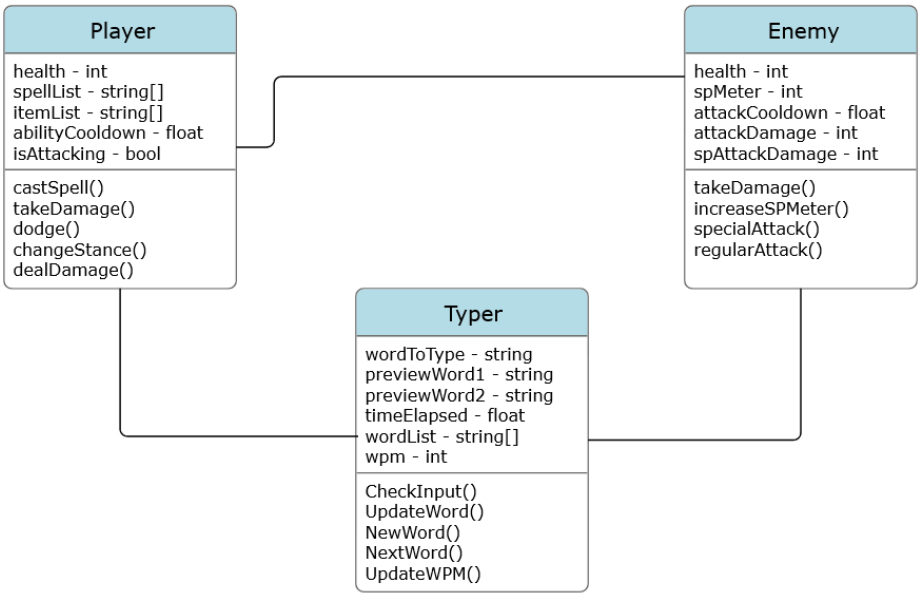
5.1 Requirements Introduction

Keyboard Warrior is a dynamic typing learning program based on the Unity game engine, consisting of a C# application which will be displayed.

The application consists of a top portion of the screen which displays a graphic for the visual representation of “fighting” within the game, text indicating the words which shall be typed, an entry field into which text matching the words to type, and sections of text containing the list of spells and items currently obtained. Spell checks are performed each time the user types any character, including letters, numbers, and punctuation.



The remainder of this document is structured as follows: Section 5.2 contains the list of features that the completed application can be expected to have, Section 5.3 contains the list of performance requirements that exists for the completed application, and Section 5.4 contains the list of software, hardware, and other resources needed for the development/deployment/execution of the application.

5.2 Functional Requirements

The completed system can be expected to have a functioning Graphical User Interface (GUI), player functionality, enemy functionality, and gamestate functionality.

5.2.1 Graphical User Interface

The Graphical User Interface for the application provides the user with a way to visualize the fight and know what needs to be typed in order to both attack and defend. The GUI will also provide the user with a way to interact with functions in order to resume, pause, and restart the game.

5.2.1.1 The GUI shall provide a visual indication of the fight between the player and the enemy

5.2.1.2 The GUI shall indicate which word the player needs to type along with previews of the next words that will come afterwards

5.2.1.3 The GUI shall show popup information to the player. Pop up information will include but not be limited to player damage dealt, player damage received, enemy damage dealt, and enemy damage received.

5.2.1.4 The GUI shall constantly show player information. Information that will remain on the screen for the duration of the game will include but not be limited to health, spell list, and item list

5.2.1.5 The GUI shall show information about the current gamestate that will include but not be limited to current time elapsed and current words per minute

5.2.1.6 The GUI shall provide a menu system to start game, resume/pause, restart, and quit

5.2.1.7 The GUI shall provide feedback using different colored letters to indicate which letters in a string are typed properly

5.2.1.8 The GUI shall show popup information after combat is over. Pop up information will include but not be limited to the number of words typed and gold earned.

5.2.1.9 The GUI shall show three different paths the user can take after a fight

5.2.2 Player functionality

The Player functionality for the application provides the user with the mechanics that can be used in order to fight against the enemy in combat. Player functionality will also allow the user to choose rewards and which path they want to go in the next stage.

5.2.2.1 The player shall be able to type the specified word provided by the GUI to deal damage while in attack stance

5.2.2.2 The player shall be able to cast spells using the prefix “/” followed by the specific word

5.2.2.3 The player shall be able to switch between defending and attacking stances using the key “Tab”

5.2.2.4 The player shall be able to dodge basic enemy attacks by typing specific words while in defense stance

5.2.2.5 The player shall be able to type specific words in order to reduce the damage of an enemies special attacks

5.2.2.6 The player shall take damage upon missing the dodge window for the enemy’s basic attack

5.2.2.7 The player shall “die”/lose once their health reaches 0

5.2.3 Enemy functionality

The Enemy functionality for the application provides the mechanics that will be used in order to fight against the user/player in combat

5.2.3.1 The enemy shall “die”/lose once their health reaches 0

5.2.3.2 The enemy shall increase their Special Attack meter if their attacks hit the player

5.2.3.3 The enemy shall do a basic attack at specified time intervals

5.2.3.4 The enemy shall use their Special Attack upon maxing out their Special Attack meter

5.2.4 Gamestate functionality

The Gamestate functionality for the application provides the features in order to change game scenes upon interacting with the GUI or finishing a fight

5.2.4.1 The gamestate shall pause the game upon pressing “Esc”

5.2.4.2 The gamestate shall restart/quit/pause/resume the game upon interacting with the respective GUI

5.2.4.3 The gamestate shall load the next stage after the player interacts with the respective GUI at the end of a fight

5.2.4.4 The gamestate shall start the first fight upon starting the game from the main menu

5.3 Performance Requirements

5.3.1 Stage change time

5.3.1.1 The application shall load the game to the next stage within 5 seconds of time the user chooses which route to go following their combat

5.3.2 Pause/Resume/Restart/Quit

5.3.2.1 The application shall pause/resume the game within 1 second of time the user submits the request to pause or resume

5.3.2.2 The application shall restart the game within 5 seconds of time the users submits the request to restart the game

5.3.2.3 The application shall quit within 10 seconds of time the user submits the request to quit the game

5.4 Environment Requirements

Following are the hardware and software requirements for Keyboard Warrior:

Operating System	Windows 7 (SP1+), Windows 10 and Windows 11
CPU	X86, x64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, DX12 capable

(These are the minimum hardware and software requirements taken from the Unity Documentation)

Windows 7 minimum is required for this project. The program will be written in C# using Microsoft Visual Studio and Unity engine.