Critical Design Decisions

1. **Power Supply Strategy**:
   - Decision: Use an external power source with a voltage regulator to ensure stable power delivery for high-demand components (e.g., vibration motors).
   - Impact: Avoided power fluctuations that could damage components or destabilize the ESP32.
2. **Modular Code Architecture**:
   - Decision: Use **header files** and separate implementation files for each component (e.g., `FSR.h`, `Display.h`, `Keypad.h`, `VibrationMotor.h`) to reduce code complexity.
   - Impact: Made the project modular and maintainable, enabling independent testing of components and faster debugging.
3. **Sensor Calibration and Filtering**:
   - Decision: Calibrate the FSR sensor dynamically at runtime and apply software-based noise filtering.
   - Impact: Improved sensor accuracy and stability, ensuring reliable data readings in various conditions.
4. **Communication Protocol**:
   - Decision: Use I2C for the display and prioritize proper wiring and pull-up resistors to prevent communication errors.
   - Impact: Ensured reliable data transfer to the display without communication bottlenecks.
5. **PWM for Vibration Motor Control**:
   - Decision: Implement PWM (Pulse Width Modulation) to fine-tune vibration motor intensity and reduce noise.
   - Impact: Achieved precise control of vibration feedback, improving user experience.
6. **Keypad Input Handling**:
   - Decision: Use software debouncing combined with interrupt-driven input detection for the keypad.
   - Impact: Eliminated false keypresses and improved the responsiveness of the input system.
7. **Debugging Approach**:
   - Decision: Use **Serial Monitor** for debugging, `#define DEBUG` flags for conditional logging, and tools like **ESP Exception Decoder** to analyze runtime errors.
   - Impact: Reduced the time spent troubleshooting by gaining detailed insights into program execution.
8. **Translation Management**:
   - Decision: Store translation strings in JSON files (`translations/en.json` and `translations/no.json`) and fetch them dynamically based on the language setting.

    - Impact: Removed hardcoded language-specific HTML files, simplifying the codebase and improving scalability for additional languages.
9. **Version Control Strategy**:
    - Decision: Use Git for version control with proper branching, frequent commits, and detailed commit messages.
    - Impact: Streamlined team collaboration and minimized merge conflicts.