Clubby
Risk Management Report

1 Introduction

Risks are an unfortunate reality of every development environment. The realization of these risks affects the development and deployment of products in a plethora of negative ways, from numerous delays and misses of deadlines to the possible cancellation of a whole project pertaining from the costs of its failures. To avoid these shortcomings as much as possible, we must plan for both mitigation and realization of these risks. By planning for the worst-case scenarios that comes from these risks, we make ourselves ready and prepare for swift and smooth reactions against their shortcomings. By documenting both previously encountered, and those that are yet to be encountered risks, we make ourselves ready for them, and make it possible to create plans for their mitigation and contingencies that we may need. To achieve these advantages, we create this document called Risk Management Report, to both prepare for and document the possible risks pertaining to our projects, to protect ourselves from the possible shortcomings as much as possible and in case of failed mitigation, prepare reactionary plans against those shortcomings, for a smoother development and deployment experience and greater results.

2 Description

Every project comes with risks, some physical some are virtual. Software projects are impacted by the risks more than other engineering projects since software projects depend on modern technology everyday and try to find solutions with new ideas rather than implementing just older ideas. This in mind, we tried to assess every possible risk in the project development process. The mitigation strategy for most of the cases tries to implement “open-closed” principle to avoid impact on code structure. The risks captured in the process listed below in descending order of impact magnitude:

1. One or more group member may be ill unexpectedly at the project time. There are 3 most probable illnesses for period of project, these are (in probability order from high to low): COVID-19, Influenza and Seasonal allergy.
   Strategy to be implemented: Redistribution of tasks of group member who is not available. If there is no member available and task does not affect other tasks' development, delaying the task deadline.

2. Since implementation details may change in development process, it is possible that database design can change in the process. However, rebuilding the database in every change is both overwhelming and time consuming.
   Strategy to be implemented: Use migration and version control strategies to avoid impact or data loss in already existing database. Accept only updates on the database design, dropping a table is not allowed unless it is a rollback.

3. As the project structure is new to each group member, some of the architectural patterns are tested for implementation thus may vary at the time of the development.
   Strategy to be implemented: Architectural design change is shared with stakeholders related to code base development, then a change is proposed. Risks are assessed for development process, and implementation is done with the lowest impact. In addition to this, architecture must be design with "open-closed" principal in mind.

4. Some changes may occur on design in the process of development since framework, development strategy and design patterns are new to group members.
   Strategy to be implemented: To minimize the change in design to avoid design pattern and coding standards conflicts, derive more basic structures and easier implementation for given case.

5. Some of the tasks will overrun the given deadline in the project. This may be a development, documentation or other task related to project.
   Strategy to be implemented: Request the status of the task, asses the risk related to it and reschedule task or reallocate members for the task with respect to risk assessment results.
6. Since the stakeholder meetings do not happen frequently and meeting times are short; some of the requirements may be misunderstood.
   Strategy to be implemented: Schedule private meeting if possible or ask questions related to problems if found. In case of feedback, assess the impact of change, implement if impact is low or create new use cases and requirements.
7. Bad deployment strategy of untested software may lead into unusable software release. This risk includes unreliable system component setups like database, web service engine, bad server configuration where software is deployed.
   Strategy to be implemented:
   *Briefing on deployment of Spring based software to the Linux based and Cloud based systems and configuration of deployment environments before deployment.
   *Deployment on smaller scale to mitigate scaling issues and complex structured deployments.
8. Since the group members do not have experience with the similar projects and schedules may vary, the time it takes to complete a task may be estimated wrongly.
   Strategy to be implemented: Assess the impact of delay, if delay does not impact delivery, extend the task deadline. If delay impacts delivery, reallocate available members to speed up the process.
9. As the meetings between developers are not frequent occurrences and the meeting times remain short; some of the use cases and thought processes may be misunderstood between developers.
   Strategy to be implemented: Increase the meeting count and duration as much as possible, create and maintain additional communication channels for unexpected or immediate questions/discussion. Create a clean communication environment between developers and Project Manager.
10. Since a project is not a person's own artifact but a whole's work, a group member may leave the project at any time and unexpectedly.
    Strategy to be implemented: Negotiate with member leaving and schedule a leaving date to mitigate impact. Until the schedule date, request completion of tasks given to member and documentation of the artifacts for other members to follow later in development.
11. Design requirements may change if one of the stakeholders generate new use cases or update existing use cases. These changes may lead to complete restructuring of code for given use case. For example: Private messaging may be moved to user space rather than club space
    Strategy to be implemented: Inform other stakeholders about the change, decide on whether to implement it or not. Analyze change if it is needed and implement the change by satisfying "open-closed" principle to minimize further changes on other components.
12. It is likely that there exist some software requirements that cannot be met by the framework. For example: Repository design pattern, front-end framework.
    Strategy to be implemented: Finding compatible frameworks, packages, or implementations for current frameworks. If the given strategy is not possible, finding and implementing a basic but capable method to current framework.
13. New members can join the group, however new members cannot have knowledge about the project architecture and process
    Strategy to be implemented: Give briefing about project process, inform member about architecture.

14. A frequently used development software's essential features turning out to have some of its essential features locked behind a paywall.
    Strategy to be implemented: Either seek alternatives in a regular basis or increase the research time given to such development software to ensure its most useful/essential features are reachable.
15. Multiple important and hard tasks, which a normal student inevitably has, ending up having their deadlines at around similar timeframes. Resulting in a drastically reduced time that one can give to the project at hand. Unlike the previous time related risks (5), this one is greatly outside of our control and firmly in the hands of outside stakeholders.
    Strategy to be implemented: Once realized request a postponement of the project deadline. If it is not granted, take the minimum necessary time from the rest of the compulsory tasks and/or sleep/personal time to get even a minimal chance to reach the finish-line. Use as much effort as needed for a closer-to-finish product before the final crunch times arrive.

3 Risk Management Report Specifications

#	Risks Forecasted in Planning	Mitigation Method
1	Framework's incapability on a software requirement	Finding compatible frameworks, packages, or implementations for current frameworks. If the given strategy is not possible, finding and implementing a basic but capable method to current framework.
2	Wrong estimation of delivery time of a development task	Reallocation of available group members to the task to speed up development process
3	Architectural design change	Resolving issue by implementing a new method with backwards compatibility
4	Database design change	Updating tables with new migrations rather than changing the old migration. This way database is persistence.

Table 1: Captured risks forecasted

#	Risks Not Forecasted in Planning	Mitigation Method
1	Use case implementation is not possible for given deadline	Project manager re-evaluated the development plan and updated group member tasks. Available group members joined to group member who had the given task, to complete the changed task in time.
2	Software that was being used for development ending up hiding some of its essential features behind a paywall.	Re-implemented, re-designed and re-thought parts of the program that needed such features. Every additional third-party software that is planned to be used is researched more thoroughly with additional checkboxes to fulfil (such as not having its essential features behind a paywall) and/or looked for their open-sourced alternatives.

3	Numerous simultaneous and outside tasks of great importance ending up in the same time frame to intended project completion, resulting in a harder time of project development at its last stages than what is intended.	Finish as much of the project as possible before such simultaneous task necessity becomes a reality. If the first mitigation fails, use as much extra time as needed for the last vestiges of project completion.

Table 2: Captured risks that are not forecasted

A. Risk Identities Descriptions:

Risk Identities	Description	Symbol
Product Risk	Product risks affect the quality or performance of the software being developed	PROD
Project Risk	Project risks affect schedule or resources	PROJ
Business Risk	Business risks affect the organization developing or procuring the software	BIZ

Table 3: Risk Identities Descriptions

B. Mitigation Strategies for Risk Types:

There are different types of risks in the project and risks with same type has similar strategies to mitigate risk. The mitigation strategies below are templates for mitigation of a risk.

1. **"Open-closed" Mitigation for Product Risks:** Most of the product risks feature a change in product. To avoid an impact on already implemented functions, the change must be done in a way that it is backwards compatible. The code can be extended but not changed. A code structure can change only it is not working as expected or has bugs.
2. **Dynamic Member Tasks for Project Risks:** Most of the project risks consists of delays in development process or change in an implementation in the time of development. To mitigate the risk of overshooting the deadline, members are reallocated in the development process dynamically and tasks are completed in a collaborative way.
3. **Inform, Brief, Propose and Change for Project and Product Risks:** For an ongoing development process, there exist many changes in requirements. Stakeholders may change their requirements; technology may not be compatible with designated strategy and architecture may need a change etc. To mitigate these kinds of changes, project group works collaboratively to find solutions. This mitigation strategy uses communication highly to meet these requirements. Project manager informs all stakeholders related to project development about one stakeholder change proposal, gives a briefing to development team in meeting tries to find a solution collaboratively and schedules the change and integrates it to development process.

C. Risk Type:

I: Indirect
D: Direct
I/D: Can be either Indirect or Direct at different times

D. Impact

1: Lowest impact on the related identity
2: Low impact on the related identity
3: Medium impact on the related identity
4: High impact on the related identity
5: Directly impacts the related identity

E. Probability

Probability of risk happening at the time of development process. This is calculated via analytics of the project and the other environmental statistics. Can be any percentage between 1% and 100%.
100% means definite certainty to occur
1% means the lowest possible chance to occur, almost impossible in small numbers

F. Magnitude

Impact * probability = Magnitude.
This value defines the importance and impact of the risk on project.

G. Owners

Project Manager: Osman Yiğit Koç

Software Architect: İlker Emre Koç

Configuration Manager: Berra Nur Sarı

Software Analyst: Eren Kumru

Software Tester: Abdil Buğra Ünal