Design Document: TextZ

1. Overview:
The Real-time Chat Application is a platform that enables users to communicate instantly with each other. It utilizes technologies like Socket.IO for real-time messaging, Node.js for the backend, MongoDB for data storage, and JWT along with Google Auth for user authentication.

2. Goals:
- Enable real-time messaging between users.
- Provide secure user authentication using JWT and Google Auth.
- Allow users to create and join chat rooms for group communication.
- Implement notifications for new messages.
- Support emojis in messages for enhanced expression.
- Provide a profile page for users to manage their avatar and display name.
- Implement search functionality for finding users and chat rooms.
- Ensure responsive design for optimal user experience across devices.

3. Architecture:
The architecture of the Real-time Chat Application consists of several components:

- Frontend:
  - Utilizes React.js for building the user interface.
  - Implements responsive design using CSS frameworks like Tailwind css.
  - Handles user interactions and communicates with the backend API.

- Backend:
  - Built with Node.js and Express.js for handling HTTP requests.
  - Utilizes Socket.IO for real-time communication between clients.
  - Implements user authentication and authorization using JWT and Google Auth.
  - Handles CRUD operations for users, chat rooms, and messages.
  - Integrates with MongoDB for data storage and retrieval.

-Database:
  - Uses MongoDB for storing user data, chat room information, and messages.
  - Provides a scalable and flexible NoSQL database solution.
  - Ensures data persistence and reliability.

4. Features:
- Real-time Messaging:
  - Users can send and receive messages instantly.
  - Socket.IO enables real-time bidirectional communication between clients and the server.

- User Authentication:
  - Supports user registration, login, and logout using JWT tokens and Google Auth.

- JWT tokens are used to authenticate API requests for authorized users.

- Group Creation:
- Users can create chat rooms and invite others to join.
- Each chat room has its own unique identifier and can be customized by the creator.

- Notifications:
- Users receive notifications for new messages in chat rooms they are part of.
- Notifications are delivered in real-time using Socket.IO.

- Emojis:
- Supports sending and receiving emojis in messages for expressive communication.
- Emojis can be selected from a predefined set or entered manually.

- Profile Management:
- Users can update their avatar and display name on their profile page.
- Changes to the profile are reflected in chat rooms and messages.

- Room Creation:
- Users can create chat rooms for private or group conversations.
- Room creators have the ability to set room permissions and invite others to join.

- Search Functionality:
- Provides a search feature for finding users and chat rooms.
- Users can search by username, display name, or room name.

- Responsive Design:
- The application is optimized for various screen sizes and devices.
- Utilizes responsive design techniques to ensure consistent user experience across platforms.

5. Conclusion:

The Real-time Chat Application provides users with a modern and efficient platform for instant communication. By leveraging technologies like Socket.IO, Node.js, MongoDB, and React.js, the application offers a seamless user experience with features such as real-time messaging, user authentication, group creation, notifications, emojis, profile management, search functionality, and responsive design.