Module	SEPR
Year	2019/20
Assessment	2
Team	Dalai Java
Members	Jack Kershaw, Max Lloyd, James Hau, Yuqing Gong, William Marr, Peter Clark.
Deliverable	Risk Assessment

Risk Assessment

While developing our product to meet the requirements of our customer, there will be various risks that we will need to consider to achieve the desired specifications. We have created a comprehensive table documenting the possible complications that could occur during the development process and deployment of the final product and our proposed solutions to mitigate the impact of such problems. We initially compiled these risks through a brainstorming session, before building on our initial ideas using relevant websites and papers [2, 3].

We have categorised the potential risks into three types: Project, Product and Business risks. Project risks are risks which could impact the deadlines we have planned, or the resources we have decided to use. Product risks are risks which will affect the final product, either by resulting in requirements not being met or the gameplay not being as smooth as it could be. This can encompass all of the potential problems that may arise in the final version of the product, from problems during development to larger, more abstract problems about aspects of the game. Business risks are risks which affect the stakeholders who we are building the game for, resulting in potentially drastic changes in requirements. Through using these categories, we believe our table is comprehensive as we have covered potential risks related to each aspect of our project in detail.

We have also categorised the risks into three levels depending on their impact on the product and the customer - High, Moderate and Low.

High impact – A problem that would actively impact the customer experience on a large scale and seriously hamper the development or deployment of the product. Risks of this kind would render the final product unacceptable to the stakeholders.

Moderate impact – A problem that would result in disruptions to core gameplay to the extent that it would not meet the specifications given by the stakeholders. It would be an inconvenience to the customer but would result in a game which is still playable.

Low impact – A minor problem that would only result in minimal inconvenience to the customer. It may have a cosmetic effect to the game but all core features specified by the stakeholders would still be present and functional.

Furthermore, we decided to categorise the potential frequencies of the risks occurring in three ways; high, moderate and low frequency. By categorising each risk both by impact and by frequency, we are able to understand which risks we need to be the most aware of during development, and hence which risks need to be monitored the closest in case mitigation is required. As our project is reasonably small and non-critical, we believe that using three impact and frequency categories will be sufficient as a number of risks will fall into each category, making them easier to monitor as a whole.

We have agreed on a risk reviewing plan which involves re-assessing the likelihood and severity of risks at two-week intervals. We have assigned an ‘owner’ to each of the risks we have identified; this person will perform the re-assessment for their given risks and report the status to the rest of the team. We decided on the owners of each risk by conducting a group meeting in which team members were assigned specific risks, and we ensured that ownership of risks was spread out evenly between team members.

Product Risks

ID	Description	Impact	Frequency	Mitigation	Owner
R1.1	Variables having non-intuitive names and hence being assigned incorrect properties	Moderate	Low	Write appropriate docstrings and always label variables appropriately.	Peter
R1.2	Using discontinued libraries with no appropriate documentation	Moderate	Low	Avoid using discontinued libraries if possible, otherwise minimise use of them during development.	Jack
R1.3	Variables of different scope having the same name and being assigned incorrect data	Moderate	Low	Global variable names should be recorded on a shared wiki to ensure all team members are aware of them and thus do not repeat them.	Max
R1.4	A file which has data which needs to be read could be moved and cause the program to crash	High	Low	Ensure file locations are updated when a file is moved, and use appropriate error-catching functions to minimise impact.	Peter

R1.5	The program or major sections of code being deleted or becoming corrupt	High	Moderate	Use version control systems such as Git to ensure that frequent backups are made.	Yuqing
R1.6	Java may not support a GUI on mobile or computer	High	Low	Test early versions of the code on the hardware specified by the customer to ensure the game is playable.	James
R1.7	Unit tests may not correctly test the functionality of classes	Moderate	High	Ensure each Unit test is proof-read by multiple team members to reduce the chance of incorrect outputs.	Peter
R1.8	Unit tests may not produce complete code coverage	Low	High	Perform manual tests to complement the Unit tests and to ensure no obvious errors are present.	Max
R1.9	There may not be code available to implement the desired features	Moderate	Low	Devise a 'reserve' feature for each desired feature which still meets the customer requirements.	Yuqing

Project Risks

ID	Description	Impact	Frequency	Mitigation	Owner
R2.1	The requirements of our product may change significantly	High	Moderate	Ensure we receive constant and frequent feedback from customers when developing	William

R2.2	Team members may leave the course or fall ill before the project is completed	Moderate	Low	Ensure all code is properly documented with meaningful identifiers, and ensure core functions are assigned to more than one person to prevent a single point of failure	Jack
R2.3	Team members may not be able to code the required functions	Moderate	Moderate	Ensure all team members have sufficient Java skills and practise before commencing	Jack
R2.4	Certain requirements may be overlooked	Moderate	Moderate	Keep all requirements in an area visible to all team members to ensure full understanding of all requirements, and that these are central to the development of the game	William
R2.5	Set deadlines may not be met	High	High	Ensure we give ourselves more time than necessary to complete each task to give ourselves time to resolve any issues	Max
R2.6	There may be conflicts within the team	Moderate	Moderate	Hold regular team meetings in which each team member is able to give their input and clearly resolve any issues	Peter
R2.7	The customer may not be enthusiastic about our final product	Low	Moderate	Ensure regular communication with the customer and present prototypes to them for feedback	Yuqing

Business Risks

ID	Description	Impact	Frequency	Mitigation	Owner
R3.1	Ideal game specifications were not researched properly, leading to poor reception of game.	Moderate	Moderate	Monitor the evolving needs of customers, adapting to the constant change in opinion.	Peter
R3.2	May need to upscale services based on number of players, leading to server downtime.	High	Moderate	Observe how the player base will grow and be expected to grow over time and put adequate systems in place to scale services.	Yuqing
R3.3	A similar product may appear on the market, leading to us programming a game no-one wants	Moderate	Moderate	Monitor any potential competition and maintain communication with stakeholders, anticipating major changes in requirements.	James
R3.4	Servers may be hacked leading to loss of customer data.	High	High	Have proper security measures in place with corresponding monitoring systems so we are aware of any ongoing or potential attack.	Will