-

Risk Assessment and Mitigation

Assessment 2 - Updated from Team 21

Hannah Thompson
Kyla Kirilov
Ben Hayter-Dalgliesh
Matthew Graham
Callum MacDonald
Chak Chiu Tsang

Assessment 1 (Team 21)

Our risk management process that we followed made sure that we had backup plans for any of the risks, with the most lively risks being covered first.

- Risk ID: uniquely identifies each risk
- Description: brief description of the risk
- Likelihood: assessed likelihood of the risk occurring (L = Low, M = Medium, H = High)
- Severity: measure of the impact the risk could have
- Mitigation: actions planned to reduce or manage the risk
- Ownership: team member responsible for mitigating the risk

Risk mitigation table structure ideas [1]

Assessment 2 (Team 25)

As we were handed over the project, a new and different set of risks became apparent and therefore it was necessary to add them to the risk register. Our risk register is presented in a tabular format below, using these categories in order to describe and evaluate the risk.

The ownership of each risk had to be updated to members of our new team because the risk register is available to all team members and stakeholders, ensuring transparency and accountability. Should an identified risk develop further, any member of the team should contact the named owner to implement mitigation and recovery steps. Therefore the owner of the risk must be a part of the current team working on the project.

In the table below, risks in **bold** were added for assessment 2.

ID	Type	Description	Likelihood	Severity	Mitigation	Owner
R1	Project	Loss of team members (e.g. due to illness)	L	H	**Introduce pair programming and groups of people working on different sections of the project.**	**Hannah**
R2	Product	Poor user engagement	M	M	Ask customer questions, gather user feedback e.g. questionnaire	**All group members**
R3	Project	The IDE a programmer uses may not support certain developer features, causing delays in dev time as documentation may refer to unavailable features	L	L	**Ensure that all members are using the same IDE**	**Kyla**
R4	Technology	Game engine features require hardware specific features to function	L	H	**Ensure all members who need access to specific game engine features have the corresponding specific hardware features**	**Callum**
R5	Product	Final game produced has library requirements that the end user cant meet	M	H	Ensure that the game produced is compiled into an executable, and there are no dependencies on unavailable/ deprecated code	**Ben**
R6	Product	Insufficient testing which	H	H	Use unit tests and have it automated to always check for	**Chak**

		can lead to bugs			bugs	
R7	Project	Project purpose and need is not well-defined	L	H	Conduct workshops or interviews to gather detailed requirements.	**Hannah**
R8	Technology	Incompatibility with different platforms or devices	M	M	Test game on different platforms, finalise what platform game should work on	**Matt**
R9	Business	Issues licensing and procuring artwork for the game	M	M	Find an open source library which allows the distribution of its art for the purpose of game development	**Kyla**
R10	Technology	Data privacy and cybersecurity	M	H	Develop data security policies and practices. **Monitor and restrict access to the repository.**	**Chak**
R11	**Project**	Time required to develop the software is underestimated	M	H	Check in with the group every week to make sure significant amount of work is being completed	**Hannah**
R12	Business	Regulatory Changes	L	M	Checking for regulatory changes and lobbying for favourable regulations can help manage risk	**Matt**

R13	Business	Development issues due to a roadblock caused by an error in the code	M	H	Use versioning control (i.e git) and branch management to determine stable and unstable versions of the game	**Ben**
R14	Product	Someone's code gets deleted or lost	M	H	Use GitHub to backup the code and ensure not everything is lost	**Callum**
R15	**Technology**	**Lack of comments in the inherited code**	**M**	**M**	**Allocate time to review the code of the existing codebase before adding new features**	**Kyla**
R16	**Project**	**Difficulty in understanding the architecture and design patterns used in the existing code**	**L**	**H**	**Organise a meeting with the previous developers in order to conduct code walkthroughs to gain insights into the architecture**	**Hannah**
R17	**Product**	**Integration challenges arise when new features need to interact with existing code**	**L**	**H**	**Implement modular design principles and establish clear interfaces to facilitate seamless integration of new features**	**Ben**
R18	**Technology**	**Compatibility issues with legacy dependencies and**	**M**	**M**	**Conduct thorough dependency analysis and update outdated dependencies to ensure**	**Matt**

		libraries used in the inherited code			**compatibility with new features**	
R19	**Project**	**Unforeseen legal or intellectual property issues associated with the inherited code**	**L**	**H**	**Perform a comprehensive legal review of the existing codebase to identify any potential intellectual property infringements or licensing conflicts**	**Callum**
R20	**Project**	**Absence of comprehensive test coverage for the inherited codebase, leading to uncertainty in the stability and reliability of new features**	**H**	**M**	**Prioritise the establishment of a testing framework and implement thorough test suites to ensure adequate coverage of both existing and newly added functionality**	**Chak**
R21	**Product**	**Risk of introducing unintended gameplay mechanics or breaking existing game features during the integration of new code**	**L**	**M**	**Conduct extensive playtesting sessions to identify and address any unintended consequences or disruptions to the overall gameplay experience caused by the addition of new features**	**Ben**

References:

[1]I. Sommerville, “Project Management”, in *Software engineering*, 10th ed. Boston, Mass. Amsterdam Cape Town Pearson Education Limited, 2016, pp. 644-651