

Project Title: ChatNest Web Application

Project team member: Vinaya kumar (22UG010574, 22CSE444)

Rounak Agasthi (22UG011252, 22CSE1049)

Somya Sagar Swain (22UG011056, 22CSE933)

Problem Statement:

In today's world, communication plays a key role in connecting people. While there are many chat applications available, most are either paid, lack customization, or don't provide advanced features like real-time updates, voice typing, or secure media sharing. The main challenge is to build a platform where users can chat in real-time, share files, send voice messages, and enjoy a smooth and secure chatting experience — all in one place.

Solution to the problem?

To build a secure and interactive real-time chat web application that supports instant messaging, voice typing, file sharing, and user activity tracking using MERN stack.

OBJECTIVE:

The main objectives of the ChatNest project are:

- **To improve front-end responsiveness and usability** – Design a responsive, user-friendly interface that adapts well across devices and screen sizes.
- **To implement RESTful APIs** – Design and integrate REST APIs for user authentication, message handling, and media storage to ensure smooth client-server communication. To provide file sharing features for images, videos, and documents.
- **To enable real-time communication** – Use **Socket.io** to establish instant, bidirectional messaging between users without the need for page refresh. To add theme customization with dark and light mode for better user experience.
- **To ensure data security and privacy** – Implement data encryption for user credentials and chat messages, protecting sensitive information stored in the database.
- **To provide user authentication and session management** – Create a secure login and registration system with token-based authentication (e.g., JWT).
- **To offer UI customization** – Provide a theme customization panel with multiple color themes and dark/light mode for a personalized user experience.
- **To include user activity tracking** – Show the online/offline status of users in real time for better interactivity.

LITERATURE SURVEY SUMMARY:

Evolution of Communication Systems

- Earlier communication systems were based on emails and static message boards, which lacked instant interaction.
- The need for real-time, instant communication led to the rise of chat applications such as WhatsApp, Messenger, and Slack.
- These platforms demonstrated the importance of low-latency communication, data synchronization, and secure message transfer.

Traditional Web Communication Methods

- Initially, communication between client and server was done through **HTTP requests**, which required refreshing the page for every new message.
- This approach caused delays and poor user experience in chat-based systems.
- The introduction of **AJAX (Asynchronous JavaScript and XML)** improved this process by enabling data exchange without full-page reloads, but it still wasn't truly real-time.

Introduction of Real-Time Web Technologies

- The emergence of **WebSocket** revolutionized web communication by allowing **two-way, persistent connections** between clients and servers.
- Libraries like **Socket.io** built on top of WebSocket simplified implementation and made **real-time chat applications** more accessible to developers.
- This advancement enabled the creation of **instant messaging systems, live notifications, and online presence tracking**.

MERN Stack as a Modern Web Development Framework

- The **MERN stack** (MongoDB, Express.js, React.js, Node.js) became popular for building scalable and efficient web applications.
- **MongoDB** provides a flexible document-based database structure suitable for storing chat messages, user data, and media.
- **Express.js** and **Node.js** manage backend logic, API routing, and server-side communication efficiently.
- **React.js** ensures a dynamic and responsive front-end user interface, enhancing usability and interactivity.

Data Security and Encryption in Chat Applications

- As online communication grows, **data privacy and security** have become critical concerns.
- Research emphasizes the use of **encryption algorithms** (like AES or bcrypt hashing) to protect user credentials and sensitive data.
- Modern chat systems integrate **end-to-end encryption** and secure authentication methods (e.g., JWT tokens) to ensure message confidentiality and data integrity.

Voice and Audio Features in Communication Tools

- Recent chat applications are incorporating **voice typing (speech-to-text)** and **audio messaging** to enhance accessibility and user experience.
- This innovation is supported by **Web Speech API**, **Media Recorder API**, and **audio streaming libraries** that enable seamless voice interaction.
- ChatNest adopts these modern techniques to provide **hands-free communication** and **multimedia-rich interaction**.

IMPLEMENTATION PLAN:

The implementation of ChatNest is divided into several structured phases to ensure smooth development, testing, and integration of all system components. Each phase focuses on specific objectives that contribute to the overall functionality and performance of the application.

1. Requirement Analysis

- Identify all **functional requirements**, including:
 - User registration and authentication
 - Real-time chat communication
 - File and media sharing (images, MP4 videos, and documents)
 - Voice typing (speech-to-text) and audio message recording
 - Online user tracking and message search functionality
 - Theme customization with dark/light modes
- Define **non-functional requirements** such as:
 - Security, performance, scalability, and usability

Technologies Used and Their Usage

1. Frontend: React.js

- **Purpose:**
React.js is used to build the **user interface (UI)** of ChatNest. It creates a fast, responsive, and interactive chat experience.
- **Usage in ChatNest:**
 - ✓ Develops different components such as Login Page, Chat Window, Profile, and Settings.
 - ✓ Handles real-time updates efficiently through state management.
 - ✓ Ensures smooth rendering of messages, media, and theme changes.
 - ✓ Makes the UI responsive and user-friendly for both desktop and mobile users.

2. Backend: Node.js with Express.js

- **Purpose:**
Node.js provides the **runtime environment**, while Express.js is used to build the **backend framework** and **RESTful APIs**.
- **Usage in ChatNest:**
 - ✓ Handles all server-side logic and routes for user authentication, message handling, and file uploads.
 - ✓ Manages communication between frontend and database.
 - ✓ Express middleware for error handling, request validation, and file management.
 - ✓ Provides endpoints like `/api/login`, `/api/register`, and `/api/messages` for data exchange.

3. Database: MongoDB

- **Purpose:**
MongoDB is a **NoSQL database** used to store user information, messages, and media in a flexible document structure.
- **Usage in ChatNest:**
 - ✓ Stores user credentials, message history, and uploaded media links.
 - ✓ Ensures fast data retrieval for real-time chat updates.
 - ✓ Uses collections such as **Users**, **Messages**, and **Media**.
 - ✓ Supports indexing and query optimization for faster search.

4. Real-time Communication: Socket.io

- **Purpose:**
Socket.io enables **real-time, two-way communication** between the server and connected clients.
- **Usage in ChatNest:**
 - ✓ Facilitates instant message delivery without refreshing the page.
 - ✓ Handles events like user-joined, message-sent.
 - ✓ Tracks online/offline users in real-time.
 - ✓ Provides smooth, live interaction between users.

5. Speech Recognition: Speech Recognition API

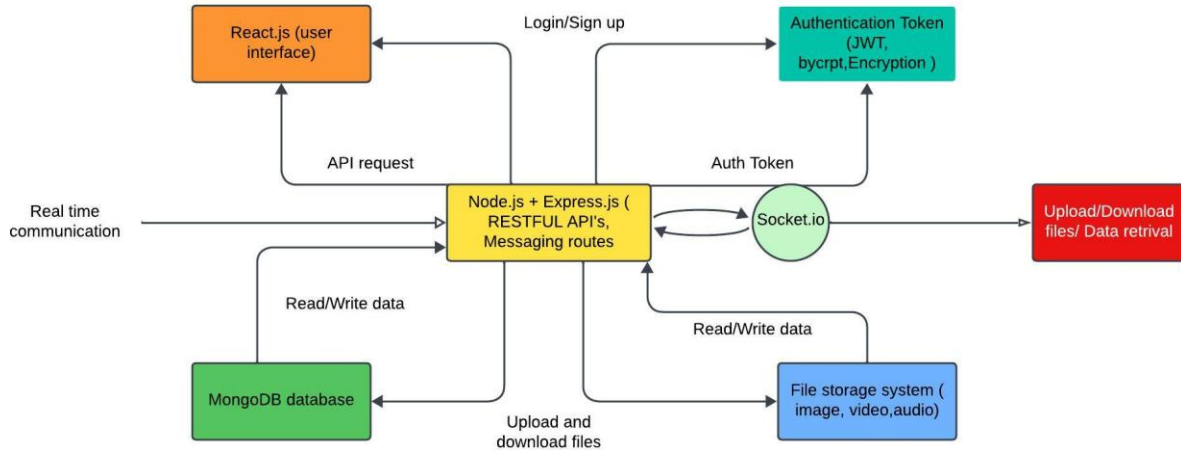
- **Purpose:**
The **Web Speech API** allows conversion of **spoken words into text** using voice input.
- **Usage in ChatNest:**
 - ✓ Enables users to send messages using **voice typing** instead of manual typing.
 - ✓ Increase accessibility for users with typing difficulties.
 - ✓ Recognizes speech in real-time and displays it in the chat input box before sending it.

6. Audio Recording: Media Recorder API

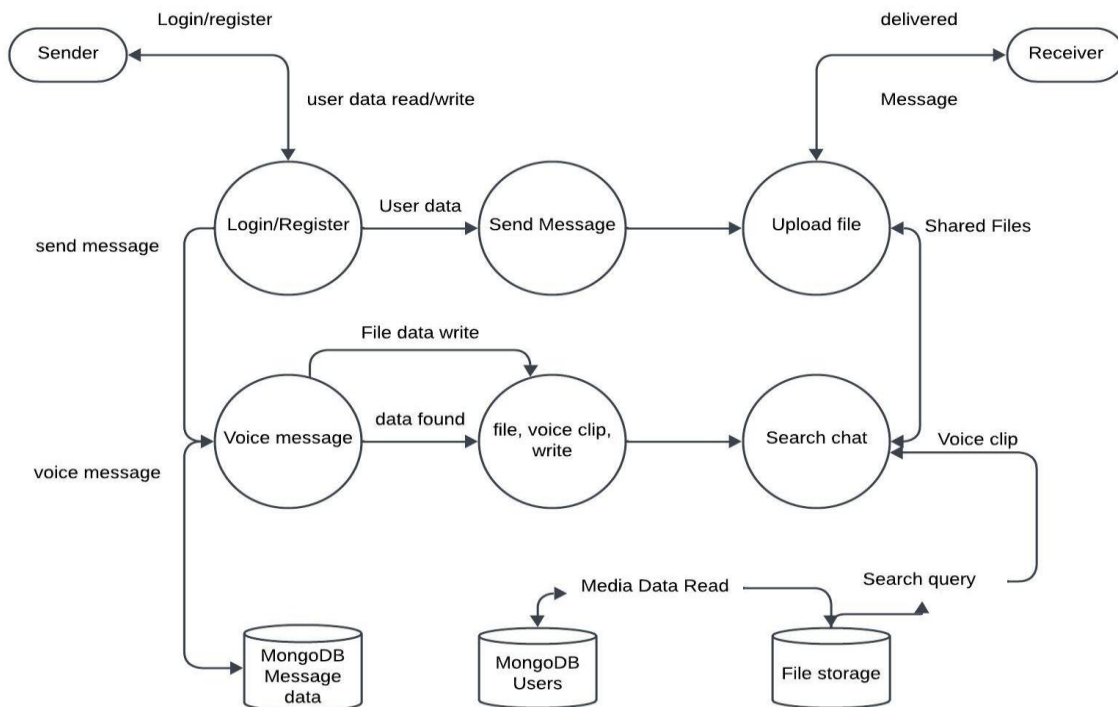
- **Purpose:**
The **Media Recorder API** is used to **capture and record audio** directly from the user's microphone.
- **Usage in ChatNest:**
 - ✓ Allows users to record and send **voice messages**.
 - ✓ Records audio in formats like .webm or .mp3 for playback within the chat.
 - ✓ Integrates with the chat system to upload and store recorded files.
 - ✓ Adds variety to communication options (text, image, audio).

MODEL DESIGN:

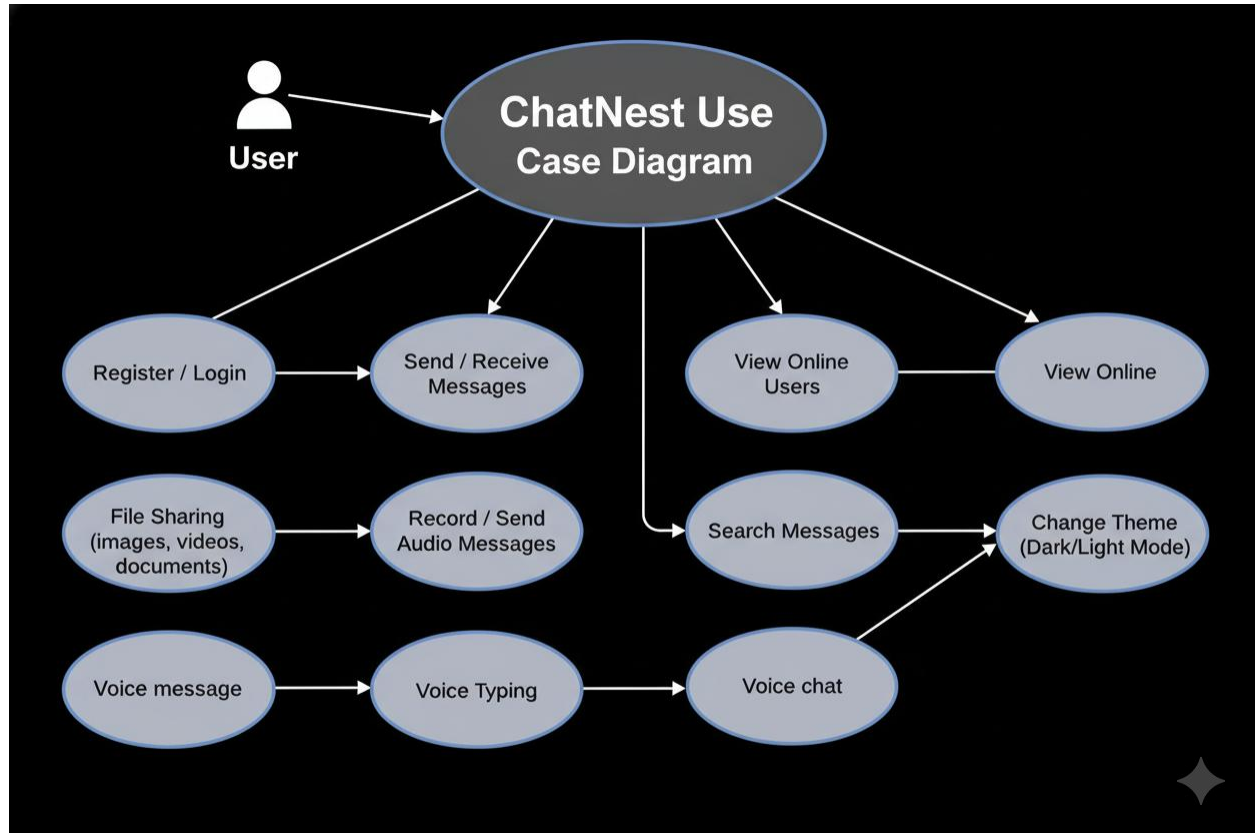
System Architecture Design



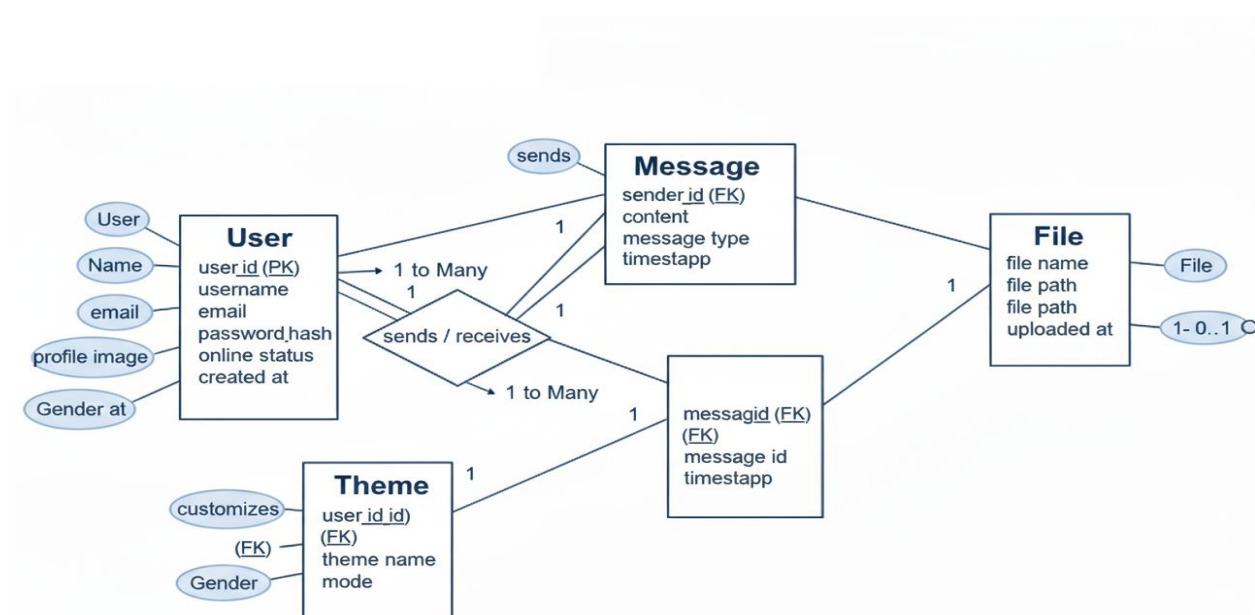
Data Flow diagram



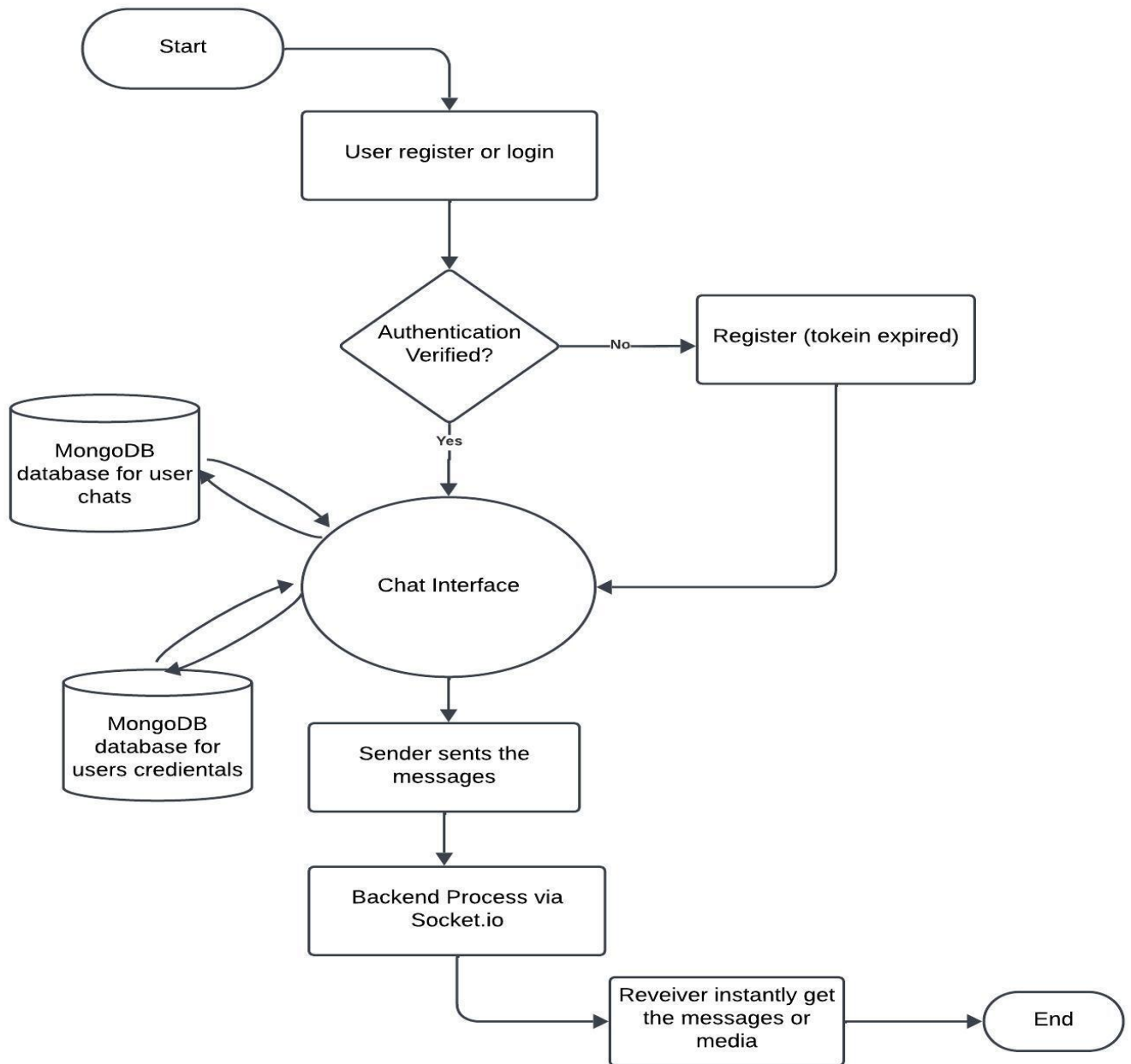
Use case diagram



Entity Relationship diagram



Flow chart diagram



Registration page

ChatNest

Settings

Create Account

Get start with your free account

Full Name

Enter your Full name

Email

Enter you gmail

Password

Password

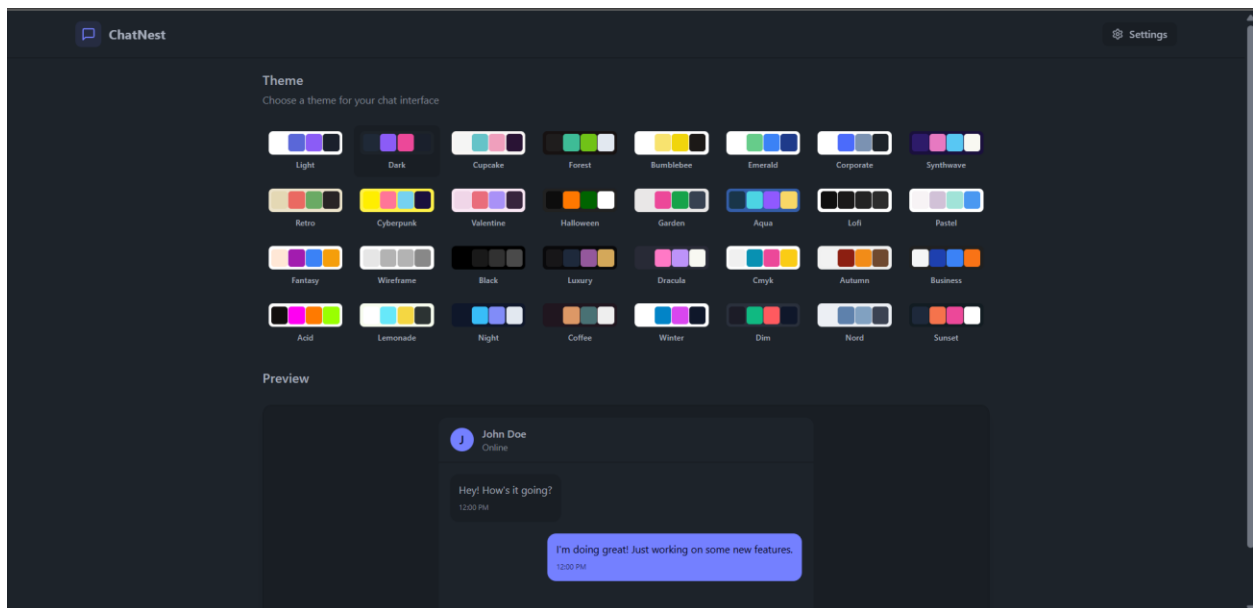
Create Account

Already have an account? [Sign in](#)

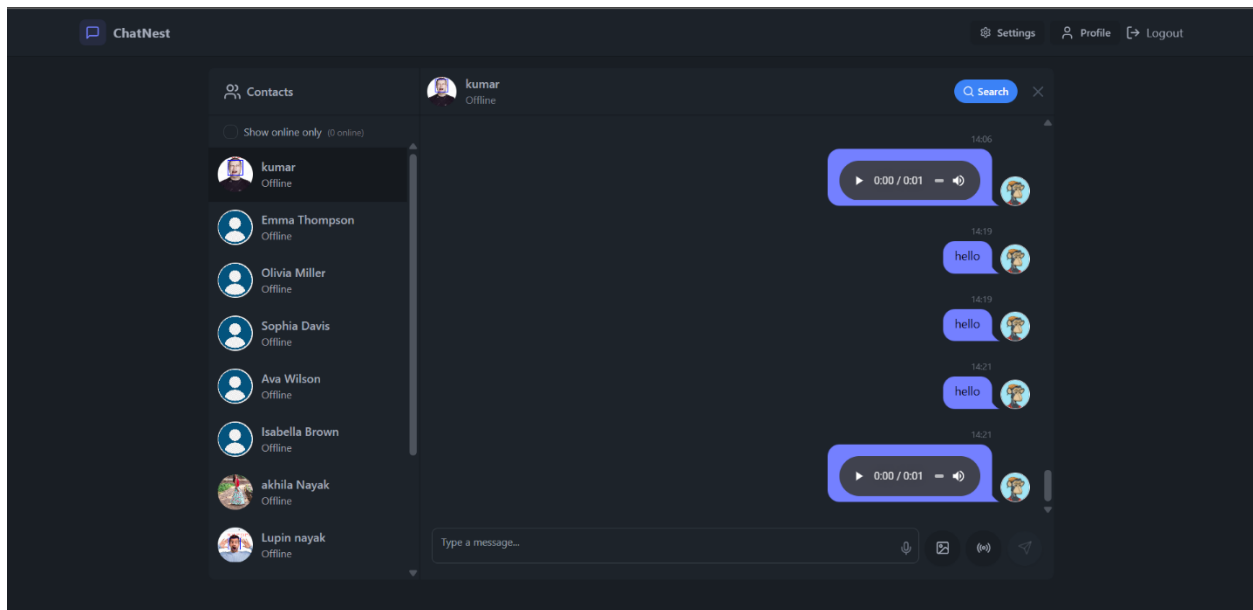
Join our community

Connect with friends, share moments, and stay in touch with your love once

Theme page



Chat page



Mr. Bhavani Sankar Panda
(Supervisor)