SOEN 390: Software Engineering Team Design Project
Winter 2021

Risk Management Plan

Submitted to Dr. Yann Gueheneuc

Team 9

Concordia University

April 7, 2021

TABLE OF CONTENTS

RISK CATEGORIES

The risks shall be divided in the following categories:

Risk categories	
Schedule Risk	**Risks that are related to the schedule. These include and not limited to:** **- Incorrect time estimation** **- Improper resource allocation** **- Change in project scope** **- Unidentified complex functionalities and time to develop them** **- Volatile requirements**
Budget Risk	**Risks that are related to the budget. These include and not limited to:** **- Incorrect budget estimation** **- Improper tracking of budget** **- Wrong budget allocation** **- Cost overrun**
Operational Risk	**Risks that are related to the operations. These include and not limited to:** **- Insufficient resources** **- Lack of communication in the team** **- Wrong responsibility identification** **- No resource planning**
Technical Risk	**Risks that are related to the technical aspect. These include and not limited to:** **- Volatile requirements** **- Lack of technology availability for solution machine** **- Lack of skilled workers** **- Difficult project modules integration**
Programmatic Risk	**Risks that are related to the change of the plan. These include and not limited to:** **- Rapid market changes** **- Running out of funds** **- Change in government regulations**

	- **Loss of contract**
Requirements Risk	**Risks that are related to the requirements. These include and not limited to:** - **Unidentified requirements** - **Late identified requirements** - **Inadequate requirements** - **Requirements that are not needed by stakeholders**
Design Risk	**Risks that are related to the design. These include and not limited to:** - **Improper architecture** - **Lack of OO features used in design**
Testing Risk	**Risks that are related to the testing phase. These include and not limited to:** - **Lack of representation of real-world scenarios with automated testing** - **False Positives** - **False Negatives**
Implementation Risk	**Risks that are related to the implementation phase. These include and not limited to:** - **Redesigned system** - **Change in technologies** - **New requirements added by stakeholders**

Risk Identification

Table 1: Top 10 Risks

Risk	Category	Description
Lack of time	Schedule	Not enough to complete tasks causing delays on the project completion
Lack of communication (team cohesion)	Operational	Team members do not report to each other and are not made aware of the progression of the project
Injection (hacking)	Technical	Hackers can attack the system by inserting malicious software such as SQL injections, DDoS attack, JS injections
Late identified requirements	Requirements	New requirements are introduced once the system is already in progress
Change in project scope	Schedule, Requirements	The project manager or the stakeholders change the requirements of the project, and therefore, the scope changes.
Improper architecture	Design	The choice of software architecture is not suitable to the system being developed.
Lack and improper testing	Testing	Testing is not performed frequently and does not cover a large part of the codebase.
Lack of reusable modules for software tasks	Implementation	The tech stack chosen does not offer many frameworks or libraries which requires developers to extra time building components.
Lack of object-oriented features	Design, Implementation	The software implementation of the system does not follow proper object-oriented features and design patterns which increase future maintainability effort.
Lack of resource planning	Operational	During the development of the project, the team realizes that they need a bigger budget, or more coders because they are not advancing in the project.

Table 2: Additional Risks

Risk	**Category**	**Description**
Lack of understanding of requirements	Requirements, Implementation	Features are implemented, but do not meet the intended use case of the requirement.
Wrong time estimation of tasks	Schedule	Tasks are not assigned the correct amount of user story points, causing schedule overruns.
Lack of prioritization	Schedule, Operational	The most important and critical tasks are not started first.
Lack of documentation	Technical	Product does not contain the required documentation to make it easy for stakeholders to utilize its features.
Lack of code quality standards	Implementation, Technical	Code does not follow coding conventions and standards which makes it difficult for developers to understand the codebase and add new features.
Late start time of tasks	Schedule	If tasks are not started on time, they may not be completed by the end of the sprint. This leads to schedule overruns.
Wrong number of order in manufacturing process	Technical	The system displays the user the wrong order. By selecting the order number, the system mixed up another order and displayed wrong manufacturing products
Product quantity and type wrongly displayed	Operational	An item may be duplicated in the table and it would lead to the wrong quantity and mix-up in the type of product
Excessive deployment time	Operational	The system takes time to deploy, preventing users from completing their transactions promptly.
Major refactoring causing task delay	Implementation	Improper following of coding standards can lead to a lot of refactoring which may hinder development time of Sprint tasks.

Feature doesn’t meet product owner’s expectations	Requirements	Features lack certain functionalities requested by the product owner.
Lack of time to finish implement certain features.	Schedule	Extra time is required for the team to deliver the features.
Major bug causing schedule overrun to fix it	Schedule	Major bug which must be fixed in order to advance in the development causes a schedule overrun.
Poor use of coding standards leading to a lot of refactoring	Technical	Coding standards are not used consistently across the codebase causing major refactoring.

Risk analysis (Relate to impact matrix)

Table 3: Risk Analysis of Top 10 Risks

Risk	Impact	Probability	Risk exposure	Severity
Lack of time	0.5	0.25	0.125	Low
Lack of communication (team cohesion)	0.5	0.25	0.125	Low
Injection (hacking)	0.75	0.5	0.375	High
Late identified requirements	0.5	0.75	0.375	High
Change in project scope	0.25	0.75	0.1875	Medium
Improper architecture	0.75	0.25	0.1875	Medium
Lack and improper testing	0.75	0.75	0.5625	High
Lack of reusable modules for software tasks	0.25	0.5	0.125	Low
Lack of object-oriented features	0.25	0.5	0.125	Low
Lack of resource planning	0.5	0.25	0.125	Low

Table 4: Risk Analysis of Additional Risks

Risk	Impact	Probability	Risk exposure	Severity
Lack of understanding of requirements	0.5	0.75	0.375	High
Wrong time estimation of	0.5	0.25	0.125	Low

tasks				
Lack of prioritization	0.5	0.25	0.125	Low
Lack of documentation	0.25	0.75	0.1875	Medium
Lack of code quality standards	0.75	0.5	0.375	High
Late start time of tasks	0.5	0.5	0.25	Medium
Wrong number of order in manufacturing process	0.75	0.5	0.375	High
Product quantity and type wrongly displayed	0.5	0.75	0.375	High
Excessive deployment time	0.25	0.75	0.1875	Medium
Major refactoring causing task delay	0.75	0.25	0.1875	Medium
Feature doesn’t meet product owner’s expectations	0.75	0.25	0.1875	Medium
Lack of time to finish implement certain features.	0.75	0.5	0.375	High
Major bug causing schedule overrun to fix it	0.5	0.25	0.125	Low
Poor use of coding standards leading to a lot of refactoring	0.75	0.25	0.1875	Medium

Risk Response (what will be done to avoid the risk)

Table 5: Risk Response for Top 10 Risks

Risk	**Solution**
Lack of time	Plan sprints before the beginning of the respective sprint. Also, start working on tasks as soon as the new sprint starts.
Lack of communication (team cohesion)	Mitigate by doing frequent meetings and making sure everyone is progressing
Injection (hacking)	Mitigate by preparing the website for SQL injections in the database
Late identified requirements	Accept, and reschedule the sprint to account for the new requirements
Change in project scope	Mitigate by taking time to elicitate the requirements and use the appropriate tools to interview the stakeholders, and making sure every possible information was extracted
Improper architecture	Avoid by consulting the software architect before starting and making sure that the proposed architecture is suitable for the ERP system.
Lack and improper testing	Avoid by doing unit tests for every method, and implement different code coverage techniques (mutation, data flow analysis, statement coverage)
Lack of reusable modules for software tasks	Avoid by selecting a tech stack which offers frameworks and libraries which can easily be reused, reducing development time.
Lack of object-oriented features	Avoid by using design patterns and using SOLID class design principles.
Lack of resource planning	Mitigate by doing frequent meetings to make sure the project has enough developers, designers, money and other useful resources

Table 6: Risk Response for additional risks

Risk	**Solution**
Lack of understanding of requirements	Mitigate by frequently discussing requirements with team members and the product owner.
Wrong time estimation of tasks	Mitigate by obtaining the opinion of all teams for user story points estimation.
Lack of prioritization	Avoid by assigning a product manager who will plan the most important and critical tasks based on the product owner's requirements.
Lack of documentation	Avoid by providing description for the code, tests, features and how to utilize the system.
Lack of code quality standards	Mitigate by following design patterns, code conventions and performing rigorous code reviews.
Late start time of tasks	Mitigate by having frequent team meetings and assessing the progress of all team members on their various tasks.
Wrong number of order in manufacturing process	Mitigate by performing a backend code review, because this problem may be due to a wrong api call.
Product quantity and type wrongly displayed	Mitigate by adjusting the frontend code and making sure users can't add two products on the same table.
Excessive deployment time	Avoid by using a server which can handle decent sized loads of users.
Major refactoring causing task delay	Mitigate by performing in depth code reviews and following coding standards and design patterns.
Feature doesn't meet product owner's expectations	Mitigate by communicating regularly with product owner and to obtain feedback and ensure requirements are being met.
Lack of time to finish implement certain features.	Avoid by doing frequent meetings with team members to ensure proper progression and stay up to date with work to be completed.
Major bug causing schedule overrun to fix it	Mitigate by performing in depth software testing to lower the risk of major bugs.
Poor use of coding standards leading to a	Mitigate by performing rigorous code

lot of refactoring	reviews in order to suggest code improvements such as design patterns.