Risk Assessment and Mitigation

Stakeholders: Richard Paige, University of York Communications Office
Team: Barney Morgan, Cameron Smith, Harry Berge, Jake Phillips, Matthew Wilkie, Rob Weddell

Method

The process of eliciting risks began with a team meeting in which all possible risks were researched [1, 2], discussed and noted down. This list was then refined to ensure little overlap and to eradicate the unnecessary risks (either extremely unlikely to happen or risks so mild that they would cause no noticeable impact). After multiple iterations of merging similar risks together and removing extra risks, a concise risk table was produced.

Following this, each risk was assigned to a specific team member whose expertise and allocated role most suited the risk and its mitigation. The job of this team member was not necessarily preventative action but rather to mitigate the issue according to our assessment if the risk were to occur. This ensured that when a problem did arise, there was a dedicated member of the team focused on fixing it as soon as possible without any fuss. (Key: PM - Project manager, S - Secretary, CI - Client interface, SA - Software architect, WD - Web developer, TC - Technical Consultant)

The values of each specific risks impacts and the likelihood of that risk occurring were then created from an average of each team member's personally assigned values. Following this, a meeting was held to discuss the results of the risk assessment and to potentially correct any outliers or errors in the values. Should a value need changing, a group consensus was required before any alteration. After further research [3], it was decided to stick to a relatively simple method of ranking the likelihood and impact of each risk given the relatively small nature of the project and the fact that the software being developed is strictly non-critical. Thus, a simple 3 tier system was proposed to rate the likelihood, and a 5 tiered scale was given to the impacts. The team felt it was unnecessary to go into any more detail (for example the PRINCE2 risk register [4]) as there was no need to split columns further than had already been done as one general risk rating would suffice for the project. By keeping it simple like this, it enabled the team to apprehend the risks with ease at a glance and thus made the list more effective at both mitigating and preventing risks. Risks were ranked as such:

- Likelihood - (Green - Red)
  - Green - Fairly unlikely to occur
  - Yellow - Mildly likely to occur
  - Red - Most likely will occur
- Impact - (1-5, one being the lowest)
  - 1 - Not very impactful (Most likely will not even be noticed as a setback)
  - 2 - Very small impact (Will be noticed but will have little effect on the assignment)
  - 3 - Mild impact/setback (Will take people time to fix but won't be seen as too much of a problem)
  - 4 - Quite significant impact/setback (Will cause a lot of disruption to the assignment)
  - 5 - Massive impact/setback (Possibly assignment breaking level of damage)

Several categories were created in order to clearly attribute risks with the team's assessment of them. Risks can be assigned any number of categories. The categories and definitions are as follows:

- **TOOL** - This risk is mitigated by our choice of tool for this aspect of the project.
- **COMM** - This risk is related to communication between group members.
- **PROD** - This risk illustrates a potential degradation in quality to the end product
- **SCHED** - This risk may mean a delay in our project schedule. Internal deadlines may need change.

In the event of a new risk being discovered, a similar process will be repeated by which we hold a team meeting where we discuss the cause of the risk and the best way of mitigating the newly discovered risk. Once the risk mitigation has been discussed, the likeliness and impact can then be discussed as well as the owner of the risk. This information should all be documented in the risk table to provide the team with a table with as much useful information as possible.

References

[1] ITProPortal, Top 10 Software Development Risks [Online] Available: https://www.itproportal.com/2010/06/14/top-ten-software-development-risks/ [Accessed 30 Oct. 2018]
[2] Existek, Top 5 Risks in Software Development [Online] Available: https://existek.com/blog/risks-software-development/ [Accessed 30 Oct. 2018]
[3] V. Antinyan et al, Defining Technical Risks in Software Development [Online] Available: http://web.student.chalmers.se/~vard/files/Defining%20technical%20risks.pdf [Accessed 30 Oct. 2018]
[4] PRINCE2 Risk Register Wiki [Online]. Available: http://prince2.wiki/Risk [Accessed 1 Nov. 2018]

ID	Identified Risk	Owner	Categories	Consequences	Likelihood	Impact	Mitigation
1	Workload increase/Tired out	PM	COMM / PROD	Can result in developers getting overworked and can lead to sloppy work	🟥	**3**	Agree on comprehensive schedule for remaining tasks allowing for rest periods/days. Allowing for non-rush based progression throughout the assignment.
2	Getting sidetracked through development and design	PM	SCHED	This will result in loss of time and bulking up of the program which will cause multiple risks to happen	🟥	**3**	Focus/complete tasks related to requirements. Optional/superfluous tasks after.
3	Bugs within program	SA	PROD	This can slow development progression and can also lead to loss of marks	🟥	**2**	Thorough testing and use of continuous integration tools will alert the team to the presence of bugs so they can be resolved as soon as they are created. *
4	Being too overly ambitious	S / PM	COMM / SCHED	Making the game too hard to code up and putting a lot of stress on the developers of the game resulting in a possible failure due to lack of working game	🟧	**4**	Certain requirements may need to be reevaluated/changed at each development sprint if they are deemed too time-consuming.
5	Art not fully completed	CI	PROD / TOOL	Can lead to an annoying looking game style and code result in errors if a call is made for a sprite that isn't real	🟧	**2**	We will have backup art from third-party sources should the art not be completed on schedule.
6	Miscommunicat ion within the team	PM	COMM	Can create a spiral of problems later down the line and means people will have to do work which can set off other risks	🟧	**4**	In the meetings, actions for members are written explicitly. Oversight is also used to check whether a member is not wasting their time.
7	Absenteeism of people within the group	S	COMM	Because people hold such personalised roles it will be hard to add another person's role onto someone else which will result in other risks becoming a reality	🟧	**4**	During our meetings, each member will give an update on their current job so that if they are absent for any reason, another group member can easily assume their role.

8	Missing project requirements	PM	SCHED / COMM	This will result in missing the specification set by the customer which may result in a lower mark	🟧	**3**	We contact the customer to confirm and update the requirements regularly.
9	Lack of commitment	PM	COMM	This will result in the same way as basically an absenteeism	🟧	**4**	By having regular meetings the group can judge how committed people in the group are and adapt accordingly.
10	Team member disagreements - small	PM	COMM	Can result in minor feuds losing time and possible commitment issues	🟧	**2**	Whenever a disagreement occurs we the group votes on the topic. If of a social nature SEPR leaders will be contacted on how to proceed.
11	Changes to specification/thoughts/design	CI	COMM	Can result in massive changes and delays to the schedule and will result in a lot of more extra work	🟧	**4**	At the beginning of the project we briefly drafted the entire project, and before we work on each section we discuss it in a group reducing the likelihood of change on already completed work.
12	Lack of knowledge	TC	TOOL / PROD	This may result in non-optimal deliverables, affecting our ability to meet the requirements.	🟧	**2**	We will make sure to consult the project manager(s) (expert) on jobs that we are not entirely sure about. We will clearly outline any assumptions in the documentation.
13	Losing access to working computer	PM	TOOL	Affected group members would be unable to complete their work.	🟩	**4**	If no suitable alternatives are available (university, friends, etc) then that member's work would be reallocated for the period required to regain access.
14	Game running slowly	SA / TC	PROD	This will make the game annoying and frustrating to play, resulting in a bad product to be released	🟩	**4**	Software developers will be in charge of checking whether the game can run on multiple types of machines, and our code will be changed accordingly. *
15	Internal/group deadlines	S	SCHED	This will result in parts of the project being done late and might lead to a spiral affect	🟩	**3**	Regular meetings to keep track of deadlines and if a deadline is missed

	missed			other pieces of work being in on time	🟩		multiple members are assigned to complete the task.
16	Submission deadlines missed	PM	SCHED	Depending on how long we leave it for can result in zero mark	🟩	**5**	Assignment deadlines are recorded and submissions are planned for in advance.
17	Large team disagreements	PM / CI	COMM	Can result in a loss of a team member and can result in loss of work	🟩	**3**	In the case of a large team disagreement, we would contact the SEPR lectures requesting information on how to proceed.
18	No backup/loss of work	PM	TOOL / SCHED	Result of loss of marks and best case losing of time	🟩	**5**	We have a back-up on the cloud and google drive. We will also be creating local back-ups just in case we need to go back further than 7 days like google drive allows for, or if someone empties the trash folder.
19	Unreadable code	SA	COMM / PROD	Same as team miscommunication and can also lead to bugs in the code	🟩	**3**	Comments will be used to explain code but will also be held strictly to the design as to not make as confusing. *****
20	No way of tracking changes to the code	SA	TOOL / PROD	The design of the code would have to be changed and loss of time would be created also result of possible errors/bugs	🟩	**3**	We are using GitHub, this will allow us to use commits when we upload new code to be able to keep track of changes. *****

***** We will be using git repositories made with GitHub to store and manage access to our code. We intend to have a 2 branches; one for production (working, stable code) and development (in flux, may require documentation, etc..). When merging commits from our development branch, we will make sure to create pull requests so that the changes can be verified and understood by all developers. This will ensure all code in our production (master) branch will be fully working and documented.