# **Things to do in a new App**

## **1. Sign Up & Login**

* **Identity Generation:**
  + When a user sets up app, the app generates a unique cryptographic key pair (public and private keys) on the device.
  + The **public key** acts as the user's identifier, while the **private key** is securely stored on the device.
* **Anonymous Identification:**
  + Users are not required to provide personal information like a phone number, email, or username.

### ****1. Sign-Up Process****

* During setup:
  + Creates the cryptographic key pair.
  + Saves the private key on the device securely, ensuring only the user can access it.
  + Optionally allows users to share their public key (via QR codes or links) to connect with others.

### ****2. Sign-In Process****

* There is no "sign-in" in the traditional sense because:
  + All user data is stored locally on the device, including their private key and message history.
  + When the app is launched, the user automatically "logs in" by accessing their locally stored data.

#### **Backup and Restore:**

* Allows users to back up their data (including the cryptographic keys) to a secure location.
* When restoring the backup on a new device, users essentially regain their identity and messages by importing their private key.

### ****3. User Authentication****

* Ensures authenticity through cryptographic signatures:
  + Every message a user sends is signed with their private key.
  + Recipients verify the authenticity of messages using the sender’s public key.

### ****4. Communication Without a Server****

* **Online Communication:**
  + If both users are online, messages are sent over the Internet, anonymizing traffic and avoiding centralized servers.
* **Offline Communication:**
  + When offline, use Bluetooth or Wi-Fi Direct for communication between nearby devices.

## **2. Contacts**

* Users connect with others by:
  + Exchanging their public keys (via QR codes or links).
  + Adding each other directly using these keys.
* Since there’s no server, connections are established peer-to-peer.

## **3. Chats**

### 1. ****Messaging (Main Chats Section)****

**Functionality:**

* **Peer-to-peer Communication:** Users can send messages directly to one another over Bluetooth, Wi-Fi Direct, or when both are online through other P2P networks.
* **Offline Message Queuing:** When a user is offline, messages are queued locally on the sending device, and the app attempts to deliver them once the recipient comes online.
* **Message Storage:** Use local databases like SQLite or decentralized storage solutions (like IPFS or Dat) to temporarily store messages until they're delivered.

**Methods:**

* **Queueing Messages Locally:** When a user sends a message but is offline or the recipient is not reachable, store it in local storage. Once the connection is re-established, the app will attempt to deliver the message.
* **Bluetooth/Wi-Fi Direct for Offline Communication:** If both devices are in close proximity, use Bluetooth or Wi-Fi Direct to exchange messages. These technologies can work in offline mode by establishing direct connections between devices.

**Technologies/Tools:**

* **Bluetooth / Wi-Fi Direct:** For offline message transmission between devices within close range.
* **SQLite:** Local database for queuing messages when the user is offline.
* **IPFS / Dat:** If you want to use a decentralized file storage mechanism, IPFS or Dat could store message data and other files in a decentralized way.

### 2. ****Group Chats Section****

**Functionality:**

* **Peer-to-peer Group Messaging:** Messages are delivered to group members using decentralized, direct connections. Each member of the group can connect to others using P2P networks.
* **Offline Handling in Groups:** When a group member is offline, their messages are queued locally, and once they are back online or in proximity to other members, the app synchronizes the messages.
* **Group Member Management:** Groups are managed through local connections, and the app could maintain a list of peers in a group on each device (without needing a central server).

**Methods:**

* **Group Membership Management:** You can handle group creation and membership via local peer-to-peer discovery. Users who wish to join a group could scan for group identifiers shared by other devices via Bluetooth/Wi-Fi Direct.
* **Data Syncing for Groups:** Sync group messages and status (e.g., read/unread) once the device is back online or in close proximity to other group members.

**Technologies/Tools:**

* **Local Databases (SQLite, LevelDB):** For storing group messages locally.
* **Bluetooth/Wi-Fi Direct:** To send messages to group members within close proximity, if they are offline.
* **Decentralized Protocols (like Libp2p or Mesh Networking):** If you wish to expand the system, mesh networking libraries like Libp2p can help set up decentralized P2P connections.

### 3. ****Poll & Event Function (Like WhatsApp)****

**Functionality:**

* **Create Polls and Events:** Users can create polls or events in a group or main chat. The app stores this data locally and shares it with others when they’re online or nearby.
* **Voting on Polls:** Users vote on polls, and the app stores responses locally until a connection is available for syncing with the rest of the group or other peers.
* **Event Reminders:** Events can be created with reminders, and once users come online, the app sends reminders about the event.

**Methods:**

* **Poll Creation:** When a user creates a poll or event, the data is stored locally and can be shared when devices are connected through Bluetooth, Wi-Fi Direct, or other P2P methods.
* **Polling Responses:** Each user’s response to a poll is stored locally on their device and sent to other participants when they are reachable.
* **Event Updates:** Event details and updates (e.g., RSVP, changes in time or location) are stored locally and propagated to users when they come online.

**Technologies/Tools:**

* **SQLite/Local Storage:** Store poll and event data locally for each user.
* **Bluetooth/Wi-Fi Direct:** For syncing polls and events between nearby devices when both are offline.
* **IPFS/Dat (Optional):** For decentralized storage of poll results and events if you need distributed and redundant data storage.

## **4. Feeds**

### ****1. Feed Section: Key Functionalities****

**User-Generated Content (Posts):**

* + Users can create posts (text, images, links, etc.).
  + Posts should be stored locally and, when possible, distributed to nearby devices (through Bluetooth or Wi-Fi Direct) or uploaded to decentralized storage like IPFS.

**Feed Display:**

* + Users' feeds will display posts in chronological order, similar to social media apps.
  + When users are online, the app syncs the feed from other peers, fetching new posts or updates.

**Offline Handling:**

* + When offline, posts are stored locally and shared once the user comes online or is in proximity to another user.
  + Users can view the content they have already synced in the feed while offline.

**Notifications and Updates:**

* + When new posts are created, the feed section can notify users (via local notifications or in-app updates) once the device is connected to the network or other peers.

**Likes, Comments, and Reactions:**

* + Users can like or comment on posts.
  + These interactions will be stored locally until synced with other peers.

### ****2. Technologies/Tools****

**SQLite/Local Database:**

* + Use SQLite (or another local storage solution) to store posts, likes, and comments when offline.

**IPFS (InterPlanetary File System):**

* + Store media files in a decentralized manner via IPFS. IPFS allows files to be stored and retrieved by their unique hash, ensuring decentralization and availability.

**Bluetooth/Wi-Fi Direct:**

* + For offline communication and syncing between nearby devices, use Bluetooth or Wi-Fi Direct. Devices can exchange data when they're in range, and this would be crucial for syncing posts, likes, comments, and other feed-related interactions.

**Mesh Networking (Libp2p, etc.):**

* + To implement a decentralized network where devices can connect and exchange information, even without a central server, use technologies like **Libp2p** or other mesh networking protocols. These will allow devices to share content in a decentralized manner, forming a peer-to-peer network.

### ****3. Data Flow for the Feed Section****

**Post Creation:**

* + User creates a post (text/media).
  + The app stores this post in the local database (SQLite).
  + If it’s a media post, it’s uploaded to IPFS, and the IPFS URL is saved.
  + The app attempts to sync the post with other peers once connected.

**Feed Display:**

* + When the user opens the feed section, the app fetches the posts from the local database.
  + It displays the posts, including any media fetched from IPFS.
  + The feed updates when new posts are synced from peers.

**Syncing:**

* + When users are online, they sync their feeds with others by exchanging posts and updates.
  + The sync process can occur in the background or when the user actively connects with a peer device via Bluetooth or Wi-Fi Direct.
  + If a post is liked or commented on, those changes are stored locally and synced once a connection is established.