Module	Engineering1 (Eng1) - COM00019
Assessment Title	Assessment 2, Cohort 2
Team	Quakthulu (Team 16)
Members	Aaron Price, Alec Coates, Charlie Curedale-Rayner, Eleanor Griffin-Smith
Deliverable	Risk Assessment and Mitigation

Risk Assessment & Mitigation

Introduction

Risk management is essential for software projects and involves the monitoring and mitigation of known risks and the iterative identification and assessment of new risks. It is crucial that problems that could threaten the project are recognised early in the development stage and preliminary mitigation measures identified, allowing for faster recovery from the impact that risk has had. [1]

Justification for Risk Format

Risks are defined in various forms:

1. **Project risks** affect the project schedule or its resources.
2. **Product risks** affect the quality or completeness of the product.
3. **Business risks** affect the organisation and how it procures or develops the software.

To manage these potential risks, the Risk Management Process (RMMM Plan) [2] will be used. This was chosen as it will provide a complete and iterable Risk Register. It is composed of:

- **Risk Mitigation** - the fundamental strategy that identifies risks, their impact, and priority and plans for mitigation.
- **Risk Monitoring** - the regular review of risks and changes to their likelihood and impact.
- **Risk Management** - the plan for dealing with a risk when/if it becomes a reality.

Risk Mitigation Process

To identify risks, a brainstorming team session was held before programming tasks were assigned. Research into the specific risks involved in video game development was also carried out [4] [5]. The goal was to keep risks simple to avoid unnecessary confusion when further iterations of the Risk Register were completed.

Based on research [3], it was decided to define risks as follows:

- **ID** - an identification string so the risk is easily referrable and identified.
- **Description** - a description of the identified risk.
- **Likelihood** - the probability of the risk occurring based on a simple, three-level tier of: Low, Moderate, High. (Simple because the project is small and non-critical).
- **Severity** - an expected level of impact if the risk occurs (using the same 3-tier level).
- **Rank** - a numerical score based on Likelihood and Severity to help prioritise and focus on the worst risks. Colour coding has been used to make high risks more visually apparent.
- **Mitigation** - plans to avoid, control or monitor the identified risks.
- **Owner** - an assigned owner to keep track of, highlight to the team and manage the risks if they become a reality.

Identified risks were added to a risk register, broken down by type of risk and sorted by Rank to prioritise key areas of focus.

Risk Monitoring & Management Process

These were implemented by including them on the agenda for the weekly team SCRUM sessions, where the owners of said risks would discuss if any changes have occurred in relation to their risks and if any further mitigation or management action was needed.

Tabular Presentation of Risks

Project Risks

ID	Description	Likelihood	Severity	Rank	Mitigation	Owner
1	Tasks assigned take more time than planned.	H	M	6	Stick to the critical path. Drop least important requirements, if needed.	Everyone
2	Member of team fails to complete a prerequisite task.	M	M	4	Assess where the problem lies. Assign more people to the task depending on the task's priority.	Aaron
8	Failure to build in time.	M	M	4	Stick to a prioritised plan of modules to code.	Alec
9	Postpone bug fixing until it's too late.	M	M	4	Stick to a prioritised plan of modules to debug.	Charlie
4	Code is lost.	L	H	3	Use GitHub to restore to a previous version of code. Team members to keep up to date local copies.	Everyone
6	Failure to break down the coding work into a detailed and prioritised plan.	L	M	2	Create a prioritised plan of modules to code.	Charlie
7	Failure to adequately monitor progress on tasks.	L	M	2	Start monitoring adherence to schedule at bi-weekly meetings.	Aaron
10	Poor team coordination.	L	M	2	Start holding more regular SCRUM stand-ups during Sprints.	Eleanor
3	Unforeseen circumstances (eg, member of team becomes sick rendering them absent for a period of time from the project).	L	L	1	Stick to the critical path. Reallocate tasks as necessary. Drop least important requirements, if needed.	Everyone
5	Sections of the code in the GitHub repository are not up to date resulting in inconsistent code.	L	L	1	Team members to commit and pull from GitHub frequently (at least daily). Create branches, if needed.	Everyone

11	Failure to allocate time for quality assurance (eg, quality of code, creatives).	L	L	1	Stick to a prioritised plan of modules to code (including time for quality control). Audit process to review code for quality and documentation prior to integration.	Alec

Product Risks

ID	Description	Likelihood	Severity	Rank	Mitigation	Owner
15	The code is not subject to rigorous testing for bugs/issues leading to the submitted version being inadequate.	M	H	6	Create a test plan to use during the testing phase. Create a time plan and review of adherence to schedule at meetings.	Aaron
17	Failure to playtest the "fun factor" so the end customer won't bid for the game/buy it. "Fun factor" is to include game play (per the product brief) that is challenging, but not too difficult, smooth and intuitive.	M	M	4	Proper prototyping and playtesting with balancing between legs, and beta tester to confirm controls are smooth and intuitive in the prototype.	Charlie
14	Lack of programming ability leads to features missing from the final product.	L	H	3	Allocate coding tasks according to team ability.	Eleanor
16	The chosen Java library (LibGDX) is no longer supported.	L	H	3	Transition to one of the other researched libraries.	Aaron
12	The features desired by the stakeholder are not implemented.	L	H	3	Request another customer meeting to discuss requirements.	Alec
18	The look (art assets) of the game is not appealing.	L	M	2	Allocate sprite creation to team members with experience with graphic design.	Charlie
13	Advanced features are implemented before/instead of fundamental features.	L	L	1	It’s impossible to implement this product’s advanced features before the fundamentals.	Aaron

Business Risks

ID	Description	Likelihood	Severity	Rank	Mitigation	Owner
19	The final product is incomplete.	M	H	6	Detail and document any missing features.	Charlie
20	The code is not iterable or able to be further developed due to poorly designed code and being undocumented.	L	H	3	Hold meetings to review code for quality.	Eleanor
21	The wrong game engine is chosen.	L	H	3	Research another game engine.	Eleanor

Bibliography

1. *Risk Management Approach And Plan*, Mitre. Accessed on: October 15, 2020. [Online]. Available: https://www.mitre.org/publications/systems-engineering-guide/acquisition-systems-engineering/risk-management/risk-management-approach-and-plan
2. *Risk Management in Software Engineering*, TutorialRide.com. Accessed on: October 15, 2020. [Online]. Available: https://www.tutorialride.com/software-engineering/risk-management-in-software-engineering.html
3. *Software Development Risk Management Plan with Examples*, Cast Software. Accessed on: October 15, 2020. [Online]. Available: https://www.castsoftware.com/research-labs/software-development-risk-management-plan-with-examples
4. *Risk Management: What, Why and How*, LeonardPerez.net. Accessed on: October 20, 2020. [Online]. Available: http://leonardperez.net/risk-management-what-why-and-how/
5. M Schmalz, H Taylor, *Risk Management in Video Game Development Projects*, ResearchGate. January 2014. Accessed on: October 20, 2020. [Online]. Available: https://www.researchgate.net/publication/262250152_Risk_Management_in_Video_Game_Development_Projects
6. I. Sommerville, *Software Engineering,* 8th Ed., Addison-Wesley, 2007, Chapter 5.4