VIETNAM NATIONAL UNIVERSITY OF HO CHI MINH CITY

INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

PROJECT REPORT

Feddit - A Forum

Course: *Web Application Development*

Group DQ

Members:

Vu Kien Quoc - ITITIU21295

Vu Van Do - ITITIU21

Table Of Content

I. Introduction....3

II. Project Architecture....5

III. Application Features Overview.... 11

IV. Core Components and Functionalities....15

V. Styling the Application.... 15

VI. Routing and Navigation....16

VII. State Management.... 16

VIII. Key Functionalities and Logic.... 17

IX. Challenges and Solutions....17

X. Testing the Application....17

XI. Conclusion.... 18

XII. References....15

XIII. Appendix....20

I. INTRODUCTION

The following report details the design, development, and functionality of Feddit, a Reddit-inspired web application built using modern web technologies. The application is designed as a single-page application (SPA), leveraging React.js for efficient state management, dynamic rendering, and seamless navigation. With a focus on user interactivity, Feddit provides core features such as post creation, upvoting/downvoting, commenting, and sharing capabilities—all presented through a clean, user-friendly interface.

The primary goal of this project is to replicate and extend key functionalities of a traditional online forum while adhering to best practices in modern web development. Feddit serves as both a functional product and an educational experience in implementing advanced React concepts, including component-based architecture, state management with hooks, and React Router for page navigation. Furthermore, the app is styled using modular CSS, ensuring scalability and responsiveness across devices.

This document provides a comprehensive overview of Feddit, encompassing its architecture, components, and key functionalities. Each section of the report delves into the application's features, including the home page with a post feed, individual post detail views, interactive comment sections, and static policy pages. Additionally, the report explores the challenges encountered during development, the solutions implemented, and future enhancements that could elevate the application's functionality.

By documenting every aspect of Feddit's development process, this report aims to offer insights into modern web application design while demonstrating the capabilities of React.js as a framework for building interactive and scalable user interfaces. The lessons learned throughout this project highlight the potential for continuous improvement and innovation in web development.

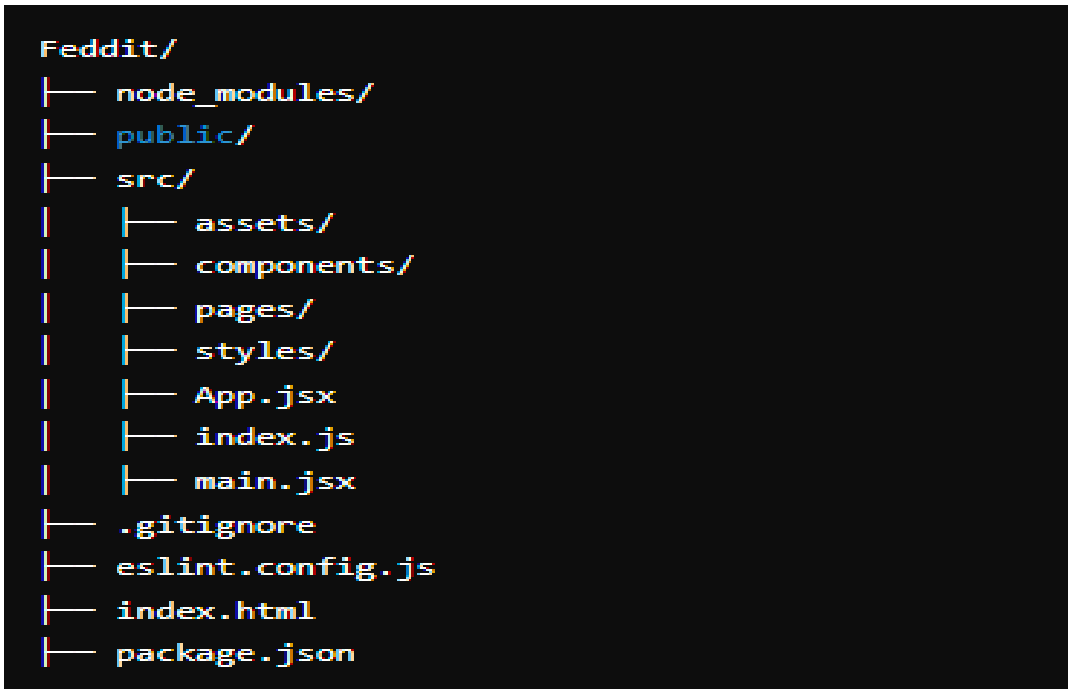
II. PROJECT ARCHITECTURE

The architecture of the "Feddit" application has been meticulously designed following modern web development principles to ensure scalability, maintainability, and responsiveness. This section provides an in-depth explanation of the project's folder structure and the purpose of each directory and file, offering insight into how different components collaborate to build the complete application.

---

1. Project Structure Overview

The "Feddit" application adopts a modular architecture, emphasizing separation of concerns and component-based design. The structure is organized as follows:



```
Feddit/
├── node_modules/
├── public/
├── src/
|    ├── assets/
|    ├── components/
|    ├── pages/
|    ├── styles/
|    ├── App.jsx
|    ├── index.js
|    ├── main.jsx
├── .gitignore
├── eslint.config.js
├── index.html
├── package.json
```

2. Node Modules Directory

The node_modules/ directory contains all the third-party dependencies and libraries required to build and run the application. These are installed using npm or yarn, as specified in the package.json file. Examples of key dependencies include:

- **React**: A JavaScript library for building user interfaces.
- **React Router**: For handling client-side routing and navigation.
- **ESLint**: For enforcing coding standards and linting the codebase.
- **Babel**: For transpiling modern JavaScript into compatible versions.

This directory is automatically generated during the setup process and should not be modified manually.

3. Public Directory

The public/ directory contains static files that are directly served to the client. These files remain unprocessed by the build tool. Important files in this directory include:

- index.html: The root HTML file where the React app is mounted. This file links the root div, which serves as the mounting point for the React components.
- Assets such as logos, favicon, and other static resources.

This directory serves as the foundation for integrating dynamic React content into the HTML skeleton.

4. Source Directory

The src/ directory is the core of the application, where all the logic, components, and styling are implemented. It is organized into four key subdirectories:

4.1. assets/

The assets/ directory stores static resources, such as images, fonts, and other files used throughout the application. This directory ensures that static resources are reusable and easily accessible across components.

4.2. components/

The components/ directory contains reusable React components. These components are smaller, modular units used to construct the application's interface. Key files include:

- Header.jsx: The navigation bar at the top of the application. It provides links to different sections like "Home," "Popular," and "Top."
- Post.jsx: A component that displays individual posts, including their title, body, upvote/downvote buttons, and share/comment options.
- PostDetails.jsx: A more detailed view of an individual post, used in the post page.
- PostFeed.jsx: The feed of posts rendered on the homepage or other sections like "Popular" or "New."
- Sidebar.jsx: The navigation sidebar with links to other pages and policies.

4.3. pages/

The pages/ directory contains React components corresponding to specific pages of the application. Each page combines multiple components to render a complete section of the app. Key files include:

- Home.jsx: Displays the main feed of posts, pulling data from the backend or a mock data source.
- CreatePost.jsx: Allows users to create a new post by entering a title, body, and optional media.

- PostPage.jsx: Provides a detailed view of an individual post, its comments, and interaction options like voting and sharing.
- Policy Pages (ContentPolicy.jsx, PrivacyPolicy.jsx, UserAgreement.jsx): Static informational pages outlining the platform’s policies.

4.4. styles/

The styles/ directory includes all the CSS files that define the application's appearance. Each component or page has a corresponding CSS file, enabling modular and scoped styling. For instance:

- Post.css: Defines the layout and appearance of individual posts.
- PostPage.css: Styles the detailed post view, including the comments section and voting buttons.
- Global Styles: Shared styles for consistency, such as index.css for general resets or base styling.

4.5. Core Files

- App.jsx: The central component that defines the main structure and routing of the application. It uses React Router to switch between pages and renders shared components like the header and sidebar.
- main.jsx: The entry point for the React application, where the App component is mounted to the root div in index.html.
- index.js: Responsible for importing necessary dependencies, initializing the app, and rendering it into the DOM.

5. Configuration Files

The project includes several configuration files:

- **.gitignore**: Lists files and directories that should not be tracked by Git, such as node_modules/ and build artifacts.
- **eslint.config.js**: Configures linting rules to enforce consistent coding standards across the project.

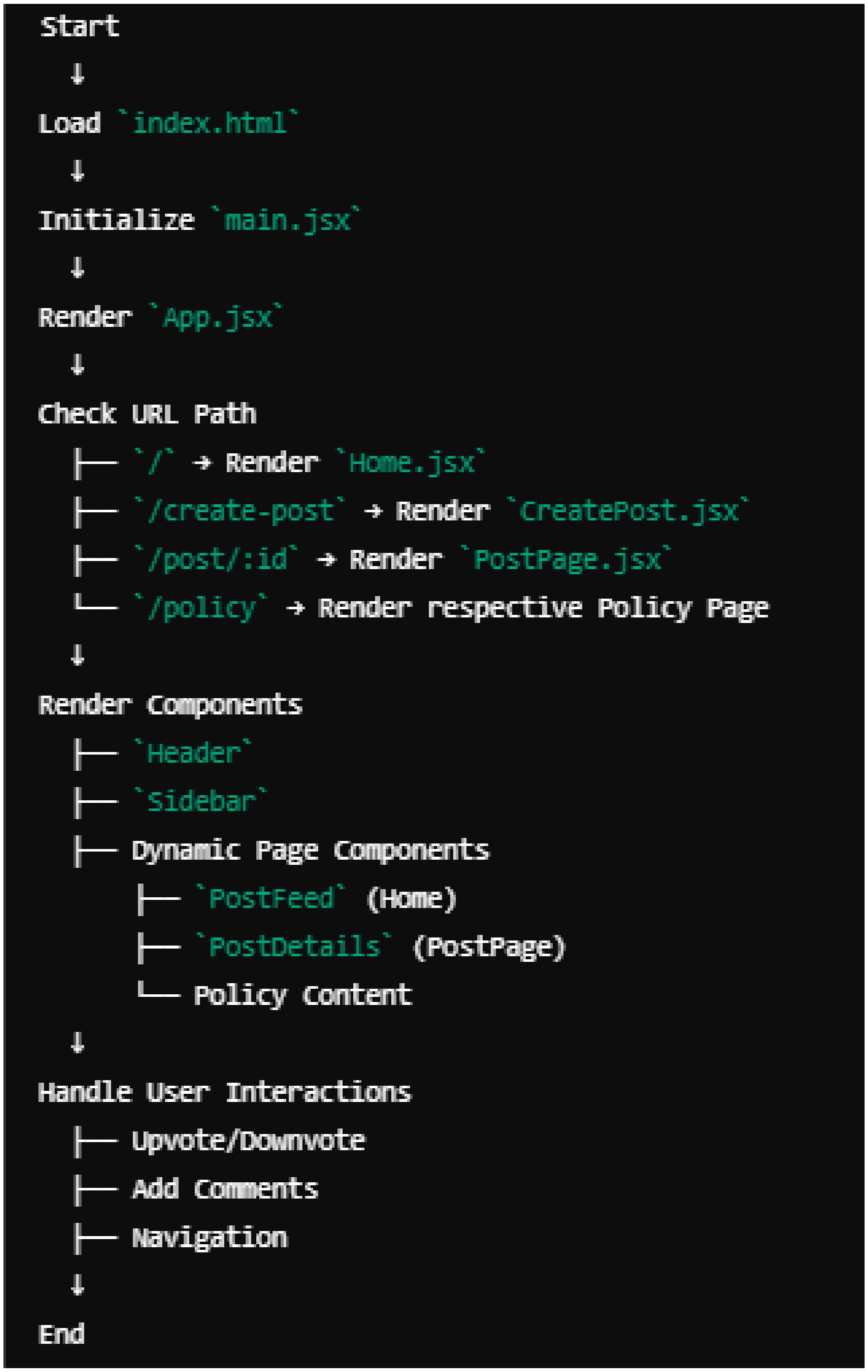
- **package.json**: Specifies project metadata, dependencies, and scripts for building, running, and testing the app.

6. Flow of the Application

The following steps describe the application flow:

- Initialization: The main.jsx file initializes the React application and renders the App.jsx component.
- Routing: Based on the current URL, React Router renders the appropriate page component, such as Home or PostPage.
- Component Rendering:
    a. The Header and Sidebar components are displayed globally.
    b. Page-specific components, such as PostFeed on the homepage or PostDetails on a post page, are rendered dynamically.
- User Interactions:
    c. Users can upvote or downvote posts using buttons in Post.jsx.
    d. Clicking a post title navigates to PostPage, displaying a detailed view of the post.
    e. Comments can be added, upvoted, or downvoted on the PostPage.
- Styling: Scoped CSS files ensure a consistent look across components and pages.

7. Flowchart



```
Start
  ↓
Load `index.html`
  ↓
Initialize `main.jsx`
  ↓
Render `App.jsx`
  ↓
Check URL Path
  ├── `/` → Render `Home.jsx`
  ├── `/create-post` → Render `CreatePost.jsx`
  ├── `/post/:id` → Render `PostPage.jsx`
  └── `/policy` → Render respective Policy Page
  ↓
Render Components
  ├── `Header`
  ├── `Sidebar`
  ├── Dynamic Page Components
      ├── `PostFeed` (Home)
      ├── `PostDetails` (PostPage)
      └── Policy Content
  ↓
Handle User Interactions
  ├── Upvote/Downvote
  ├── Add Comments
  ├── Navigation
  ↓
End
```

III. Application Features Overview

Feddit is a front-end application designed to replicate the core functionalities of a discussion-based forum. The following section highlights the primary features of the application:

3.1. Home Page: Displaying a Feed of Posts

The home page acts as the central hub of Feddit, displaying a feed of posts dynamically rendered from the PostFeed component. Each post is displayed in a card format, with functionality for upvotes, downvotes, commenting, and sharing. The feed is organized in chronological or categorical order, providing users an intuitive browsing experience.

3.2 Post View Page: Detailed Post View

The post view page offers a detailed layout for individual posts. Users can view the title, content, media, and associated comments. The page includes upvote/downvote functionality for both the post and individual comments, as well as a share button to copy the post's URL to the clipboard.

3.3 Commenting System

Users can engage in discussions through the commenting system, which supports upvoting/downvoting comments. The system utilizes state management to dynamically update the comments in real time. Nested comments can also be implemented as a future enhancement.

3.4 Sidebar Navigation

The sidebar navigation facilitates seamless movement between different categories such as Home, Popular, New, and Top. The sidebar is styled to provide clear active states, ensuring a user-friendly navigation experience.

3.5 Policies Pages

Dedicated pages for Content Policy, Privacy Policy, and User Agreement provide users with detailed guidelines on using the platform. These static pages are accessible from the footer or sidebar links.

IV. Core Components and Functionalities

4.1 Header Component (Header.jsx)

Function: The header serves as the primary navigation bar, featuring the app's branding, a search bar, and a log-in button.

Key Functionalities:

- **Branding:** Displays the name "Feddit" prominently.
- **Search Bar:** Allows users to search posts based on keywords.
- **Navigation:** Provides links to essential sections of the app, ensuring accessibility.

Code Walkthrough: The Header.jsx component is implemented with simple JSX and styled using Header.css. It uses React's state to manage user input in the search bar, with placeholder integration for future backend support.

4.2 Sidebar Component (Sidebar.jsx)

Function: The sidebar offers vertical navigation between categories such as Home, Popular, New, and Top.

Key Functionalities:

- Displays links with clear labels.
- Visual active states for the selected category using conditional styling.
- Extensible structure for adding new categories.

Styling: The Sidebar.css file ensures responsive design and hover effects. It adopts a clean layout, ensuring consistency with the app's theme.

4.3 PostFeed Component (PostFeed.jsx)

Function: The PostFeed component renders all posts dynamically on the home page using React's map() function.

Key Features:

- **Dynamic Rendering:** Maps through an array of post objects, passing data to the Post component for individual rendering.
- **Integration with Post Component:** Ensures modularity and separation of concerns.

Code Highlight: The PostFeed.jsx leverages props to ensure that data flows seamlessly between the feed and individual posts. This enhances maintainability and scalability.

4.4 Post Component (Post.jsx)

Function: The Post.jsx component represents an individual post card displayed on the home page.

Key Features:

- **Voting System:** Users can upvote or downvote posts, with real-time updates using React's useState().
- **Media Support:** Displays images or videos associated with posts.
- **Interaction Buttons:** Comment and Share buttons allow further engagement.

State Management:

- useState() is used to manage vote counts.

- Event handlers (onClick) are employed to update the state dynamically.

4.5 PostPage Component (PostPage.jsx)

Function: The PostPage component provides a detailed view of a single post, showcasing its content, media, and comments.

Key Features:

- **Post Details:** Displays the title, timestamp, author, upvote/downvote buttons, and share functionality.
- **Comment Section:** Allows users to add new comments and vote on existing ones.
- **Clipboard Copy:** The share button copies the post URL to the clipboard.

Code Highlight: The component uses useParams() to retrieve the post ID from the URL, dynamically fetching the corresponding post data.

4.6 Comments Functionality

Nested Comments: The commenting system supports nested replies, allowing users to engage in threaded discussions. Each comment includes upvote/downvote functionality.

State Management: The comments are managed using useState() to ensure real-time updates when users add or interact with comments.

4.7 Policies Pages

Static Pages: The policies pages, implemented as ContentPolicy.jsx, PrivacyPolicy.jsx, and UserAgreement.jsx, outline the app's usage guidelines. These components use simple JSX structures with basic styling from CSS files.

4.8 Connection to MongoDB (Server.js):

Server.js is responsible for establishing connection between MongoDB and the project, creating a local server to run the web application.

4.9 Comments Model (Comment.js):

Comments model defines the fields of a Comment Object, with author and post fields refer to User model and Post model. One user can make multiple comments in one post.

4.10 Post Model (Post.js):

Post model defines the fields of a Post Object, with author field refers to the User model. One post belongs to one user and can have multiple comments.

4.11 User Model (User.js):

User model defines the fields of an User Object, with posts field refers to the Post Model. One user can have multiple posts.

4.12 Authentication Route (auth.js):

The authentication route handles all operations of user identification and authentication.

- The POST /register route allows new users to register. It extracts username and password from the request body, hashes the password using bcrypt before saving it to the database, and returns a 201 status code if the process is successful.

- The POST /login route checks if the user exists by querying the database with the provided username. If the user is found, it compares the provided password with the stored hashed password. If the credentials are valid, a JWT is generated and returned to the user along with username.
- The POST /forgot-password route allows users to request a password reset by checking the user's email and generating a reset token with expiration time. An email is sent to the user with a link to reset their password using 'nodemailer'.
- The POST /reset-password/:token: routes allows users to reset their password using a token. It checks if the token is valid and not expired. If so, it hashes the new password and updates the user's password in the database. The reset token and its expiration date are cleared to prevent reuse.

4.13 Post Route (posts.js):

The posts.js file defines the routes for handling operations related to posts in a Node.js application using the Express framework.

- The POST / route allows clients to create a new post. It receives the post data from the request body and creates a new instance of the 'Post' Model using the provided data. The new post is saved to the database using 'await newPost.save()'. Upon successful action, a response with status code 201 is sent back to the client along with the newly created post in JSON format.
- GET /: route retrieves all posts from the database, using 'Post.find()' to fetch all documents in the posts collection and send them back to the client in JSON format.
- PUT /:id: route allows clients to update existing posts. The id of the post to be updated is extracted from the request parameters ('req.params').

Post.findByIdAndUpdate(id, req.body, { new: true }) is used to find the post by its ID and update it with the new data provided in the request body. The { new: true } option ensures that the updated document is returned. The updated post is sent back to the client in JSON format.

- DELETE /:id: This route allows clients to delete a post. The id of the post to be deleted is extracted from the request parameters. Post.findByIdAndDelete(id) is used to find and delete the post by its ID. A response with a status code of 204 (No Content) is sent back to indicate that the deletion was successful and there is no content to return.

V. Styling the Application

5.1 Styling Methodology

Feddit employs a modular approach to styling, ensuring that each component has its dedicated CSS file. This approach enhances scalability and maintainability.

5.2 Overview of Key CSS Files

- **Post.css:** Styles for individual posts.
- **PostPage.css:** Styles for detailed post views and comments.
- **Sidebar.css:** Sidebar design with hover and active effects.
- **App.css:** Global styles for layout and responsiveness.

5.3 Specific Styling Highlights

- **Voting Buttons:** Designed with hover effects and intuitive color coding (orange for upvotes, blue for downvotes).
- **Comment Section:** Clean layout with nested comment support.
- **Share Button:** Interactive hover effects and seamless clipboard functionality.

VI. Routing and Navigation

React Router Implementation: Feddit uses React Router for single-page application (SPA) navigation, enabling smooth transitions between pages.

Key Routes:

- / – Home Page.
- /post/:id – Individual Post View.
- /create-post – Post Creation Page.

Benefits: SPA architecture enhances user experience by eliminating the need for full-page reloads, ensuring faster navigation.

VII. State Management

React Hooks: State management is handled using React hooks (useState, useParams), enabling efficient data flow across components.

Examples:

- **Voting System:** Real-time updates for upvotes/downvotes.
- **Commenting System:** Dynamic rendering and state updates when new comments are added.

Advantages: React Hooks provide a clean and functional way to manage state, eliminating the need for complex state management libraries in this application.

VIII. Key Functionalities and Logic

8.1 Upvote/Downvote Logic

Posts and comments feature upvote/downvote buttons, with real-time updates using React's useState() and event handlers.

8.2 Share Button Logic

The share button copies the post URL to the clipboard, implemented using the navigator.clipboard.writeText() API.

8.3 Comment Rendering

The comments section dynamically renders nested comments, leveraging state updates to display new comments in real time.

8.4 User Authentication

Users can login, create a new user, and change the password using authentication routes.

8.5 CRUD Operations for Posts

Users can create, receive, modify, and delete posts using the posts routes.

IX. Challenges and Solutions

Challenges:

- **Dynamic Upvote Counts:** Ensuring accurate real-time updates across components.
- **Clipboard Copy:** Handling browser compatibility for the clipboard API.
- **CSS Styling:** Maintaining a consistent layout across pages.

Solutions:

- Leveraging React's useState() for state synchronization.
- Using fallback logic for older browsers in clipboard functionality.
- Modularizing CSS for consistent design.

X. Testing the Application

- **Manual Testing:** All functionalities were tested manually to ensure seamless user experience.

- **Test Cases:**
- **Voting Buttons:** Validating real-time updates.
- **Comment Submission:** Ensuring new comments render correctly.
- **Navigation:** Verifying smooth transitions between pages.

XI. Future Enhancements

- Extending support for nested comments.
- Introducing a dark mode toggle for improved accessibility.

XII. Conclusion

Feddit successfully replicates the core functionalities of a discussion forum, showcasing React's power for building interactive and scalable front-end applications. Through this project, key skills such as state management, component-based architecture, and CSS styling were honed. The application's modular design ensures scalability, paving the way for future enhancements such as backend integration and advanced features.

XIII. References

- **React Documentation**
    - Link: https://react.dev
    - Description: Comprehensive documentation and tutorials for React.js, including state management, components, and routing.
- **React Router Documentation**
    - Link: https://reactrouter.com
    - Description: Official documentation for React Router, detailing SPA navigation, dynamic routing, and integration.
- **CSS Tricks**
    - Link: https://css-tricks.com
    - Description: A resource for modern CSS techniques and design patterns, including responsive layouts and hover effects.
- **MDN Web Docs - JavaScript**

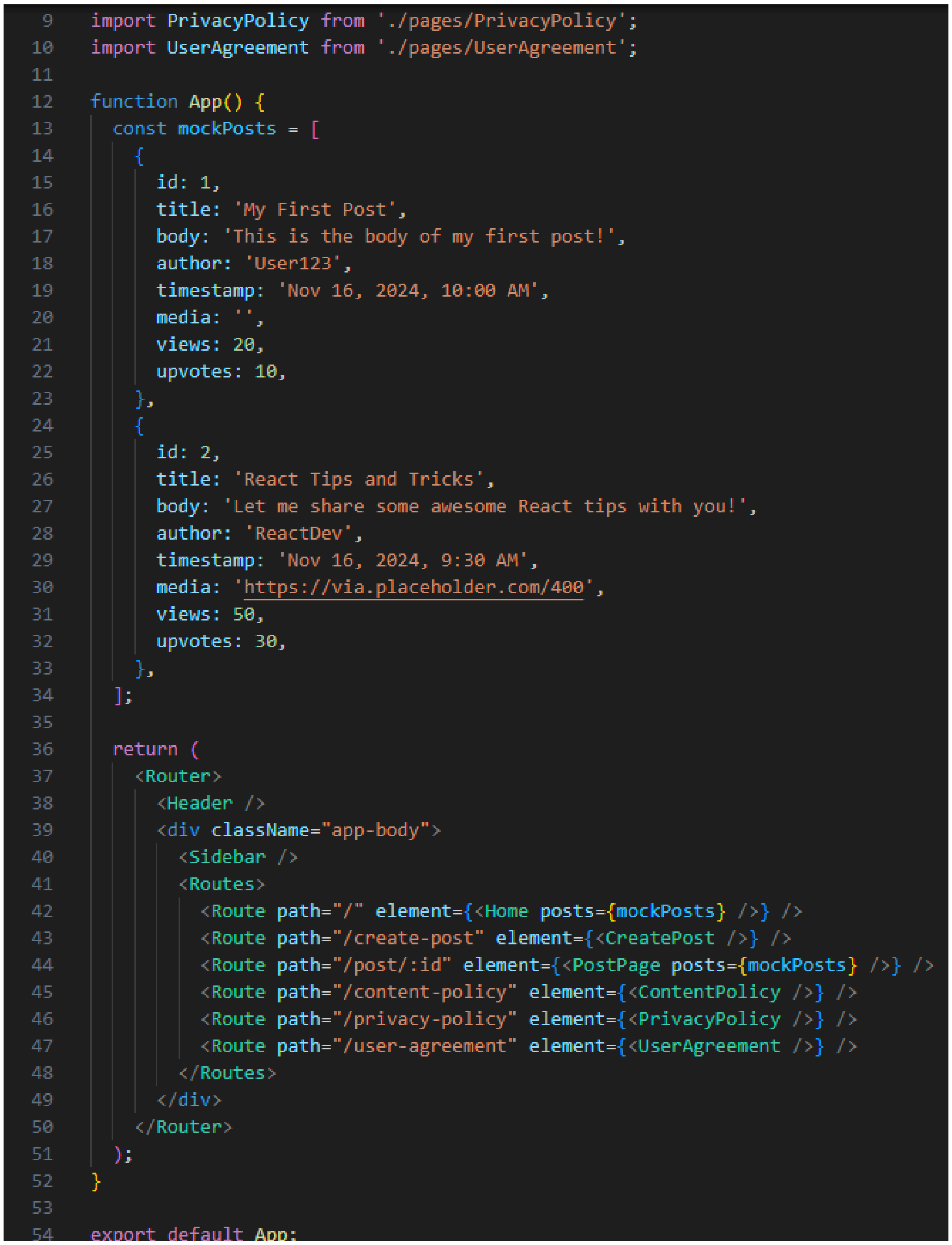
- Link: https://developer.mozilla.org/en-US/docs/Web/JavaScript
- Description: Mozilla Developer Network (MDN) documentation on JavaScript, covering ES6 features, DOM manipulation, and more.

- **Can I Use**
  - Link: https://caniuse.com
  - Description: A resource to check browser compatibility for CSS features, JavaScript functions, and modern web standards.
- **Stack Overflow**
  - Link: https://stackoverflow.com
  - Description: A community-driven platform for troubleshooting coding issues and sharing solutions for React, CSS, and JavaScript.
- **CodePen**
  - Link: https://codepen.io
  - Description: An interactive coding platform to experiment with HTML, CSS, and JavaScript designs, often used for UI inspiration.
- **W3Schools React Tutorial**
  - Link: https://www.w3schools.com/react/
  - Description: Beginner-friendly tutorials on React.js basics, components, props, and state management.
- **Netlify Blog - SPA Development**
  - Link: https://www.netlify.com/blog/
  - Description: Articles and guides on building Single Page Applications (SPAs) with modern frameworks like React.js.
- **GitHub React Repository**
  - Link: https://github.com/facebook/react
  - Description: Official React.js source code and documentation repository on GitHub.

XIV. Appendix

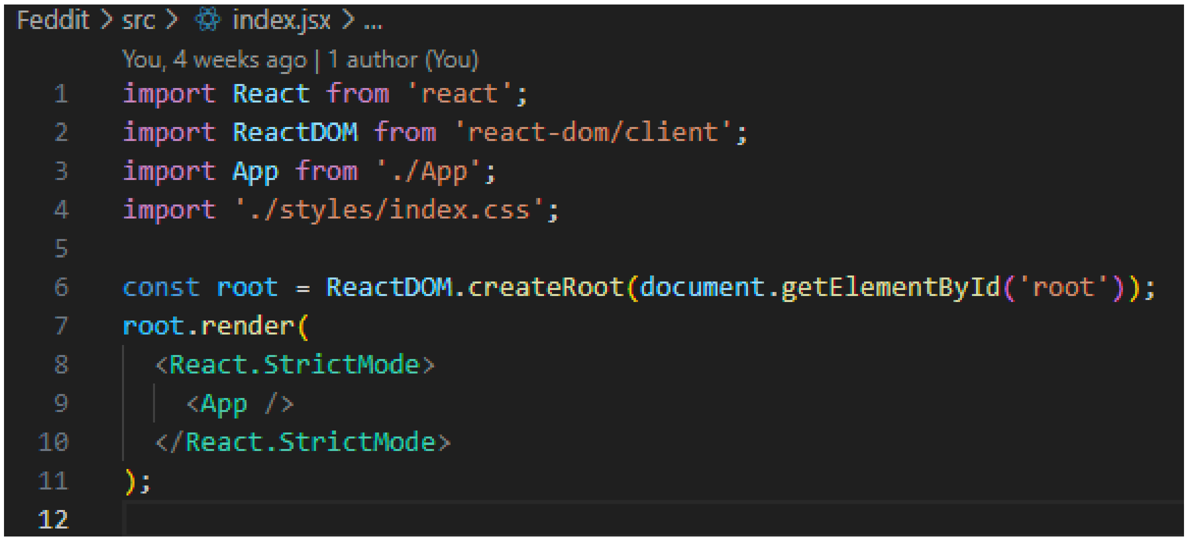
a) <u>Front-end:</u>

- **Main:**
    - **App.jsx:**

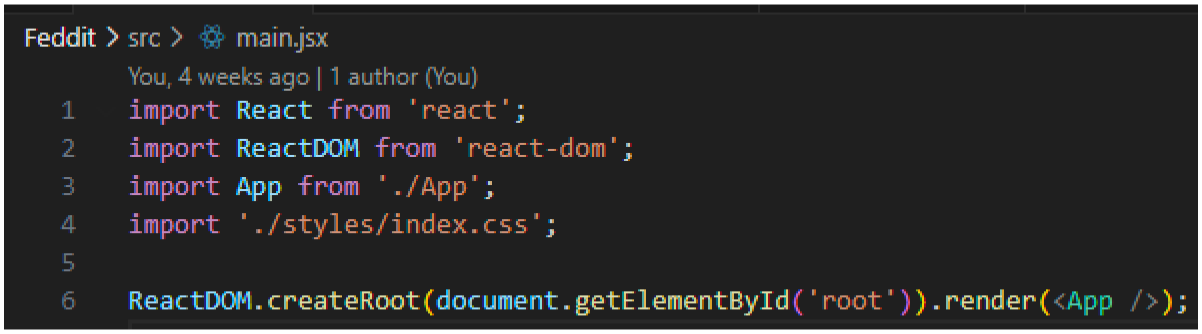


```jsx
import PrivacyPolicy from './pages/PrivacyPolicy';
import UserAgreement from './pages/UserAgreement';

function App() {
  const mockPosts = [
    {
      id: 1,
      title: 'My First Post',
      body: 'This is the body of my first post!',
      author: 'User123',
      timestamp: 'Nov 16, 2024, 10:00 AM',
      media: '',
      views: 20,
      upvotes: 10,
    },
    {
      id: 2,
      title: 'React Tips and Tricks',
      body: 'Let me share some awesome React tips with you!',
      author: 'ReactDev',
      timestamp: 'Nov 16, 2024, 9:30 AM',
      media: 'https://via.placeholder.com/400',
      views: 50,
      upvotes: 30,
    },
  ];

  return (
    <Router>
      <Header />
      <div className="app-body">
        <Sidebar />
        <Routes>
          <Route path="/" element={<Home posts={mockPosts} />} />
          <Route path="/create-post" element={<CreatePost />} />
          <Route path="/post/:id" element={<PostPage posts={mockPosts} />} />
          <Route path="/content-policy" element={<ContentPolicy />} />
          <Route path="/privacy-policy" element={<PrivacyPolicy />} />
          <Route path="/user-agreement" element={<UserAgreement />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

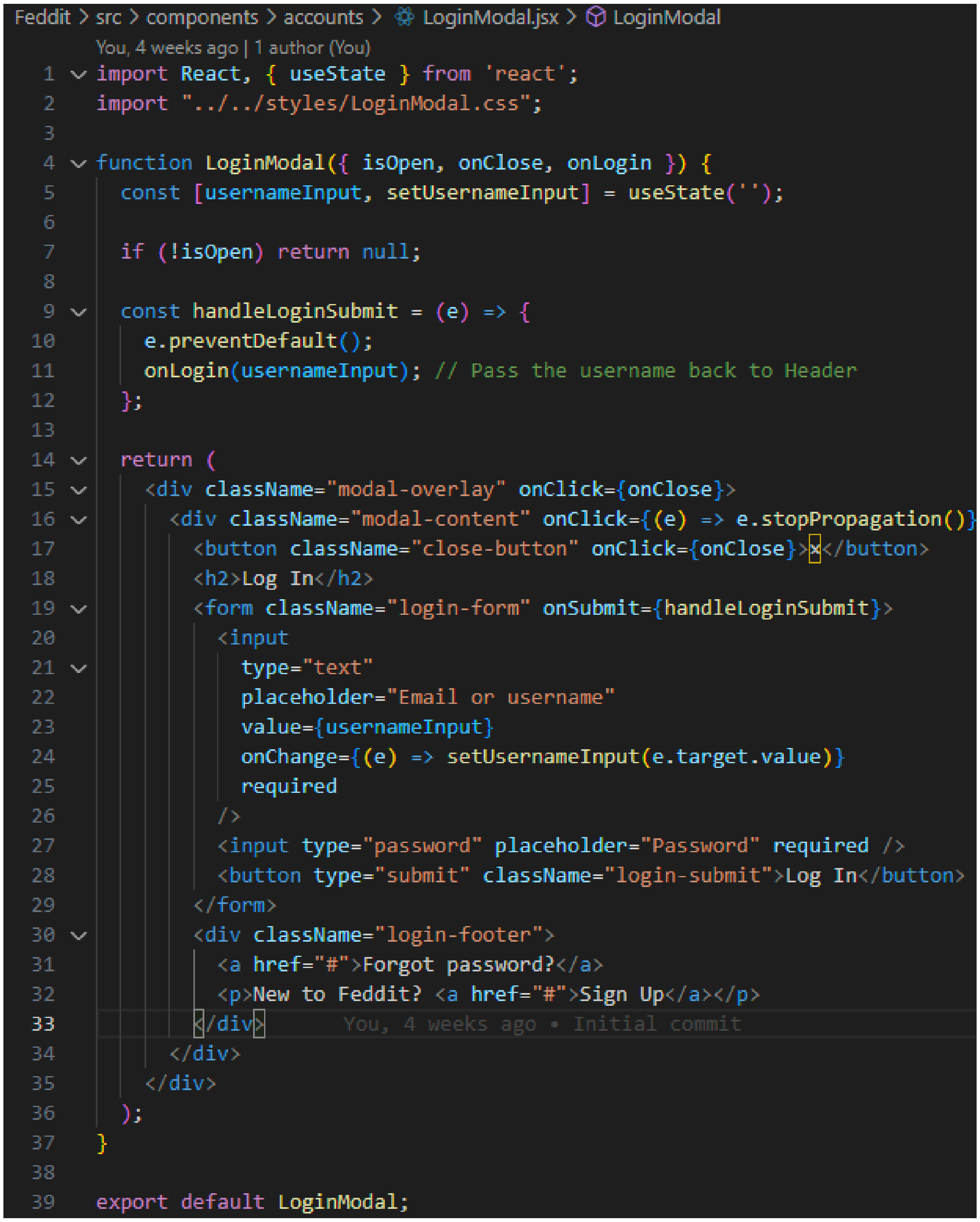
    - **index.jsx:**


Feddit > src > index.jsx > ...
You, 4 weeks ago | 1 author (You)
```jsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import './styles/index.css';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

```

- **main.jsx:**


Feddit > src > main.jsx
You, 4 weeks ago | 1 author (You)
```jsx
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import './styles/index.css';

ReactDOM.createRoot(document.getElementById('root')).render(<App />);
```

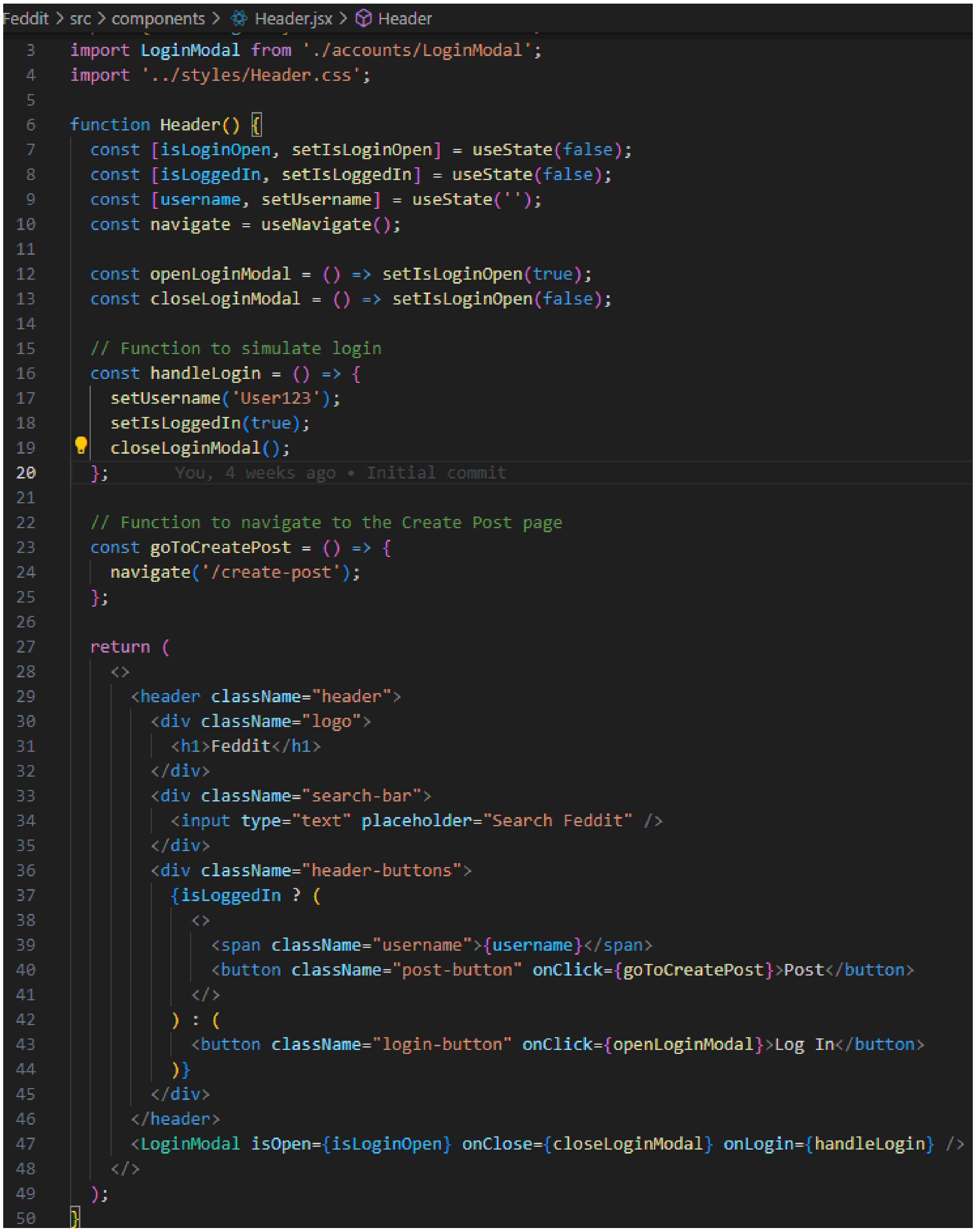
- **Account:**
  - **LoginModal.jsx:**


Feddit > src > components > accounts > LoginModal.jsx > LoginModal

You, 4 weeks ago | 1 author (You)

```jsx
import React, { useState } from 'react';
import "../../styles/LoginModal.css";

function LoginModal({ isOpen, onClose, onLogin }) {
  const [usernameInput, setUsernameInput] = useState('');

  if (!isOpen) return null;

  const handleLoginSubmit = (e) => {
    e.preventDefault();
    onLogin(usernameInput); // Pass the username back to Header
  };

  return (
    <div className="modal-overlay" onClick={onClose}>
      <div className="modal-content" onClick={(e) => e.stopPropagation()}
        <button className="close-button" onClick={onClose}>×</button>
        <h2>Log In</h2>
        <form className="login-form" onSubmit={handleLoginSubmit}>
          <input
            type="text"
            placeholder="Email or username"
            value={usernameInput}
            onChange={(e) => setUsernameInput(e.target.value)}
            required
          />
          <input type="password" placeholder="Password" required />
          <button type="submit" className="login-submit">Log In</button>
        </form>
        <div className="login-footer">
          <a href="#">Forgot password?</a>
          <p>New to Feddit? <a href="#">Sign Up</a></p>
        </div>        You, 4 weeks ago • Initial commit
      </div>
    </div>
  );
}

export default LoginModal;
```

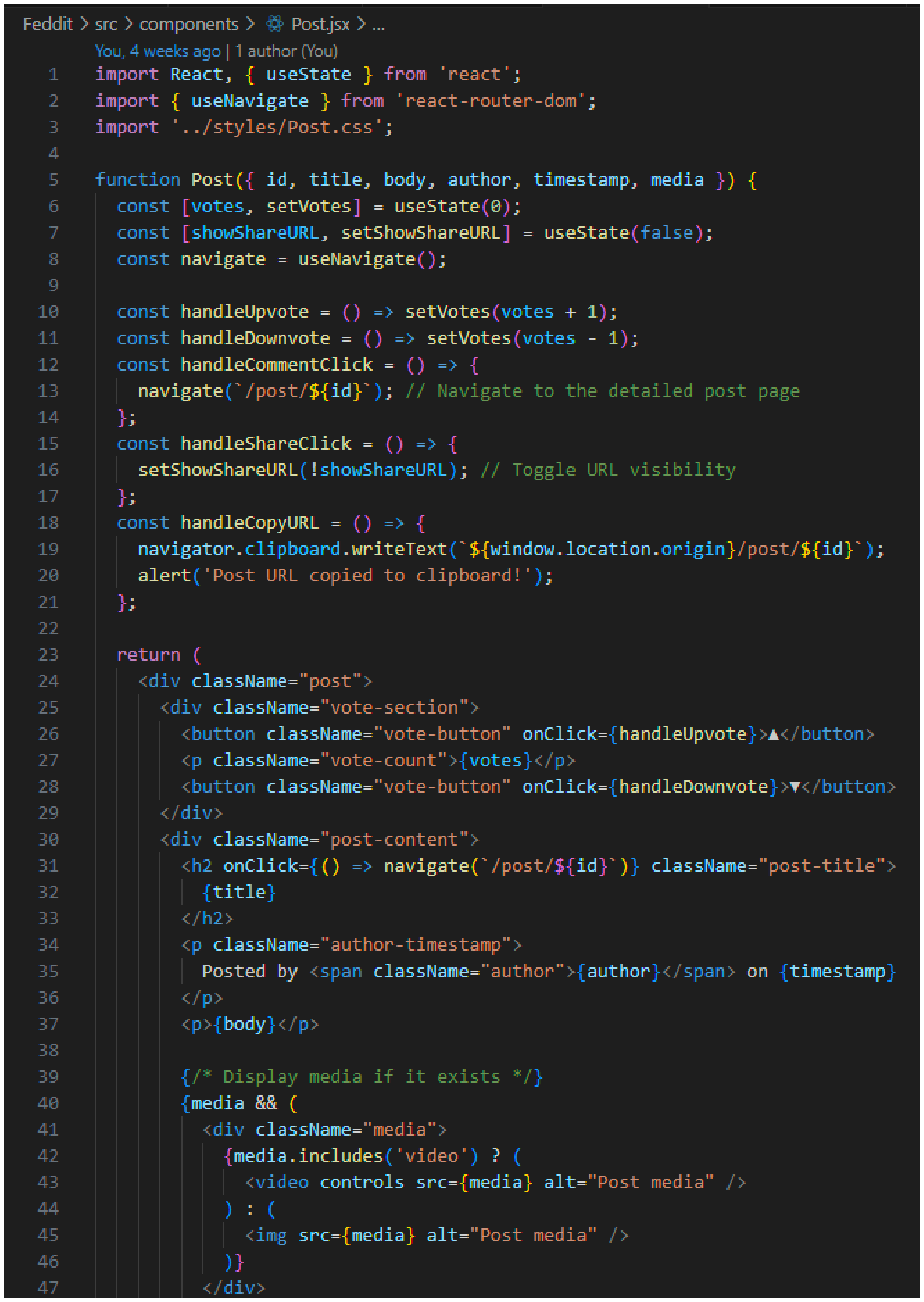
- **Components:**
  - **Headers.jsx:**



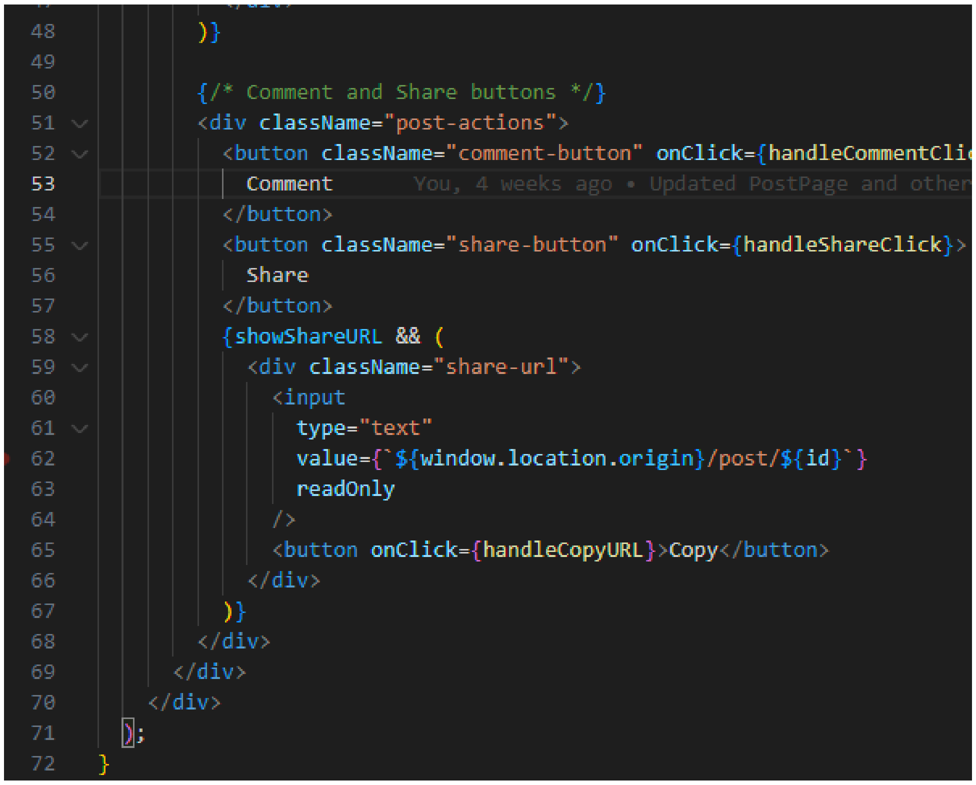
Feddit > src > components > Header.jsx > Header

```
import LoginModal from './accounts/LoginModal';
import '../styles/Header.css';

function Header() {
  const [isLoginOpen, setIsLoginOpen] = useState(false);
  const [isLoggedIn, setIsLoggedIn] = useState(false);
  const [username, setUsername] = useState('');
  const navigate = useNavigate();

  const openLoginModal = () => setIsLoginOpen(true);
  const closeLoginModal = () => setIsLoginOpen(false);

  // Function to simulate login
  const handleLogin = () => {
    setUsername('User123');
    setIsLoggedIn(true);
    closeLoginModal();
  };      You, 4 weeks ago • Initial commit

  // Function to navigate to the Create Post page
  const goToCreatePost = () => {
    navigate('/create-post');
  };

  return (
    <>
      <header className="header">
        <div className="logo">
          <h1>Feddit</h1>
        </div>
        <div className="search-bar">
          <input type="text" placeholder="Search Feddit" />
        </div>
        <div className="header-buttons">
          {isLoggedIn ? (
            <>
              <span className="username">{username}</span>
              <button className="post-button" onClick={goToCreatePost}>Post</button>
            </>
          ) : (
            <button className="login-button" onClick={openLoginModal}>Log In</button>
          )}
        </div>
      </header>
      <LoginModal isOpen={isLoginOpen} onClose={closeLoginModal} onLogin={handleLogin} />
    </>
  );
}
```

- **Post.jsx:**

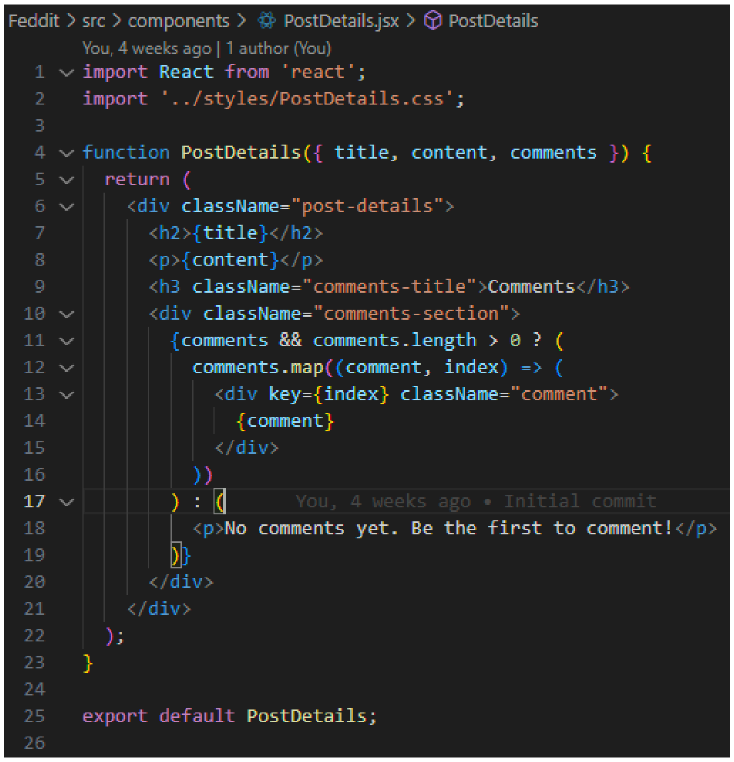

Feddit > src > components > Post.jsx > ...
You, 4 weeks ago | 1 author (You)

```jsx
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import '../styles/Post.css';

function Post({ id, title, body, author, timestamp, media }) {
  const [votes, setVotes] = useState(0);
  const [showShareURL, setShowShareURL] = useState(false);
  const navigate = useNavigate();

  const handleUpvote = () => setVotes(votes + 1);
  const handleDownvote = () => setVotes(votes - 1);
  const handleCommentClick = () => {
    navigate(`/post/${id}`); // Navigate to the detailed post page
  };
  const handleShareClick = () => {
    setShowShareURL(!showShareURL); // Toggle URL visibility
  };
  const handleCopyURL = () => {
    navigator.clipboard.writeText(`${window.location.origin}/post/${id}`);
    alert('Post URL copied to clipboard!');
  };

  return (
    <div className="post">
      <div className="vote-section">
        <button className="vote-button" onClick={handleUpvote}>▲</button>
        <p className="vote-count">{votes}</p>
        <button className="vote-button" onClick={handleDownvote}>▼</button>
      </div>
      <div className="post-content">
        <h2 onClick={() => navigate(`/post/${id}`)} className="post-title">
          {title}
        </h2>
        <p className="author-timestamp">
          Posted by <span className="author">{author}</span> on {timestamp}
        </p>
        <p>{body}</p>

        {/* Display media if it exists */}
        {media && (
          <div className="media">
            {media.includes('video') ? (
              <video controls src={media} alt="Post media" />
            ) : (
              <img src={media} alt="Post media" />
            )}
          </div>
```



```
            )}

            {/* Comment and Share buttons */}
            <div className="post-actions">
              <button className="comment-button" onClick={handleCommentCli
                Comment       You, 4 weeks ago • Updated PostPage and other
              </button>
              <button className="share-button" onClick={handleShareClick}>
                Share
              </button>
              {showShareURL && (
                <div className="share-url">
                  <input
                    type="text"
                    value={`${window.location.origin}/post/${id}`}
                    readOnly
                  />
                  <button onClick={handleCopyURL}>Copy</button>
                </div>
              )}
            </div>
          </div>
        </div>
      );
    }
```

- PostDetails.jsx:

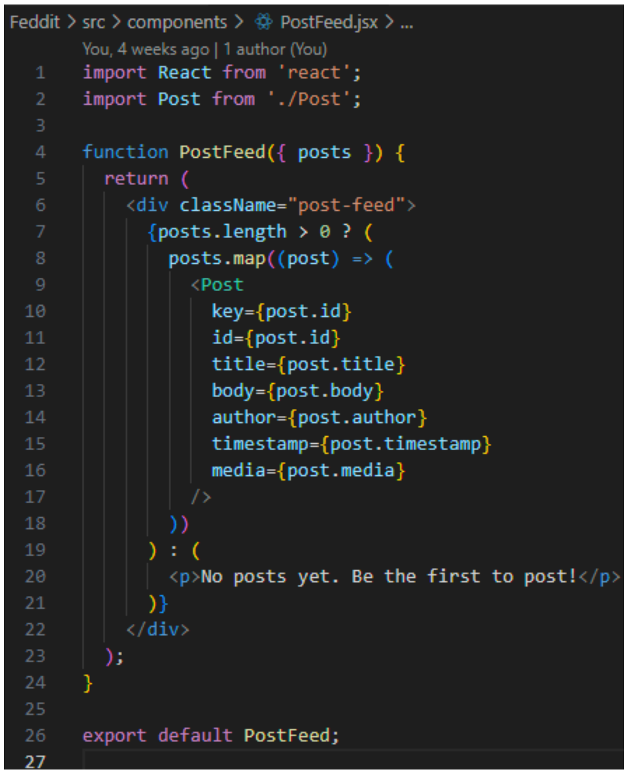


Feddit > src > components > PostDetails.jsx > PostDetails

You, 4 weeks ago | 1 author (You)

```
import React from 'react';
import '../styles/PostDetails.css';

function PostDetails({ title, content, comments }) {
  return (
    <div className="post-details">
      <h2>{title}</h2>
      <p>{content}</p>
      <h3 className="comments-title">Comments</h3>
      <div className="comments-section">
        {comments && comments.length > 0 ? (
          comments.map((comment, index) => (
            <div key={index} className="comment">
              {comment}
            </div>
          ))
        ) : (       You, 4 weeks ago • Initial commit
          <p>No comments yet. Be the first to comment!</p>
        )}
      </div>
    </div>
  );
}

export default PostDetails;
```

- PostFeed.jsx:

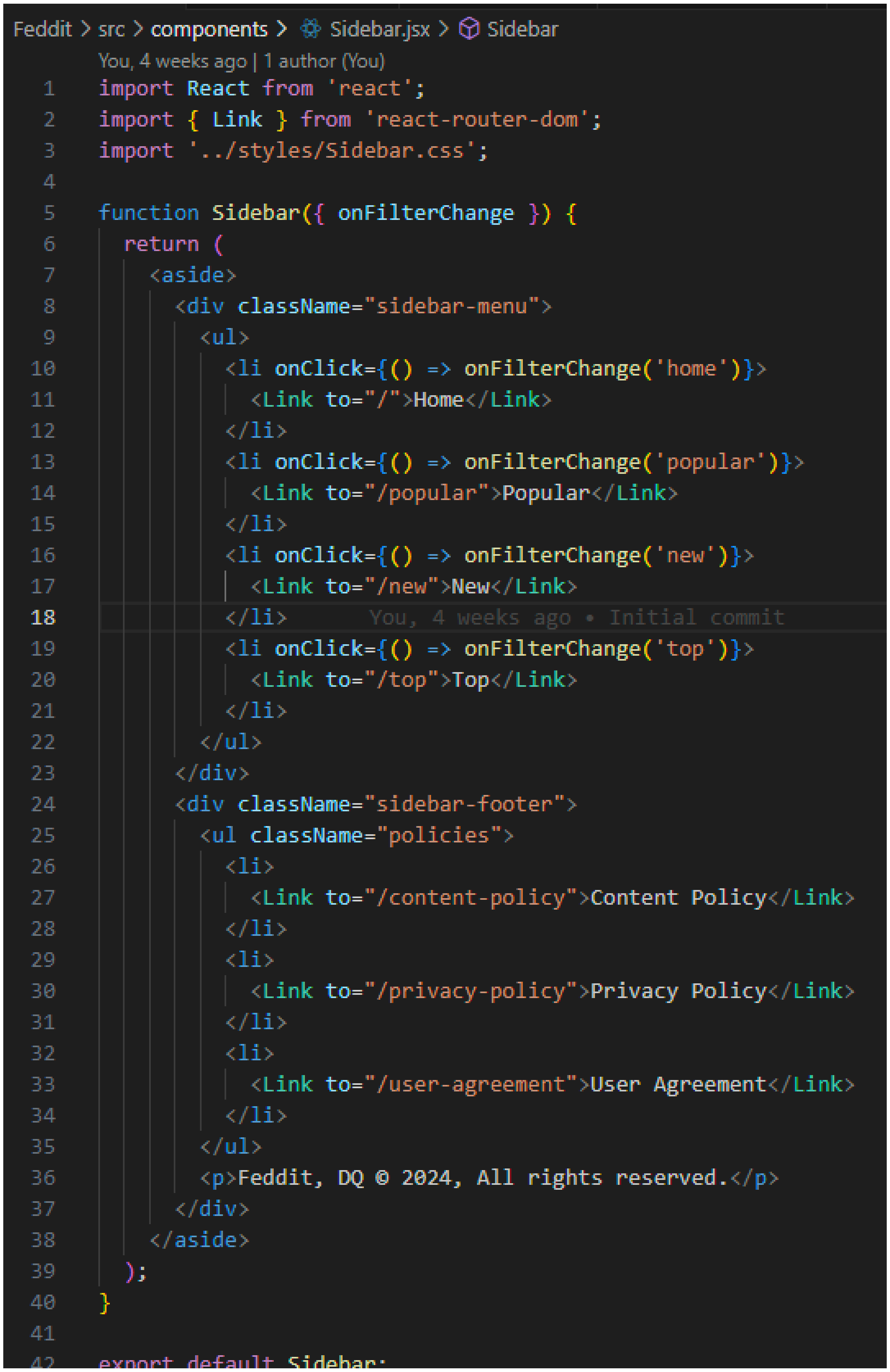


Feddit > src > components > PostFeed.jsx > ...

You, 4 weeks ago | 1 author (You)

```jsx
import React from 'react';
import Post from './Post';

function PostFeed({ posts }) {
  return (
    <div className="post-feed">
      {posts.length > 0 ? (
        posts.map((post) => (
          <Post
            key={post.id}
            id={post.id}
            title={post.title}
            body={post.body}
            author={post.author}
            timestamp={post.timestamp}
            media={post.media}
          />
        ))
      ) : (
        <p>No posts yet. Be the first to post!</p>
      )}
    </div>
  );
}

export default PostFeed;
```

- Sidebar.jsx:

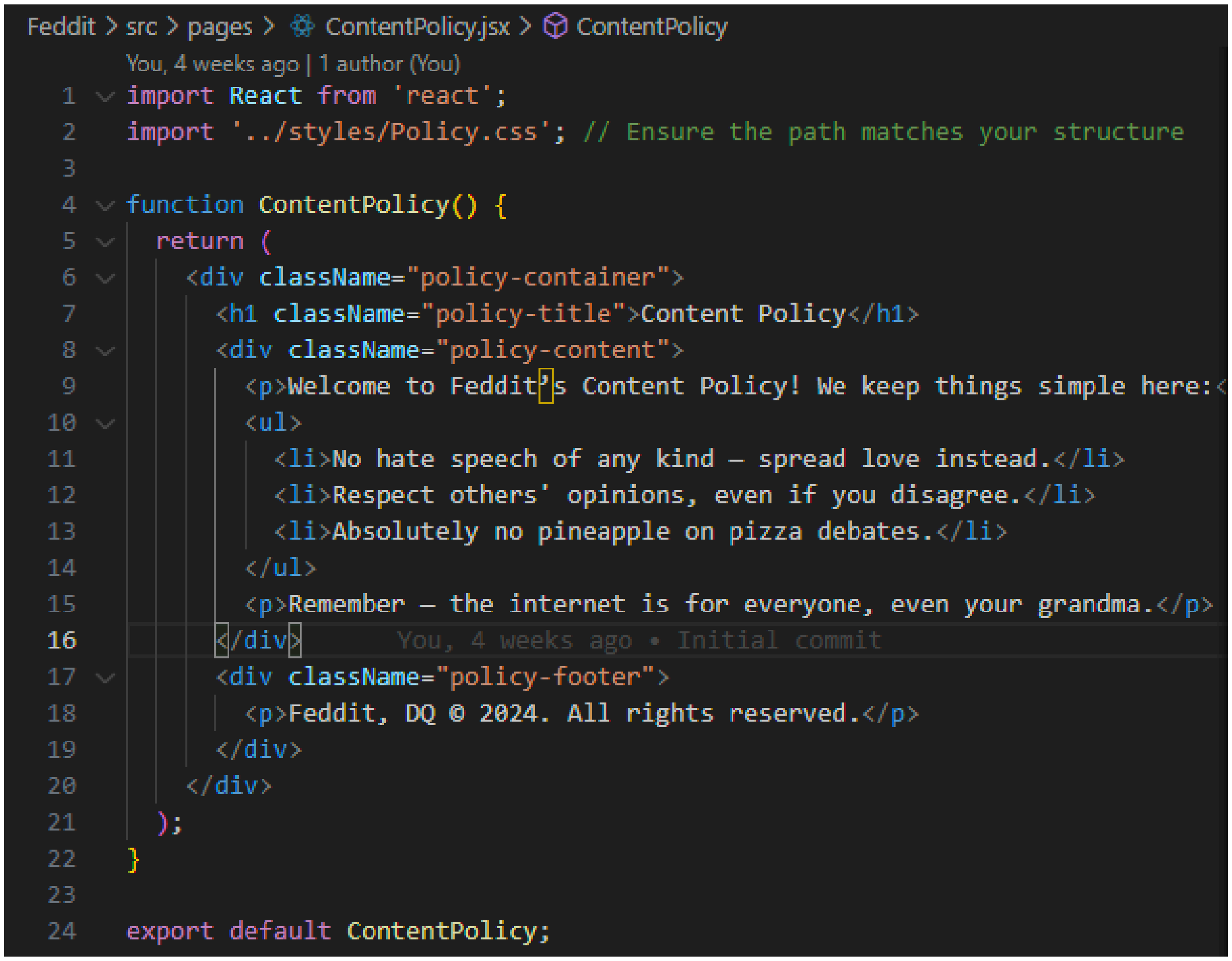


Feddit > src > components > Sidebar.jsx > Sidebar

You, 4 weeks ago | 1 author (You)

```jsx
import React from 'react';
import { Link } from 'react-router-dom';
import '../styles/Sidebar.css';

function Sidebar({ onFilterChange }) {
  return (
    <aside>
      <div className="sidebar-menu">
        <ul>
          <li onClick={() => onFilterChange('home')}>
            <Link to="/">Home</Link>
          </li>
          <li onClick={() => onFilterChange('popular')}>
            <Link to="/popular">Popular</Link>
          </li>
          <li onClick={() => onFilterChange('new')}>
            <Link to="/new">New</Link>
          </li>        You, 4 weeks ago • Initial commit
          <li onClick={() => onFilterChange('top')}>
            <Link to="/top">Top</Link>
          </li>
        </ul>
      </div>
      <div className="sidebar-footer">
        <ul className="policies">
          <li>
            <Link to="/content-policy">Content Policy</Link>
          </li>
          <li>
            <Link to="/privacy-policy">Privacy Policy</Link>
          </li>
          <li>
            <Link to="/user-agreement">User Agreement</Link>
          </li>
        </ul>
        <p>Feddit, DQ © 2024, All rights reserved.</p>
      </div>
    </aside>
  );
}

export default Sidebar;
```

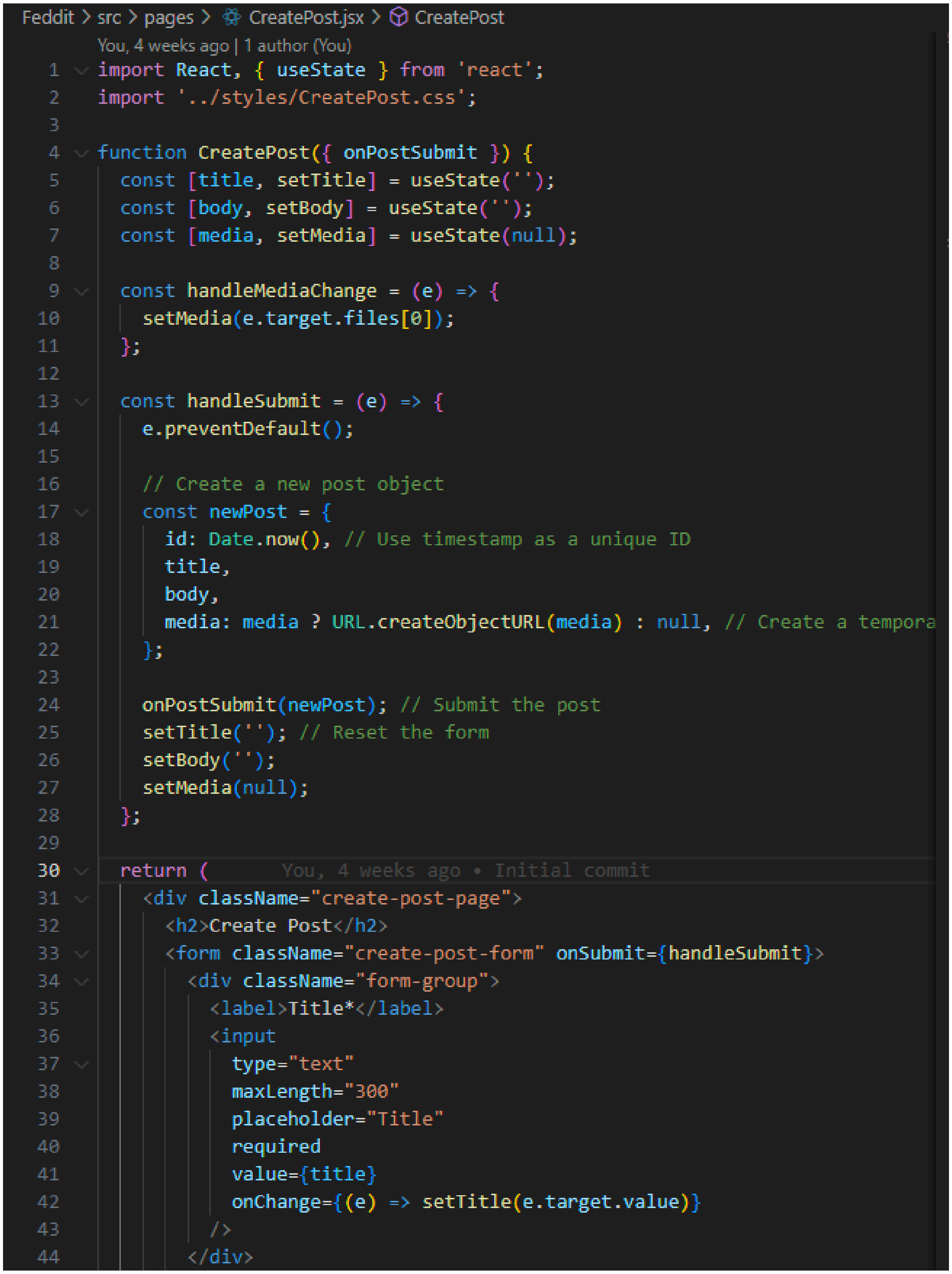
- **Pages:**
    - **ContentPolicy.jsx:**



Feddit > src > pages > ContentPolicy.jsx > ContentPolicy
You, 4 weeks ago | 1 author (You)

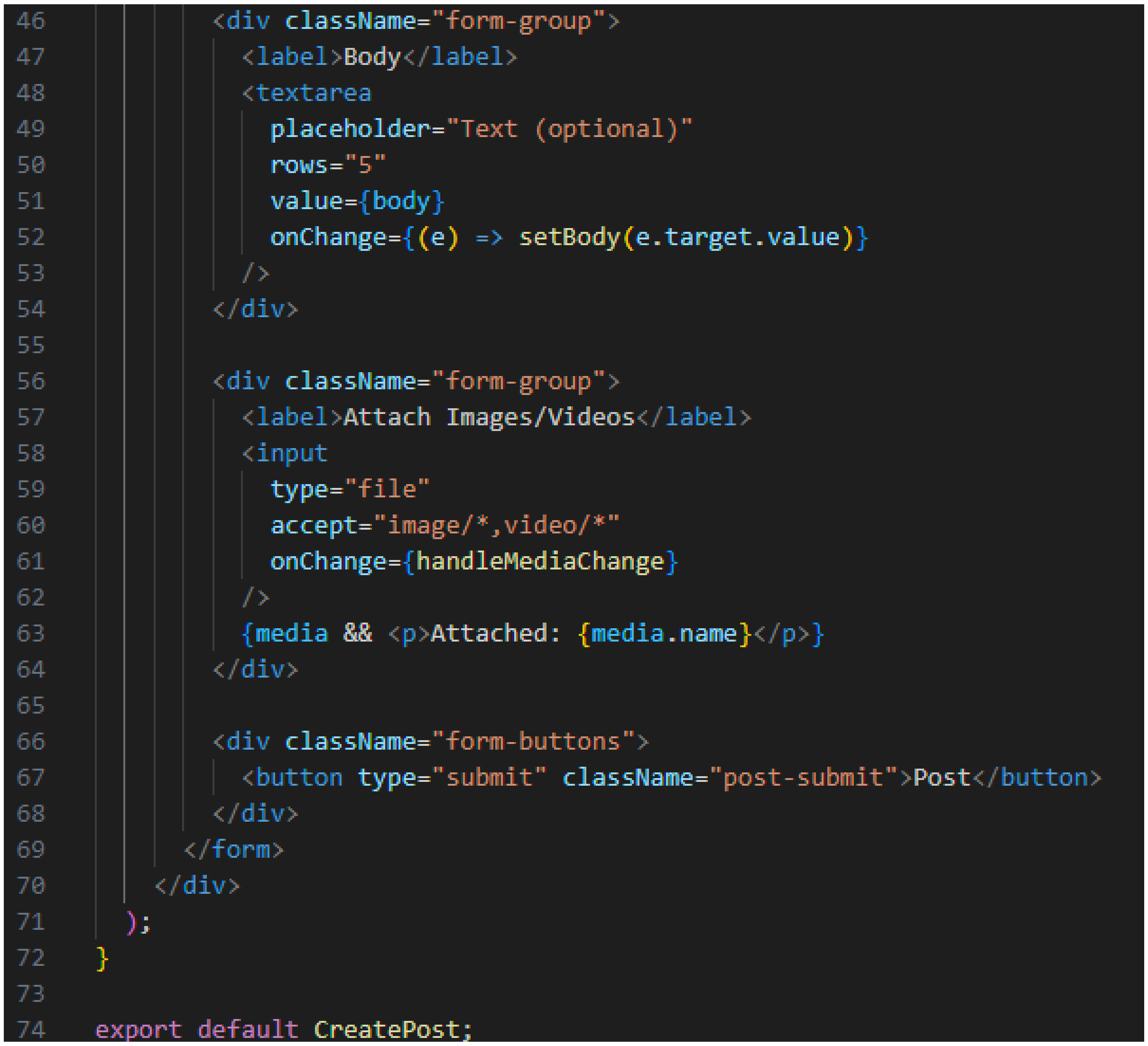
```jsx
import React from 'react';
import '../styles/Policy.css'; // Ensure the path matches your structure

function ContentPolicy() {
  return (
    <div className="policy-container">
      <h1 className="policy-title">Content Policy</h1>
      <div className="policy-content">
        <p>Welcome to Feddit's Content Policy! We keep things simple here:<
        <ul>
          <li>No hate speech of any kind – spread love instead.</li>
          <li>Respect others' opinions, even if you disagree.</li>
          <li>Absolutely no pineapple on pizza debates.</li>
        </ul>
        <p>Remember – the internet is for everyone, even your grandma.</p>
      </div>        You, 4 weeks ago • Initial commit
      <div className="policy-footer">
        <p>Feddit, DQ © 2024. All rights reserved.</p>
      </div>
    </div>
  );
}

export default ContentPolicy;
```

- **CreatePost.jsx:**


Feddit > src > pages > CreatePost.jsx > CreatePost

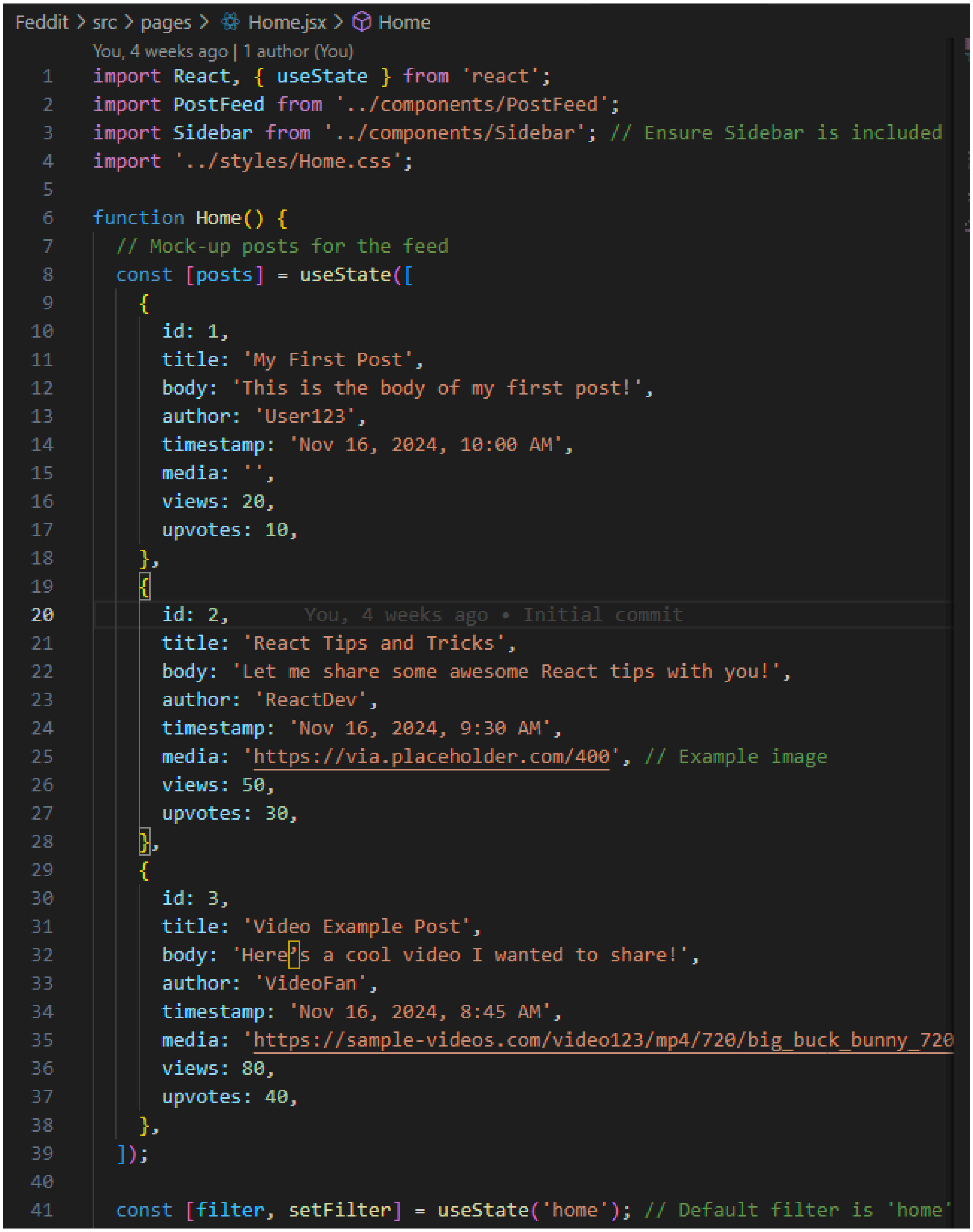
You, 4 weeks ago | 1 author (You)

```jsx
import React, { useState } from 'react';
import '../styles/CreatePost.css';

function CreatePost({ onPostSubmit }) {
  const [title, setTitle] = useState('');
  const [body, setBody] = useState('');
  const [media, setMedia] = useState(null);

  const handleMediaChange = (e) => {
    setMedia(e.target.files[0]);
  };

  const handleSubmit = (e) => {
    e.preventDefault();

    // Create a new post object
    const newPost = {
      id: Date.now(), // Use timestamp as a unique ID
      title,
      body,
      media: media ? URL.createObjectURL(media) : null, // Create a tempora
    };

    onPostSubmit(newPost); // Submit the post
    setTitle(''); // Reset the form
    setBody('');
    setMedia(null);
  };

  return (        You, 4 weeks ago • Initial commit
    <div className="create-post-page">
      <h2>Create Post</h2>
      <form className="create-post-form" onSubmit={handleSubmit}>
        <div className="form-group">
          <label>Title*</label>
          <input
            type="text"
            maxLength="300"
            placeholder="Title"
            required
            value={title}
            onChange={(e) => setTitle(e.target.value)}
          />
        </div>
```



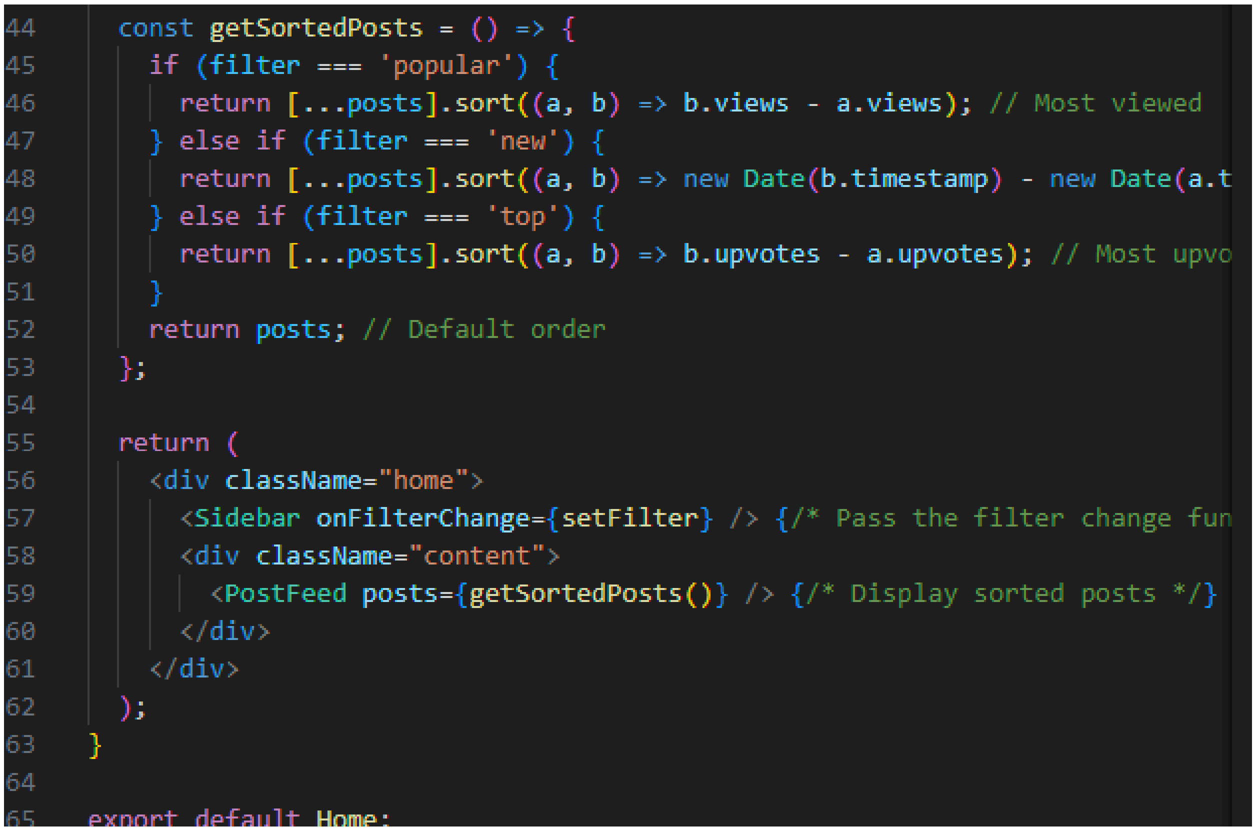
```
          <div className="form-group">
            <label>Body</label>
            <textarea
              placeholder="Text (optional)"
              rows="5"
              value={body}
              onChange={(e) => setBody(e.target.value)}
            />
          </div>

          <div className="form-group">
            <label>Attach Images/Videos</label>
            <input
              type="file"
              accept="image/*,video/*"
              onChange={handleMediaChange}
            />
            {media && <p>Attached: {media.name}</p>}
          </div>

          <div className="form-buttons">
            <button type="submit" className="post-submit">Post</button>
          </div>
        </form>
      </div>
  );
}

export default CreatePost;
```

- **Home.jsx:**


Feddit > src > pages > Home.jsx > Home

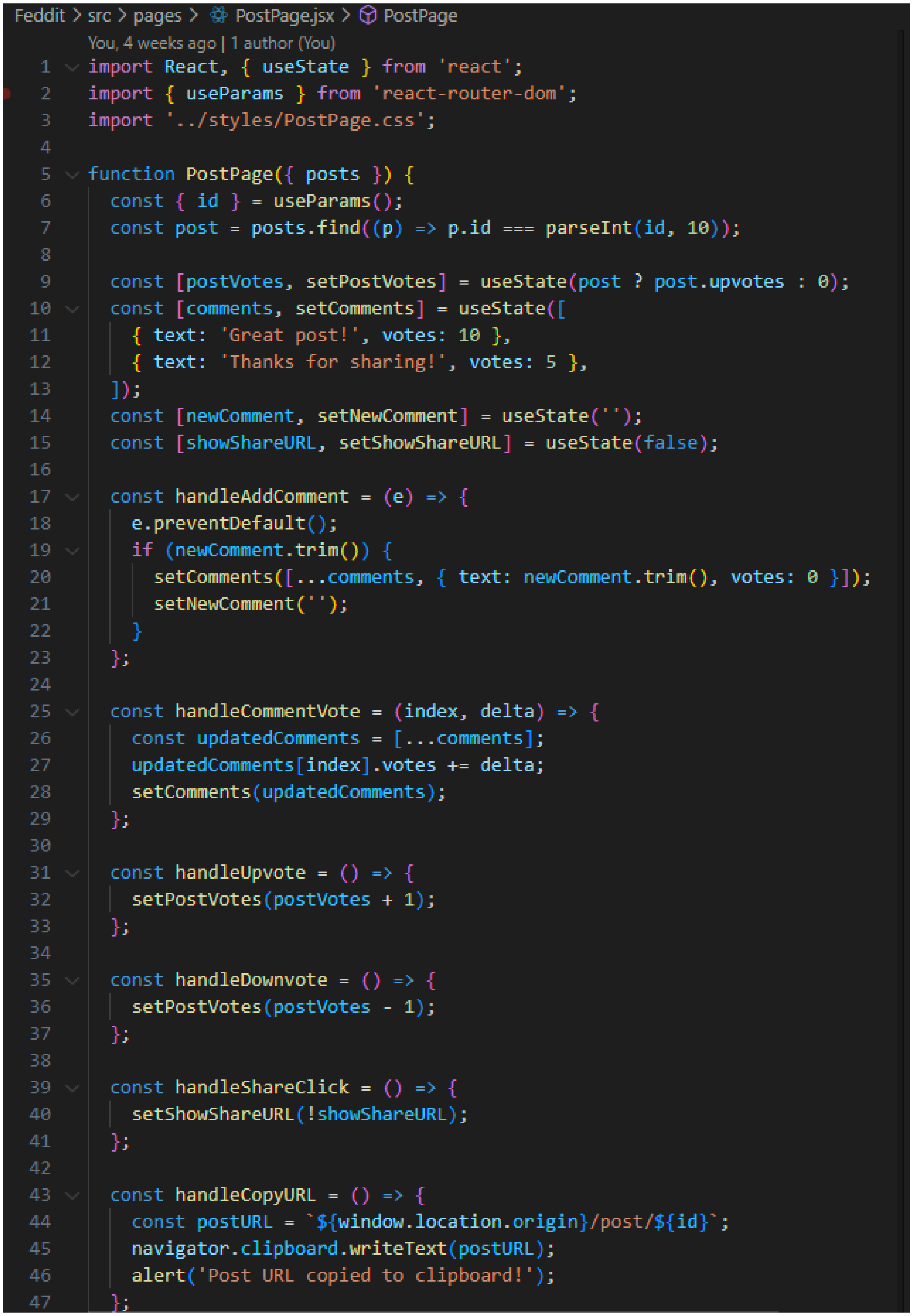
You, 4 weeks ago | 1 author (You)

```jsx
import React, { useState } from 'react';
import PostFeed from '../components/PostFeed';
import Sidebar from '../components/Sidebar'; // Ensure Sidebar is included
import '../styles/Home.css';

function Home() {
  // Mock-up posts for the feed
  const [posts] = useState([
    {
      id: 1,
      title: 'My First Post',
      body: 'This is the body of my first post!',
      author: 'User123',
      timestamp: 'Nov 16, 2024, 10:00 AM',
      media: '',
      views: 20,
      upvotes: 10,
    },
    {
      id: 2,        You, 4 weeks ago • Initial commit
      title: 'React Tips and Tricks',
      body: 'Let me share some awesome React tips with you!',
      author: 'ReactDev',
      timestamp: 'Nov 16, 2024, 9:30 AM',
      media: 'https://via.placeholder.com/400', // Example image
      views: 50,
      upvotes: 30,
    },
    {
      id: 3,
      title: 'Video Example Post',
      body: 'Here's a cool video I wanted to share!',
      author: 'VideoFan',
      timestamp: 'Nov 16, 2024, 8:45 AM',
      media: 'https://sample-videos.com/video123/mp4/720/big_buck_bunny_720
      views: 80,
      upvotes: 40,
    },
  ]);

  const [filter, setFilter] = useState('home'); // Default filter is 'home'
```



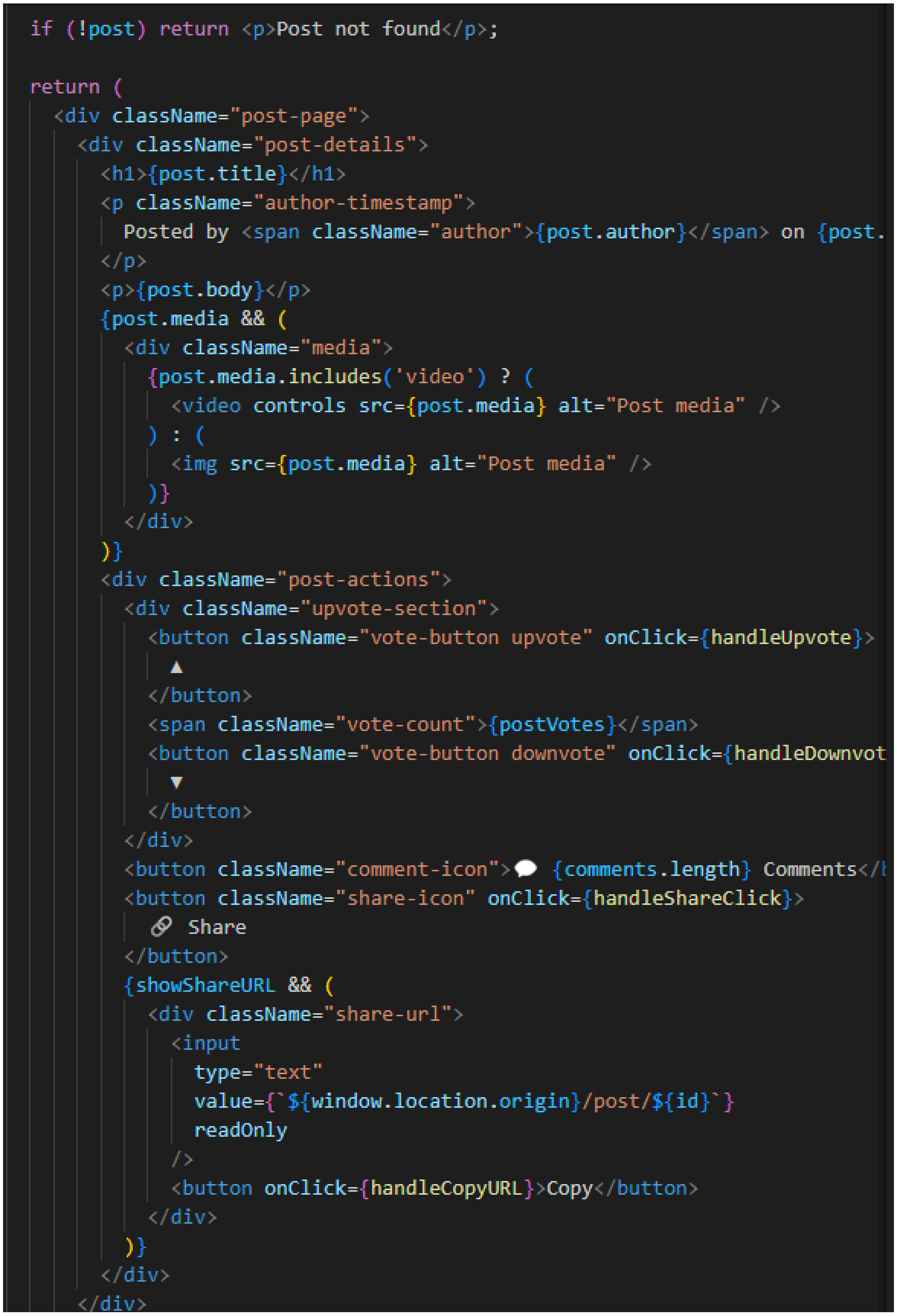
```
  const getSortedPosts = () => {
    if (filter === 'popular') {
      return [...posts].sort((a, b) => b.views - a.views); // Most viewed
    } else if (filter === 'new') {
      return [...posts].sort((a, b) => new Date(b.timestamp) - new Date(a.t
    } else if (filter === 'top') {
      return [...posts].sort((a, b) => b.upvotes - a.upvotes); // Most upvo
    }
    return posts; // Default order
  };

  return (
    <div className="home">
      <Sidebar onFilterChange={setFilter} /> {/* Pass the filter change fun
      <div className="content">
        <PostFeed posts={getSortedPosts()} /> {/* Display sorted posts */}
      </div>
    </div>
  );
}

export default Home;
```

- **PostPage.jsx:**

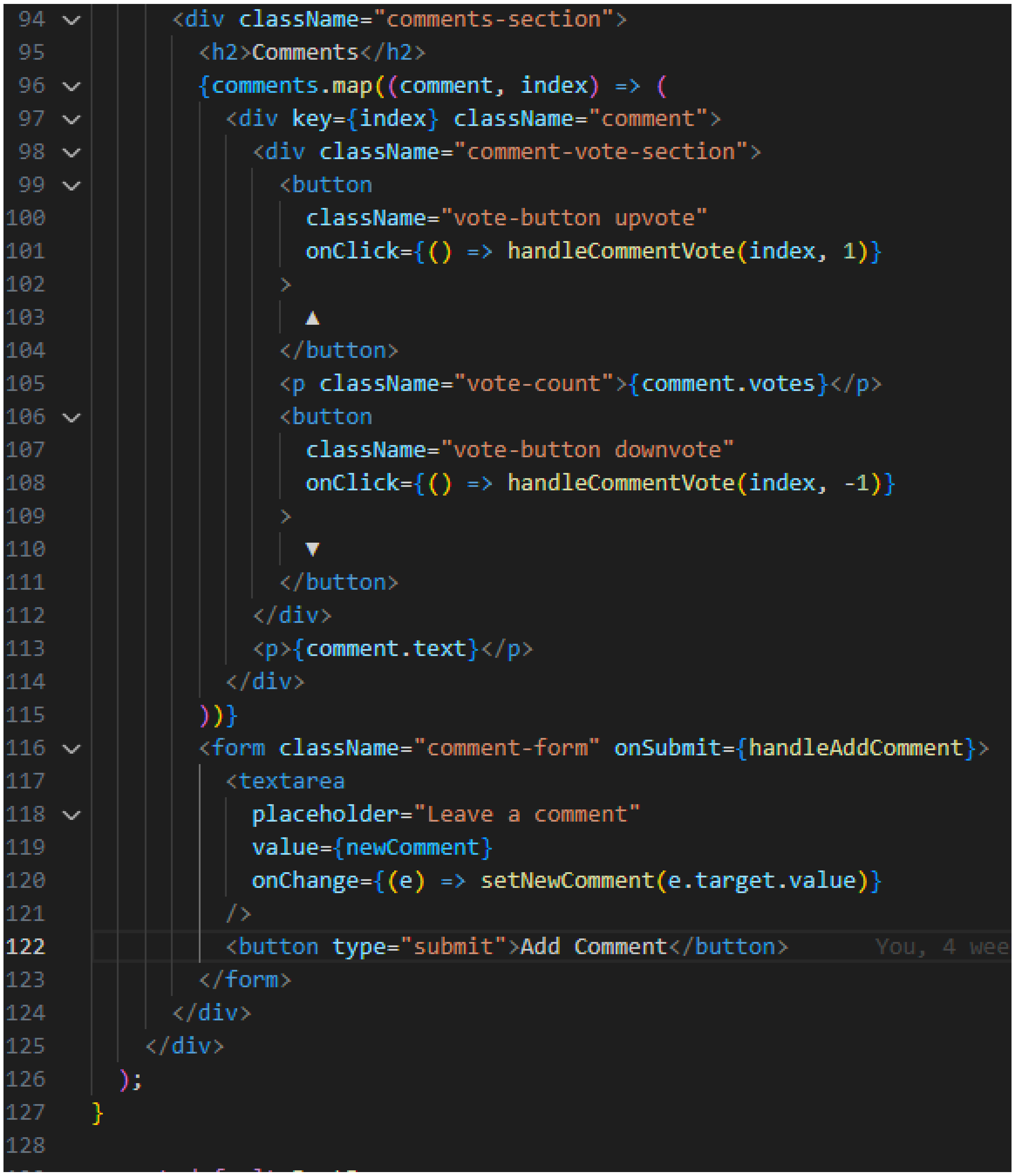

Feddit > src > pages > PostPage.jsx > PostPage

You, 4 weeks ago | 1 author (You)

```jsx
import React, { useState } from 'react';
import { useParams } from 'react-router-dom';
import '../styles/PostPage.css';

function PostPage({ posts }) {
  const { id } = useParams();
  const post = posts.find((p) => p.id === parseInt(id, 10));

  const [postVotes, setPostVotes] = useState(post ? post.upvotes : 0);
  const [comments, setComments] = useState([
    { text: 'Great post!', votes: 10 },
    { text: 'Thanks for sharing!', votes: 5 },
  ]);
  const [newComment, setNewComment] = useState('');
  const [showShareURL, setShowShareURL] = useState(false);

  const handleAddComment = (e) => {
    e.preventDefault();
    if (newComment.trim()) {
      setComments([...comments, { text: newComment.trim(), votes: 0 }]);
      setNewComment('');
    }
  };

  const handleCommentVote = (index, delta) => {
    const updatedComments = [...comments];
    updatedComments[index].votes += delta;
    setComments(updatedComments);
  };

  const handleUpvote = () => {
    setPostVotes(postVotes + 1);
  };

  const handleDownvote = () => {
    setPostVotes(postVotes - 1);
  };

  const handleShareClick = () => {
    setShowShareURL(!showShareURL);
  };

  const handleCopyURL = () => {
    const postURL = `${window.location.origin}/post/${id}`;
    navigator.clipboard.writeText(postURL);
    alert('Post URL copied to clipboard!');
  };
```

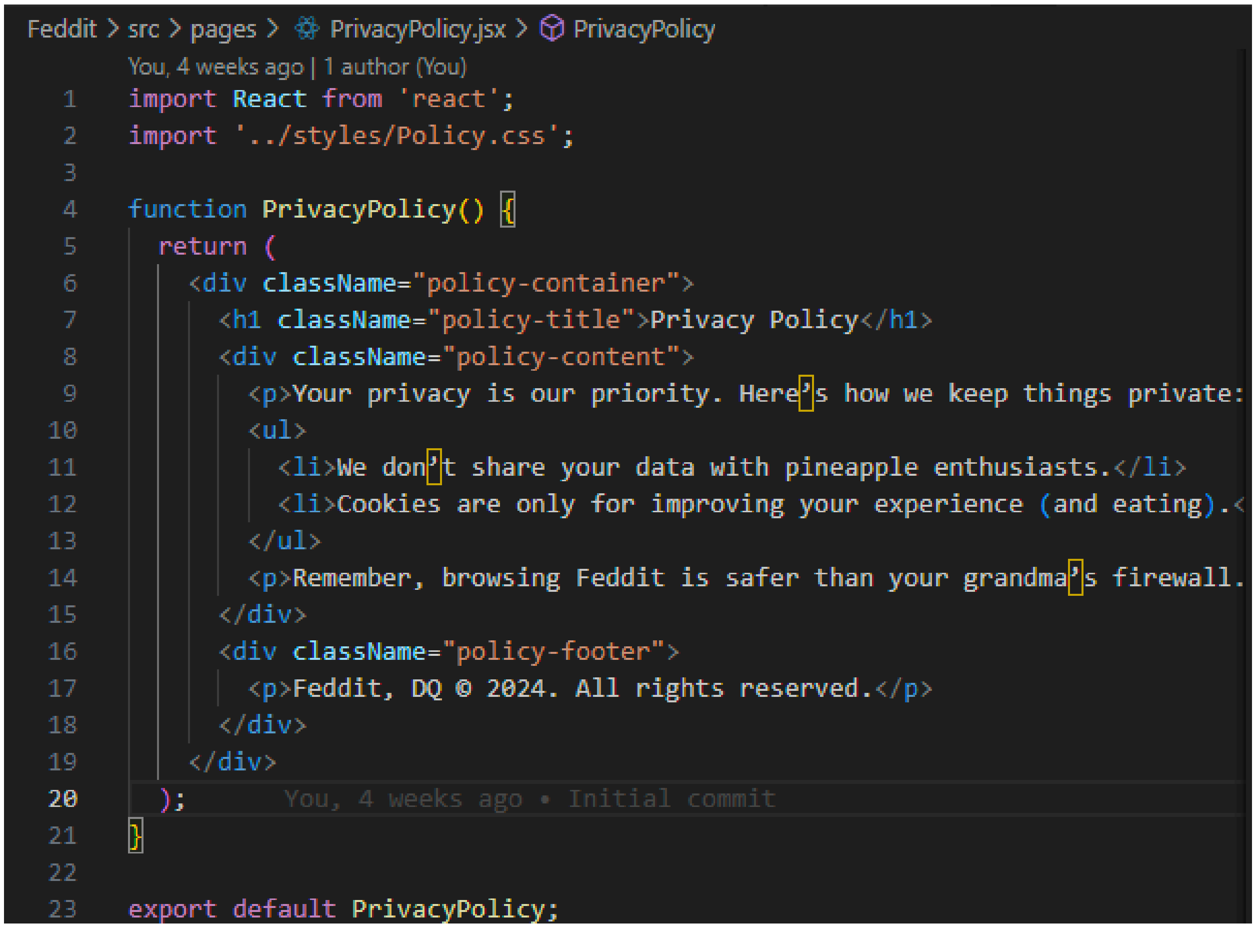


```
if (!post) return <p>Post not found</p>;

return (
  <div className="post-page">
    <div className="post-details">
      <h1>{post.title}</h1>
      <p className="author-timestamp">
        Posted by <span className="author">{post.author}</span> on {post.
      </p>
      <p>{post.body}</p>
      {post.media && (
        <div className="media">
          {post.media.includes('video') ? (
            <video controls src={post.media} alt="Post media" />
          ) : (
            <img src={post.media} alt="Post media" />
          )}
        </div>
      )}
      <div className="post-actions">
        <div className="upvote-section">
          <button className="vote-button upvote" onClick={handleUpvote}>
            ▲
          </button>
          <span className="vote-count">{postVotes}</span>
          <button className="vote-button downvote" onClick={handleDownvot
            ▼
          </button>
        </div>
        <button className="comment-icon">💬 {comments.length} Comments</l
        <button className="share-icon" onClick={handleShareClick}>
          🔗 Share
        </button>
        {showShareURL && (
          <div className="share-url">
            <input
              type="text"
              value={`${window.location.origin}/post/${id}`}
              readOnly
            />
            <button onClick={handleCopyURL}>Copy</button>
          </div>
        )}
      </div>
    </div>
```



```
        <div className="comments-section">
          <h2>Comments</h2>
          {comments.map((comment, index) => (
            <div key={index} className="comment">
              <div className="comment-vote-section">
                <button
                  className="vote-button upvote"
                  onClick={() => handleCommentVote(index, 1)}
                >
                  ▲
                </button>
                <p className="vote-count">{comment.votes}</p>
                <button
                  className="vote-button downvote"
                  onClick={() => handleCommentVote(index, -1)}
                >
                  ▼
                </button>
              </div>
              <p>{comment.text}</p>
            </div>
          ))}
          <form className="comment-form" onSubmit={handleAddComment}>
            <textarea
              placeholder="Leave a comment"
              value={newComment}
              onChange={(e) => setNewComment(e.target.value)}
            />
            <button type="submit">Add Comment</button>        You, 4 wee
          </form>
        </div>
      </div>
    );
  }

```

- **PrivacyPolicy.jsx:**

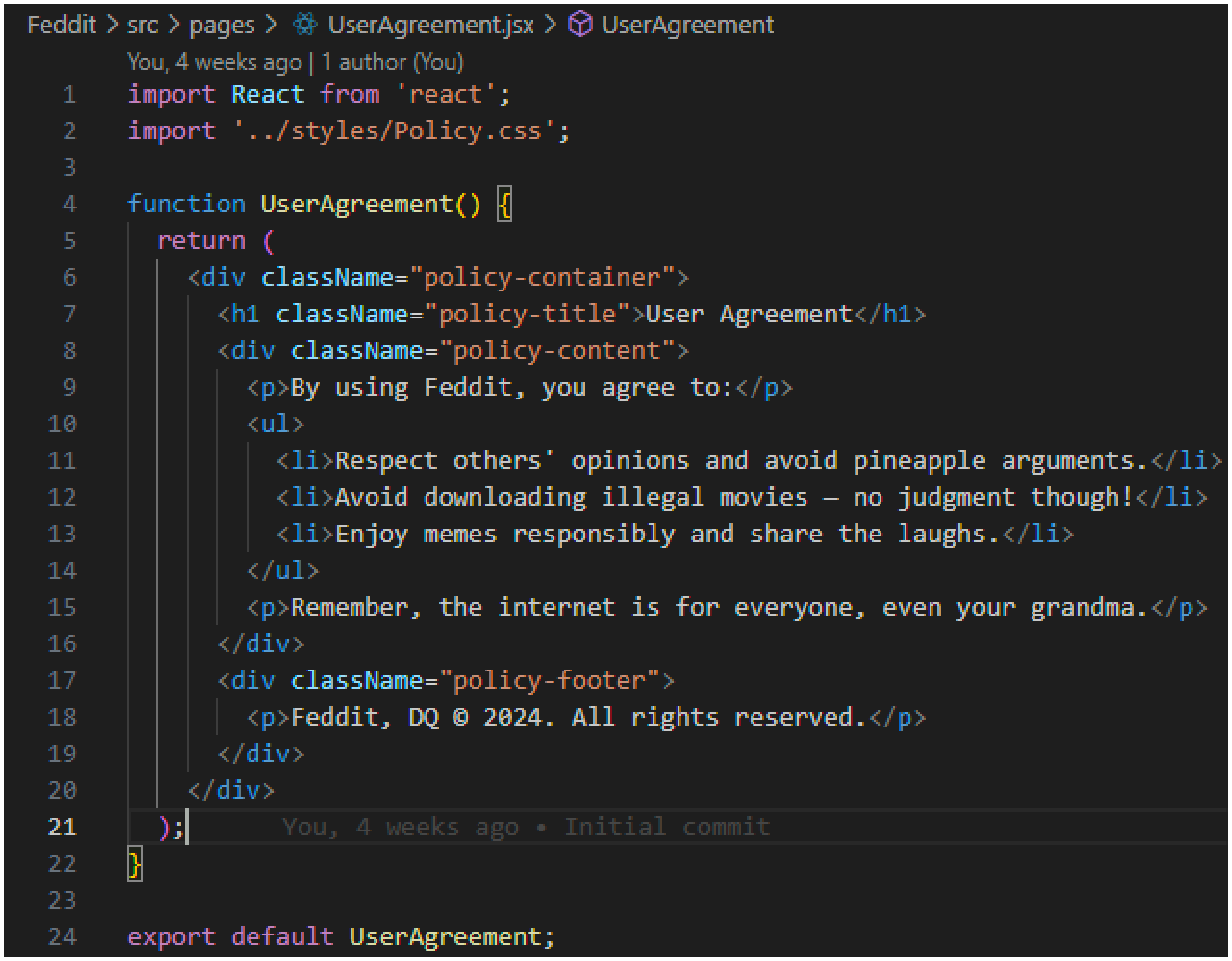


Feddit > src > pages > PrivacyPolicy.jsx > PrivacyPolicy

You, 4 weeks ago | 1 author (You)

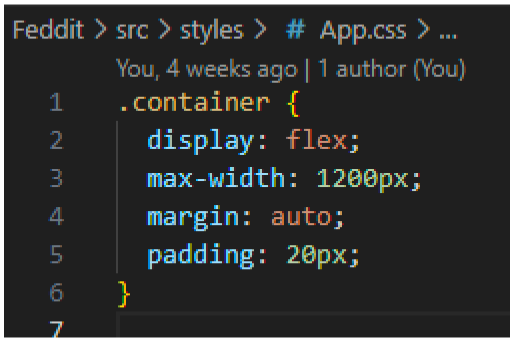
```jsx
import React from 'react';
import '../styles/Policy.css';

function PrivacyPolicy() {
  return (
    <div className="policy-container">
      <h1 className="policy-title">Privacy Policy</h1>
      <div className="policy-content">
        <p>Your privacy is our priority. Here’s how we keep things private:
        <ul>
          <li>We don’t share your data with pineapple enthusiasts.</li>
          <li>Cookies are only for improving your experience (and eating).<
        </ul>
        <p>Remember, browsing Feddit is safer than your grandma’s firewall.
      </div>
      <div className="policy-footer">
        <p>Feddit, DQ © 2024. All rights reserved.</p>
      </div>
    </div>
  );      You, 4 weeks ago • Initial commit
}

export default PrivacyPolicy;
```

- **UserAgreement.jsx:**


Feddit > src > pages > UserAgreement.jsx > UserAgreement
You, 4 weeks ago | 1 author (You)

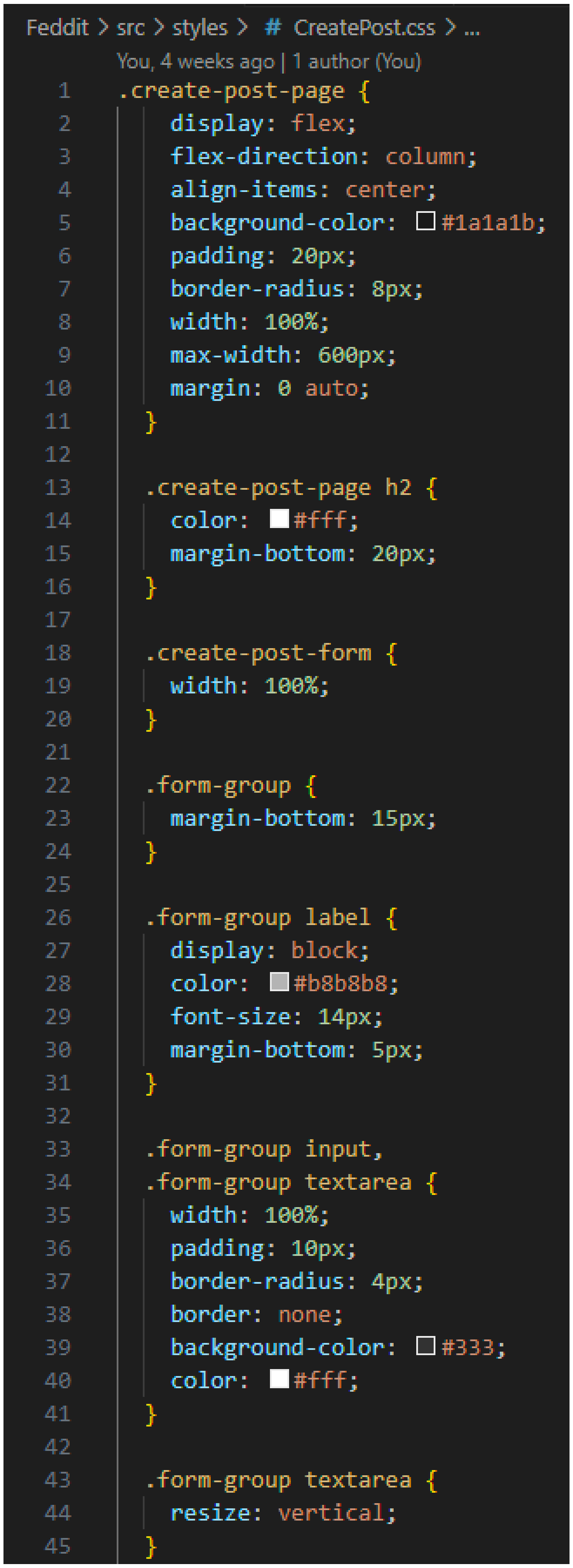
```jsx
import React from 'react';
import '../styles/Policy.css';

function UserAgreement() {
  return (
    <div className="policy-container">
      <h1 className="policy-title">User Agreement</h1>
      <div className="policy-content">
        <p>By using Feddit, you agree to:</p>
        <ul>
          <li>Respect others' opinions and avoid pineapple arguments.</li>
          <li>Avoid downloading illegal movies — no judgment though!</li>
          <li>Enjoy memes responsibly and share the laughs.</li>
        </ul>
        <p>Remember, the internet is for everyone, even your grandma.</p>
      </div>
      <div className="policy-footer">
        <p>Feddit, DQ © 2024. All rights reserved.</p>
      </div>
    </div>
  );        You, 4 weeks ago • Initial commit
}

export default UserAgreement;
```

- **Styles:**
  - **App.css:**


Feddit > src > styles > # App.css > ...
You, 4 weeks ago | 1 author (You)

```css
.container {
  display: flex;
  max-width: 1200px;
  margin: auto;
  padding: 20px;
}
```

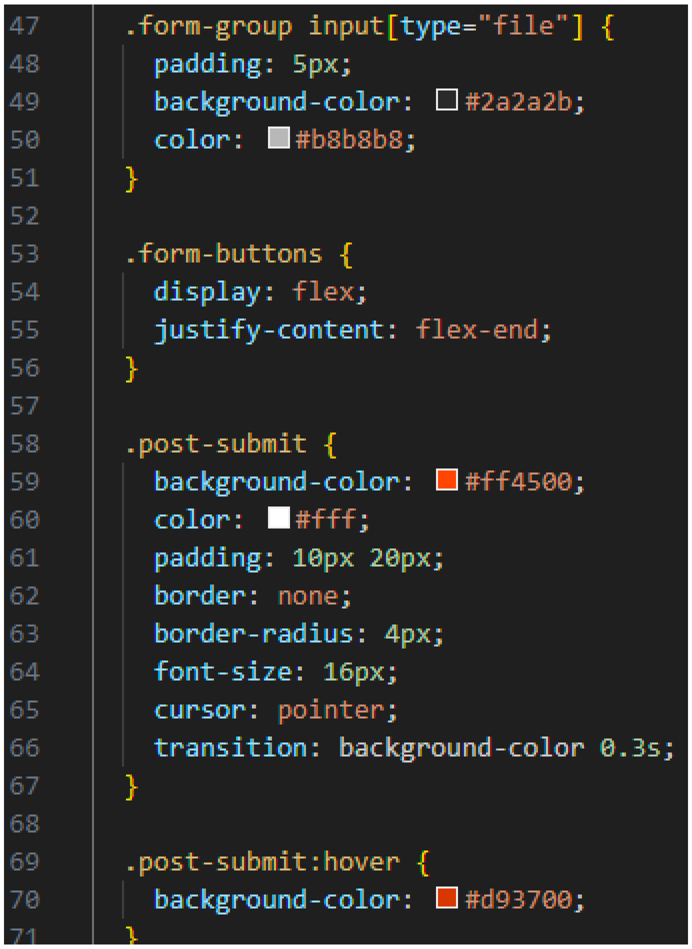
- **CreatePost.css:**



Feddit > src > styles > # CreatePost.css > ...

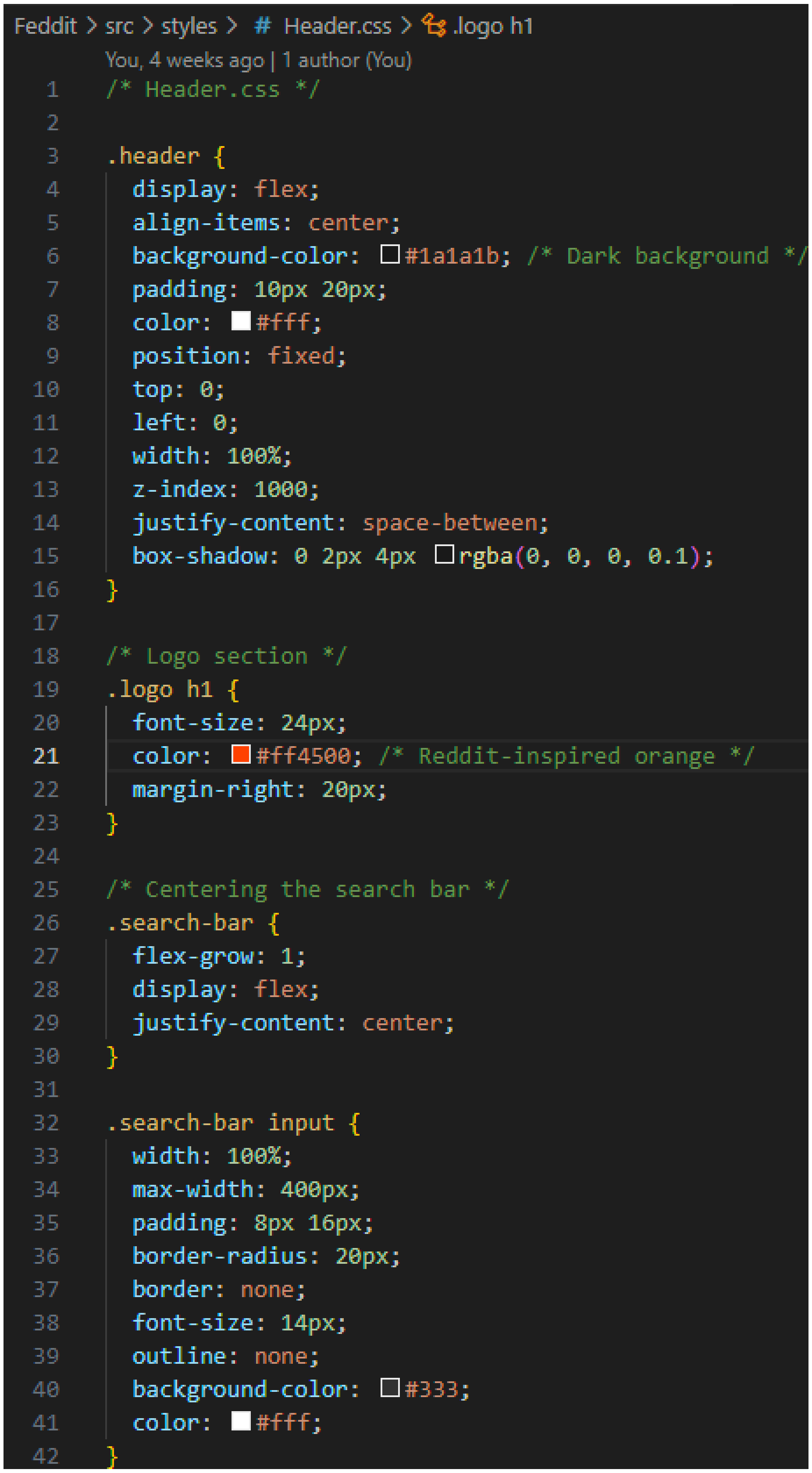
You, 4 weeks ago | 1 author (You)

```css
.create-post-page {
    display: flex;
    flex-direction: column;
    align-items: center;
    background-color: #1a1a1b;
    padding: 20px;
    border-radius: 8px;
    width: 100%;
    max-width: 600px;
    margin: 0 auto;
  }

  .create-post-page h2 {
    color: #fff;
    margin-bottom: 20px;
  }

  .create-post-form {
    width: 100%;
  }

  .form-group {
    margin-bottom: 15px;
  }

  .form-group label {
    display: block;
    color: #b8b8b8;
    font-size: 14px;
    margin-bottom: 5px;
  }

  .form-group input,
  .form-group textarea {
    width: 100%;
    padding: 10px;
    border-radius: 4px;
    border: none;
    background-color: #333;
    color: #fff;
  }

  .form-group textarea {
    resize: vertical;
  }
```

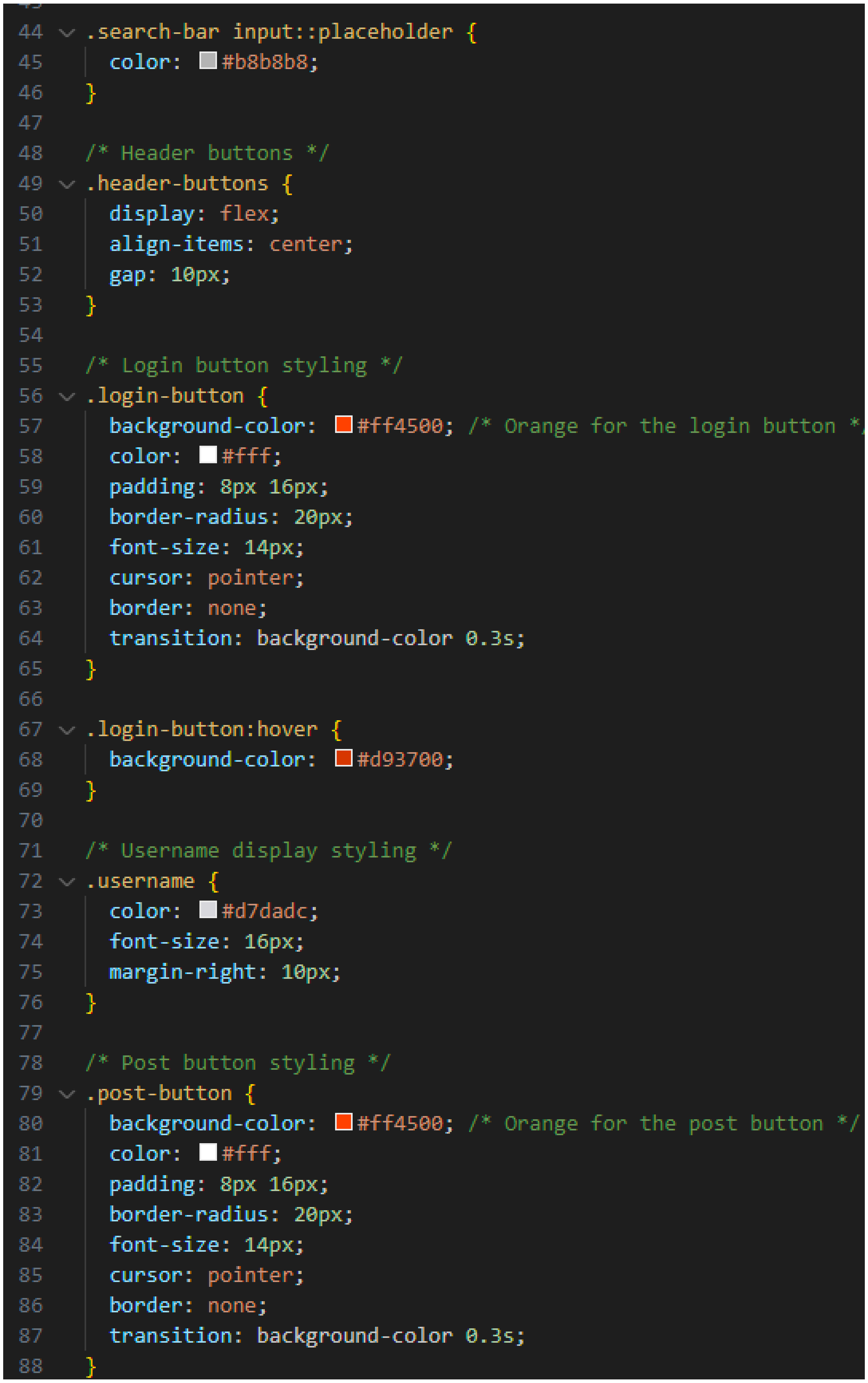


```
.form-group input[type="file"] {
  padding: 5px;
  background-color: #2a2a2b;
  color: #b8b8b8;
}

.form-buttons {
  display: flex;
  justify-content: flex-end;
}

.post-submit {
  background-color: #ff4500;
  color: #fff;
  padding: 10px 20px;
  border: none;
  border-radius: 4px;
  font-size: 16px;
  cursor: pointer;
  transition: background-color 0.3s;
}

.post-submit:hover {
  background-color: #d93700;
}
```

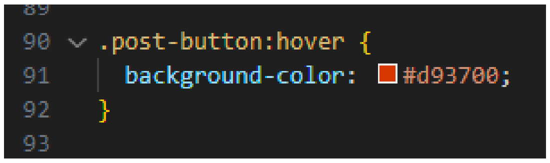
- **Header.css:**


Feddit > src > styles > # Header.css > .logo h1
You, 4 weeks ago | 1 author (You)

```css
/* Header.css */

.header {
  display: flex;
  align-items: center;
  background-color: #1a1a1b; /* Dark background */
  padding: 10px 20px;
  color: #fff;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  z-index: 1000;
  justify-content: space-between;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

/* Logo section */
.logo h1 {
  font-size: 24px;
  color: #ff4500; /* Reddit-inspired orange */
  margin-right: 20px;
}

/* Centering the search bar */
.search-bar {
  flex-grow: 1;
  display: flex;
  justify-content: center;
}

.search-bar input {
  width: 100%;
  max-width: 400px;
  padding: 8px 16px;
  border-radius: 20px;
  border: none;
  font-size: 14px;
  outline: none;
  background-color: #333;
  color: #fff;
}
```

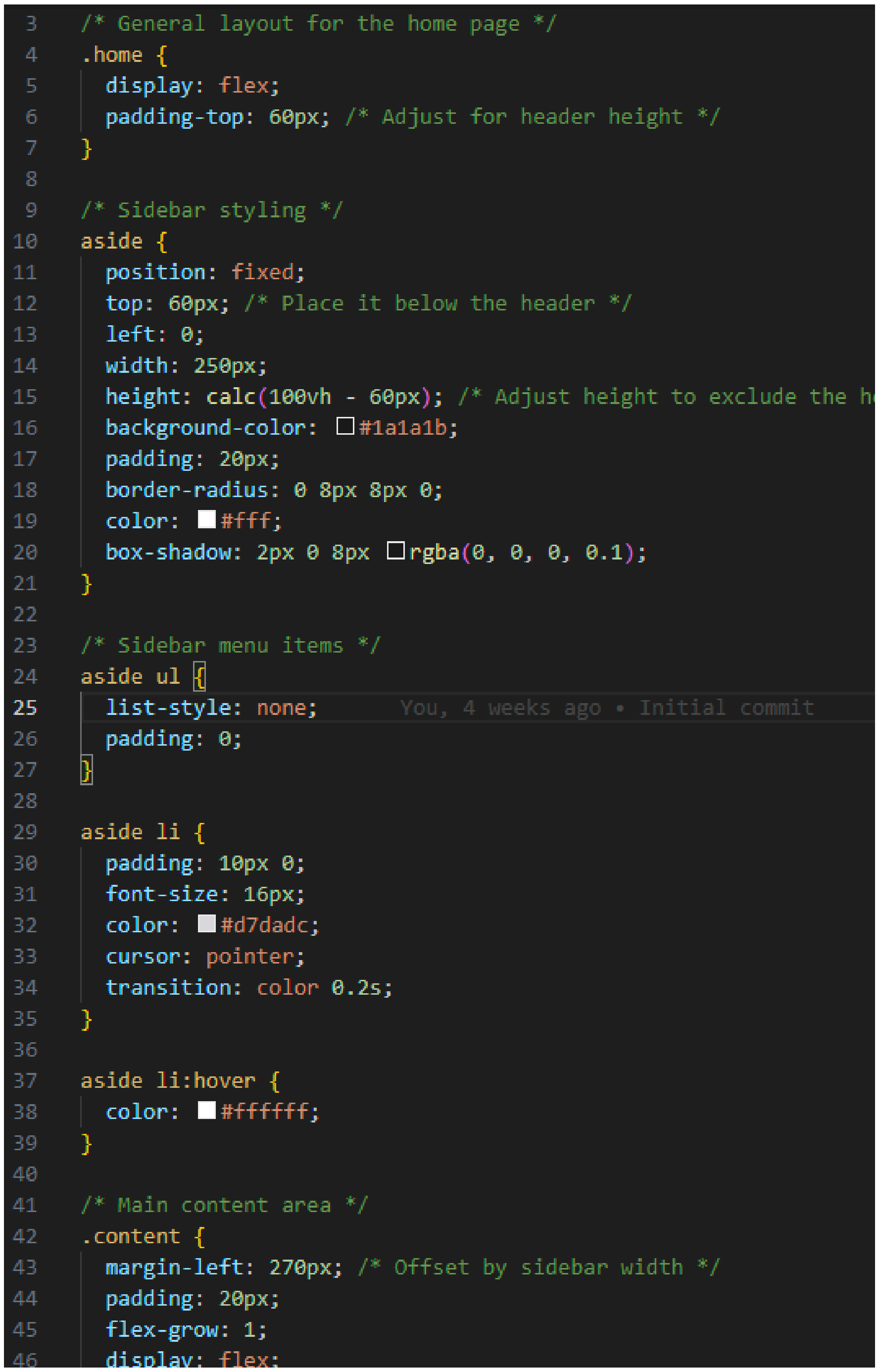


```css
.search-bar input::placeholder {
  color: #b8b8b8;
}

/* Header buttons */
.header-buttons {
  display: flex;
  align-items: center;
  gap: 10px;
}

/* Login button styling */
.login-button {
  background-color: #ff4500; /* Orange for the login button *
  color: #fff;
  padding: 8px 16px;
  border-radius: 20px;
  font-size: 14px;
  cursor: pointer;
  border: none;
  transition: background-color 0.3s;
}

.login-button:hover {
  background-color: #d93700;
}

/* Username display styling */
.username {
  color: #d7dadc;
  font-size: 16px;
  margin-right: 10px;
}

/* Post button styling */
.post-button {
  background-color: #ff4500; /* Orange for the post button */
  color: #fff;
  padding: 8px 16px;
  border-radius: 20px;
  font-size: 14px;
  cursor: pointer;
  border: none;
  transition: background-color 0.3s;
}
```



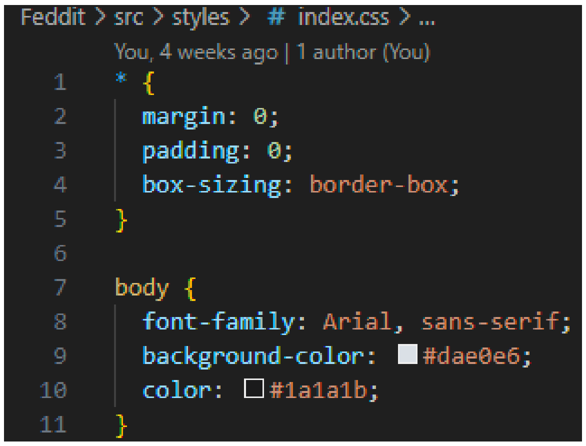
```css
.post-button:hover {
  background-color: #d93700;
}
```

- **Home.css:**



```css
/* General layout for the home page */
.home {
  display: flex;
  padding-top: 60px; /* Adjust for header height */
}

/* Sidebar styling */
aside {
  position: fixed;
  top: 60px; /* Place it below the header */
  left: 0;
  width: 250px;
  height: calc(100vh - 60px); /* Adjust height to exclude the h
  background-color: #1a1a1b;
  padding: 20px;
  border-radius: 0 8px 8px 0;
  color: #fff;
  box-shadow: 2px 0 8px rgba(0, 0, 0, 0.1);
}

/* Sidebar menu items */
aside ul {
  list-style: none;        You, 4 weeks ago • Initial commit
  padding: 0;
}

aside li {
  padding: 10px 0;
  font-size: 16px;
  color: #d7dadc;
  cursor: pointer;
  transition: color 0.2s;
}

aside li:hover {
  color: #ffffff;
}

/* Main content area */
.content {
  margin-left: 270px; /* Offset by sidebar width */
  padding: 20px;
  flex-grow: 1;
  display: flex;
```

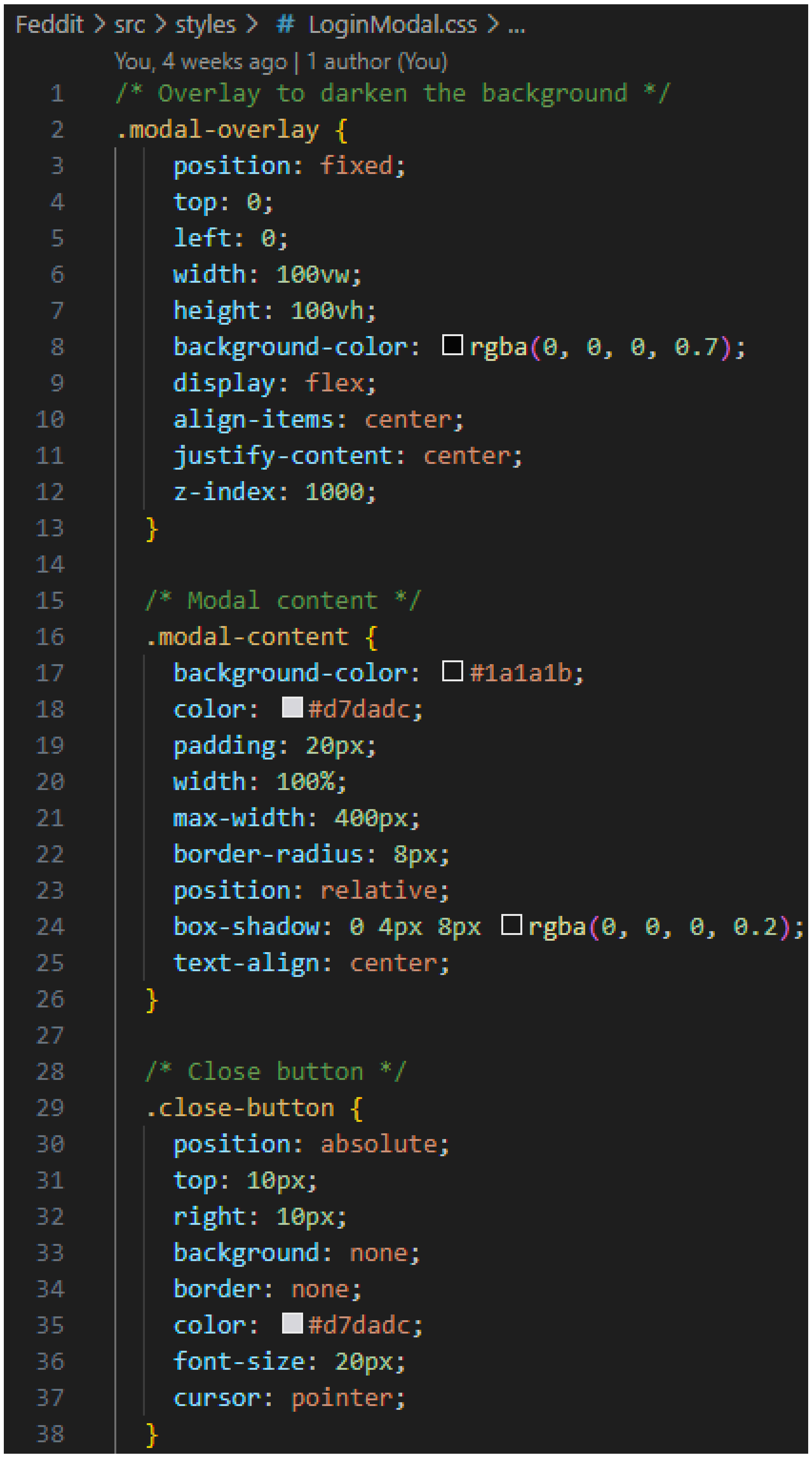
- **index.css:**


Feddit > src > styles > # index.css > ...

You, 4 weeks ago | 1 author (You)

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: Arial, sans-serif;
  background-color: #dae0e6;
  color: #1a1a1b;
}
```

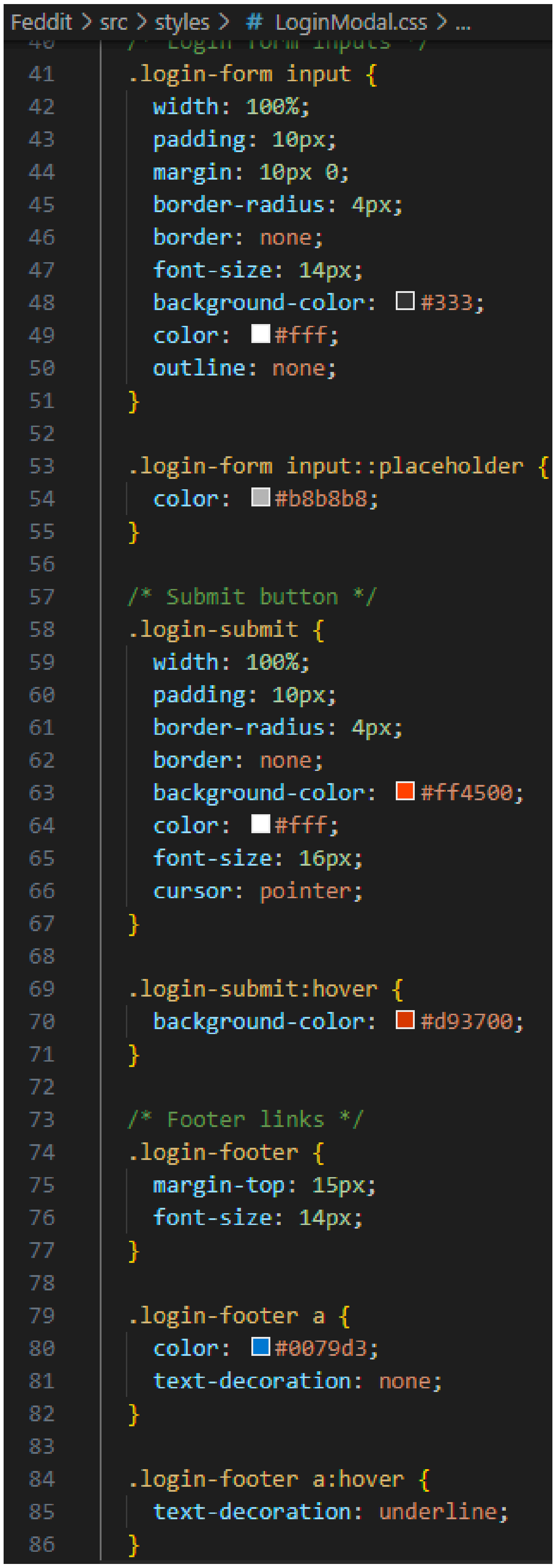
- **LoginModal.css**



Feddit > src > styles > # LoginModal.css > ...

You, 4 weeks ago | 1 author (You)

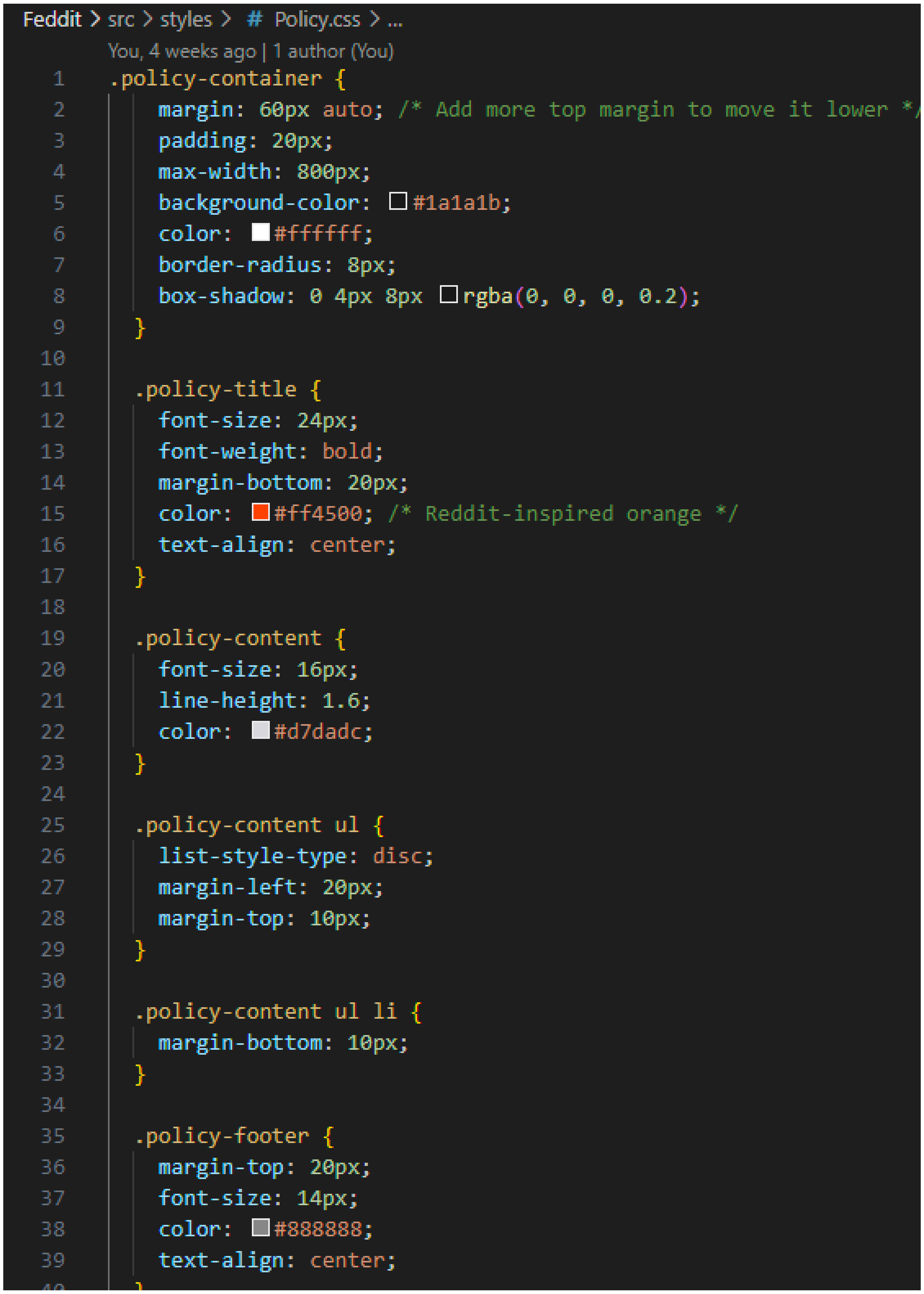
```css
/* Overlay to darken the background */
.modal-overlay {
    position: fixed;
    top: 0;
    left: 0;
    width: 100vw;
    height: 100vh;
    background-color: rgba(0, 0, 0, 0.7);
    display: flex;
    align-items: center;
    justify-content: center;
    z-index: 1000;
  }

  /* Modal content */
  .modal-content {
    background-color: #1a1a1b;
    color: #d7dadc;
    padding: 20px;
    width: 100%;
    max-width: 400px;
    border-radius: 8px;
    position: relative;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    text-align: center;
  }

  /* Close button */
  .close-button {
    position: absolute;
    top: 10px;
    right: 10px;
    background: none;
    border: none;
    color: #d7dadc;
    font-size: 20px;
    cursor: pointer;
  }
```



Feddit > src > styles > # LoginModal.css > ...

```css
/* Login form inputs */
.login-form input {
  width: 100%;
  padding: 10px;
  margin: 10px 0;
  border-radius: 4px;
  border: none;
  font-size: 14px;
  background-color: #333;
  color: #fff;
  outline: none;
}

.login-form input::placeholder {
  color: #b8b8b8;
}

/* Submit button */
.login-submit {
  width: 100%;
  padding: 10px;
  border-radius: 4px;
  border: none;
  background-color: #ff4500;
  color: #fff;
  font-size: 16px;
  cursor: pointer;
}

.login-submit:hover {
  background-color: #d93700;
}

/* Footer links */
.login-footer {
  margin-top: 15px;
  font-size: 14px;
}

.login-footer a {
  color: #0079d3;
  text-decoration: none;
}

.login-footer a:hover {
  text-decoration: underline;
}
```

- **Policy.css:**

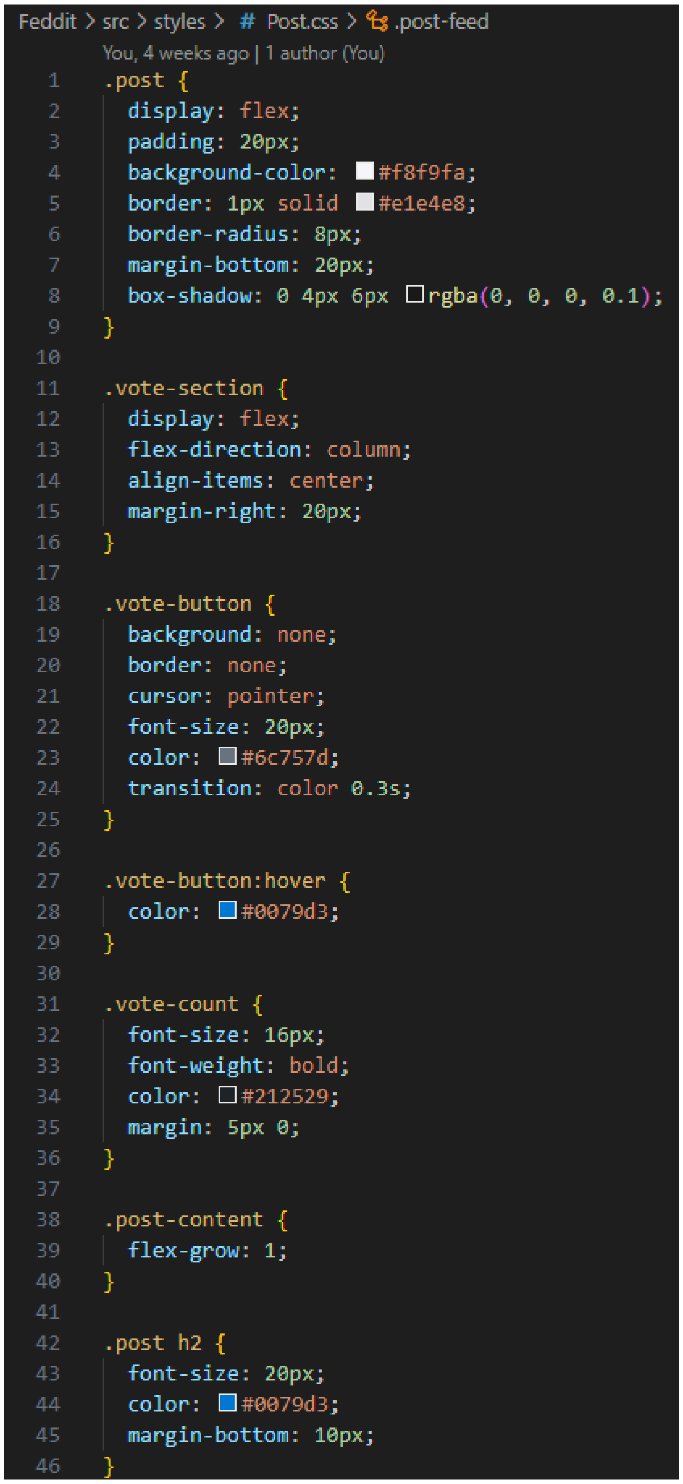


Feddit > src > styles > # Policy.css > ...

You, 4 weeks ago | 1 author (You)

```css
.policy-container {
    margin: 60px auto; /* Add more top margin to move it lower */
    padding: 20px;
    max-width: 800px;
    background-color: #1a1a1b;
    color: #ffffff;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
  }

  .policy-title {
    font-size: 24px;
    font-weight: bold;
    margin-bottom: 20px;
    color: #ff4500; /* Reddit-inspired orange */
    text-align: center;
  }

  .policy-content {
    font-size: 16px;
    line-height: 1.6;
    color: #d7dadc;
  }

  .policy-content ul {
    list-style-type: disc;
    margin-left: 20px;
    margin-top: 10px;
  }

  .policy-content ul li {
    margin-bottom: 10px;
  }

  .policy-footer {
    margin-top: 20px;
    font-size: 14px;
    color: #888888;
    text-align: center;
```

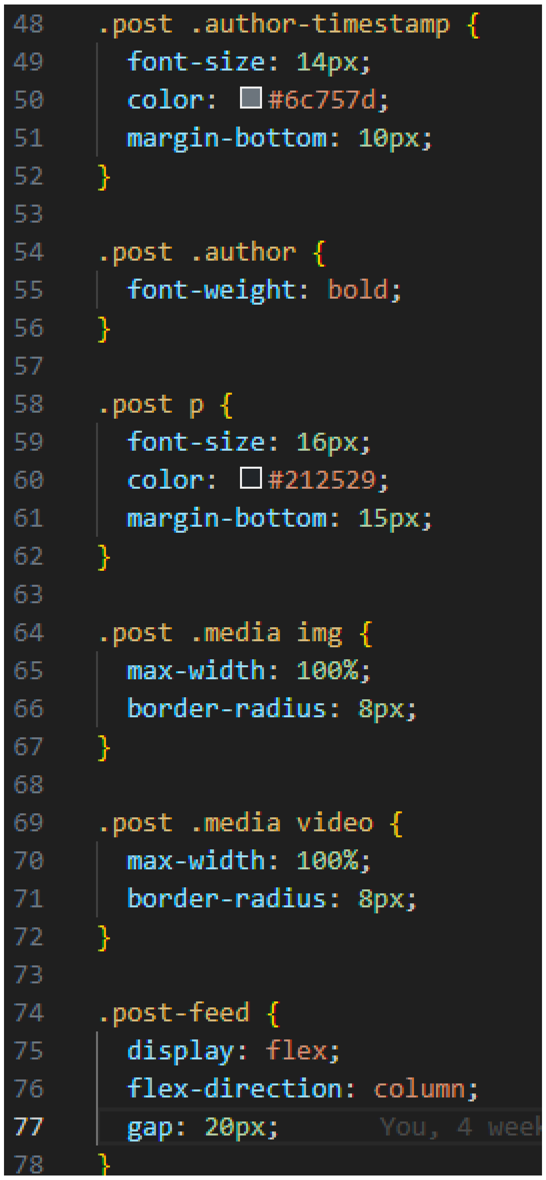
- Post.css:



Feddit > src > styles > # Post.css > .post-feed

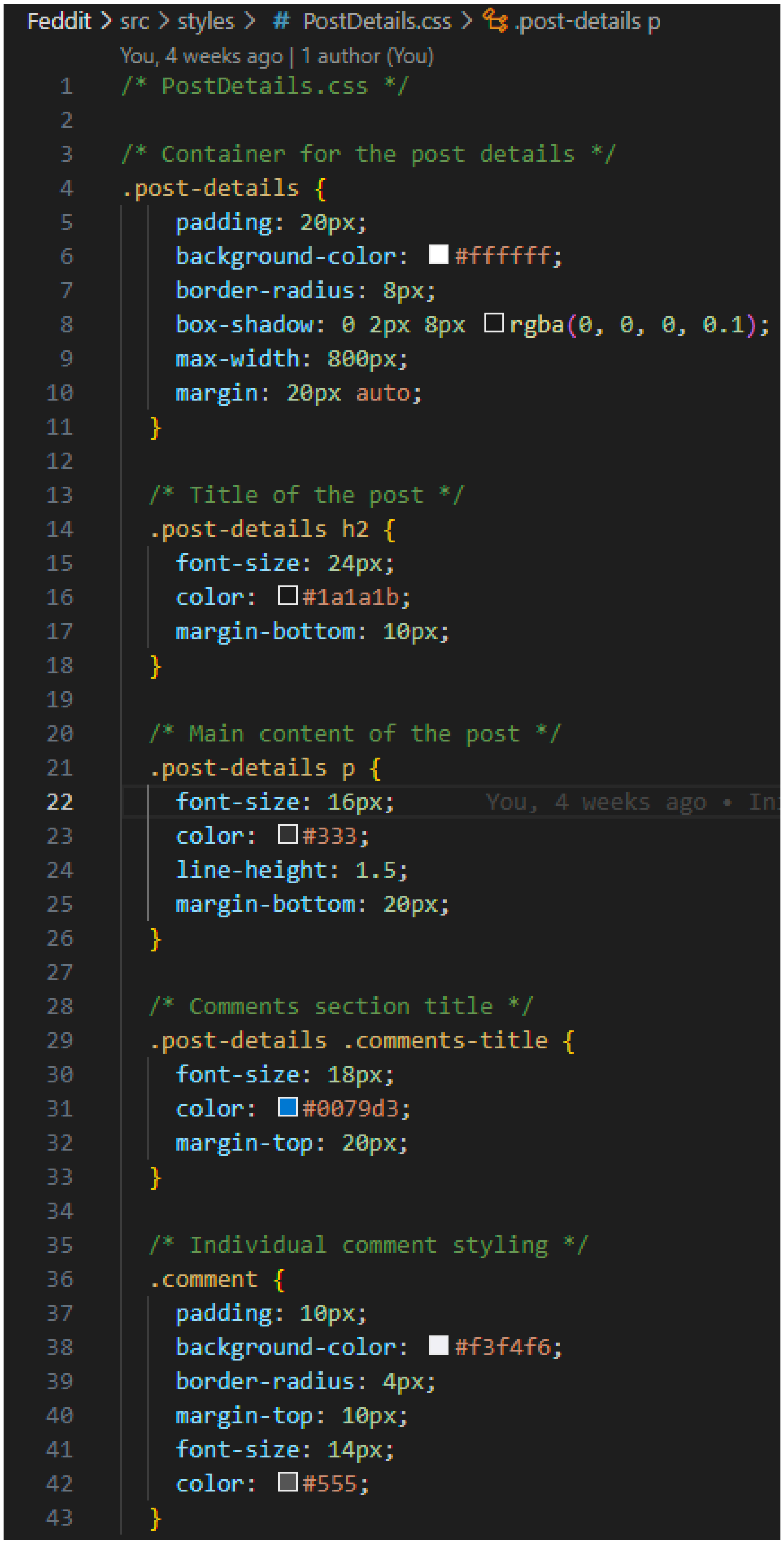
You, 4 weeks ago | 1 author (You)

```css
.post {
  display: flex;
  padding: 20px;
  background-color: #f8f9fa;
  border: 1px solid #e1e4e8;
  border-radius: 8px;
  margin-bottom: 20px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

.vote-section {
  display: flex;
  flex-direction: column;
  align-items: center;
  margin-right: 20px;
}

.vote-button {
  background: none;
  border: none;
  cursor: pointer;
  font-size: 20px;
  color: #6c757d;
  transition: color 0.3s;
}

.vote-button:hover {
  color: #0079d3;
}

.vote-count {
  font-size: 16px;
  font-weight: bold;
  color: #212529;
  margin: 5px 0;
}

.post-content {
  flex-grow: 1;
}

.post h2 {
  font-size: 20px;
  color: #0079d3;
  margin-bottom: 10px;
}
```



```css
.post .author-timestamp {
  font-size: 14px;
  color: #6c757d;
  margin-bottom: 10px;
}

.post .author {
  font-weight: bold;
}

.post p {
  font-size: 16px;
  color: #212529;
  margin-bottom: 15px;
}

.post .media img {
  max-width: 100%;
  border-radius: 8px;
}

.post .media video {
  max-width: 100%;
  border-radius: 8px;
}

.post-feed {
  display: flex;
  flex-direction: column;
  gap: 20px;          You, 4 week
}
```

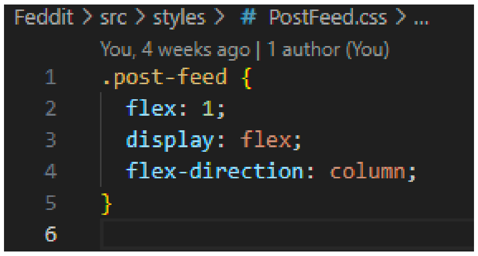
- PostDetails.css:


Feddit > src > styles > # PostDetails.css > .post-details p

You, 4 weeks ago | 1 author (You)

```css
/* PostDetails.css */

/* Container for the post details */
.post-details {
    padding: 20px;
    background-color: #ffffff;
    border-radius: 8px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    max-width: 800px;
    margin: 20px auto;
  }

  /* Title of the post */
  .post-details h2 {
    font-size: 24px;
    color: #1a1a1b;
    margin-bottom: 10px;
  }

  /* Main content of the post */
  .post-details p {
    font-size: 16px;        You, 4 weeks ago • In
    color: #333;
    line-height: 1.5;
    margin-bottom: 20px;
  }

  /* Comments section title */
  .post-details .comments-title {
    font-size: 18px;
    color: #0079d3;
    margin-top: 20px;
  }

  /* Individual comment styling */
  .comment {
    padding: 10px;
    background-color: #f3f4f6;
    border-radius: 4px;
    margin-top: 10px;
    font-size: 14px;
    color: #555;
  }
```

- **PostFeed.css:**

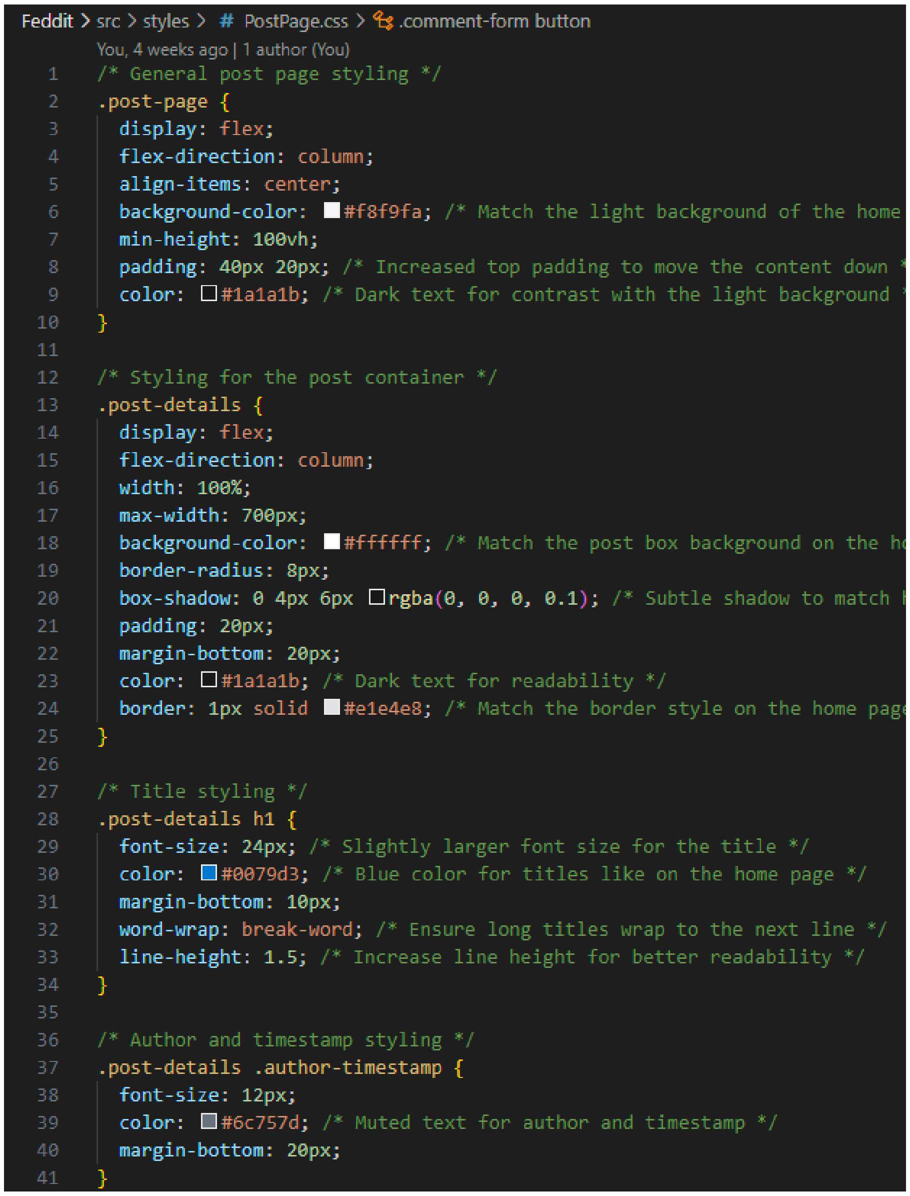


Feddit > src > styles > # PostFeed.css > ...

You, 4 weeks ago | 1 author (You)

```css
.post-feed {
  flex: 1;
  display: flex;
  flex-direction: column;
}

```

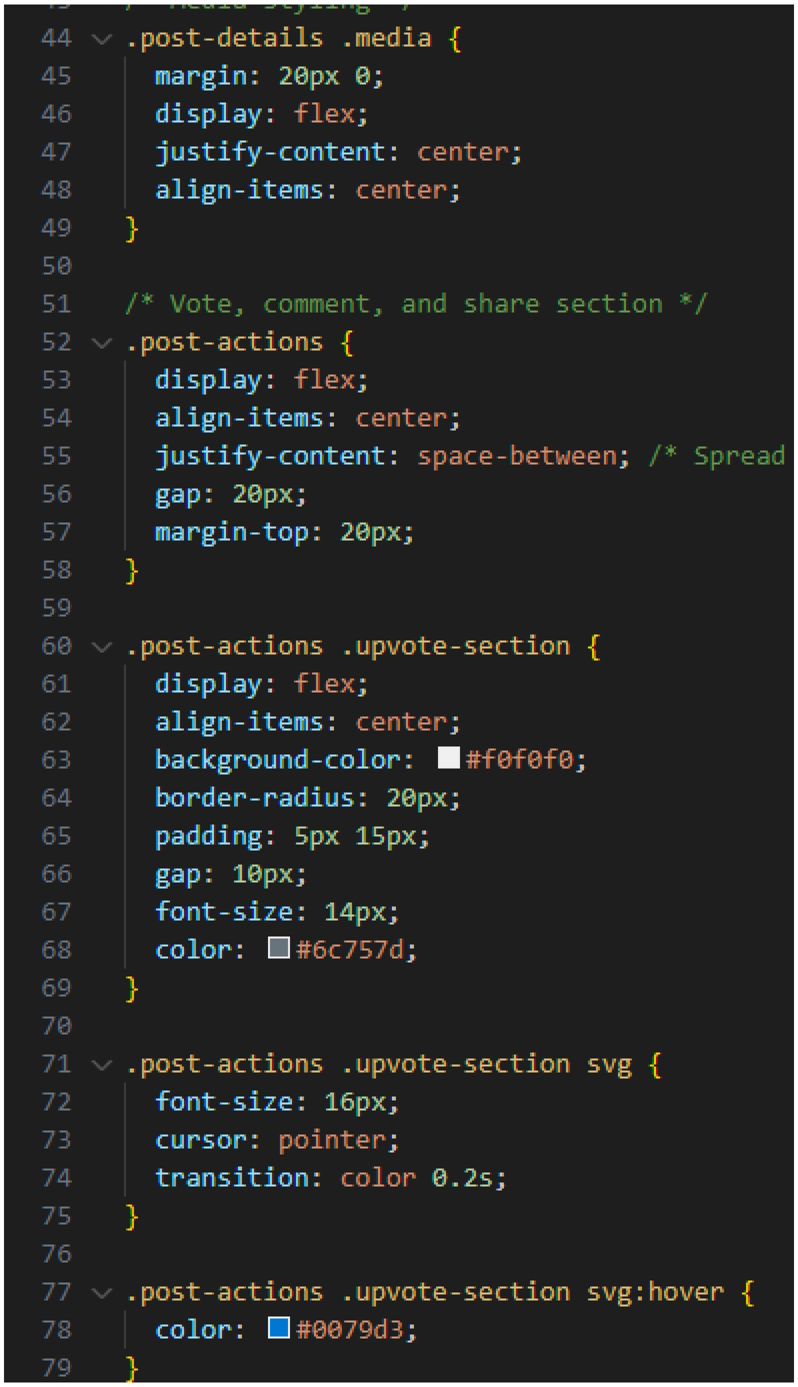
- **PostPage.css:**



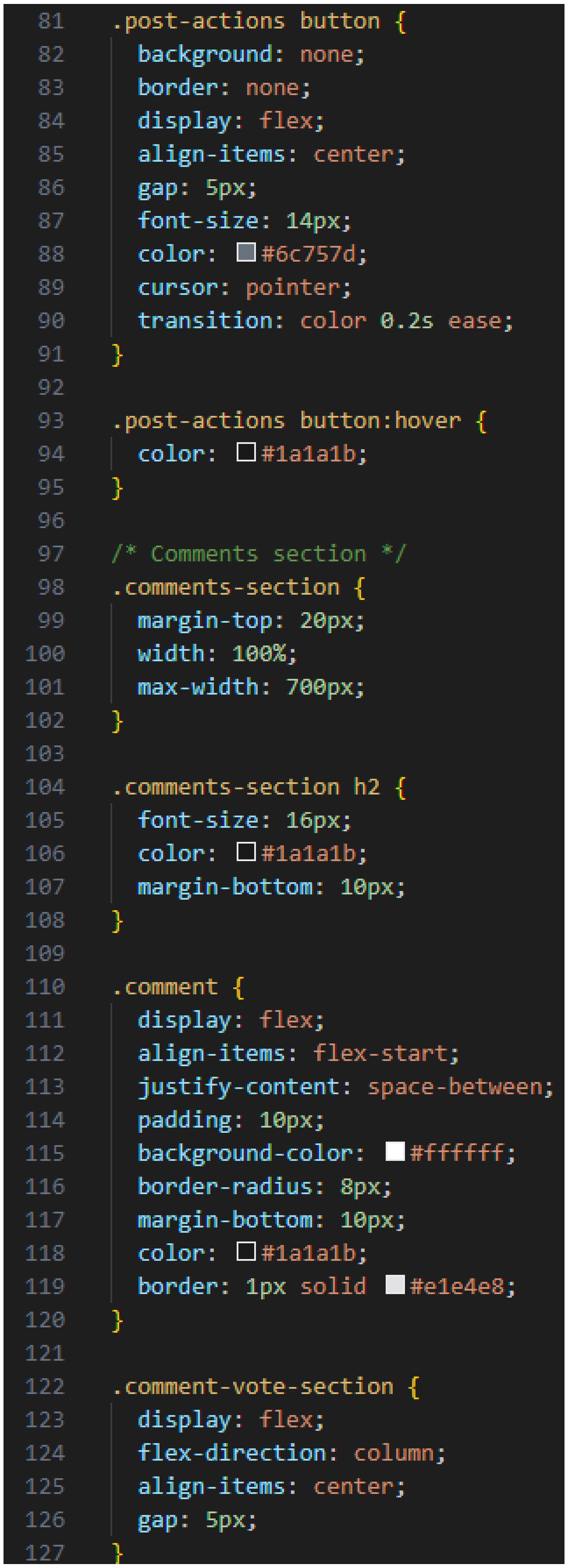
Feddit > src > styles > # PostPage.css > .comment-form button

You, 4 weeks ago | 1 author (You)

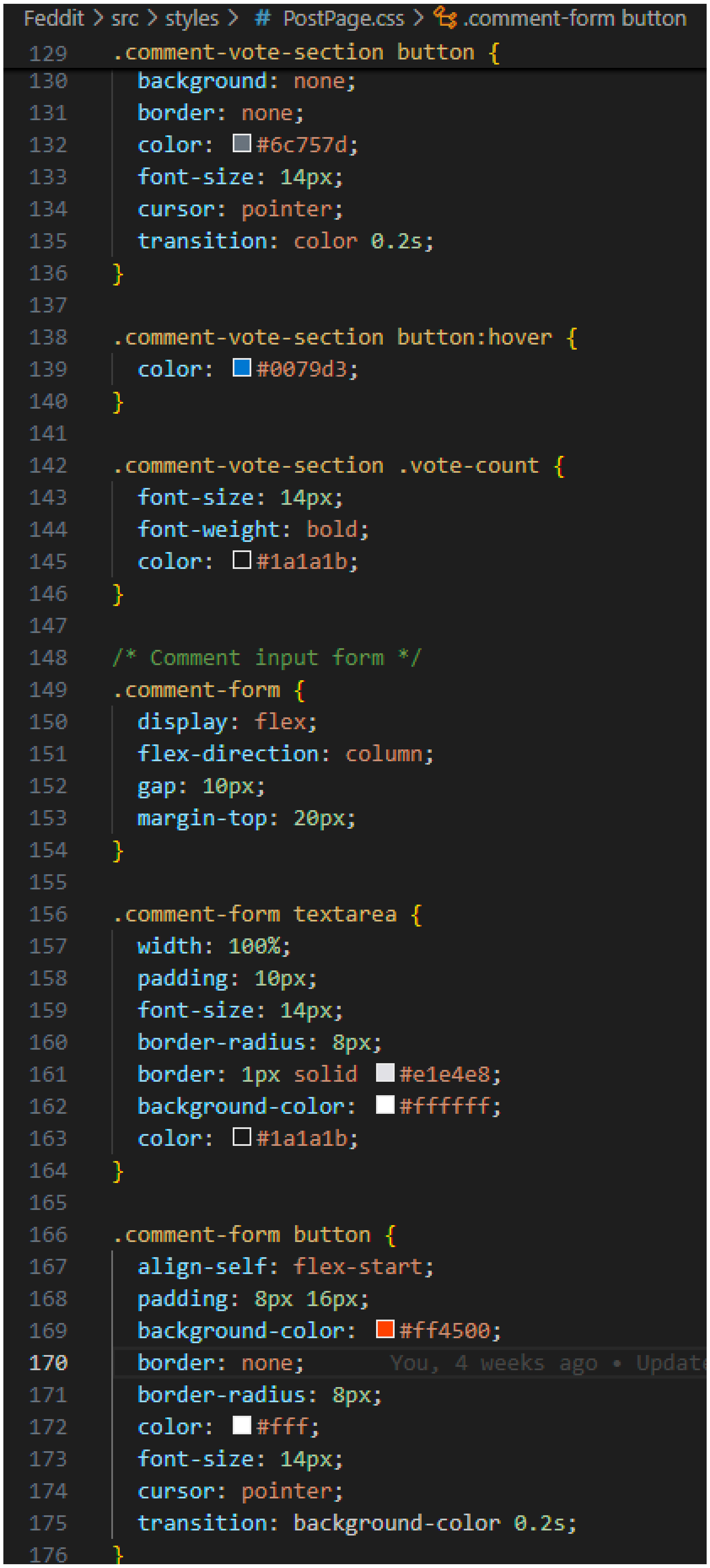
```css
/* General post page styling */
.post-page {
  display: flex;
  flex-direction: column;
  align-items: center;
  background-color: #f8f9fa; /* Match the light background of the home
  min-height: 100vh;
  padding: 40px 20px; /* Increased top padding to move the content down *
  color: #1a1a1b; /* Dark text for contrast with the light background *
}

/* Styling for the post container */
.post-details {
  display: flex;
  flex-direction: column;
  width: 100%;
  max-width: 700px;
  background-color: #ffffff; /* Match the post box background on the h
  border-radius: 8px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1); /* Subtle shadow to match h
  padding: 20px;
  margin-bottom: 20px;
  color: #1a1a1b; /* Dark text for readability */
  border: 1px solid #e1e4e8; /* Match the border style on the home page
}

/* Title styling */
.post-details h1 {
  font-size: 24px; /* Slightly larger font size for the title */
  color: #0079d3; /* Blue color for titles like on the home page */
  margin-bottom: 10px;
  word-wrap: break-word; /* Ensure long titles wrap to the next line */
  line-height: 1.5; /* Increase line height for better readability */
}

/* Author and timestamp styling */
.post-details .author-timestamp {
  font-size: 12px;
  color: #6c757d; /* Muted text for author and timestamp */
  margin-bottom: 20px;
}
```



```css
.post-details .media {
  margin: 20px 0;
  display: flex;
  justify-content: center;
  align-items: center;
}

/* Vote, comment, and share section */
.post-actions {
  display: flex;
  align-items: center;
  justify-content: space-between; /* Spread
  gap: 20px;
  margin-top: 20px;
}

.post-actions .upvote-section {
  display: flex;
  align-items: center;
  background-color: #f0f0f0;
  border-radius: 20px;
  padding: 5px 15px;
  gap: 10px;
  font-size: 14px;
  color: #6c757d;
}

.post-actions .upvote-section svg {
  font-size: 16px;
  cursor: pointer;
  transition: color 0.2s;
}

.post-actions .upvote-section svg:hover {
  color: #0079d3;
}
```

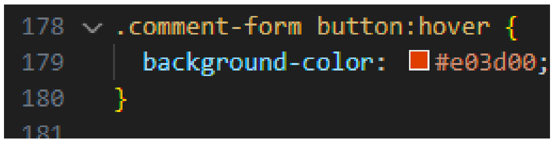


```css
.post-actions button {
  background: none;
  border: none;
  display: flex;
  align-items: center;
  gap: 5px;
  font-size: 14px;
  color: #6c757d;
  cursor: pointer;
  transition: color 0.2s ease;
}

.post-actions button:hover {
  color: #1a1a1b;
}

/* Comments section */
.comments-section {
  margin-top: 20px;
  width: 100%;
  max-width: 700px;
}

.comments-section h2 {
  font-size: 16px;
  color: #1a1a1b;
  margin-bottom: 10px;
}

.comment {
  display: flex;
  align-items: flex-start;
  justify-content: space-between;
  padding: 10px;
  background-color: #ffffff;
  border-radius: 8px;
  margin-bottom: 10px;
  color: #1a1a1b;
  border: 1px solid #e1e4e8;
}

.comment-vote-section {
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 5px;
}
```



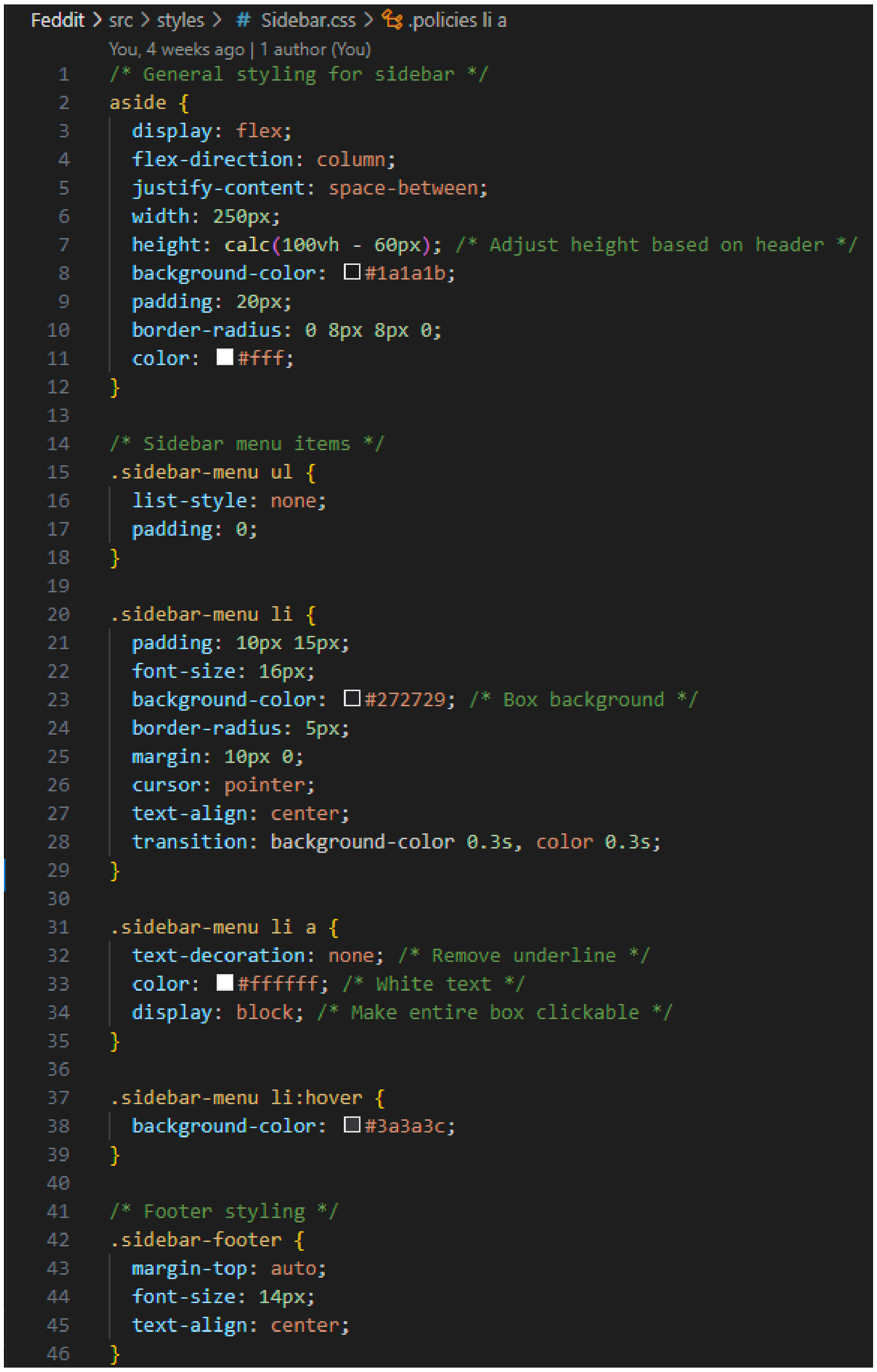
Feddit > src > styles > # PostPage.css > .comment-form button

```css
.comment-vote-section button {
  background: none;
  border: none;
  color: #6c757d;
  font-size: 14px;
  cursor: pointer;
  transition: color 0.2s;
}

.comment-vote-section button:hover {
  color: #0079d3;
}

.comment-vote-section .vote-count {
  font-size: 14px;
  font-weight: bold;
  color: #1a1a1b;
}

/* Comment input form */
.comment-form {
  display: flex;
  flex-direction: column;
  gap: 10px;
  margin-top: 20px;
}

.comment-form textarea {
  width: 100%;
  padding: 10px;
  font-size: 14px;
  border-radius: 8px;
  border: 1px solid #e1e4e8;
  background-color: #ffffff;
  color: #1a1a1b;
}

.comment-form button {
  align-self: flex-start;
  padding: 8px 16px;
  background-color: #ff4500;
  border: none;          You, 4 weeks ago • Update
  border-radius: 8px;
  color: #fff;
  font-size: 14px;
  cursor: pointer;
  transition: background-color 0.2s;
}
```



```
.comment-form button:hover {
  background-color: #e03d00;
}
```

- **Sidebar.css:**

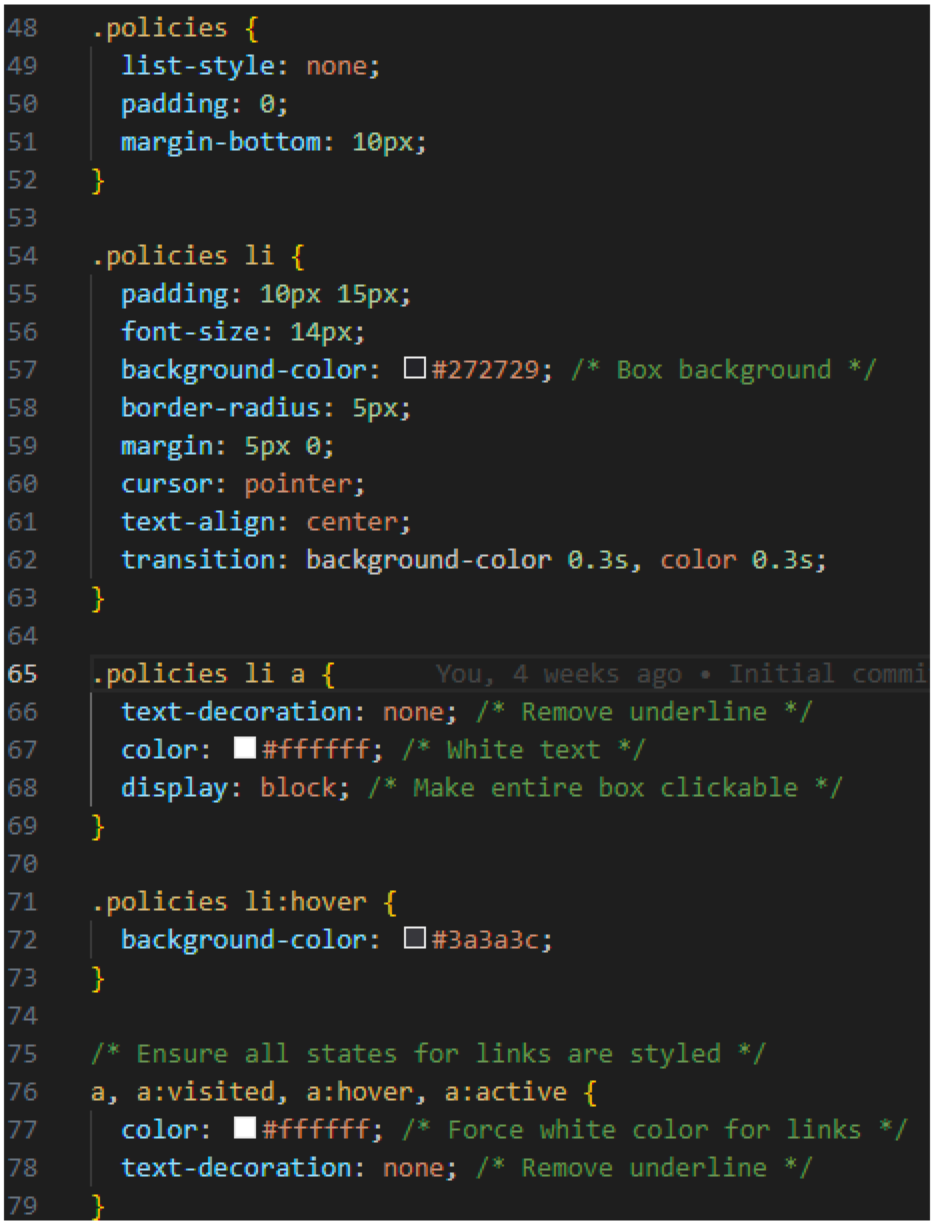


Feddit > src > styles > # Sidebar.css > .policies li a

You, 4 weeks ago | 1 author (You)

```
/* General styling for sidebar */
aside {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  width: 250px;
  height: calc(100vh - 60px); /* Adjust height based on header */
  background-color: #1a1a1b;
  padding: 20px;
  border-radius: 0 8px 8px 0;
  color: #fff;
}

/* Sidebar menu items */
.sidebar-menu ul {
  list-style: none;
  padding: 0;
}

.sidebar-menu li {
  padding: 10px 15px;
  font-size: 16px;
  background-color: #272729; /* Box background */
  border-radius: 5px;
  margin: 10px 0;
  cursor: pointer;
  text-align: center;
  transition: background-color 0.3s, color 0.3s;
}

.sidebar-menu li a {
  text-decoration: none; /* Remove underline */
  color: #ffffff; /* White text */
  display: block; /* Make entire box clickable */
}

.sidebar-menu li:hover {
  background-color: #3a3a3c;
}

/* Footer styling */
.sidebar-footer {
  margin-top: auto;
  font-size: 14px;
  text-align: center;
}
```

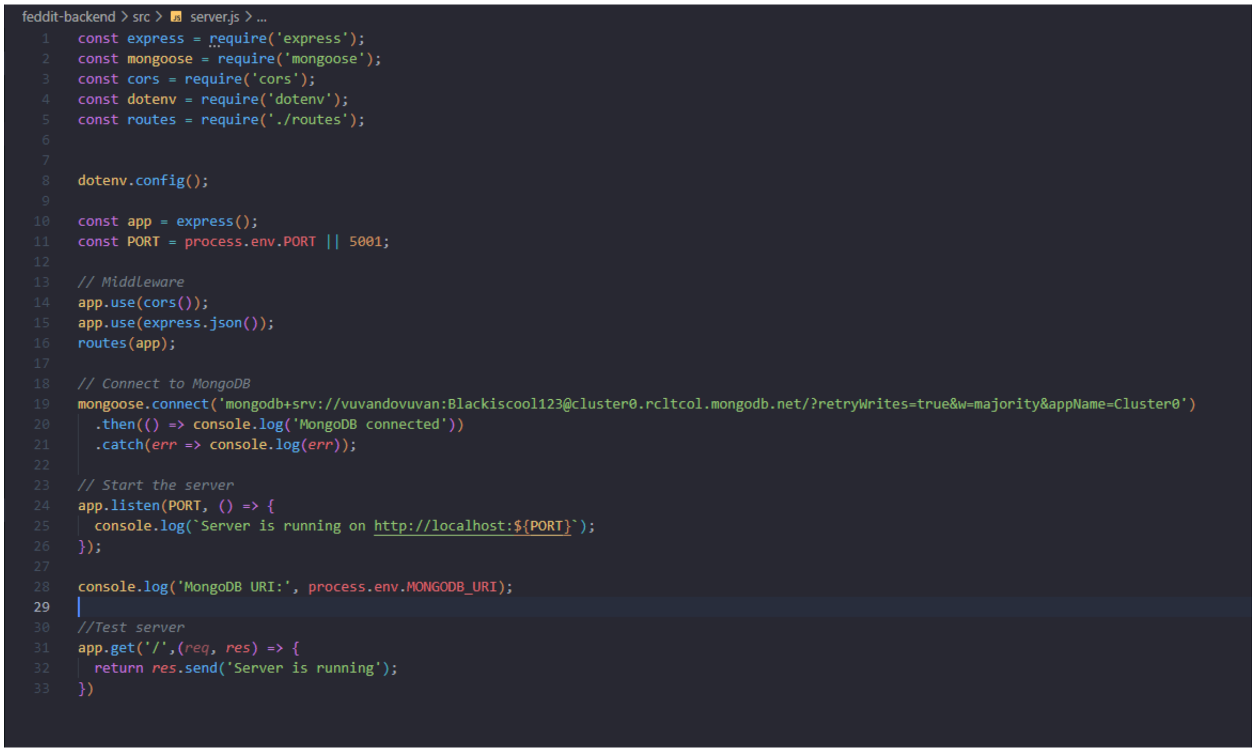
a



```css
.policies {
  list-style: none;
  padding: 0;
  margin-bottom: 10px;
}

.policies li {
  padding: 10px 15px;
  font-size: 14px;
  background-color: #272729; /* Box background */
  border-radius: 5px;
  margin: 5px 0;
  cursor: pointer;
  text-align: center;
  transition: background-color 0.3s, color 0.3s;
}

.policies li a {        You, 4 weeks ago • Initial commi
  text-decoration: none; /* Remove underline */
  color: #ffffff; /* White text */
  display: block; /* Make entire box clickable */
}

.policies li:hover {
  background-color: #3a3a3c;
}

/* Ensure all states for links are styled */
a, a:visited, a:hover, a:active {
  color: #ffffff; /* Force white color for links */
  text-decoration: none; /* Remove underline */
}
```

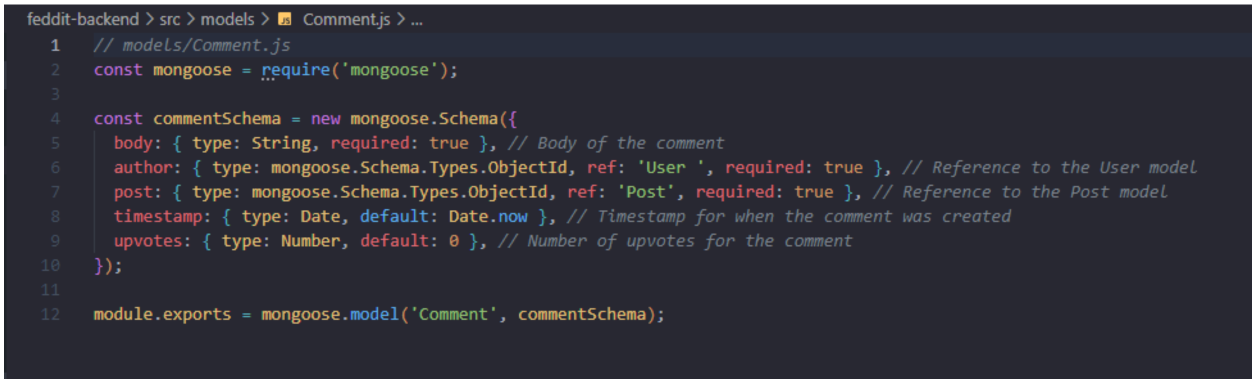
b) **<u>Back-end:</u>**

- **Server.js:**


feddit-backend > src > server.js > ...

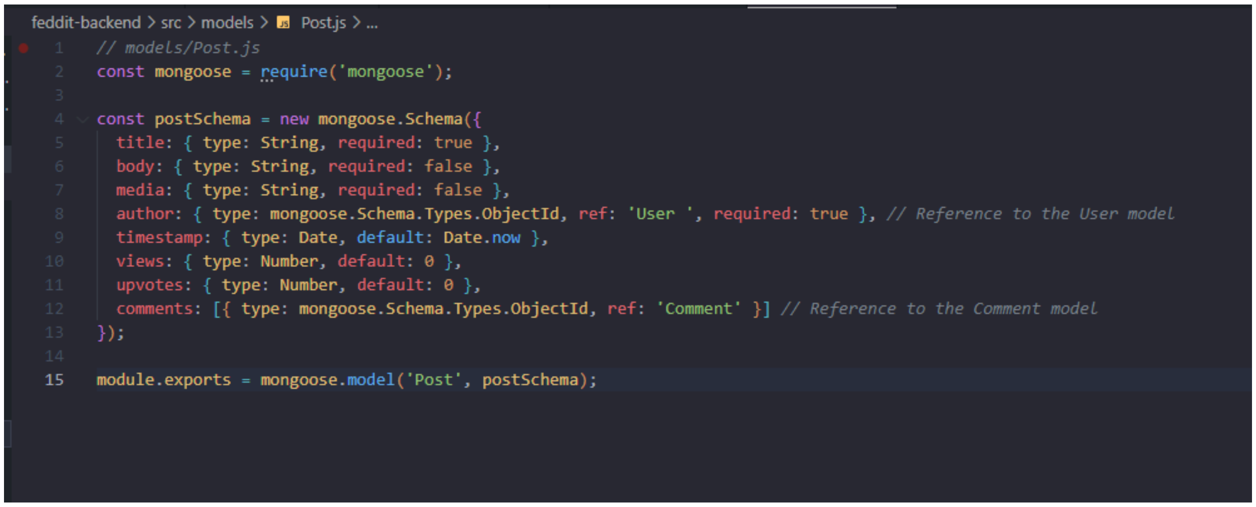
```javascript
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const dotenv = require('dotenv');
const routes = require('./routes');


dotenv.config();

const app = express();
const PORT = process.env.PORT || 5001;

// Middleware
app.use(cors());
app.use(express.json());
routes(app);

// Connect to MongoDB
mongoose.connect('mongodb+srv://vuvandovuvan:Blackiscool123@cluster0.rcltcol.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0')
  .then(() => console.log('MongoDB connected'))
  .catch(err => console.log(err));

// Start the server
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});

console.log('MongoDB URI:', process.env.MONGODB_URI);

//Test server
app.get('/',(req, res) => {
  return res.send('Server is running');
})
```

- Comments Model:


feddit-backend > src > models > Comment.js > ...

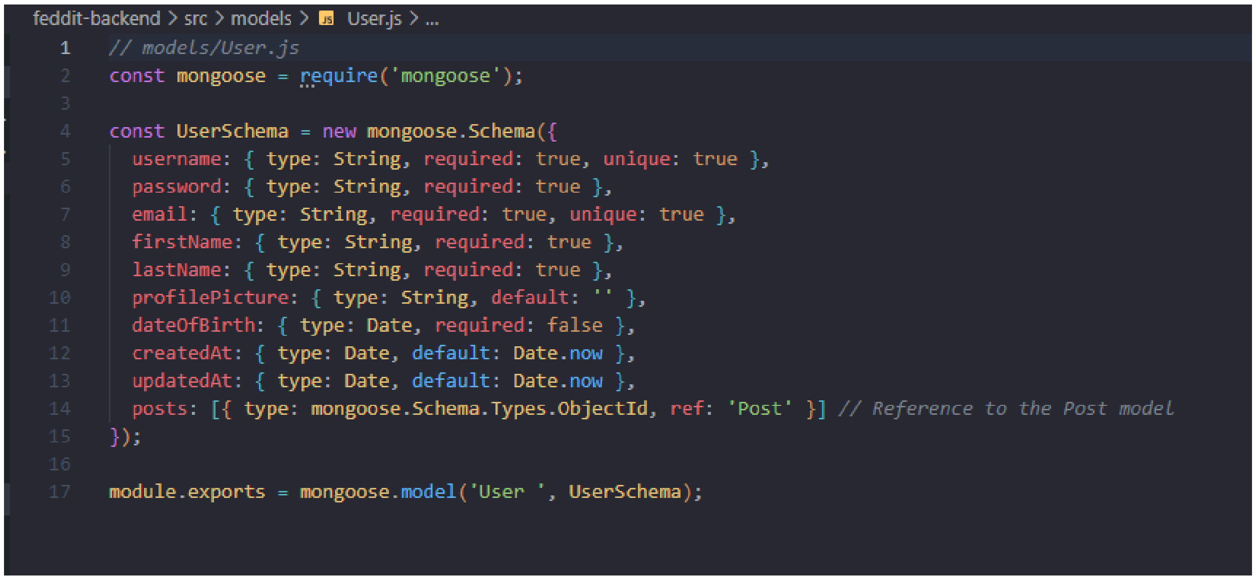
```javascript
// models/Comment.js
const mongoose = require('mongoose');

const commentSchema = new mongoose.Schema({
  body: { type: String, required: true }, // Body of the comment
  author: { type: mongoose.Schema.Types.ObjectId, ref: 'User ', required: true }, // Reference to the User model
  post: { type: mongoose.Schema.Types.ObjectId, ref: 'Post', required: true }, // Reference to the Post model
  timestamp: { type: Date, default: Date.now }, // Timestamp for when the comment was created
  upvotes: { type: Number, default: 0 }, // Number of upvotes for the comment
});

module.exports = mongoose.model('Comment', commentSchema);
```

- Post Model:


feddit-backend > src > models > Post.js > ...

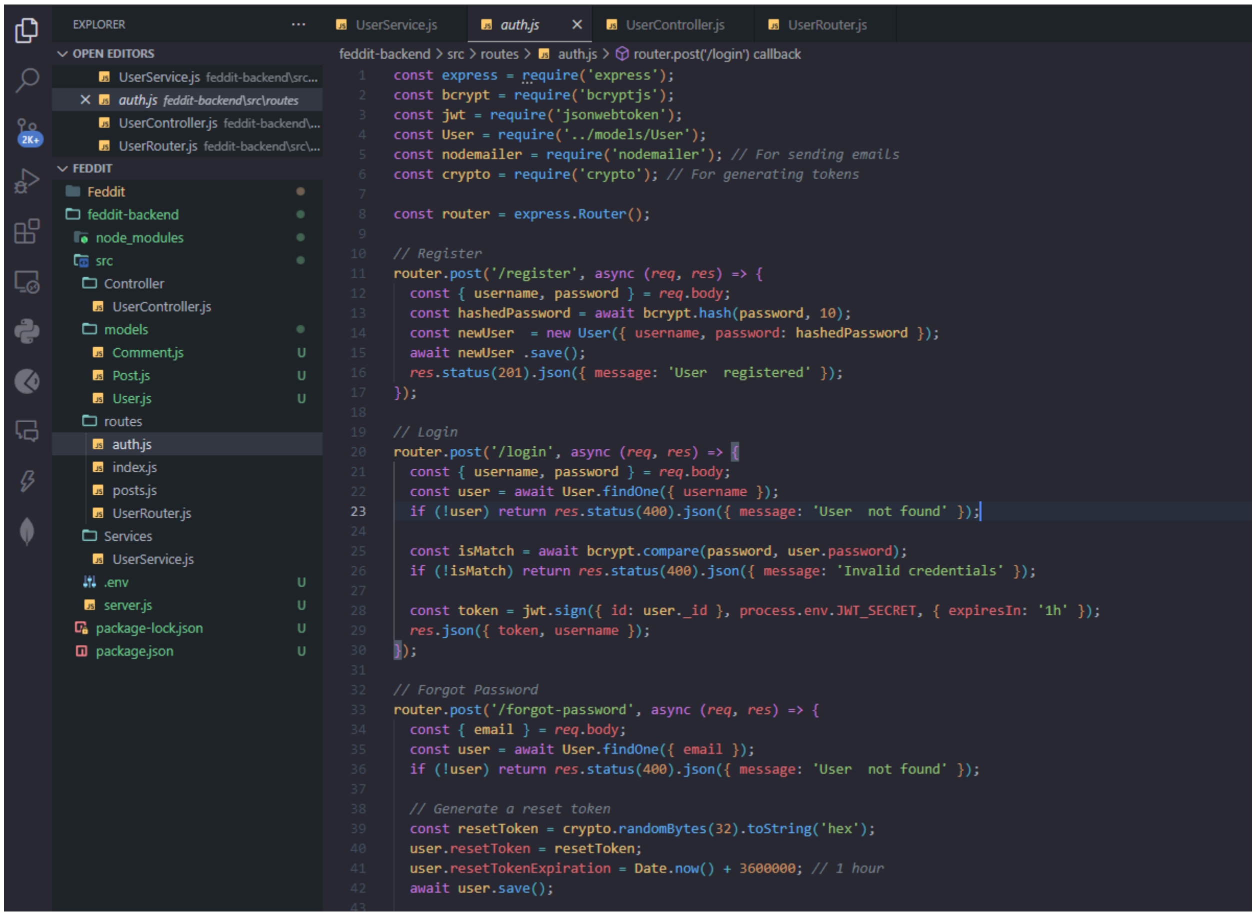
```javascript
// models/Post.js
const mongoose = require('mongoose');

const postSchema = new mongoose.Schema({
  title: { type: String, required: true },
  body: { type: String, required: false },
  media: { type: String, required: false },
  author: { type: mongoose.Schema.Types.ObjectId, ref: 'User ', required: true }, // Reference to the User model
  timestamp: { type: Date, default: Date.now },
  views: { type: Number, default: 0 },
  upvotes: { type: Number, default: 0 },
  comments: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Comment' }] // Reference to the Comment model
});

module.exports = mongoose.model('Post', postSchema);
```

- User Model:

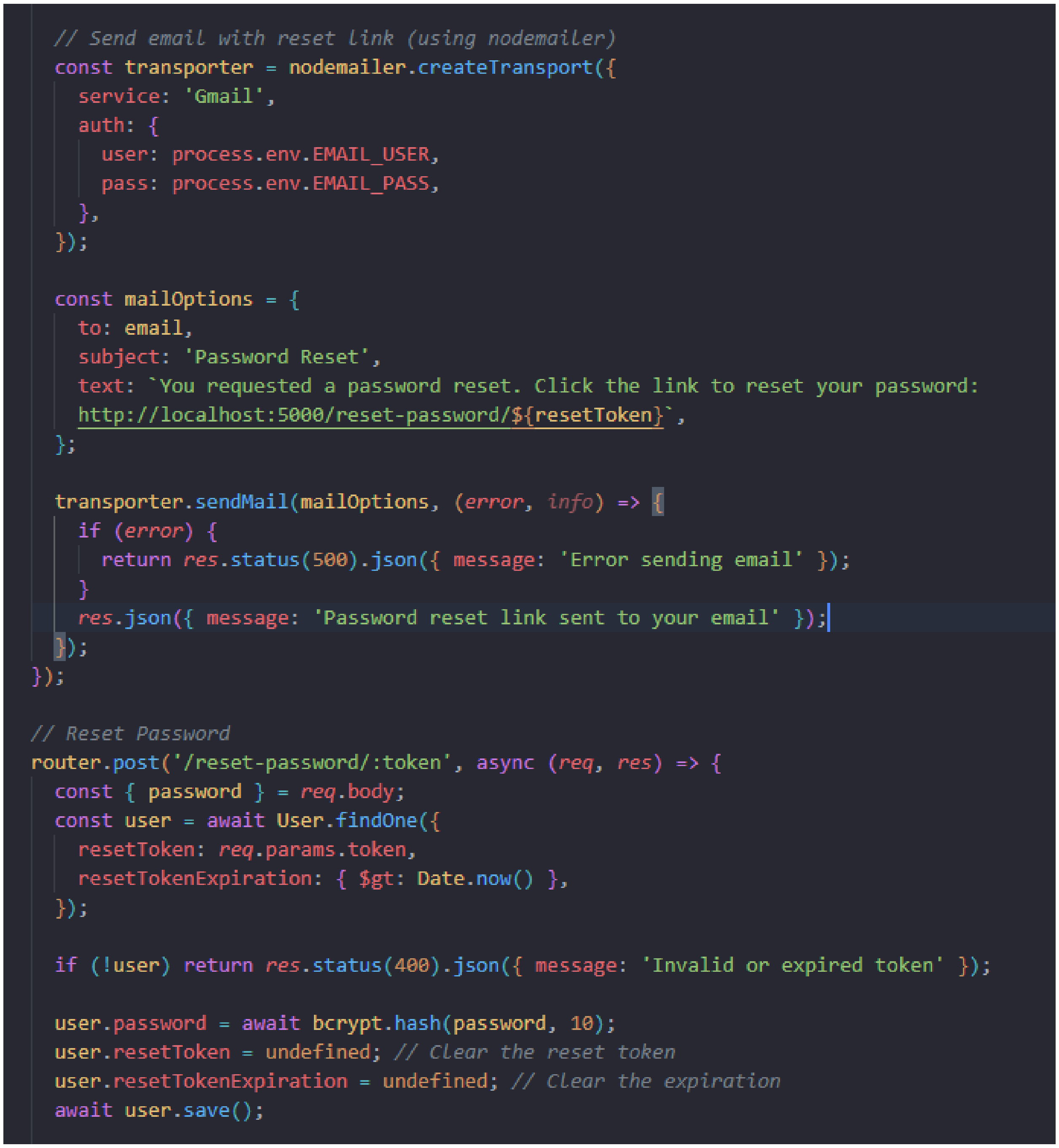

feddit-backend > src > models > User.js > ...

```javascript
// models/User.js
const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  firstName: { type: String, required: true },
  lastName: { type: String, required: true },
  profilePicture: { type: String, default: '' },
  dateOfBirth: { type: Date, required: false },
  createdAt: { type: Date, default: Date.now },
  updatedAt: { type: Date, default: Date.now },
  posts: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Post' }] // Reference to the Post model
});

module.exports = mongoose.model('User ', UserSchema);
```

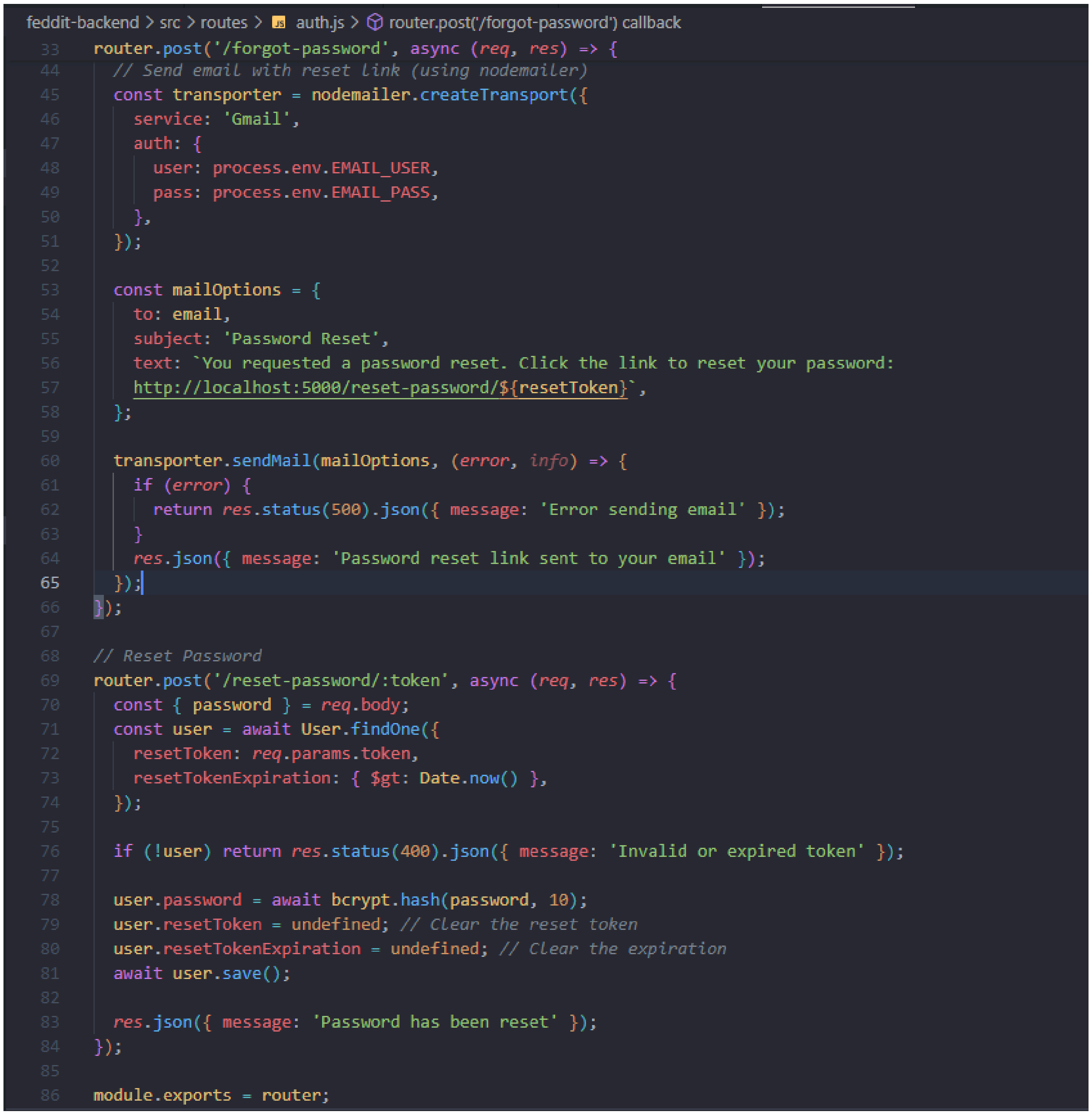
- Authentication Route:



```javascript
const express = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const User = require('../models/User');
const nodemailer = require('nodemailer'); // For sending emails
const crypto = require('crypto'); // For generating tokens

const router = express.Router();

// Register
router.post('/register', async (req, res) => {
  const { username, password } = req.body;
  const hashedPassword = await bcrypt.hash(password, 10);
  const newUser  = new User({ username, password: hashedPassword });
  await newUser .save();
  res.status(201).json({ message: 'User  registered' });
});

// Login
router.post('/login', async (req, res) => {
  const { username, password } = req.body;
  const user = await User.findOne({ username });
  if (!user) return res.status(400).json({ message: 'User  not found' });

  const isMatch = await bcrypt.compare(password, user.password);
  if (!isMatch) return res.status(400).json({ message: 'Invalid credentials' });

  const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, { expiresIn: '1h' });
  res.json({ token, username });
});

// Forgot Password
router.post('/forgot-password', async (req, res) => {
  const { email } = req.body;
  const user = await User.findOne({ email });
  if (!user) return res.status(400).json({ message: 'User  not found' });

  // Generate a reset token
  const resetToken = crypto.randomBytes(32).toString('hex');
  user.resetToken = resetToken;
  user.resetTokenExpiration = Date.now() + 3600000; // 1 hour
  await user.save();
```



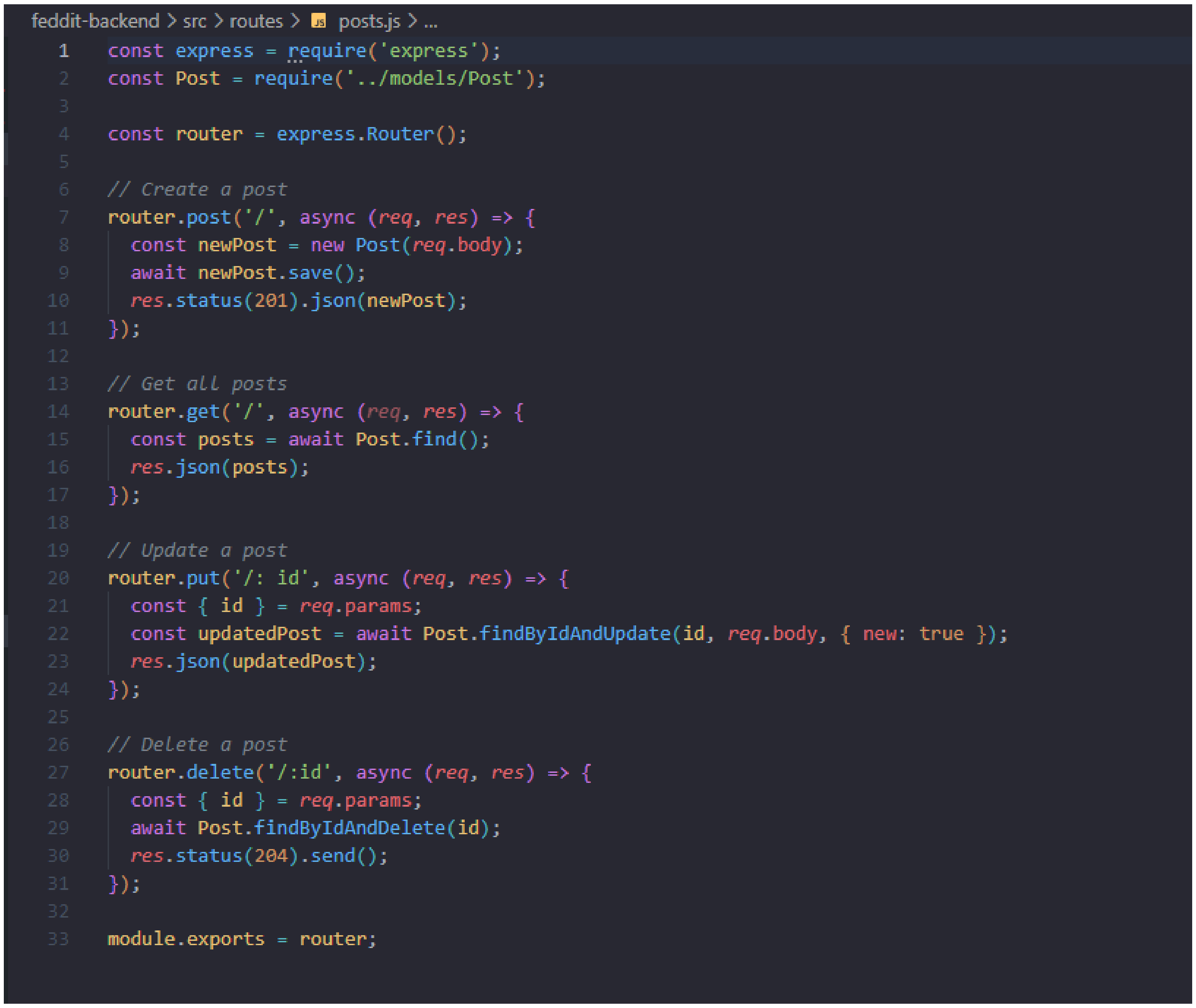
```
  // Send email with reset link (using nodemailer)
  const transporter = nodemailer.createTransport({
    service: 'Gmail',
    auth: {
      user: process.env.EMAIL_USER,
      pass: process.env.EMAIL_PASS,
    },
  });

  const mailOptions = {
    to: email,
    subject: 'Password Reset',
    text: `You requested a password reset. Click the link to reset your password:
    http://localhost:5000/reset-password/${resetToken}`,
  };

  transporter.sendMail(mailOptions, (error, info) => {
    if (error) {
      return res.status(500).json({ message: 'Error sending email' });
    }
    res.json({ message: 'Password reset link sent to your email' });
  });
});

// Reset Password
router.post('/reset-password/:token', async (req, res) => {
  const { password } = req.body;
  const user = await User.findOne({
    resetToken: req.params.token,
    resetTokenExpiration: { $gt: Date.now() },
  });

  if (!user) return res.status(400).json({ message: 'Invalid or expired token' });

  user.password = await bcrypt.hash(password, 10);
  user.resetToken = undefined; // Clear the reset token
  user.resetTokenExpiration = undefined; // Clear the expiration
  await user.save();
```



feddit-backend > src > routes > auth.js > router.post('/forgot-password') callback

```
router.post('/forgot-password', async (req, res) => {
  // Send email with reset link (using nodemailer)
  const transporter = nodemailer.createTransport({
    service: 'Gmail',
    auth: {
      user: process.env.EMAIL_USER,
      pass: process.env.EMAIL_PASS,
    },
  });

  const mailOptions = {
    to: email,
    subject: 'Password Reset',
    text: `You requested a password reset. Click the link to reset your password:
    http://localhost:5000/reset-password/${resetToken}`,
  };

  transporter.sendMail(mailOptions, (error, info) => {
    if (error) {
      return res.status(500).json({ message: 'Error sending email' });
    }
    res.json({ message: 'Password reset link sent to your email' });
  });
});

// Reset Password
router.post('/reset-password/:token', async (req, res) => {
  const { password } = req.body;
  const user = await User.findOne({
    resetToken: req.params.token,
    resetTokenExpiration: { $gt: Date.now() },
  });

  if (!user) return res.status(400).json({ message: 'Invalid or expired token' });

  user.password = await bcrypt.hash(password, 10);
  user.resetToken = undefined; // Clear the reset token
  user.resetTokenExpiration = undefined; // Clear the expiration
  await user.save();

  res.json({ message: 'Password has been reset' });
});

module.exports = router;
```

- **Post Route:**



feddit-backend > src > routes > posts.js > ...

```javascript
const express = require('express');
const Post = require('../models/Post');

const router = express.Router();

// Create a post
router.post('/', async (req, res) => {
  const newPost = new Post(req.body);
  await newPost.save();
  res.status(201).json(newPost);
});

// Get all posts
router.get('/', async (req, res) => {
  const posts = await Post.find();
  res.json(posts);
});

// Update a post
router.put('/: id', async (req, res) => {
  const { id } = req.params;
  const updatedPost = await Post.findByIdAndUpdate(id, req.body, { new: true });
  res.json(updatedPost);
});

// Delete a post
router.delete('/:id', async (req, res) => {
  const { id } = req.params;
  await Post.findByIdAndDelete(id);
  res.status(204).send();
});

module.exports = router;
```