Code Requirement Document

Requirement Title: App Installations System

Product Name: Miko Mini

Technology Stack: Vert.x

Objective:

Develop a system in Vert.x that automates the installation and updating of apps on Miko Mini, similar to Playstore/Appstore, without any user interface. The system will handle app installations sequentially, track the progress through various states, and automatically retry failed installations. The system will also provide detailed analytics for each installation.

Requirements:

1. **System Overview:**
    - The system will automate the installation of apps on Miko Mini based on the app data present in a database.
    - There will be no user interface; all operations will be performed in the background.
    - Apps will be installed one at a time to avoid conflicts.
    - The system should also handle automatic updates for apps when new versions are available.
2. **App Installation States:**
    - **SCHEDULED:** The app is queued for installation.
    - **PICKEDUP:** The app has been picked up for downloading and installation.
    - **ERROR:** An error occurred during installation. The system will retry the installation up to three times.
    - **COMPLETED:** The app installation has been successfully completed.
3. **Vert.x Code Requirements:**
    - **App Installation:**
        - Write Vert.x code to automate the installation of all apps listed in the database.
        - Ensure the installation process handles one app at a time.
    - **State Management:**
        - Implement state transitions in the code to move apps through SCHEDULED, PICKEDUP, ERROR, and COMPLETED states.

        - Ensure state transitions are logged with timestamps for tracking.
    - **Error Handling & Retry Logic:**
        - Implement logic to automatically retry an app installation up to 3 times if it enters the ERROR state.
        - After 3 failed attempts, trigger an email notification indicating the issue.
    - **Auto Updates:**
        - Implement logic to detect when a new version of an app is available in the database and automatically schedule it for update.
    - **Rescheduling:**
        - After all scheduled apps have been processed, automatically reschedule any apps in the ERROR state for reinstallation.
4. **Analytics & Logging:**
    - Implement logging within Vert.x to track state transitions (SCHEDULED, PICKEDUP, ERROR, COMPLETED) with timestamps.
    - Track retry attempts and error messages for each app.
    - Provide an interface (API or otherwise) to query this analytics data for reporting purposes.
5. **Notifications:**
    - Implement an email notification system in Vert.x that triggers an alert if an app fails to install after 3 retries.
    - The email should include details such as the app name, error encountered, and timestamps for each failed attempt.
6. **Concurrency Management:**
    - Ensure that only one app is picked up and installed at a time to prevent conflicts.
    - Vert.x's event-driven model should be leveraged to manage the queue and handle state transitions.
7. **Database:**
    - Use a database to store app details, installation states, and analytics data.
    - The database should be optimized for high write operations and allow for efficient querying of historical data.

Additional Considerations:

1. **Performance & Scalability:**
    - Ensure that the system is capable of handling multiple app installations efficiently while maintaining a low latency.
    - Design the system to be scalable to handle future growth in the number of apps.
2. **Error Handling:**
    - Implement detailed logging and error messages in Vert.x to aid in debugging failed installations.
    - Provide fallback mechanisms if the retry logic fails (e.g., sending more detailed logs in the email notification).
