Architectural Decisions

Prepare for

Kamini Sehmi

Prepared by

Kamalpreet Kaur, Anmol Sandhu, Manmohik Chahal, Avneet Kaur

School for Advanced Digital Technology

SAIT

Requested by

Kamini Sehmi

**Architectural Decision Record (ADR)**

### **1. Native, Web, or Hybrid App**

* **Decision**: Hybrid App
* **Rationale**: A hybrid approach allows development for both iOS and Android with a single codebase, saving time and resources while providing a consistent experience across platforms. This is ideal for a study planner app where users may access it on various devices.
* **Consequences**: The hybrid approach enables faster development and broader reach, but it may have slightly reduced performance compared to fully native applications. This could be a minor limitation but is generally acceptable for an app with moderate functionality like a study planner.

### **2. UI Framework**

* **Decision**: React Native
* **Rationale**: React Native is chosen for its cross-platform capabilities, allowing efficient development for both iOS and Android. The team’s familiarity with JavaScript and React also enables faster development and troubleshooting.
* **Consequences**: React Native’s wide adoption means good community support and access to resources. However, the team will need to keep up with React Native updates and best practices to maintain optimal performance and compatibility.

### **3. Backend Language**

* **Decision**: Node.js with Express
* **Rationale**: Node.js with Express is selected as a lightweight backend solution that handles basic CRUD operations effectively. This setup is straightforward, aligns with the team’s JavaScript expertise, and provides flexibility for future scaling if necessary.
* **Consequences**: Node.js with Express allows for quick, scalable development that integrates well with JavaScript-based frontends. However, it may require additional effort to handle extensive backend operations if the app’s needs expand significantly in the future.

### **4. Permissions**

* **Decision**: Notifications and Local Storage Permissions
* **Rationale**: The app requires notification permissions to alert students of upcoming assessments and deadlines. Local storage permission is needed to cache data locally for offline accessibility, ensuring the planner functions even without an internet connection.
* **Consequences**: Notifications improve user engagement by providing timely reminders, and local storage enhances usability by supporting offline access. However, permissions must be carefully managed to avoid overwhelming or annoying users with excessive notifications.

### **5. Data Storage**

* **Decision**: MySQL (Remote Database)
* **Rationale**: MySQL is selected as a structured remote database solution that supports reliable data storage and cross-device synchronization, essential for users managing tasks on multiple devices. Its robustness and scalability make it suitable for tracking tasks, reminders, and assessment schedules.
* **Consequences**: MySQL’s structured approach ensures data integrity and supports complex querying. This setup requires a stable internet connection for data access, and managing remote database security is crucial to protect user data.

### **6. Additional Frameworks or Technology Stacks**

* **Decision**: Google Analytics (or Simple Log Files)
* **Rationale**: For simplicity, we’ll use Google Analytics to capture basic usage data, or even simple logging for minimal tracking. This provides essential insights with minimal setup, reducing overhead while still allowing the team to monitor basic user engagement.
* **Consequences**: Using Google Analytics or basic logs keeps tracking simple and minimizes setup complexity, though it may not capture in-depth insights. The team may need to scale to more advanced analytics in the future if detailed user data becomes necessary.