Safety Requirements

- **Roll/Pitch Limit Reset:**
  When the robot's roll or pitch exceeds ±30°, **the system shall** immediately initiate a reset procedure to restore stability.
  Design Solutions:
    - Integrate high-accuracy IMUs and sensor fusion algorithms for continuous orientation monitoring at a 20Hz update rate.
    - Execute threshold-based interrupts within the control loop to trigger a reset sequence.
    - Log event data (e.g., sensor readings, timestamp, error state) for subsequent analysis.
- **Collision/Hazard Mitigation:**
  When a collision or hazard is detected by onboard sensors, **the system shall** execute an emergency stop and initiate a safe fallback maneuver.
  Design Solutions:
    - Use sensor arrays (e.g., LiDAR, stereo cameras, ultrasonic sensors) for real-time obstacle detection.
    - Design a low-latency interrupt (≤50 ms) to immediately halt joint actuators.
    - Develop fallback algorithms to reorient the robot and safely re-assess the environment.
- **Torque Management:**
  When motor torque measurements exceed predefined safe limits, **the system shall** actively reduce torque output to prevent damage and overheating.
  Design Solutions:
    - Continuously monitor motor current, torque, and temperature via embedded sensors.
    - Implement dynamic torque-limiting algorithms using real-time thermal and mechanical models.
    - Provide alerts for maintenance if over-torque conditions persist, and log all related events.
- **Sensor Fault Tolerance:**
  When sensor data (e.g., joint angles, velocities) is inconsistent or unavailable, **the system shall** switch to a fail-safe estimation mode.
  Design Solutions:
    - Deploy redundant sensors (e.g., dual IMUs) to ensure data integrity.
    - Utilize Kalman filters and alternative odometry methods to maintain state estimation during sensor faults.

    - Execute a controlled stop or system reset if error thresholds are surpassed.
- **Communication Link Failure:**
  When a critical communication channel (e.g., between the onboard controller and remote monitoring station) is lost, **the system shall** immediately transition to a predefined safe operational mode.
  Design Solutions:
    - Continuously monitor communication status and implement heartbeat signals.
    - Activate a safe mode that limits mobility and maintains balance.
    - Log communication loss events and notify system administrators.
- **Environmental Extremes Protection:**
  When ambient conditions (such as temperature, humidity, or dust levels) exceed predefined safe operating limits, **the system shall** initiate emergency protocols to mitigate risks to hardware integrity and operational safety.
  Design Solutions:
    - Deploy environmental sensors to monitor ambient conditions in real time.
    - Establish safe operating thresholds and trigger controlled shutdown or low-power modes.
    - Log environmental breaches and alert maintenance personnel for further inspection.
- **Battery/Power Management Safety:**
  When battery voltage or power supply parameters drop below safe operating thresholds, **the system shall** enter a low-power safe mode to prevent abrupt shutdowns or unsafe behavior.
  Design Solutions:
    - Integrate real-time battery monitoring to track voltage, current, and temperature.
    - Define power safety thresholds and trigger safe-mode routines when these are breached.
    - Log power anomalies and alert maintenance teams to preemptively address potential failures.
- **Velocity/Yaw Constraint Enforcement:**
  When the robot’s linear velocity deviates by >15% from the target or angular yaw velocity exceeds safe thresholds, the system shall adjust gait parameters to restore compliance.
  Design Solutions:
    - Monitor velocity/yaw via Kalman-filtered estimates and IMU data.
    - Penalize deviations in the RL reward function to encourage compliance.

    - Trigger gait recalibration if thresholds are breached.
- **Learning Stability Monitoring:**
  When the RL policy's performance degrades (e.g., increased falls or torque violations), the system shall revert to a stable baseline policy.
  Design Solutions:
    - Track policy performance metrics (e.g., fall rate, reward trends).
    - Maintain a fallback policy (e.g., pre-trained gait) for emergencies.
    - Validate new policies in a simulated sandbox before deployment.

Maintainability Requirements

- **Performance Anomaly Logging:**
  When performance metrics (e.g., deviations in joint angles or abnormal sensor noise) exceed defined thresholds, **the system shall** log the incident and alert maintenance teams.
  Design Solutions:
    - Integrate a high-resolution logging system with timestamped records of sensor data and control actions.
    - Set up threshold-based alerts within the software for real-time anomaly detection.
    - Provide remote diagnostic dashboards for system monitoring and fault analysis.
- **Seamless Software Updates:**
  When a validated software update is available, **the system shall** apply the update during scheduled idle periods without interrupting active operations.
  Design Solutions:
    - Use A/B partitioning or containerized environments to support seamless firmware updates.
    - Verify updates using cryptographic checksums and digital signatures.
    - Automatically roll back to the previous stable version in case of boot or performance failures.
- **Detailed Error Diagnostics:**
  When an operational error occurs, **the system shall** record detailed diagnostic data, including state variables, actions taken, and environmental context.
  Design Solutions:
    - Implement a circular logging buffer that captures and stores diagnostic information with high temporal resolution.

    - Include metadata such as terrain type, joint status, and stability indicators.
    - Enable secure remote access for log retrieval and post-mortem analysis.
- **Modular Hardware Replacement:**
  When a hardware component (e.g., a joint motor) requires replacement, **the system shall** support plug-and-play modularity without the need for full system reconfiguration.
  Design Solutions:
    - Standardize connectors and interfaces to ensure compatibility across components.
    - Abstract hardware dependencies within the control software to enable hot-swapping.
    - Automate calibration routines for newly installed components to ensure accurate operation.
- **Training Data Integrity:**
  When new training data is collected, the system shall validate and curate it to prevent corruption.
  Design Solutions:
    - Flag anomalous data (e.g., sensor spikes, implausible joint angles).
    - Use version control for datasets and model checkpoints.
    - Automatically purge low-quality data samples.
- **Reset Mechanism Health Checks:**
  When the reset policy is triggered, the system shall verify its success and log diagnostics.
  Design Solutions:
    - Confirm post-reset stability via IMU and joint feedback.
    - Log reset success/failure rates and root causes (e.g., terrain type).

Privacy & Security Requirements

- **Secure Data Transmission:**
  When operational data is transmitted externally, **the system shall** encrypt the data using AES-256 for storage and TLS 1.3 for communication.
  Design Solutions:
    - Encrypt logs and telemetry both at rest and during transmission.
    - Limit data transmission to essential diagnostics and performance metrics.

  - Regularly update cryptographic protocols to mitigate emerging vulnerabilities.
- **Unauthorized Access Prevention:**
  When unauthorized access is detected, **the system shall** immediately block the connection and alert system administrators.
  Design Solutions:
  - Deploy network firewalls and intrusion detection systems (IDS) to monitor and block suspicious activity.
  - Use hardware-based authentication for command execution.
  - Maintain detailed access logs (with IP/MAC addresses and timestamps) for forensic analysis.
- **Remote Command Authentication:**
  When a remote reset or control command is issued, **the system shall** authenticate the source via digital signatures and multi-factor authentication (MFA).
  Design Solutions:
  - Require MFA for any remote commands affecting system safety.
  - Validate command integrity using public-key cryptography.
  - Log all remote command events with associated user/device identifiers.
- **Firmware Update Verification:**
  When firmware or software updates are initiated, **the system shall** verify the authenticity and integrity of the update package before installation.
  Design Solutions:
  - Implement a secure boot process that loads only cryptographically signed firmware/software.
  - Use Trusted Platform Modules (TPMs) to securely store keys and perform digital signature checks.
  - Enforce an update verification process and periodically audit the update mechanism.
- **Physical Tamper Detection:**
  When unauthorized physical access is detected, the system shall disable actuators and erase sensitive data.
  Design Solutions:
  - Use tamper-evident seals and accelerometers to detect physical intrusion.
  - Encrypt onboard storage and perform secure wipe on tamper detection.
- **Estimation Fail-Safe:**
  When Kalman filter errors exceed thresholds (e.g., velocity estimate

divergence), the system shall switch to conservative motion modes.
Design Solutions:

- Cross-validate estimates with leg odometry and contact sensors.
- Reduce speed and trigger a reset if estimation fails.