Tasks for App Developers

1. Research & Planning

- **Choose the Tech Stack**:
  - Select a language/framework that supports **portability** (e.g., Python + Tkinter, Electron.js, JavaFX).
  - Ensure compatibility with **Windows 10/11** (no admin rights or installations needed).
- **Define Core Features**:
  - **Text Messaging**: Send/receive individual and group messages.
  - **Internet Connectivity**: Use sockets or a lightweight protocol (e.g., TCP/UDP, WebSockets).
  - **GUI**: Simple, intuitive interface (e.g., chat window, contact list).
- **Set Up GitHub**:
  - Create a repo for collaboration.
  - Use branches for features (e.g., `gui-dev`, `networking`).

2. Development

- **GUI Implementation**:
  - Design windows for:
    - Login/User selection.
    - Contact list (individuals/groups).
    - Chat window with message history.
  - Ensure responsiveness (scaling for different screens).
- **Networking**:
  - **Local Testing**: Simulate messaging over LAN first.
  - **Internet Connectivity**:
    - Use a **central server** (e.g., Flask for Python) or peer-to-peer (harder).
    - Handle IP/port configuration (e.g., manual input or auto-discovery).
  - **Security**: Basic encryption (e.g., TLS for sockets) if time permits.
- **Features**:
  - **Message Handling**:
    - Send/receive texts in real-time.
    - Store chat history locally (e.g., SQLite or text files).
  - **Group Chats**: Allow creating/joining groups.
  - **Notifications**: Visual alerts for new messages.

3. Testing & Debugging

- **Portability Testing**:
  - Verify the app runs on **fresh Windows 10/11 machines** without installations.
  - Bundle dependencies (e.g., PyInstaller for Python).
- **Functionality**:
  - Test messaging across different networks (LAN, Internet).

    - Stress-test with multiple users.
- **Edge Cases**:
    - Handle disconnections, invalid inputs, and server downtime.

4. Documentation

- **User Guide**:
    - How to launch the app (e.g., double-click executable).
    - Steps to connect (entering IP/server details).
- **Developer Notes**:
    - Setup instructions (e.g., `pip install -r requirements.txt`).
    - Code structure (e.g., `client.py`, `server.py`).

5. Final Deliverables

- **App Files**:
    - Portable executable (e.g., `.exe` or standalone `.jar`).
    - Server code (if applicable).
- **Demo Preparation**:
    - Record a short video showing:
        - Installation-free launch.
        - Sending/receiving messages locally and over the Internet.

Tools & Resources

- **Languages**: Python (Tkinter), JavaScript (Electron), Java (JavaFX).
- **Libraries**:
    - `socket` (Python), `net` (Java), `ws` (WebSockets).
    - `PyInstaller` (for packaging Python apps).
- **Tutorials**:
    - Python Chat App Tutorial
    - Electron.js Guide

Timeline (for App Team)

Week	Task
Week 7	Research, GUI mockups, GitHub setup.
Week 8	Core messaging + networking.
Week 9	Group chats, testing, debugging.
Week 10	Portability fixes, final docs.

Goal: A minimal but functional app that meets all requirements **without overcomplicating**. Focus on reliability over extra features.

Let me know if you need clarifications or additional details!