SYSTEM DESIGN DOCUMENT

for

CHATWAVE

Made by : Gurjaipal Singh
IIT DHARWAD

September 4, 2024

Contents

1 System Design Document

1.1 Overview

This document outlines the system design for a Messaging Service prototype named ChatWave with core and advanced features, including user registration, real-time messaging, voice and video calling, and more. The prototype leverages modern web technologies to ensure a seamless user experience.

1.2 Architecture

- **Frontend: Next.js**
  Next.js was chosen for its powerful features like server-side rendering (SSR) and static site generation, ensuring a fast, SEO-friendly, and performant frontend.
- **Backend: Node.js with Express.js**
  Node.js is well-suited for building scalable and efficient server-side applications. Express.js adds simplicity and flexibility in creating RESTful APIs.
- **WebSockets: Socket.io**
  Socket.io is ideal for implementing real-time bidirectional communication between clients and servers, crucial for features like messaging and real-time updates.
- **Authentication: Firebase Authentication**
  Firebase Authentication simplifies user authentication, offering secure login via Google and other providers with minimal configuration.
- **Database: PostgreSQL with Prisma ORM**
  PostgreSQL provides robust relational database capabilities, while Prisma simplifies database access and management, allowing seamless switching between multiple databases.
- **Cloud Storage: Firebase Storage**
  Firebase Storage was used for storing media files such as images, voice notes, and video recordings. It integrates well with Firebase Authentication.
- **UI Styling: Tailwind CSS**
  Tailwind CSS offers a utility-first approach to styling, enabling rapid UI development with customizable design elements.

- **Real-time Communication: WebRTC**
  WebRTC is used for implementing peer-to-peer voice and video calling features, ensuring low-latency communication.

1.3 System Components

- **User Authentication:** Managed by Firebase Authentication, supporting Google login.
- **Messaging Service:** Real-time chat powered by Socket.io for sending and receiving messages.
- **Voice and Video Calling:** WebRTC handles peer-to-peer communication for voice and video calls.
- **Database Management:** Prisma ORM with PostgreSQL for efficient and flexible database management.
- **File Storage:** Firebase Storage for managing and storing images, voice notes, and other media files.
- **UI Components:** Tailwind CSS for building a clean and responsive user interface.

1.4 Data Flow

1. **User Authentication:** Users authenticate via Google, and Firebase issues a secure token.
2. **Database Interaction:** Prisma ORM interacts with PostgreSQL to store and retrieve user data, messages, and media references.
3. **Real-Time Messaging:** Socket.io manages real-time communication, ensuring instant delivery of messages and real-time updates.
4. **Media Handling:** Firebase Storage is used to store and retrieve images, voice notes, and other media, with URLs stored in the database.
5. **Voice and Video Calls:** WebRTC establishes peer-to-peer connections for voice and video calls, with signaling handled by the server.

2 Setup and Deployment Guide

2.1 Prerequisites

- **Node.js:** v14.x or later
- **PostgreSQL:** v13.x or later
- **Firebase Account:** Set up Firebase Authentication and Firebase Storage
- **Prisma:** Installed globally or locally within the project

2.2 Installation Steps

1. **Clone the Repository:**
   git clone https://github.com/Gurjaipal17/ChatWave
   cd ChatWave

2. **Setup Client:**
   cd client
   yarn

3. **Setup Server**
   cd server
   yarn
   npx prisma init
   npx prisma generate

4. **Run the Backend Server:**
   yarn start

5. **Run the Development Server:**
   yarn dev

2.3 Dependencies and Libraries

- **Next.js**: Framework for server-rendered React applications.
- **Socket.io**: Enables real-time, bidirectional communication between web clients and servers.
- **Node.js**: JavaScript runtime environment for building backend services.
- **Firebase**: Provides authentication, storage, and real-time database services.
- **Prisma**: ORM for PostgreSQL, simplifying database management and queries.
- **Tailwind CSS**: Utility-first CSS framework for rapid UI development.
- **PostgreSQL**: Relational database system for storing structured data.
- **WebRTC**: Real-time communication protocol for voice and video calling.

2.4 Why These Technologies Were Chosen

- **Next.js** provides a robust framework for building fast and scalable web applications, making it ideal for this project.
- **Socket.io** is essential for real-time messaging and communication, ensuring an interactive user experience.
- **Firebase** offers secure and reliable authentication and storage services, simplifying user management and media handling.
- **Prisma ORM and PostgreSQL** offer flexible and powerful database management capabilities, crucial for handling the app's data efficiently.
- **Tailwind CSS** allows for rapid and consistent UI development, ensuring a clean and intuitive user interface.
- **WebRTC** is chosen for its low-latency peer-to-peer communication, essential for voice and video calling features.