**Task: Develop a WhatsApp-like Chat Application with AI WhatsApp Bot Integration**

---

**Objective:**
Build a web-based chat application inspired by WhatsApp, enabling real-time messaging between users and seamless integration with an AI-driven WhatsApp agent (WhatsEase). The main goal is to assess a candidate's proficiency in Python (backend), web fundamentals, and their ability to adapt to the WhatsEase codebase.

---

### Requirements

**Backend:**

- Use **Python** (FastAPI or Flask preferred).
- Database: Use **MongoDB** or **PostgreSQL**.
- Implement a RESTful API for CRUD operations on messages and user accounts.
- Define a `Message` model with fields:
  - `message_id` (unique identifier)
  - `sender` (user's email)
  - `recipient` (user's email, or bot identifier)
  - `content` (string)
  - `timestamp` (datetime)
  - `status` (enum: "Sent", "Delivered", "Read")
  - `is_bot_response` (boolean)
- User authentication using **JWT** and session management.
- Integrate a basic AI bot structured for WhatsApp (WhatsEase) using a simple intent-response framework. The bot should reply contextually to user questions.
- Implement **WebSocket** (preferably Socket.IO or plain WebSocket) for real-time message transfer (not abstracted through services like Firebase).
- Logging of user and bot activities (message sent, delivered, read).

---

**Frontend:**

- Use **Vanilla JavaScript** or **React** (no UI frameworks like Bootstrap/Material-UI/Tailwind).
- Create responsive components: chat list, message viewer, input box.
- Validate forms/user input for sending messages.
- Show real-time updates of sent, received, and read messages using WebSocket.
- Routing with React Router (for inbox, chat, bot chat).
- Display error/loading states for better UX.
- Accessibility: Ensure ARIA roles and keyboard navigation.

---

**Additional Features:**

- Implement search/filter for chats and users.
- Activity log with recent actions ("User X sent a message to Bot Y", "Bot Y replied to User X").
- Bot interaction history and context retention.
- The system should efficiently handle **1,000+ simultaneous users**.
- Deploy on a cloud platform (Heroku, AWS, Vercel).

---

**Deployment & Submission:**

- Deploy the app and provide a working live URL.
- Submit a **GitHub repo** with clear setup instructions.
- README should cover environment setup, running the app locally, and deploying.

---

**Constraints:**

- No premade UI frameworks/packages for styling/layout.
- Real-time messaging must use native WebSockets or Socket.IO (no third-party real-time abstractions).
- App must scale for at least 1,000 users concurrently.

---

**Evaluation Criteria:**

- Quality of Python code (readability, modularity, docstrings).
- Good REST API design and adherence to HTTP standards.
- Real-time handling and performance via WebSockets.
- Usability and accessibility of frontend.
- Robust error handling and coverage of edge cases.
- Clear authentication and context-sensitive bot integration.
- Completeness of instructions and deployment.
- Innovation, initiative, and familiarity with web basics.

---

Build and deploy your WhatsApp-like chat app (with AI bot) and share the live URL and GitHub repo for evaluation.