**Detailed Requirements**

**Name:** Yazan Agbaria

**Project:** Real-Time Chat Application

**GitHub Link:** <https://github.com/YazanIgbariya/Detailed-Design>

**Functional/non-Functional and Technological Requirements: Real-Time Chat Application**

**Functional Requirements**

**1.1 Key Features**

1. **User Authentication and Authorization:**
   * Secure registration and login functionality.
   * JWT-based authentication for secure login processes.
2. **Real-Time Messaging:**
   * One-to-one and group chat functionality using WebSocket technology.
   * Delivery indicators (sent, delivered, read).
3. **Group Management:**
   * Ability to create, manage, and join chat groups.
   * Admins can add or remove participants.
4. **Multimedia Sharing:**
   * Support for sharing images, videos, and files.
   * Preview capability for multimedia content within the chat interface.
5. **Real-Time Language Translation:**
   * Support for basic message translation between one or two languages.
6. **Gamification:**
   * Optional basic gamification features for initial user engagement.
7. **Notifications:**
   * Push notifications for new messages and group activities.
8. **Search Functionality:**
   * Search messages, users, and groups.

**1.2 Use Cases**

1. **Use Case: Sending a Message:**
   * **Pre-conditions:** User is authenticated and has an active chat session.
   * **Basic Flow:** User types a message, clicks send, and the message is delivered in real-time.
   * **Alternate Flow:** If the recipient is offline, the message is queued and delivered upon reconnection.
2. **Use Case: Joining a Group Chat:**
   * **Pre-conditions:** User is authenticated.
   * **Basic Flow:** User receives an invite or finds a public group and joins.
   * **Post-conditions:** User can send messages and view the group's chat history.

**Technological Requirements**

**2.1 Programming and Frameworks**

* **Frontend:**
  + Frameworks: React.js.
  + Libraries for real-time UI updates (e.g., Socket.IO-client).
* **Backend:**
  + Frameworks: Node.js with Express.js.
  + Real-time communication using WebSocket with Socket.IO.

**2.2 Database**

* **Database Management Systems:**
  + MongoDB for storing user data, messages, and group information.

**2.3 Cloud Services**

* **Hosting:** Custom hosting setup for scalable deployment.
* **APIs:**
  + Google Translate API for language translation.
  + Custom notification service implemented on the backend for flexibility.

**2.4 Security**

* **Authentication:** JWT-based authentication.
* **Encryption:** Ensure secure storage and transmission of user credentials and messages.

**2.5 Performance**

* Target response time: <500ms for message delivery.
* Support for up to 500 concurrent users during initial testing and scaling.

**2.6 Scalability**

* Prepare for future scalability using modular design.

**2.7 Monitoring and Maintenance**

* Basic monitoring through server-side logging for tracking errors and performance.

**Non-Functional Requirements**

* **Performance:**
  + The app should deliver messages with minimal delay, ensuring responsiveness.
* **Security:**
  + Ensure secure login and encryption for messages.
* **Usability:**
  + Design an intuitive interface for easy navigation.
* **Scalability:**
  + Prepare for future scalability using flexible technologies and services.
* **Reliability:**
  + Ensure basic error handling for common issues like message delivery failures.

**Stakeholders:**

* **End-Users**: Individuals and groups using the chat application.
* **Developers**: Building and maintaining the application.