Here’s a breakdown tailored to your game project:

**8. Risk Identification and Assessment**

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Likelihood** | **Impact** | **Mitigation Strategy** |
| **Scope Creep** | Medium | High | Stick to your initial scope and prioritize features using a roadmap. Only add new features if time allows. |
| **Time Management Issues** | High | High | Use a detailed timeline or Gantt chart. Break tasks into smaller milestones and set weekly goals. Use version control (e.g., Git) to track progress. |
| **Technical Bugs or Crashes** | High | High | Regularly test each feature, use Unity’s debug tools, and fix bugs during development sprints. |
| **Insufficient Playtesting/Feedback** | Medium | Medium | Involve testers early. Use feedback forms or ask friends/classmates to test and report issues. |
| **Asset or Code Loss** | Low | High | Use GitHub for version control and make frequent backups of your project files. |
| **Balancing Issues (Combat/Difficulty)** | Medium | Medium | Create a difficulty scaling system and test different scenarios. Adjust based on playtest feedback. |
| **Unfamiliarity with Unity Features or C#** | High | Medium | Allocate time to learn new concepts. Use Unity Learn, tutorials, and forums. Don’t hesitate to ask for help. |
| **Hardware Limitations (Testing on Low-End PCs)** | Medium | Medium | Test the game on different hardware. Optimize assets, reduce draw calls, and limit resource-heavy effects. |
| **Team or Supervisor Communication Issues** | Low | High | Schedule regular meetings, send updates, and document decisions clearly. Have backup communication channels. |