**Task Description: Real-Time Chat Application with MongoDB Integration**

As part of the hiring process, you will develop a **real-time chat application** backend using **Node.js**, **Socket.IO**, and **MongoDB**. This task is designed to evaluate your ability to handle WebSocket communication, database integration, and backend development. Below are the detailed requirements and expectations for the task.

**Requirements**

1. **User Connection Management**:

• Implement WebSocket communication using Socket.IO to manage user connections.

• Assign each user a unique username upon joining (sent from the frontend) and track their online status.

2. **Real-Time Messaging**:

• Enable real-time message broadcasting to all connected users.

• Include the sender’s username in each message broadcast.

3. **Database Integration**:

• Use **MongoDB** to store:

• User details (e.g., username, timestamps for connection/disconnection).

• Chat messages (e.g., sender, message content, timestamp).

• Develop database schemas/models using **Mongoose** or native MongoDB driver.

4. **User Status Updates**:

• Notify all connected users whenever someone joins or leaves the chat.

• Provide a real-time updated list of active users to all connected clients.

5. **Chat History**:

• Implement an API endpoint to fetch chat history from MongoDB, sorted by timestamp.

6. **Error Handling**:

• Ensure robust error handling for connection failures, invalid data, or database errors.

**Deliverables**

1. **Backend Code**:

• A fully functional Node.js server with Socket.IO and MongoDB integration with a simple GUI for front end.

2. **Key Features**:

• Real-time messaging functionality.

• Persistence of messages and user data in MongoDB.

• Broadcasts for user connections and disconnections.

3. **API Endpoints**:

• A GET /messages endpoint to retrieve the last 50 chat messages from MongoDB.

4. **Documentation**:

• Clear instructions on how to run the project locally, including setup for MongoDB.

5. **Optional Enhancements (Bonus Points)**:

• Authentication for users using JWT or a similar mechanism.

• Typing indicators (e.g., “User is typing…”).

**Evaluation Criteria**

• **Code Quality**: Clean, well-organized, and follows best practices.

• **Real-Time Functionality**: Proper implementation of WebSocket events and real-time updates.

• **Database Integration**: Efficient use of MongoDB for storing and retrieving data.

• **Scalability**: Ability to handle multiple simultaneous connections.

• **Error Handling**: Robust error management for unexpected scenarios.

• **Documentation**: Clear setup instructions and code comments.

**Time Estimate: 3 Days**

Submit the completed project as a GitHub repository link or a zipped folder. Include any additional notes or considerations in a README.md file.