Perfect 👍 Let’s restructure your project plan for **Option B (network-connected version first, mesh fallback later if time permits)**.  
I’ll divide tasks for **Person A, B, C, D** (like your earlier version) + write what exactly each role needs to deliver.

**🚨 Project: Disaster-Resilient Communication App (Phase 1 = With Network)**

**Tagline:** *“When networks fail, people connect.”*

**👩‍🎨 Person A – Mobile App UI/UX (Frontend)**

**Responsibilities:**

* Build the **React Native/Flutter app** interface.
* Key screens/features:
  + **SOS Button**: Large, one-tap, easy to use.
  + **Form (optional)**: Add victim’s name/message (short).
  + **Message Inbox**: List of sent/received SOS messages.
  + **GPS fetching**: Automatically attach location to SOS.
* Handle **SMS sending integration** (trigger SMS API or Android SMSManager).
* Later (stretch): Add map view in the app showing nearby SOS (pulled from Firebase).

**Tools:** React Native/Flutter, Expo (if RN), Android SMSManager API, Google Maps API (stretch).

**👨‍💻 Person B – Communication Layer (Network First)**

**Responsibilities:**

* Handle **message transmission** from app → outside world.
* Implement two modes (start with network version):
  1. **With Network**:
     + SOS goes out via SMS (or via Firebase if internet is available).
  2. **Fallback (if time)**:
     + Implement **Bluetooth/WiFi Direct mesh** using Android Nearby Connections API.
* Ensure SOS packet includes:
  1. Victim Name
  2. Message/Voice (compressed text/short voice)
  3. GPS location
  4. Timestamp

**Tools:** Android SMS APIs, Firebase SDK, (stretch) Nearby Connections API.

**👩 Person C – Backend + Rescue Dashboard**

**Responsibilities:**

* Setup **Firebase** (Firestore or Realtime DB + Hosting).
* Build **Rescue Dashboard (web app)**:
  + Live map with **GPS pins** of victims.
  + SOS message details (name, text, time).
  + Filter by “new” or “resolved” alerts.
* Connect **SMS → Firebase → Dashboard** flow (Person B pushes data to Firebase, Person C shows it).
* Ensure real-time updates for demo (when one phone sends SOS, dashboard refreshes immediately).

**Tools:** Firebase, React.js, Google Maps API.

**👨 Person D – Integration + Pitch/Demo**

**Responsibilities:**

* Ensure **end-to-end workflow** connects smoothly:
  + Device A (user SOS) → SMS/Internet → Firebase → Dashboard.
* Write **integration code** to push SOS from app into Firebase (when internet is available).
* Prepare **demo script + pitch slides**:
  + Problem (real disasters).
  + Solution (how your app works).
  + Live demo flow (step-by-step).
  + Stretch goals (mesh networking).
* Be **demo manager** during final presentation.

**Tools:** Firebase SDK, Google Slides/Canva.

**🕒 Suggested Timeline (till 4th Oct, ~36 hrs)**

**Phase 1 (0–12 hrs):**

* A: Build UI skeleton (SOS button, message form, inbox).
* B: Setup SMS sending (dummy messages).
* C: Setup Firebase + dummy dashboard with test entries.
* D: Draft pitch outline + integration skeleton.

**Phase 2 (12–24 hrs):**

* A: Add GPS + polish UI.
* B: Connect SOS → Firebase (through SMS/internet).
* C: Connect dashboard to Firebase, show live updates on map.
* D: Link mobile app with backend, test full SOS → Firebase → Dashboard flow.

**Phase 3 (24–36 hrs):**

* **End-to-end test with 2–3 phones.**
* Finalize pitch deck + practice demo.
* If time: add map view in app, test basic Bluetooth mesh.

✅ This way, by **Phase 2 itself you’ll already have a working prototype**, and Phase 3 is polish + stretch features.