1.5 | Risk Assessment

Introduction

Risk Management is the process of identifying and understanding the dangers to a project[1]. There are a multitude of risks of different severity and likelihood and it is important to identify and catalogue all of these as soon as possible so that steps can be taken to reduce any likelihood of failing to produce the intended deliverables[1]. This is a non-profit project so any identified risks will pertain mostly to deadlines and technical aspects, rather than profit, or company sustainability.

Identify and Group

To identify risk we must have a formal method of approach to ensure we consider all possible options. Chevron's method of cataloguing risk [2] breaks down the project into 7 distinct parts that can be assessed to find the risks in the specific area. This is not wholly suitable for us as this only assesses risks to information, we will expand upon this to make our own categories. We will then go through each category with the requirements in mind; identifying and categorising risks to the project before inserting them into the risk table. This will be repeated several time throughout the primary stages of the project as in dynamic projects the chances of risks changing as new elements are introduced is high[3]. After all risks are identified we can then start to take steps to reduce the risk to the project by the process of risk mitigation[1].

Ranking of Risk

We will rank project risks according to two categories: the likelihood of the risk occurring, and the severity of the risk itself. Likelihood is the measure of how often in the project a risk will occur on a scale of **Low**, **Medium** and **High**. *Low* means occurring once or less throughout the project and *High* means occurring several times throughout the process. Severity is ranked between **Low**, **Medium**, **High** and **Critical**. *Low* is a slight change, possibly prompting the need for team meeting. *Medium* is a significant change to architecture or code which will require extra work or a reworking of existing features. H*igh* shows a main flaw in a stage of the process which requires large amounts of work and may take several days to fix. *Critical* is a major disruption to work which could cause the project to be delayed or never finished, these should not happen in our project as the small scale of the project make it hard for such catastrophes to occur.

Categorising

As stated, we have decided upon categories into which our risks fall into, where any number of categories can pertain to a single risk. The categories follow:

- **Development** (affecting the development of the project and could lead to delays)
- **Technical** (affect the software or hardware used in the project)
- **Client** (risks where the client is involved in some capacity)

The table below provides a description of table headings relevant to the upcoming risk assessment table.

Heading:	Index	Risk	Description	Category	Severity	Likelihood	Mitigation
Description:	The identifying number to the risk	A title	A more indepth description of the consequences	As described	As described	As described	The steps we will take to reduce risk of this error affecting us

1.5.2- The Risks

1	Requirement from customer is unclear	Unclear on how to implement requirement could lead to wrong implementation.	**Development Client**	**Low**	**High**	This is an easily solvable issue, part of our planning process includes regular meetings with the client. This is the time to solve such issues and discuss all the requirements in detail. A good clear set of requirements at the start of the project will make all development far easier.
2	Requirement is missed	Requirement not implemented and clients wishes not met.	**Client**	**High**	**Low**	Constant rereading and discussing of the requirements makes this likelihood low, use of any spiral, or RAD technique makes the chances of this occurring low. Our implemented GANNT should ensure all the requirements are implemented as we have pre included all the system requirements into our development schedule.
3	Requirements are unachievable	A pair of requirements contradict each other or are far to unrealistic to be successfully implemented.	**Client**	**High**	**Low**	As long as the issue is spotted early on then fixing it will be easy, a meeting with the Client will solve any discrepancy. A greater problem would be if an implementation is more difficult than the team is capable of. This can be solved through proper planning and structure which show us if we are getting behind schedule and so able to solve the problem early.
4	Requirement is changed	A late change creates extra work that makes changes to the product necessary.	**Client**	**Medium**	**Medium**	As discussed in risk 1 and 2 we have an intended schedule of regular meetings with the client, meaning any changes to requirements should come about early on giving us time to change the product to suit the new brief. Our RAD process is suited to large amounts of change as it is.
5	Requirement causes disagreement	Team wastes time discussing how to implement a feature of the game.	**Development**	**Medium**	**Medium**	There are two ways of solving this, firstly meetings within the team to discuss and review work come about naturally as part of our dynamic team work. If this does not solve the problem it will be taken on to a client meeting as discussed in risk 1. As the client has the final say their should be no dissatisfaction within the team.
6	Group disagreement	Members of the team have different views on how to implement a part of the project.	**Development**	**Low**	**High**	This could be anything from the language chosen to a character design, for trivial decisions such as character design or other superficial features a general meeting should be enough to settle the disagreement. Larger concerns such as language choice should be solved before development starts. This gives time to decide a method so that people inexperienced in the language have time to learn it.
7	Team member become dissatisfied	One member disagrees with how the group is progressing, or dislikes the project, leading to a lower quality of work.	**Development**	**Medium**	**Medium**	This is likely to happen if the member's suggestions and ideas members are constantly shut down and he is given the all the boring or difficult work. Within our regular meetings he should have time to discuss this, he also would have the option of talking to the module head; going into the well set up support system provided to help with team problems.

8	Team member misses session	One scheduled session missed with or without warning. Productivity drops as an area is not developed.	**Development**	**Low**	**High**	Meetings are set up ahead of time so availability should be known ahead of time, this means meetings will either be scheduled for times when all members are present or that the member will have arranged for his absence and provided any work he needed to have done meaning his absence does not disrupt the session and all changes and progress will be recorded to keep him up to date.
9	Team member has extended period of inaction	Member is unavailable to contribute work for several days. His responsibilities may not be completed.	**Development**	**Medium**	**Medium**	As stated in Risk 8 all changes and progress is kept by all sub teams, meaning if a member is missing we will know exactly what he was doing and as we are using a shared resource format we have access to all his work meaning we can carry on where he left off, and he can then catch up when he returns using the same method. As stated in our code review methods in the methods section a cleaner will ensure all code is well documented to make it easier for anyone to takeover any piece of work.
10	Team member leaves group	Member leaves university or module. Risk of loosing all ideas/work done by him.	**Development**	**High**	**Low**	As stated in risk 9 we can to some degree take over any work done, it would also be prudent to not let any one member be totally in charge of one area of the project to reduce the cost of losing him. There is also a support system in the uni in case of losing a member.
11	Team member does not contribute	Member is capable of doing work but will not contribute leaving team at disadvantage.	**Development**	**High**	**Medium**	We will start by gently trying to encourage the member and try to work to his schedule or needs. If this gentle method does not work we will follow the step by step guide given in the module guide to resolve the issue if general team meetings and encouragement does not work. We will try given him work that he is interested in to try and motivate him.
12	Team member struggles with task	Member experiences problems but is willingingly trying to fix the problem.	**Development**	**Low**	**Medium**	As long as this is discovered early, which it will be thanks to RAD, we will have time to help the member with his work to guide him in how to do it so he can perform later well in later parts. We can also assign him to group work with a more skilled partner who can help guide him through the basics without taking over all the work we will not just assign the work unless the deadline is near.
13	Workload is unevenly distributed	Some members feel they are doing more work than others.	**Development**	**Low**	**Medium**	We will be able to share thoughts on workload during RAD meetings, at this point we can discover who's working on what and what their progress is changing if necessary. It is important to give an even split of coding and report writing to ensure no one is dissatisfied with the type of work or pigeon holed unnecessarily.
14	Deadline missed	Work is behind schedule and not ready on time. This will lead to mark drop.	**Development**	**High**	**Low**	Within each meeting there will be a progress check, this will be compared to the set plan allowing progress to be kept in schedule. This means that any time we are behind schedule we will soon know so can instantly take steps to get back up to speed or simplify the project to reduce work needed.
15	Work is lost	Storage breaks or computer	**Development**	**High**	**Low**	We are using shared online storages and all paperwork is scanned, to lose

		crash leads to lost work[4]				work the server would have to completely crash along with all are separate computers as we have downloaded versions of all work individually done.
16	Wrong version of code used	Old version of code updated and saved over new copy, several old copies acting at once.[4]	**Technical**	**Medium**	**Low**	Using a repository system gives us up-to-date versions of code so that trying to push out of date code will result in an warning, we can then individually go through the code to correct changes and ensure its now the correct version. As code is submitted regularly with good documentation labelling there will not be a wrong version used for long periods of time.
17	Module becomes unsupported	A Language, GUI, or Program used in the project. becomes unsupported on device.	**Technical**	**High**	**Low**	The hardware and software we are using is all very popular well used platforms, so the chance of it becoming unsupported is negligible. The only things that might become unobtainable such as sound and graphics being made copyright are all easily replaced.
18	Finished project does not match customers expectations	Requirements not fully met according to the client. Game would not meet requirements.	**Client**	**High**	**Low**	As stated in Risks 1-4 we will have regular meetings with the client and discuss the project meaning he will be aware of progress and can speak of his complaints at any time given us the chance to change or include any input.
19	Attempt to create a game far more complex than needed	Too much attempted makes a game which is unplayable or too complex to finish in time.	**Development**	**Medium**	**Medium**	This will happen either when the finished game goes through testing and is unsatisfactory to the testers, or when the project starts going behind schedule. Either way by having a set method leaving a good amount of time for testing and redevelopment we will be able to change any features deemed unnecessary or too complex. We could also consider a sounding board approach, pitching game ideas to a neutral board of critics to assess response.
20	Finished game included unknown bugs	After team believes the game to be done and after release a bug is found in the code.	**Technical Client**	**High**	**Low**	We will have sufficient time after development for several cycles of test review redo which will include testors from outside our team. This should find any bugs leaving time to fix them. We will also have code review for any completed piece of code by a team member who did not write it leading a team of other group members through a defined report structure of testing.
21	Finished game included offensive content or copyrighted material	The game will have to be taken down or changed after the finished period due to issues.	**Client**	**High**	**Low**	The game uses elements of the university and all of this must be reviewed by the client to ensure it gives the right image. Our testers will be given questionnaires which will ask if they found any content offensive. Copyrighting will not be a problem as our music/images will be taken from free repositories intended for amateur game creation, with a creative-commons license.

References

1. "Enterprise Security Architecture: A Business Driven Approach" by John Sherwood, Andrew Clark, and David Lynas. (CMP Books, San Francisco, 2005)

2. Michael J. Lewis Chevron, Houston, Texas, Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop Article No. 5

3. Pak, C., & Cannady, J. (2009). Asset priority risk assessment using hidden Markov models. Proceedings of the 10th ACM Conference on Sig-information Technology Education, Fairfax, Virginia, 65–73.

4. Whitman, M. E. (2004). In defense of the realm: Understanding the threats to information security. International Journal of Information Management, 24(1), 43–57.