Review- 2
Web Based Chat Application

- Guide:
  - MADALA GURU BRAHMAM
- Team members:
  - K.Sasank – 21BCE8434
  - Boya Karthik Naidu - 21BCE9600
  - J.V.S Nithesh – 21BCE8090

Introduction & Problem Statement

Purpose & Target Users

Our chat application aims to provide real-time communication for teams. The app addresses the need for a centralized platform for instant messaging, file sharing, and group discussions.

Original Problem

The original problem was fragmented communication across different tools. This application consolidates communication into a single platform.

Specific User Needs

- Real-time communication
- Group chats
- File sharing

Key Performance Indicators

- Message delivery time
- User engagement

Week 1-2	Requirement Analysis & Design
Week 3-4	Backend & API Development
Week 5-8	Frontend Development
Week 9-10	Integration, Testing & Debugging
Week 11-12	Deployment & Final Adjustments

Results & Overall Progress: Project Timeline

Our project timeline demonstrates consistent progress across key milestones. We've successfully set up the database, completed the API, and designed the user interface. Feature completion, including user authentication, chat functionality, and notifications, is proceeding as planned. The milestones are on track for a successful project completion.

1 **Database Setup**

MongoDB setup and schema definition completed.

2 **API Completion**

Backend API endpoints developed and tested.

3 **UI Design**

Frontend UI designed and implemented.

System Architecture Diagram

MongoDB

Database for storing user data and messages.

Express.js

Backend framework for handling API requests.

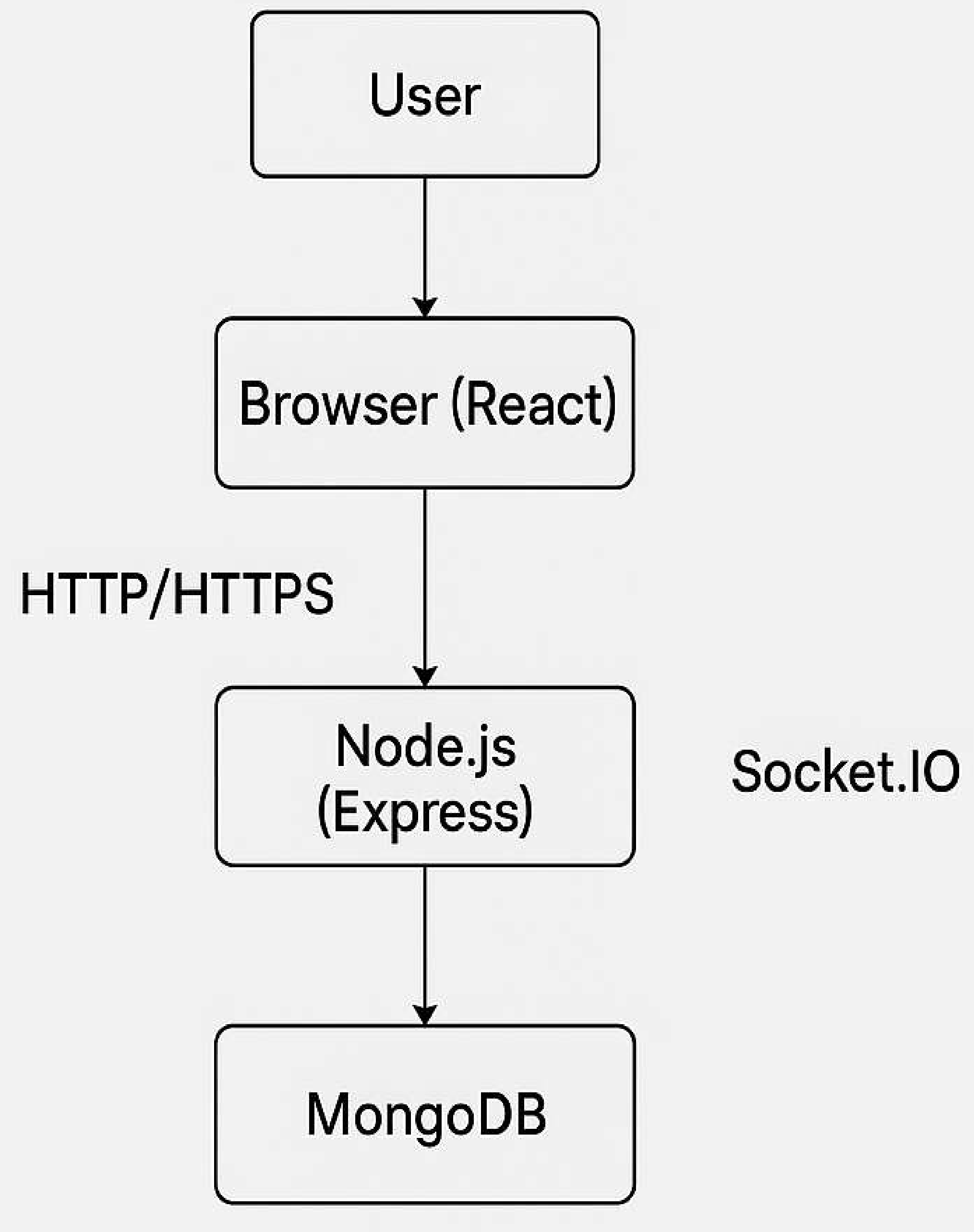
React

Frontend library for building the user interface.

Node.js

Runtime environment for executing JavaScript server-side.

The system architecture is based on the MERN stack: MongoDB, Express.js, React, and Node.js. The frontend communicates with the backend via API requests, which in turn interact with the MongoDB database. This architecture ensures scalability and maintainability.



System Architecture

Database Schema: MongoDB

Users

- _id (ObjectId)
- username (String)
- email (String)
- password (String)

Messages

- _id (ObjectId)
- chatroom (ObjectId)
- sender (ObjectId)
- content (String)

Chatrooms

- _id (ObjectId)
- name (String)
- users (Array of ObjectIds)

The MongoDB database schema includes collections for Users, Chatrooms, and Messages. Each collection contains key fields with appropriate data types, such as ObjectId, String, and Array. Indexing strategies optimize query performance across these collections.

Backend & API Design: Express.js & Node.js

1. **/users**

   Handles user registration and login.

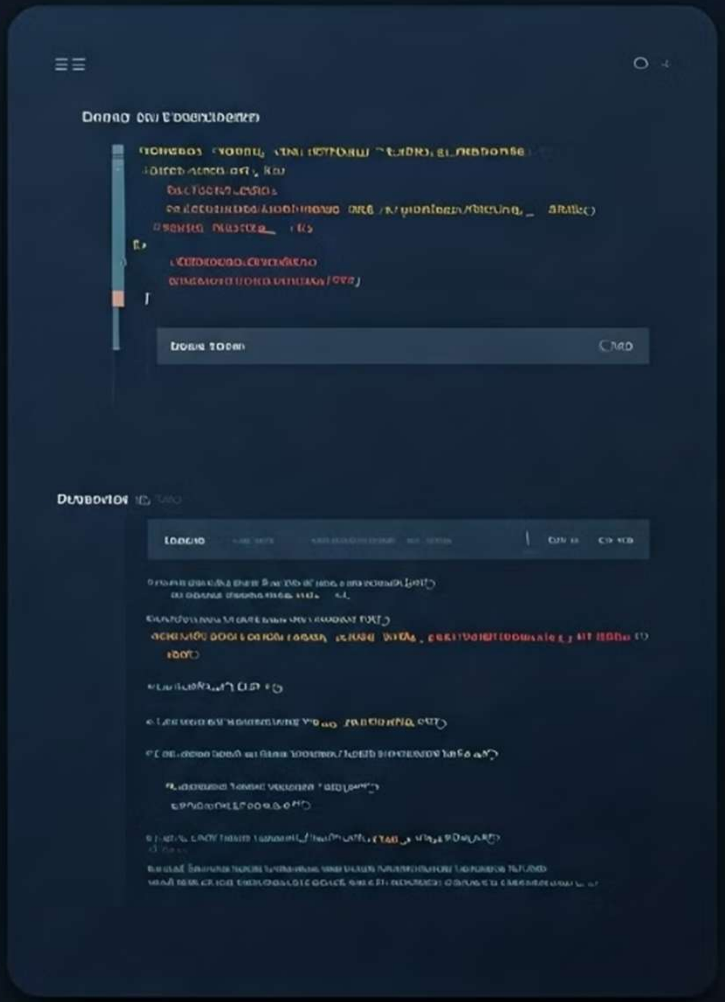
2. **/chatrooms**

   Manages chatroom creation and retrieval.

3. **/messages**

   Handles message sending and retrieval.

The backend API is designed using Express.js and Node.js. Key API endpoints include /users, /chatrooms, and /messages. Authentication and authorization are implemented using JWT (JSON Web Tokens). Socket.IO is used for real-time communication, and Mongoose facilitates interaction with MongoDB.



Backend & API Design

User Registration

POST /api/users/register requires username, email, and password. It outputs a User object and JWT token, validating email format and password strength.

User Login

POST /api/users/login requires email and password. It outputs a User object and JWT token for authenticated access.

Get All Chats

GET /api/chats retrieves all chats for a user, requiring JWT token authentication. Average response time is under 200ms under load.

Create New Chat

POST /api/chats creates a new chat, requiring an array of user IDs as input. It outputs a Chat object upon successful creation.

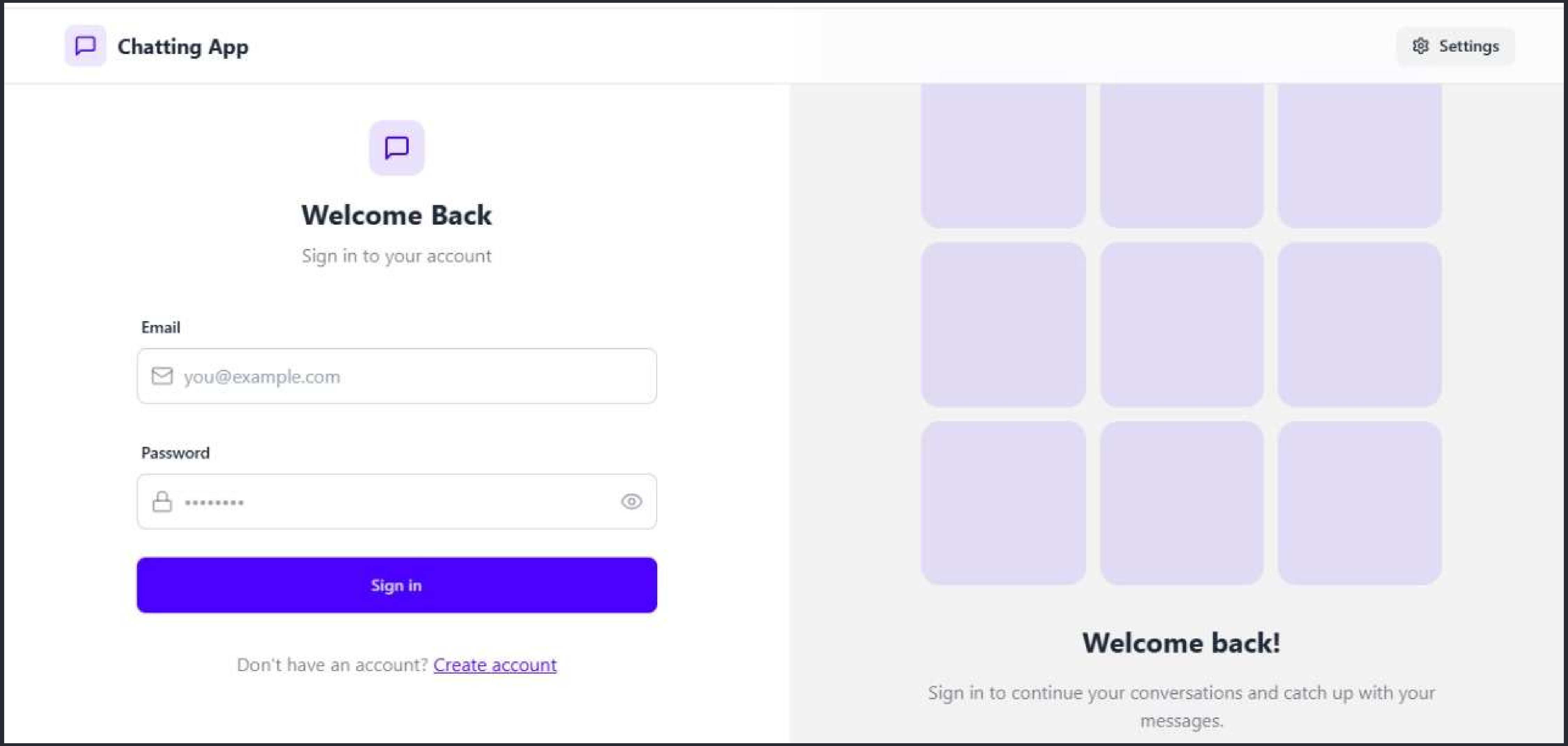
Frontend UI Snapshots: React

Login/Registration

Chat Interface

User Profile

The frontend UI is built using React. Key UI elements include login/registration, chat interface, and user profile. The UI is structured with React components and utilizes libraries such as Material UI, Redux, and Axios. The user interface is responsive across different devices, providing a consistent experience.



Chatting App

Welcome Back

Sign in to your account

Email

you@example.com

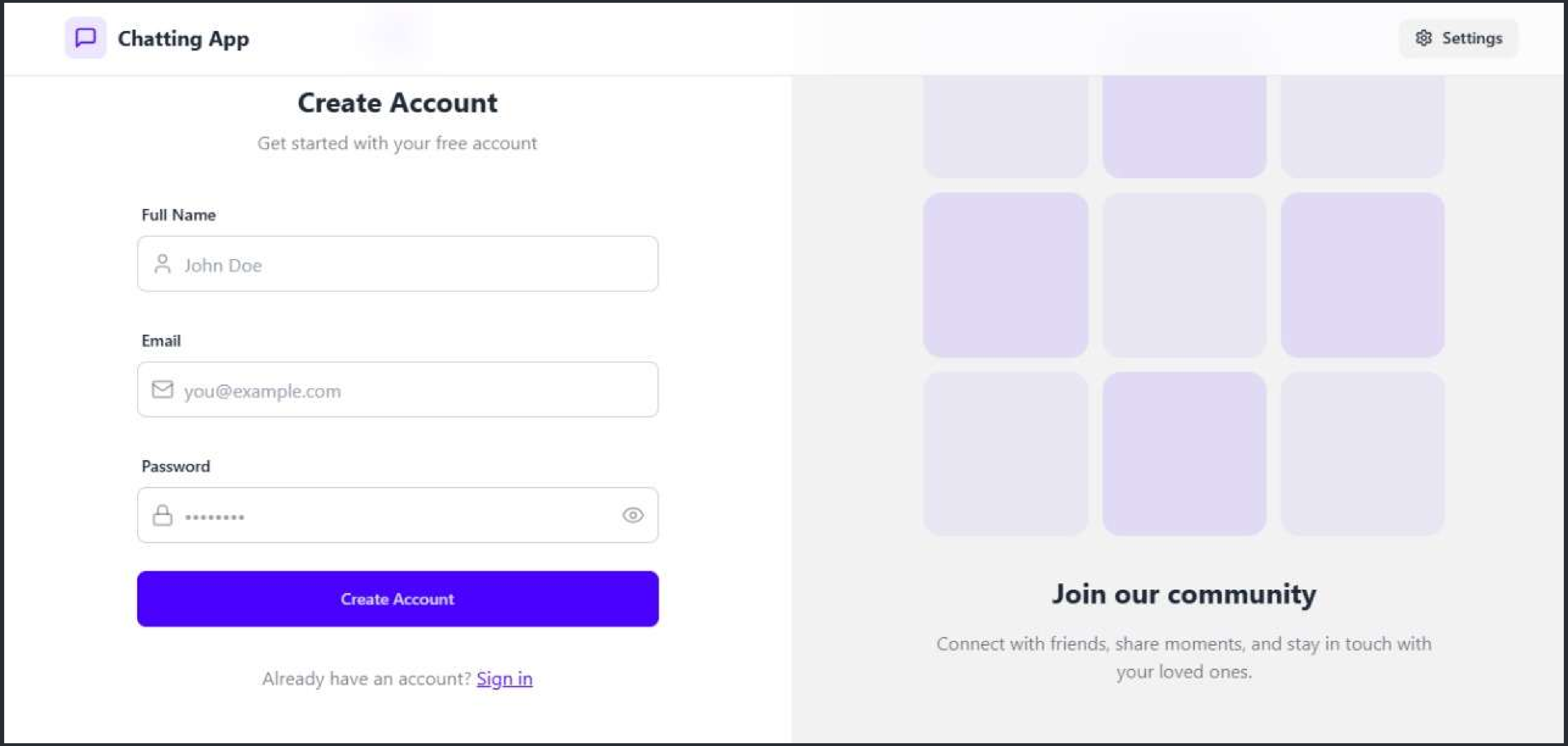
Password

••••••••

Don't have an account? Create account

Welcome back!

Sign in to continue your conversations and catch up with your messages.



Create Account

Get started with your free account

Full Name

John Doe

Email

you@example.com

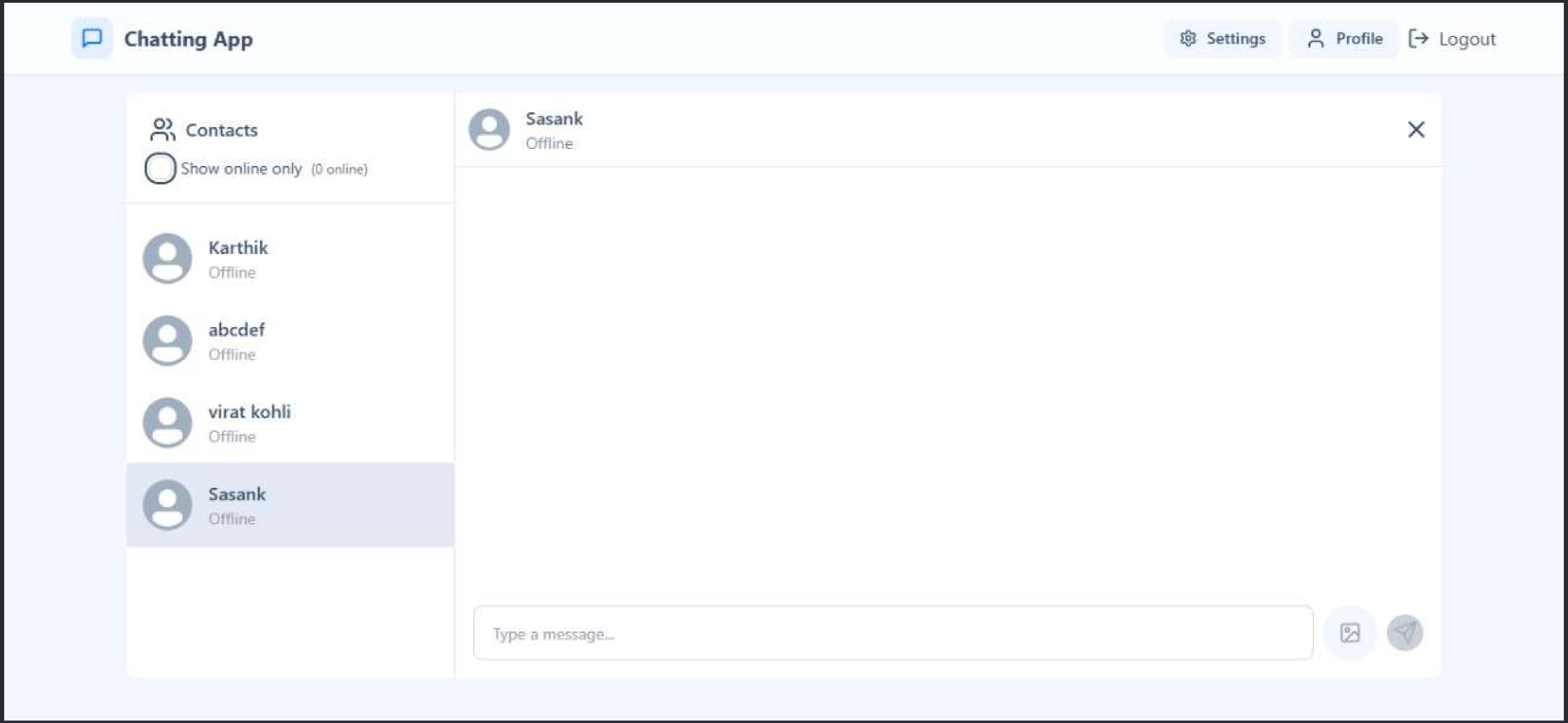
Password

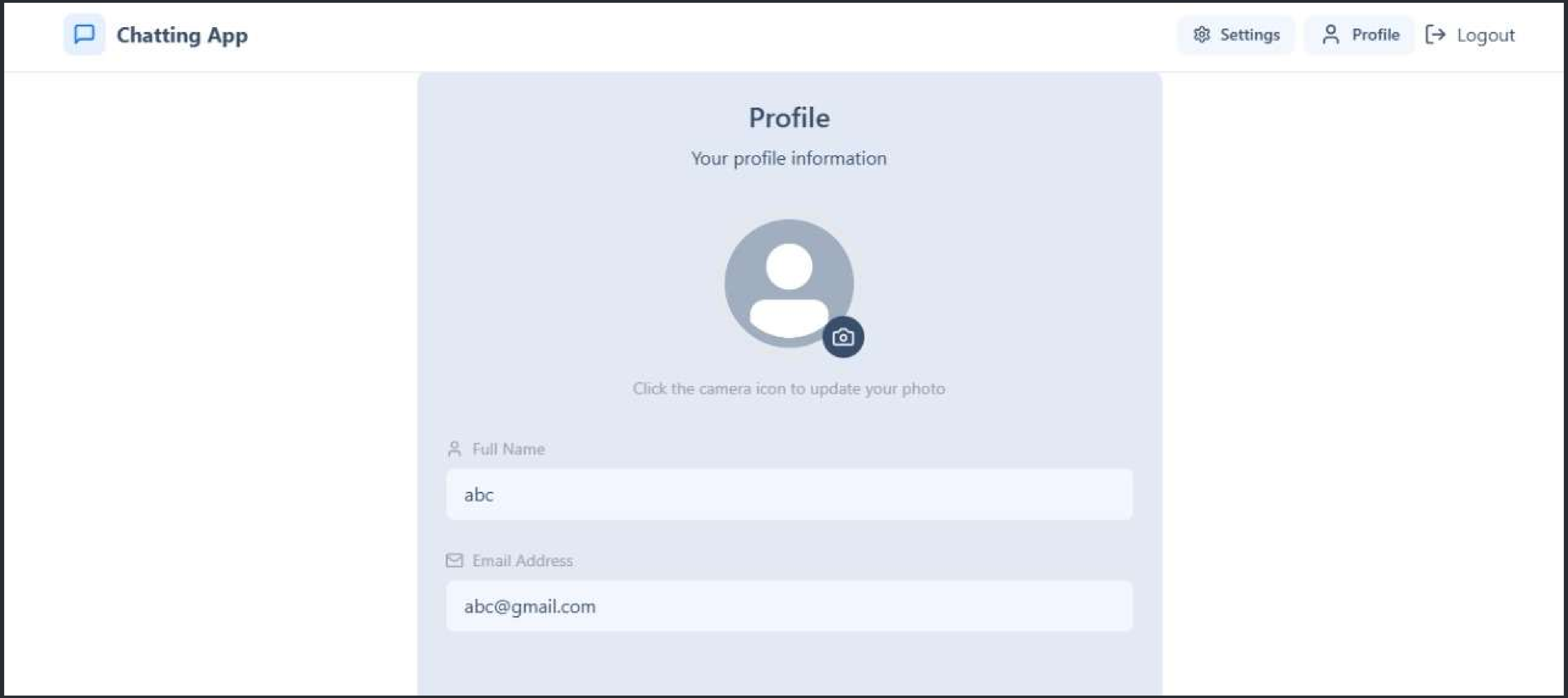
••••••••

Already have an account? Sign in

Join our community

Connect with friends, share moments, and stay in touch with your loved ones.


Chatting App
Settings
Profile
Logout
Contacts
Show online only (0 online)
Karthik
Offline
abcdef
Offline
virat kohli
Offline
Sasank
Offline
Sasank
Offline
Type a message...



Profile

Your profile information

Click the camera icon to update your photo

Full Name

abc

Email Address

abc@gmail.com

Originality & Unique Features

1. End-to-End Encryption
2. Customizable Themes
3. Advanced Search

Our chat app differentiates itself through end-to-end encrypted messaging using AES-256, ensuring user privacy and security. It also features a unique UI design with customizable themes, allowing users to personalize their experience. An advanced search functionality enables users to quickly find specific messages and files within the chat history.

Analytical Skills: Performance Optimization

10K

Messages/Second

500

Concurrent Users

20%

Latency Reduction

Performance optimization included load testing, identification of bottlenecks, and implementation of solutions such as database indexing and caching. We achieved substantial improvements. Load testing results demonstrate the application's ability to handle 10,000 messages per second with 500 concurrent users. We reduced latency by 20%.

Analytical Skills: Security Measures

Input Validation

Prevents malicious data entry.

Output Encoding

Protects against XSS attacks.

HTTPS

Ensures secure data transmission.

Security measures address threats such as XSS, CSRF, and data breaches. Implemented measures include input validation, output encoding, HTTPS, password hashing, and rate limiting. Vulnerability scanning results confirm compliance with security standards such as GDPR. Security is a top priority in our development process.

Conclusion

In this phase of our project, we have successfully implemented key functionalities, including real-time messaging using WebSockets, user authentication with JWT/Firebase Auth, and a structured database in MongoDB for efficient message storage and retrieval. The frontend, developed using React, provides a seamless chat experience, while the backend, built with Node.js and Express, ensures secure and fast communication. Throughout the development process, we encountered challenges such as managing real-time data efficiently, optimizing database queries, and ensuring secure authentication.