Test Plan for Posture Pal Application

Introduction

- **Objective**: This test plan outlines the procedures and strategies for testing the Posture Pal application to ensure that all features function as expected and meet user requirements. The primary objective is to ensure that the application delivers reliable posture monitoring and alerts for students, the target user group.
- **Scope**: This plan covers functional, usability, performance, and compatibility testing for the mobile app, backend services, and hardware components.

Features to be Tested

The following key features will be tested in the Posture Pal application:

1. **Posture Monitoring and Alerts**:
    - Test if the app detects poor posture based on a predefined threshold and triggers the appropriate alert (vibration, sound, or notification).
2. **Alert Customization**:
    - Verify if users can customize the type of alert (vibration or sound) and ensure that the system follows these settings.
3. **Posture History Dashboard**:
    - Test if posture data is accurately stored and displayed on the dashboard, including slouching times and correction notifications.
4. **Sensor Auto Disable**:
    - Ensure that the sensor is disabled automatically when the user remains inactive for a set period.
5. **Pairing with Device**:
    - Verify if the wearable device successfully pairs with the app and maintains a stable connection.
6. **Device Overheating Prevention:**
    - Test if the app detects when the device overheats and triggers a warning or takes action to prevent damage or discomfort to the user.

Testing Strategy

a. Functional Testing

- **Objective**: Validate that the app's core features work as expected.
- **Approach**:
    - Simulate different posture scenarios to test alerts.

- Check if users can switch between vibration and sound alerts seamlessly.
- Verify that posture history logs are correctly updated with each slouching event.
- Test the sensor's auto-disable functionality after inactivity.
- Test if the temperature sensor detects when the device overheats.

b. Usability Testing

- **Objective**: Ensure that the app's user interface is intuitive and easy to navigate especially for students.
- **Approach**:
  - Conduct tests with students to assess the ease of use such as pairing the device, customizing alerts, and viewing the posture history.
  - Gather user feedback on the app's layout, navigation, and overall user experience.

c. Performance Testing

- **Objective**: Test how the app performs under different conditions particularly during prolonged use.
- **Approach**:
  - Test real-time posture monitoring during extended study sessions.
  - Measure the app's responsiveness and stability when handling large datasets (e.g., posture history logs).

d. Compatibility Testing

- **Objective**: Ensure the app functions properly across different devices and environments.
- **Approach**:
  - Test the app on multiple Android devices with different OS versions to ensure compatibility.
  - Evaluate performance under various network conditions to assess the reliability of wireless communication.

Test Case Scenarios

Scenario 1: Posture Monitoring

- **Test Objective**: Verify that the app detects poor posture and triggers the appropriate alert.

- **Steps**:
    1. Simulate a posture angle exceeding the pre-defined threshold.
    2. Confirm that the selected alert (vibration, sound, or push notification) is triggered.
    3. Verify that the app records the slouching event in the posture history.

Scenario 2: Alert Customization

- **Test Objective**: Ensure users can switch between vibration and sound alerts.
- **Steps**:
    1. Access settings and switch from vibration to sound alerts.
    2. Simulate poor posture.
    3. Confirm that the sound alert is triggered.

Scenario 3: Posture History

- **Test Objective**: Validate that slouching data is logged and displayed in the dashboard.
- **Steps**:
    1. Simulate several slouching events.
    2. Open the posture history dashboard.
    3. Verify that all events are logged with the correct timestamps.

Scenario 4: Auto Sensor Disable

- **Test Objective**: Ensure that the sensor automatically disables after a period of inactivity.
- **Steps**:
    1. Leave the device idle for 10 minutes.
    2. Verify that the sensor disables itself after the set time.

Scenario 5: Device Pairing

- **Test Objective**: Verify that the app pairs correctly with the wearable device.
- **Steps**:
    1. Initiate pairing from the app.
    2. Confirm that the app establishes and maintains a stable connection.

Scenario 6: Device Overheating Prevention

- **Test Objective**: Verify that the app detects when the device overheats and triggers a warning or takes preventive action.
- **Steps**:
    1. Simulate conditions that cause the device to overheat (e.g., extended usage).
    2. Confirm that the app detects the overheating and displays a warning to the user.
    3. Verify that the app takes appropriate action, such as disabling certain features or alerting the user to take corrective steps.

Pass/Fail Criteria

- **Pass**: The test passes if all the expected results are met, and the app behaves as intended in the scenario.
- **Fail**: The test fails if the app does not produce the expected outcome or crashes during testing.

Tools and Resources

Hardware:

1. **BNO055 Orientation Sensor**:
    - Tracks user posture with gyroscope, accelerometer, and magnetometer.
    - Connects to **ESP8266** via I2C protocol.
2. **DHT22 Temperature and Humidity Sensor**:
    - Monitors device temperature to prevent overheating.
    - Connects to **ESP8266** via GPIO.
3. **ESP8266 Microcontroller**:
    - Collects sensor data and triggers alerts via actuators.
    - Central control unit for the system.
4. **Piezo Buzzer**:
    - Provides auditory alerts for poor posture.
    - Controlled by **ESP8266** through GPIO.
5. **Vibration Motor**:
    - Vibrates to alert the user about poor posture.
    - Triggered by **ESP8266** via GPIO.

6. **LiPo Battery**:
    - Powers the entire system for portable operation.

Front-End:

- **Mobile App (Android Studio, Kotlin):** Allows users to adjust settings, view posture history, customize alerts, and interact with the wearable device in real-time.

Back-End:

- **AWS**: Stores account and posture data.
- **PubNub**: Manages real-time data communication.

Test Devices:

- Multiple Android devices with various OS versions for compatibility testing.

Testing Tools:

- **Android Studio** for performance testing and debugging.
- **Breadboard** for hardware connections.
- **Multimeter** to check power levels from the LiPo battery.

Conclusion

This test plan outlines the steps and procedures that will be used to test the core features of the Posture Pal application. By conducting thorough functional, usability, performance, and compatibility testing the goal is to deliver a reliable, user-friendly posture monitoring app for students that meets their needs and ensures effective posture correction.