

Test Plan Report

Nam Le - jssb25

Charles Dubois-Veltman -
rklb65

Joshua Pulham - tgbp44

James Watson - nqcv52

Sam Ezeh - vqqw43

Akanksha Sirohia -
nwrz52

Project Summary

Our project is to reimagine Skills Build as an RPG, which facilitates accessing IBM Skills Build courses by linking to them throughout the game and having sections in our game have questions from the IBM Skills Build website where the user is rewarded for answering them correctly. We are doing this by making a 2D educational RPG game where there are puzzles for the user to complete and bosses which the user must defeat by answering questions from the IBM Skills Build website [1]. This is outlined in greater detail in our Requirement Specification [2], which we also reference to throughout this document (when specifying which requirements each test relates to).

Document Structure

This report outlines the plan for our tests and is divided into three main sections.

1. The **test overview**, in which we list all our unit tests, integration tests, system tests and user acceptance tests. We also describe our plan for integration testing. It also includes a diagram of our test workflow, which summarises our plan concisely. In this section, we show our testing workflow, and also explain our rationale for types of tests conducted (black box vs white box).
2. The **test case** section outlines some test cases in detail, covering unit tests, system tests and user acceptance tests. Note that we do not include the integration tests in this section.
3. The **testing context** gives additional background information about the tests. This includes the development workflow, our choice of testing environment and our choice of classes of severity. This section will not only outline these choices but also explain the reasoning and provide justification for these choices.
4. The **testing outcomes** show a list of successful tests that we have conducted so far.
5. The **user acceptance survey and questionnaire** show the document that will be used to conduct our UAT and retrieve some user feedback.

Tests Overview

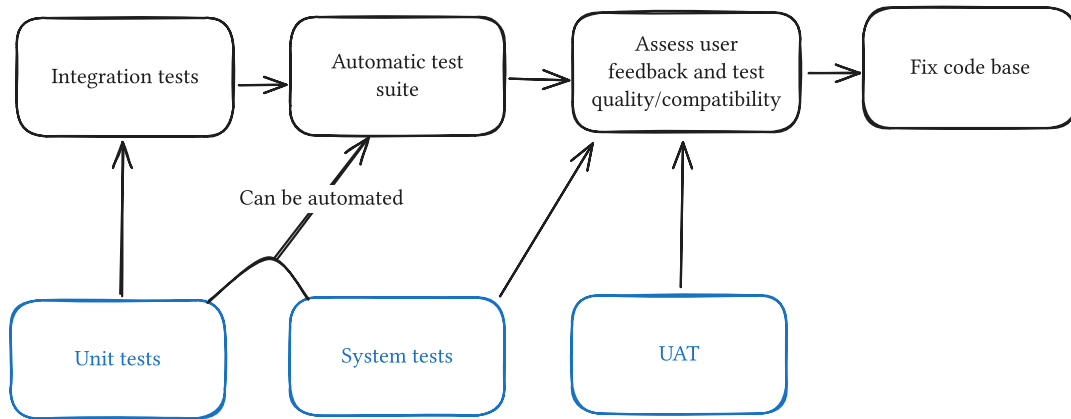


Figure 1: Testing workflow

For our tests, we will use the above structure Figure 1. All unit [3], integration [4] and some systems tests can be automatically ran and will therefore be included in our automatic test suite. After conducting the tests, we will do a review of failed tests (of all nature), and fix the code base accordingly from the feedback.

Below are four lists of tests, representing our unit tests, integration tests, systems tests and user acceptance tests. In each of these lists (besides integration tests), we pick 6 tests, which will be outlined in greater detail in the subsequent section.

For now, all our tests planned are black box tests. This is because most units are still a work in progress, which means we do not know their internal structure. Without knowing this internal structure, it is impossible to do or plan white box testing, as white box testing “cannot test expected functionality that does not exist in the codebase” [5]. Once we have completed more of the project, we will add white box testing, specifically unit, and integration white box tests.

Unit Tests

We have designed these tests with triggers and events such that they can easily be automated, and since these are built-in Unity functionality [6]. All unit tests are modelled after functional requirements (FRs).

Test ID	Description	Oracle
unit_test-01	Player movement	Player moves in expected direction on up/down/left/right movement event trigger
unit_test-02	Camera on/off logic	Camera activates/deactivates depending on character proximity check
unit_test-03	Checkpoint save on reset	Player reset to last checkpoint on death/fail event
unit_test-04	Scene transition	Scene transition zone loads the current next scene on entrance
unit_test-05	Accessible inventory	Inventory opens on associated button trigger
unit_test-06	New game file generation	New game file generates and overwrites selected save on new game trigger
unit_test-07	Accessible pause menu	Pause menu opens on associated button trigger
unit_test-08	Worlds progression	Player progresses onto the next world on boss defeated trigger
unit_test-09	Game exit	Main menu opens on quit button trigger in pause menu
unit_test-10	Close game	Game closes on exit game button trigger in main menu
unit_test-11	Collisions	Player remains in place on attempted movement event trigger onto wall/object tile
unit_test-12	Skill Tree	Skill tree node unlocks permanently on unlock trigger
unit_test-13	Combat damage logic	Entity loses health proportional to damage value on damage event trigger
unit_test-14	Random multiple Choice Questions	Questions appear in different order on generation request

Integration Testing Overview

Our project has many components which combine to make the final product; therefore we have judged it appropriate to carry out integration tests [7] as to make sure that the modules not only work on their own but also work in conjunction with each other.

The main strategies for integration testing are big-bang, bottom-up, top-down and hybrid [8]. Out of all of these, we have chosen to use bottom-up testing for the following reasons:

- **Phased approach:** A bottom-up approach allows us to start integration testing before all components are ready.
- **Helps identify the source of problems efficiently:** If a combination of tests is not working, we know that one of its components is at fault.
- **Compatibility with Agile:** The incremental nature of bottom-up testing fits with our agile [7] development approach.
- **Parallel development:** We can each work on the base components separately and independently.
- **Drivers vs Stubs:** Writing drivers is usually more simple than writing stubs.

Integration Tests

Test ID	Description	Oracle
int_test-01	Questions and health	Player loses health when question answered incorrectly, boss loses health when question answered correctly
int_test-02	Questions appear from database in game	Multiple choice questions shown in game are correctly retrieved from SQLite database
int_test-03	Skill Tree and player progress	Nodes unlocked in the skill tree are in accordance with the progress of the player
int_test-04	Skill tree and stats	Nodes unlocked in the skill tree affect player object stats
int_test-05	Stats and combat	Stats from unlocked nodes have the intended effect on combat
int_test-06	Pausing and game events	Pause menu, while opened, stops all animations currently playing and ongoing game events. No actions should take place in the game world in this state

Below is a diagram showing how all of these integration tests fit together with our bottom-up approach:

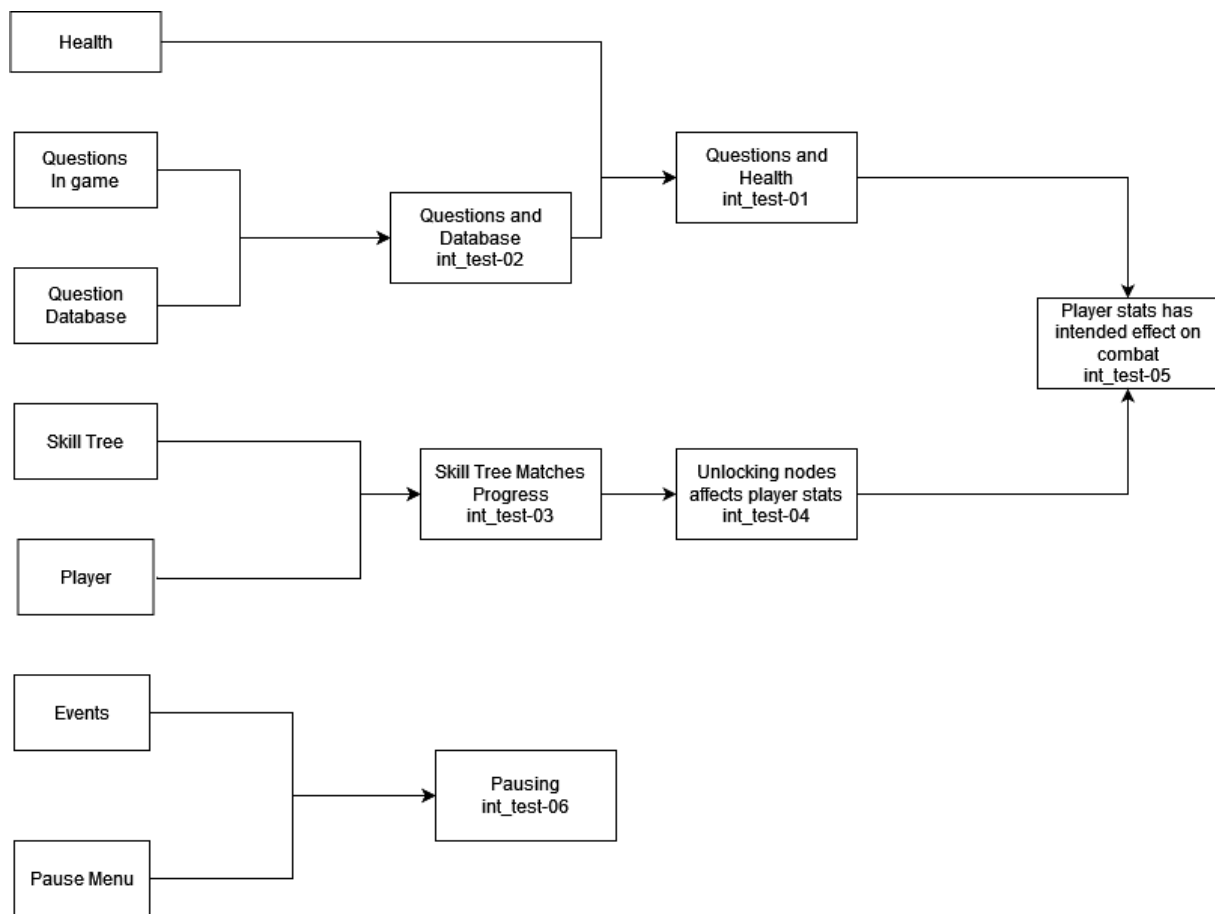


Figure 2: Integration Test Diagram

System Tests

We also highlight for each test whether it corresponds to a functional or non-functional requirement, as our system tests are directly linked to the requirement specification. Note that since the Requirement Specification is a living document, we updated some sections after the initial feedback, which included the addition of some NFRs, such as NFR3.8 - Stable FPS.

Test ID	Description	Oracle	FR/ NFR
sys_test-01	Main menu	There should be a main menu with a title and background from which the user can start a new game, change options such as sound, continue a previous game by pressing buttons	FR
sys_test-02	Skill tree / progression	There should be an aspect of progression for the player where the player can unlock nodes in a skill tree to upgrade their stats throughout the game based on Skills Build courses	FR
sys_test-03	Controls	Game should be playable on keyboard and controllers such as Xbox One & Xbox 360 controller, Nintendo Switch Pro controller and PS4/PS5 controllers	FR
sys_test-04	Frame rate	The frame rate should be 60FPS or more with minimum 20 moving object in a scene	NFR
sys_test-05	Pause menu Settings	Setting such as audio level and controls can be changed from the pause menu	FR
sys_test-06	Room Transition Delay	Time taken to switch between any 2 room should be less than 3 seconds	NFR
sys_test-07	Auto Saving	Auto saves should occur automatically throughout the game	FR

User Acceptance Tests

All user acceptance tests are modelled after non-functional requirements (NFRs). The way we are approaching user acceptance tests is benchmark testing. Our user acceptance tests are as follows:

Test ID	Description	Oracle
uat_test-01	Response time	All events occur within 10 milliseconds
uat_test-02	Movement Smoothness	User is satisfied with the smoothness of the movement
uat_test-03	Accessibility	All users thoroughly understand the gameplay elements
uat_test-04	Fair Combat	Combat is fair, with bosses, player health and question difficulty reflecting their progress in the game
ua_test-05	Impactful Abilities	users feel that the skills tree abilities are meaningful in combat
uat_test-06	Visual Design	Visual Design of game is appealing
uat_test-07	Puzzle Fairness	Puzzles are fun and solvable by all users
uat_test-08	Distinct Worlds	Worlds are distinct from one another and can be identified by the user by their design alone
uat_test-09	Soundtrack	Music in game is appealing and fits each area
uat_test-10	Question Design	Questions are built into the game well and don't impact the flow of the game
uat_test-11	Graphics Accessibility	Graphics abide by the Game Accessibility Guidelines, so the game is enjoyable for users suffering from colour blindness

Test Cases

Unit Tests

Test Case ID	unit_test-01
Description of test	Tests that player object position is updated given movement events in correct directions
Related requirement specification	FR1.1 FR1.2
Pre-requisite for test	<ul style="list-style-type: none">• Player object
Test procedure	<ol style="list-style-type: none">1. Get player object x-y coordinate2. Move player object with movement events
Test material used	<ul style="list-style-type: none">• Debug Console• Unity Test Framework
Expected result (test oracle)	<ul style="list-style-type: none">• The player object should be moved by the correct amount and in the correct direction of the event <pre>FUNCTION test_position_down_update() player = initialize_player() initial_X = player.get_X() initial_Y = player.get_Y() move_player_down_events(5) updated_X = player.get_X() updated_Y = player.get_Y() ASSERT updated_X EQUAL TO initial_X ASSERT updated_Y EQUAL TO initial_Y - 5 END FUNCTION</pre> <ul style="list-style-type: none">• Using the same structure, check movement in the other cardinal directions and assert accordingly
Comments	Movement consists of 4 directions: Up, down, left, right, as this is how movement is implemented in classic RPGs
Created by	JW
Test environment	Windows 10/11, Unity test environment
Failure Severity	Catastrophic

Test Case ID	unit_test-02
Description of test	Tests that camera is enabled when placeholder object is in range and disabled otherwise
Related requirement specification	FR1.2 FR2.5
Pre-requisite for test	<ul style="list-style-type: none"> • Camera object
Test procedure	<ol style="list-style-type: none"> 1. Place player object in camera object range 2. Remove player object from camera object range 3. Move player object back in camera object range
Test material used	<ul style="list-style-type: none"> • Debug Console • Unity Test Framework
Expected result (test oracle)	<ul style="list-style-type: none"> • Any camera should activate when the placeholder object is in its range, and deactivate if they are outside of it <pre> FUNCTION test_camera_deactivation() // Create camera with radius 50, 50 at (0,0) camera = initialize_camera(0, 0, 50, 50) placeholder = initialize_placeholder(0, 0) ASSERT camera.is_active() // Move placeholder outside of camera range move_placeholder_up_event(51) ASSERT NOT camera.isActive() END FUNCTION </pre> <ul style="list-style-type: none"> • Using the same structure, make another test for when re-entering the camera range
Comments	Functionality should be implemented in a way that allows easy switching between placeholder object and player object
Created by	CDV
Test environment	Windows 10/11, Unity test environment
Failure Severity	Catastrophic

Test Case ID	unit_test-03
Description of test	Tests that game state is reset to last checkpoint on event trigger
Related requirement specification	FR1.3 FR2.3
Pre-requisite for test	<ul style="list-style-type: none"> • Save System
Test procedure	<ol style="list-style-type: none"> 1. Create a save to simulate reaching a checkpoint 2. Trigger reset event
Test material used	<ul style="list-style-type: none"> • Debug Console • Unity Test Framework
Expected result (test oracle)	<ul style="list-style-type: none"> • Save file at checkpoint should be the same as save file after doing a sequence of actions and reloading <pre> FUNCTION test_checkpoint_activation() saver = initialize_saver() placeholder = initialize_placeholder(0, 0) save_file_1 = saver.save_state('save_1.txt') // Simulate events, then load checkpoint move_placeholder_up_event(5) load_checkpoint() save_file_2 = saver.save_state('save_2.txt') // Saves should be equal as // we are storing the game state ASSERT save_file_1 EQUALS TO save_file_2 END FUNCTION </pre>
Comments	In this context, the event trigger should be a death (health reaching 0) or closing the game, but this does not need to be implemented for this test, as a placeholder event can be used
Created by	CDV
Test environment	Windows 10/11, Unity test environment
Failure Severity	Catastrophic

Test Case ID	unit_test-04
Description of test	Tests that entering a scene transition loads the intended next scene
Related requirement specification	FR2.5 FR4.3 NFR3.4 NFR3.7
Pre-requisite for test	<ul style="list-style-type: none"> Two placeholder scenes
Test procedure	<ol style="list-style-type: none"> Load Scene 1 Trigger next scene loader
Test material used	<ul style="list-style-type: none"> Debug Console Unity Test Framework
Expected result (test oracle)	<ul style="list-style-type: none"> Correct scene (in this case, Scene 2) gets loaded after scene transitioning event called <pre> FUNCTION test_scene_1_update() // Create two scenes scene1 = initialize_scene('scene 1') scene2 = initialize_scene('scene 2') scene1.show() ASSERT scene1.visible EQUALS TO True // Move to Scene 2 transition_scene('scene 2') ASSERT scene1.visible EQUALS TO False ASSERT scene2.visible EQUALS TO True END FUNCTION </pre>
Comments	It is critically important to make sure that the player object does not get sent to an unintended scene, as a mistake like this could lead to a “softlock” [9] making it unbeatable
Created by	JW
Test environment	Windows 10/11, Unity test environment
Failure Severity	Catastrophic

Test Case ID	unit_test-05
Description of test	Tests that inventory can be accessed through a simulated button trigger
Related requirement specification	FR1.3 FR2.2 FR2.4
Pre-requisite for test	<ul style="list-style-type: none"> • Inventory System
Test procedure	<ol style="list-style-type: none"> 1. Trigger inventory pop-up event 2. Trigger inventory close event
Test material used	<ul style="list-style-type: none"> • Debug Console • Unity Test Framework • Placeholder inventory PNG
Expected result (test oracle)	<ul style="list-style-type: none"> • Inventory shown on screen after pop-up event and hidden after close event <pre> FUNCTION test_inventory_open() // Initialize inventory with nothing inventory = initialize_inventory() ASSERT inventory.visible EQUALS TO False // On KBR, button 'i' opens inventory keyboard_input('i') ASSERT inventory.visible EQUALS TO True END FUNCTION </pre> <ul style="list-style-type: none"> • Using the same structure, make another test for closing inventory
Comments	Pause menu cannot be accessed during room transitions and cutscenes
Created by	JP
Test environment	Windows 10/11, Unity test environment
Failure Severity	Moderate

Test Case ID	unit_test-06
Description of test	Tests that new game files can be generated through a trigger
Related requirement specification	FR1.3 FR2.1 FR2.3
Pre-requisite for test	<ul style="list-style-type: none"> • Save System
Test procedure	<ol style="list-style-type: none"> 1. Trigger new game file generation procedure 2. Select save file to overwrite 3. Delete all existing save files 4. Trigger new game file generation procedure
Test material used	<ul style="list-style-type: none"> • Debug Console • Unity Test Framework • Existing save file
Expected result (test oracle)	<ul style="list-style-type: none"> • When the user attempts to start a new game and chooses a file to overwrite, the old file must be deleted, and the new save should be present <pre> FUNCTION test_new_game() saver = initialize_saver() saver.save_state('save_1.txt') // Overwrite save 1 generate_new_game('save_1.txt', 'new_save.txt') ASSERT exists('save_1.txt') EQUALS TO False ASSERT exist('new_save.txt') EQUALS TO True END FUNCTION </pre> <ul style="list-style-type: none"> • We can use the same structure to also make sure that if no save files exists, no old files need to be deleted
Comments	There will be a separate button for continuing to play a saved game
Created by	JP
Test environment	Windows 10/11, Unity test environment
Failure Severity	Catastrophic

System Tests

Test Case ID	sys_test-01
Description of test	Tests that all main menu contains: a background image, new game button, continue button, options button, credits button and exit button
Related requirement specification	FR1.3 FR2.1 FR2.3
Pre-requisite for test	<ul style="list-style-type: none"> • Save System • Placeholder scene • Control input reader
Test procedure	<ol style="list-style-type: none"> 1. Load into main menu 2. Press start new game button 3. Select save file to overwrite 4. Reset back to main menu and try the rest of the following buttons: Continue, options, credit, exit game (skipping step 3)
Test material used	<ul style="list-style-type: none"> • Debug Console • Placeholder background PNG • Existing save file
Expected result (test oracle)	<ul style="list-style-type: none"> • Background image is loaded in • New game button prompts user to overwrite an existing save file (if it exists) then loads placeholder scene and creates new save • Continue game button loads most recent save state • Option button opens the options menu. Settings set in this menu are stored even after reloading game • Credits button loads the credits • Exit game button exits the application
Comments	Background image could also be a GIF
Created by	JP
Test environment	Windows 10/11, Unity test environment
Failure Severity	Catastrophic

Test Case ID	sys_test-02
Description of test	Tests the functionality of the game's progression: The Skills Build skill tree, where the user can upgrade their stats based on the Skills Build courses
Related requirement specification	FR2.4 FR3.1 FR3.2
Pre-requisite for test	<ul style="list-style-type: none"> • Combat System • Skill Tree menu • Skill nodes
Test procedure	<ol style="list-style-type: none"> 1. Defeat placeholder enemy (using debug console) 2. Access Skills Build menu 3. Allocate a skill point to a skill node 4. Enter combat encounter 5. Repeat for all skill nodes (Use debug console to add more skill points)
Test material used	<ul style="list-style-type: none"> • Debug Console
Expected result (test oracle)	<ul style="list-style-type: none"> • Skill point counter increases upon combat win • Skills Build menu can be accessed and loads correctly • Skill point is added to a skill; skill point is removed from the player object • Cannot unlock skill with 0 skill points • Skill has intended effect in combat
Comments	We aim to have at least 4 different skill trees, each containing 3 skill nodes. The skill trees represent the Skills Build courses mentioned by the client
Created by	JW
Test environment	Windows 10/11, Unity test environment
Failure Severity	Serious

Test Case ID	sys_test-03
Description of test	Tests that game functions with keyboard and controller
Related requirement specification	FR1.2 FR1.3 FR4.1 NFR1.2
Pre-requisite for test	<ul style="list-style-type: none"> • Player Movement • Interactable objects
Test procedure	<ol style="list-style-type: none"> 1. Load into hub world 2. Move using WASD and arrow keys 3. Interact with an interactable object using Space bar 4. Repeat step 2 using a controller's D-pad and controller's joystick 5. Repeat step 3 using the interact button on the controller (example: X on a PS5 controller or A on an Xbox controller)
Test material used	<ul style="list-style-type: none"> • Mouse and keyboard • PS4, PS5, Xbox and Nintendo Switch controllers
Expected result (test oracle)	<ul style="list-style-type: none"> • All movement inputs move the player object in the intended direction • All interact buttons interact with the test object when pressed. We use a sound cue to identify whether the object has been interacted with
Comments	As long as the controller is recognized by windows when plugged in, it will also work on Unity. Failure Severity for one of the input methods not functioning is Serious. If all methods are non-functional, then this becomes Catastrophic
Created by	JP
Test environment	Windows 10/11, Unity test environment
Failure Severity	Catastrophic

Test Case ID	sys_test-04
Description of test	Tests that the game runs at stable 60fps or above after adding a certain number of moving objects to scene
Related requirement specification	FR2.5 FR4.3 FR4.4 NFR3.8
Pre-requisite for test	<ul style="list-style-type: none"> • Scene • Moving objects/entity • Camera object
Test procedure	<ol style="list-style-type: none"> 1. Load camera and scene 2. Add one moving object inside camera range 3. Check FPS counter 4. Repeat steps 2-3 until FPS hits 60
Test material used	<ul style="list-style-type: none"> • Debug Console • Unity Profiler
Expected result (test oracle)	<ul style="list-style-type: none"> • Game FPS should stay at or above 60 for a minimum of 20 moving objects in scene
Comments	Tests should be done on various laptops running windows to test varying GPUs and CPUs
Created by	JP
Test environment	Windows 10/11, Unity test environment
Failure Severity	Serious

Test Case ID	sys_test-05
Description of test	Sound and controls should be able to be changed through the pause menu
Related requirement specification	FR1.3 FR2.2 FR5.1
Pre-requisite for test	<ul style="list-style-type: none"> • Pause menu • Sound system • Control input reader
Test procedure	<ol style="list-style-type: none"> 1. Open pause menu 2. Go to settings 3. Change interact button from space to enter (with controller, do an equivalent button mapping) 4. Set the following values to 0 (originally these are at 100): master volume, sound effect volume, background volume 5. Repeat step 4 with value 50.
Test material used	<ul style="list-style-type: none"> • Speaker • Keyboard and mouse • Controllers
Expected result (test oracle)	<ul style="list-style-type: none"> • Interact button is bound to enter (or equivalent for controller) • All game sound can be turned off or lowered
Comments	There could be more settings that we choose to add later on, however, these settings are the main and most important ones.
Created by	JP
Test environment	Windows 10/11, Unity test environment
Failure Severity	Serious

Test Case ID	sys_test-06
Description of test	Room Transition Delay
Related requirement specification	FR4.3 NFR 3.4
Pre-requisite for test	<ul style="list-style-type: none"> • Five scenes with puzzles
Test procedure	<ol style="list-style-type: none"> 1. Queue up 5 back-to-back scene transitions 2. Extract time between first and last transition in the Unity Profiler 3. Repeat 10 times and get an average
Test material used	<ul style="list-style-type: none"> • Unity Profiler
Expected result (test oracle)	<ul style="list-style-type: none"> • All room transitions should take less than 15 seconds in total (3 seconds per transition)
Comments	Ideally this should be tested for all possible transitions as some rooms may take longer to load than others, however this quite difficult to automatically test, and so only 5 is chosen. Also note that a room is equivalent to a scene
Created by	CDV
Test environment	Windows 10/11, Unity test environment
Failure Severity	Moderate

User Acceptance Tests

Test Case ID	uat_test-01
Description of test	Tests that events in response to user input occurs with as little delay as possible to prevent the game from appearing “laggy”
Related requirement specification	FR1.1 FR1.3 FR4.3 NFR3.1
Pre-requisite for test	<ul style="list-style-type: none"> • Audio system • Game object • Control input reader
Test procedure	<ol style="list-style-type: none"> 1. Start screen recording software 2. Start game 3. Open scene 4. Perform user interaction 5. Close game 6. End screen recording 7. Open screen recorded video 8. Verify all user interaction events receive responses within the allotted time
Test material used	<ul style="list-style-type: none"> • Mouse and keyboard • Screen recording software • Speaker • Unity profiler • Survey form
Expected result (test oracle)	<ul style="list-style-type: none"> • All events occur within 10 milliseconds
Comments	Due to the vast amounts of scenes involving user interaction, it will be infeasible to manually verify the response time to all potential series of user interactions. As such, we will take a random sample of scenes from the game and perform only a handful of player actions
Created by	SE
Test environment	Windows 10/11, Unity test environment
Failure Severity	Serious

Test Case ID	uat_test-02
Description of test	Tests that the user is satisfied with player movement, with movement being concise and animations feeling natural
Related requirement specification	FR1.1 FR1.2 NFR3.2
Pre-requisite for test	<ul style="list-style-type: none"> • Player movement • Player animation
Test procedure	<ol style="list-style-type: none"> 1. Recruit survey participants 2. Present survey participants with game 3. Survey participants play the game for an allotted amount of time: <ul style="list-style-type: none"> • User enters a Room in-game • Use directional keys on the keyboard to move • Repeat using controller 4. Request user to answer questionnaire on if the movement is smooth 5. Collect responses
Test material used	<ul style="list-style-type: none"> • Mouse and keyboard • Controller • Survey form
Expected result (test oracle)	<ul style="list-style-type: none"> • User is satisfied with the smoothness of the movement
Comments	Test will be evaluated by asking several users to test their satisfaction with the movement of the player. If 70% of users are satisfied with the movement, the test is successful
Created by	JP
Test environment	Windows 10/11, Unity test environment
Failure Severity	Moderate

Test Case ID	uat_test-03
Description of test	Tests that users understand the game and the IBM Skills Build content
Related requirement specification	FR4.1 FR4.2
Pre-requisite for test	<ul style="list-style-type: none"> • Game object • Puzzles built to test player's knowledge of IBM Skills Build content • Functional combat system with questions from IBM Skills Build
Test procedure	<ol style="list-style-type: none"> 1. Recruit survey participants 2. Present survey participants with game 3. Survey participants play the game for an allotted amount of time: <ul style="list-style-type: none"> • Solve a randomly assigned puzzle • Solve a randomly assigned question 4. Present survey participants with the question "How well did you understand the game?" 5. Present survey participants with the question "How well did you engage with the IBM skills-build content/questions?" 6. Collect responses
Test material used	<ul style="list-style-type: none"> • Survey form
Expected result (test oracle)	<ul style="list-style-type: none"> • All users have a high degree of confidence in how well they have understood the game, and feel that it is a good way to engage with IBM Skills Build content
Comments	To recruit our survey participants we will take a sample of university students from both in and out of the Computer Science department
Created by	SE
Test environment	Windows 10/11, Unity test environment
Failure Severity	Moderate

Test Case ID	uat_test-04
Description of test	Tests that the users feels that combat is fair
Related requirement specification	FR1.3 FR3.1 FR4.5 FR4.6 NFR3.5
Pre-requisite for test	<ul style="list-style-type: none"> • Combat System
Test procedure	<ol style="list-style-type: none"> 1. Recruit survey participants 2. Present participants with boss combat 3. Ask participants “Do you think that the enemy has too much, too little or just the right amount of health?” 4. Ask participants “Do you feel that the main character has too much, too little or just the right amount of health?” 5. Collect responses
Test material used	<ul style="list-style-type: none"> • Survey form
Expected result (test oracle)	<ul style="list-style-type: none"> • User is satisfied with the fairness of combat
Comments	It would be ideal if we can recruit a variety of participants with a range of knowledge of the Skills Build courses, to collect additional data on how responses vary based on knowledge
Created by	JW
Test environment	Windows 10/11, Unity test environment
Failure Severity	Moderate

Test Case ID	uat_test-05
Description of test	Tests that the users feels that the Skills Build skill tree abilities feel impactful
Related requirement specification	FR1.1 FR3.1 FR3.2 FR4.5 FR4.6 NFR 3.5 NFR3.7
Pre-requisite for test	<ul style="list-style-type: none"> • Combat System • Skills Build Skill Tree
Test procedure	<ol style="list-style-type: none"> 1. Recruit survey participants 2. Present participants with boss combat, with one or more Skills Build skill tree abilities enabled, and then with no Skills Build tree abilities enabled 3. Ask participants: “Do you think that the Skills Build tree abilities made a significant impact on the combat” 4. Collect responses
Test material used	<ul style="list-style-type: none"> • Survey form
Expected result (test oracle)	<ul style="list-style-type: none"> • User feels like abilities are impactful
Comments	Ideally, we should have every person do at least 3 combat sections, each iteration with different abilities, to collect more reliable responses.
Created by	JW
Test environment	Windows 10/11, Unity test environment
Failure Severity	Moderate

Test Case ID	uat_test-06
Description of test	Tests that the users feels that game has meaningful progression
Related requirement specification	FR3.2 FR4.5 FR4.6 NFR3.5 NFR3.7
Pre-requisite for test	<ul style="list-style-type: none"> • Puzzles • Skills-build tree
Test procedure	<ol style="list-style-type: none"> 1. Recruit survey participants 2. Allow survey participants to play through a random puzzle scene of the game 3. Allow survey participants to play through a random question scene of the game 4. Upon completion, present survey participants with the question: "Do you believe the game had meaningful progression?" 5. Present users with the question: "Do you believe the progression between the topics from the IBM Skills Build was smooth and easy to follow?" 6. Collect responses
Test material used	<ul style="list-style-type: none"> • Survey form
Expected result (test oracle)	<ul style="list-style-type: none"> • We expect that all users feel the game had reasonable progression • We expect that all users feel comfortable with the presentation of the IBM Skills Build content
Comments	If the responses to these survey questions are unsatisfactory, we will conduct a further round of surveys to ask the participants which particular areas they felt the game had poor progression
Created by	SE
Test environment	Windows 10/11, Unity test environment
Failure Severity	Moderate

Testing Outcomes

As this is merely a test **plan** report, we have not done all the tests we have listed, and are still at the unit testing stage (as the order of the tests we are doing is unit tests -> integration tests -> system tests -> user acceptance tests).

The following is a list of the unit tests we have carried out and their outcomes:

Test ID	Test Name	Test Outcome
unit_test-01	Player Movement	PASSED
unit_test-02	Camera Logic	PASSED
unit_test-04	Scene Transition	PASSED
unit_test-10	Close game	PASSED
unit_test-11	Collisions	PASSED

Testing Context

This section will go through our tools and environment used. We will also go through our choice of failure severity.

Workflow

We make use of GitHub and automated testing in order to provide a continuous integration workflow. Frequent commits and pushes help our members understand the code more, allowing for easier merges and bug fixes. [10]

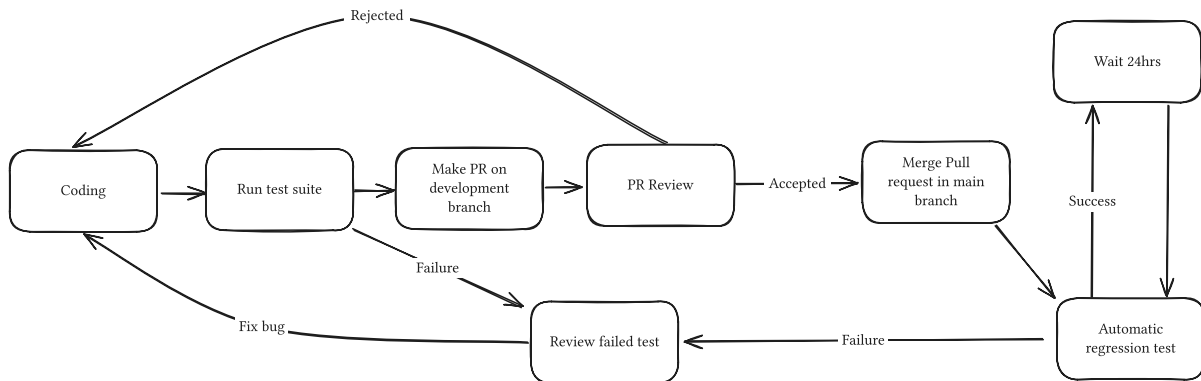


Figure 3: Development workflow

When developing the game, we will go through a few phases following the GitHub recommended workflow [11]:

1. Coding and updating the game codebase.
2. Running automatic test suite.
3. Make a pull request onto the development branch.
4. Pull request review and quality check code.
5. Merge PR to main branch for deployment. For our use case, deployment means that this code can belong in the main code base, and if we decide to make an executable, the included feature should function as expected.
6. Automatic regression test every 24 hours at midnight every day.

For all tests, if any failure is detected, we will immediately go into a review phase and go back to step 1.

Notice that this is a general development workflow with testing interwoven, which is not necessarily the steps we will take to conduct the tests mentioned above.

Tools and Environment

As per the request of our client [1], our game must run on Windows and be developed using Unity. Using an executable for UAT is an option, however we will lose the built-in benefits provided by Unity for testing (Unity testing environment, Unity Profiler).

For that reason, we have chosen the Unity testing environment for UAT, as it is easier to fix or change the code for users in real-time in-case of a game breaking bug. Also, it comes with a developer console, so we could more closely monitor and log which component of the code caused the issue. For more performance related tests, Unity also offers the Unity Profiler, a tool that will help us analyse more accurately [12].

For unit and systems tests, we will use the Unity Test Framework (UTF), as it allows for easy test writing and automatic execution, especially when it comes to unit tests [13]. It would also be easier to

set this up on Unity, and has a lower barrier of entry, allowing access to all members of our group to write and run tests.

We decided on testing on Windows 10/11 and forego older versions as the majority of users will be on these newer OS (95% of users are on these versions [14]). Also, since Microsoft has officially stopped support for Windows 7/8.1 and other older versions [15], we do not want to spend any time on backwards compatibility. The difference between Windows 10 and 11 should be very minor, but we will tolerate a 5-10% difference in performance between the two platform [16].

With regard to controls, in addition to keyboard and mouse input, we will need to test whether they are compatible with console controllers. This is because there is a 50:50 split in player base between controller vs keyboard and mouse users when it comes to adventure games [17], and we want our game to be accessible to as many people as possible.

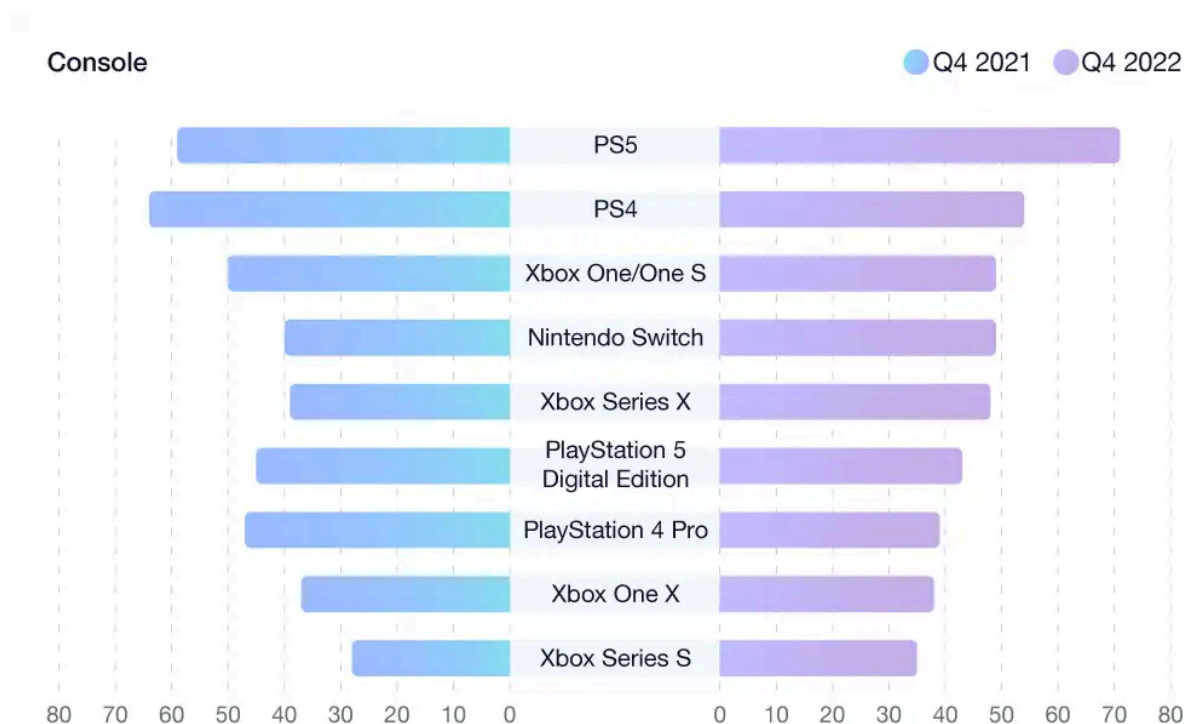


Figure 4: Primary console user base in the UK in percentage (%) [18]

In Figure 4, we can see the most utilised consoles of 2021 and 2022. **Notice:** The cumulative percentage is more than 100% as some people own multiple consoles. Console utilisation is relevant for us is because we need to identify which controllers to use for our tests. From this data, we will be doing our test on PS4/5, Xbox, and Nintendo Switch controllers, as these are the most utilised console controllers of this generation of gamers.

All tests will be stored in a separate folder for reuse. Furthermore, all automatable tests (all units tests, integration tests and some system tests) will be automatically ran using GitHub Actions at the start each day as a form of regression testing [19]. Feedback will be sent by email to all member of our team, and any failed tests will be dealt with by our Technical Lead (specified in the Requirement Specification).

Failure severity

For our testing system, we decided on using a 3-level scale to categorize faults, as this method is traditional for game development [20]. These include:

Severity	Explanation	Example	Associated tests
Moderate	Inconveniences which does not take away from the game.	Missing textures, missing sound effects, grammatical mistakes, music not looping.	unit_test-05, sys_test-06, uat_test-02, uat_test-04, uat_test-05, uat_test-06
Serious	Bugs which impedes/gets in the way of game progression, whilst not totally breaking the game or stopping progress.	Frequent FPS drops, hitbox misaligned, camera not functioning as expected (trailing behind, off-centred). Wrong room transition/warp.	sys_test-02, sys_test-03, sys_test-04, sys_test-05, uat_test-01, uat_test-03
Catastrophic	Game breaking issues, requiring a full restart or forcing user to fully restart to resume progression. Might cause data loss (like corrupting save files).	Game freeze, game crash, massive FPS drop, game mechanics breaking (negative health, infinite damage), soft-locks [9], camera/keyboard non-functional.	unit_test-01, unit_test-02, unit_test-03, unit_test-04, unit_test-06, sys_test-01, sys_test-03

This system also maps nicely to our functional requirement's **MuShCo** system. **Catastrophic** faults **must** be addressed first, then **serious** faults **should** be addressed, and **moderate** faults **could** be addressed later. This allows us to infer the severity of faults by matching the tests with our functional requirements.

Due to the inherent nature of our project, some severity levels are not applicable, such as **infectious**, as we only have one running component in our project (the game) and so this shutting down cannot spread to anything else.

User acceptance survey and questionnaire

To evaluate our game from the perspective of user feedback, we designed a questionnaire that will allow us to gain insight into how well we have accomplished our quality assurance goals. The questionnaire asks participants questions about their experiences on matters such as accessibility and understanding.

IBM Skills Build platformer

User Acceptance Questionnaire

02/24

FEBRUARY 2024

CONTEXT

This is an educational 2D RPG game that features questions from the IBM Skills Build website. These questions will allow you as the player to progress throughout the game when you answer these questions correctly.

Every area in the game represents a particular Skills Build course and the questions presented in this area will test understanding of the skills learned from the course in question.

During the game, you will solve puzzles and come across “bosses” who will pose questions to you. The game introduces an element of risk in the form of a health and combat system in which the game will deduct a “heart” from the player upon answering a question incorrectly. Losing all hearts will result in the player restarting the entire boss fight.

INSTRUCTIONS

You are to play through the game adhering to the instructions outlined on your designated survey participant card. Please take note of any issues or difficulties you may face while playing the game. Once you have finished playing through the game, please evaluate your experiences and report on how you felt the game performed in aspects such as graphics quality, gameplay mechanics, engagement, progression, and accessibility.

See the survey participant card for questions.

Figure 5: The first page of the user acceptance survey

Survey participant card

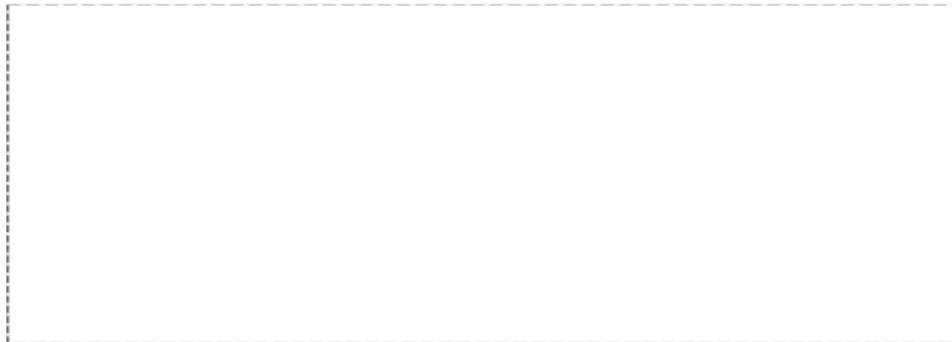
QUESTIONS

Answer the questions below and write your responses on your response sheet

1. How well did you understand the game?
2. How well did you engage with the IBM Skills Build content and the in-game questions?
3. Do you think the enemy has too much, too little or just the right amount of health?
4. Do you feel that the main character has too much too little or just the right amount of health?
5. Do you believe the game had meaningful progression?
6. Do you believe the progression between the topics from the IBM skills build was smooth and easy to follow?

FREE FORM RESPONSE

Write any feedback you would like to give concerning your experience playing the game.



OVERALL RATING

On a scale from 1 to 7, how would you rate your experience playing the game?



Figure 6: The questions posed to survey participants

Bibliography

- [1] J. McNamara, “Reimagine Skills Build as an RPG game - Briefing clarification”. Oct. 13, 2023.
- [2] J. Pulham, J. Watson, C. Dubois-Veltman, S. Ezech, N. Le, and A. Sirohia, “Group 17 Requirement Specification”. Nov. 23, 2023.
- [3] CodiumAI, “Unit Test Automation”. [Online]. Available: <https://codium.ai/glossary/unit-test-automation/>
- [4] Aakanksha Dixit, “Efficiency at its Best, Automated Integration Testing”. [Online]. Available: <https://opkey.com/blog/efficiency-at-its-best-automated-integration-testing>
- [5] imperva, “What is White Box Testing”. [Online]. Available: <https://www.imperva.com/learn/application-security/white-box-testing/>
- [6] Unity, “EventTrigger”. [Online]. Available: <https://docs.unity3d.com/2018.2/Documentation/ScriptReference/EventSystems.EventTrigger.html>
- [7] R. Jain, “Bottom-Up Integration Testing- An Overview”. [Online]. Available: <https://testsigma.com/blog/bottom-up-integration-testing/>
- [8] M. Ellis, “The Complete Guide to Integration Testing”. [Online]. Available: <https://blog.hubspot.com/website/integration-testing>
- [9] “What Does Softlock and Hardlock Mean in Video Games?”. [Online]. Available: <https://makeuseof.com/softlock-hardlock-in-video-games-explained/>
- [10] Github, “About continuous integration”. [Online]. Available: <https://docs.github.com/en/actions/automating-builds-and-tests/about-continuous-integration>
- [11] Github, “About workflows”. [Online]. Available: <https://docs.github.com/en/actions/using-workflows/about-workflows>
- [12] Unity, “Testing and quality assurance tips for Unity projects”. [Online]. Available: <https://unity.com/how-to/testing-and-quality-assurance-tips-unity-projects#regression-testing>
- [13] “Unit Testing”. [Online]. Available: <https://docs.unity3d.com/Manual/testing-editor-test-runner>
- [14] “Desktop Windows Version Market Share Worldwide”. [Online]. Available: <https://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide>
- [15] Microsoft, “End of support for Windows 7 and Windows 8.1”. [Online]. Available: <https://www.microsoft.com/en-gb/windows/end-of-support>
- [16] J. Lauken, “Is Windows 11 Good For Gaming?”. [Online]. Available: <https://lifewire.com/is-windows-11-good-for-gaming-7153341>
- [17] J. Jimenez, “Only around 10% of Steam gaming sessions are played with a controller”. [Online]. Available: <https://pcgamer.com/only-around-10-of-steam-gaming-sessions-are-played-with-a-controller/>
- [18] N. Baker, “Online gaming statistics 2023”. [Online]. Available: <https://uswitch.com/broadband/studies/online-gaming-statistics/>
- [19] B. Douglas, “4 ways we use GitHub Actions to build GitHub”. [Online]. Available: <https://github.blog/2022-04-05-4-ways-we-use-github-actions-to-build-github/>
- [20] G. M. Pollock, “Criteria 1 – Understanding types of defects (bugs) within a game build”. [Online]. Available: <https://georgepollboa.wordpress.com/year-12/term-3-cmpnmo3-2d-animation-and-testing/games-development/unit-77-designing-tests-for-computer-games-2/unit-77-designing-tests-for-computer-games/>