**Implementation and Development**

**Project RPG**

***CIS 5898***

***Joseph Blenkhorn***

Contents

[Document Introduction 3](#_Toc193545704)

[Project Updates 3](#_Toc193545705)

[Source Code 3](#_Toc193545706)

[Documentation 7](#_Toc193545707)

[Explanation & Comments 7](#_Toc193545708)

[Code Files Documentation 7](#_Toc193545709)

[Testing Plan and Test Cases 10](#_Toc193545710)

[Testing Plan 10](#_Toc193545711)

[Test Cases 10](#_Toc193545712)

[User Interface Design and Prototypes 11](#_Toc193545713)

[Main Menu UI 11](#_Toc193545714)

[Play Screen UI / UX 11](#_Toc193545715)

[Webstore & Business Analytics UI 12](#_Toc193545716)

[ARtstyle UI/UX Inspirations 13](#_Toc193545717)

[Deployment Plan 14](#_Toc193545718)

[Course Based 14](#_Toc193545719)

[Real World 15](#_Toc193545720)

[Version Control History 16](#_Toc193545721)

[Weekly Progress Reports 17](#_Toc193545722)

[Integration & Testing Report 18](#_Toc193545723)

[Modules 18](#_Toc193545724)

[Testing & Debugging 18](#_Toc193545725)

# Document Introduction

This is the Summary document outlining each required deliverable for the Implementation and Development Phase.

Some of the deliverables for this document, including the source code and Test Plans, are provided as separate documents within the assignment submission.

# Project Updates

Below is a list of updates since the prior submission.

**Project Planning and Design Deliverable Rev2**

Following the development of the project, the Project Planning and Design Document has updated the following:

* Use-Case and Sequence diagrams changed to different UML Syntax format
* Updated class diagrams to reflect code development

The Document has been resubmitted alongside the submission of the I&D Assignment

# Source Code

The source code is available in two formats

1. Local Zip Folder titled ‘Source\_Code’
2. GitHub repository.

The current repository is set to private to prevent public access to the source code. The GitHub format, and publishes will be demonstrated during the project demonstration.

An example screenshot of the GitHub Repository is shown in the figure below:

A screenshot of a computer

AI-generated content may be incorrect.

Figure : GitHub Private Repository

A screenshot of a computer

AI-generated content may be incorrect.

Figure :GitHub Desktop Application

To Access the source code C# Files, navigate through the source code folder to the following file location:

***Source\_Code.zip\Source\_Code\GameProject2\Assets\Scripts***

The C# code files are followed by the ‘.cs’ file extension. The unity editor generates the ‘.cs.meta’ files for compatibility purposes.

A screenshot of a computer

AI-generated content may be incorrect.

Figure : Source Code

# Documentation

Documentation for Project RPG is available in various documents and environments. The information below highlights how the documentation is stored, organized, and made available for the user during this phase of the project submission.

### Explanation & Comments

The source code is made available in the ‘Source Code’ Zip File (see above in Source Code Section). The code is written in C# with the ‘.cs’ file extension.

The C# files include comments and explanations that describe what the code is doing within the game, and also how some specific functions are operating. Some basic Object-Oriented Programming techniques are explained, however due to redundancy and volume, some functions are not described with every instance. Examples of this include:

* If / Else Functions
* Variable Types (int, float, Array, etc)

A screen shot of a computer

AI-generated content may be incorrect.

Figure : Example of Code Comments from C# Script

### Code Files Documentation

The Source Code Zip files include two traditional ‘README’ files. The first file is the automatically Generated GitHub file, with a specific file extension.

The second file (README2) provides a standard .txt file type that can opened with any text editor. This file includes explanation on how to install the program, open the program in the Game Editor for development, as well as provides documentation for how to play the game.

A close-up of a computer screen

AI-generated content may be incorrect.

Figure : Example Snapshot of README2 Documentation

A screenshot of a computer program

AI-generated content may be incorrect.

Figure : Snapshot of README2 on how to play

# Testing Plan and Test Cases

The information below summarizes the testing plan and the subsequent test cases. For the complete deliverable of the Testing Plan and Test Cases, see the Folder in the Assignment submission titled “Testing Plan and Test Cases”

### Testing Plan

The Testing Plan outlines the critical tests that must be performed during the Testing and Validations Phase. Each Item in the testing plan has a subset Test Case, outlining the steps needed to validate the testing and project requirements.

**Testing Plan Summary (Full Document in Deliverable)**

1. Functionality Testing
2. Performance & Load Testing
3. Useability and User Acceptance Testing Phase 1
4. Useability and User Acceptance Testing Phase 2 (If Applicable)
5. Bug Reporting and Resolution
6. Regression Testing (If Applicable)
7. Report Writing
8. Validation Report

### Test Cases

For each item in the Testing Plan listed, a documented Test Case is required on how each test is performed.

The test cases and reports will be performed and populated during the Testing and Validation Phase of the project.

For the Test Case Documents, refer to the Testing Plan and Test Cases folder in the Assignment Deliverable.

# User Interface Design and Prototypes

The information includes mock-ups of the systems UI and UX.

### Main Menu UI

A screenshot of a video game

AI-generated content may be incorrect.

### Play Screen UI / UX

A screenshot of a computer

AI-generated content may be incorrect.

### Webstore & Business Analytics UI

A screenshot of a computer

AI-generated content may be incorrect.

### ARtstyle UI/UX Inspirations



Figure : Top-down Simulation Style, inspired by 'Overcooked'



Figure : Art Style Inspired by 16bit arcade games, such as 'Zelda'

# Deployment Plan

The deployment plan below outlines two scenarios for the software ‘Project RPG’, including a course based one, and a real-world scenario.

### Course Based

The status of Project RPG achieves the initial outlined objectives, which establishes a core gameplay loop that can be expanded upon to increase the scope of the game, and overall enjoyment of the game process. Deploying the project with the course’s outlined scope is achieved through the following steps:

1. **Finishing Project Test Plan**

Finishing the outlined test plan and test cases is critical to ensure that the game meets the initial established project requirements, and functions in the environment it was intended for. After the initial round of testing and regression has been completed, the game can start the next steps for deployment.

1. **Creating a Deployment Environment Object:**

Once the game is in a complete state, I (the developer) can begin registering the object on the PC gaming platform ‘Steam’. Steam is the leading global deployment platform for computer gaming, and gives developers comprehensive control over the phase, price and marketing of the product. Other deployment platforms, such as EA, Microsoft and Sony require that a game is already in an established state before deployment can begin on those platforms.

The first step in creating an object on Steam is to provide all the information required for a user to download, such as:

* + Source code
  + Launch Icons
  + Background Images
  + Game description
  + Recommended and Minimum Computer Specifications
  + Current Product Phase (Alpha, Beta, etc)
  + Pricing

With all this information provided and a registration fee of $100 USD, the game object can be uploaded to Steam for anyone to download. Once the game is available for download or purchase, the user will be able to download the game, and launch the game on the PC, assuming the specifications are met.

### Real World

The process of Indie-Game Development often spans many years and requires meticulous planning to increase the individuals’ odds of success in the game development market. The deployment plan listed below outlines these steps assuming this amount of time and planning is available to the individual.

Assuming Project RPG was budgeted multiple years of development and deployment, the deployment plan would resemble the following steps:

1. **Finishing Project Test Plan (Repeat: See Above)**
2. **Continued development for future Improvement:**

With the core gameplay loop confirmed, adding the future improvements to the game is critical to ensure the game can thrive in the game world, and provide hours or repetitive fun gameplay. For this step, I would budget an additional 2 years to implement the following features to the game:

1. Local Save Storage
2. Assortment of different gaming products (Games, consoles, controllers, etc)
3. Condition features (damaged, poor, like-new, etc)
4. Refurbishment center
5. Unlockable Items (staff, better equipment, etc)
6. Achievements
7. Bidding system for eBay store
8. Day/night cycle
9. Player Fatigue
10. Customer review system
11. Delivery options
12. **Robust Test Plan and Testing Cases:**

It will be important to repeat the testing plan outlined above, but with extensive user feedback criteria; extending tens, if not hundreds of players.

1. **Creating a Deployment Environment Object (Repeat: see above)**
2. **Attending conferences**

Marketing is critical for game development. Most starting indie game developers do not have the funds to financially market their product heavily, so it is important to attend game conferences and shows to gain a following for your product.

1. **Continued Support and Development**

It is important in the indie game market to continue supporting the game throughout the games lifecycle. This can help prevent negative feedback, as well as continue to generate revenue throughout the course of the game. Having a robust and functioning product can propel an indie-game developer to develop more games and establish a cumulative revenue stream.

# Version Control History

Project RPG is revision controlled through Git. Details on the Github storage are outlined in the Source Code section above.

# Weekly Progress Reports

Weekly Progress Reports are submitted on Tuesdays or Wednesdays and maintained through the Canvas Assignment (example below)

A screenshot of a computer

AI-generated content may be incorrect.

Figure : Example of Weekly Status Report

# Integration & Testing Report

### Modules

Project RPG is build using all existing Unity and C# packages, and there is no custom or non-traditional API packages used to achieve the projects final objectives. The traditional Unity specific APIs all function as expected and are listed below and in the Class Structures section of the Planning document. See below for details on debugging:

* **InputActionReference**: Traditional movement function that assigns traditional movement keys, such as W,A,S & D.
* **RigidBody2d**: Traditional Feature used to add gravity and collision detection to objects
* **Sprite**: Code object used for assigning images to objects (players, objects, environment)
* **SpriteRenderer**: Used for updating sprites during function changes
* **Animator**: Used for showing moving sprites while a function is true or false.

### Testing & Debugging

Throughout development, various bugs were produced and resolved prior to this phase of the project submission:

1. Player X-axis flipping
   1. When changing direction, the player would disappear or not turn
2. Product / Box stage
   1. This was the largest development piece
   2. Adding various stages and conditions would often break one of the stage update features, including selling status, and stage confusion
3. Revenue
   1. Adding new business features, such as sell price, purchase price, and stock sales contributed to various mathematical bugs that had to be ironed out.