Chat App Project Documentation

Project Title: Chat App

Version: 1.0

Author: Ashar Ul Haq <backend>

---

1. Project Scope

The Chat App is a simple, yet functional, one-on-one messaging platform that enables real-time communication between users. The application is designed to demonstrate core functionalities such as user authentication, real-time messaging via WebSockets, and secure data handling using RESTful APIs. The project serves as a foundation for more advanced chat features and can be extended with multimedia messaging, and notifications.

2. Project Objective

The objective of this project is to build a functional chat application that allows users to communicate in real-time via one-on-one messaging. The project also includes features like user profile updates and password change functionalities. The project started on **09/08/24** and was completed on **19/08/24**.

3. Key Features

- **User Authentication**: Secure sign-up and login functionality using JWT tokens.
- **One-on-One Messaging**: Users can send and receive messages in real-time.
- **Chat History**: Users can view previous messages when opening a chat with another user.
- **Real-Time Updates**: Active users are displayed with an indication of their online status.
- **User Profile Update**: Allow users to update their profile information.
- **Change Password**: Enable users to change their passwords securely.

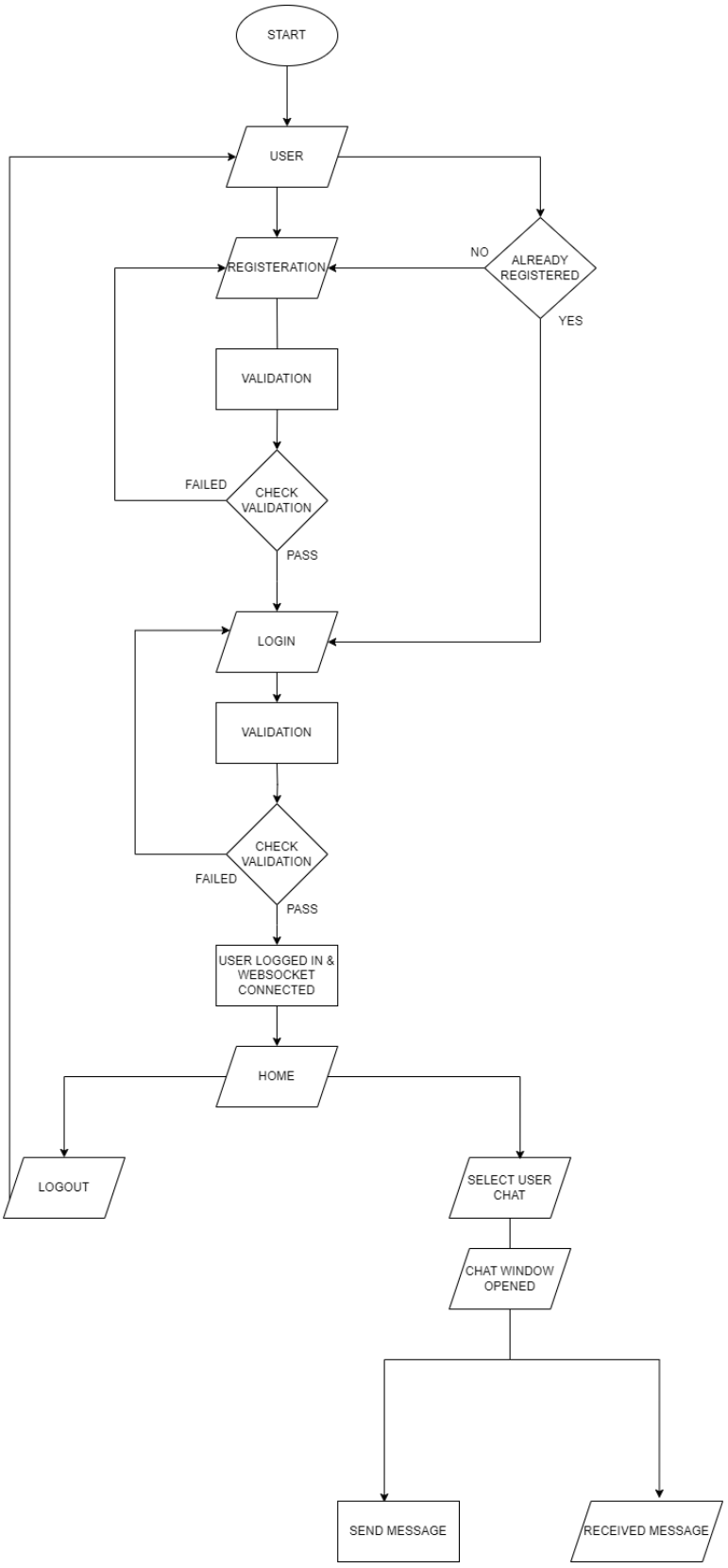
4. Technologies Used

- **Node.js**: Server-side JavaScript runtime for building the backend.
- **Express**: Web framework for Node.js, used to create RESTful APIs.
- **Prisma**: ORM for database management and querying with PostgreSQL.
- **PostgreSQL**: Relational database for storing user data, messages, and chat records.
- **WebSockets**: Protocol for real-time communication between the client and server.
- **Zod**: Validation library for schema validation in the backend.
- **JWT (JSON Web Token)**: Used for secure authentication and session management.

5. Backend Flow Summary

1. **User Authentication**: Users sign up and log in using JWT tokens for session management.
2. **WebSocket Connection**: After successful login, a WebSocket connection is established for real-time messaging.
3. **Chat Handling**: When a user sends a message, the message is saved to the database with relevant metadata (sender, recipient, timestamp).
4. **Chat Retrieval**: On chat initiation, previous messages are retrieved from the database if a chat exists, otherwise, a new chat is created.
5. **Unread Messages**: Unread messages are displayed to the recipient with a white dot when they receive a new message.
6. **Profile Management**: Users can update their profile information and change their passwords securely via dedicated endpoints.

FLOW DIAGRAM:



6. Folder Structure

```
/chat-app                                   # Root directory of the chat application
├── /.vscode                                # VS Code configuration files
├── /dist                                   # Compiled and build output files
├── /node_modules                           # Project dependencies installed by npm
├── /prisma                                 # Prisma configuration and schema files
│   └── schema.prisma                       # Prisma schema file defining the data models
├── /src                                    # Source code of the application
│   ├── /middlewares                        # Middleware functions for Express
│   │   └── auth.middleware.ts              # Middleware for authentication
│   ├── /modules                            # Main application modules
│   │   ├── /auth                           # Authentication module
│   │   │   ├── auth.controller.ts          # Handles authentication logic
│   │   │   ├── auth.route.ts               # Defines authentication routes
│   │   │   ├── auth.schema.ts              # Validation schemas for authentication
│   │   │   └── auth.service.ts             # Business logic for authentication
│   │   ├── /chat                           # Chat module
│   │   │   ├── chat.controller.ts          # Handles chat-related requests
│   │   │   ├── chat.route.ts               # Defines chat routes
│   │   │   ├── chat.schema.ts              # Validation schemas for chat
│   │   │   └── chat.service.ts             # Business logic for chat
│   │   ├── /message                        # Message module
│   │   │   └── message.service.ts          # Business logic for handling messages
│   │   ├── /user                           # User management module
│   │   │   ├── user.controller.ts          # Handles user-related requests
│   │   │   ├── user.route.ts               # Defines user routes
│   │   │   ├── user.schema.ts              # Validation schemas for users
│   │   │   └── user.service.ts             # Business logic for user management
│   │   └── /websocket                      # WebSocket server setup
│   │       ├── websocket.server.ts         # WebSocket server configuration
│   │       └── ws.schema.ts                # WebSocket-related validation schemas
│   ├── /utils                              # Utility functions and configuration helpers
│   │   ├── db.util.ts                      # Database utility functions
│   │   └── env.util.ts                     # Environment variable configuration helper
│   └── index.ts                            # Main entry point of the application
├── .env                                    # Environment variable configuration file
```

8. Conclusion

This documentation outlines the basic structure and flow of the Chat App project. The project provides a solid foundation for developing more complex chat applications and demonstrates the integration of various technologies to achieve real-time communication, secure authentication, and efficient data management. Further enhancements could include multimedia messaging, and notifications, expanding the app's functionality and user experience.

Github link: https://github.com/AsharUlHaq/chat-app-PEN