Chat App With Sentiment Analysis

Internship Project Report

Submitted
by

T.Rishendra, 21ECB0B48, NIT WARANGAL
V.Rahul, 21ECB0B65, NIT WARANGAL
Y.ArunSrivatsa, 21ECB0B67, NIT WARANGAL

Under the guidance of

Prof L. Anjaneyulu

Professor

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, WARANGAL

June 2023

NATIONAL INSTITUTE OF TECHNOLOGY

WARANGAL - 506 004

CERTIFICATE

This is to certify that V.Rahul, Y.Arun Srivatsa, T.Rishendra of National Institute of Technology have successfully completed a Project titled "Chat app with Sentiment Analysis", as part of Summer Internship Programme under my guidance at National Institute of Technology, Warangal, Telangana, during 20th May 2024 to 30th June 2024.

Prof L.Anjaneyulu, ECE

Visvesvaraya Centre for Skill Development

NIT, Warangal

Table of Contents:

Abstract

This project seeks to create a simple real-time chat application with sentiment analysis utilizing modern web development technologies. This application, which uses Next.js for server-side rendering, Pusher for real-time communication, Sentiment for mood detection, and React for component-based architecture, allows users to engage in live discussions while receiving instant feedback on the sentiment of their words. This tutorial explains how to set up, develop, and deploy a functional and interactive real-time chat system.

The necessity of a chat application with sentiment analysis :

In today's digital world, real-time communication has become essential for both personal and professional encounters. Instant messaging, online gaming discussions, and collaborative tools all rely significantly on real-time data sharing to deliver smooth user experiences. However, standard chat programmes frequently lack the ability to comprehend and interpret the emotional tone of a conversation, which can be critical for moderating debates, providing customer assistance, or just increasing user engagement.

- Integrating sentiment analysis into chat systems bridges this gap by analyzing the emotional content of communications in real time. This feature can help in a variety of ways.

- **Enhanced User Experience:** By giving users immediate feedback on the mood of communications, they may better understand the tone of their conversations, encouraging more thoughtful and empathetic communication.

- **Data Insights:** Sentiment analysis can provide valuable insights into user behavior and engagement, allowing businesses to tailor their services and increase customer satisfaction.

- Sentiment analysis can help with **mental health monitoring** in personal chat applications by detecting indicators of distress or depression in users' messages and providing timely treatment.

This project not only showcases the technical implementation of a real-time chat application, but it also emphasizes the significance of recognising and analyzing user sentiment in order to promote healthier and more engaging digital interactions.

Problem Statement

The demand for real-time communication solutions has increased in a variety of fields, including customer assistance, social networking, and team collaboration. However, designing such applications has numerous hurdles, including providing seamless real-time updates, maintaining performance, and including additional features such as sentiment analysis to improve user interaction. This project addresses these issues by developing a real-time chat programme that not only allows for instant conversation but also analyzes the sentiment of messages in real time.

Key Challenges:

Providing real-time updates.

- **Latency**: Keeping message delivery delays to a minimum is crucial for keeping the conversation flowing and users engaged.

- **Synchronization:** Keeping all participants' viewpoints synchronized, especially in big groups or high-traffic areas, is critical for avoiding confusion and ensuring everyone is on the same page.
  Performance:

- **Scalability:** As the number of users increases, the programme must handle the increasing load while maintaining speed and dependability.

- **Resource Management:** The efficient utilization of server and network resources to offer a smooth experience, particularly under high-traffic conditions.

Integration with Sentiment Analysis:

- **Real-Time Processing:** Sentiment analysis must be completed rapidly enough to provide immediate response without disrupting the discussion flow.

- **Accuracy:** In order to give relevant insights, the sentiment analysis system must accurately assess message tone and context.

- **User Acceptance:** Using sentiment analysis should improve the user experience without becoming intrusive or disrupting the natural flow of discourse.

Addressing the Challenges:

This project addresses these challenges by combining modern web development technologies with thoughtful design:

Technology Stack Used:

Next.js: Utilized for its server-side rendering capabilities, Next.js guarantees that websites load swiftly and effectively, increasing both performance and SEO.

Pusher: Pusher, a powerful real-time communication service, manages the complexities of managing WebSocket connections, assuring low-latency message delivery and synchronization.

React,js: React's component-based architecture enables modular and maintainable code, making it easier to build, update, and expand applications.

Efficient Data Handling: By reducing the quantity of data transferred over the network and optimizing data processing on both the client and server sides, the application performs well even under severe demand.

Load Balancing: Implementing load balancing strategies ensures that server resources are used efficiently and that no single server becomes a bottleneck.

User Interface Design: Sentiment analysis feedback is presented in an informative yet informative manner, so that it enriches rather than detracts from the user experience.

Context-Aware Analysis: Efforts are being made to fine-tune the sentiment analysis algorithm to account for context and reduce misinterpretations, hence enhancing feedback accuracy.

By solving these problems, this project intends to develop a powerful and user-friendly real-time chat programme that not only allows for instant communication but also improves user interactions through sentiment analysis. This technique not only fits the current demand for quick communication, but it also incorporates emotional intelligence, making digital conversations more insightful and engaging.

Project Description

The project entails developing a real-time chat application that uses a variety of cutting-edge web technologies to deliver a seamless and dynamic user experience. The primary purpose of this programme is to allow for immediate communication while also providing insights into the emotional tone of the conversation using sentiment analysis. Below is a full description of the application's functionality and the technologies used:

Application Features:

1. Real-Time Messaging.

- **Instant Communication**: Users can send and receive messages in real time, allowing for rapid feedback and conversation.
- **Live changes:** The application instantly changes the chat interface, eliminating the need for page refreshes and preserving the flow of conversation.

2. Sentimental Analysis:

- **Mood Detection:** The mood library analyzes each message's mood in real time.
- **Emoji Feedback:** Based on the sentiment analysis, an appropriate emoji is displayed alongside the message to reflect the mood (e.g., happy, sad, neutral).

3. Chat History

- **Message Storage:** The application keeps track of all chat messages, allowing users to browse back and review prior conversations.
- **Persistent Data:** Chat history is saved so that it can be retrieved even if the user logs out and back in.

4. User-friendly Interface:

- **Responsive Design:** The chat interface is responsive, meaning it works well on desktop and mobile devices.
- **Intuitive Layout:** The user interface is simple and straightforward, making it easy for users to explore and interact with the chat.

Technologies used

1. Next.js

- **Server-Side Rendering:** Next.js supports server-side rendering of React components, which enhances performance and SEO.
- **Routing and API Routes:** Includes built-in routing and API handling, making the development process easier.

2. Pusher:

- **Pusher** handles real-time updates and message synchronisation across clients.
- **WebSockets:** It uses WebSockets to ensure low-latency communication, which results in a seamless and responsive user experience.

3. Sentiment:

- **Text Analysis:** The Sentiment library is used to determine the emotional tone of texts.
- **Immediate Feedback:** The analysis is done in real time, so the programme can display relevant emojis right away.

4. React:

- **Component-Based Architecture**: React's component-based approach enables modular and maintainable code.
- **Dynamic UI:** With React, you can create dynamic user interfaces that update in real time.

5. Express:

- **Server Creation:** Express is used to build the server that processes API requests and serves the app.
- **Middleware Support:** Offers a strong framework for managing middleware, routing, and other server-side functionality.

6. Axios:

- **HTTP Requests:** Axios is used to send HTTP requests to APIs and get data.
- **Promise-Based:** Its promise-based architecture makes handling asynchronous processes easier.

7. Node.js:

- **Server Runtime:** The server's runtime environment is Node.js, which allows JavaScript to be used on the server side.
- **Efficient Performance:** Node.js is known for its non-blocking, event-driven architecture, which allows it to handle multiple connections efficiently.

Implementation Plan:

1. Setup.

- Create a new Next.js project.
- Install the necessary dependencies (Pusher, Sentiment, React, Express, Axios, Node.js).

2. Backend Development:

- Set up an Express server using Node.js.
- Integrate Pusher to provide real-time communication.
- Create API routes to handle chat messages and sentiment analysis.

3. Front End Development:

- Create React components for your chat interface.
- Pusher supports real-time message handling.
- Integrate the Sentiment library to analyse texts and display appropriate emojis.

4. Testing and deployment:

- Thoroughly test the application to ensure that real-time updates and sentiment analysis function properly.
- Deploy the application on a suitable hosting service.

By adhering to this strategy, the project hopes to produce a functional and engaging real-time chat application that improves user interaction through sentiment analysis, delivering quick feedback on the emotional tone of chats.

Objectives of the Project

The project seeks to create a full real-time chat platform that uses sentiment analysis to improve user interaction. The project's primary objectives are given below, each of which is critical to the application's overall functioning and user experience.

1. Create a real-time chat application using sentiment analysis.

- Real-Time Communication: The primary goal is to build a chat application that allows users to send and receive messages instantly. This ensures that conversations run smoothly without interruptions.
- Sentiment Analysis Integration: By including sentiment analysis into the chat, the programme can determine the tone and emotion behind each message. This function gives information about the emotional context of the conversation, making conversations more meaningful.

2. Provide a responsive and intuitive user interface.

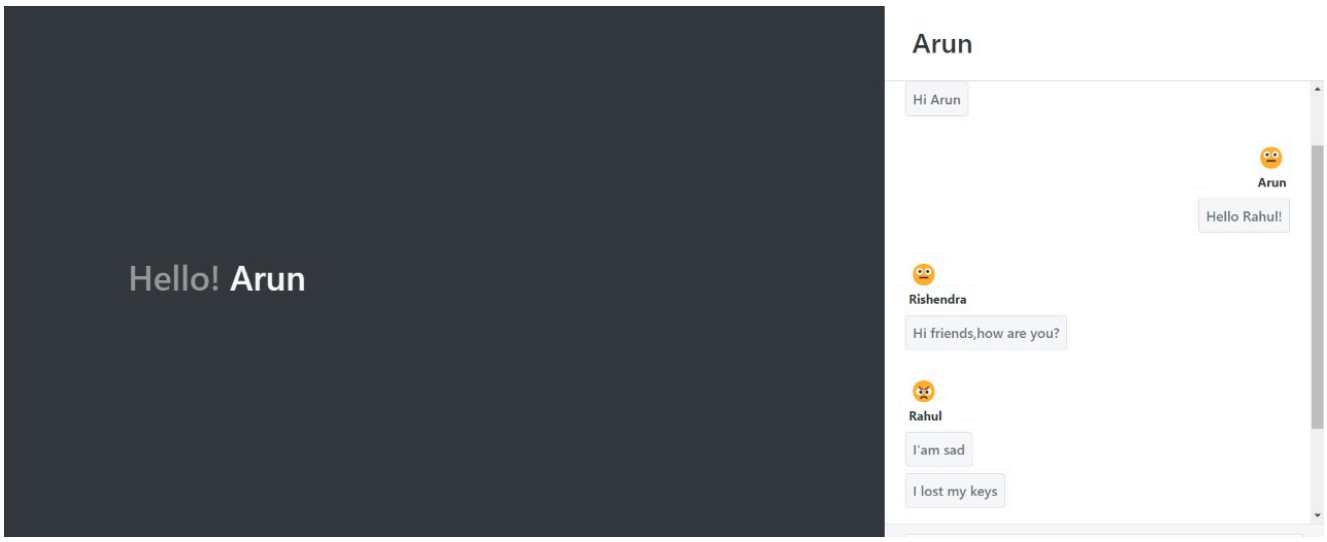
- Responsive Design: The application will be built to work fluidly on a variety of devices and screen sizes. This offers a consistent user experience across desktop, tablet, and mobile devices.
- Intuitive Navigation: The user interface will be built with usability in mind. Sending messages, viewing sentiment analysis, and accessing conversation history will all be easy to find and navigate.
- Aesthetically pleasing: The interface will be visually appealing, with a modern design that encourages user participation and satisfaction.

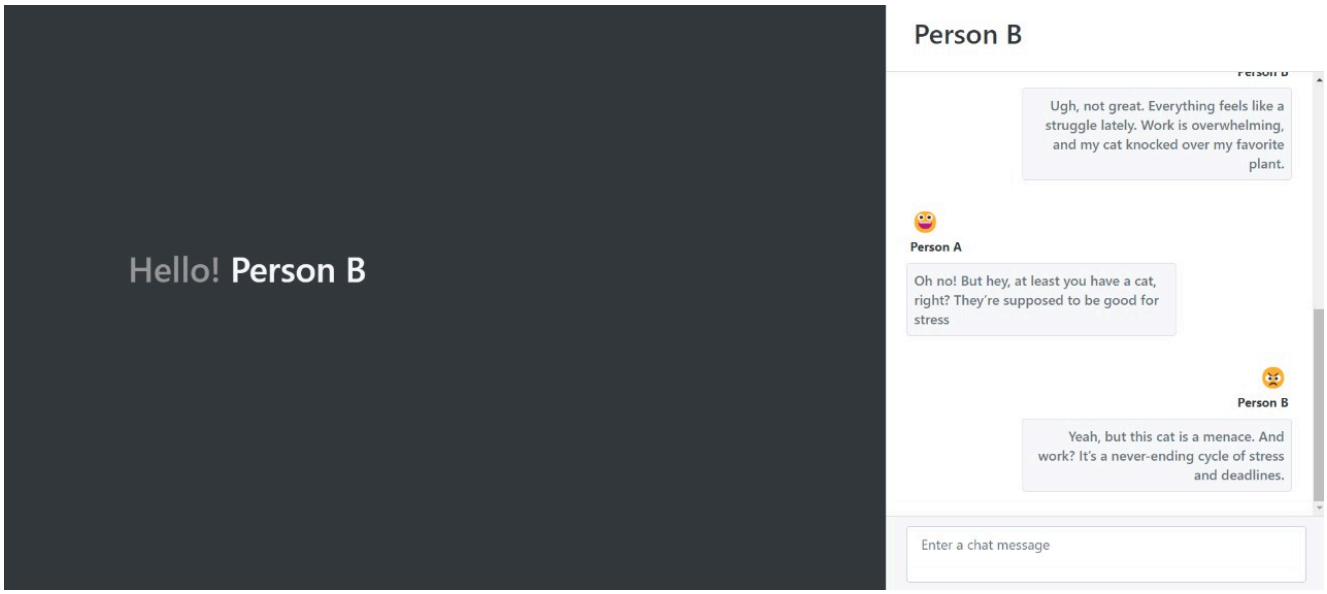
3. Use Pusher's WebSocket Technology for Real-Time Message Delivery. This ensures that messages are sent and received with minimal latency.

4. Analyse and display the sentiment of each message.

- Real-Time Sentiment Analysis: The Sentiment library will analyze each communication for sentiment (positive, neutral, or negative) in real time. This analysis will be completed quickly so that users receive fast feedback.
- Visual Feedback: The sentiment analysis results will be shown in an easily interpretable format, such as color-coded text or icons, to provide consumers a clear understanding of the tone of the conversation.

Results


Hello! Arun
Arun
Hi Arun
Arun
Hello Rahul!
Rishendra
Hi friends,how are you?
Rahul
I'am sad
I lost my keys


Hello! Person B
Person B
Ugh, not great. Everything feels like a struggle lately. Work is overwhelming, and my cat knocked over my favorite plant.
Person A
Oh no! But hey, at least you have a cat, right? They're supposed to be good for stress
Person B
Yeah, but this cat is a menace. And work? It's a never-ending cycle of stress and deadlines.
Enter a chat message

Conclusion

This project explains how to integrate real-time communication and sentiment analysis into a web application utilizing modern technologies such as Next.js, Pusher, Sentiment, and React.

- Pusher enables real-time communication, ideal for customer assistance, social networking, and team collaboration.
- Sentiment Analysis Integration: The Sentiment library enables sentiment analysis, which offers real-time feedback on the tone and emotional context of each communication, hence improving user interactions by providing insights into the mood of conversations.
- User Interface and Experience.: React ensures a dynamic and responsive user interface, resulting in a seamless user experience across multiple devices. The design emphasizes simplicity of navigation and visual feedback, making the application pleasant and simple to use.
- Maintaining Chat History: The application stores and retrieves chat history using persistent storage and user authentication, allowing users to access their conversation past and giving continuity in talks.
- Broader implications: Real-time feedback and sentiment analysis increase user interactions, with broad applications in customer service, social media, online communities, and mental health assistance.
- Future Prospects: Future enhancements could incorporate advanced sentiment analysis techniques, user customisation choices, increased security measures, and accessibility enhancements, thereby increasing the application's versatility and use.

This project uses Next.js, Pusher, Sentiment, and React to create a web application with real-time communication and sentiment analysis. It addresses issues with real-time updates, performance, and user engagement, offering a full solution for quick and intelligent communication tools.

References

1. Next.js Documentation: https://nextjs.org/docs
2. Pusher Documentation: https://pusher.com/docs
3. Sentiment GitHub Repository: https://github.com/thisandagain/sentiment
4. React Documentation: https://reactjs.org/docs/getting-started.html
5. Express Documentation: https://expressjs.com/
6. Axios Documentation: https://axios-http.com/docs/intro
7. Node.js Documentation: https://nodejs.org/en/docs/