Team FooBar()

Risk Management

Version 2.0

1. **Time underestimation (Estimation)**

- Risk Classification
    - Project
- Description
    - The time and resources required to develop the software is underestimated.
- Probability
    - High
- Effects
    - Tolerable
- Strategy
    - *Avoidance* – We should make sure we set aside more than enough time to accomplish each task.

2. **Software defects (Technology)**

- Risk Classification
    - Project and product
- Description
    - As the size of the project increases, the more likely it is that new bugs will arise in the software.
- Probability
    - High
- Effects
    - Serious
- Strategy
    - *Avoidance* – To avoid this, we should fix existing bugs during development as soon as they are discovered.

3. **Poor presentation (Organization)**

- Risk Classification
    - Project and product
- Description
    - If we are unprepared for the presentation of our software, it could lead to a poor presentation and deduction of points
- Probability
    - Moderate
- Effects
    - Serious
- Strategy
    - *Avoidance* – To avoid this, we should meet and rehearse before the actual presentation.

4. Failure to meet all requirements (Requirements)

- Risk Classification
    - Project and product
- Description
    - There is a risk of not meeting all requirements due to poor time management.
- Probability
    - Moderate
- Effects
    - Tolerable
- Strategy
    - *Avoidance* – To avoid this, we should be prepared, multitask, and use our time wisely.

5. Schedule conflicts (People)

- Risk Classification
    - Project
- Description
    - Some team members may have conflicting schedules that do not allow them to make every team meeting
- Probability
    - High
- Effects
    - Serious
- Strategy
    - *Contingency* – If this happens, we will make sure to provide team member(s) with the appropriate information from the meeting.

6. Lack of effort (People)

- Risk Classification
    - Project and product
- Description
    - There may be a lack of effort in some aspects of development.
- Probability
    - High
- Effects
    - Serious
- Strategy
    - *Avoidance* – Lack of effort should be avoided to ensure that our project and product is of the highest quality that we are capable of producing.

7. Team members working outside of their assigned task(s) (Organization)

- Risk Classification
    - Project
- Description
    - A team member may work on a feature that is not assigned to them and potentially break the code without the other member knowing how to fix it.
- Probability
    - Low
- Effects
    - Serious
- Strategy
    - *Avoidance* – This risk could be avoided completely by assigning each member their own tasks and making sure that they understand that they should not work outside of their own features.

8. Vague requirements (Requirements)

- Risk Classification
    - Product
- Description
    - The requirements may be written vaguely to where they could be misinterpreted, therefore making features not do what the client is expecting them to do.
- Probability
    - Very high
- Effects
    - Catastrophic
- Strategy
    - *Contingency* – When this occurs, the team members should make sure they have the same understanding of the task. The team member losing a task should find a task or help another team member to take they place of that task.

9. Team member illness (People)

- Risk Classification
    - Project
- Description
    - Team member may be ill and unable to work on the project.
- Probability
    - Very high
- Effects
    - Catastrophic
- Strategy
    - *Contingency* – Reorganize task assignments to where team member's tasks overlap with others.

10. Client dissatisfaction (Client)

- Risk Classification
    - Product
- Description
    - Even though our product may meet most or all requirements, the client may not be satisfied with the final product
- Probability
    - Low
- Effects
    - Insignificant
- Strategy
    - *Avoidance* – Meet with the client to get feedback about our progress to make sure we are on the right track.

11. Unreadable generated code (Tools)

- Risk Classification
    - Project
- Description
    - Code generated or integrated from other projects may not adhere to our programming conventions.
- Probability
    - Moderate
- Effects
    - Tolerable
- Strategy
    - *Contingency* – When this happens, we should reorganize, rewrite, and restructure as much code as possible to make it understandable and readable by all team members.

12. Lack of proper subject training (Tools)

- Risk Classification
  - Project
- Description
  - Team members may lack knowledge and understanding of particular requirements that need to be implemented.
- Probability
  - Very high
- Effects
  - Serious
- Strategy
  - *Contingency* – If something is encountered that a team member is unfamiliar with, we should seek other team members for help on that particular subject. If no team member is familiar with the subject, seek guidance from Dr. Cherry. If Dr. Cherry's help is not enough, YouTube.

13. Unexpected project scope expansion (Requirements)

- Risk Classification
  - Project
- Description
  - The project scope may be unexpectedly expanded by the addition of new requirements.
- Probability
  - Moderate
- Effects
  - Tolerable
- Strategy
  - *Minimization* – We should minimize the effects by anticipating the addition of requirements and implementing these potential requirements before they hit the paper.

14. Implementing changes before pulling the latest version (Technology)

- Risk Classification
  - Project
- Description
  - Team members may work on code and implement new functionality before pulling the latest version from the repository.
- Probability
  - Low
- Effects
  - Insignificant
- Strategy
  - *Contingency* – If this occurs, we will revert back to a previous iteration in the history of our version control. The team member who forgot to pull the latest version will have to re-implement their changes.

15. All team members get drafted for selective service (People)

- Risk Classification
  - Project and product
- Description
  - All team members may be drafted for selective service at the same time.
- Probability
  - Very low
- Effects
  - Catastrophic
- Strategy
  - *Minimization* – We should minimize the effects of this occurring by finishing our tasks as soon as possible to ensure we would get some credit for the project if this were to occur.

Version 1.0

16. Team miscommunication (People)

- Risk Classification
  - Project
- Description
  - We experienced miscommunication during team meetings and in the planning process which caused us to do more work individually.
- Probability
  - High
- Effects
  - Tolerable
- Strategy
  - *Avoidance* – During team meetings, we should have further clarified what each task consisted of. We used software to list each team member's tasks and their descriptions to help with this, but the miscommunication still occurred.

17. Team group size (Organization)

- Risk Classification
  - Project
- Description
  - Adding an extra member to the team resulted in more complications with the overall project.
- Probability
  - Very high
- Effects
  - Serious
- Strategy
  - *Minimization* – If each member had the same experience and knowledge about game development, we could minimize on leadership roles.

18. High work load (Requirements)

- Risk Classification
  - Project and product
- Description
  - The extensive requirements and high work load proposed a risk to getting the project done before the deadline
- Probability
  - Very high
- Effects
  - Catastrophic
- Strategy
  - *Contingency* – We could have gone to the client and explained the situation but the deadline could not be pushed back and the work load would not be minimized. Our contingency plan was to deal with the damage and finish the basic development of the software.

19. Choice of programming language (Tools)

- Risk Classification
  - Project
- Description
  - There was a risk in selecting the appropriate programming language for game development that would mesh with each individual team member well.
- Probability
  - Moderate
- Effects
  - Tolerable
- Strategy
  - *Minimization* – We met as a team and made a list of potential languages to use and then simplified the list and made our decision.

20. Software size underestimation (Estimation)

- Risk Classification
  - Project and product
- Description
  - We initially underestimated the size of the software, thinking it would be a simple maze game.
- Probability
  - High
- Effects
  - Serious
- Strategy
  - *Minimization* – GanttChart software helped us break down each task into subtasks and visualize the workflow and the actual size of the project.