IntelliChat

Front End Development (22CS021)

Submitted by

Nikhil Kumar(2310993889)
Nandan Sharma (2310993884)

Semester: 5

BE-CSE (Artificial Intelligence)

Guided by

Dr. Swati Malik

CHITKARA UNIVERSITY INSTITUTE OF ENGINEERING & TECHNOLOGY

CHITKARA UNIVERSITY, RAJPURA

September 2025

ACKNOWLEDGEMENTS

With immense please we**, Mr. Nikhil kumar, Mr. Nandan Sharma**, presenting the **“IntelliChat – Real-Time Messaging App with Integrated AI Analytics Assistant”** project report as part of the curriculum of **‘ BE-CSE (AI) ’**.

We would like to express our sincere thanks to **Dr. Swati Malik**, for his valuable guidance and support in completing our project.

We would also like to express our gratitude towards our dean **Dr. Sushil Narang** for giving us this great opportunity to do a project on **“IntelliChat – Real-Time Messaging App with Integrated AI Analytics Assistant”**. Without their support and suggestions, this project would not have been completed.

Signature………..

Name: Nikhil Kumar

Roll No: 2310993889

Signature……….

Name: Nandan Sharma

Roll No:2310993884

Table of Contents

Abstract

In the modern digital era, messaging applications have become an integral part of our daily lives. Platforms like WhatsApp and Telegram allow for seamless communication but lack features that let users analyze their own chat data. As conversations grow, so does the need for tools that provide meaningful insights into messaging patterns, behavior, and sentiment.

IntelliChat is a full-stack real-time messaging application inspired by WhatsApp, designed with an integrated **AI analytics assistant** that offers users the ability to explore and understand their chat history. This assistant allows users to upload exported chat files and receive visual feedback through interactive dashboards and analysis reports—directly within the app.

Built using the **MERN stack** (MongoDB, Express.js, React.js, Node.js), the application supports live chat via **Socket.IO** and uses **Python** scripts for Natural Language Processing (NLP). These scripts perform **sentiment analysis, user activity breakdown, frequency trends**, and **basic topic modeling**, returning results that are visualized using Chart.js on the frontend.

A floating assistant UI presents users with multiple analysis options in the form of cards. Clicking any card triggers a report popup with charts and summaries, enabling non-technical users to interact with their data meaningfully.

Currently, core functionalities such as login, messaging, chat file uploads, and sentiment analytics are implemented. Future development will include live analysis, group chat insights, and downloadable reports.

In summary, IntelliChat redefines communication by merging real-time messaging with built-in AI-driven insights—making chat data not just readable, but intelligent.

Project Report:

Frontend Web

Application

Project Title: IntelliChat – Real-Time Messaging App with Integrated AI Analytics Assistant

---

1 Introduction

1.1 Objective

In our modern technology driven world, the digital communication platforms form the heart of the personal, educational, and professional interaction. WhatsApp, Telegram, and Signal messaging platforms have transformed the way people communicate by providing real-time communication regardless of the spread of geographical boundaries. Yet, these platforms are intended more as real-time text-sharing systems, and do not incorporate built-in options that assist users in studying their communication habits, feelings, and modes of engagement. The need to have intelligent systems beyond messaging is increasing as conversations become larger and more complex and require something more meaningful that reflects back on oneself.

A solution to this need has been created in the form of IntelliChat, the Real-Time Messaging App with Built-In AI Analytics Assistant. The main aim of the project is to create and deploy a platform capable of providing an experience that is not just based on instant communication but also empowers the users with an analytics-based feedback on their chat data. Users can get in-depth analysis by posting exported chat files, which can point out sentiment patterns, the frequency of user activity, commonly discussed topics and even can give predictive forecasts on future communication behavior.

Technically speaking, IntelliChat will be developed on the MERN stack (MongoDB, Express.js, React.js, Node.js), which is a scalable, modular, and high-performance architecture. Socket.IO supports real time messaging and ensures instant message delivery and live presence updates. In analytics, Python based scripts utilizing Pandas, Natural Language toolkit, TextBlob and Prophet are installed into the backend to do sentiment analysis, natural language processing and time-series prediction. The findings are translated into interactive charts with Chart.js, and allow the analysis of information by even non-technical users. Tailwind CSS provides a clean, modern and mobile responsive interface.

The IntelliChat will have a distinct characteristic, a floating AI Assistant that will serve as a portal to the analytics system. It gives insights as cards and modal popups hence, accessible and visually appealing data. The system will incorporate the use of JWT based authentication and secure data management measures to ensure security and user confidence.

To sum up, the task of IntelliChat is to turn the common chat applications into smart ecosystems that integrate messaging with analytics. The combination of communication and AI-generated contributions is designed to benefit the areas of education (analysis of participation), work teamwork (team productivity insights), and mental health (monitoring emotional well-being), which is the next phase of development of digital communication platforms.

1.2Technology Stack

I. **React.js:** Used to build a dynamic, component-based frontend with support for interactive chat UI, analysis dashboard, and assistant modals.

II. **Tailwind CSS:** A utility-first CSS framework applied for creating responsive, mobile-friendly layouts with custom styling and consistent design patterns.

III. **Node.js with Express.js:** Handles backend server logic, API routing, user authentication, chat processing, and integration with Python scripts via child processes.

IV. **MongoDB (NoSQL):** Serves as the primary database to store user details, messages, and uploaded chat data securely in a flexible, document-oriented format.

V. **Socket.IO:** Enables real-time, bidirectional communication between users, allowing instant message delivery and presence updates within the chat interface.

VI. **Python (Pandas, TextBlob, NLTK, Prophet):** Python scripts are used to perform natural language processing and predictive analytics on uploaded chat files.

VII. **Chart.js (via react-chartjs-2):** Visualizes sentiment, activity trends, and frequency analysis in the form of interactive graphs and charts within the frontend.

VIII. **JSON & REST APIs:** Used for data exchange between frontend, backend, and Python scripts to ensure modularity and clean architecture.

IX. **CDNs & External Libraries:** External fonts, icon packs, and libraries are loaded via CDNs to reduce loading times and enhance UI consistency.

2. Project Description

IntelliChat is a sophisticated, full-stack web application designed as a next-generation communication platform that merges the real-time messaging capabilities of popular apps like WhatsApp with a powerful, integrated AI-driven analytics assistant. The project's primary objective is to transcend traditional messaging by empowering users with meaningful, data-driven insights into their own communication patterns, sentiments, and behaviors. Built on the robust and scalable MERN stack (MongoDB, Express.js, React.js, Node.js), the application provides a modern, high-performance architecture capable of handling both instant communication and complex data processing. The frontend, or the client-side of the application, is crafted using React.js to create a dynamic, component-based user interface that includes the interactive chat screen, the analysis dashboard, and various assistant modals. This user interface is styled with Tailwind CSS, a utility-first framework that ensures a clean, modern, and fully responsive design that adapts seamlessly across different devices. Further visual refinement is achieved through a custom CSS styling layer, which applies specific design tweaks to elements like modals, buttons, and chart containers to enhance the overall user experience and ensure visual clarity.

The core of its messaging functionality is the Real-Time Chat Engine, which leverages Socket.IO to establish a persistent, bidirectional WebSocket connection between users. This engine facilitates instant message delivery and synchronization, creating a fluid and seamless live chat experience. To ensure user privacy and data security, the application incorporates a comprehensive Authentication Module that manages secure login and registration processes using industry-standard practices like JSON Web Tokens (JWT) for session management and hashed passwords for storing user credentials securely. The standout feature of IntelliChat is its AI analytics capability, which begins with the Chat Upload Handler. This component allows users to upload exported chat history files (in .txt format) directly into the application for analysis. Once a file is uploaded, the Node.js backend initiates the Python Script Integration, executing specialized Python scripts using the child_process module. These scripts utilize powerful Natural Language Processing (NLP) libraries such as NLTK and TextBlob to perform in-depth analysis, including sentiment analysis to gauge the emotional tone of conversations, user activity breakdowns to identify participation frequency, and trend analysis to visualize message volume over time. The processed insights are then sent back to the client and presented through the Analytics Assistant UI, which features a user-friendly interface with clickable cards for different analysis options. The final step is Report Visualization, where the analytical data is rendered into interactive and easy-to-understand charts—including pie charts, bar graphs, and line graphs—using Chart.js within the React components. To maintain optimal performance and visual consistency, the application also loads external resources like fonts and other dependencies through Content Delivery Networks (CDNs).

3. Flowchart:

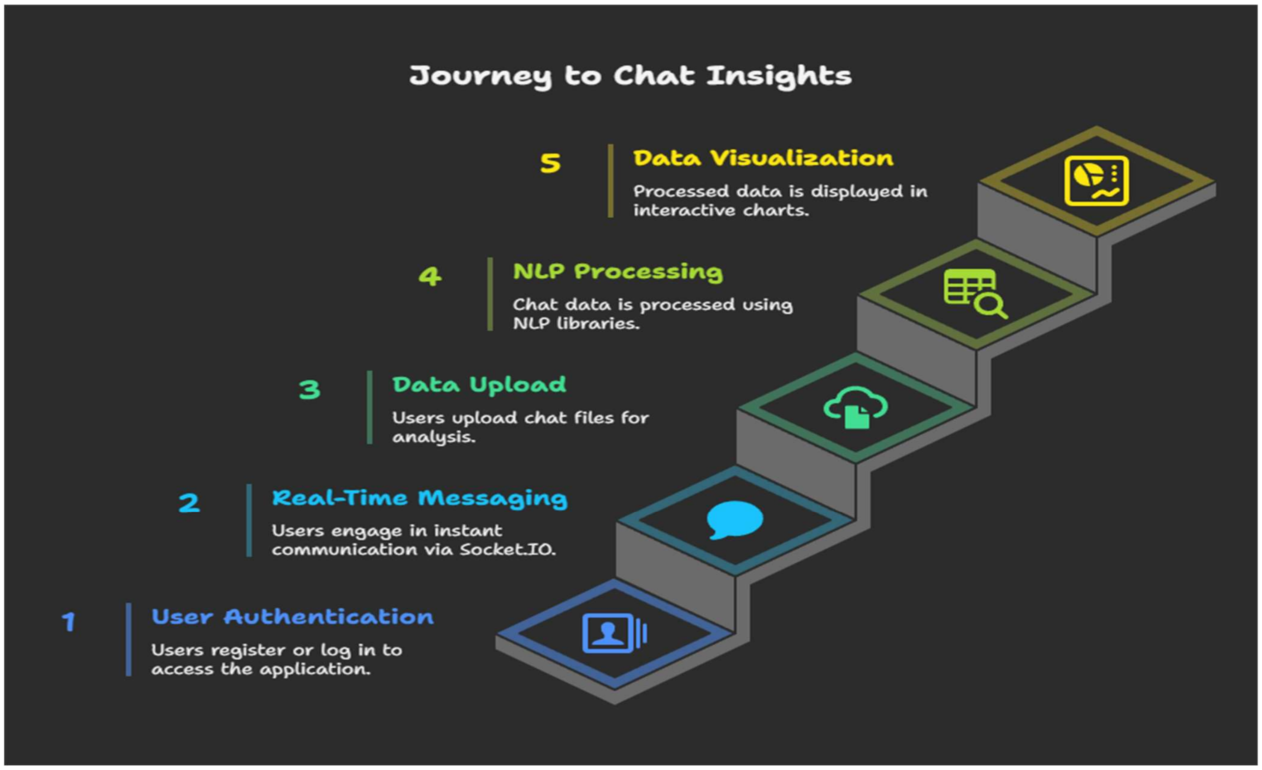

Journey to Chat Insights
5
Data Visualization
Processed data is displayed in interactive charts.
4
NLP Processing
Chat data is processed using NLP libraries.
3
Data Upload
Users upload chat files for analysis.
2
Real-Time Messaging
Users engage in instant communication via Socket.IO.
1
User Authentication
Users register or log in to access the application.

Fig 1

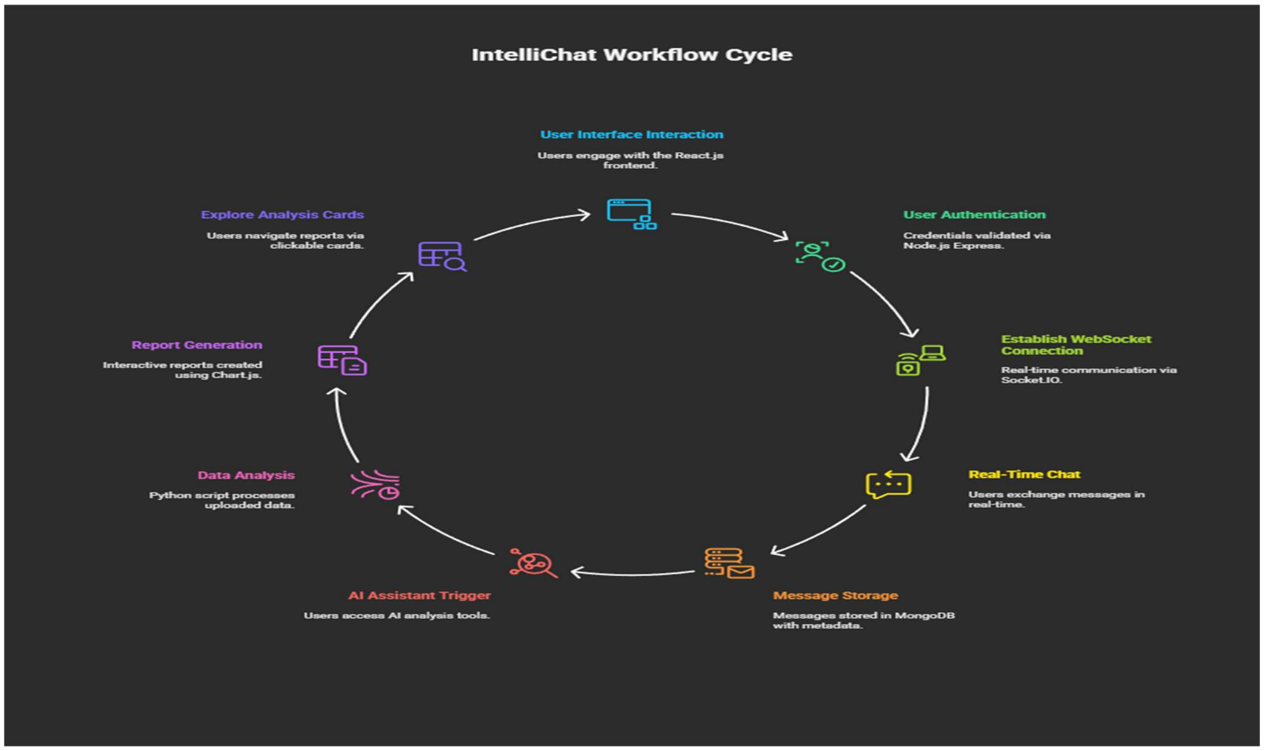

IntelliChat Workflow Cycle
User Interface Interaction
Users engage with the React.js frontend.
User Authentication
Credentials validated via Node.js Express.
Establish WebSocket Connection
Real-time communication via Socket.IO.
Real-Time Chat
Users exchange messages in real-time.
Message Storage
Messages stored in MongoDB with metadata.
AI Assistant Trigger
Users access AI analysis tools.
Data Analysis
Python script processes uploaded data.
Report Generation
Interactive reports created using Chart.js.
Explore Analysis Cards
Users navigate reports via clickable cards.

Fig 2

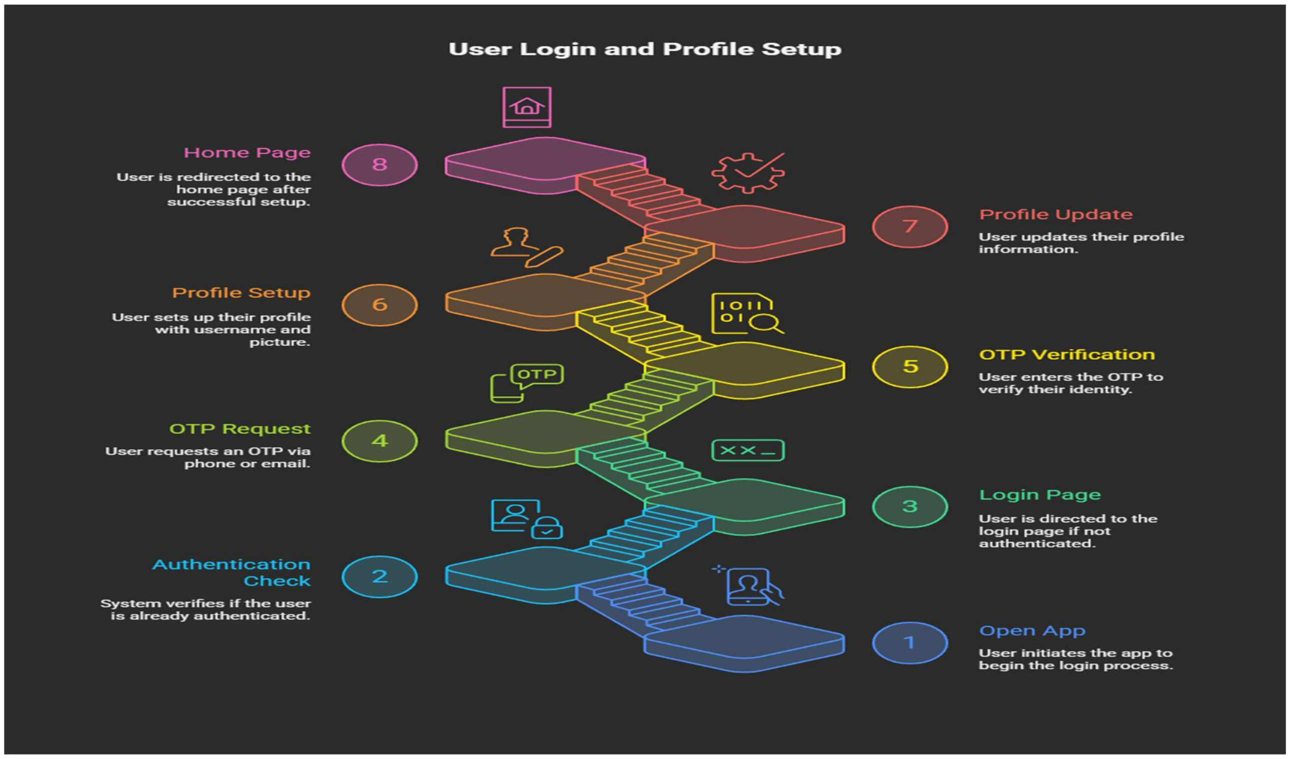

User Login and Profile Setup
8
Home Page
User is redirected to the home page after successful setup.
7
Profile Update
User updates their profile information.
6
Profile Setup
User sets up their profile with username and picture.
5
OTP Verification
User enters the OTP to verify their identity.
OTP
4
OTP Request
User requests an OTP via phone or email.
3
Login Page
User is directed to the login page if not authenticated.
2
Authentication Check
System verifies if the user is already authenticated.
1
Open App
User initiates the app to begin the login process.

Fig 3

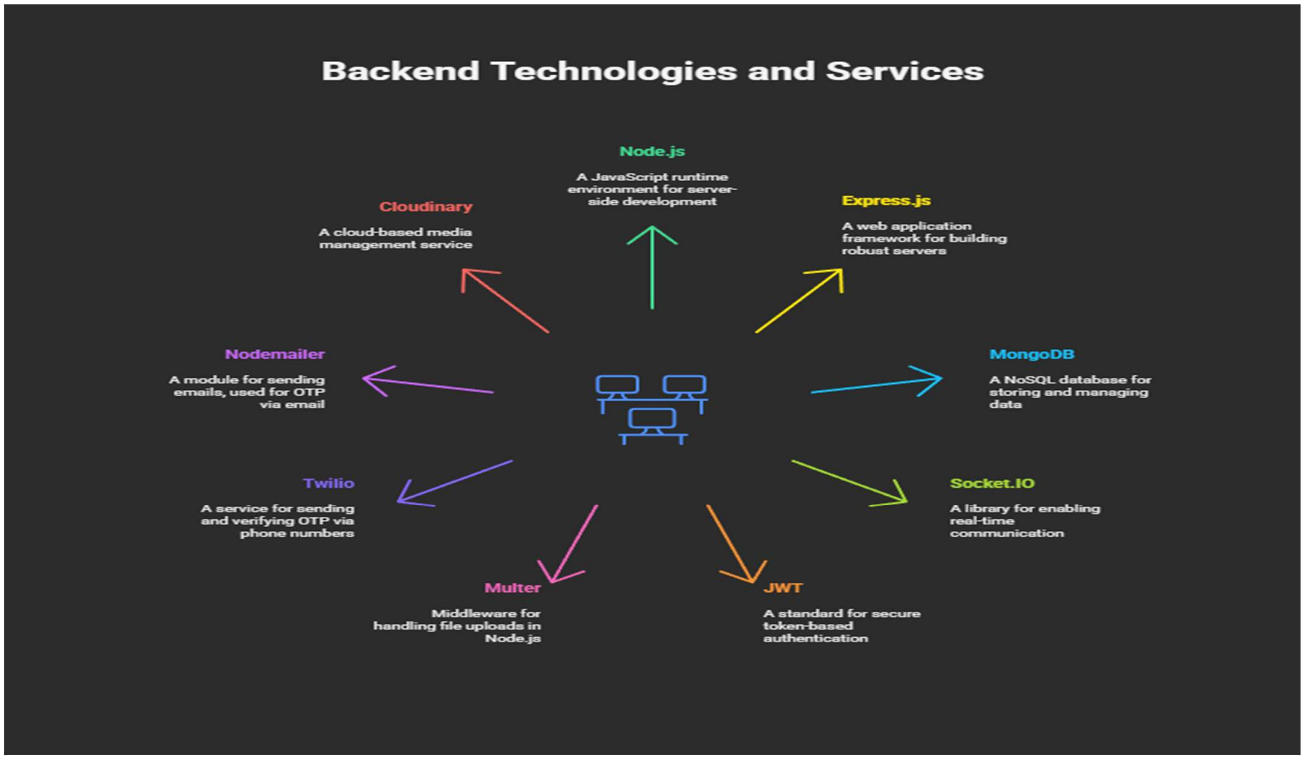

Backend Technologies and Services
Node.js
A JavaScript runtime environment for server-side development
Express.js
A web application framework for building robust servers
MongoDB
A NoSQL database for storing and managing data
Socket.IO
A library for enabling real-time communication
JWT
A standard for secure token-based authentication
Multer
Middleware for handling file uploads in Node.js
Twilio
A service for sending and verifying OTP via phone numbers
Nodemailer
A module for sending emails, used for OTP via email
Cloudinary
A cloud-based media management service

Fig 4

4. System Architecture & Design

4.1 High-Level System Architecture

The IntelliChat application follows a full-stack architecture combining a React-based frontend with a Node.js and Express.js backend. Real-time messaging is enabled via Socket.IO, while MongoDB handles user and message data. For AI-driven analytics, the backend triggers Python scripts that process uploaded chat files and return insights to the client. The architecture can be represented as:

User (Browser)

↓

React.js + Tailwind (Frontend UI)

↓

Node.js + Express.js (Backend Server)

↓

MongoDB (Database)

↓

Python Scripts (NLP + Analytics Engine)

- **Client–Server Communication:** User actions like login, messaging, or chat file upload are processed through RESTful APIs and WebSocket channels. The backend ensures secure routing, authentication, and real-time data flow.

- **Real-Time Messaging:** Implemented using **Socket.IO**, allowing bidirectional, event-based communication between connected clients.

- **NLP Script Integration:** The Node.js server uses child_process to execute Python scripts for analytics tasks like sentiment analysis, user activity tracking, and predictive modeling.

- **Chart Rendering:** The frontend receives JSON results from Python scripts and renders them into charts and graphs using Chart.js inside the React interface.

- **CDN Integration:** External libraries such as fonts and chart utilities are loaded via CDNs for optimized performance and visual consistency.

4.2 Detailed System Design

- **UI Layer:** Developed using **React.js** with reusable components, routing, and responsive layouts styled using **Tailwind CSS** utility classes. This ensures a mobile-friendly, intuitive user interface that supports both messaging and analytics.
- **Styling Layer:** Custom enhancements are applied via scoped CSS and Tailwind extensions to style chat bubbles, input fields, buttons, assistant modals, and chart containers. Visual consistency is maintained throughout the application.
- **Logic Layer:** All dynamic and interactive behavior is handled through JavaScript logic embedded in React components. Key logic features include:

  - **Socket-based real-time chat** functionality for instant message transmission.
  - **Authentication system** using JWT tokens and hashed passwords to secure user sessions.
  - **Chat file upload and validation** to ensure compatibility with the analytics engine.
  - **Python script execution** via Node.js child_process for analytics processing.
  - **Frontend rendering of reports** using Chart.js, displaying data like sentiment trends and user activity.
  - **Floating assistant interaction logic**, allowing users to trigger different types of analysis via clickable cards.
  - **Modal-based popups** that dynamically fetch and render analysis results in visually engaging formats.

5 Features Implemented

Feature	Description
● **Real-Time Messaging**	Enables one-on-one instant chat between users using **Socket.IO** for real-time, event-based communication.
● **User Authentication**	Secure login and registration system with hashed passwords and JWT-based session management.
● **Chat File Upload**	Allows users to upload exported WhatsApp chat files in .txt format for further analysis.
● **Floating Assistant UI**	A persistent, clickable icon that opens a dashboard with multiple analysis options in the form of cards.
● **Sentiment Analysis**	Analyzes chat tone using NLP and displays positive, negative, or neutral sentiment across the conversation.
● **User Activity Report**	Displays per-user message counts, participation frequency, and most active periods using visual charts.
● **Frequency & Trends Chart**	Shows message volume trends over time and predicts future activity using time-series forecasting models.
● **Interactive Report Modals**	Opens popup windows with charts (bar, pie, line graphs) rendered via Chart.js based on analysis results.

6 Challenges Faced

- **Real-Time Messaging Integration**: A primary challenge was integrating a seamless and reliable real-time messaging system. This was successfully addressed by implementing
- **Socket.IO**, which enabled bi-directional, event-based communication between users. The solution involved creating user-specific rooms and event listeners to handle instant message delivery and presence updates, ensuring a fluid chat experience.
- **Integrating Python NLP with a Node.js Backend**: Bridging the gap between the JavaScript-based Node.js backend and the Python environment used for NLP was a significant technical hurdle. The solution involved using the child_process module in Node.js to execute Python scripts for analytics tasks such as sentiment analysis and predictive modeling. Data was exchanged between these environments using JSON and REST APIs to maintain a clean and modular architecture.
- **Parsing and Standardizing Uploaded Chat Files**: Exported `.txt` chat files often have inconsistent formatting for timestamps, sender names, and system messages, which can break the data analysis pipeline. To solve this, a robust chat upload handler was developed to manage file parsing and server-side validation. The Python scripts then use libraries like
- **Pandas** to process, clean, and standardize the uploaded data, creating a uniform structure for reliable analysis.
- **Responsive Layout Tuning**: Ensuring the application's interface remained consistent and functional across various devices and screen sizes was a key challenge. This was overcome by leveraging
- **Tailwind CSS**, a utility-first framework, along with its media queries to build a consistent and mobile-friendly design that provides an optimal user experience on all devices.
- **Managing Complex Frontend State**: In a real-time application, efficiently managing the state of incoming messages, user presence, and dynamic analytics data in React can be complex. The project addressed this by implementing the lightweight state management library
- **Zustand**. This provided a centralized store to handle global UI state, ensuring that updates from the server triggered minimal and targeted re-renders of only the necessary components, thus maintaining high performance.
- **Ensuring Data Security and User Privacy**: Protecting sensitive user conversations and personal data was a critical challenge. The application implemented a multi-layered security approach, including a secure login and registration system with hashed passwords and
- **JWT-based session management** to protect API endpoints and user sessions. This ensures that all user data is handled securely and that only authenticated users can access their information.

7 Result

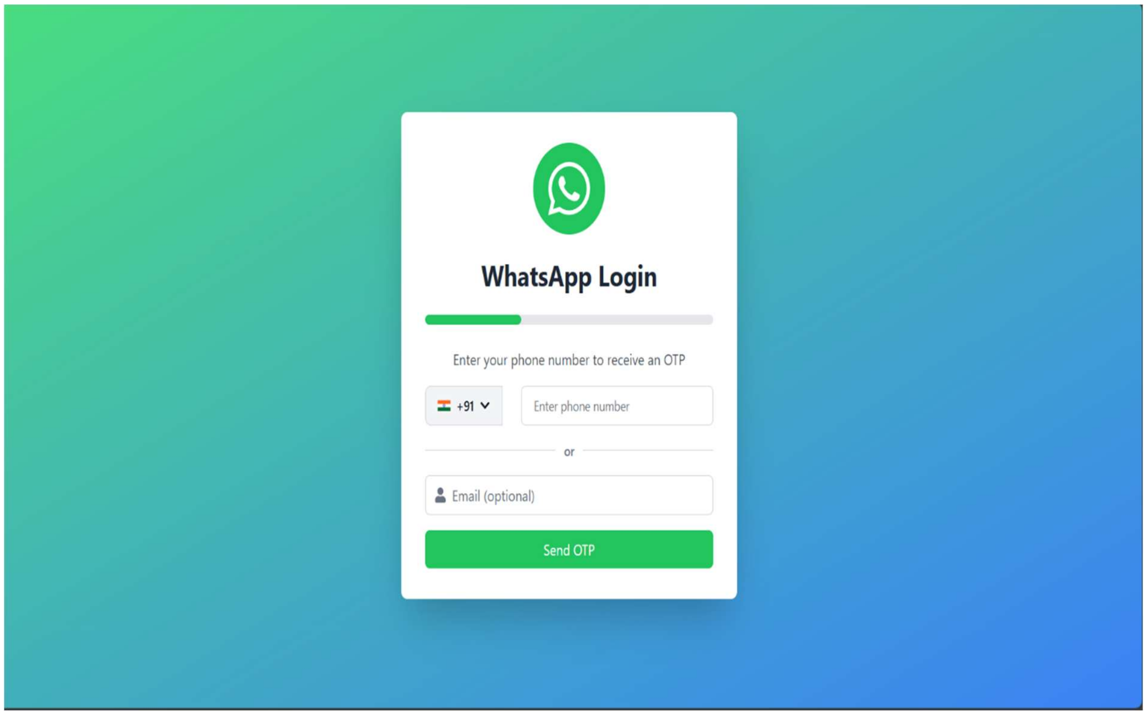


Fig. 5

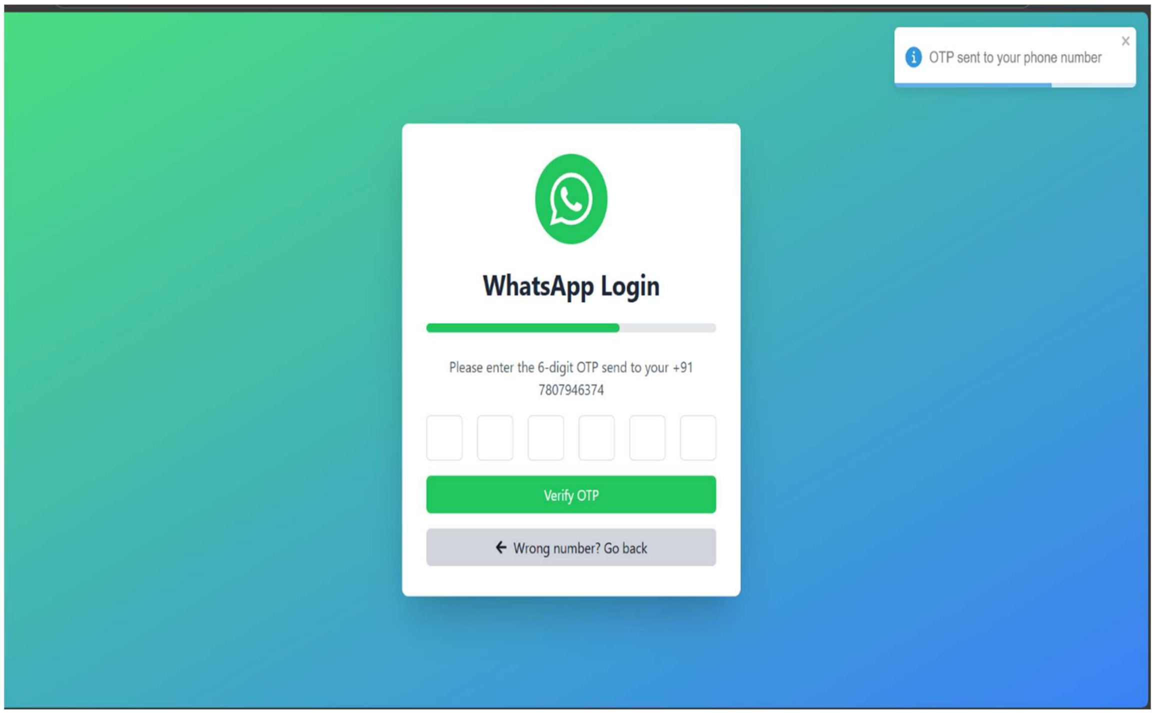
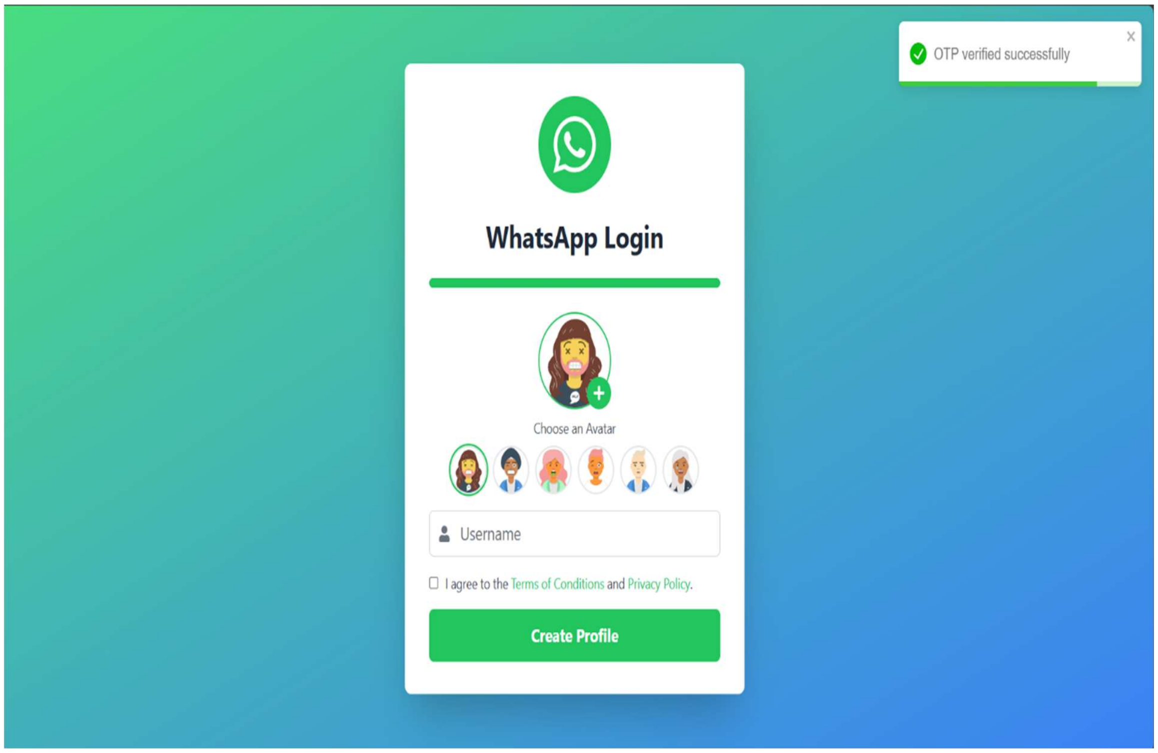

OTP sent to your phone number
WhatsApp Login
Please enter the 6-digit OTP send to your +91 7807946374
Verify OTP
Wrong number? Go back

Fig. 6



OTP verified successfully

WhatsApp Login

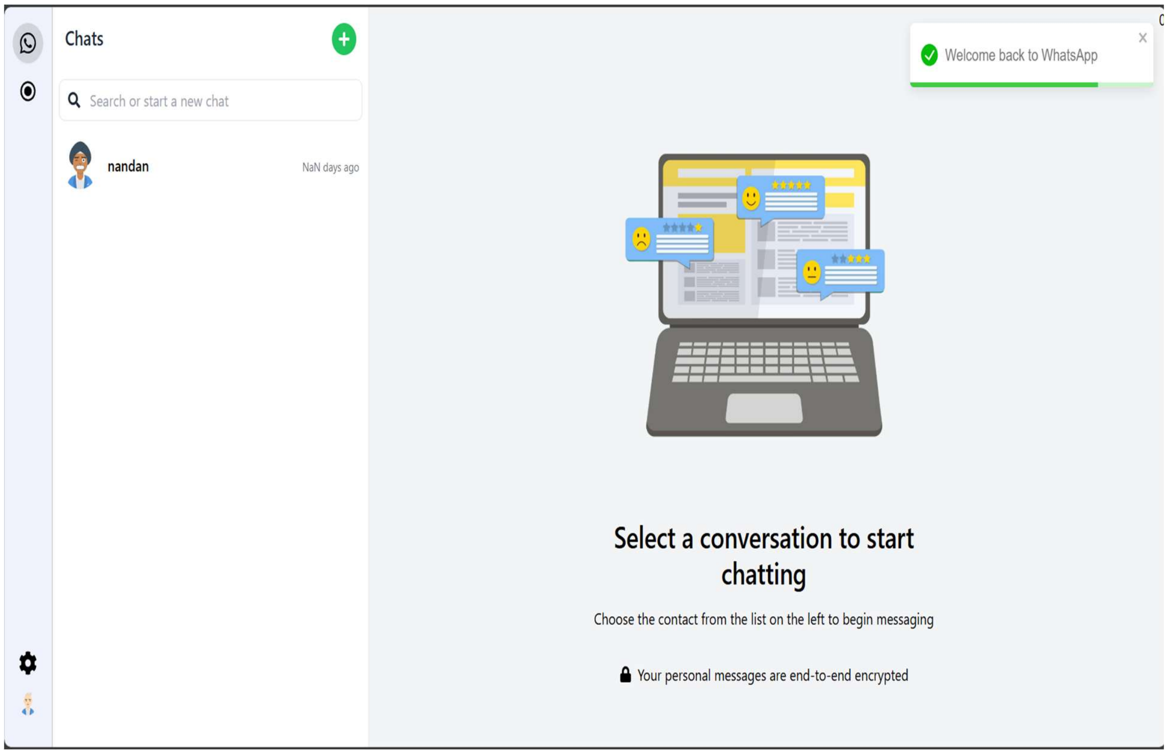
Choose an Avatar

Username

I agree to the Terms of Conditions and Privacy Policy.

Create Profile

Fig.7



Chats

Search or start a new chat

nandan NaN days ago

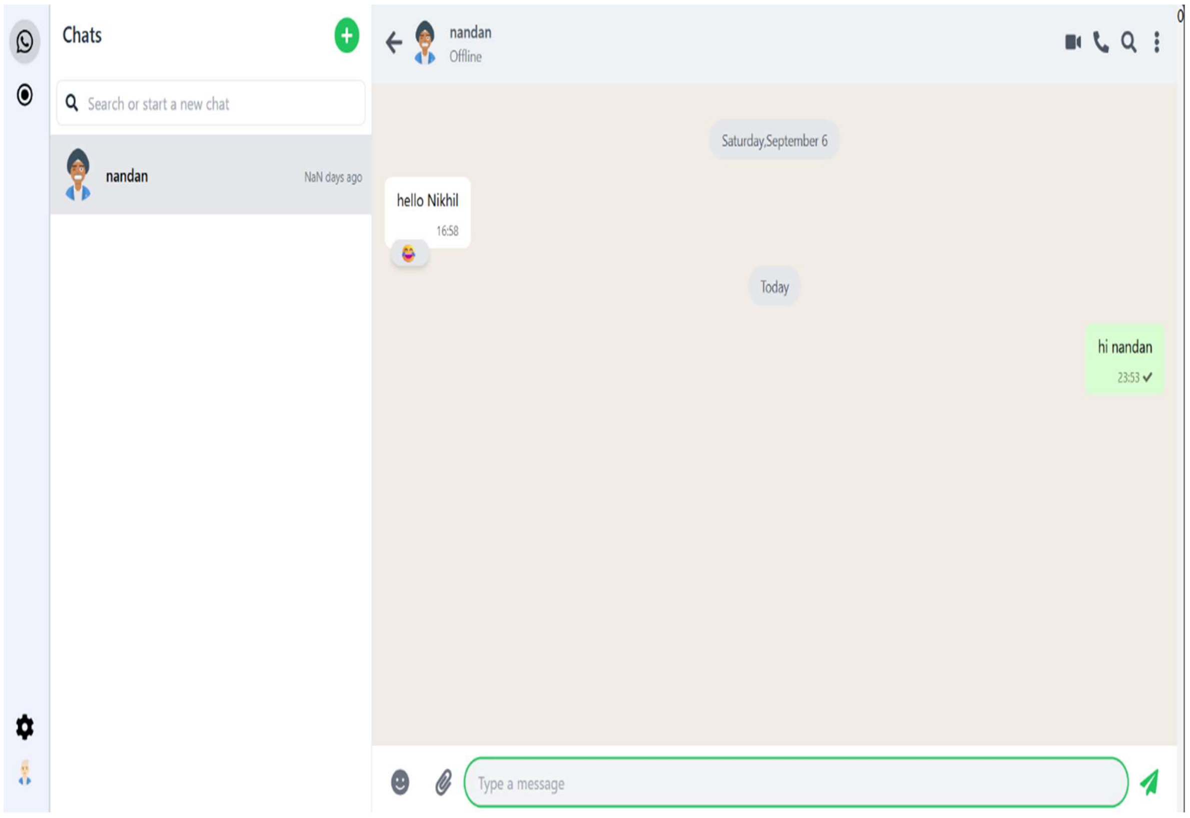
Welcome back to WhatsApp

Select a conversation to start chatting

Choose the contact from the list on the left to begin messaging

Your personal messages are end-to-end encrypted

Fig. 8



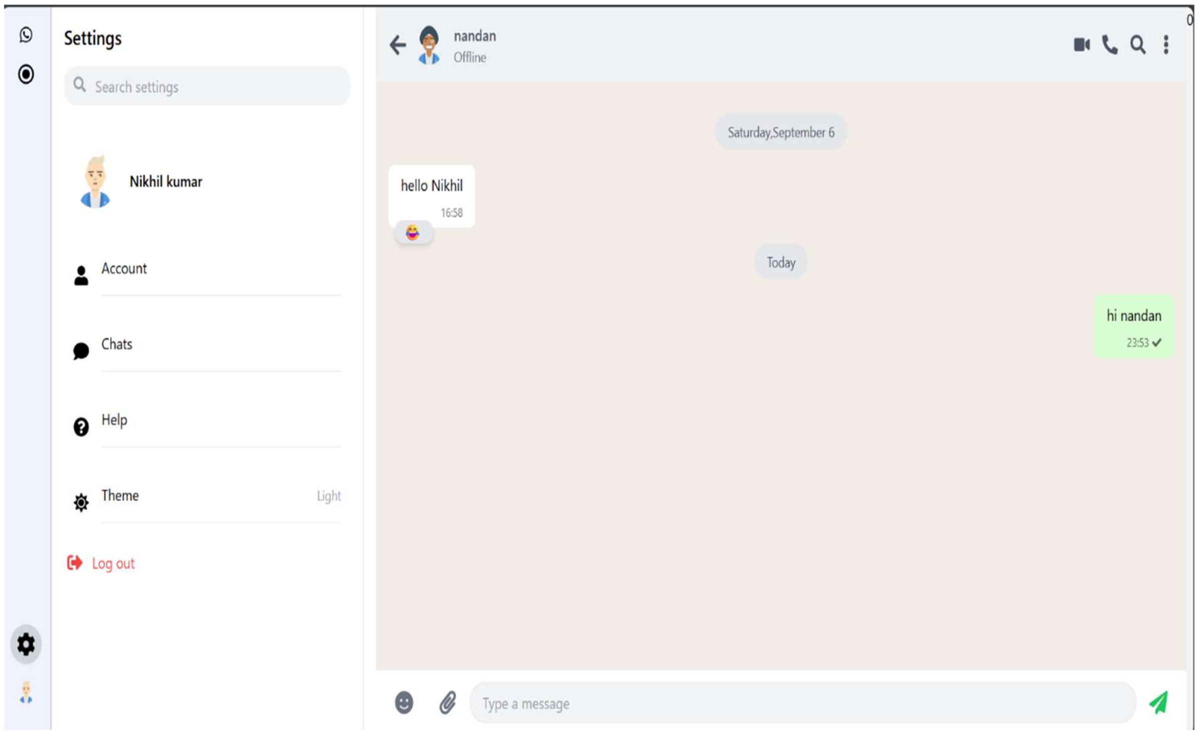
Saturday,September 6

hello Nikhil 16:58

Today

hi nandan 23:53 ✓

Fig. 9



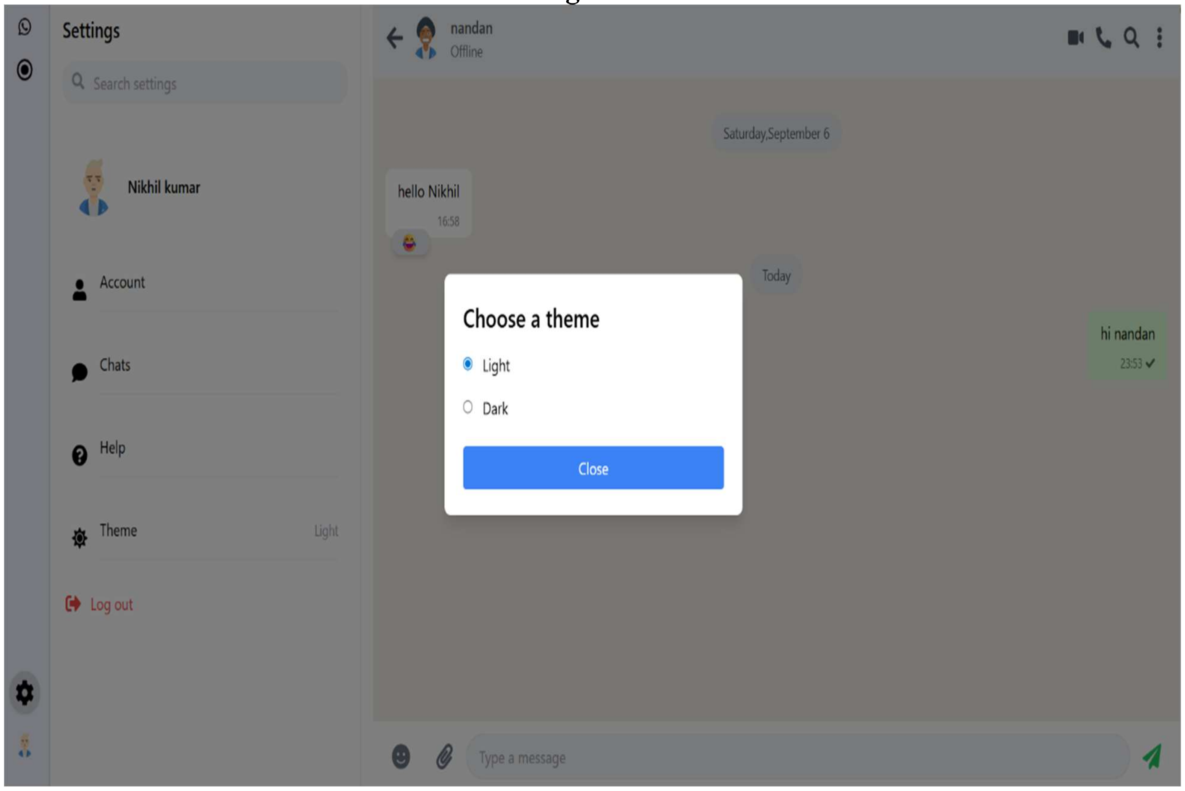
Saturday,September 6

hello Nikhil 16:58

Today

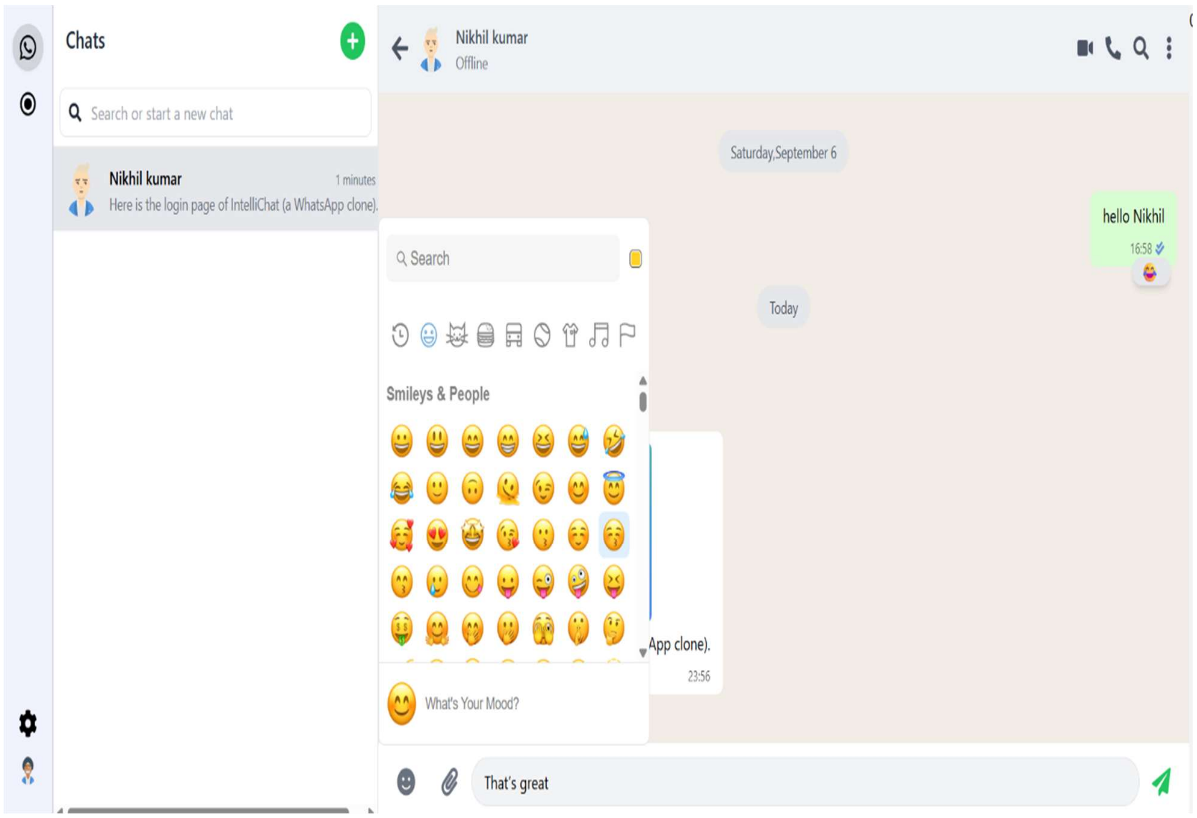
hi nandan 23:53 ✓

Fig. 10



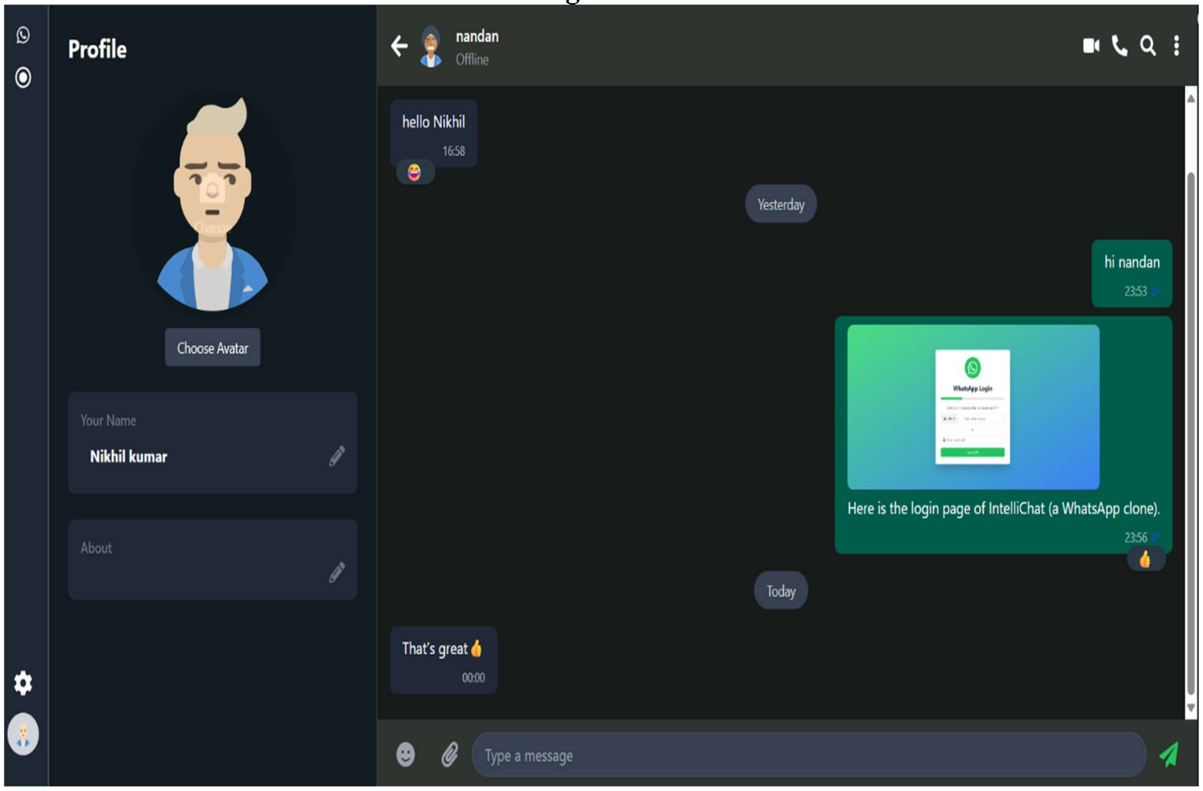
Settings
Search settings
Nikhil kumar
Account
Chats
Help
Theme
Light
Log out
nandan
Offline
Saturday,September 6
hello Nikhil
16:58
Today
hi nandan
23:53
Choose a theme
Light
Dark
Close
Type a message

Fig. 11



Chats
Search or start a new chat
Nikhil kumar
1 minutes
Here is the login page of IntelliChat (a WhatsApp clone).
Nikhil kumar
Offline
Saturday,September 6
hello Nikhil
16:58
Today
Search
Smileys & People
What's Your Mood?
App clone).
23:56
That's great

Fig. 12


Profile

Your Name: Nikhil kumar

About

nandan
Offline

hello Nikhil
16:58

Yesterday

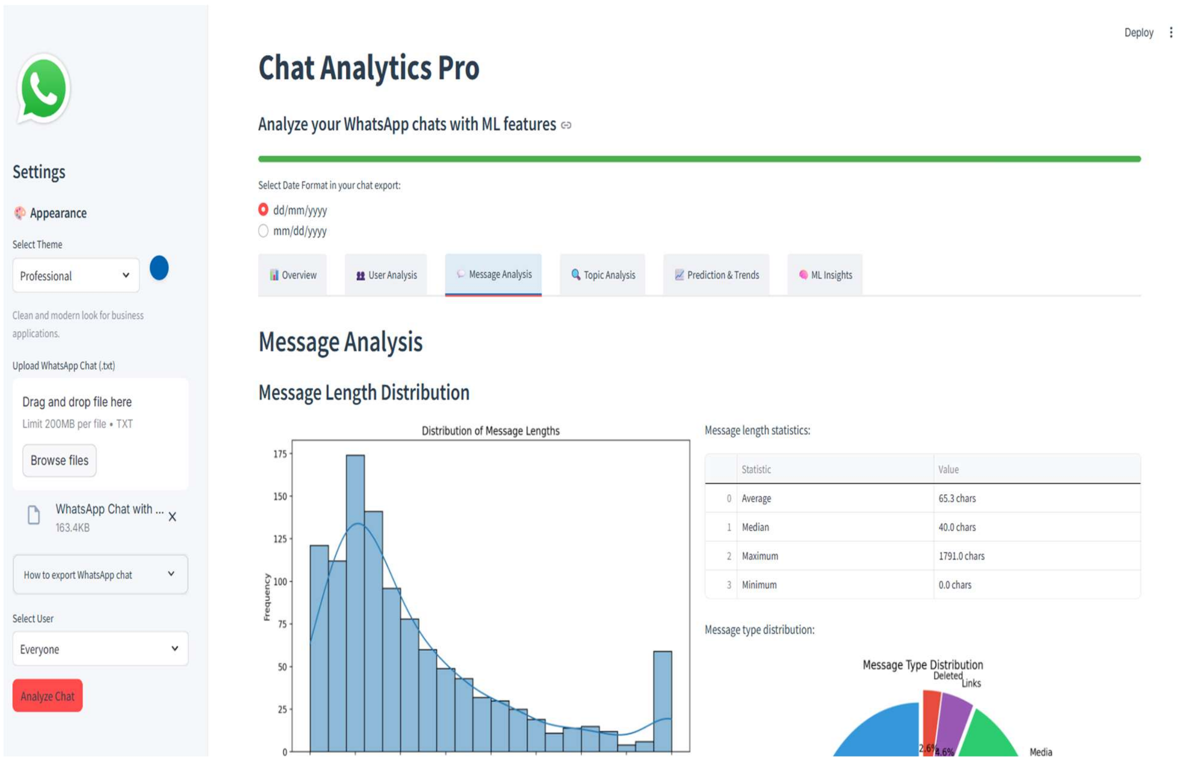
hi nandan
23:53

Here is the login page of IntelliChat (a WhatsApp clone).
23:56

Today

That's great 👍
00:00

Fig. 13


Chat Analytics Pro

Analyze your WhatsApp chats with ML features

Select Date Format in your chat export:
[✓ dd/mm/yyyy]
mm/dd/yyyy

Message Analysis

Message Length Distribution

Message length statistics:

	Statistic	Value
0	Average	65.3 chars
1	Median	40.0 chars
2	Maximum	1791.0 chars
3	Minimum	0.0 chars

Message type distribution:

Settings
Appearance
Select Theme: Professional
Clean and modern look for business applications.
Upload WhatsApp Chat (.txt)
WhatsApp Chat with ... 163.4KB
Select User: Everyone

Fig. 14

8. Conclusion:

The development of IntelliChat successfully demonstrates the integration of modern full-stack technologies with AI-powered analytics to create a next-generation communication platform. Unlike conventional messaging applications, IntelliChat not only facilitates secure, real-time conversations but also empowers users with meaningful insights derived from their own chat data. By leveraging the MERN stack, Socket.IO, and Python-based NLP, the system delivers innovative features such as sentiment analysis, user activity reports, and predictive trends through an intuitive and accessible interface. The project highlights the immense potential of combining live messaging with data intelligence to enhance user engagement, self-awareness, and decision-making. In essence, IntelliChat is a significant step forward in redefining digital communication by merging usability with intelligence—transforming ordinary conversations into actionable knowledge.

Beyond its technical achievements, IntelliChat champions a new paradigm of user empowerment and data literacy. By placing powerful analytical tools directly into the hands of the user, the application fosters a deeper understanding of interpersonal dynamics and emotional well-being as reflected in their digital conversations. This capability extends its utility beyond personal use, opening up potential applications in educational contexts for analyzing student participation, professional settings for gaining insights into team productivity, or even personal well-being for monitoring emotional health trends. Ultimately, IntelliChat serves as a forward-thinking model for how personal data can be used constructively, shifting the focus from passive communication to active, insightful self-reflection.

9. Future Scope

While IntelliChat already integrates real-time communication with AI-driven analytics, there is significant scope for expansion to make it comparable with leading global messaging platforms. Some proposed future enhancements include:

1. **Status/Story Feature** – Adding a dedicated status page similar to WhatsApp or Instagram Stories, allowing users to share text, images, or short videos that disappear after 24 hours.
2. **Voice and Video Calling System** – Integrating WebRTC-based peer-to-peer calling features to enable one-to-one voice and video communication directly within IntelliChat.
3. **Integrated AI Chatbot** – Embedding a conversational AI assistant that can answer queries, suggest responses, or provide automated insights in real-time alongside messaging. This would extend IntelliChat's utility beyond analytics into intelligent assistance.
4. **Group Chat Analytics** – Expanding analysis to group-level conversations, identifying dominant speakers, activity trends, and sentiment variation within communities.
5. **Cross-Platform Deployment** – Deploying IntelliChat on scalable cloud platforms (such as **Render, Vercel, or AWS**) to ensure high availability and real-world usability. Mobile app deployment (Android/iOS via React Native) can further enhance accessibility.
6. **Enhanced Security and Encryption** – Implementing end-to-end encryption for all messages and analytics data, ensuring maximum user privacy and compliance with modern data protection standards.

10. Appendix:

- **GitHub Repo : https://github.com/nikhil7591/IntelliChat**
- **React.js Documentation** – https://react.dev
- **Express.js Documentation** – https://expressjs.com
- **MongoDB Documentation** – https://www.mongodb.com/docs
- **Socket.IO Documentation** – https://socket.io/docs
- **Tailwind CSS** – https://tailwindcss.com/docs
- **Python NLP (NLTK/TextBlob)** – https://www.nltk.org / https://textblob.readthedocs.io
- **Zustand** – https://zustand-demo.pmnd.rs
- **React Icons** – https://react-icons.github.io/react-icons
- **Cloudinary** – https://cloudinary.com/documentation