Steps for Mobile Application Penetration testing of DIVA-Beta APK

1. **Setting Up the Android Emulator in Android Studio**:
    - First, I open Android Studio and navigate to the **AVD Manager** (Android Virtual Device Manager).
    - I create a new virtual device by choosing a hardware profile that suits my testing needs, like a Pixel device.
    - Then, I select a system image (preferably the latest stable Android version) and finalize the setup by naming the device and customizing any necessary settings.
    - Once the emulator is set up, I start it to ensure it's working correctly.
2. **Installing the DIVA-Beta APK Using Windows PowerShell**:
    - After launching the emulator, I navigate to the folder where the *`diva-beta.apk`* is stored on my Windows machine.
    - I open Windows PowerShell in that directory by right-clicking and selecting "Open PowerShell window here."
    - I use the command *`adb install diva-beta.apk`* to install the APK on the running emulator.
    - I verify the installation by checking the app drawer in the emulator for the DIVA app.
3. **Decompiling the APK with JADX-GUI**:
    - I open JADX-GUI on my Windows machine.
    - Using the GUI, I load the *`diva-beta.apk`* file.
    - I explore the decompiled source code, focusing on the *`activity`* files, manifest, and any other critical components that might reveal vulnerabilities.
    - I take notes on any suspicious code patterns, hard-coded credentials, or weak cryptographic practices.
4. **Extracting Stored Logs Using PowerShell**:
    - I return to PowerShell and run the command *`adb logcat > logs.txt`* to capture the log output from the Android emulator.
    - This command saves all the logs to a file named `logs.txt` in the current directory.
    - I analyze the logs for any sensitive information that might be exposed, such as unencrypted data, user credentials, or application errors that could be exploited.
5. **Performing SQL Injection to Bypass Login**:
    - With the app running on the emulator, I identify any input fields, especially login forms, that might be vulnerable to SQL injection.
    - I attempt to inject common SQL payloads, such as `' OR '1'='1` in the username and password fields.
    - I monitor the app's behavior, checking if it bypasses authentication and grants access to restricted areas.
    - I document my findings, noting down the specific payloads that were successful.
6. **Other Necessary Steps**:
    - **Exploring Insecure Data Storage**: I use `adb shell` to explore the file system of the emulator and check if the app stores sensitive data insecurely, such as unencrypted user data or hard-coded keys.

    - **Examining App Permissions**: I review the app's permissions in the `AndroidManifest.xml` file (decompiled using JADX-GUI) to identify any over-permissions that could be exploited.
7. **Final Documentation and Reporting**:
    - After completing all the tests, I compile a detailed report that includes my methodology, findings, screenshots, and any recommendations for mitigating the identified vulnerabilities.
    - This report is essential for conveying the risks and necessary fixes to the development team.

These steps ensure that I thoroughly assess the security posture of the DIVA-Beta APK, identifying potential vulnerabilities that could be exploited by malicious actors.