Open-Source Initiative Mobile Stack Discussion
15 May 2025

Key Goals & Requirements for the Mobile App:

- Open-source, vendor-agnostic, easy-to-use, fast, and responsive.
- **Crucial:** Strong localization (for non-English speaking communities), accessibility (low vision, older/slower Android phones).
- **Core Features:** Displaying air quality data on a map, push notifications (e.g., for high pollution alerts), home screen widgets, showcasing community efforts, and providing accurate data (via backend QA/QC).

Main Discussion Points & Options Explored:

1. **PWA (Progressive Web App) using existing Vue/Capacitor/Ionic:**
    - **Pros:** Leverage existing web codebase and team expertise, faster initial development, resource-efficient.
    - **Cons:** Installation experience can be clunky for non-technical users. Reliability and full functionality of push notifications and especially widgets were questioned. May not meet the "feels like a native app" expectation.
2. **Native Apps (iOS & Android):**
    - **Pros:** Best performance, full access to device features (critical for widgets, lock screen features), familiar user experience.
    - **Cons:** Requires separate development efforts or a robust cross-platform solution, more resource-intensive. AirGradient currently has limited in-house native development experience.
3. **Hybrid Approaches:**
    - **Native Wrapper with Web View:** A native app shell primarily displaying the web content. This could ease installation and allow gradual native feature integration.
    - **Cross-Platform (e.g., Kotlin Multiplatform - KMP):** Could allow shared logic with native UIs, potentially offering a balance. The choice of stack influences which developers are attracted.
    - **Start Simple, Prototype:** Build a very simple, polished app focusing on a core use case (e.g., displaying local AQI via a widget) to prove value, gather user feedback, and attract community developers. Multiple small prototypes with different technologies could be explored.

Key Concerns & Considerations Raised:

- **Developer Resources:** AirGradient's current lack of deep native mobile expertise and the need to attract community contributors.

- **User Experience:** Installation process, performance on older/less powerful devices (especially in LMIC countries), and the importance of features like widgets for at-a-glance information.
- **Feature Feasibility:** Ensuring core needs like reliable push notifications and widgets are well-supported by the chosen stack.
- **Design Consistency:** Whether the app should use native OS components or a unified custom design.
- **Long-term Maintenance:** The implications of chosen stack on future development and support.

Proposed Next Steps:

1. **Define MVP Scope & Requirements:** AirGradient will create clearer design documents and mockups for a Minimum Viable Product (MVP), focusing on key features like the map, push notifications, and widgets. This will act as a "product manager" role.
2. **Community Prototyping:** Encourage interested community members to build small prototypes based on these MVP requirements using different technical approaches (PWA in wrapper, KMP, full native, etc.).
3. **Evaluate & Decide:** Compare the prototypes based on performance, developer experience, feature achievement, and alignment with project goals.
4. **Communication & Management:** Utilize Discord for discussions and polls, and GitHub for code. AirGradient will provide project management resources.
5. **Urgency for Core Functionality:** Participants stressed the importance of robust and reliable core features (like notifications for health warnings) for real-world impact.

The overall sentiment was to start with a well-defined, simple but highly useful core, gather community input and contributions through prototyping, and then make an informed decision on the technology stack.