# ALX1\_SWD2\_M2d

# Simple social media app

## Project Link

<https://github.com/DevTrimo/social-media-app.git>

## Team:

1. Mohamed Mosaad
2. Ahmed Ibrahim
3. Emad Mamdouh
4. Mohamed Salah
5. Ziad Khaled
6. Nourhan Ahmed

## Overview

This social media web application allows users to connect with friends, view profiles, and manage friendships. It utilizes React for the frontend, Material-UI for styling, and Redux for state management.

## Frontend

## Components

1.FlexBetween

A reusable styled component that creates a flex container with items spaced evenly and vertically centered.

2. Friend:

A component that displays a friend's information and allows the user to add or remove them as a friend.

3. UserImage:

A component that displays a user's profile picture.

4. WidgetWrapper:

A styled component that provides a consistent wrapper for widgets in the application.

## Key Features:

**Friend Management:** Users can add or remove friends using the Friend component.

**Profile Navigation:** Clicking on a friend's name navigates to their profile page.

**Responsive Design:** Utilizes Material-UI for a responsive and consistent user interface.

**State Management:** Uses Redux for managing application state, including user information and friend lists.

## State Management

The application uses Redux for state management. Key state slices include:

**user:** Contains user information, including the friends list

**token:** Stores the authentication token

Redux state management for the application using Redux Toolkit. It creates a slice for authentication and app-wide state management.

**Key Features:**

* Manages application-wide state
* Handles user authentication
* Controls theme mode (light/dark)
* Manages posts and friends lists

## API Integration

The app interacts with a backend API for data fetching and updates

## Styling

The application uses Material-UI for styling, with custom theme settings accessible via

useTheme

## Routing

React Router is used for navigation between different views in the application.

## Scenes:

## 1. HomePage

The HomePage is the main landing page for authenticated users. It displays the user's feed, allows for creating new posts, and shows widgets for user information and friend lists.

## 2. Form

Form component handles both login and registration functionality for the application.

It uses the formik library for form management and validation, and

yup` for defining validation schemas.

Key Features:

* Supports both login and registration forms
* Validates form inputs using yup schemas
* Handles form submission and interacts with the backend API
* Allows users to upload a profile picture during registration
* Dispatches Redux actions for authentication and navigation

## 3.Navbar

The Navbar component is a responsive navigation bar that appears at the top of the application. It provides navigation, search functionality, and access to user-related actions.

* **Key Features:**
* Responsive design for both desktop and mobile views
* Search functionality (on desktop view)
* Theme toggle (light/dark mode)
* Quick access to messages, notifications, and help User menu with logout option
* Mobile menu toggle for smaller screens

## 4.ProfilePage

ProfilePage component provides a comprehensive view of a user's profile in the social media

web app. It combines various widgets to display user information, posts, and friend list, while

 maintaining a responsive design for different screen sizes.

**Key Features:**

* Displays user information
* Shows user's posts
* Displays user's friend list
* Allows creating new posts
* Responsive design for both desktop and mobile views

**Backend**

Each of these controllers plays a critical role in social media app, handling various user interactions and data management tasks.

## 1. Auth Controller

1. register:
   * Purpose: Registers a new user.
   * Function: Takes user details (name, email, password, etc.) from the request body. Hashes the password and saves the user in the database. Responds with the created user data.
2. login:
   * Purpose: Logs in an existing user.
   * Function: Takes email and password from the request body. Finds the user by email, compares the provided password with the stored hashed password. Generates a JWT token if credentials are valid and responds with the token and user data.

## 2. Users Controller

1. getUser:
   * Purpose: Fetches a user by ID.
   * Function: Takes user ID from the request parameters. Finds and returns the user data.
2. getUserFriends:
   * Purpose: Retrieves a list of a user's friends.
   * Function: Takes user ID from the request parameters. Finds the user and retrieves their friends. Formats and responds with the friends' data.
3. addRemoveFriend:
   * Purpose: Adds or removes a friend for a user.
   * Function: Takes user ID and friend ID from the request parameters. Adds or removes the friend from the user's friend list and vice versa. Updates and saves both user and friend data. Responds with the updated friends list.

## 3. Posts Controller:

1. createPost:
   * Purpose: Creates a new post.
   * Function: Takes post details (user ID, description, picture path) from the request body. Finds the user by user ID. Creates and saves a new post. Responds with the created post data.
2. getFeedPosts:
   * Purpose: Fetches all posts for the feed.
   * Function: Retrieves all posts from the database and responds with the posts data.
3. getUserPosts:
   * Purpose: Fetches posts by a specific user.
   * Function: Takes user ID from the request parameters. Finds and returns posts created by the user.
4. likePost:
   * Purpose: Likes or unlikes a post.
   * Function: Takes post ID from the request parameters and user ID from the request body. Finds the post and toggles the like status. Updates and saves the post data. Responds with the updated post data.