***ConnectSphere:***

***A Social Media App***

**by**

**Akshay Maruti Kamble**

Submitted on: 30-03-2024

Project Duration: 01-03-2024 to 30-03-2024

Contact Information: akshaykamble1431@gmail.com

***Table Of Contents***

|  |  |  |
| --- | --- | --- |
| Sr.No | Chapter | Page No. |
| 1. | Introduction |  |
| 2. | Objectives |  |
| 3. | Technologies Used |  |
| 4. | System Architecture |  |
| 5. | Features |  |
| 6. | Future Improvements |  |
| 7. | Conclusion |  |

***Table Of Figures***

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

***1. Introduction***

In today's digital age, social media platforms have become integral parts of our daily lives, facilitating communication, connection, and collaboration on a global scale. With the increasing demand for user-centric and feature-rich social networking experiences, the development of innovative social media applications has become paramount.

In response to this demand, the project titled "ConnectSphere" aims to deliver a comprehensive and immersive social media platform designed to connect users in meaningful ways.

***2. Objectives***

The primary objective of ConnectSphere is to provide users with a seamless and engaging social media experience that fosters interaction, communication, and collaboration. To achieve this objective, the project focuses on the following key goals:

***Technological Innovation:*** Leveraging modern web development technologies and frameworks, including React.js and Node.js, to build a robust and scalable social media platform.

***User Authentication and Authorization:*** Implementing secure user authentication and authorization mechanisms using JSON Web Tokens (JWT) to ensure the privacy and security of user data.

***Real-time Data Management:*** Utilizing React Query to efficiently manage and fetch data from the server, enabling real-time updates and seamless user interactions.

***Responsive User Interface:*** Designing a responsive and intuitive user interface using React.js and CSS to enhance user experience across various devices and screen sizes.

***3. Technologies Used***

ConnectSphere is developed using a combination of frontend and backend technologies to deliver a comprehensive and feature-rich social media experience. The key technologies and tools utilized in the development process include:

***Frontend Technologies:*** React.js, JavaScript, JSX, React Query

***Backend Technologies:*** Node.js, Express.js

***Database:*** MySQL

***Authentication:*** JSON Web Tokens (JWT)

By leveraging these technologies, ConnectSphere aims to deliver a cutting-edge social media platform that seamlessly integrates user-centric features with modern web development practices.

***4. System Architecture***

***React.js (Frontend):***

* ***Component-Based Architecture:*** React.js follows a component-based architecture, where the UI is broken down into reusable components.
* ***Virtual DOM:*** React maintains a virtual representation of the DOM in memory, allowing efficient updates and rendering of UI components.
* ***State Management:*** React uses states to manage component data and re-renders components when the state changes.
* ***Unidirectional Data Flow:*** Data flows in one direction, from parent to child components, ensuring predictable and manageable state updates.

***Node.js and Express.js (Backend):***

* ***Event-Driven Architecture:*** Node.js follows an event-driven architecture, where asynchronous, non-blocking I/O operations are handled through event loops.
* ***Middleware Stack:*** Express.js provides a middleware stack for handling HTTP requests and responses. Each middleware function can modify the request or response objects.
* ***Routing:*** Express.js allows you to define routes to handle different HTTP methods and URL patterns, making it easy to create RESTful APIs.
* ***Database Integration:*** Node.js can integrate with various databases, including MySQL, using libraries like Sequelize or MySQL.js to perform database operations.

***MySQL (Database):***

* ***Relational Database Management System (RDBMS):*** MySQL is an open-source RDBMS that organizes data into tables with rows and columns, and supports SQL queries for data manipulation.
* ***Schema Definition:*** MySQL requires defining a schema that outlines the structure of the database, including tables, columns, data types, and relationships between tables.
* ***Query Processing:*** MySQL processes SQL queries efficiently, utilizing indexes and query optimization techniques to retrieve data quickly.
* ***Concurrency Control:*** MySQL implements concurrency control mechanisms to ensure data consistency and isolation in multi-user environments.

***JSON Web Token (JWT) (Authentication):***

* ***Token-Based Authentication:*** JWT is a compact, URL-safe token format used for authentication and authorization.
* ***Stateless Authentication:*** JWT enables stateless authentication, where the server does not need to store session data for each user.
* ***Payload:*** JWT consists of three parts: header, payload, and signature. The payload contains claims or attributes about the user.
* ***Cryptographic Signing:*** JWT tokens are digitally signed by the server to ensure their integrity and prevent tampering.

***React Query (Data Fetching):***

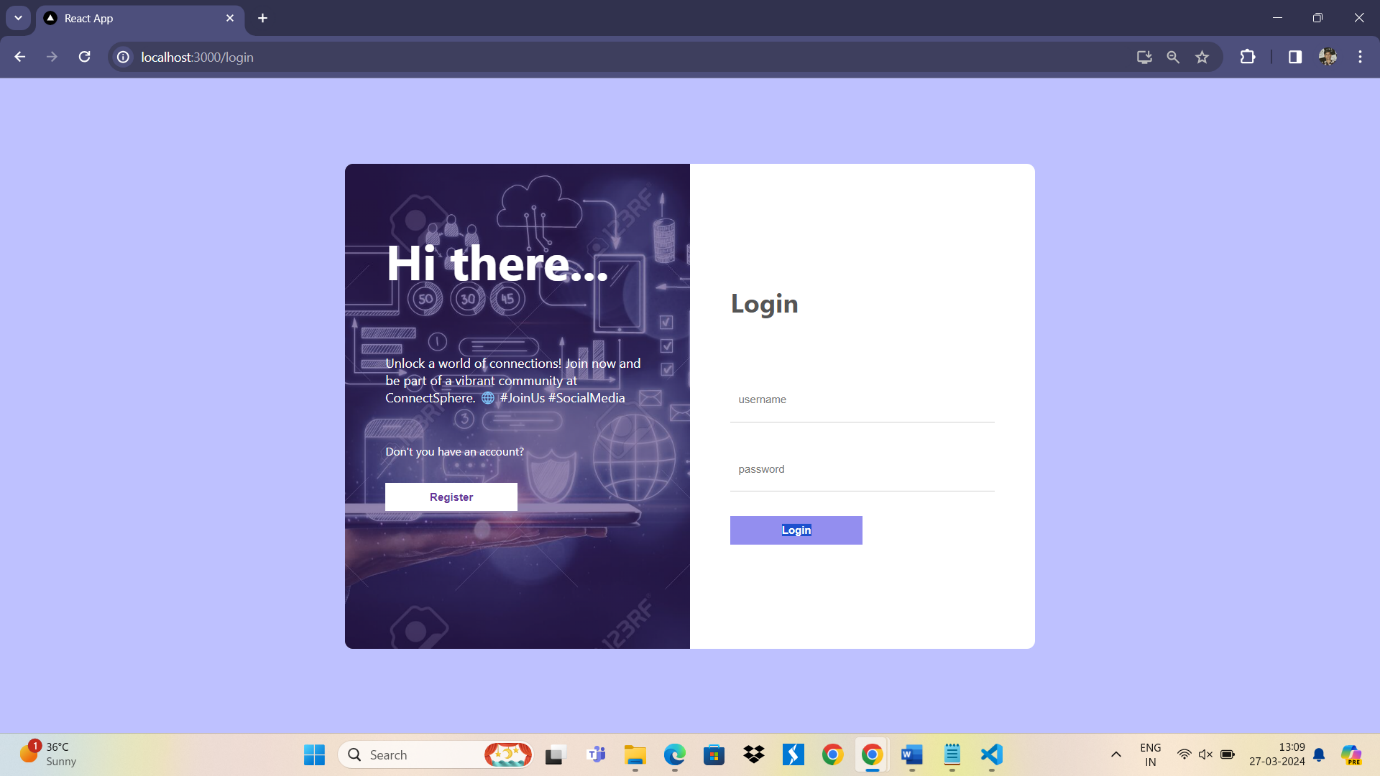
* ***Data Fetching and Caching:*** React Query provides hooks for fetching and caching data from APIs in React applications.
* ***Query Invalidation:*** React Query automatically invalidates and refetches stale data based on configurable criteria, such as cache timeouts or manual triggers.
* ***Optimistic Updates:*** React Query supports optimistic updates, allowing UI to be updated optimistically before the server responds, providing a smoother user experience.

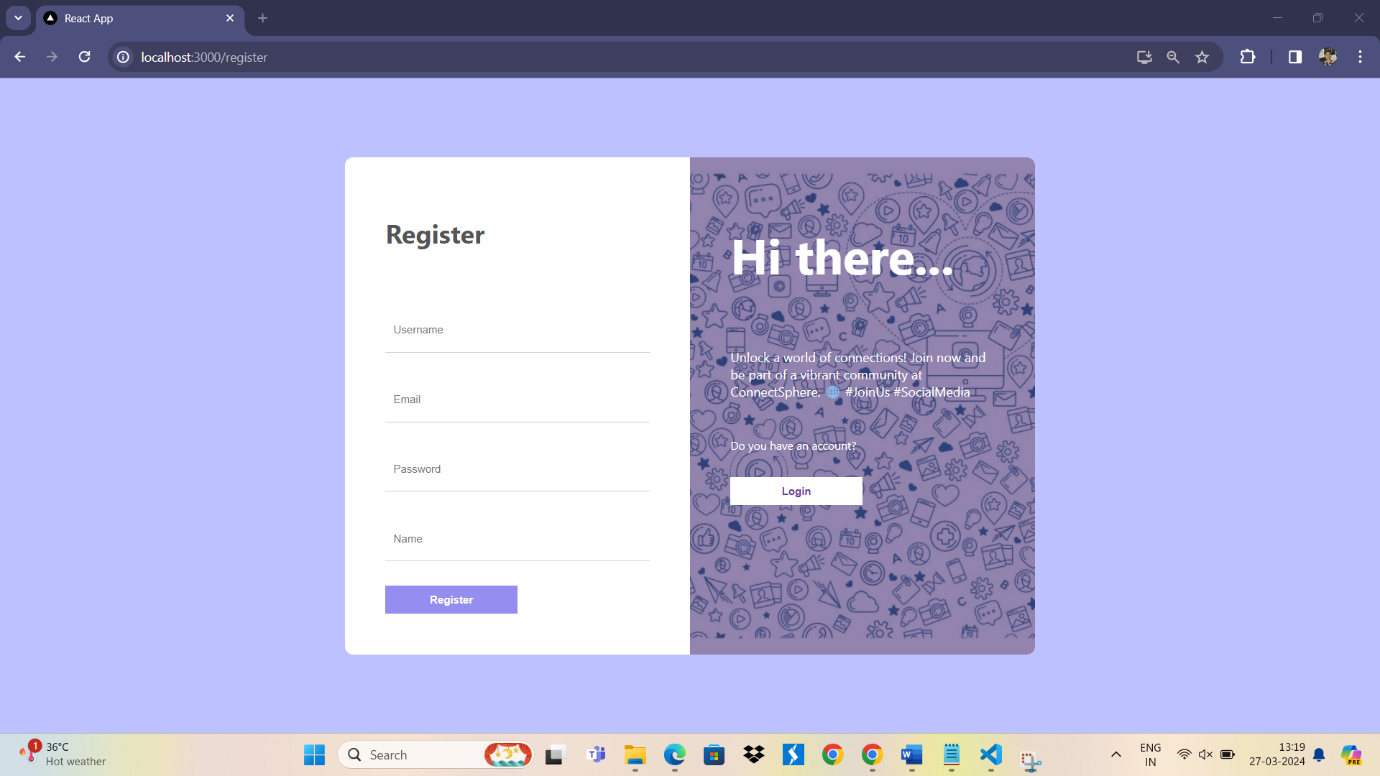
***5.Features***

1. *User Authentication and Authorization*

User registration and login functionality with secure password hashing.

Authorization mechanisms to control access to certain features or content based on user roles.

figure5.1 Login Page

figure5.2 Register Page

The Login component represents the login page of social media app. It allows users to authenticate and access their accounts.

***State Management:***

Utilizes the useState hook to manage state variables:

inputs: Tracks the values of username and password input fields.

err: Stores any error messages encountered during the login process.

***Navigation:***

Uses the useNavigate hook from React Router to handle navigation within the app.

***Event Handling:***

Defines a handleChange function to update the inputs state when user inputs change.

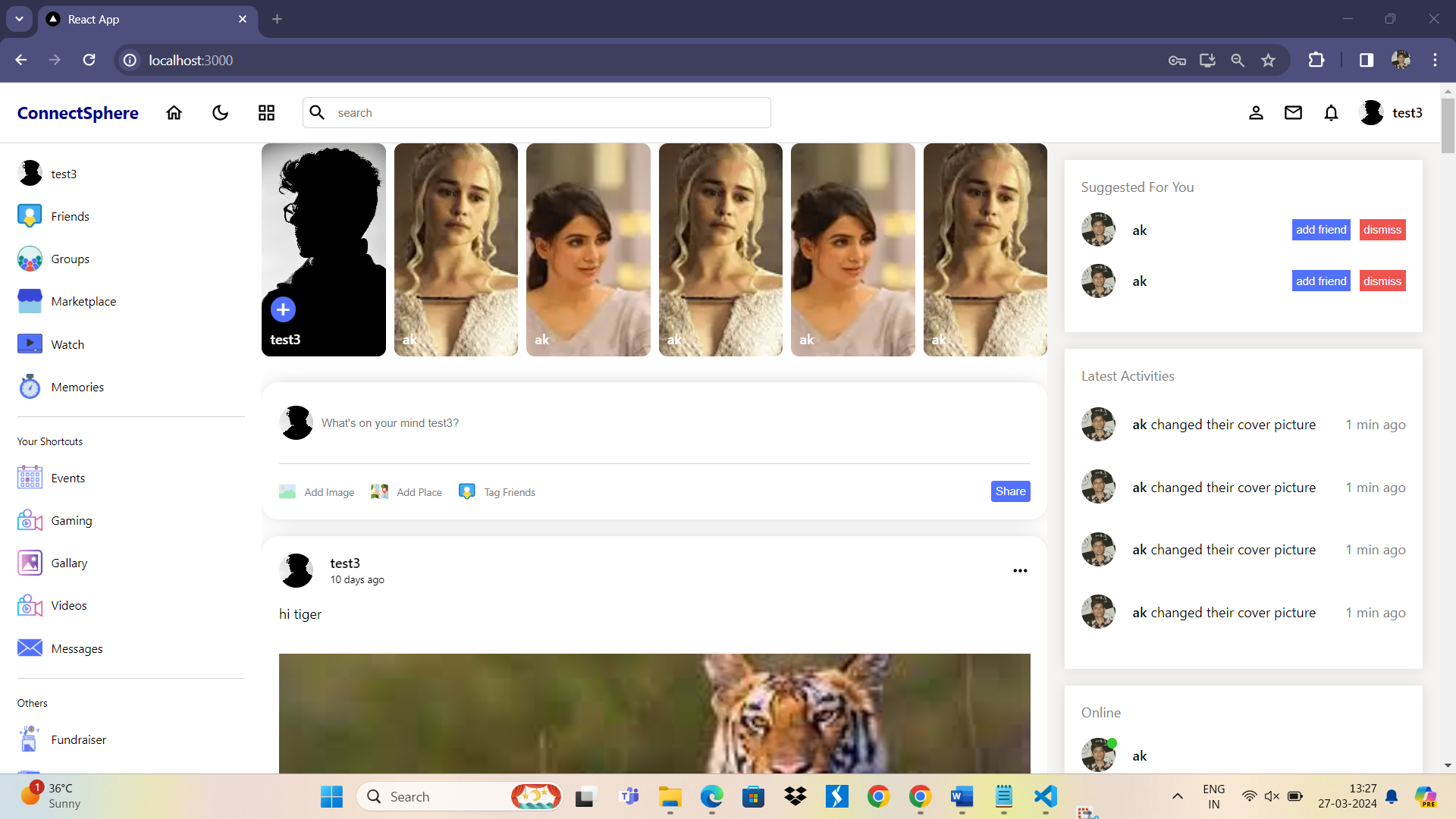
Implements a handleLogin function to handle the login process when the user submits the login form.

Calls the login function from the AuthContext to authenticate the user.

If successful, navigates the user to the home page ("/").

If an error occurs, displays the error message in the UI.

1. Dashboard

figure5.3 Dashboard

The dashboard serves as the central hub for users to access and interact with various features and functionalities of the ConnectSphere social media platform. Designed with user convenience and engagement in mind, the dashboard provides a comprehensive overview of personalized content, social interactions, and account management options.

1. DarkMode / LightMode

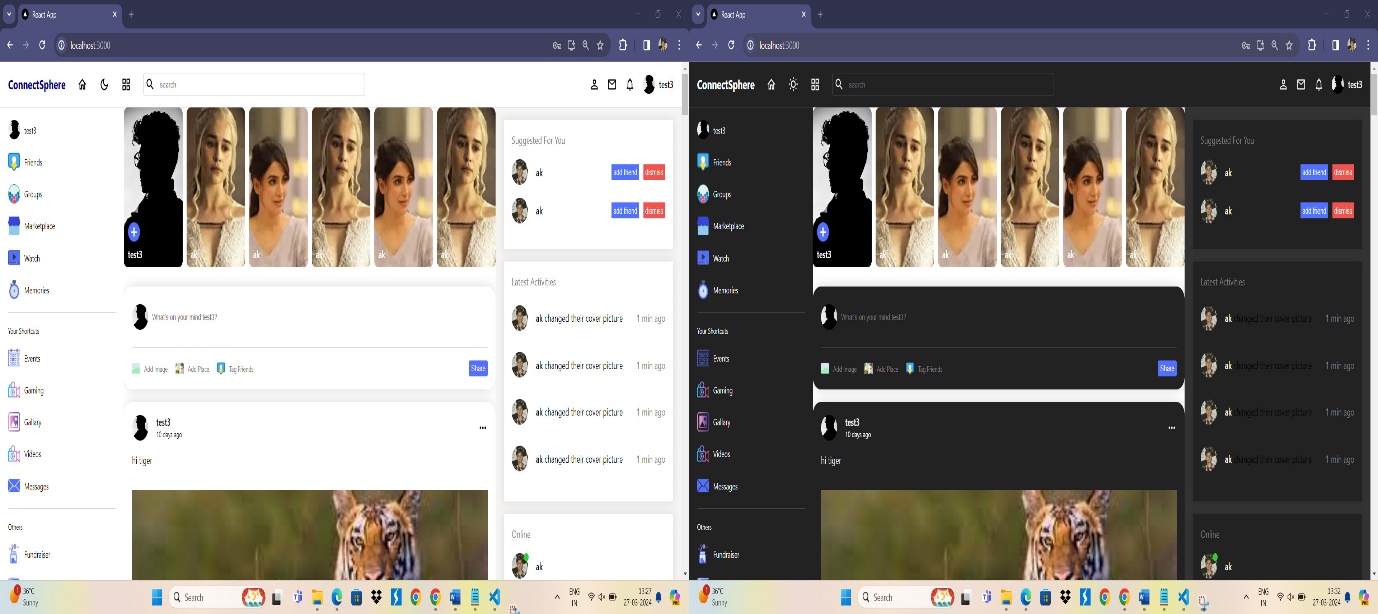


Figure5.4 Dark/Light Mode

***DarkModeContextProvider Component:***

The DarkModeContextProvider component is responsible for managing the dark/light mode state and providing it to child components within the application.

It receives children as a prop, representing the child components wrapped by the provider.

***State Management:***

Utilizes the useState hook to define a state variable darkMode, which stores the current mode (true for dark mode, false for light mode).

The initial state is determined by retrieving the darkMode value from local storage using localStorage.getItem. If no value is found, the default state is set to false (light mode).

***Toggle Functionality:***

Defines a toggle function that toggles the darkMode state between true and false when called.

This function is passed down to child components via the context value, allowing them to trigger the mode toggle.

***Effect Hook:***

Uses the useEffect hook to persist the darkMode state to local storage whenever it changes.

This ensures that the user's preferred mode is maintained across page refreshes and sessions.

***DarkModeContext Context:***

Creates a context object DarkModeContext using the createContext function.

This context provides a way to pass the darkMode state and toggle function down the component tree.

***Context Provider:***

Wraps the child components with the DarkModeContext.Provider component, passing down the darkMode state and toggle function as context values.

Child components can consume these values using the useContext hook or by accessing the context directly.

***Usage:***

To enable dark/light mode in your application, wrap the root component or relevant sections with the DarkModeContextProvider.

Child components can then access the darkMode state and toggle function via the DarkModeContext context, allowing them to adjust their appearance and behaviour accordingly.