HUrricane

**Risk Management Report**

# **Introduction**

[Briefly describe the content of the Risk Management Report.]

This Risk Management Report provides a comprehensive list of known and open risks associated with the Food Application System project, prioritized in order of importance and linked to specific mitigation or contingency actions. This report serves as a main point for project activities and guides the organization of iterations.

# **Description**

[This artifact is a list of known and open risks to the project, sorted in order of importance and associated with specific mitigation or contingency actions. This list identifies, in decreasing order of priority, all the risks associated to a project. It serves as a focal point for project activities and is the basis around which iterations are organized.]

This section of the document outlines the known and potential risks to the project, ranked by priority. Each risk is linked to specific mitigation or contingency measures. A more comprehensive breakdown of these potential risks can be found in the Risk List document too. It’s important to keep in mind that this list may change over the course of the project, as unexpected issues may arise that were not initially anticipated.

| Potential risk | Mitigation Strategy or Contingency Actions |
| --- | --- |
| Functionality Bugs (Errors in core features.) | Test features incrementally; prioritize fixing critical bugs. |
| Inadequate Unit Testing Coverage (Many functions are untested, increasing the risk of undetected bugs in production.) | Define testing criteria; enforce minimum coverage through CI pipelines. |
| Time Management Issues (Underestimating time for coding, testing, or documentation.) | Create a timeline with milestones; allocate buffer time for delays. |
| Inconsistent API Responses (API endpoints return data in varying formats, causing integration issues with the frontend.) | Standardize API responses; implement response schema validation. |
| Version Control Conflicts(Merge conflicts frequently occur due to lack of branch discipline and unclear commit messages.) | Define branching strategy; implement commit message guidelines and conflict resolution sessions. |
| Database Issues (Frequent connection timeouts and slow query performance due to suboptimal indexing and inefficient query design.) | Review indexing strategy; optimize queries and adjust connection pooling parameters. |
| Submission Risks (Issues during submission (e.g., file corruption, late upload).) | Test submission process early; submit ahead of deadline. |
| Demo Standard Accessibility Issue(The interface is unable to access the defined demo standards; this issue leads to disruptions in demo presentations and user experience.) | Review and update the configuration to comply with demo standards; integrate the relevant guideline documents. |
| Team Coordination Problems (Miscommunication or uneven workload among team members.) | Assign roles clearly; use tools like GitLab and hold regular check-ins. |
| Device Compatibility (App not working across browsers or devices.) | Test on multiple platforms; use responsive design principles. |
| Scope Creep (Adding features beyond assignment requirements (e.g., payment options).) | Define MVP based on assignment brief; avoid extra features unless time permits. |
| Incomplete Use Cases for Demo Release (Critical use cases planned for the demo release are missing, leading to uncertainty in the application's workflow.) | Perform a comprehensive use case analysis; review requirements and update the demo documentation accordingly. |
| Feature Deployment Delays (Features take longer than estimated due to unexpected technical complications and unclear task breakdowns.) | Improve sprint planning accuracy; involve technical staff in estimation and risk assessment. |
| Ambiguous Requirement Specifications (Lack of clear specifications leads to differing interpretations among team members.) | Conduct requirement clarification meetings; maintain a shared, updated specification document. |
| Authentication Token Expiry Issues (UI components were changed without updating wireframes or informing the dev team, causing implementation mismatches.) | Establish a UI change log; notify developers about updates and maintain wireframe version control. |
| User Interface Issues (Confusing or unintuitive UI design (e.g., unclear buttons).) | Sketch wireframe; get early peer feedback and simplify design. |
| Untracked Design Changes(UI components were changed without updating wireframes or informing the dev team, causing implementation mismatches.) | Establish a UI change log; notify developers about updates and maintain wireframe version control. |
| Data Management Errors (Poor database design or lack of validation for mock data (e.g., category, orders).) | Design simple schema early; test with sample data and add validation. |

# **Risk Management Report Specifications**

[Key Considerations

This list should capture the critical and serious risks. If you find this list extending beyond 20, carefully consider whether they are really serious risks. Tracking more than 20 risks is an onerous task.

Tailoring

Representation Options

Option: list of risks captured in the project plan

In this approach you put the overall risk list in the project plan. The iteration plan will contain only the tasks you will be doing during the iteration to mitigate the risks. This will ensure that the iteration plan contains only iteration information. The project plan has to be revisited constantly as you update risks.]

Risk Management Report Specifications is a summary of the 7 major risks encountered during the project's development, along with the strategies we applied to address them. This list has been continuously updated over the course of the project.

| Risk ID | Risk Title | Risk Description | Strategies of Mitigation | Seriousness |
| --- | --- | --- | --- | --- |
| R1 | Time Management Issues | Underestimating time for coding, testing, or documentation. | 1-Creating a timeline with milestones with proximate time intervals for every goal..  2-Allocate buffer time for possible delays. | Medium |
| R2 | Functionality Bugs | Errors in core features. | 1-Test features incrementally, starting to prioritization from critical risks.  2-Repeating the tests periodically for every implementation milestone. | High |
| R3 | Inadequate Unit Testing Coverage | Many functions are untested, increasing the risk of undetected bugs in production. | 1-Define testing criteria; enforce minimum coverage through CI pipelines.  2-Specialize the criteria per test case. | Medium |
| R4 | Nonsynchronous implementation | Inconsistent code, misaligned features, delays between deliveries, and difficulty in testing. | 1-Follow the roles assigned before implementation.  2-Track the implementations of all developers.  3-Ensure adequate communication between group members. | High |
| R5 | Shortage on Third Party Services | APIs or third-party services could be unavailable during implementation or testing. | 1-Reduce dependency on third-party services by using local versions where possible.  2-Have a backup plan for services dependent on third-party applications.  3-Identify convenient alternative applications in advance. | Medium |
| R6 | Security Vulnerabilities | Lack of proper authorization and confidentiality could lead to critical information leaks. | 1-Implement proper authorization controls using suitable applications. 2-Follow relevant security regulations when working with sensitive or personal data. | High |
| R7 | Lack of Requirement Clarity | This can lead to misaligned development, wasted resources, and project delays. | 1-Ensure all requirements are documented and validated with stakeholders. 2-Continuously review and adjust requirements throughout the project. 3-Use prototypes to confirm requirements before final implementation. | Medium |
| R8 | Integration Issues | Integration with legacy or existing systems may introduce compatibility problems, data migration challenges, or system downtime. | 1-Conduct compatibility assessments in earlier implementation phases.  2-Keep the plan up-to-date for thorough integration testing and data migration.  3-Establish fallback procedures in case any failure.. | Medium |

# **Traceability Table**

|  | Description | Risk Management Specification | Summation |
| --- | --- | --- | --- |
| Yusuf Küçüköner | 2h | 2h | 4 |
| Salih Eren Yüzbaşıoğlu | 1h | 3h | 4 |
| Şükriye Öztürk | 2h | 3h | 5 |
| Mustafa Furkan Ateş | 1h | 3h | 4 |
| Bedirhan Gençaslan | 0h | 5h | 5 |

# **Prompts**

ChatGPT: <https://chatgpt.com/share/67faee84-3328-800f-a43d-782ec576ff4e>