Risk Assessment & Mitigation

Overview of our risk assessment

The risk assessment measures potential issues that may occur during this project's lifetime and how, as a team but also as individuals as part of it, we can mitigate them. We also have a plan for contingency - which details how we would deal with the risk should it occur.

There are a few **amendments** we have made since Assessment 1 to our Risk Assessment:

- From feedback received and looking at the document again we think we did put too much of an onus on the project client/sponsor to clarify elements of the development and in some cases, assist the team. This is quite excessive and as software developers responsibility for development ends at us and should not involve the client/sponsor.
- Also amended is that our first document may have placed too much responsibility for risks on the whole team rather than individuals. The Software Development project is a team project and should be considered as a team effort, however in that team we have roles and delegations and the majority of our risks need to have a person/team responsible for these. And in some cases, placing a risk on the whole team has the potential to be troublesome because in essence - it is easy to say that when everybody is responsible for a risk, no one person is.

On the whole though we consider our document to still be appropriate - risks are laid out in categories and for each risk there are full details, including a score which takes into account the likelihood of the risk occurring and the impact should the risk occur. We then explain how these risks can be mitigated (avoided or reduced in impact and likelihood), and the contingency for each risk; how the team would manage with the project as best as possible should the risk occur.

Our risks with quality are deemed suitably important as we are developing software that needs advertising and interest in assessment terms, so risks related to that area are pertinent (our game could be developed well but if nobody wants it, the project is inoperative and vice versa).

Likelihood \ Impact	Negligible	Moderate	Significant	Severe
Unlikely	Low	Low	Low	Medium
Possible	Low	Medium	Medium	High
Likely	Low	Medium	High	High
Almost Certain	Low	High	High	High

Figure 1: how are risks are prioritised and rated on Impact and Likelihood aspects

Risk Details			Risk Score			Mitigation and Contingency			Action	Raised
Category	Name	No.	Likeli-hood	Impact	Score	Mitigation	Contingency	Score after		
Software Requirement Risks	Requirements being impossible to fulfil	1.1	1	3	3	Contact project customer and explain the circumstances.	Attempt to accommodate the new requirements as much as possible.	3	HT, AD	15-Oct
	Change of requirements (SCRUM Feature Creep[1])	1.2	3	3	9	Keep work and code well documented, making it easy to modify.		7	AW, HT	15-Oct
	Requirements being poorly defined	1.3	2	3	6	Discuss and untangle the requirements as a team and decide on how to act on them.	Interpret requirements as a team as best possible.	4	HT, AD	15-Oct
	Not inspecting requirements correctly	1.4	2	3	6	When reading / receiving requirements, discuss and interpret as a team. Check understanding.	N/A	4	HT, AD	15-Oct
Risks with Implementation	Inadequate knowledge about tools	2.1	2	3	6	Members responsible for the implementation to find helpful resources, aiding knowledge of tools.	Use available tools to the best knowledge available / change tools.	3	HT, SD, RE	15-Oct
	Lack of skill	2.2	1	3	3	Find and make use of online tools, reading materials that will help.	Complete project brief to best standard possible.	2	HT, SD, RE	15-Oct

Risks with Implementation	Difficulty of implementation	2.3	3	3	9	Coding team to discuss problems with implementation and their solutions, seeking help where necessary.	Seek online support, documentation and any other useful source.	6	AW, Coding team: HT, SD, RE	15-Oct
	Lack of tools	2.4	1	3	3	Seek help online and/or technical support to acquire necessary tools.	Build product using the tools that the team can access at the time.	2	Coding team: HT, SD, RE	15-Oct
	Tools failure	2.5	2	3	6	Contact technical support where possible.	Use best available alternative tool.	6	as above	15-Oct
	SCRUM sprint time is too short to implement the necessary features	2.6	3	2	6	Plan time for the sprint accordingly so to give enough time for all the necessary tasks.	Increase the number of team members and redistribute work during the sprint.	4	HT, SD, RE, TP	24-Oct
Risks with Team Members	Human Error	3.1	2	3	6	Work completed by team must be checked and fully documented by other members to avoid mistakes.	Attempt to fix or isolate any errors made.	4	HT, SD, RE, TP	15-Oct
	Disagreement between team members	3.2	3	2	6	Disagreements should be discussed and decisions made democratically.	Team chair to make the final decision to avoid arguments.	4	Team chair: AW	15-Oct
	Short-term loss of team member (illness)	3.3	3	3	9	All work to be shared with all team members, no one person should	Redistribute work amongst other team members.	7	Team chair: AW	15-Oct

	Long-term loss of team member (dropout etc)	3.4	2	4	8	have exclusive knowledge / access to a resource.	Redistribute work, explain situation to project customer.	6	Team chair: AW	15-Oct
Risks with Software Quality	Documentation is inadequate	4.1	1	3	3	Update documentation regularly to ensure software is easy to use and understand.	Create as much documentation as possible when software is ready.	4	TP, AW, AD	15-Oct
	Lack of design documentation	4.2	2	4	8	Ensure documentation is kept updated and referred to throughout the project.		6	HT, AD	15-Oct
	Product has errors due to a lack of testing	4.3	2	3	6	Throughout the project, create and document tests of all features, of the program on all relevant platforms.	Perform as much testing as possible on project completion.	5	AW, HT	15-Oct
	Lack of interest for product during swap process	4.4	2	3	6	Ensure the "public face" of the product is kept updated and attractive to the end user. Maintain good software quality.	Gather any negative feedback and attempt to change the "public face" of the program.	5	TP, AW, AD	15-Oct

[1] A feature creep is defined as features being added to a project after the original scope is defined. [3, para. 2]

References:

[1] Putnam, Lawrence H., and Douglas T. Putnam, Software Investment Management. Fort Worth, 1986
[2] I. Sommerville, Software engineering, 7th ed. United Kingdom: Pearson Studium, 2007.
[3] C. Keith, Agile game development with Scrum. United States: Addison-Wesley Educational Publishers, 2010.