|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Risk Identification | Risk Type | Causes | Likelihood | Severity | How to Avoid Risk (planning) | How to Monitor Risk |
| Falling Behind Schedule | People/Estimation | * Not updating group members on progress * Spending too long on certain tasks * Devoting too much time to research/planning | High | High | 1. Regular meetings 2. Contingency plans 3. Checklists 4. Share realistic completion dates 5. Change schedule when necessary | 1. Update group members on progress 2. Count how many times the schedule has had to be changed |
| Software Becomes Inadequate for Project | Technology | * Improper assessment of software requirements * Unexpected update/change in software | Moderate | High | 1. Keep list of alternatives 2. Thorough research of software choice | 1. Check changes within software updates |
| Failure for Group to Work Congruently Towards Tasks | People | * Not updating groups members on progress * Not working on tasks together within small groups | Moderate | Moderate | 1. Regular meetings 2. Timetabled hours to perform tasks together | 1. Track group meeting attendance |
| Forgetting About Smaller Tasks | People | * Not following PERT chart * Thinking of them as insignificant | Low | Low | 1. Follow PERT chat 2. Give each person their own task | 1. Checking checklist for number of incomplete tasks |
| Inflexible Implementation of Street Names, Cards etc | Requirement | * Narrow minded planning/design | Low | High | 1. Make it a topic at the next meeting 2. Assess software options based on group’s prior knowledge | 1. Keep list of tasks with undecided methods of completion |
| Sudden Growth in Requirements | Requirement | * Not identifying parts which should be flexible * Not planning on how they could be made flexible | Moderate | High | 1. Careful planning | 1. Multiple people should test code for flexibility |
| Improper Designs for Code/UI/ | Planning | * Neglection of planning stage * Unrealistic design | Moderate | Low | 1. Coders review design plans | 1. Always share design ideas with people who will implement it |
| Incorrect Interpretation of Specification | Requirement | * Failure to agree as a group what certain phrases mean * Failure to use the same interpretation of the specification | Moderate | High | 1. Create clearer version of specification 2. Ask Watson Games for clarification | 1. Group reviews of group members tasks |
| Absence of Team Members | People | * Team members falling ill * Conflicting priorities of team members * Team members dropping out from the project | Low | High | 1. Ensure collaboration of tasks to share knowledge 2. Ensure team members provide advanced notice of absence | 1. Review progress at weekly meeting |
| Unnecessary Addition of Extra Features | Requirement | * Failure to adhere to set of requirements * Misinterpreting user requests | Moderate | Low | 1. Produce a clear set of requirements 2. Comply with mandatory requirements first and foremost | 1. Discuss relevance & time cost of extra features at team meetings |
| Game Breaking Bugs | Planning | * Poor/unsophisticated code * Inadequate low-level design | Moderate | High | 1. Identify the most severe bugs during the test phase 2. Leave sufficient time to fix bugs 3. Document software | 1. Document bugs during testing |
| Code That’s Difficult to Use and Maintain | Maintenance | * Inadequate documentation of software * Suitable low-level design | Low | High | 1. Use pair programming to ensure readable code 2. Utilise Javadoc appropriately | 1. Review code on a constant basis |
| Inadequate Performance | Requirement | * Inefficient code * Poor choice of software * Inadequate design | Low | Moderate | 1. Investigate the use of multithreading 2. Detailed low-level design | 1. Rigorous system level testing |