#### **Software Requirements**

#### **Specification**

#### 

#### **for**

#### 

#### 

#### **Build-The-Bot**

#### 

#### 

#### 

#### 

#### **Version 1.0**

#### 

#### 

#### **Prepared by**

#### **Julia Waggoner, Karen Hernandez, Miranda Castle, Truc Le**

#### 

#### 

#### **DePauw University**

**1. Introduction**

**1.1 Purpose**

This text adventure game is intended to take players through a series of challenges and brain teasers in order to collect pieces of a mysterious robot.

**1.2 Intended Audience and Reading Suggestions**

Anyone who is looking for a fun time, as long as that person can read and write English.

**1.3 Project Scope**

The goal of this game is to enable user to instruct the game character on his or her quest to explore a virtual space shuttle through typing in commands. The game will allow the user to become a better problem-solver.

**2. Overall Description**

**2.1 Product Perspective**

This game is being developed by students in the Objected Oriented Software Development course at DePauw University. The goal of this project is to develop an Object Oriented text adventure game.

**2.2 Product Features**

The main features of this game are:

• Reading in player’s command

• Applying the player’s command within the scope of the game

• Outputting the results of the players commands

**2.3 User Classes and Characteristics**

The intended users for this game is everyone and anyone who wants to have fun and is willing to stretch their brains to problems solve.

**2.4 Operating Environment**

This application is designed to work with a Java Virtual Machine in a desktop environment. Users of this application are expected to be running either a Windows, MacOS, or Linux desktop operating system.

**2.5 Design and Implementation Constraints**

The game will match the player’s command to the set of possible commands and respond accordingly. When player types in an invalid command, the game will prompt user to restate. Otherwise, the game carries out the command. A text-based interface will be provided for the user to interact with the game.

**2.6 User Documentation**

If the player types “help”, a help manual will appear with a list of valid commands.

**3. System Features**

The following features, with their associated requirements, will be implemented in the final revision of this software system:

**3.1 Reading in Commands**

**3.1.1 Description and Priority**

*A player can type in a command to instruct the game character to perform an action.*

Valid Commands:

• **quit**

Quit the game.

• **save**

Save the game so that player can pick up where he or she left off the next time he or she plays the game.

• **load**

Load the most recently saved game so that player can pick up from there.

• **inventory**

Show a list of objects that player has collected.

• **look**

Show the current location’s name, description, non-hidden items, and the number of moves that the player has made.

• **examine ‘item’**

Show the specified item’s name and description.

• **take ‘item’**

Take the specified item from the current location and put it in player’s inventory.

• **take ‘item’ from ‘container’**

Take the specified item from the specified container and put it in player’s inventory.

• **put ‘item’**

Take the specified item from player’s inventory, and put it in the current location.

• **put ‘item’ in ‘container’**

Take the specified item from player’s inventory, and put it in the specified container.

• **give ‘item’ to ‘character’**

Take the specified item from player’s inventory, and give it to the specified character.

• **go ‘direction’**

Go towards the specified direction.

Valid Directions:

• **front**: Go forward.

• **back**: Go backward.

• **right**: Go right.

• **left**: Go left.

**3.1.2 Functional Requirements**

REQ-1: The game must be able to understand whether the command has an effect on the game.

REQ-2: The game must offer some means of feedback for invalid commands.

REQ-3: The game must apply the command if the command is valid.

REQ-4: The game must give feedback after every move the player makes.

Feedbacks for Invalid Commands:

• Unknown command

• Object not found

• Container not found

• Receiver not found

• Cannot take this

• Can’t go that way

**3.2 Applying Commands**

**3.2.1 Description and Priority**

*The game matches the player’s command to its set of possible commands.*

**3.2.2 Functional Requirements**

REQ-1: The game must change its environment according to the command.

REQ- 2: Every command has only one effect on the game environment.

**3.3 Outputting Commands**

**3.3.1 Description and Priority**

*The game must be able to output text that describes the effects of the player’s command on the game environment.*

**3.3.2 Functional Requirements**

REQ-1: The game must know if the player has typed an invalid command and let the player know that it is invalid.

REQ-2: The game must output the correct description of the altered environment after a command is executed.

REQ-3: The game must always be able to inform the player of the current environment they are in.

Environment includes:

• Eleven unique rooms.

• Puzzles and brain-teasers that player must solve to complete the game.

• Hidden objects.

**4. External Interface Requirements**

**4.1 Hardware Interfaces**

The software will run on a desktop or laptop and no additional hardware is needed.

**4.2 Software Interfaces**

This text adventure game will run on Java Virtual Machine on either a Windows, MacOS, or Linux desktop operating system.

**5. Other Nonfunctional Requirements**

**5.1 Performance Requirements**

This application is intended to be used interactively, so users should not be expected to wait for the completion of any of the operations provided by the application.