**Non-Functional Requirements**

* Performance:

1. Critical screens, such as team status or reports, must load within 2 seconds under a load of up to 20 concurrent users.
2. The system must handle up to 50 simultaneous requests without performance degradation
3. The system must ensure no data loss during an unplanned shutdown and recover to full functionality within 5 minutes.

* User Interface (UI/UX):

1. The interface must be responsive and compatible with modern browsers (Chrome, Firefox, Edge).
2. All key actions must be achievable within a maximum of 5 clicks from the main screen.
3. Dashboards and reports should feature intuitive graphical elements, including charts, metrics, and easy-to-use filters.

* Data Security:

1. User authentication must be implemented using Single Sign-On (SSO) or a secure token mechanism.
2. Sensitive data, such as employee names and working hours, must be stored in a secure Database.
3. Access to administrative features (e.g., adding new employees) must be restricted based on user roles and permissions.

* Accessibility:

1. Features such as text enlargement and high-contrast mode must be available.

* Maintenance and Upgrades:

1. The codebase must be well-documented to allow new developers to onboard quickly.

* Scalability:

1. The system must scale linearly and support teams of up to 1,000 employees without performance loss.
2. Adding new projects, teams, and requirements should not require significant changes to the system architecture.

* Availability:

1. The system must operate with 99.9% uptime throughout the year.
2. Planned maintenance must be communicated to users at least 48 hours in advance.

**Risk Assessment**

**1. Technical Risks**

* **Risk**: Performance degradation with increasing number of users.
  + **Mitigation**: Implement stress testing during development to identify bottlenecks early. Optimize database queries and caching mechanisms.
* **Risk**: Incompatibility with modern browsers or mobile devices.
  + **Mitigation**: Conduct cross-browser and device compatibility testing during development.
* **Risk**: Security vulnerabilities in sensitive data handling (e.g., encryption, authentication).
  + **Mitigation**: Follow best practices for secure coding, including regular security audits and penetration testing.

**2. Operational Risks**

* **Risk**: Resistance from users due to unfamiliarity with the system.
  + **Mitigation**: Provide training materials and user-friendly documentation. Conduct workshops for managers and employees.
* **Risk**: Errors in data input (e.g., incorrect employee or project details).
  + **Mitigation**: Implement input validation and error-handling mechanisms in the system.

**3. Project Risks**

* **Risk**: Delays in meeting deadlines due to scope creep or underestimated complexity.
  + **Mitigation**: Define clear deliverables for each phase, and use Agile methodology for incremental progress and continuous feedback.
* **Risk**: Dependency on third-party libraries or tools causing integration issues.
  + **Mitigation**: Choose reliable, well-documented libraries and maintain backups or alternatives.

**4. Legal and Compliance Risks**

* **Risk**: Non-compliance with data protection laws if handling employee data.
  + **Mitigation**: Ensure compliance by consulting legal experts and implementing strict data protection measures.

**Proof of Concept (PoC) Plan**

**Objective**

Validate the feasibility of the proposed system by demonstrating its ability to:

* Manage employee and project data.
* Display team status and project utilization visually.
* Handle issue creation and resolution workflows effectively.

**Scope**

The PoC will focus on:

1. Basic employee and project management functionality.
2. Visualization of team and project utilization metrics (e.g., dashboards).
3. Creating and resolving issues within the system.

**Timeline**

* **Week 1**: Define PoC objectives, gather requirements, and set up the development environment.
* **Week 2-3**: Develop the core modules:
  + Employee and project management interface.
  + Dashboard with utilization metrics (static data for now).
* **Week 4**: Implement the issue creation and resolution workflow.
* **Week 5**: Test the PoC with a small group of users (e.g., one team manager and a few employees).
* **Week 6**: Collect feedback and refine the PoC.

**Success Criteria**

* The system should handle basic employee and project data entry without errors.
* Dashboards should display utilization data in an intuitive manner.
* The issue creation and resolution workflow should be functional and easy to use.
* Feedback from test users indicates satisfaction with ease of use and functionality.

**Deliverables**

* A functional PoC system accessible to test users.
* Documentation outlining key functionalities and known limitations of the PoC.
* A report summarizing feedback and recommendations for the full-scale implementation.