

# SE 3XA3: Module Guide

## Zombie Survival Kit

Team #6, Group 6ix  
Mohammad Hussain hussam17  
Brian Jonatan jonatans  
Shivaansh Prasann prasanns

November 29, 2018

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Anticipated and Unlikely Changes</b>	<b>2</b>
2.1	Anticipated Changes . . . . .	2
2.2	Unlikely Changes . . . . .	2
<b>3</b>	<b>Module Hierarchy</b>	<b>3</b>
<b>4</b>	<b>Connection Between Requirements and Design</b>	<b>4</b>
<b>5</b>	<b>Module Decomposition</b>	<b>5</b>
5.1	Hardware Hiding Modules (M1) . . . . .	5
5.2	Behaviour-Hiding Module . . . . .	5
5.2.1	Interactable (Object) (M7) . . . . .	5
5.2.2	Enemy (Object) (M8) . . . . .	5
5.2.3	ItemStore (Object) (M9) . . . . .	6
5.2.4	InteractableController (Character) (M10) . . . . .	6
5.2.5	EquipmentUI (M11) . . . . .	6
5.2.6	EquipmentSlotUI (M12) . . . . .	6
5.2.7	InventoryUI (M13) . . . . .	6
5.2.8	InventorySlotUI (M14) . . . . .	7
5.2.9	CharacterCombat (Character) (M15) . . . . .	7
5.2.10	Zombie (Object) (M20) . . . . .	7
5.2.11	FirstPersonController (Character) (M21) . . . . .	7
5.2.12	Gun (Object) (M22) . . . . .	8
5.2.13	BulletDamage (Objects) (M23) . . . . .	8
5.3	Software Decision Module . . . . .	8
5.3.1	Item (Component) (M2) . . . . .	8
5.3.2	ConsumableItem (Component) (M3) . . . . .	9
5.3.3	EquipmentItem (Component) (M4) . . . . .	9
5.3.4	EquipmentManager (Manager) (M5) . . . . .	9
5.3.5	InventoryManager (Manager) (M6) . . . . .	9
5.3.6	CharacterStats (Manager) (M16) . . . . .	9
5.3.7	PlayerStats (Manager) (M17) . . . . .	10
5.3.8	ZombieStats (Manager) (M18) . . . . .	10
5.3.9	Stat (Component) (M19) . . . . .	10
<b>6</b>	<b>Traceability Matrix</b>	<b>11</b>
<b>7</b>	<b>Use Hierarchy Between Modules</b>	<b>13</b>

## List of Tables

1	<b>Revision History</b>	ii
2	Module Hierarchy	4
3	Trace Between Requirements and Modules	11
4	Trace Between Anticipated Changes and Modules	11
5	Use hierarchy among modules in table format for Figure 1	14

## List of Figures

1	Use hierarchy among modules	13
---	-----------------------------	----

Table 1: **Revision History**

Date	Version	Notes
11/08/2018	1.0	Added initial content and uses hierarchy figure
11/09/2018	1.1	Mohammad - Added my modules and finished req-module conneciton

# 1 Introduction

Zombie Survival Kit is a template for game developers to use as a starting ground for their own implementation of a First-Person Zombie Survival Shooter game. This project includes various mechanics that can be customized by game developers according to their own preferences.

The purpose of this module guide is to provide an overview of the various modules that have been implemented in this project and to serve as a reference document to help identify possible changes and facilitate the 'design for change' principle of our project. This document also serves as a starting point to help isolate any errors or bugs that may be encountered in the future.

This document includes a list of all the modules that have currently been implemented in our project, a Uses hierarchy for all modules, all the dependencies between various modules, as well as certain changes that may or may not be implemented in future versions of the project.

## 2 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 2.1, and unlikely changes are listed in Section 2.2.

### 2.1 Anticipated Changes

**AC1:** The specific hardware on which the software is running.

**AC2:** The animations used for firing weapons in the game.

**AC3:** The sounds used for firing and reloading weapons in the game.

**AC4:** The rate of firing for the gun in the game.

**AC5:** The values of health and damage for various characters in the game.

### 2.2 Unlikely Changes

**UC1:** Input/Output devices (Input: File, Keyboard, and Mouse, Output: File, Memory, and Screen).

**UC2:** There will always be a source of input data external to the software.

**UC3:** The Unity API may be changed/updated.

### 3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 2. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

- M1:** Hardware-Hiding Module
- M2:** Item (Component)
- M3:** ConsumableItem (Component)
- M4:** EquipmentItem (Component)
- M5:** EquipmentManager (Manager)
- M6:** InventoryManager (Manager)
- M7:** Interactable (Object)
- M8:** Enemy (Object)
- M9:** ItemStore (Object)
- M10:** InteractableController (Character)
- M11:** EquipmentUI
- M12:** EquipmentSlotUI
- M13:** InventoryUI
- M14:** InventorySlotUI
- M15:** CharacterCombat (Character)
- M16:** CharacterStats (Manager)
- M17:** PlayerStats (Manager)
- M18:** ZombieStats (Manager)
- M19:** Stat (Component)
- M20:** Zombie (Object)
- M21:** FirstPersonController (Character)
- M22:** Gun (Object)
- M23:** BulletDamage (Objects)

Level 1	Level 2	Level 3
Hardware-Hiding Module	M1	
	M7	
	M10	
	M11	
	M12	
	M13	
Behaviour-Hiding Module	M14	M8, M9 (Inherits M7)
	M15	
	M20	
	M21	
	M22	
	M23	
Software Decision Module	M2	M3, M4 (Inherits M2)
	M5	
	M6	
	M16	M17, M18 (Inherits M16)
	M19	

Table 2: Module Hierarchy

## 4 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 3.

## 5 Module Decomposition

### 5.1 Hardware Hiding Modules (M1)

**Secrets:** The data structure and algorithm used to implement the virtual hardware.

**Services:** Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

**Implemented By:** OS

### 5.2 Behaviour-Hiding Module

**Secrets:** The contents of the required behaviours.

**Services:** Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

**Implemented By:** –

#### 5.2.1 Interactable (Object) (M7)

**Secrets:** Component on an instantiated Game object. Used to differentiate which Game objects can be interacted with by the player.

**Services:** If a Game object with M7 as a component receives a valid user input, it will produce some output.

**Implemented By:** Interactable.cs

#### 5.2.2 Enemy (Object) (M8)

**Secrets:** Component on an instantiated Game object; inherits M7. Used to differentiate different enemies that the player can attack.

**Services:** If a Game object with M8 as a component receives a valid user input, it will store attack the enemy.

**Implemented By:** Enemy.cs



### 5.2.3 ItemStore (Object) (M9)

**Secrets:** Component on an instantiated Game object; inherits M7. Used to differentiate which Game objects can be stored in the player's inventory.

**Services:** If a Game object with M9 as a component receives a valid user input, it will store the item in the player's inventory.

**Implemented By:** ItemStore.cs

### 5.2.4 InteractableController (Character) (M10)

**Secrets:** Allows the user to instruct the player to interact with Game objects with M7 or M9 as components through keyboard input.

**Services:** Converts the keyboard input into an output dictated by M7 or M9.

**Implemented By:** InteractableController.cs

### 5.2.5 EquipmentUI (M11)

**Secrets:** Input is determined by any changes that occur in the inventory (i.e. if an item is added or removed from the inventory).

**Services:** Converts the changes in the inventory to be displayed in the equipment UI.

**Implemented By:** EquipmentSlot.cs

### 5.2.6 EquipmentSlotUI (M12)

**Secrets:** Input is determined by user input through the mouse. Acquires user input if the user clicks on the remove button on any slot in the equipment UI.

**Services:** Once user input is acquired, the equipment item equipped moves back into the player's inventory and both the Equipment UI and Inventory UI updates accordingly. input parameters module.

**Implemented By:** EquipmentSlot.cs

### 5.2.7 InventoryUI (M13)

**Secrets:** Input is determined by any changes that occur in the inventory (If an item is added or removed from the inventory)

**Services:** The input converts the changes in the inventory to be displayed in the inventory UI.

**Implemented By:** InventoryUI.cs

### 5.2.8 InventorySlotUI (M14)

**Secrets:** Input is determined by user input through the mouse. Acquires user input if the user clicks on the remove button or the icon button on any slot in the inventory UI.

**Services:** If an input on the remove button is acquired, the item in the inventory is removed from the inventory and the Inventory UI is updated accordingly. If an input on the icon button is acquired, the item is used in some way (depending on the item) and the inventory UI is updated (E.g. If the item is a consumable item, the item's healthModifier will be added to the player's current health; if the item is an equipment item, the item will be equipped, the player's stats will be updated through the attackModifier and defenceModifier, and the equipment UI will be updated).

**Implemented By:** InventorySlot.cs

### 5.2.9 CharacterCombat (Character) (M15)

**Secrets:** Serves as a component on an instantiated Player or Zombie game object. Used to apply damage from one character to another.

**Services:** If the attack cooldown reaches 0, the Attack() method is called to apply damage from the attacker to the target and the cooldown is reset.

**Implemented By:** CharacterCombat.cs

### 5.2.10 Zombie (Object) (M20)

**Secrets:** Serves as a component on an instantiated Zombie game object. Used to define the enemy's movement and attacking behaviour.

**Services:** In passive state, the enemy wanders around in random directions and switches to attack state if the player gets too close. In attack state, the zombie follows and attacks the player and returns back to spawn if it runs too far.

**Implemented By:** Zombie.cs

### 5.2.11 FirstPersonController (Character) (M21)

**Secrets:** Serves as a component on the instantiated Player game object. Used to control player movement and camera.

**Services:** -

**Implemented By:** Unity Standard Asset Scripts

### 5.2.12 Gun (Object) (M22)

**Secrets:** Input is determined by user input through the mouse or the keyboard. Acquires user input if the user clicks on the left mouse button or the R button on the keyboard when a gun is equipped.

**Services:** If an input through the left mouse button is acquired, the gun either fires a bullet or reloads itself (depending on the number of bullets in the clip) and the corresponding sound effect is played. If an input on the R button is acquired, the gun is reloaded.

**Implemented By:** Gun.cs

### 5.2.13 BulletDamage (Objects) (M23)

**Secrets:** Serves as a component on an instantiated Bullet GameObject. Used to detect collisions with other objects in the game environment and take corresponding actions.

**Services:** If a Bullet GameObject collides with an object having a Collider component attached, the module checks if the object is an Enemy. If yes, then the Enemy takes some amount of damage and the bullet gets destroyed. In all other cases, the bullet gets destroyed

**Implemented By:** BulletDamage.cs

## 5.3 Software Decision Module

**Secrets:** The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

**Services:** Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

**Implemented By:** –

### 5.3.1 Item (Component) (M2)

**Secrets:** Represents the basis of all items that can be used by the player; holds the name of the item, and the sprite icon *not* described in the SRS.

**Services:** -

**Implemented By:** Item.cs

### 5.3.2 ConsumableItem (Component) (M3)

**Secrets:** Represents the basis of all items classified as a consumable that can be "eaten" by the player; holds the "healthModifier" (integer) that, when the consumable item is "eaten", modifies the player's health by this amount. Inherits M2.

**Services:** -

**Implemented By:** ConsumableItem.cs

### 5.3.3 EquipmentItem (Component) (M4)

**Secrets:** Represents the basis of all items classified as an equipment that can be equipped by the player; holds the "attackModifier" and "defenceModifier" that, when the equipment is equipped, changes the attack and defence stats of the player by these amounts. It also holds "equipSlot" that determines which slot this equipment item belongs to when equipped. Inherits M2.

**Services:** Includes an enum class called equipSlot that consists of {Head, Chest, Legs, Primaryhand, Offhand, Feet}. Each instance represents the different areas in which equipment items can be equipped to.

**Implemented By:** EquipmentItem.cs

### 5.3.4 EquipmentManager (Manager) (M5)

**Secrets:** Manages any equipment items that are equipped.

**Services:** Has an array used to hold any equipped equipment items.

**Implemented By:** EquipmentManager.cs

### 5.3.5 InventoryManager (Manager) (M6)

**Secrets:** Manages any items in the player's inventory.

**Services:** Contains a list that stores all the items in the inventory.

**Implemented By:** Inventory.cs

### 5.3.6 CharacterStats (Manager) (M16)

**Secrets:** Holds damage and armour stats for any character (player or zombies) and the method to take damage and remove from the current health.

**Services:** Includes two stat variables that track the damage and armour of the character.

**Implemented By:** CharacterStats.cs

### 5.3.7 PlayerStats (Manager) (M17)

**Secrets:** Manages modifiers from consumable and equipment items to be applied and removed from the player. Inherits M16.

**Services:** -

**Implemented By:** PlayerStats.cs

### 5.3.8 ZombieStats (Manager) (M18)

**Secrets:** Manages events that occur when the zombie dies. Inherits M16.

**Services:** -

**Implemented By:** ZombieStats.cs

### 5.3.9 Stat (Component) (M19)

**Secrets:** Manages initial values and modifiers to a stat.

**Services:** Has a list of integers to hold each added armour or damage modifier

**Implemented By:** Stat.cs

## 6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
R1	M21
R2	M21
R3	M20
R4	M20
R5	M20
R6	M20
R7	M20, M16
R8	M2, M3, M4, M18
R9	M2, M3, M4, M18
R10	M7, M9, M10
R11	M13
R12	M2, M3, M4, M5, M6, M11, M12, M13, M14
R13	M8, M10, M22
R14	Not yet implemented.
R15	Not yet implemented.
R16	M15, M16
R17	M8, M15, M16, M23
R18	M16
R19	M17

Table 3: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M22
AC3	M22
AC4	M22
AC5	M16

Table 4: Trace Between Anticipated Changes and Modules



## 7 Use Hierarchy Between Modules

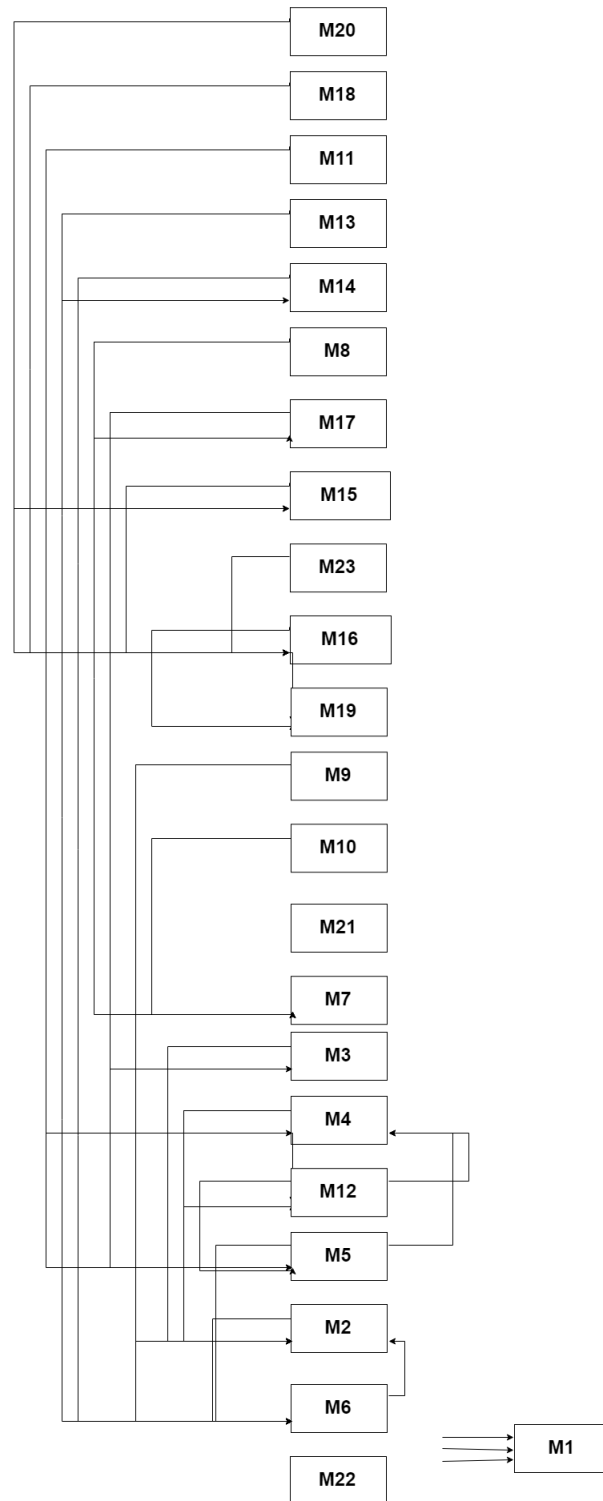


Figure 1: Use hierarchy among modules



Modules	Uses
M1	
M2	M6
M3	M2, M6
M4	M2, M5, M12
M5	M6, M4
M6	M2
M7	
M8	M15, M16, M17, M7
M9	M2, M7, M6
M10	M7
M11	M5, M12
M12	M4, M5
M13	M6, M9
M14	M6
M15	M16
M16	M19
M17	M16, M19, M5, M3, M4
M18	M16
M19	
M20	M15, M16
M21	
M22	
M23	M18

Table 5: Use hierarchy among modules in table format for Figure 1