# SOCIAL MEDIA ANALYTICS DASHBOARD

# A FULL STACK PROJECT REPORT

# Submitted by

#### GIRIDHARAN S

#### (22ITR025)

# KAVIYA S

#### (22IT044)

#### JAGAN S

#### (22ITL122)

#### 

#### *in partial fulfillment of the requirements*

#### *for the award of the degree*

#### *of*

#### BACHELOR OF TECHNOLOGY

**IN**

**INFORMATION TECHNOLOGY**

## DEPARTMENT OF INFORMATION TECHNOLOGY

<

width1.00” >



# KONGU ENGINEERING COLLEGE

# (Autonomous)

**PERUNDURAI ERODE – 638 060**

**NOVEMBER 2024**

## DEPARTMENT OF INFORMATION TECHNOLOGY

## KONGU ENGINEERING COLLEGE

**(Autonomous)**

**PERUNDURAI ERODE – 638060**

**NOVEMBER 2024**

**BONAFIDE CERTIFICATE**

This is to certify that the Project report entitled **SOCIAL MEDIA ANALYTICS DASHBOARD** is the bonafide record of project work done by **GIRIDHARAN S (22ITR025), KAVIYA S (22ITR044)** and **JAGAN S (22ITL122)** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in **INFORMATION TECHNOLOGY** of Anna university, Chennai during the year 2024-2025.

**SUPERVISOR HEAD OF THE DEPARTMENT**

**(Signature with seal)**

Date:

Submitted for the end semester viva voce examination held on \_\_\_\_\_\_\_\_\_\_\_ .

**INTERNAL EXAMINER**   **EXTERNAL EXAMINER**

## DEPARTMENT OF INFORMATION TECHNOLOGY

## KONGU ENGINEERING COLLEGE

**(Autonomous)**

**PERUNDURAI ERODE – 638060**

**NOVEMBER 2024**

**DECLARATION**

We affirm that the Project Report titled **SOCIAL MEDIA ANALYTICS DASHBOARD** being submitted in partial fulfillment of the requirements for the award of Bachelor of Technology is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**Date: GIRIDHARAN S**

**(22ITR025)**

**KAVIYA S**

**(22ITR089)**

**JAGAN S**

**(22ITL122)**

I certify that the declaration made by the above candidates is true to the best of my knowledge.

**Date: Name and Signature of the Supervisor with seal**

## ABSTRACT

In today’s digital landscape, understanding social media performance is crucial for brands, influencers, and individuals alike. This Social Media Analytics Dashboard is a full-stack web application designed to provide users with a comprehensive, centralized view of their social media metrics. Built using the MERN stack (MongoDB, Express.js, React.js, Node.js), this project integrates with popular social media platforms like YouTube and Twitter, allowing users to monitor key performance indicators in real time. The application fetches data via APIs from each platform, consolidating metrics such as follower count, engagement rates, likes, and post interactions. A backend built with Node.js and Express.js manages API requests and data processing, storing essential metrics in MongoDB for historical tracking and easy access. The frontend, developed in React.js, provides a user-friendly interface featuring interactive graphs, charts, and tables, giving users an intuitive experience for tracking and comparing metrics across platforms.

## ACKNOWLEDGEMENT

## First and foremost, we acknowledge the abundant grace and presence of Almighty throughout different phases of the project and its successful completion.

## We wish to express our gratefulness to our beloved Correspondent Thiru.A.K.ILANGO B.Com., M.B.A., LLB., and all the trust members of Kongu Vellalar Institute of Technology Trust for providing all the necessary facilities to complete the project successfully.

## We express our deep sense of gratitude to our beloved Principal Dr.V.BALUSAMY B.E.(Hons)., M.Tech., Ph.D., for providing us an opportunity to complete the project.

## We express our gratitude to Dr. S. ANANDAMURUGAN M.E., Ph.D., Head of the Department, Department of Information Technology for his valuable suggestions.

## We are thankful to our Project Coordinators Dr. G.K. KAMALAM BE., ME., PhD., and Ms. R. SANDHIYA BE., ME., for the valuable guidance and support to complete our project successfully.

## We are highly indebted to Dr. S. VARADHAGANAPATHY BE.,ME.,PhD., Department of Information Technology for her valuable supervision and advice for the fruitful completion of the project.

## We are thankful to the faculty members of the Department of Information Technology for their valuable guidance and support.

## TABLE OF CONTENTS

| **CHAPTER No** | **TITLE** | **PAGE No** |
| --- | --- | --- |
|  | **ABSTRACT** | **iv** |
|  | **LIST OF FIGURES** | **viii** |
|  | **LIST OF ABBREVIATIONS** | **ix** |
| **1.** | **INTRODUCTION** | **1** |
|  | 1.1 INTRODUCTION | 1 |
|  | 1.2 OBJECTIVE | 1 |
| **2.** | **SYSTEM SPECIFICATION** | **2** |
|  | 2.1 HARDWARE REQUIREMENTS | 2 |
|  | 2.2 SOFTWARE REQUIREMENTS | 2 |
|  | 2.3 SOFTWARE DESCRIPTION | 3 |
|  | 2.3.1 Visual Studio Code | 3 |
|  | 2.3.2 NodeJS | 4 |
|  | 2.3.3 MongoDB | 5 |
|  | 2.3.4 ReactJS | 6 |
| **3.** | **SYSTEM DESIGN** | **9** |
|  | 3.1 USE CASE DIAGRAM | 9 |
|  | 3.2 CLASS DIAGRAM | 10 |
|  | 3.3 SEQUENCE DIAGRAM | 11 |
|  | 3.4 ACTIVITY DIAGRAM | 12 |
|  | 3.5 DATABASE DESIGN | 13 |
|  | 3.6 [MODULES DESCRIPTION](#_heading=h.35nkun2) | 14 |
|  | 3.6.1 Authentication Module | 14 |
|  | 3.6.2 YouTube Module | 14 |
|  | 3.6.3 Twitter Module | 15 |
| **4.** | **SYSTEM TESTING** | **16** |
|  | 4.1 [SYSTEM TESTING](#_heading=h.3as4poj) | 16 |
|  | 4.2 [UNIT TESTING](#_heading=h.2jxsxqh) | 16 |
|  | 4.3 [MODULE TESTING](#_heading=h.z337ya) | 17 |
|  | 4.4 [INTEGRATION TESTING](#_heading=h.3j2qqm3) | 17 |
|  | 4.5 [VALIDATION TESTING](#_heading=h.1y810tw) | 17 |
|  | **RESULTS** | **18** |
|  | **CONCLUSION AND FUTURE WORK** | **19** |
|  | [**APPENDIX 1- CODING**](#_heading=h.4i7ojhp) | **20** |
|  | [**APPENDIX 2- SNAPSHOTS**](#_heading=h.1pxezwc) | **37** |
|  | **REFERENCES** | **41** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE No.** | **FIGURE NAME** | **PAGE No.** |
| 3.1 | USE CASE DIAGRAM | 9 |
| 3.2 | CLASS DIAGRAM | 10 |
| 3.3 | SEQUENCE DIAGRAM | 11 |
| 3.4 | ACTIVITY DIAGRAM | 12 |
| A2.1 | LOGIN PAGE | 22 |
| A2.2 | SIGNUP PAGE | 25 |
| A2.3 | DASHBOARD PAGE | 27 |
| A2.4 | YOUTUBE STATISTICS PAGE | 29 |
| A2.5 | YOUTUBE VIDEOS PAGE | 29 |
| A2.6 | YOUTUBE LIKES PAGE | 34 |
| A2.7 | TWITTER PAGE | 32 |
| A2.8 | TWITTER GRAPH PAGE | 32 |

# LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| **CSS** | Cascading Style Sheets |
| **HTML** | HyperText Markup Language |
| **JS** | JavaScript |
| **API** | Application Programming Interface |
| **URL** | Uniform Resource Locator |

## CHAPTER 1

## INTRODUCTION

## INTRODUCTION

In an era where social media serves as a primary channel for communication, marketing, and brand-building, the ability to monitor and analyze social media performance is essential. Both businesses and individuals rely on insights from social platforms to understand audience engagement, measure growth, and optimize content strategy. However, managing and interpreting data across multiple social media platforms can be challenging and time-consuming. This Social Media Analytics Dashboard project addresses this need by providing a unified platform that aggregates, visualizes, and stores social media data across various platforms, including Twitter and YouTube. The dashboard offers a comprehensive solution for users to track their social media metrics and gain valuable insights to inform their strategies.

## OBJECTIVE

The project aims to create a centralized, user-friendly tool that allows users to monitor and manage their social media metrics across multiple platforms in real time. By the key performance indicators from platforms such as Twitter and YouTube, this dashboard enables users to track and analyze their engagement, follower growth, and content performance from a single interface. It aims to facilitate data-driven decision-making by visualizing historical trends and displaying insights through interactive charts and graphs. Additionally, the project prioritizes accessibility and security, ensuring that only authorized users can access their sensitive data. By achieving these goals, the Social Media Analytics Dashboard provides users with the insights they need to optimize their social media presence and effectively grow their online influence.

## CHAPTER 2

## SYSTEM SPECIFICATION

* 1. **HARDWARE SPECIFICATION**

**Processor** : Intel i3

**Processor Speed** : 250MHz to 667MHz

**RAM** : 4GB RAM

**Hard Disk** : 256GB

**Keyboard** : Standard 104 enhanced

**Mouse** : Local PS/2

## SOFTWARE REQUIREMENTS

## Platform : Visual Studio Code

## Server-Side Script : NodeJS

## Database : MongoDB

## Library : ReactJS

## SOFTWARE DESCRIPTION

## Visual Studio Code

## Visual Studio Code is a versatile and user-friendly code editor used by developers worldwide. It supports a wide range of programming languages and frameworks, including JavaScript, React, and Node.js. For this project, VS Code provides a convenient environment for writing, editing, and debugging code related to the server-side scripting with Node.js, as well as client-side scripting with React.js. Its built-in terminal and debugging tools streamline the development process, while its customizable features allow developers to tailor the editor to their specific needs, enhancing productivity and efficiency throughout the project lifecycle.

## ES6+ React/Redux/React-Native

## ES6 React/Redux/React-Native snippets provide code templates designed to simplify and accelerate the development process for building applications with React, Redux, and React Native. These snippets utilize the powerful features introduced in ES6, allowing developers to quickly scaffold common patterns like React components, Redux actions, reducers, and React Native components. By incorporating ES6 syntax, such as arrow functions, destructuring, template literals, and modules, these snippets make it easier to write clean and concise code. With ES6 snippets, developers can efficiently generate standardized code structures that promote consistency and reduce the chance of errors, cutting down on boilerplate code. This collection of snippets supports the creation of both web applications in React and cross-platform mobile applications in React Native, while simplifying state management in Redux. Whether developing user interfaces, managing state, or working on cross-platform features, these ES6 snippets help developers stay focused on creating features and solving problems instead of repeating boilerplate setup.

## NodeJS

Node.js is a powerful server-side JavaScript runtime environment commonly used for building scalable and efficient web applications. In our project, Node.js serves as the server- side scripting language, handling tasks such as routing, handling HTTP requests, and interacting with the MongoDB database. Its event-driven architecture and non-blocking I/O operations make it well-suited for handling concurrent connections and processing requestsefficiently. With its vast ecosystem of libraries and frameworks, Node.js allows for rapid development and deployment of server-side applications. Additionally, its compatibility with Visual Studio Code provides a seamless development experience, enabling developers to write, test, and debug server-side code effectively within a unified environment. Overall, Node.js plays a crucial role in this project by powering the backend infrastructure and facilitating the seamless integration of frontend and backend components.

### Bcryptjs

### Bcryptjs is a library used for hashing passwords in JavaScript. It provides a secure way to hash passwords before storing them in a database or comparing them during authentication processes. By using bcryptjs, developers can ensure that user passwords are encrypted and protected against common security threats like brute force attacks and rainbow table attacks.

### Express

### Express is a minimalist web application framework for Node.js. It provides a robust set of features for building web servers and APIs, including routing, middleware support, template engines, and HTTP utilities. With its simple and flexible design, Express enables developers to quickly create scalable and maintainable web applications.

### Jsonwebtoken

### JSON Web Token is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. JWTs are commonly used for authentication and authorization in web applications. The jsonwebtoken library provides utilities for generating, parsing, and verifying JWTs, making it easy to implement token-based authentication and authorization systems.

### Nodemon

### Nodemon is a utility that monitors for changes in Node.js applications and automatically restarts the server when changes are detected. It eliminates the need to manually stop and restart the server every time a code change is made, improving developer productivity and workflow efficiency. Nodemon supports features like watching for file changes, ignoring specific files or directories and running custom scripts before and after restarting the server, making it an essential tool for Node.js development.

1. **Puppeteer**

Puppeteer is a Node.js library that provides a high-level API to control headless browsers, such as Google Chrome or Chromium, using the DevTools protocol. It's commonly used for web scraping.

1. **Node-fetch**

Node-fetch is a lightweight module that brings the Fetch API to Node.js, allowing you to make HTTP requests, just like the fetch API in the browser. It's particularly useful for applications requiring data fetching, such as REST API calls, web scraping, or interacting with external resources.

### MongoDB

### MongoDB is a flexible and scalable NoSQL database platform ideal for managing complex data structures, making it invaluable for our project's ecommerce website. With MongoDB's document-oriented architecture, we can efficiently store and manage diverse types of data relevant to our ecommerce platform, such as user information, appointment details, and messages. Its schema-less nature allows for dynamic and evolving data models, accommodating the dynamic nature of ecommerce data. For this project, MongoDB's scalability is crucial as it can seamlessly handle increasing volumes of data and traffic, ensuring optimal performance even during peak usage periods. This scalability ensures that the website remains responsive and can accommodate growth without sacrificing performance. MongoDB's powerful query language and indexing capabilities enable us to perform complex queries efficiently, facilitating tasks such as retrieving user information, appointment details, and message history. This enhances the overall user experience by providing fast and relevant data retrieval. Additionally, MongoDB's support for high availability and automatic failover ensures that our ecommerce website remains accessible and reliable. This feature minimizes downtime and ensures uninterrupted service for users, which is essential for maintaining customer satisfaction and trust. Overall, MongoDB provides a robust and scalable e database solution that empowers us to create a secure, efficient, and highly performant ecommerce website. Its flexibility, scalability, and powerful features make it an indispensable component of our project, enabling us to meet the diverse data management needs of our ecommerce platform effectively.

## ReactJS

React, a widely-used JavaScript library, plays a crucial role in our ecommerce project. Its component-based architecture facilitates the creation of reusable UI elements, simplifying the development of complex interfaces typical in ecommerce websites. React's virtual DOM optimizes performance by minimizing full page reloads, ensuring a seamless user experience even under high traffic. Furthermore, React's extensive community support provides access to a plethora of third-party libraries and tools, streamlining the development process and enhancing productivity. Additionally, React's flexibility allows for seamless integration with other libraries such as Redux, enabling efficient state management for handling complex data in our ecommerce platform. Overall, React significantly contributes to improving user experience and efficiency in our ecommerce project.

### axios:

Axios is a promise-based HTTP client for making HTTP requests in browser-based and Node.js applications. It provides a simple and intuitive API for performing asynchronous HTTP requests, handling request and response data, and intercepting request and response lifecycle events. Axios supports features like request and response interception, request cancellation, automatic JSON data parsing, and more, making it a popular choice for handling AJAX requests in JavaScript applications.

### react-dom:

React DOM is a package that serves as the entry point for working with the DOM in React applications. It provides methods for rendering React components into the DOM, updating component state, handling events, and interacting with the browser's Document Object Model (DOM). React DOM is essential for building React applications that interact with the user interface and manipulate the DOM based on user interactions and data changes.

### react-redux:

React Redux is the official React bindings for Redux, a predictable state container for JavaScript applications. It provides a set of APIs and components that simplify the process of integrating Redux with React applications, including <Provider> component for passing the Redux store to React components, connect() function for connecting React components to the Redux store, and <Selector> component for selecting data from the Redux store. React Redux enables developers to manage application state more efficiently and maintain application consistency across components.

### react-router-dom:

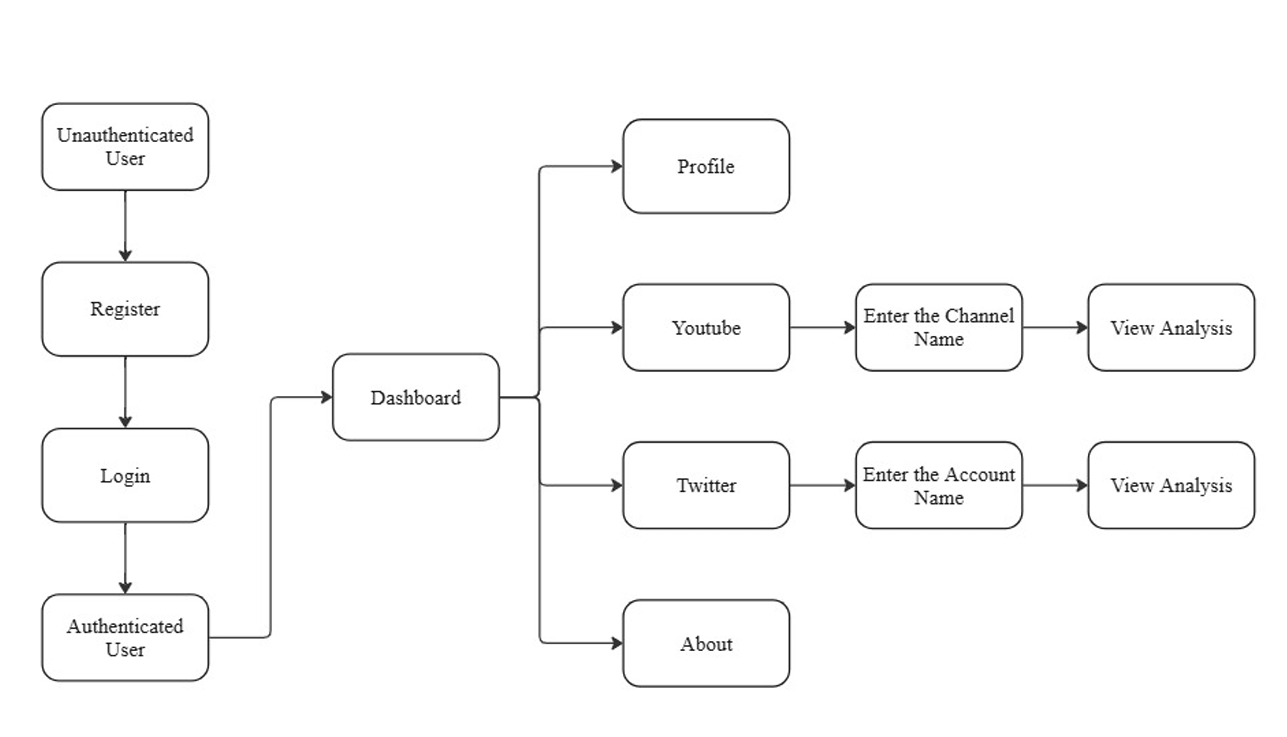
React Router DOM is a package that provides declarative routing for React applications. It allows developers to define routes and navigation logic using components and props, making it easy to create single-page applications with multiple views. React Router DOM provides a simple and intuitive API for defining route configurations, handling navigation events, and passing route parameters, making it a popular choice for managing client-side routing in React applications.

## CHAPTER 3

## SYSTEM DESIGN

* 1. **USE CASE DESIGN**

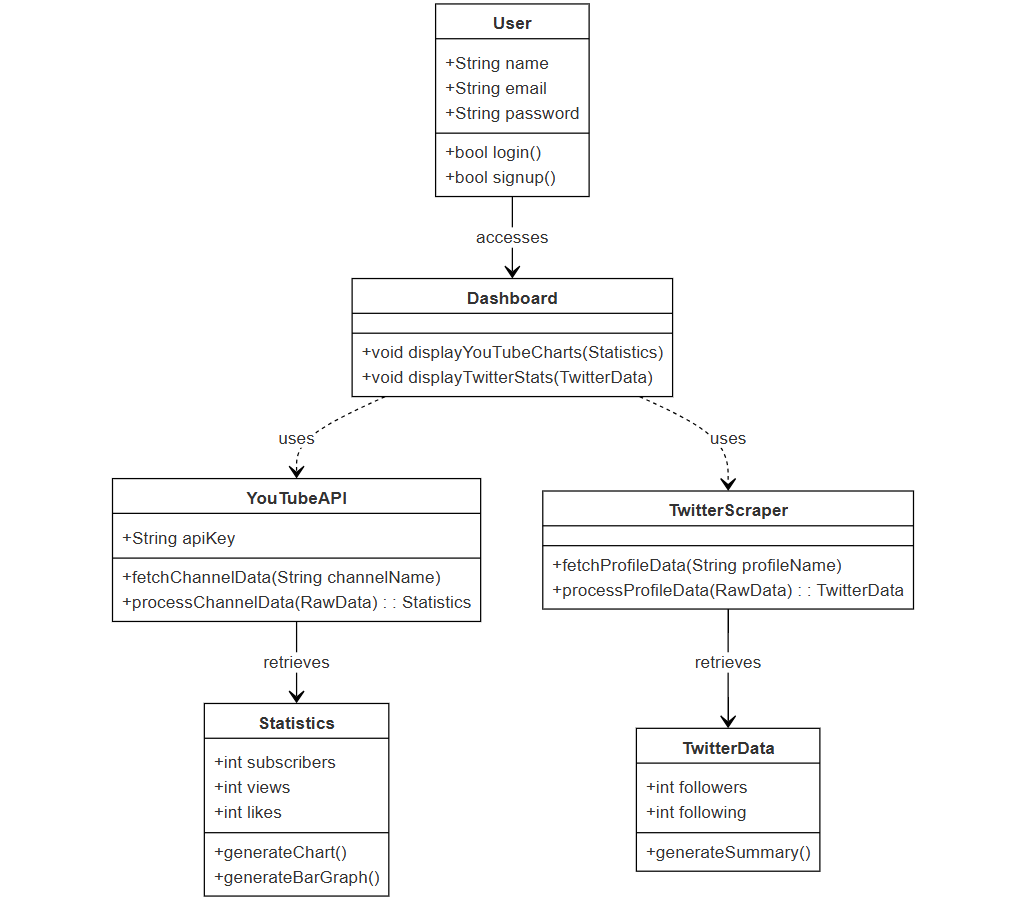
A use case diagram is a dynamic or behavior diagram in UML (Unified Modelling Language).Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions and functions that the system needs to perform. A "system" is something being developed or operated, here a website. The "actors" are people or entities operating under defined roles within the system as shown below in Figure 3.1.



**Fig 3.1 Use Case Diagram**

## CLASS DIAGRAM

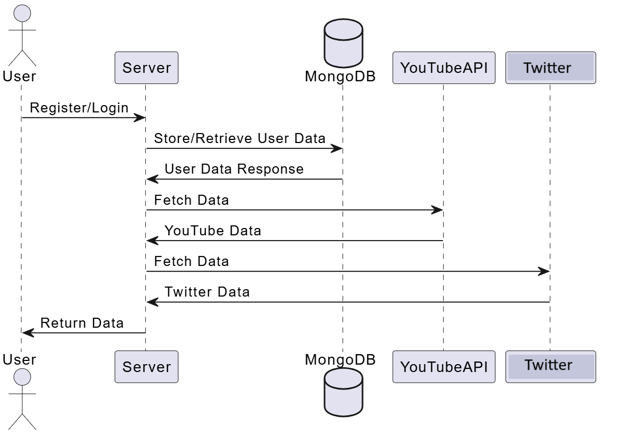
A class diagram is a static structure diagram in UML (Unified Modeling Language) that illustrates the structure of a system by showing the classes of the system, their attributes, methods, and relationships between them. Classes represent the blueprint for objects, encapsulating data and behavior. Attributes are the data members of a class, while methods represent the operations that can be performed on the class's objects.



**Fig 3.2 Class Diagram**

## SEQUENCE DIAGRAM

A sequence diagram is a form of interaction diagram which shows objects as lifelines running down the page, with their interactions over time represented as messages drawn as arrows from the source lifeline to the target lifeline. Sequence diagrams are good at showing which objects communicate with which other objects and what messages trigger those communications as shown below in Figure 3.3. Sequence diagrams are not intended for showing complex procedural logic.



**Fig 3.3 Sequence Diagram**

## 3.4. ACTIVITY DIAGRAM

In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities. The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched or concurrent and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc. as shown in the Figure 3.4 below.

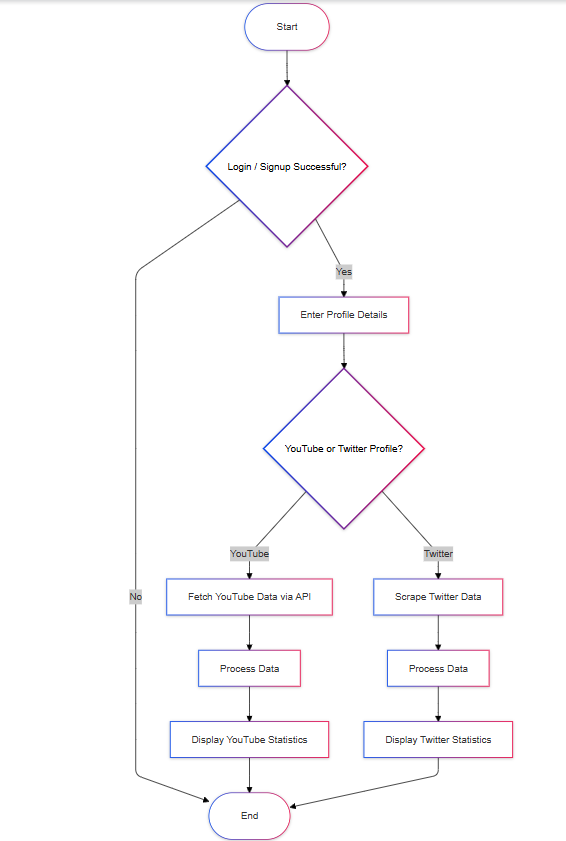


Fig 3.4 Activity Diagram

## DATABASE DESIGN

* + 1. **USER SCHEMA**

const userSchema = new mongoose.Schema({

  email: {

type: String,

required: true,

unique: true

},

  password: {

type: String,

required: true

},

});

const User = mongoose.model('User', userSchema);

## MODULES DESCRIPTION

## AUTHENTICATION MODULE

## This module handles how you log in to the Social Media Analytics Dashboard and create an account. It's like the security guard at the door, making sure only authorized users can access their contact information.

## Login: You enter your email and password to access your existing account. The system checks if your credentials are correct and allows you to proceed if they are.

## Signup: If you're new, you can create an account by providing your email and entering a secure password. The system ensures your email is unique and stores your information securely.

## Security: The module uses password hashing to protect your password from being stolen. This means even if someone gets access to the database, they can't see your actual password.

## 3.6.2 YOUTUBE MODULE

This module provides comprehensive analytics for YouTube channels for the entered channel.

**Key Metrics:** Channel name, Subscriber count, Total video views, Number of uploaded videos, Recent videos' analytics (views, likes).

**Functionality:** It uses YouTube Data API v3 to fetch the channel's information and video metrics. This module will display insights into the channel's performance and the engagement level of the latest videos.

## 3.6.3 TWITTER MODULE

This module provides analytics for Twitter profile for which the user enters in the search box.

**Key Metrics:** Username, Number of posts, Number of followers, Number of followings, Recent tweets' engagement (likes, comments, reposts, views), Joined date.

**Functionality:** It uses web scraping techniques to fetch data directly from Twitter using Puppeteer and displays a detailed profile overview along with the analytics of the latest tweets.

## CHAPTER 4

**SYSTEM TESTING**

## 4.1 SYSTEM TESTING

## The goal of the software business worldwide has always been to offer software products of the highest quality with distinctive characteristics. The team however, cannot ensure these aspects without testing software components under a variety of anticipated and unforeseen circumstances. The process of detecting flaws in a developed product is called software testing. Additionally, it helps in the detection of flaws, gaps and missing requirements by determining whether the actual findings can be reconciled with the anticipated outcomes.

## The last step before a product is introduced to the market is testing. It involves looking at, analyzing, observing, and rating several features of a product. Software testing is crucial because it allows any faults or errors in the software to be found early and fixed before the software product is delivered. Reliability, security and high performance are all ensured by thoroughly tested software, which also leads to saving time, cost effectiveness, and customer pleasure. Four phases are included in system testing.

* + - Unit testing
    - Module testing
    - Integration testing
    - Validation testing

## UNIT TESTING

Unit testing focuses on individual functions and modules, such as API fetch requests, data parsing functions, and visualization components, to ensure that each piece operates independently without errors. For example, unit testing verifies that individual API calls return expected data structures and that data rendering components correctly display this data in charts and tables.

## MODULE TESTING

Module testing assesses each social media analytics module (YouTube, Twitter) separately to verify that each component functions as intended. This includes testing the retrieval of metrics, the processing of engagement data, and the correct display of information on the dashboard for each platform. Module testing ensures that specific modules work reliably on their own, without interference from other parts of the system.

## 

## INTEGRATION TESTING

Integration testing checks how the social media modules work together and with the backend to provide a unified user experience. This testing phase examines data flow between the frontend and backend, ensuring, for example, that data fetched by the backend from social media APIs is correctly processed, stored, and rendered on the frontend. It also verifies that interactions between different modules are smooth and that user actions are accurately reflected across the application.

## VALIDATION TESTING

Validation testing ensures that the Social Media Analytics Dashboard meets the intended requirements and provides a positive user experience. This includes verifying data accuracy in the analytics presented, confirming responsiveness and usability across devices, and ensuring overall functionality aligns with the end-user's expectations. Validation testing is the final quality assurance step before deployment, ensuring that the dashboard operates as a reliable and informative tool for monitoring social media performance across platforms.

## CHAPTER 5

## RESULTS

The Social Media Analytics Dashboard provides a powerful and centralized solution for tracking and analyzing social media performance across multiple platforms. The application’s frontend, developed in React, delivers an interactive and responsive user experience, allowing users to monitor key metrics and visualize data seamlessly. Each social media platform module, including YouTube, Twitter is integrated using respective APIs or scraping methods, offering users a real-time overview of essential metrics such as follower counts, post engagements, and recent activity. Secure login and authorization processes ensure data privacy, with user information stored in MongoDB to support scalable access and storage. The backend, built with Node.js and Express, facilitates efficient data processing and serves as a reliable API layer for the frontend, handling requests and data updates across social media platforms. By providing users with tools to analyze trends, identify engagement patterns, and track growth over time, the dashboard empowers businesses and individuals to make data-driven decisions that enhance their social media strategies. The application’s modular architecture, combined with adherence to best practices in software development, creates a maintainable and extensible codebase. This structure enables future updates, additional platform integrations, and feature expansions with minimal disruption. Overall, the Social Media Analytics Dashboard serves as an invaluable asset for users looking to optimize their social media presence and increase engagement through informed insights and streamlined management.

**CHAPTER 6**

## CONCLUSION AND FUTURE WORK

The Social Media Analytics Dashboard is an effective, centralized solution for tracking social media metrics across multiple platforms, including YouTube, Twitter. By leveraging the MERN stack, the application combines a responsive, user-friendly frontend with a secure and scalable backend, ensuring reliable data processing and visualization. With real-time analytics and key performance metrics at their fingertips, users gain valuable insights to enhance social media strategies, improve engagement, and track growth over time. The dashboard's modular design and adherence to best practices in development further reinforce its flexibility and maintainability, making it a versatile tool for businesses, influencers, and individuals.

Looking forward, the Social Media Analytics Dashboard could introduce more advanced features to meet evolving user needs. Expanding platform integrations to include additional social networks like Pinterest would broaden users' analytical scope. Incorporating predictive analytics and machine learning could enable users to anticipate trends and optimize posting times. Customizable dashboards and automated report generation would add a personalized touch, allowing users to tailor insights to their specific requirements. Real-time alerts for performance spikes and a dedicated mobile app would further enhance user experience, enabling more dynamic and accessible management of social media presence. Together, these enhancements would solidify the dashboard’s role as a comprehensive, cutting-edge solution for social media analytics and strategic growth.

## APPENDIX 1

## CODING

### App.js :

import React from 'react';

import { Routes, Route, Router, Navigate, useLocation } from 'react-router-dom';

import Dashboard from './components/Dashboard';

import YouTube from './components/YouTube';

import Login from './components/Login';

import Signup from './components/Signup';

import Navbar from './components/Navbar';

import About from './components/About';

import Twitter from './components/Twitter';

const App = () => {

const location = useLocation();

const isAuthPage = location.pathname === '/login' || location.pathname === '/signup';

return (

<div>

{!isAuthPage && <Navbar />}

<Routes>

<Route path="/dashboard" element={<Dashboard />} />

<Route path="/youtube" element={<YouTube />} />

<Route path="/twitter" element={<Twitter />} />

<Route path="/about" element={<About />} />

<Route path="/login" element={<Login />} />

<Route path="/signup" element={<Signup />} />

</Routes>

</div>

);

};

export default App;

### Dashboard.js :

import React from 'react';

import { Link } from 'react-router-dom';  // Add Link for navigation

import '../styles/Dashboard.css';

function Dashboard() {

    return (

        <div className="landing-page">

            <section className="features-section">

                <h2>Dashboard Highlights</h2>

                <div className="features">

                    <div className="feature">

                        <h3>Comprehensive Analytics</h3>

                        <p>Track follower growth, engagement rates, and post performance across all platforms.</p>

                    </div>

                    <div className="feature">

                        <h3>Real-Time Data</h3>

                        <p>Get up-to-the-minute updates on your social media statistics.</p>

                    </div>

                    <div className="feature">

                        <h3>Customizable Reports</h3>

                        <p>Create and share custom reports with your team for actionable insights.</p>

                    </div>

                    <div className="feature">

                        <h3>Data Visualization</h3>

                        <p>Use visually intuitive charts, graphs for data interpretation.</p>

                    </div>

                    <div className="feature">

                        <h3>Platform-Specific Metrics</h3>

                        <p>Each platform has unique metrics such as video views, subscriber count, Date joined, Followers count, Following count</p>

                    </div>

                    <div className="feature">

                        <h3>Comprehensive Analytics</h3>

                        <p>Track follower growth, engagement rates, and post performance across all platforms.</p>

                    </div>

                </div>

            </section>

            <footer className="footer">

                <p>&copy; 2024 Social Media Analytics Dashboard</p>

            </footer>

        </div>

    );

}

export default Dashboard;

### Login.js :

### import React, { useState } from 'react';

### import { useNavigate, Link } from 'react-router-dom';

### import '../styles/LoginSignup.css';

### const Login = () => {

### const [email, setEmail] = useState('');

### const [password, setPassword] = useState('');

### const [error, setError] = useState(null);

### const [loading, setLoading] = useState(false);

### const navigate = useNavigate();

### const handleSubmit = async (e) => {

### e.preventDefault();

### setLoading(true);

### setError(null);

### try {

### const response = await fetch('http://localhost:5000/login', {

### method: 'POST',

### headers: {

### 'Content-Type': 'application/json',

### },

### body: JSON.stringify({ email, password }),

### });

### const data = await response.json();

### if (response.ok) {

### localStorage.setItem('authToken', data.token);

### navigate('/dashboard');

### } else {

### setError(data.error || 'Login failed');

### }

### } catch (error) {

### console.error('Error during login:', error);

### setError('An error occurred. Please try again.');

### } finally {

### setLoading(false);

### }

### };

### return (

### <div className="login-container">

### <div className="auth-card">

### <form onSubmit={handleSubmit} className="auth-form">

### <h2>Login</h2>

### {error && <p className="error">{error}</p>}

### <input

### type="email"

### placeholder="Email"

### value={email}

### onChange={(e) => setEmail(e.target.value)}

### required

### />

### <input

### type="password"

### placeholder="Password"

### value={password}

### onChange={(e) => setPassword(e.target.value)}

### required

### />

### <button type="submit" disabled={loading}>

### {loading ? 'Logging In...' : 'Log In'}

### </button>

### <p className="login-link">

### Don't have an account? <Link to="/signup">Signup</Link>

### </p>

### </form>

### </div>

### </div>

### );

### };

### export default Login;

### Signup.js:

import React, { useState } from 'react';

import { Link } from 'react-router-dom';

const Signup = () => {

    const [name, setName] = useState('');

    const [email, setEmail] = useState('');

    const [password, setPassword] = useState('');

    const [error, setError] = useState('');

    const [loading, setLoading] = useState(false);

    const handleSubmit = async (e) => {

        e.preventDefault();

        setLoading(true);

        setError('');

        const userData = { name, email, password };

        try {

            const response = await fetch('http://localhost:5000/signup', {

                method: 'POST',

                headers: {

                    'Content-Type': 'application/json',

                },

                body: JSON.stringify(userData),

            });

            const data = await response.json();

            if (response.ok) {

                localStorage.setItem('authToken', data.token);

                alert('Signup successful!');

                window.location.href = '/login';

            } else {

                setError(data.error || 'Signup failed');

            }

        } catch (error) {

            console.error('Error:', error);

            setError('An error occurred. Please try again later.');

        } finally {

            setLoading(false);

        }

    };

    return (

        <div className="signup-container">

            <div className="auth-card">

                <h1 className="dashboard-title">Social Media Analytics Dashboard</h1>

                <form onSubmit={handleSubmit} className="auth-form">

                    <h2>Sign Up</h2>

                    <input

                        type="text"

                        placeholder="Name"

                        value={name}

                        onChange={(e) => setName(e.target.value)}

                        required

                    />

                    <input

                        type="email"

                        placeholder="Email"

                        value={email}

                        onChange={(e) => setEmail(e.target.value)}

                        required

                    />

                    <input

                        type="password"

                        placeholder="Password"

                        value={password}

                        onChange={(e) => setPassword(e.target.value)}

                        required

                    />

                    {error && <p className="error">{error}</p>}

                    <button type="submit" disabled={loading}>

                        {loading ? 'Signing Up...' : 'Sign Up'}

                    </button>

                    <p className="login-link">

                        Already a user? <Link to="/login">Login</Link>

                    </p>

                </form>

            </div>

        </div>

    );}

export default Signup;

### Dashboard.js :

import React from 'react';

import { Link } from 'react-router-dom';

import '../styles/Dashboard.css';

function Dashboard() {

    return (

        <div className="landing-page">

            <section className="features-section">

                <h2>Dashboard Highlights</h2>

                <div className="features">

                    <div className="feature">

                        <h3>Comprehensive Analytics</h3>

                        <p>Track follower growth, engagement rates, and post performance across all platforms.</p>

                    </div>

                    <div className="feature">

                        <h3>Real-Time Data</h3>

                        <p>Get up-to-the-minute updates on your social media statistics.</p>

                    </div>

                    <div className="feature">

                        <h3>Customizable Reports</h3>

                        <p>Create and share custom reports with your team for actionable insights.</p>

                    </div>

                    <div className="feature">

                        <h3>Data Visualization</h3>

                        <p>Use visually intuitive charts, graphs for data interpretation.</p>

                    </div>

                    <div className="feature">

                        <h3>Platform-Specific Metrics</h3>

                        <p>Each platform has unique metrics such as video views, subscriber count, Date joined, Followers count, Following count</p>

                    </div>

                    <div className="feature">

                        <h3>Comprehensive Analytics</h3>

                        <p>Track follower growth, engagement rates, and post performance across all platforms.</p>

                    </div>

                </div>

            </section>

            <footer className="footer">

                <p>&copy; 2024 Social Media Analytics Dashboard</p>

            </footer>

        </div>

    );

}

export default Dashboard;

### YouTube.js :

## import React, { useState } from 'react';

## import ChannelStatistics from './ChannelStatistics';

## import VideoLikesChart from './VideoLikesChart';

## import Videos from './Videos';

## import '../styles/YouTube.css';

## function YouTube() {

## const [channelData, setChannelData] = useState(null);

## const [channelName, setChannelName] = useState('');

## const fetchYouTubeData = async () => {

## if (!channelName) return;

## try {

## const response = await fetch(`http://localhost:5000/youtube/${channelName}`);

## const data = await response.json();

## console.log('Fetched Data:', data);

## setChannelData(data);

## } catch (error) {

## console.error('Error fetching YouTube data:', error);

## }

## };

## return (

## <div className="youtube-container">

## <video autoPlay loop muted className="video-background">

## <source src="D:\1\_credit\_project\smad\_0811\frontend\src\images\bg.mp4" type="video/mp4" />

## </video>

## <h1 className="youtube-title">YouTube Channel Analytics</h1>

## <div className='input-group'>

## <input

## type="text"

## value={channelName}

## onChange={(e) => setChannelName(e.target.value)}

## placeholder="Enter YouTube Channel Name"

## className="input-field"

## />

## <button onClick={fetchYouTubeData} className="fetch-button">

## Fetch Data

## </button>

## </div>

## {channelData && (

## <>

## <div className="card">

## <ChannelStatistics channelData={channelData} channelName={channelName} />

## </div>

## <div className="card">

## <Videos channelData={channelData} channelName={channelName} />

## </div>

## <div className="card">

## <VideoLikesChart videoStats={channelData.videoStats} />

## </div>

## </>

## )}

## </div>

## );}

## export default YouTube;

**Twitter.js**

import React, { useState } from 'react';

import axios from 'axios';

import PostsChart from './PostsChart';

import '../styles/Twitter.css';

function Twitter() {

  const [username, setUsername] = useState('');

  const [data, setData] = useState(null);

  const [statusMessage, setStatusMessage] = useState('');

  const fetchUserData = async () => {

    setStatusMessage('Fetching data...');

    setData(null);

    try {

      const response = await axios.post('http://localhost:5000/get\_profile', { username });

      setData(response.data);

      setStatusMessage('Data fetched successfully!');

    } catch (error) {

      setStatusMessage('Network slow. Please try again later.');

      console.error('Error fetching data', error);

    }

  };

  return (

    <div className="app-container">

      <h1>Twitter Profile Engagement</h1>

      <input

        type="text"

        placeholder="Enter Twitter username"

        value={username}

        onChange={(e) => setUsername(e.target.value)}

      />

      <button onClick={fetchUserData}>Fetch Profile</button>

      {statusMessage && <p className="status-message">{statusMessage}</p>}

      {data && (

        <div className="profile-info">

          <h2>Profile Information</h2>

          <div className="profile-info-grid">

            <div className="profile-photo-card">

              <img src={data.photo} alt="User Profile" className="profile-photo" />

              <h3>{data.name}</h3>

            </div>

            <div className="profile-stats">

              <p><strong>Posts:</strong> {data.posts}</p>

              <p><strong>Followers:</strong> {data.followers}</p>

              <p><strong>Following:</strong> {data.following}</p>

              <p><strong>Joined Date:</strong> {data.joined\_date}</p>

            </div>

          </div>

          <h3>Recent Tweets</h3>

          <div className="posts-section">

            {data.posts\_data.length > 0 ? (

              <PostsChart postsData={data.posts\_data} />

            ) : (

              <p>No posts available.</p>

            )}

          </div>

        </div>

      )}

    </div>

  );

}

export default Twitter;

**VideoLikesChart.js**

import React from 'react';

import { Bar } from 'react-chartjs-2';

import '../styles/YouTube.css';

function VideoLikesChart({ videoStats }) {

const videoLikesChartData = {

labels: videoStats?.map((video) => {

const title = video.title;

const firstSeparatorIndex = title.indexOf('|');

const lastSeparatorIndex = title.lastIndexOf('|');

if (firstSeparatorIndex === -1 || firstSeparatorIndex === lastSeparatorIndex) {

const words = title.trim().split(' ');

return words[words.length - 1];

} else {

return title.slice(firstSeparatorIndex + 1, lastSeparatorIndex).trim();

}

}) || [],

datasets: [

{

label: 'Top 5 Video Likes',

data: videoStats?.map((video) => video.likeCount) || [],

backgroundColor: 'rgba(255, 99, 132, 0.6)',

borderColor: 'rgba(255, 99, 132, 1)',

borderWidth: 2,

},

],

};

const formatNumber = (num) => {

if (num >= 1e6) {

return (num / 1e6).toFixed(1) + 'M';

} else if (num >= 1e3) {

return (num / 1e3).toFixed(1) + 'K';

}

return num;

};

return (

<div className="chart-container">

<h2>Top 5 Video Likes</h2>

<div className="chart">

<Bar

data={videoLikesChartData}

options={{

responsive: true,

plugins: {

legend: { display: true, position: 'top' },

},

scales: {

y: {

beginAtZero: true,

grid: { color: 'rgba(200, 200, 200, 0.3)' },

ticks: {

callback: (value) => formatNumber(value),

},

},

x: {

grid: { color: 'rgba(200, 200, 200, 0.3)' },

},

},

}}

/>

</div>

</div>

);

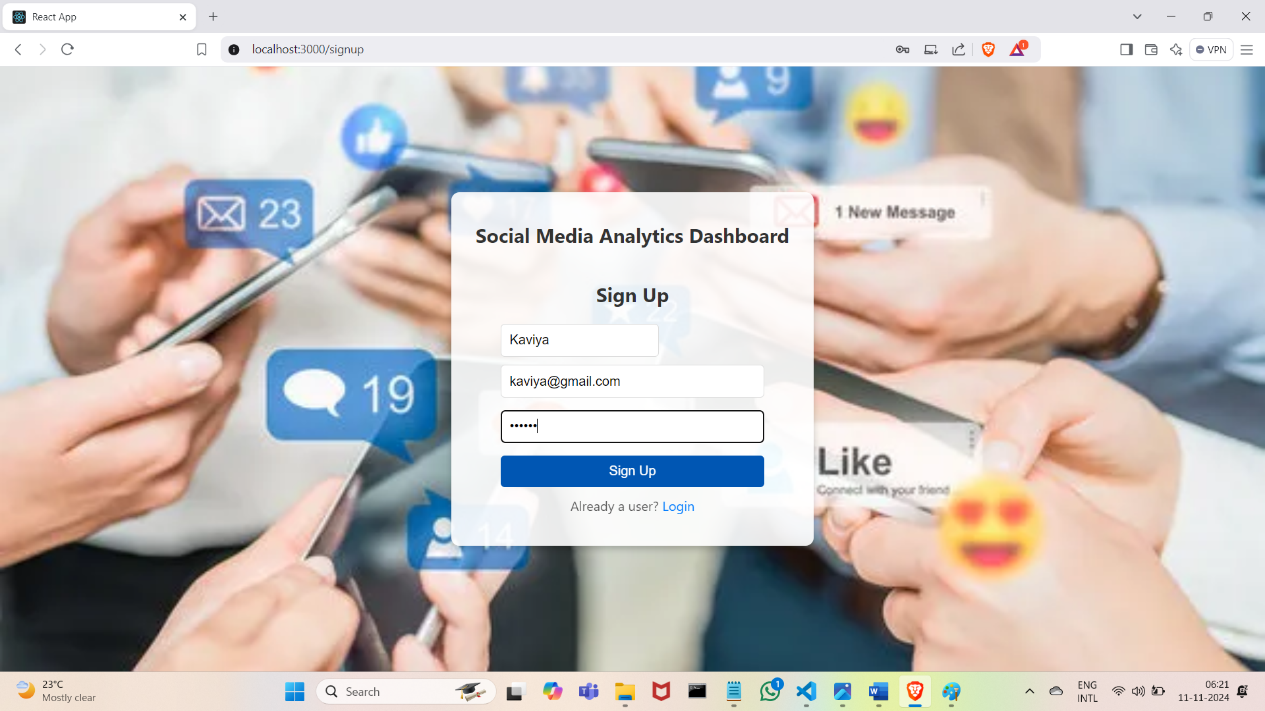
}

export default VideoLikesChart;

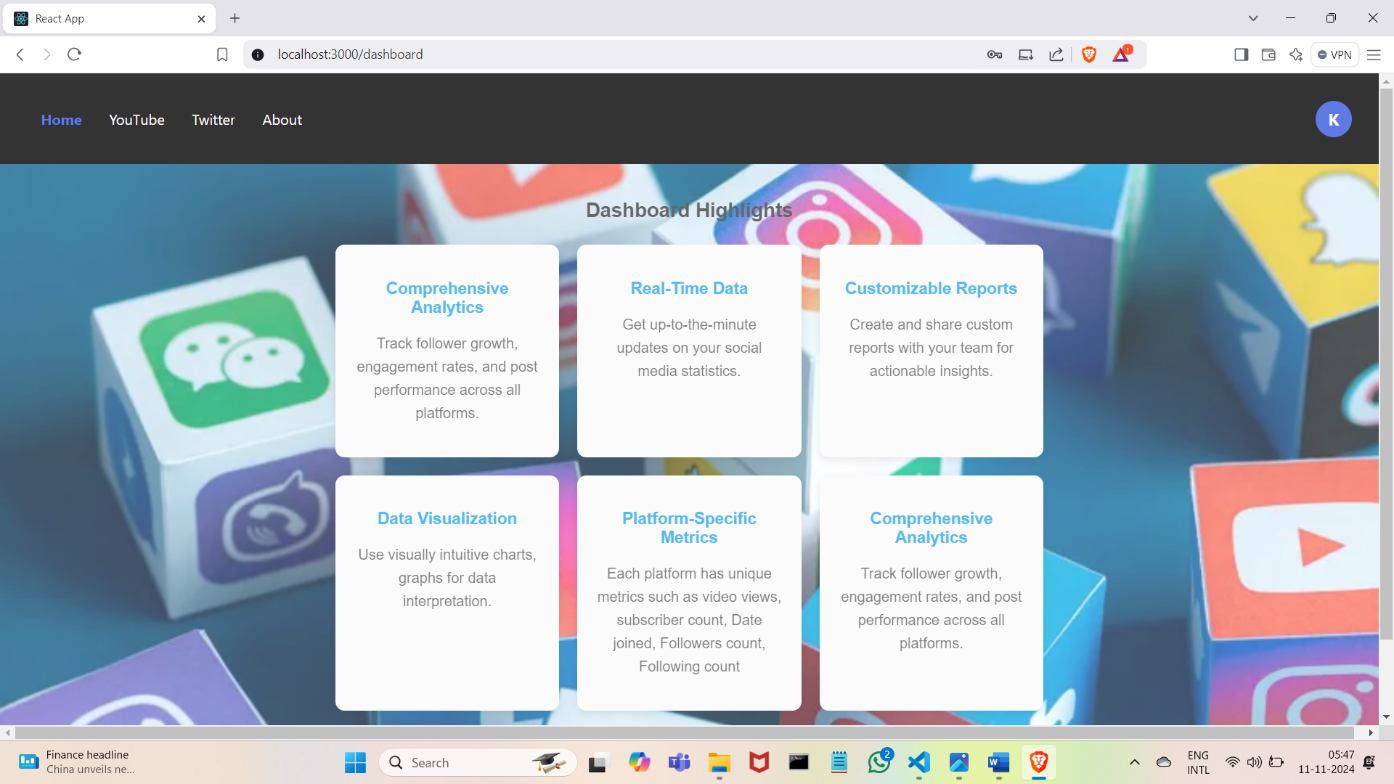
## APPENDIX 2 SNAPSHOTS



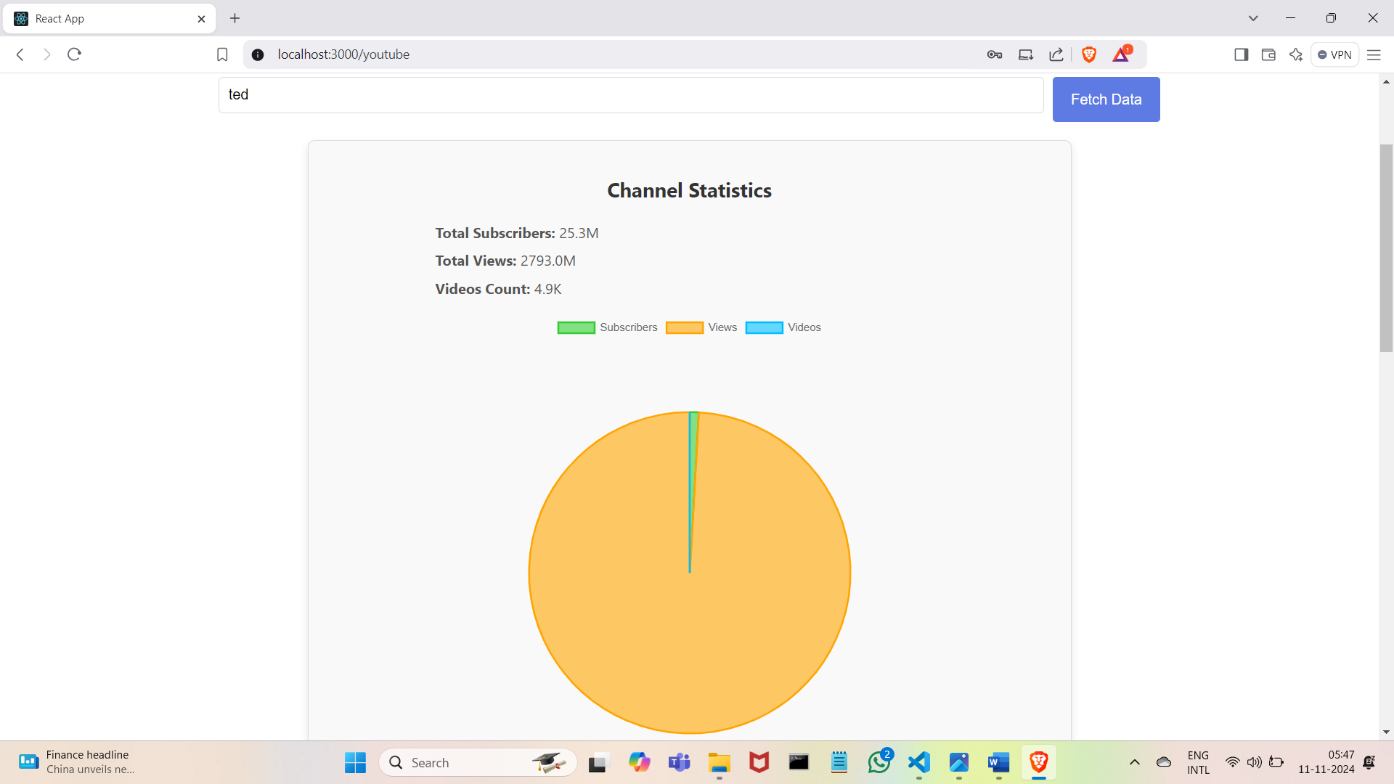
### Figure A2.1 Login Page

****

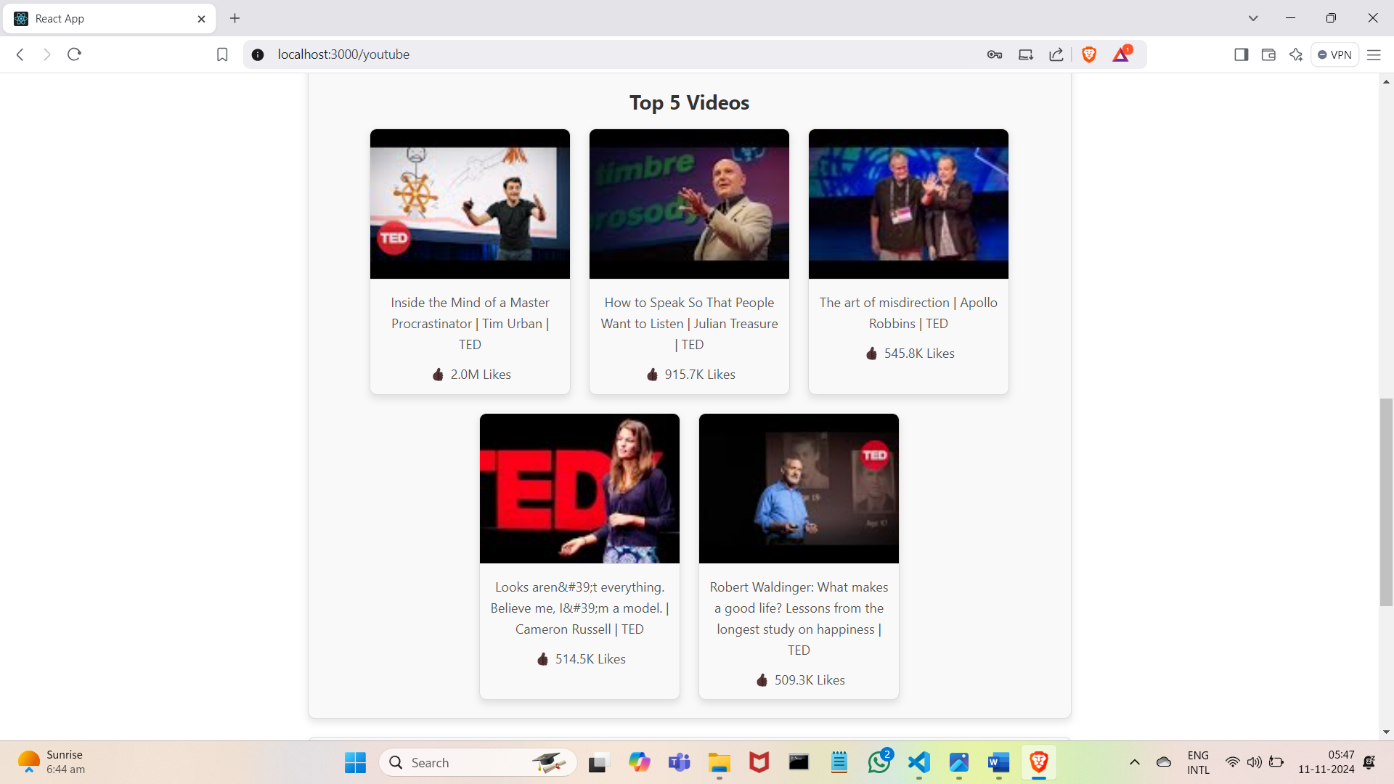
**Figure A2.2 Signup Page**



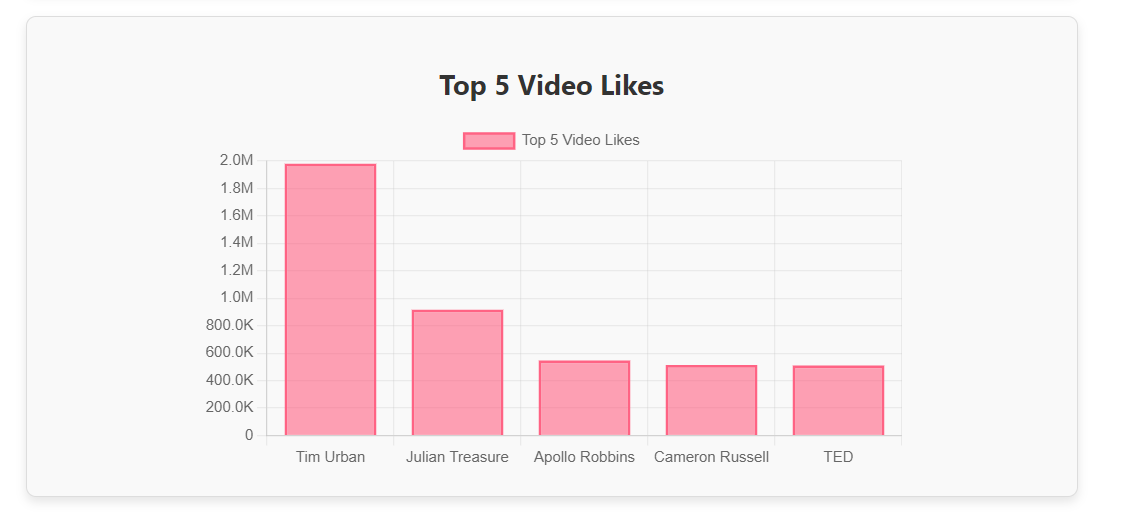
### Figure A2.3 Dashboard Page

****

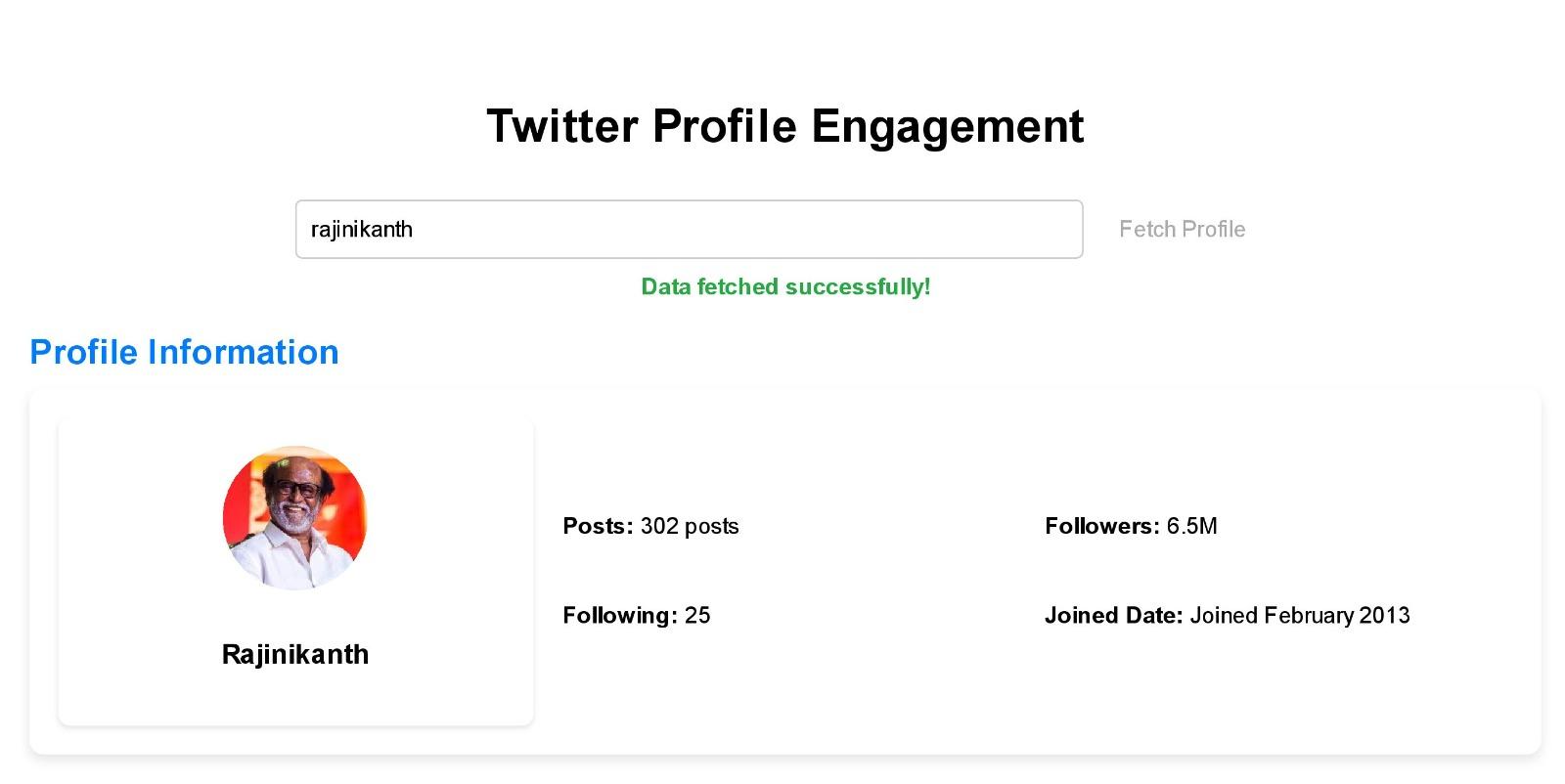
**Figure A2.4 YouTube Statistics Page**



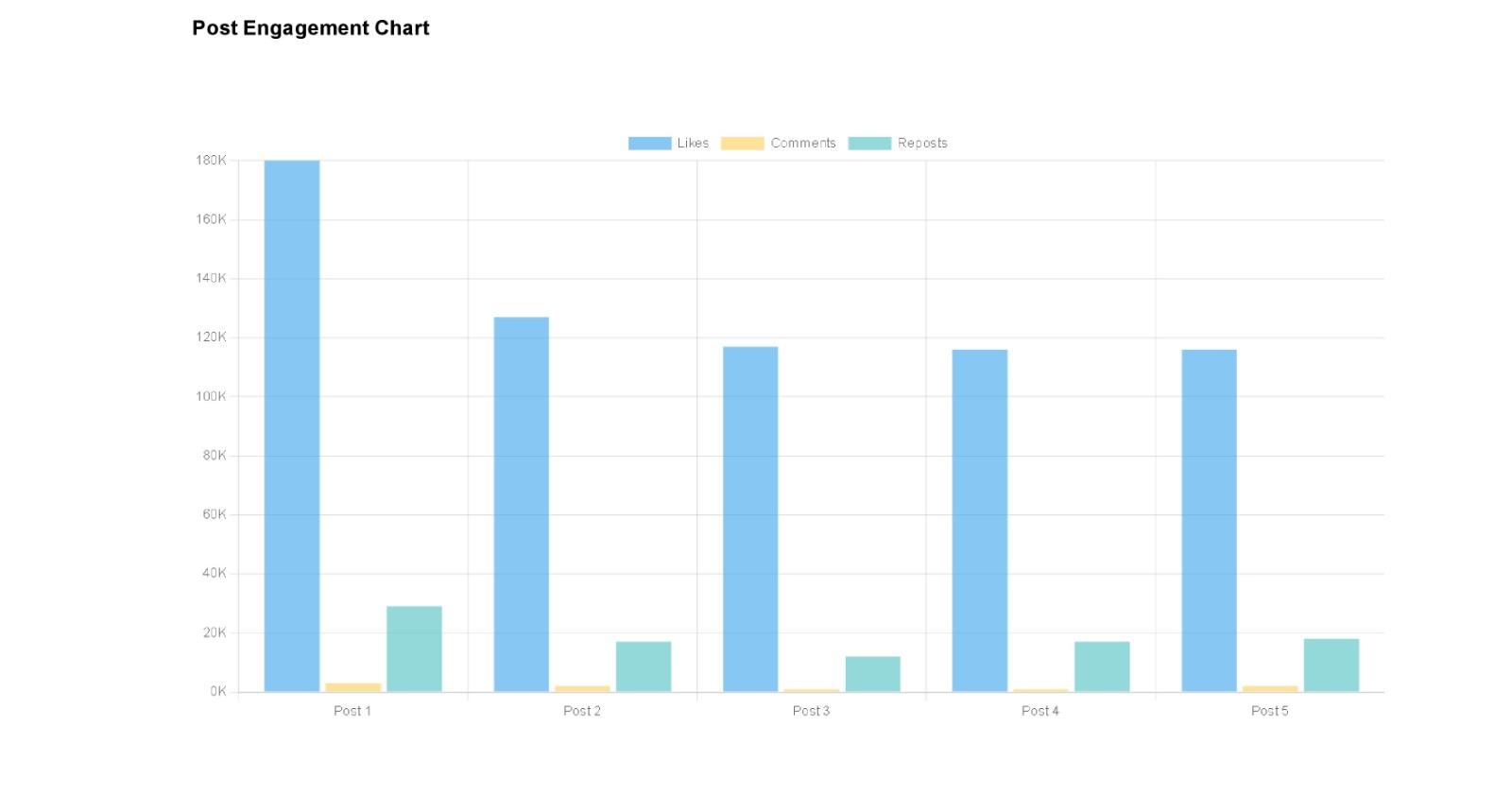
### Figure A2.5 YouTube Videos Page

****

**Figure A2.6 YouTube likes Page**



### Figure A2.7 Twitter page



### Figure A2.8 Twitter Graph page

## REFERENCES

https://expressjs.com/en/guide/using-middleware.html

https://[www.mongodb.com/developer/languages/javascript/getting-started-with-mongodb-](http://www.mongodb.com/developer/languages/javascript/getting-started-with-mongodb-) and-mongoose/

<https://github.com/GiridharanS1729/social-media-analytics-dashboard .git>