Risk Register

Sustainable Work through Women-in-tech Application for Older Women in Malaysia and Thailand: Integrating Action Research and Design Science Approach

Contents

Introduction

Our project requires us to develop an application that helps support the eldery by recommending content from the internet that is beneficial to users. Such content may include educational videos to brush up the skillsets of the eldery that are falling behind in the technological era. This document aims to outline the risks that will be faced in the process of development and the strategy to deal with each risk listed.

The risk management process typically involves four major phases. Starting off with **risk identification**, where the team identifies risks related to our project. This is followed by **risk analysis**, where we prioritise resources for each risk management based on their likelihood of occurrence and potential impact. Next, a **risk response planning** is done, in order to come up with potential options and action plans to mitigate said risks. Finally, **Risk monitoring and control** is carried out, where the identification, analysis and planning of newly introduced risks is done. Previously identified risks are also reevaluated to verify the proposed risk response strategies and effectiveness.

This document will be split into a few parts, namely:

1. Organizational Risks: Elaboration of risks that exist because of the staff body within or outside the team
2. Customer Representative Risks: Elaboration of risks that exist because of the clients.
3. Hardware Environment Risks: Elaboration of risks that exist due to the hardware (computers, servers, internet connection, communications equipment) that the team utilizes to work on the application

4. Software Environment Risks: Elaboration of risks that exist because of operating system, IDE, libraries, third-party code, the source code of our application, and other systems the team relies on

The scales used in this document are defined in the following tables:

Table 1: Impact Scale

Scale Point	**Time**
1	Poses a small delay in terms of deadlines of a feature (1 day past the initial deadline for one feature or multiple features adding up to this amount).
2	Poses a minor delay in terms of deadlines of a feature (2 - 3 days past the initial deadline for one feature or multiple features adding up to this amount).
3	Poses a moderate delay in terms of deadlines of a feature (4 - 5 days past the initial deadline for one feature or multiple features adding up to this amount)
4	Poses a large delay in terms of deadlines of the project (6 - 7 days past the initial deadline for one feature or multiple features adding up to this amount).
5	Poses a large delay in terms of deadlines of the project and creates uncertainty of being completed (uncertain deadline extension).

Table 2: Likelihood Scale

Scale Point	**Probability**	**Description**
1	0-10%	Not likely to occur
2	11-40%	Likely to occur

3	41-60%	Has a good chance of occuring
4	61-90%	Very Likely to occur
5	91 - 100%	Almost certain to occur

Table 3: Risk Matrix

Impact	5	5	10	15	20	25
	4	4	8	12	16	20
	3	3	6	9	12	15
	2	2	4	6	8	10
	1	1	2	3	4	5
		1	2	3	4	5
		Likelihood				

Table 4: Risk classification

Scale	**Classification**
1 - 3	Low risk
4 - 7	Moderate risk
8 - 12	High risk
> 12	Extremely high risk

Organizational Risks

1. Low Work Ethic From Team Members

- Description: When someone procrastinates too much or there is something preventing that team member from doing work, impeding progress, there is a risk of deliverables not being delivered on time.
- Impact: 3
    - Team members may procrastinate and not do their work which has the potential of causing major delays to development.
- Likelihood: 3
    - With multiple team members in the project, it is likely that some members are likely to be preoccupied with other units and projects, causing their lack of work ethic in this project.
- Risk: 9
- Strategies for monitoring
    - Frequently view and review GitLab commits
    - Team members write and read daily scrum updates to track other member's progress
- Strategies for mitigation
    - Conduct more proactive group meetings where each member has to do their work during the call while sharing their screen.This will ensure at least some higher level of engagement from the team members.

2. Weak non- functional requirements

- Description: Although before assigning tasks, in order to keep it fair, estimates of each user story is conducted. The estimates are not always accurate since by working on the user story the member in charge may experience issues that have not been accounted for in the estimate. Some of these issues are very hard to predict before attempting the user story. This may lead to prioritization on the core functionality of the story and might ignore the non functional requirements aspect.
- Impact: 3
    - Prioritization of functional requirements over non functional requirements may result in a non user-friendly application.
- Likelihood: 3
    - It can be very hard to estimate user stories beforehand as some potential difficulties may not be expected.
- Risk: 9
- Strategies for monitoring:
    - Conduct frequent quality assurance tests with a non technical person and provide his/her feedback to other members during a meeting. Ensure non-functional requirements such as user interfaces are up to par.
- Strategies for mitigation:
    - Improve application based on the feedback received. Unsatisfactory non-functional requirements need to be brushed up and fixed.

3. Dependency of features between different development teams

- Description: Features between different parts of the application depend on each other; As such, if not coordinated well enough, teams may choose to develop features that are not capable of being fully functional until the other team has completed their implementation. There is a risk of teams do not complete their designated features on time, impeding progress of other teams

- Impact: 5
    - Impedes development severely; Teams may not be able to complete their designated tasks on time
- Likelihood: 3
    - Very likely to occur when teams do not communicate well with each other
- Risk: 15
- Strategies for monitoring:

    - Conduct frequent checks that the user stories that are chosen to be implemented in the current program increment is preferably independent
- Strategies for mitigation:
    - If user stories are dependent on other features, make sure teams pick these cohesive features on the same program increment.

Customer Representative Risks

4. Unrealistic Requirements

- Description: The customer representative provides an unrealistic requirement when asked for, making it very hard or impossible to deliver.
- Impact: 3
    - Since the team does not have a realistic scope of deliverables, it is very likely for the team to underdeliver on different parts of the project.
- Likelihood: 4
    - It is very likely for the client to specify unrealistic requirements due to the difficulty in estimating the amount of work required.
- Risk: 12
- Strategies for monitoring
    - Conduct review of every feature before including it in the project backlog.
    - Constantly clarify any and all requirements that seem unrealistic and their purpose in the software.
- Strategies for mitigation
    - Product owner needs to be somewhat experienced in requirement gathering to communicate with the client
    - Come up with a different feature(s) that satisfies the need based on the user story if the previous user story is too large.

5. Contradictory Requirements given by client

- Description: Different requirements given by the client are contradicting, making it impossible to decide on what to deliver.
- Impact: 2
    - Contradicting requirements will halt progress in the project as it will not allow the team to proceed with the implementation of features for which the requirements contradict, requiring us to consult with the client.
- Likelihood: 4
    - The client may not be familiar with the technicality of the software, and as a result, provide requirements that contradict one another.
- Risk: 8
- Strategies for monitoring:
    - Group members need to be clear of the client's requirements, noting when they are contradictions between requirements
- Strategies for mitigation:

    - Refine requirements with clients
    - Review the user stories to come up with the need of the features contradicting each other to find a way to implement their needs without implementing those specific features.

6. Potential Change of Requirements

- Description: The requirements change halfway through the project, causing a lot of the progress to be abandoned, having to rework on a large portion of the project and possibly risking not being able to finish the project on time.
- Impact: 3
    - A change in requirements may require the team to begin working on new features and possibly having to integrate new features with features already implemented.
- Likelihood: 3
    - The client is often unaware of their full needs and requirements, and there is a big possibility of them changing existing requirements and/or adding new requirements to the project.
- Risk: 15
- Strategies for monitoring
    - Keep minutes of every session with the representative to keep track of their requirements
- Strategies for mitigation
    - Ensure the customer representative is sure before proceeding with the development by inquiring about the purpose and need of the feature before beginning its implementation.

7. Vague requirements specified by clients

- Description: Client's requirements are vague, may cause different interpretations between clients and the team, ultimately causing a product that will not be accepted by the client in terms of specification.
- Impact: 2
    - A vague requirement may require a group discussion as well as another day for further clarification from clients.
- Likelihood: 5
    - The client usually does not have a fixed idea on what they want, and will usually give vague answers unless they are asked in detail
- Risk: 10
- Strategies for monitoring:
    - Keeping track of the requirements specified by client

    - Have constant meetings with the client / product owner.
- Strategies for mitigation:
    - Present newly-implemented features to the client to ensure that that is the feature they had in mind.

Hardware Environment Risks

8. Backend Database Server Crashes Unexpectedly

- Description: Backend database server crashes, user data is corrupted or lost.
- Impact: 5
    - A corrupted database would be devastating; Users would need to complete their questionnaires all over again for the system to give out tailored recommendations.
- Likelihood: 2
    - If hosted in a safe, secure and stable server, it is unlikely that the server would crash unexpectedly.
- Risk: 3
- Strategies for monitoring:
    - Ensure that server load is optimal at any given time
- Strategies for mitigation:
    - Block out users from using the app when server memory is limited
    - Backup database into multiple servers; Ensure that other servers can be used as a substitute when the main server is unavailable.

9. Server for git crashes, files get corrupted

- Description: Gitlab server crashes, get our files corrupted and unable to use them causing the team to lose a large portion of the progress done on the project risking the project not being delivered on the set deadline.
- Impact: 5
    - If gitlab server's crashes and loses all of our work done, project data may be lost and it may cause us to restart the entire project from the beginning.
- Likelihood: 1
    - It is very unlikely to happen as the server is maintained by well equipped IT professionals and most people have a local repository of the last pulled version
- Risk: 5
- Strategies for monitoring
    - Consistently check if any new changes were made to the gitlab file that needs to be backed up
    - Frequently pull the most up to date files from github to be stored in the local repository as a backup.
- Strategies for mitigation
    - Push the most recently pulled local backup back to the git server, to continue the project.

Software Environment Risks

10. Team members are not familiar with the used programing languages / frameworks / third party software components

- Description: Team has not much experience with the used programing language,frameworks or third party software component, causing a member to deliver slow and ultimately costing the team's efficiency
- Impact: 3
    - It may take a week to get used to a new language or third party software component.
- Likelihood: 1
    - Since the languages/frameworks used are decided beforehand according to team members' overall experience, it is very unlikely for team members to have difficulties with the language/framework in use.
- Risk: 3
    - A member that is not familiar with the components used would slow down the development process
- Strategies for monitoring
    - Check if new team members are experienced with the third party software components
- Strategies for mitigation
        - Do spikes
    - Use forums such as StackOverflow to seek help.
    - Work in pairs to allow team members to communicate with one another in case one requires help from the other.
    - Read and understand the software documentation for further understanding

11. Third party service goes down

- Description: Third party service used in the application such APIs (YouTube, Google Text To Speech, Google Translate) goes offline.
- Likelihood: 1

    - Both API services are developed by Google. Google is known to be reliable in a way that their services are rarely down, and operated at a good functional level.
    - Since these services are real businesses, it is unlikely to go down as it will cost them to have a long down time.
- Risk: 1
    - A downtime on YouTube would prevent the app from downloading non-copyrighted videos; The app is unable to recommend selected videos to the users.
- Strategies for monitoring
    - None
- Strategies for mitigation
    - Download a predefined queue of videos that would be recommended to the user in the future. The videos are still being able to be recommended even if YouTube is down.

12. Incompatible User Interface for some devices or browser

- Description: Since we are building a web-based application, the styling of the user interface may go out of bounds when using certain devices or browsers due to incompatibility.
- Impact: 4
    - May ruin user experience
- Likelihood: 3
    - Since there are countless platforms out in the market, it is likely that some would have this problem
- Risk: 12
- Strategies for monitoring
    - Check if application is compatible with popular web browsers such as chrome, safari
- Strategies for mitigation
    - Fix styling issues once found out.

13. User Privacy Leak through security vulnerability

- Description: User information such as name and phone number being obtained illegally/immorally by malicious individuals/syndicates
- Impact: 5

    - Malicious activities such as telephone scams or user impersonation can be done by using the user's private information.
- Likelihood: 2
    - Since the application does not record highly sensitive information such as user's bank account details and addresses, it is unlikely that individuals would have the incentive to target our application.
- Risk: 10
- Strategies for monitoring
    - Assign a team to perform security tests on our application regularly
    - Ensure that application has no bugs that poses as a serious security threat
- Strategies for mitigation
    - Hire third parties that specialize in security vulnerabilities to test the application.
    - Regularly push out updates that patches security vulnerabilities.

Conclusion

By using measures above, risks will be less likely to occur due to the proposed monitoring strategies, and even if it does, the team will have mitigation measures to reduce their impact on the project. Hence, Team members should follow the guidelines stated in this document to minimize the possibility of risks from occurring.