Tech Stack Research: Mobile App Frameworks

Based on the MVP requirements (API integration, mapping, user accounts, community features, marketplace, notifications, multi-language) and future needs (IoT, AI, scalability), considering cost-effectiveness and performance:

Options:

1. **React Native:**
    - **Pros:** Cross-platform (iOS & Android from one codebase - cost-effective), large community, vast library ecosystem (including mapping, UI kits aligning with GrowWise style), potentially faster development cycles, JavaScript/TypeScript based (large developer pool).
    - **Cons:** Performance might lag behind native for highly complex animations or computations (less likely an issue for MVP), reliance on native bridges can sometimes add complexity.
    - **Scalability:** Good. Mature framework used by large apps.
    - **AI/IoT:** Good library support for API calls, background tasks, and potential native module integration for future hardware.
2. **Flutter:**
    - **Pros:** Cross-platform (iOS & Android), excellent performance (compiles to native code), expressive UI toolkit (good for custom designs like GrowWise), growing community, developed by Google.
    - **Cons:** Dart language (smaller developer pool than JS), ecosystem is newer than React Native (though rapidly growing).
    - **Scalability:** Very good, designed for high performance.
    - **AI/IoT:** Good support for API calls, growing support for IoT/native integrations.
3. **Native iOS (Swift) & Android (Kotlin/Java):**

- **Pros:** Best possible performance and access to native device features, latest OS updates immediately available.
- **Cons:** Requires separate development teams/efforts for iOS and Android (significantly higher cost and time), harder to maintain consistency.
- **Scalability:** Excellent platform-specific scalability.
- **AI/IoT:** Direct access to native SDKs for AI (CoreML, TensorFlow Lite) and hardware communication.

Recommendation Leaning:

For cost-effectiveness and rapid MVP development while maintaining good performance and scalability, **React Native** or **Flutter** are strong contenders.

- **React Native** might have a slight edge due to the larger developer pool and potentially more mature libraries for specific niche requirements.
- **Flutter** offers potentially better out-of-the-box performance and a highly flexible UI system.

Native development is likely too costly for the initial phase given the budget constraints mentioned.

Next Step: Research backend technologies.