Ryan Hatch  
CS 360 Mobile Architect & Programming  
October 27, 2024

**Event Tracker App Launch and Development Summary**

**Requirements and Goals:**

The Event Tracker app was built to make managing events easy. Users can add, view, and delete events with just a few taps, while receiving timely notifications to stay on top of things. The main goals were to create a smooth, organized experience for tracking events, complete with user authentication, an events database, and support for both notifications and SMS reminders.  
Key requirements for the Event Tracker include:

* User authentication for a secure way to access personal events and their data.
* Users and Events database management with CRUD operations for reading, adding, editing, and deleting entries.
* Permissions for Notifications and SMS integration for more reliable reminders.

**Application Details from Latest Build:**

* **Application ID:** com.event.track
* **Version Code**: 1
* **Version Name:** 1.0
* **APK Variants:**
  + **Release Build:** app-release.apk
  + **Debug Build:** app-debug.apk
* **Supported SDKs:**
  + **Minimum SDK Version:** API level 28 via Android 9.0.
  + **Maximum SDK Version for Dexing:** API level 34 via Android 14.
    - **Baseline Profiles:**
      * **API 28-30:** Developed and optimized for Android 9 – 11.
      * **API 31 and above:** Optimized for Android 12 and newer models.

**Supported Android Versions:**

The Event Tracker app supports Android versions from API level 28 to 34, covering devices from Android 9 through Android 14. Baseline profiles have been implemented separately for versions 28-30 and 31+, optimizing performance for each range. This setup helped to make the UX run more smoothly and optimize the performance on older Android versions while still making sure that the responsiveness on newer devices is also running with optimal performance.

**Permissions Requested:**

The app requests only essential permissions to support its core functionality. The “**Send SMS”** is to be able to properly send out event notifications, the “**Receive SMS” is** to manage responses for event confirmations, and “**Post Notifications”** to be able to properly deliver event reminders on time. Each permission has a clear purpose that is tailored towards enhancing the user experience (UX) while keeping privacy a priority.

**Monetization Plan:**

The app will be release ed under a **“freemium model”** to better balance the accessibility and premium options. It will have non-intrusive ads within the interface for free users. For any users who want an ad-free experience, a one-time in-app purchase will be available as an option to be able to remove ads entirely. Also, users will be able to unlock premium features through a one-time purchase, for example recurring events, export options, ability to store and manage a larger database of events, having more advanced notifications, and the ability to have more control and customization over events, reminders, and users.

**User Interface (UI) Design:**

The UI design prioritizes user-friendly navigation and simplicity:

* **Main Screen**: Guides users to log in or register a new account.
* **Login/Registration Screen**: Simple authentication to access the events.
* **Event Display Screen**: Displays all of the events in a clean list format with options to add, edit, or delete events.

The Event Trackers UI/UX was designed for straightforward and easy navigation, tailored towards keeping things simple and consistent.

**Development Approach and Techniques:**

The development approach for the Event Tracker mainly was tailored towards using a few main strategies. Incremental development allowed each of the core functions, like the login and event storage, to be built, tested, and debugged individually. This made sure that everything ran optimally for the UX and its UI design before I continued to develop more towards the app. A modular code structure kept the project organized, with database helper functions which helped to keep all of the interactions as simple and clear as possible. To improve performance, baseline profiles were tailored for Android API levels 28-30 and 31+, so that the app runs smoothly across different models and versions. Using these techniques helped to make development more efficient and helped to create a solid foundation for future development or additions, or even as a template for another application, especially because of its ability to function on a large range of different models of Android.

**Testing and Validation:**

Testing was done to make sure that each component in the app performed as I expected it to. Permission handling was also verified to make sure that each permission functioned smoothly without interfering with the UX. Database operations like data reading, saving, and deleting, were tested to make sure that the CRUD operations functioned efficiently. Error handling was also added and confirmed that login validations and permissions denials gave the users clear and helpful feedback. The testing phase helped me find and hone-in on specific edge cases to help make the apps UX more stable, reliable, and user-friendly.

**Workflow For Overcoming Challenges:**

One major challenge that I ran into was integrating SMS permissions without interrupting the user experience (UX). To fix this, I added a layered permission-checking system to make sure that the app only requested permissions when they were absolutely required or necessary, which in the end made the overall workflow more optimal.

**Demonstrating Knowledge and Skills:**

Integrating an app with a full database with UI components for managing events really showed a solid grasp of Android’s database handling and UI design. This project provided hands-on experience in building a smooth UI and UX that seamlessly connects what users see with everything running behind the scenes. It was a great way to learn about creating an experience that feels natural and efficient, while balancing both the user interface and behind-the-scenes elements that control the user experience.