**Development Decisions for GlucoGuide App**

**Overview**

The GlucoGuide app was developed to provide a mobile solution for managing user glucose data, initially targeting iOS via Expo Go on a Windows development environment. The following decisions shaped its architecture and implementation.

**1. Choice of React Native with Expo**

* **Reason**: React Native was selected for its cross-platform capabilities, allowing development for both iOS and Android from a single JavaScript codebase. Expo was chosen to simplify the setup, leveraging pre-built modules and Expo Go for testing without requiring a Mac for iOS initially.
* **Impact**: Enabled rapid prototyping and deployment on the developer's iPhone 15 Pro, though it limited standalone iOS builds without a Mac.

**2. Switch from expo-sqlite to AsyncStorage**

* **Reason**: Initial attempts to use expo-sqlite for local storage failed with a TypeError: openDatabase is not a function error, likely due to runtime incompatibilities in Expo Go on iOS from a Windows environment. AsyncStorage was adopted as a lightweight, Expo-supported alternative for key-value storage, avoiding native module issues.
* **Impact**: Ensured functionality with persistent user data (e.g., username, control level) and allowed progression despite the iOS limitation, with plans to expand to full user fields (minGlucose, maxGlucose, etc.).

**3. CSV Import for Initial Data**

* **Reason**: Implementing a CSV import (users.csv) was chosen to populate user data flexibly, supporting multiple users (e.g., "gooduser", "testuser") and aligning with the app’s goal of managing glucose-related profiles. The decision to copy the CSV from assets to the document directory addressed file accessibility in Expo Go.
* **Impact**: Enabled data persistence and testing, though required handling deprecation warnings (e.g., getInfoAsync) by switching to the legacy expo-file-system API.

**4. UI Design and Feature Prioritization**

* **Reason**: The UI was kept simple (login and dashboard screens) to focus on core functionality (user authentication and data display). A success alert was added to enhance user feedback, while full field display (e.g., carb ratio) was deferred to iterative updates.
* **Impact**: Provided a functional base for testing, with plans to refine based on user needs and add features like logout confirmation.

**5. iOS Development Constraints and Future Plans**

* **Reason**: The lack of a Mac on the Windows machine restricted iOS to Expo Go, prompting a decision to delay standalone builds. Research into virtual machines (e.g., Parallels) or cloud Mac services (e.g., MacStadium) was initiated to enable future .ipa file generation.
* **Impact**: Limited current deployment options but ensured progress, with a roadmap to address this limitation.

**6. Version Control with GitHub**

* **Reason**: Using GitHub for version control was chosen to track changes, facilitate collaboration, and provide a backup. The branch naming issue (master vs. main) was resolved to align with modern standards.
* **Impact**: Secured the codebase and enabled easy rollbacks or sharing, supporting long-term development.

**Conclusion**

These decisions balanced immediate functionality with long-term scalability. The app now supports basic user management on iOS via Expo Go, with plans to enhance data fields, improve UI, and enable standalone builds. Future iterations will address Mac setup and additional features based on user feedback.