1. Technology selection

- **Programming language**: Java
    - **Why**: Java has strong library support and is suitable for handling complex logic and data structures.
- **Data Structure**:
    - Use ArrayList to store sensor data and calculation results.
    - Use HashMap to store the final statistical results.
- **Data Processing**:
    - Perform preliminary processing of the sensor data (e.g. dealing with negative angles).
    - Sort and statistically analyze the data.

2. Architecture design

- **Modular design**:
    - **Sensor**: encapsulates sensor data.
    - **Action classes** (e.g., Flexion, Extension, Abduction, etc.): Each class is responsible for calculating the amplitude of motion for an action.
    - **Analysis**: is responsible for data storage, calculation, and statistical analysis.
- **Dataflows**:
    1. Enter data (54 data points).
    2. Create a Sensor object.
    3. Invoke the methods of the action class to calculate the amplitude of motion for each action.
    4. Store the results in the corresponding ArrayList.
    5. Call the Statistical method to perform statistical analysis on the stored data and return the final result.

3. Quality Assurance

- **Test Strategy**:

    - **Unit Tests**: Write unit tests for each action class method to ensure that they are correct.
    - **Integration Testing**: Test the synergy of the Calculate and Statistical methods.
    - **System testing**: Testing the functionality of the entire system to ensure that the input data is processed correctly and returns the expected results.
- **Code Quality Management**:
    - Follow Java code specifications to ensure that the code is readable and maintainable.

Additional Notes:

1. **Data preprocessing**:
    - In the ExRotation and InRotation classes, negative angles are treated (if(a < 0) a += 360).
2. **Scalability**:
    - The current code is clearly structured to extend new motion types or sensor data processing logic.