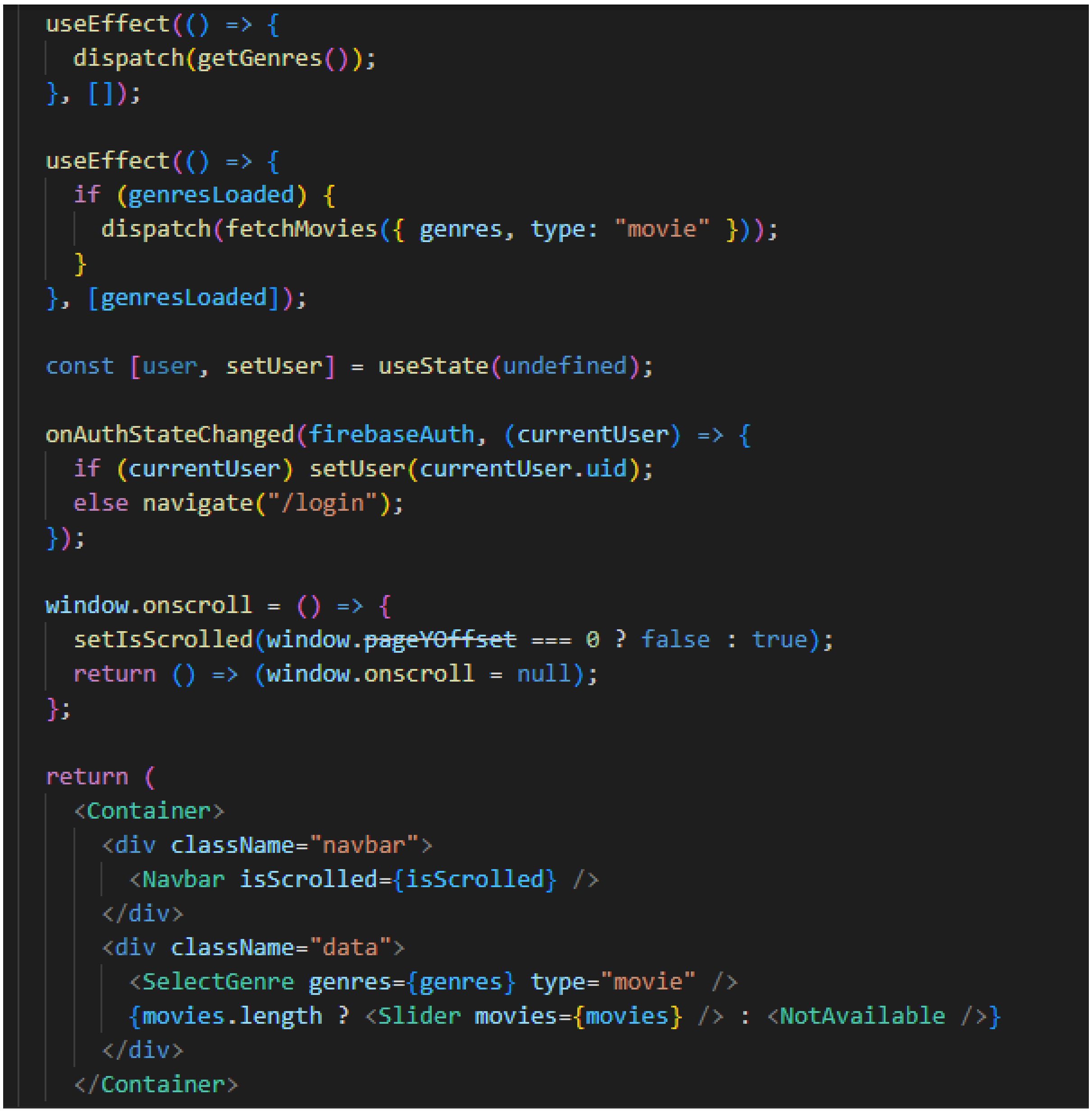
Login.jsx



```
const handleLogin = async () => {
  try {
    await signInWithEmailAndPassword(firebaseAuth, email, password);
  } catch (error) {
    console.log(error.code);
  }
};

onAuthStateChanged(firebaseAuth, (currentUser) => {
  if (currentUser) navigate("/");
});

return (
  <Container>
    <BackgroundImage />
    <div className="content">
      <Header />
      <div className="form-container flex column a-center j-center">
        <div className="form flex column a-center j-center">
          <div className="title">
            <h3>Login</h3>
          </div>
          <div className="container flex column">
            <input
              type="text"
              placeholder="Email"
              onChange={(e) => setEmail(e.target.value)}
              value={email}
            />
            <input
              type="password"
              placeholder="Password"
```

1. **The code defines a React component called `Login` for user authentication with email and password.**
2. **It includes a background image, a header, and a form for login using Firebase authentication.**
3. **State variables (`email` and `password`) manage user input, and the `handleLogin` function attempts to sign in.**
4. **The component redirects to the home page if a user is already authenticated.**
5. **Styled-components are used for layout, styling, and responsiveness, providing a visually appealing login interface.**

Movies.jsx

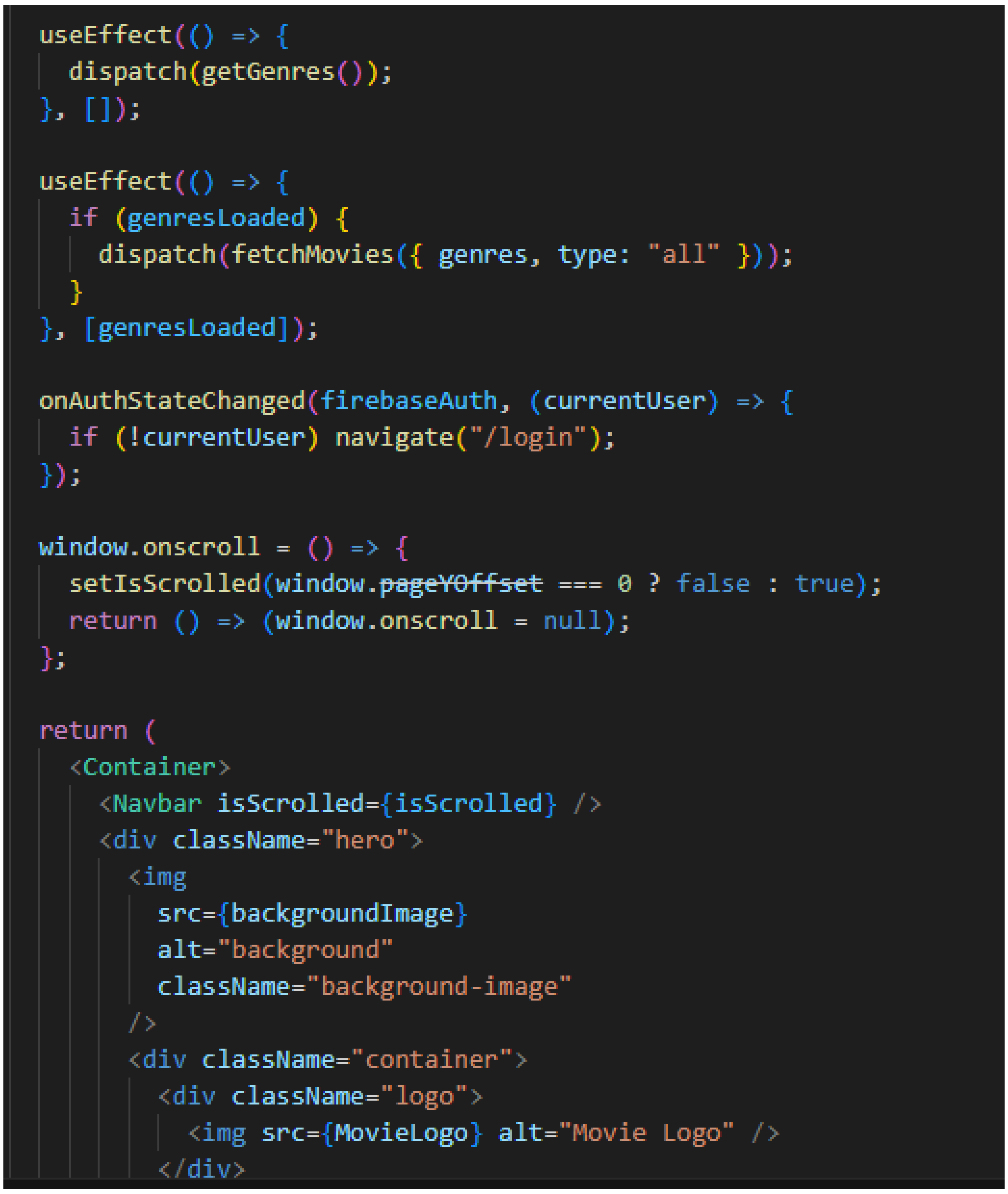


```
useEffect(() => {
  dispatch(getGenres());
}, []);

useEffect(() => {
  if (genresLoaded) {
    dispatch(fetchMovies({ genres, type: "movie" }));
  }
}, [genresLoaded]);

const [user, setUser] = useState(undefined);

onAuthStateChanged(firebaseAuth, (currentUser) => {
  if (currentUser) setUser(currentUser.uid);
  else navigate("/login");
});

window.onscroll = () => {
  setIsScrolled(window.~~pageYOffset~~ === 0 ? false : true);
  return () => (window.onscroll = null);
};

return (
  <Container>
    <div className="navbar">
      <Navbar isScrolled={isScrolled} />
    </div>
    <div className="data">
      <SelectGenre genres={genres} type="movie" />
      {movies.length ? <Slider movies={movies} /> : <NotAvailable />}
    </div>
  </Container>
```

1. **The code defines a React component named `MoviePage` for displaying movies with genre selection.**
2. **It includes a navigation bar (`Navbar`) and a movie slider (`Slider`) with genre selection options.**
3. **Redux is used to manage state, including movies, genres, and genre loading status.**
4. **The component fetches genres and movies on mount and adjusts the UI based on user authentication.**

5. **Styling is done using styled-components, and a "NotAvailable" component is displayed if no movies are present.**

Netflix.jsx

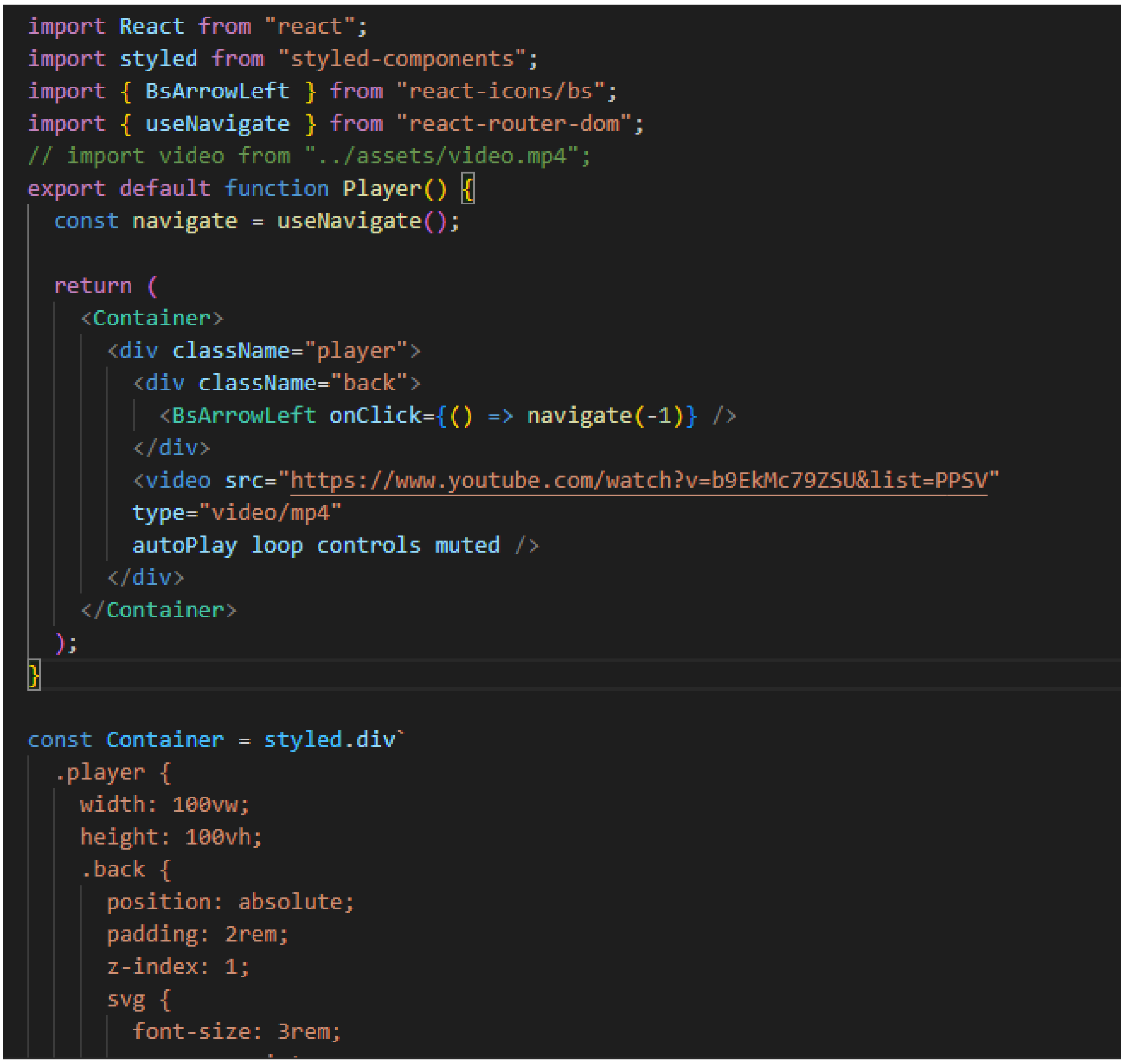


```
useEffect(() => {
  dispatch(getGenres());
}, []);

useEffect(() => {
  if (genresLoaded) {
    dispatch(fetchMovies({ genres, type: "all" }));
  }
}, [genresLoaded]);

onAuthStateChanged(firebaseAuth, (currentUser) => {
  if (!currentUser) navigate("/login");
});

window.onscroll = () => {
  setIsScrolled(window.~~pageYOffset~~ === 0 ? false : true);
  return () => (window.onscroll = null);
};

return (
  <Container>
    <Navbar isScrolled={isScrolled} />
    <div className="hero">
      <img
        src={backgroundImage}
        alt="background"
        className="background-image"
      />
      <div className="container">
        <div className="logo">
          <img src={MovieLogo} alt="Movie Logo" />
        </div>
```

1. **The code defines a React component named** `Netflix` **for the main page with a hero section and movie slider.**
2. **It includes a navigation bar (**`Navbar`**), fetches movies and genres using Redux, and adjusts UI based on user authentication.**

3. The hero section features a background image, a movie logo, and buttons for playing and getting more information.
4. Styling is achieved using styled-components, with dynamic changes based on scroll position and user interaction.
5. The component utilizes icons from `react-icons` and provides a visually engaging interface for navigating and streaming movies.

Player.jsx

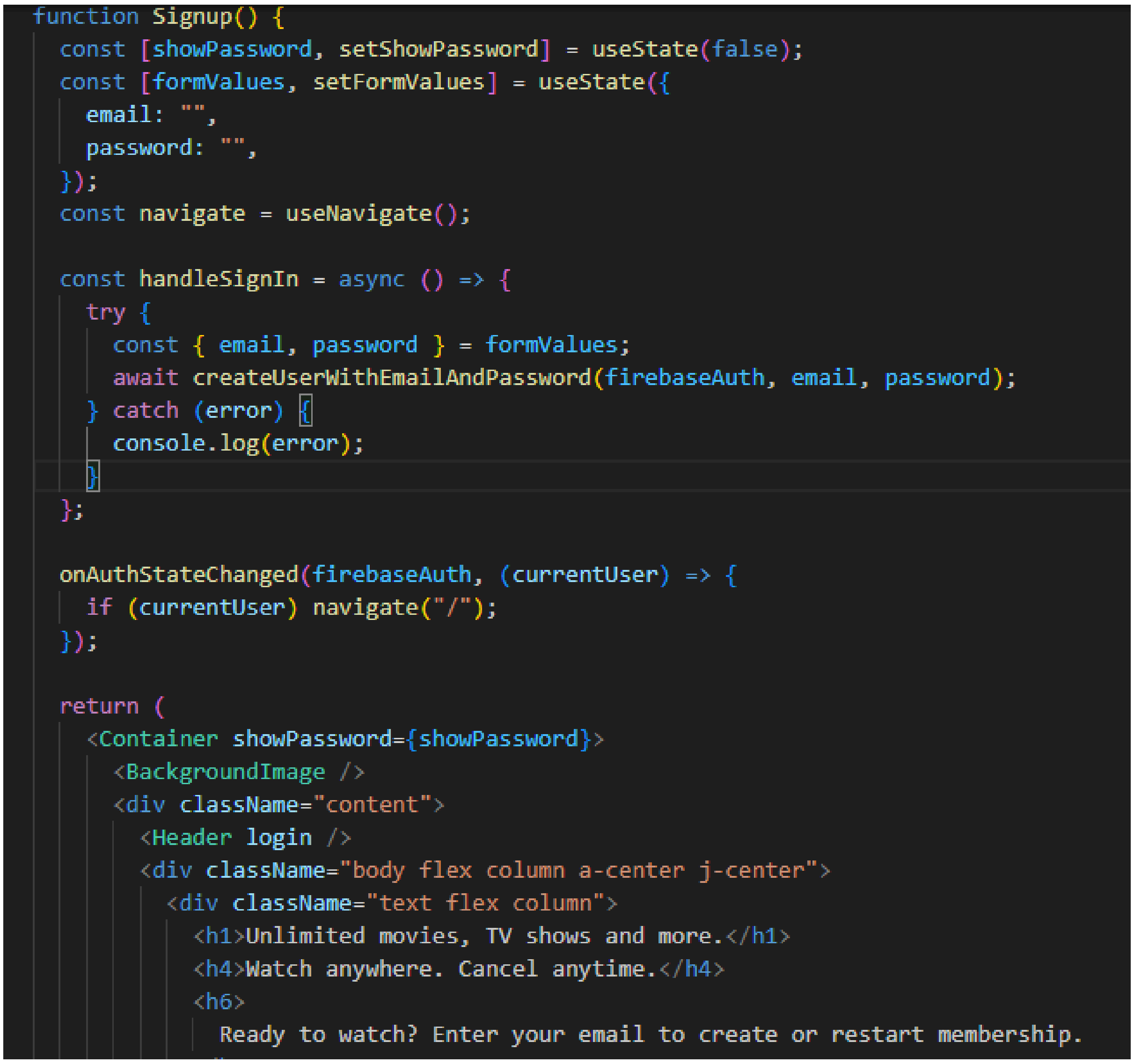


```jsx
import React from "react";
import styled from "styled-components";
import { BsArrowLeft } from "react-icons/bs";
import { useNavigate } from "react-router-dom";
// import video from "../assets/video.mp4";
export default function Player() {
  const navigate = useNavigate();

  return (
    <Container>
      <div className="player">
        <div className="back">
          <BsArrowLeft onClick={() => navigate(-1)} />
        </div>
        <video src="https://www.youtube.com/watch?v=b9EkMc79ZSU&list=PPSV"
        type="video/mp4"
        autoPlay loop controls muted />
      </div>
    </Container>
  );
}

const Container = styled.div`
  .player {
    width: 100vw;
    height: 100vh;
    .back {
      position: absolute;
      padding: 2rem;
      z-index: 1;
      svg {
        font-size: 3rem;
```

1. The code defines a React component named `Player` for video playback with a back button.

2. **It uses styled-components for styling and includes the** `BsArrowLeft` **icon for navigation.**
3. **The component employs the** `useNavigate` **hook from react-router-dom to handle navigation.**
4. **The video is sourced from a YouTube link, set to autoplay, loop, and muted, with playback controls.**
5. **Styling ensures the video fills the entire viewport, and a back button allows navigation to the previous page.**

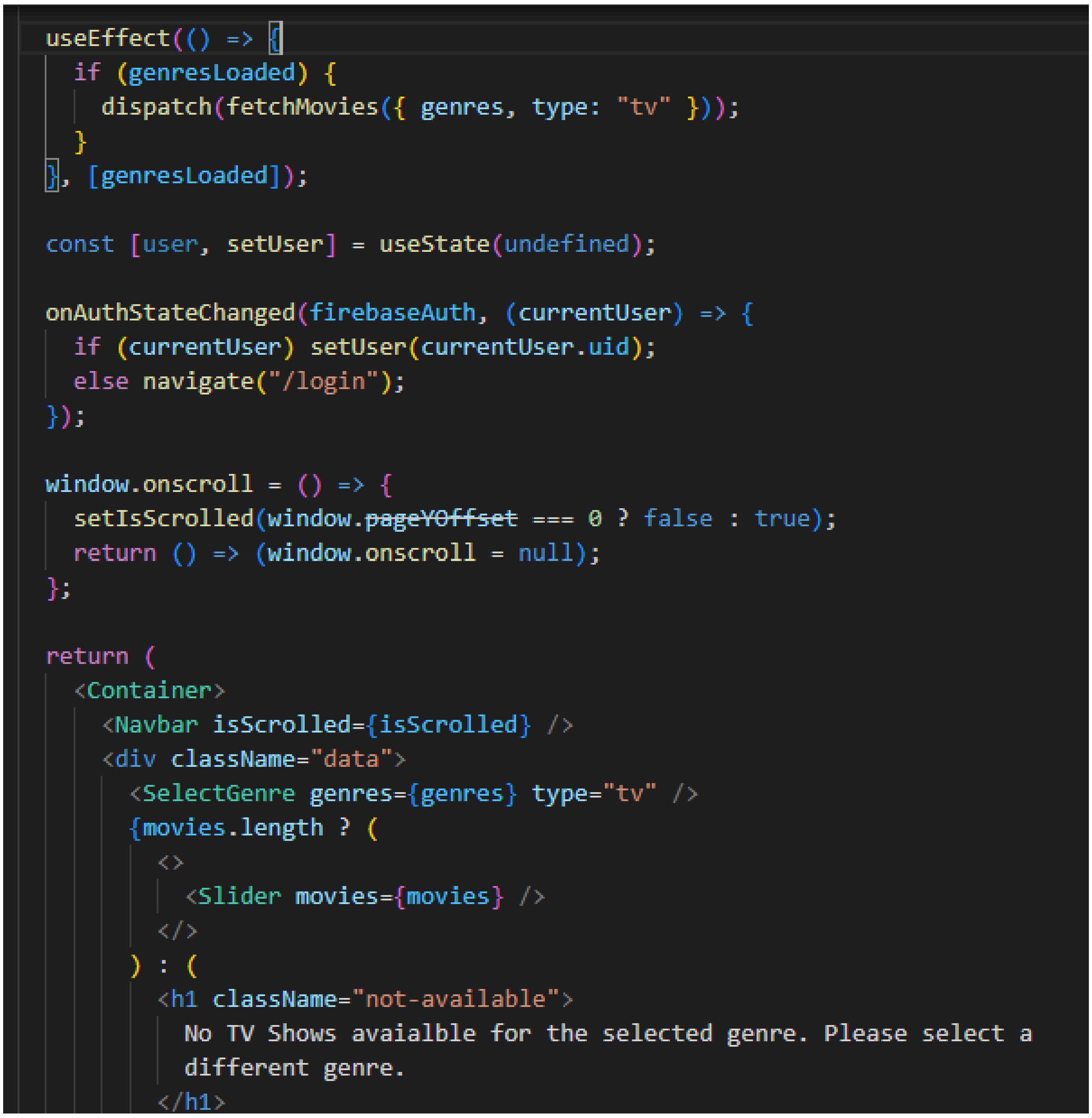
Signup.jsx



```
function Signup() {
  const [showPassword, setShowPassword] = useState(false);
  const [formValues, setFormValues] = useState({
    email: "",
    password: "",
  });
  const navigate = useNavigate();

  const handleSignIn = async () => {
    try {
      const { email, password } = formValues;
      await createUserWithEmailAndPassword(firebaseAuth, email, password);
    } catch (error) {
      console.log(error);
    }
  };

  onAuthStateChanged(firebaseAuth, (currentUser) => {
    if (currentUser) navigate("/");
  });

  return (
    <Container showPassword={showPassword}>
      <BackgroundImage />
      <div className="content">
        <Header login />
        <div className="body flex column a-center j-center">
          <div className="text flex column">
            <h1>Unlimited movies, TV shows and more.</h1>
            <h4>Watch anywhere. Cancel anytime.</h4>
            <h6>
              Ready to watch? Enter your email to create or restart membership.
```

1. **The code defines a React component named `Signup` for user registration using Firebase authentication.**
2. **It features a background image, header, and a form with email input, password input (conditional), and signup/login buttons.**
3. **State is managed for input values, and the `handleSignIn` function attempts to create a new user account.**
4. **The component dynamically adjusts its layout based on whether the password input is visible (`showPassword` state).**
5. **Styling is done with styled-components, providing a visually appealing and responsive signup interface.**

TVShows.jsx

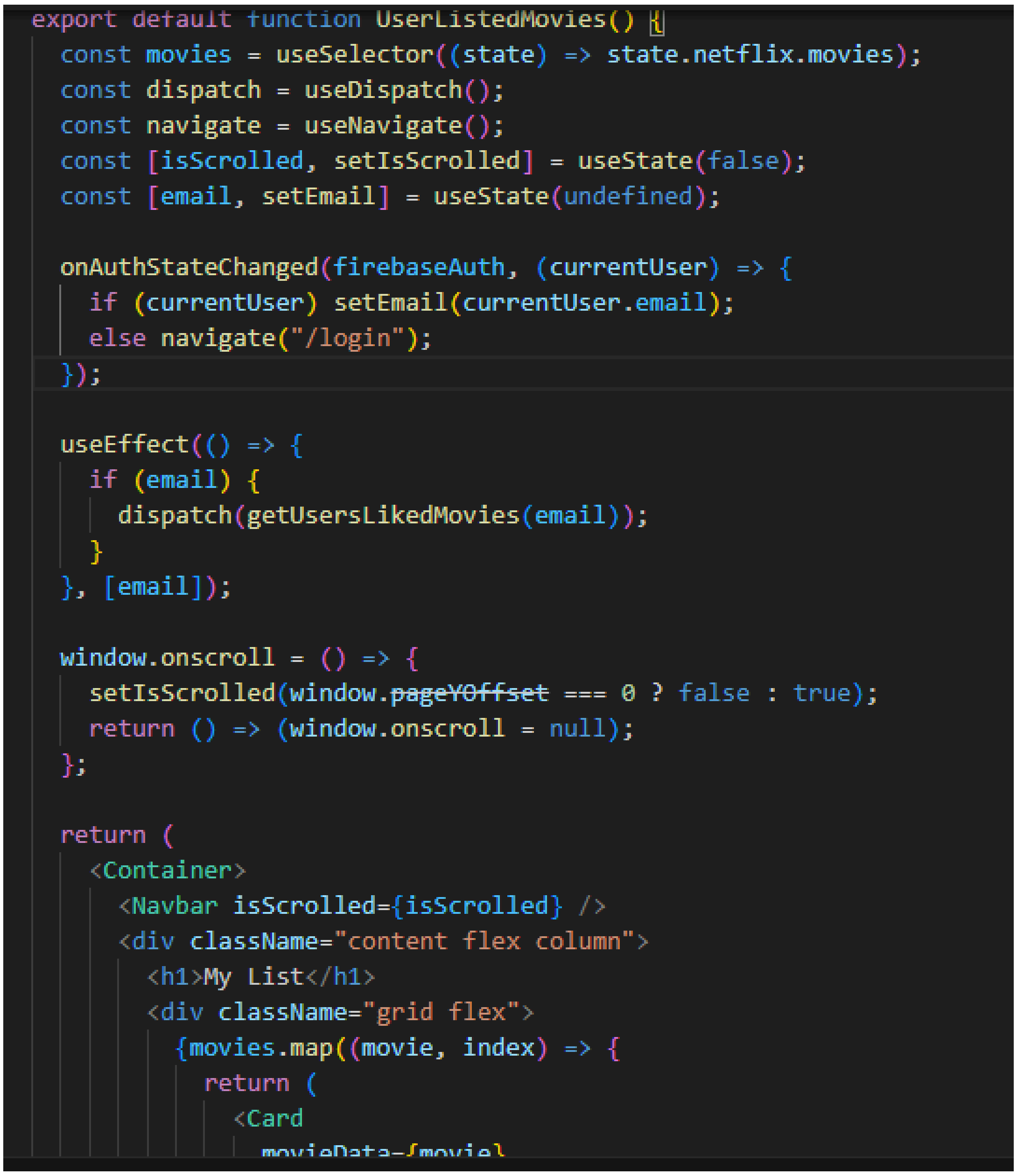


```
useEffect(() => {
  if (genresLoaded) {
    dispatch(fetchMovies({ genres, type: "tv" }));
  }
}, [genresLoaded]);

const [user, setUser] = useState(undefined);

onAuthStateChanged(firebaseAuth, (currentUser) => {
  if (currentUser) setUser(currentUser.uid);
  else navigate("/login");
});

window.onscroll = () => {
  setIsScrolled(window.~~pageYOffset~~ === 0 ? false : true);
  return () => (window.onscroll = null);
};

return (
  <Container>
    <Navbar isScrolled={isScrolled} />
    <div className="data">
      <SelectGenre genres={genres} type="tv" />
      {movies.length ? (
        <>
          <Slider movies={movies} />
        </>
      ) : (
        <h1 className="not-available">
          No TV Shows avaialble for the selected genre. Please select a
          different genre.
        </h1>
```

1. **The `TVShows` component is a React page for displaying TV shows, featuring a navigation bar, genre selection, and a slider for TV show cards.**
2. **It uses Redux for state management, fetching genres and TV show data from a store.**
3. **The component dynamically adjusts its layout based on whether TV show data is available, showing either a slider or a message indicating no shows for the selected genre.**

4. **Authentication state is checked using Firebase, and users are redirected to the login page if not authenticated.**
5. **The page is styled using styled-components, and it handles scrolling effects based on the window's scroll position.**

UserListedMovies.jsx



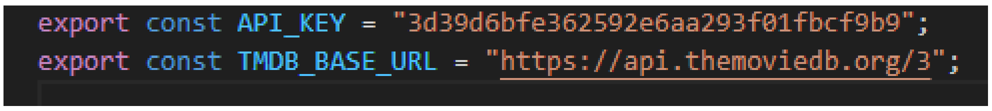
```
export default function UserListedMovies() {
  const movies = useSelector((state) => state.netflix.movies);
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const [isScrolled, setIsScrolled] = useState(false);
  const [email, setEmail] = useState(undefined);

  onAuthStateChanged(firebaseAuth, (currentUser) => {
    if (currentUser) setEmail(currentUser.email);
    else navigate("/login");
  });

  useEffect(() => {
    if (email) {
      dispatch(getUsersLikedMovies(email));
    }
  }, [email]);

  window.onscroll = () => {
    setIsScrolled(window.~~pageYOffset~~ === 0 ? false : true);
    return () => (window.onscroll = null);
  };

  return (
    <Container>
      <Navbar isScrolled={isScrolled} />
      <div className="content flex column">
        <h1>My List</h1>
        <div className="grid flex">
          {movies.map((movie, index) => {
            return (
              <Card
                movieData={movie}
```

1. **The** `UserListedMovies` **component is a React page displaying movies that a user has liked or added to their list.**

2. It utilizes Firebase authentication to retrieve the user's email and Redux for state management, fetching liked movies from the store.
3. The layout includes a navigation bar, a heading "My List," and a grid of movie cards.
4. The page dynamically adjusts its content based on the user's authentication status, redirecting to the login page if not authenticated.
5. Scrolling effects are handled based on the window's scroll position, and styled-components are used for styling.

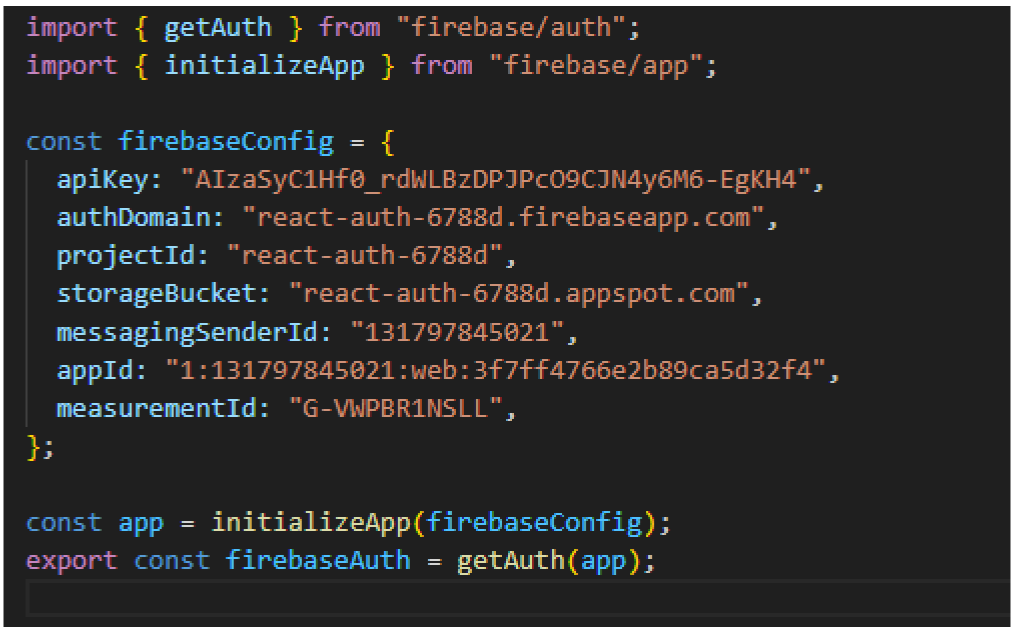
Utils used:

Constants.jsx



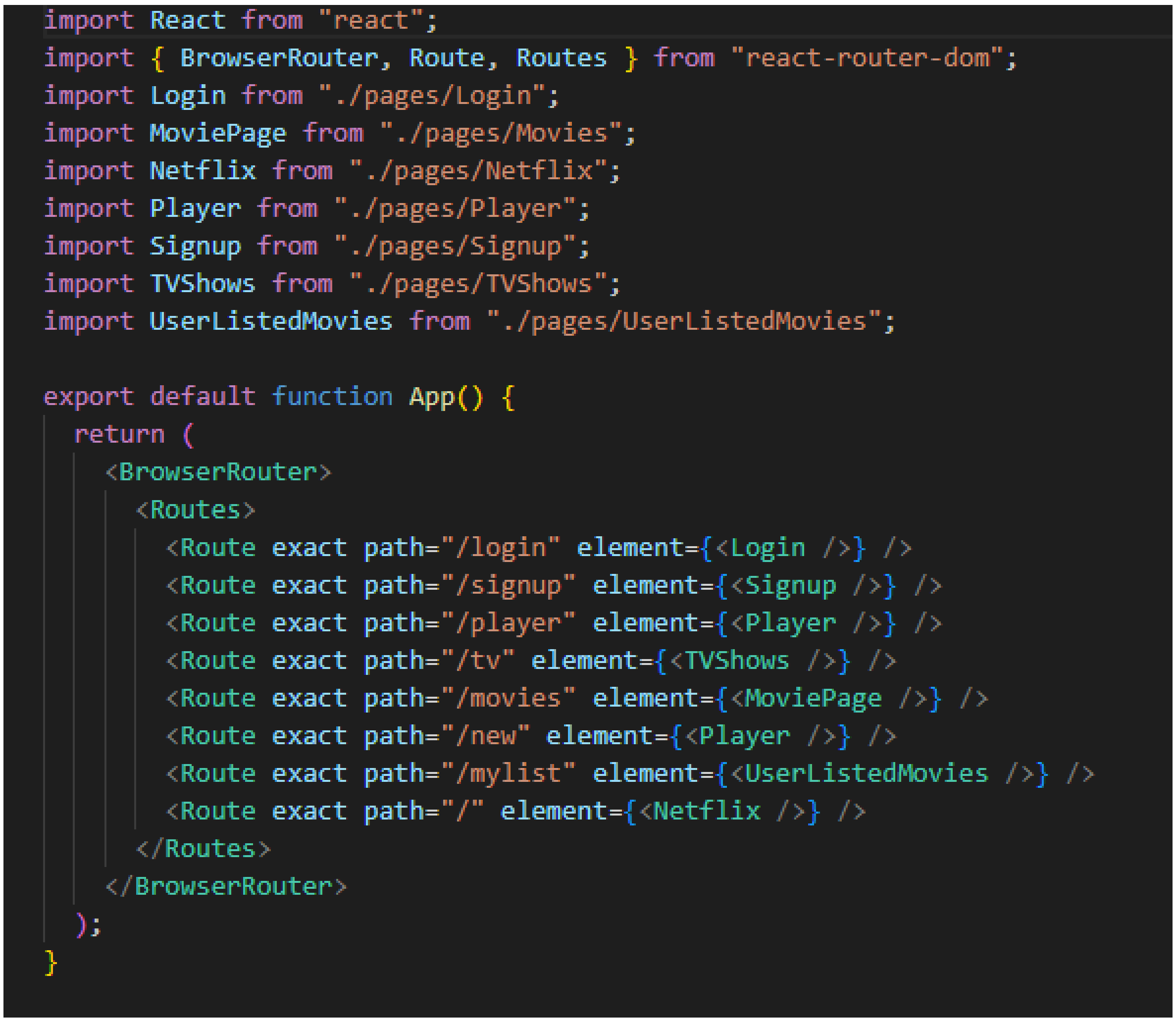
```
export const API_KEY = "3d39d6bfe362592e6aa293f01fbcf9b9";
export const TMDB_BASE_URL = "https://api.themoviedb.org/3";
```

firebase-config.js



```
import { getAuth } from "firebase/auth";
import { initializeApp } from "firebase/app";

const firebaseConfig = {
  apiKey: "AIzaSyC1Hf0_rdWLBzDPJPcO9CJN4y6M6-EgKH4",
  authDomain: "react-auth-6788d.firebaseapp.com",
  projectId: "react-auth-6788d",
  storageBucket: "react-auth-6788d.appspot.com",
  messagingSenderId: "131797845021",
  appId: "1:131797845021:web:3f7ff4766e2b89ca5d32f4",
  measurementId: "G-VWPBR1NSLL",
};

const app = initializeApp(firebaseConfig);
export const firebaseAuth = getAuth(app);
```

App.js File



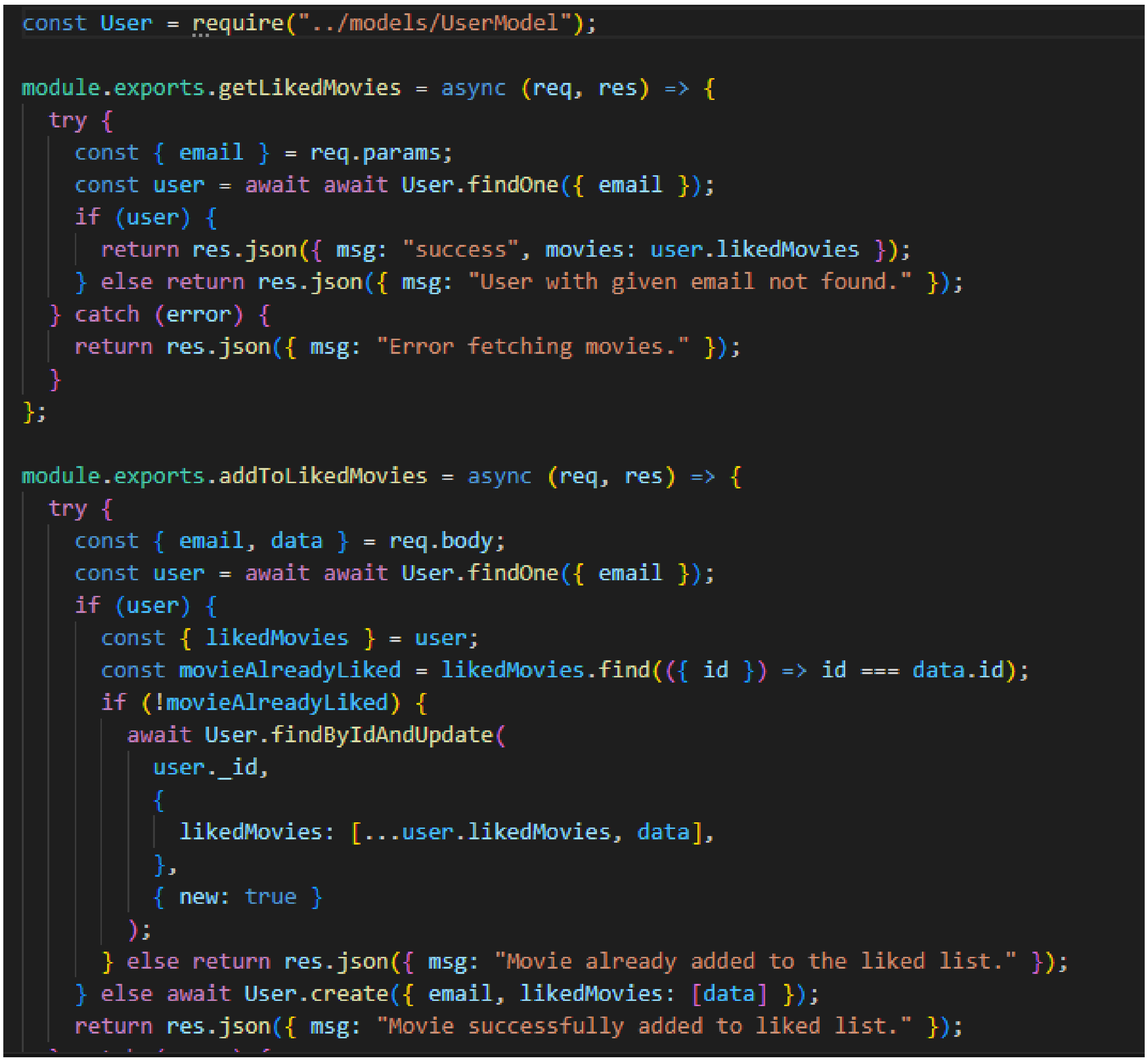
```
import React from "react";
import { BrowserRouter, Route, Routes } from "react-router-dom";
import Login from "./pages/Login";
import MoviePage from "./pages/Movies";
import Netflix from "./pages/Netflix";
import Player from "./pages/Player";
import Signup from "./pages/Signup";
import TVShows from "./pages/TVShows";
import UserListedMovies from "./pages/UserListedMovies";

export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route exact path="/login" element={<Login />} />
        <Route exact path="/signup" element={<Signup />} />
        <Route exact path="/player" element={<Player />} />
        <Route exact path="/tv" element={<TVShows />} />
        <Route exact path="/movies" element={<MoviePage />} />
        <Route exact path="/new" element={<Player />} />
        <Route exact path="/mylist" element={<UserListedMovies />} />
        <Route exact path="/" element={<Netflix />} />
      </Routes>
    </BrowserRouter>
  );
}
```

1. **The `App` component defines routes using React Router for different pages in a streaming service application.**
2. **Pages include login, signup, player, TV shows, movies, new releases, and a user's listed movies.**
3. **Components like `Login`, `Signup`, `Player`, `TVShows`, `MoviePage`, `Netflix`, and `UserListedMovies` are rendered based on the specified routes.**

Netflix-api folder

Controllers

UserController.jsx



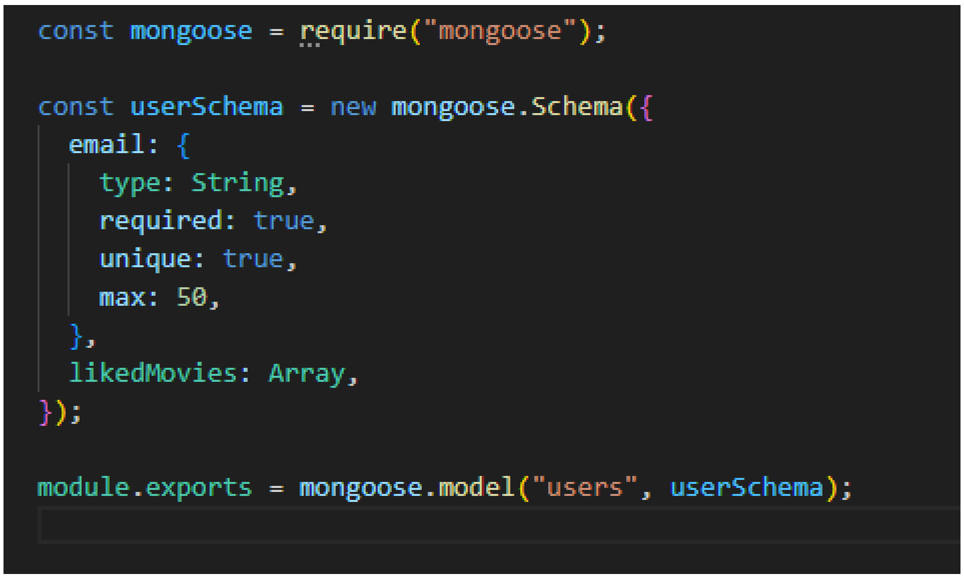
```
const User = require("../models/UserModel");

module.exports.getLikedMovies = async (req, res) => {
  try {
    const { email } = req.params;
    const user = await await User.findOne({ email });
    if (user) {
      return res.json({ msg: "success", movies: user.likedMovies });
    } else return res.json({ msg: "User with given email not found." });
  } catch (error) {
    return res.json({ msg: "Error fetching movies." });
  }
};

module.exports.addToLikedMovies = async (req, res) => {
  try {
    const { email, data } = req.body;
    const user = await await User.findOne({ email });
    if (user) {
      const { likedMovies } = user;
      const movieAlreadyLiked = likedMovies.find(({ id }) => id === data.id);
      if (!movieAlreadyLiked) {
        await User.findByIdAndUpdate(
          user._id,
          {
            likedMovies: [...user.likedMovies, data],
          },
          { new: true }
        );
      } else return res.json({ msg: "Movie already added to the liked list." });
    } else await User.create({ email, likedMovies: [data] });
    return res.json({ msg: "Movie successfully added to liked list." });
```

1. This Node.js module exports functions related to managing a user's liked movies, adding, fetching, and removing movies from the database.
2. The `getLikedMovies` function retrieves a user's liked movies based on the provided email.
3. The `addToLikedMovies` function adds a new movie to a user's liked list or creates a new user with the liked movie if the user doesn't exist.

4. The `removeFromLikedMovies` function removes a movie from a user's liked list based on the provided email and movie ID.
5. Error messages and success messages are returned in JSON format as responses.

Models

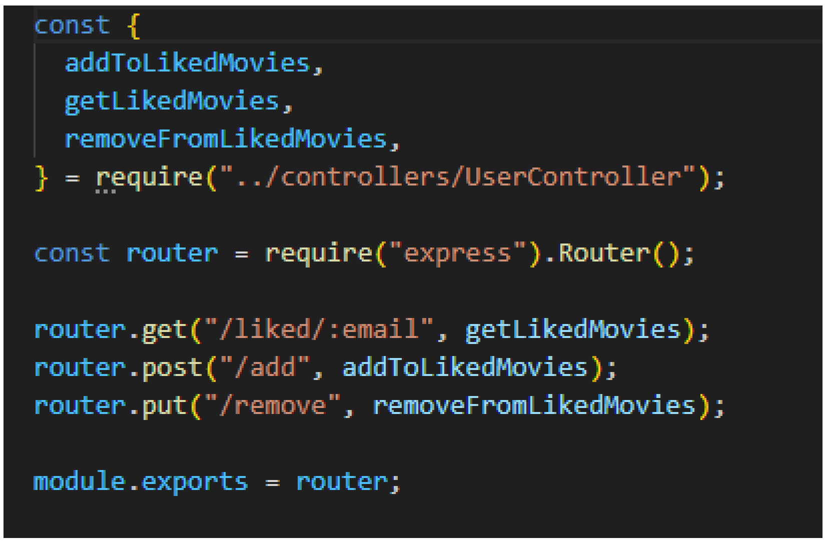
UserModel.js



```
const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({
  email: {
    type: String,
    required: true,
    unique: true,
    max: 50,
  },
  likedMovies: Array,
});

module.exports = mongoose.model("users", userSchema);
```

This is a Mongoose schema definition for a user model with an email field (string, required, unique) and an array field for liked movies. The schema is then exported as the "users" model.

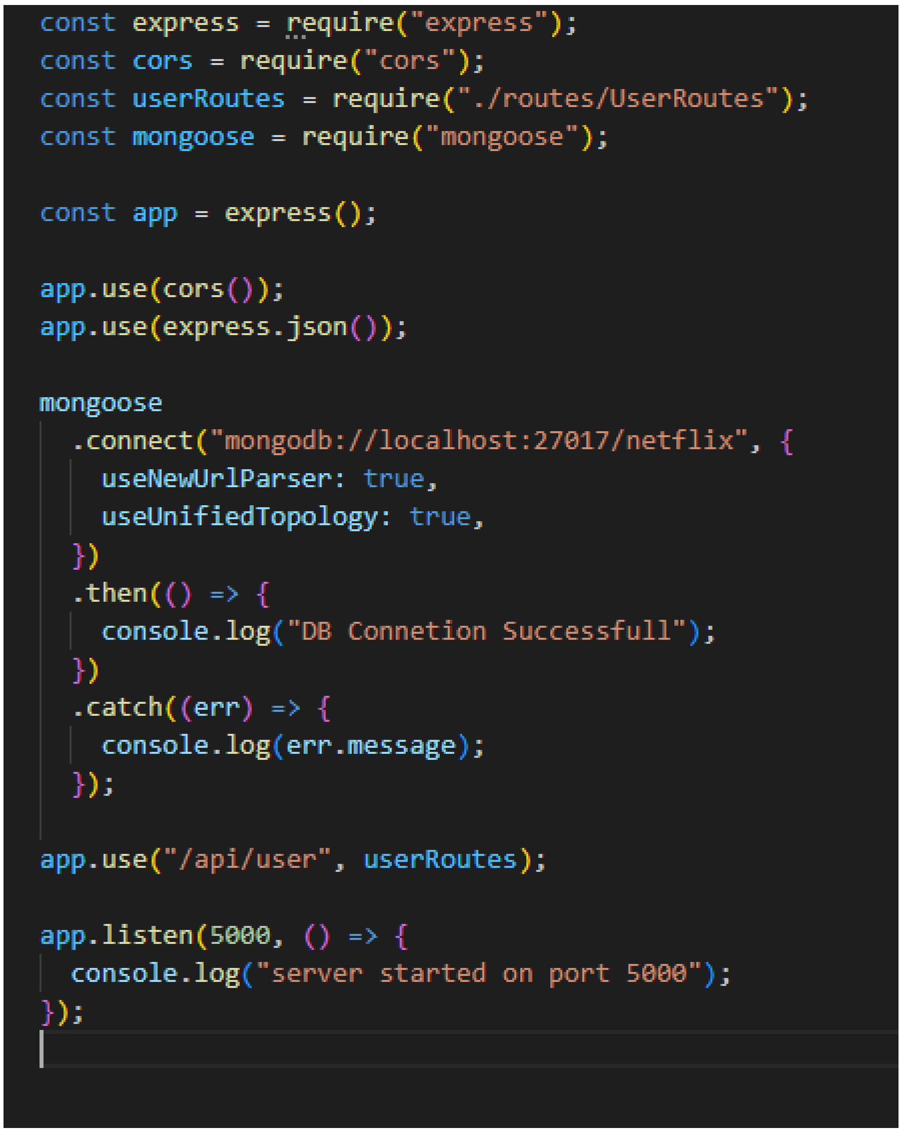
Routes.js



```
const {
  addToLikedMovies,
  getLikedMovies,
  removeFromLikedMovies,
} = require("../controllers/UserController");

const router = require("express").Router();

router.get("/liked/:email", getLikedMovies);
router.post("/add", addToLikedMovies);
router.put("/remove", removeFromLikedMovies);

module.exports = router;
```

This is an Express router configuration for handling routes related to user liked movies. It includes endpoints for retrieving liked movies, adding a movie to liked movies, and removing a movie from liked movies. The corresponding controller functions are imported and assigned to the respective routes.

Server.js File



```
const express = require("express");
const cors = require("cors");
const userRoutes = require("./routes/UserRoutes");
const mongoose = require("mongoose");

const app = express();

app.use(cors());
app.use(express.json());

mongoose
  .connect("mongodb://localhost:27017/netflix", {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  })
  .then(() => {
    console.log("DB Connetion Successfull");
  })
  .catch((err) => {
    console.log(err.message);
  });

app.use("/api/user", userRoutes);

app.listen(5000, () => {
  console.log("server started on port 5000");
});
```

This is a Node.js server using Express that connects to a MongoDB database named "netflix." It includes CORS middleware, routes for user-related actions (imported from `UserRoutes`), and listens on port 5000. The server establishes a successful connection to the MongoDB database upon startup.

CONCLUSION

In the culmination of our exploration into the Streamify website, it becomes evident that this platform for video streaming haven has successfully woven together the threads of trends and technology to deliver a superlative streaming experience. The intuitive interface, coupled with responsive design and thoughtful features, reflects a commitment to making video streaming not just a transaction but an engaging journey. The robust technological architecture supporting Streamify ensures not only the platform's reliability but also its scalability. It contributes significantly to the platform's appeal, allowing users to explore from a wide range of movie and TV shows..

REFERENCES

1. React JS Documentation. (https://reactjs.org/)
2. HTML Living Standard. (https://html.spec.whatwg.org/)
3. CSS MDN Web Docs. (https://developer.mozilla.org/en-US/docs/Web/CSS)
4. Bootstrap Documentation. (https://getbootstrap.com/)
5. Express Documentation. (https://expressjs.com/)
6. Visual Studio Code Documentation. (https://code.visualstudio.com/docs)
7. Git Documentation. (https://git-scm.com/doc)
8. GitHub Guides. (https://guides.github.com/)