Risk Management

Outlining the risk factors involved with the development of the software solution and the process behind it, analysing and developing a plan of action for each risk.

# Risk Factor 1

|  |  |
| --- | --- |
| **Risk Identification**  The time estimated to develop the final product may be underestimated.  **Impact**  Incomplete product, dissatisfied customer. | **Analysis**  Poor time management:   * Team members may become side-tracked over the duration of the project. * Could hinder progress made for certain tasks. * Low likelihood.   Technical problems with toolset:   * We may experience problems with the toolset during development. * Solutions to said issues may take a while to find. * Medium likelihood. |
| **Mitigation**  Be lenient with initial time management – by being generous with allocating time per task during planning, increases the chance that tasks actually get completed ahead of schedule. | **Monitoring**  Assign a team member to regularly check on progress of tasks in comparison with plan. |

# Risk Factor 2

|  |  |
| --- | --- |
| **Risk Identification**  Learning curve for learning new technology and processes for the project may be longer than expected.  **Impact**  Final product doesn’t hit all specification points. | **Analysis**  Lack of experience:   * Fair to assume some aspects of the project may be new to some/all team members. * This inexperience may mean time required to learn new aspects may take longer than expected before solution can be developed. * Medium likelihood.   Technical problems with programming:   * The programming section of the project will be the most susceptible to technical issues. * Bugs/errors/issues may hold up progress. * Medium likelihood. |
| **Mitigation**  Make sure all team members can cover/aid the main programmer(s) of the team – will ensure extra help is always at hand if any issues are encountered.  Have the more confident programmers lead the programming phase – plays to the strengths of the team. | **Monitoring**  Have a team member oversee and organise the programming tasks.  Regular reviewing of progress. |

# Risk Factor 3

|  |  |
| --- | --- |
| **Risk Identification**  Team member(s) may become ill/unavailable.  **Impact**  Greater stress on other team members, who may become too thinly stretched to deal with workload. | **Analysis**  Uncontrollable factors may impact the team:   * Uncontrollable factors including illness, injury, bereavement etc. could happen at any time. * In such cases, it is understandable the team member becomes distant from the project. * Low likelihood. |
| **Mitigation**  Difficult to prepare for something so unpredictable. However, during planning, try to allocate workload evenly so that each member is able to cover a part of a teammates dropped workload.  Outline essential and less/non-essential spec – by making sure essential requirements of the software are met, allows the team to complete a basic build of software before solving aspects that aren’t as essential in the grand scheme. | **Monitoring**  Regular reporting of work completed, and individual plans on what to work on next on the agenda.  Have a project manager maintain an even and balanced workload across the team, re-allocating when necessary. |

# Risk Factor 4

|  |  |
| --- | --- |
| **Risk Identification**  A major readjustment to the software requirements/specification.  **Impact**  A full readjustment might not be possible to complete within the given timeframe – leading to an incomplete final product. | **Analysis**  Client may be oblivious/not willing to be understanding of how big of an impact the readjustment is:   * May make the task at hand too much for the team to handle at short notice. * There may be difficulty in implementing a new requirement. * A modification to an existing requirement may require adjustments throughout the code rather than just one part. * Low likelihood.   The readjustment could be out of the team’s scope of knowledge/ability:   * The client may ask for the implementation of something that the team aren’t fully confident with. * Would require time to learn and plan how to implement – which would be difficult under already strict time constraints. * Medium likelihood. |
| **Mitigation**  Try to make the code as modular as possible, with the aim of making parts of the code/software as little reliant on other parts of the code as possible – this will help to reduce the amount of code that would need major refactoring in the event of an adjustment being made. | **Monitoring**  Have the overseer of the programming side check over code to make sure submitted code has as few dependencies as possible. |

# Risk Factor 5

|  |  |
| --- | --- |
| **Risk Identification**  The size of the software may be underestimated.  **Impact**  Final product may be unnecessarily large in size or bigger than planned. Could be an issue for customers. | **Analysis**  Whilst no limit has been placed on the size of the software and memory required by it, we want the software to be accessible and open to a large audience, so size and memory is important.   * Software too large in size could put off some customers. * This could lead to lower sales and/or poorer customer satisfaction. * Low likelihood. |
| **Mitigation**  Identify a target for file size and memory required by the game before developing the software. As a team, we have agreed that the game will require at least 50mb of RAM in order to run, so that we know how RAM-hungry we’re able to make the software, preventing the software from exceeding that limit. | **Monitoring**  Regular test runs of the code during development, allowing us to monitor how much RAM the game uses during the test. |

# Risk Factor 6

|  |  |
| --- | --- |
| **Risk Identification**  The execution time of the code produced may be slower than expected.  **Impact**  Poor user experience, leading to dissatisfied customers. | **Analysis**  Depending on how we go about implementing the game, the software we produce will behave differently and at different speeds.   * It’s important that the code executes efficiently and quickly to provide good user experience. * Coding complex solutions to simple tasks could cause this. * Customers may become unhappy with slow loading times etc. * Low likelihood. |
| **Mitigation**  Always look to try to implement the most time efficient solutions in the code.  Furthermore, leave some time at the end of the programming phase to allow thorough proof-reading double checking to be made to make sure that the most optimal time solution has been used. | **Monitoring**  Have the programming overseer check that all submitted code is of optimal time efficiency – this will be easier done in increments as new code is submitted, compared to having to do a single large check of the whole code at the end of the project. |

# Risk Factor 7

|  |  |
| --- | --- |
| **Risk Identification**  Hardware combination on user’s side may not allow game to run properly/at all.  **Impact**  Unhappy customers and a possible bug/error in the software. | **Analysis**  It is entirely possible that some hardware combination doesn’t happen to work with the game on a customer’s computer.   * There are a number of reasons why this may be a case. * We are only able to test the game to a more general, widespread degree. * There may well be a very specific situation which prevents the game from working, of which we were unable to test for. * Medium likelihood. |
| **Mitigation**  Once the game has been completed and has passed all main and unit testing, we will make an effort to test the game on as many different personal machines as possible, to simulate how the game would be distributed to customers by the client.  It will also be beneficial for us to test the game on PCs running different operating systems and software, etc. | **Monitoring**  Have the programming overseer review all code – if a section of code has been submitted that could be made more concise or streamlined, then have him edit said code to fit better within the overall code. |

# Risk Factor 8

|  |  |
| --- | --- |
| **Risk Identification**  AI could become “too smart” and learn the game too well.  **Impact**  Could end up creating an AI that is too good at the game, resulting in users struggling too much against AI players. | **Analysis**  There are many ways of implementing an AI user into the Property Tycoon game, but some methods of AI implementation could cause the AI to be forever learning the game, and if allowed to do so, in certain scenarios the AI could become “too good” for the average user and becoming almost unbeatable.   * Could result in human players becoming frustrated with the game due to not winning enough. * Important to make sure a fair and satisfactory AI implementation is utilised. * Low likelihood. |
| **Mitigation**  Conduct research into the AI implementation subject – try to find which methods of implementation are best suited to a scenario of Property Tycoon. | **Monitoring**  Convey regular reviews and tests to make sure the AI is at a reasonable level and not too difficult to beat for the average user. |

# Risk Factor – Exceptional Addition

Having completed the risk management plan at an earlier date, the increase in the severity of the coronavirus outbreak in the UK has unsurprisingly caused a lot of difficulties and change for the project. As a result, we felt it necessary to revisit the risk management plan and analyse this new risk factor and outline how we would go about tackling the issue.

|  |  |
| --- | --- |
| **Risk Identification**  Coronavirus outbreak.  **Impact**  Project may take longer to complete, or sections may not be as high standard as expected. | **Analysis**  Negative impact on internal communications:   * Social distancing rules out in-person meetings and groupwork, which can often be more beneficial than digital communication. * High likelihood.   Removed access to certain resources:   * University and libraries closed. * Not all team members have access to personal computers of the same standard as the university PCs. * High likelihood. |
| **Mitigation**  Ensure that digital communications within the group are more structured and detailed – this will help to make up for the detail lost with face-to-face communication.  More frequent communication – increase the frequency of team meetings via video/voice chat.  Delegate tasks that are more intense and computer-demanding to team members that have access to higher spec machines. | **Monitoring**  Staff level monitoring (project manager).  More frequent team chats/reviews. |