**1. Project Overview**

The real-time chat application is built using **Angular 19**, **WebSockets (Socket.IO)**, and **NgRx** for state management. It includes features like:

* **Real-time Messaging**
* **Typing Indicators**
* **Offline Push Notifications**
* **User Authentication**
* **Private Messaging**
* **Emoji Support**

**2. High-Level Architecture**

The application follows a **modular architecture** with:

* **Frontend (Angular 19 + NgRx) for UI & State Management**
* **WebSockets (Socket.IO) for real-time communication**
* **Backend (Node.js + Express + MongoDB/MySQL) for data storage**

**Technology Stack:**

| **Layer** | **Technology** |
| --- | --- |
| Frontend | Angular 19, NgRx, Bootstrap |
| Backend | Node.js, Express, Socket.IO |
| Database | MongoDB/MySQL |
| State Mgmt | NgRx |

**3. Component & State Design**

**3.1 Angular Components**

| **Component** | **Responsibility** |
| --- | --- |
| AppComponent | Root component |
| ChatComponent | Main chat UI, messages list |
| MessageInputComponent | Handles message input field |
| TypingIndicatorComponent | Shows when a user is typing |
| UserListComponent | Displays active users |

**3.2 NgRx State Management**

**Actions**

* [Chat] Send Message
* [Chat] Receive Message
* [Chat] Set Typing Status
* [Auth] Login
* [Auth] Logout

**Reducers**

* **Chat Reducer:** Handles incoming/outgoing messages, typing status.
* **Auth Reducer:** Manages user authentication state.

**Selectors**

* selectMessages: Get chat messages.
* selectTypingUsers: Get users currently typing.
* selectCurrentUser: Get logged-in user info.

**4. Design Flow Diagram**

**4.1 Message Flow**

graph LR

User1 -- Types Message --> ChatComponent

ChatComponent -- Dispatch Action --> NgRx Store

NgRx Store -- Updates State --> ChatComponent

ChatComponent -- Emit Event --> WebSocket Server

WebSocket Server -- Broadcast Message --> Other Users

Other Users -- Receive Event --> ChatComponent

ChatComponent -- Update Store --> UI Updates

**4.2 Typing Indicator Flow**

graph LR

User1 -- Starts Typing --> ChatComponent

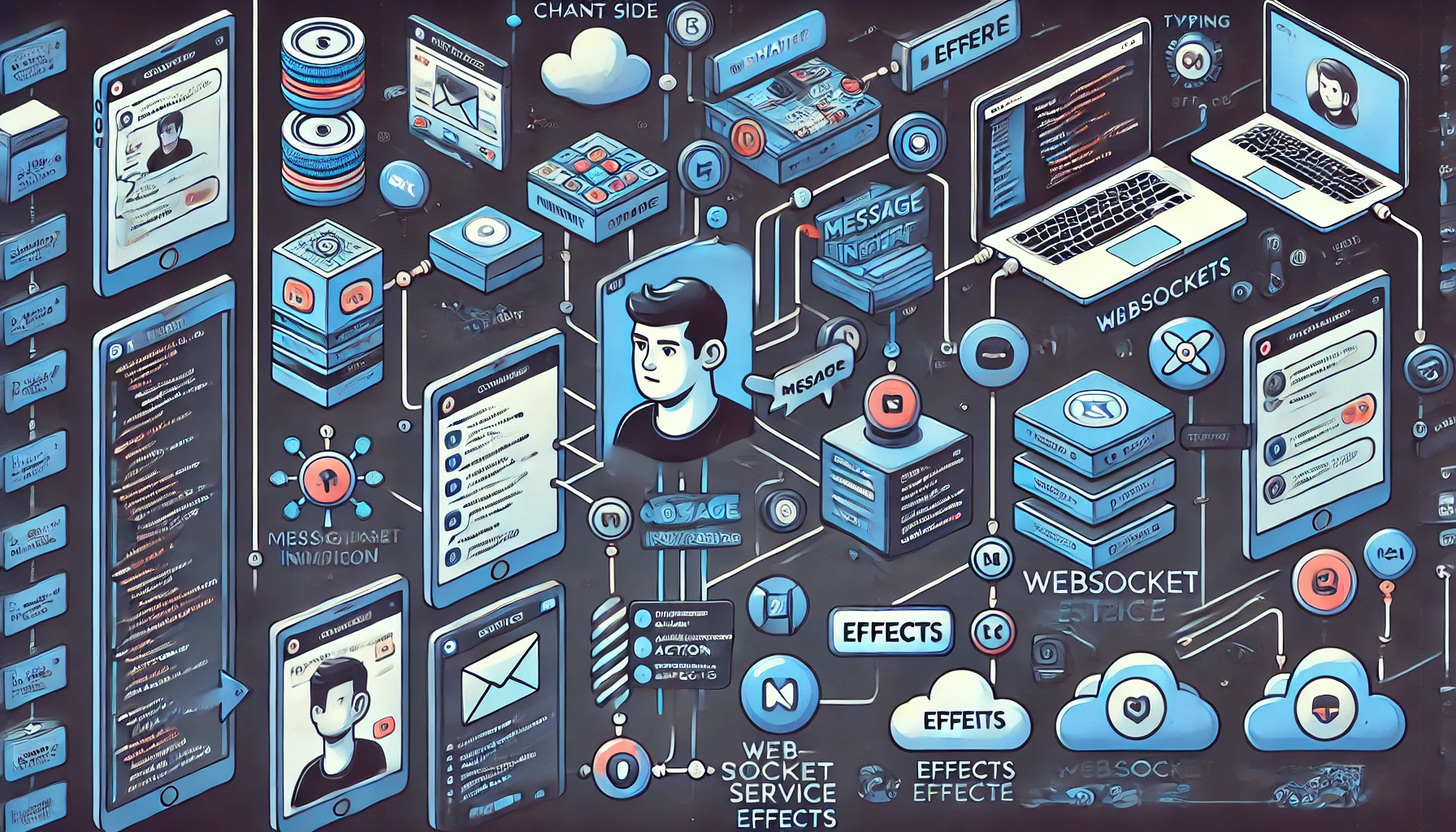
ChatComponent -- Dispatch [Set Typing Status] --> NgRx Store

NgRx Store -- Emit Event --> WebSocket Server

WebSocket Server -- Broadcast Typing Event --> Other Users

Other Users -- UI Updates --> TypingIndicatorComponent

**4.3 Design Flow Diagram**

****

**5. Conclusion**

This document outlines the design flow of the **Real-time Chat Application**, detailing the component structure, state management, and WebSocket communication flow. The **design flow diagrams** help visualize message and typing indicator interactions, ensuring a clear implementation strategy.