SOFTWARE REQUIREMENTS SPECIFICATION

for

Choose Your Own Adventure

Prepared by
Jack Carr
Rich Williams

Hood College Computer Science
CS475 Spring

February 9, 2023

Contents

Revision History

Name	Date	Reason For Changes	Version
Rich Williams	4 Feb 2023	Initial Edit	Version 1.0
Rich Williams	7 Feb 2023	Edit to Software Requirements and document format	Version 1.1

1 Introduction

1.1 Purpose

The software requirements specification for CYOA is intended to provide a detailed explanation of the features and design specifications for this application. Additionally, it will serve as a tool to communicate the design plans and intentions of the team.

1.2 Document Conventions

This Document will be published with the current date on the cover page, with updates made as required, or requested by the client or development team.

1.3 Intended Audience and Reading Suggestions

This document is intended for CS475 Professor, and classmates. our team, and anyone else interested in learning more about the CYOA application. The remainder of this document provides the general project description and a technical outline for the requirements of this system. Section two will give a high level description of the project functionality and implementation details. Section three will give a more detailed description of the specific requirements for different components of the application, which include various interfaces and functional requirements.

1.4 Project Scope

A simple end to end user experience relayed in one event.

1.5 References

Python: high-level programming language PYGame: a high-level Python Web framework CYOA: Choose Your Own Adventure

2 Overall Description

2.1 Project Perspective

The project is meant for a younger audience meaning that it is targeted towards a younger audience, but anyone can play it. The game is meant to be played for fun. The project needs to be fun and the story that is being told needs to be understandable.

2.2 Project Functions

TBD

2.3 User Classes and Characteristics

A general user will be considered to be capable of general operation of a computer and peripherals. The operation of this game will not require any use of advanced computing skills.

2.4 Operating Environment

There is one starting point. Everyone will start the same. After the player chooses which path they branch out to, the game will give a random scenario to make each story unique in its own way. There is no save points to this game. The game will not remember what the user has done if the game is closed at any point. Each time the game is booted up, the game starts at the beginning. The game will have a graphical interface similar to a visual novel. (Possibility), at the end of the game, there is a summary of the path you took to get to a particular ending. If a player fails a puzzle, the game will not freeze but will guide the user to a different path than if they completed the puzzle. This path will be treated as a "bad" path that will most likely end the user's story. Interactions with the game is with using keys on the keyboard given a prompt or using the mouse to click a button on a screen to proceed to the next event. The game takes place in one genre (fantasy).

2.5 Design and Implementation Constraints

2.6 User Documentation

TBD

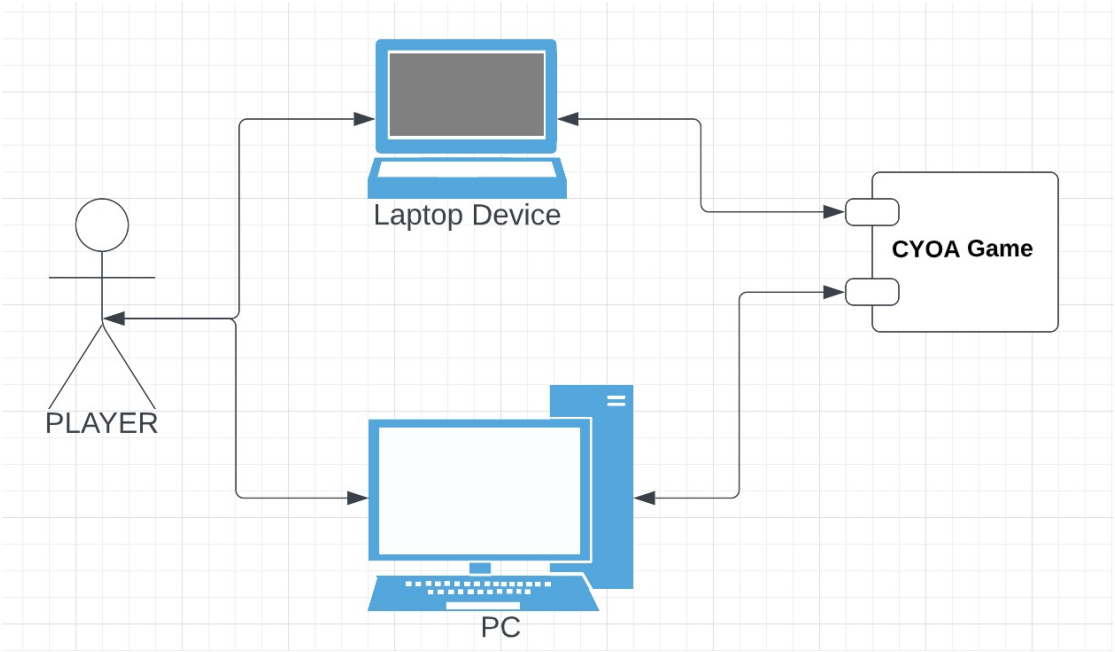

Figure 2.1: Operating Environment

2.7 Assumptions and Dependencies

2.7.1

Assumptions: The game is single player only. The game is meant for a younger audience, but anyone can play it. The game will have a graphical interface similar to a visual novel. The game takes place in one genre (fantasy). Interactions with the game are done with keys on the keyboard or by clicking buttons on the screen. The the minimum technical requirements for playing the game are intended to match current off the shelf computing equipment with backward capability towards older machines.

2.7.2

Dependencies: The design and implementation of the game is dependent on the technology used for creating the graphical interface. The schedule for the project is dependent on the development team's ability to complete the tasks on time.

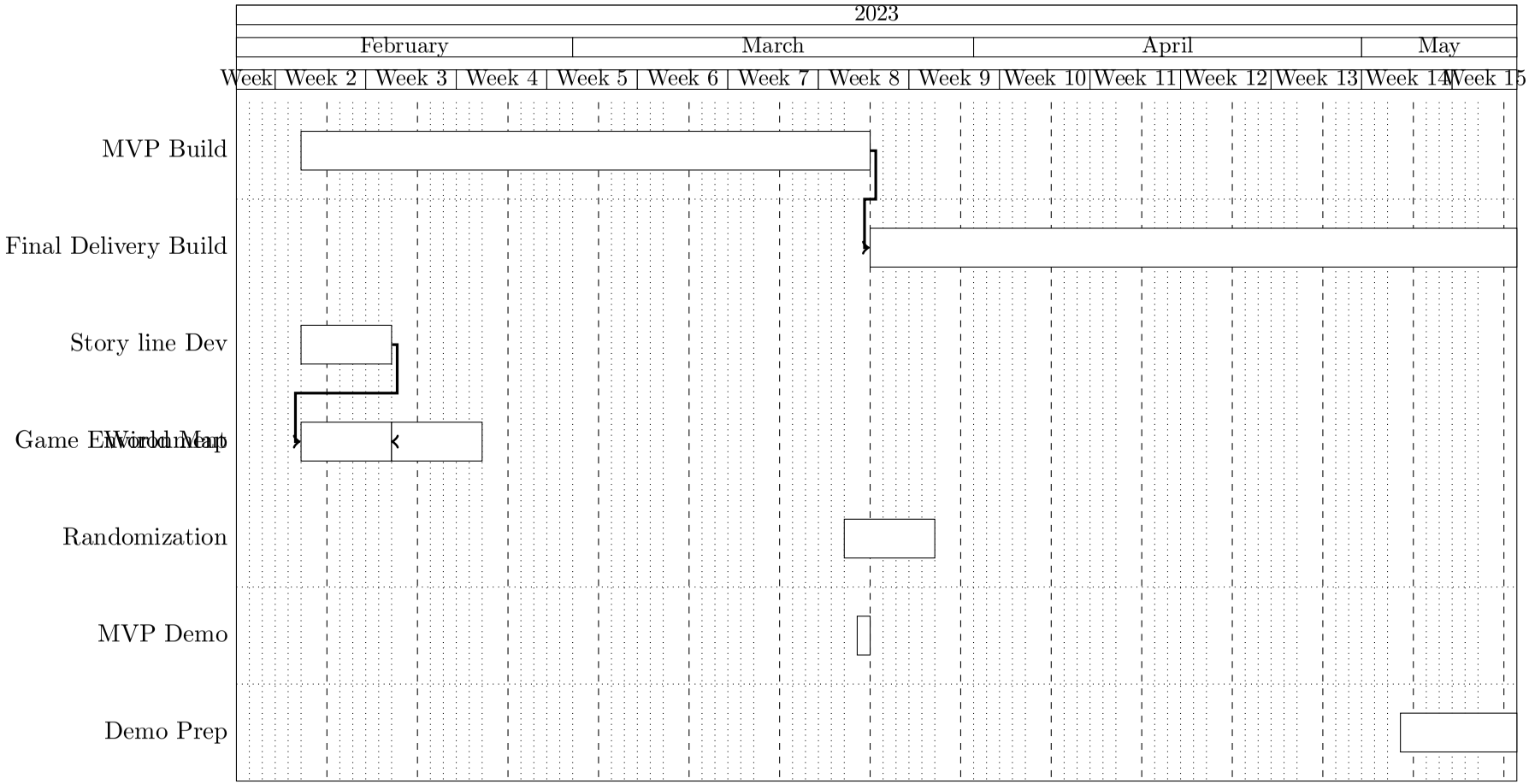

Figure 2.2: 14 Week Timeline.

3 External Interface Requirements

3.1 User Interfaces

A Computer with Standard User interface of Mouse and Keyboard.

3.2 Hardware Interfaces

See 3.1

3.3 Software Interfaces

Python 3.10 and The PyGame Library.

3.4 Communications Interfaces

No outside communications will be entertained.

4 System Features

TBD

5 Other Nonfunctional Requirements

5.1 Performance Requirements

TBD

5.2 Safety Requirements

TBD

5.3 Security Requirements

None expected as system handles no user logins or passwords.

5.4 Software Quality Attributes

The system handles no user logins or passwords. %sectionBusiness Rules

6 Other Requirements

6.1 Appendix A: Glossary

TBD

6.2 Appendix B: Analysis Models

Not Utilized

6.3 Appendix C: To Be Determined List

Project Functions User Documentation System Features Performance Requirements Safety Requirements Glossary

6.4 Appendix D: General over view of tasks * Indicates a Minimum Viable Product (MVP)Item.

6.4.1 *

J.C
Plan the story: Start by writing a story and mapping out the different paths that the player can take. Determine the number of scenes, characters, and choices the player will have. Also, plan the biomes of Earth that the player will be able to visit.

6.4.2 *

R.W
Set up the game window: Use pygame's display module to set up the game window. This will include setting the screen size, creating a window title, and setting the background color.

6.4.3 *

J.C
Find images and audio: Load the images and audio files the game will use in the game using the pygame.image and pygame.mixer modules.

6.4.4 *

R.W
Create a clickable world map: Use the pygame.draw module to create a clickable world map that displays the different biomes of Earth. Use the pygame.mouse module to handle mouse clicks and start the player's adventure in the chosen biome.

6.4.5 *

R.W
Implement randomization: Add programmatic randomization to the game so that no single play through is the same. Use python's random module to generate random events, choices, and scenes.

6.4.6 *

J.C R.W
Create the game loop: Write a game loop that will handle the main logic of the game. This will include displaying the text for each scene, handling player choices, and moving between scenes based on player choices.

6.4.7 *

J.C
Add characters and dialogue: Use the pygame.font module to display dialogue and character images on the screen.

6.4.8 *

R.W
Implement player choices: Use the pygame.event module to handle player input and implement the different choices available to the player.

6.4.9

J.C R.W
Implement mini games: Create a GUI-based tic-tac-toe game and a pong game using pygame. These mini games can be integrated into the main game loop and triggered at various points in the player's adventure.

6.4.10

J.C R.W
Test and debug: Test the game and debug any errors encountered,use pygame.error to display error messages and help diagnose the problem.

6.4.11

J.C R.W
Finalize and present game: on completion of development, and full player testing has been accomplished, finalize and present it.