**1. Introduction**

This document is the software design for our video game project, which was built using the Godot Engine and integrated DevOps practices. Its purpose is to guide the software development lifecycle by providing a structured plan that aligns technical and collaborative workflows. This phase is crucial because it sets the foundation for developing, testing, deploying, and maintaining the application.

Our game will run on PCs with Windows and Linux operating systems and will be lightweight enough to operate on systems with integrated graphics. The development will utilize **Godot 4.x**, with **GDScript** as the main language. The project will also include tools such as GitHub (for version control and CI/CD), Discord (for communication), and Trello (for task tracking). Automated testing will be implemented using **GDUnit3**, and GitHub Actions will manage the build and test pipelines.

A computer screen shot of a computer

AI-generated content may be incorrect.**Entity Relationship Diagram (ERD)**

**Sequence Diagrams (UML)**

A diagram of a game

AI-generated content may be incorrect.

**Architecture Design Diagram**

A diagram of a computer

AI-generated content may be incorrect.

**Class Diagram**

A screenshot of a computer

AI-generated content may be incorrect.

**Entity Relationship Diagram (ERD)**

This ERD shows how different things in the game—like players, enemies, and levels—connect and talk to each other. It’s like a map of friendships. Each box is a character (like “Player” or “Enemy”), and the lines show how they are related. It helps us organize game data clearly and avoid confusion.

**Sequence Diagram (UML)**

The sequence diagram shows the order of actions. It explains who does what and when. Characters (like Players or Game Managers) talk to each other step-by-step. It’s great for showing how game actions happen in time—from pressing a button to what the game does next. It's super helpful for planning.

**Architecture Design Diagram**

This diagram is like a bird' s-eye view of the game’s brain. It shows how big parts of the game (such as sound, graphics, and logic) work together. Each box has a special job, and arrows show how they talk. It helps developers build, connect, and update parts without breaking them. It's like building blocks for games.

**Class Diagram**

The class diagram is a blueprint for the game's characters, items, and everything. Each box is a class with powers (methods) and secrets (variables). It shows who can do what and who gets what. It helps us reuse code, fix bugs fast, and make everything work like a team.