Our Manual Test Cases

Tabular Overview

Below is a tabular overview of the manual test cases we used to test our project and their result. Scroll down for a more detailed description of each test case and the steps involved to test it.

ID	DESCRIPTION	RESULT
UI/UX Tests		
TC_UI_01	Main Menu Buttons	PASS
TC_UI_02	Button Feedback	PASS
TC_UI_03	Settings Menu Functionality	PASS
TC_UI_04	Game Info Bar Statistics	PASS
TC_UI_05	Animated Students	PASS
Gameplay Tests		
TC_GAME_01	Building Placement and Price	PASS
TC_GAME_02	Building Placement with Collisions	PASS
TC_GAME_03	Game Paused	PASS
TC_GAME_04	Game Over	PASS
Event Tests		
TC_EVENT_01	“Busy Week” Event	PASS
TC_EVENT_02	“Alumni Donation” Event	PASS
TC_EVENT_03	“Building Discount” Event	PASS
TC_EVENT_04	“Rain” Event	PASS
TC_EVENT_05	“Roses” Event	PASS
Difficulty Tests		
TC_DIFF_01	“Normal” Difficulty	PASS
TC_DIFF_02	“Easy” Difficulty	PASS
TC_DIFF_03	“Hard” Difficulty	PASS

Achievement Tests		
TC_ACH_01	“Entrepreneur” Achievement	PASS
TC_ACH_02	“I Heart Uni” Achievement	PASS
TC_ACH_03	“One Hundredth Student” Achievement	PASS
Leaderboard Tests		
TC_LEAD_01	Leaderboard Initialisation	PASS
TC_LEAD_02	Leaderboard with No Name	PASS
TC_LEAD_03	Leaderboard with Name	PASS
TC_LEAD_04	Leaderboard with Same Name	PASS
TC_LEAD_05	Leaderboard with 5+ Entries	PASS
Non-Functional Requirement Tests		
TC_PERF_01	Game Performance	PARTIAL PASS
TC_USE_01	Ease of Use/Playability	PASS

UI/UX Tests

TC_UI_01

- **Objective:** Verify that the main menu is intuitive and all navigation buttons are functional.
- **Linked Requirements:** UR_EASE_OF_USE, FR_USER_INTERFACE
- **Steps:**
    1. Load the game.
    2. Navigate through all the menu button options (Play, Leaderboard, Help, Settings, and Quit).
- **Expected Outcome:** The main menu is easy to navigate and all the button options function correctly and redirect users to the respective page.
- **Actual Outcome:** **PASS - All buttons work and are easy to navigate.**

TC_UI_02

- **Objective:** Verify that all buttons provide visual feedback when interacted with.
- **Linked Requirements:** NFR_INTERACTIVE_ELEMENTS_REACTION
- **Steps:**
    1. Hover and/or click buttons within the game and its menus.
    2. Observe for any colour changes.
- **Expected Outcome:** All buttons provide visual feedback (mainly via the way of colour change) within 1 second of interaction.
- **Actual Outcome:** **PASS - All interactive menu items provide visual feedback.**

TC_UI_03

- **Objective:** Verify that the settings menu functions correctly.
- **Linked Requirements:** FR_SETTINGS
- **Steps:**
    1. Load the settings menu.
    2. Adjust the different functionalities; sound level, difficulty level, and debug keys.
- **Expected Outcome:** All the functionalities work as expected; the sound level is adjusted based on the user adjustment of the volume slider, the difficulty level is adjusted based on the user selection, and the debug keys are enabled/disabled based on user choice.
- **Actual Outcome:** **PASS - The debug keys work when enabled and don't when disabled, the volume slider adjusts the volume accurately, and the difficulty level of the game accurately reflects the level chosen.**

TC_UI_04

- **Objective:** Verify the visibility of vital gameplay statistics on the Info Bar and their operability.
- **Linked Requirements:** UR_SCORE, FR_TIMER, FR_BUILDING_COUNTER
- **Steps:**
    1. Start the game.
    2. Observe the statistics in the top bar; Time, Timeline Description, Money, Satisfaction, and Building Type counters.
- **Expected Outcome:** All the statistics can be viewed throughout the gameplay, and are updated to accurately display the current gameplay stage.

- **Actual Outcome:** **PASS - The statistics accurately update. The timer decreases as time goes on, the year timeline progresses as the timer goes on, the amount of money changes based on the events of the game, the satisfaction level varies based on the events of the game, and the number of students and buildings accurately change too.**

TC_UI_05

- **Objective:** Verify the presence of walking students on the map once buildings have been placed.
- **Linked Requirements:** FR_BACKGROUND
- **Steps:**
    1. Start the game.
    2. Observe the map's background for any animations or elements.
    3. Place a building, then repeat step 2.
- **Expected Outcome:** There are animated students walking across the map between placed buildings.
- **Actual Outcome:** **PASS - Walking students only appear once buildings are placed.**

Gameplay Tests

TC_GAME_01

- **Objective:** Verify that buildings can be placed on the map, and that the player knows the price of it and that such price is deducted from their money.
- **Linked Requirements:** FR_BUILDINGS
- **Steps:**
    1. Start the game.
    2. Place one of each type of building can be placed on the map; Study Building, Accommodation Building, Eating Building, Basketball Court, and Club.
- **Expected Outcome:** Each of the buildings should be able to be placed on the game's map by users during gameplay, and the user should be able to see how much the building costs, and then see their amount of money reduced by that.
- **Actual Outcome:** **PASS - Each building type can be placed and the price the player is told is correctly subtracted from their total amount of money.**

TC_GAME_02

- **Objective:** Verify that buildings cannot be placed on obstacles or out of bounds.
- **Linked Requirements:** UR_BUILDING_LIMITS, FR_OBSTACLES, NFR_ERROR_MESSAGES
- **Steps:**
    1. Start the game.
    2. Attempt to place a building on each type of obstacle; other building, road, body of water, and outside of the map's bounds.
    3. Observe the gameplay's reaction.
- **Expected Outcome:** All attempts should be rejected by the game - the player should not be able to place a building on any invalid locations and a red border should appear around the building.
- **Actual Outcome:** **PASS - A building can not be placed on any type of obstacle, or out of bounds.**

TC_GAME_03

- **Objective:** Verify that buildings cannot be placed, events do not progress, and statistics do not change, while the game is paused and that there is a message saying the game is paused.
- **Linked Requirements:** UR_TIME, FR_TIMER, UR_BUILDING_LIMITS
- **Steps:**
    1. Start the game, is there a message saying the game is paused.
    2. Press play, then pause again.
    3. Attempt to place buildings, and check if the timer pauses.
    4. Start the game again, and pause it midway through an event.
    5. Verify that the event does not continue/end whilst the game is paused.
- **Expected Outcome:** While the game is paused, the user should be notified and nothing should happen; the player should not be able to place any buildings, the timer should stop, and events shouldn't continue.
- **Actual Outcome:** **PASS - The message "Game is paused" is shown every time the game is paused. Money, Student Count, Year Timeline, and Satisfaction Values do not change while being paused, and events and the timer are paused while the game is paused.**

TC_GAME_04

- **Objective:** Verify the game ends when the timer reaches 0:00.
- **Linked Requirements:** UR_TIME, NFR_END_OF_GAME
- **Steps:**
    1. Start the game.
    2. Wait for the timer to run out, and look out for a game-over screen.
- **Expected Outcome:** When the timer reaches 0:00 i.e. the game has played for 5 minutes, the user is directed to the game-over screen.
- **Actual Outcome:** **PASS - The user is redirected to the Game Over screen the second the timer reaches 0:00.**

Event Tests

TC_EVENT_01

- **Objective:** Verify that the Busy Week event occurs, the user is informed, and that it affects gameplay.
- **Linked Requirements:** UR_EVENTS, FR_EVENTS
- **Steps:**
    1. Start the game.
    2. Wait for a Busy Week event to occur, and observe any changes to the game.
- **Expected Outcome:** When the event occurs, it is a neutral event and thus not much happens, but additional animated students should appear on the map, and the player should be informed of the event.
- **Actual Outcome:** **PASS - The game is paused, a pop-up window appears, and the number of animated students on the map increases once the user resumes gameplay.**

TC_EVENT_02

- **Objective:** Verify that the Alumni Donation event occurs, the user is informed, and that it affects gameplay.

- **Linked Requirements:** UR_EVENTS, FR_EVENTS
- **Steps:**
    1. Start the game.
    2. Wait for the Alumni Donation event to occur, and observe any changes to the game.
- **Expected Outcome:** When the event occurs, the player should be given a random boost to the amount of money they currently have i.e. +£5000, and they should be informed of the event.
- **Actual Outcome:** **PASS - The game is paused, a pop-up window appears, and the amount of money increases by a random amount of money once the user resumes gameplay.**

TC_EVENT_03

- **Objective:** Verify that the Building Discount event occurs, the user is informed, and that it affects gameplay.
- **Linked Requirements:** UR_EVENTS, FR_EVENTS
- **Steps:**
    1. Start the game.
    2. Wait for the Building Discount event to occur, and observe any changes to the game.
- **Expected Outcome:** When the event occurs, the cost to build each building should be reduced by 20% for a random duration, and the player should be informed of the event.
- **Actual Outcome:** **PASS - The game is paused, a pop-up window appears, the cost of a building is reduced by 20% and when placed the amount of money is reduced by the amended price, and once the event ended the building prices returned to normal.**

TC_EVENT_04

- **Objective:** Verify that the Rain event occurs, the user is informed, and that it affects gameplay.
- **Linked Requirements:** UR_EVENTS, FR_EVENTS
- **Steps:**
    1. Start the game.
    2. Wait for the Rain event to occur, and observe any changes to the game.
- **Expected Outcome:** When the event occurs, the game's satisfaction level should drop, and they should be informed of the event.
- **Actual Outcome:** **PASS - The game is paused, a pop-up window appears, and the satisfaction level randomly decreases once the user resumes the game.**

TC_EVENT_05

- **Objective:** Verify that the Roses event occurs, the user is informed, and that it affects gameplay.
- **Linked Requirements:** UR_EVENTS, FR_EVENTS
- **Steps:**
    1. Start the game.
    2. Wait for the Roses event to occur, and observe any changes to the game.
- **Expected Outcome:** When the event occurs, the game's satisfaction level should increase, and they should be informed of the event.

- **Actual Outcome:** **PASS - The game is paused, a pop-up window appears, and the satisfaction level randomly increases once the user resumes the game.**

Difficulty Tests

TC_DIFF_01

- **Objective:** Test that the "Normal" level of difficulty is implemented when the game is played without a difficulty level selection, or when selected after a different difficulty level. Also, verify that the gameplay conditions for the Normal level are implemented.
- **Linked Requirements:** UR_DIFFICULTY_SETTINGS, FR_DIFFICULTY_ SELECTION, FR_DIFFICULTY_SETTINGS
- **Steps:**
    1. Start the game.
    2. Verify that the gameplay instance starts with £75,000 and that any type event occurs including negative (Normal and Hard specific), and building discount (Normal and Easy specific).
    3. Restart the game on a different difficulty level, finish the game, select "Normal" level, and repeat steps 1 and 2 again.
- **Expected Outcome:** Whether the game starts without any difficulty level selection, or with a "Normal" level of difficulty, the player will receive £75,000 and will experience any of the potential events during the 5 minute window of gameplay.
- **Actual Outcome:** **PASS - The player started with £75000, the building discount event occurred, and the rain event occurred. Changed difficulty to Easy, played, then re-selected Normal - correct functionalities occurred.**

TC_DIFF_02

- **Objective:** Test that the "Easy" level of difficulty is implemented when the user selects such in the settings menu, and verify that the gameplay conditions reflect the user's selection.
- **Linked Requirements:** UR_DIFFICULTY_SETTINGS, FR_DIFFICULTY_ SELECTION, FR_DIFFICULTY_SETTINGS
- **Steps:**
    1. Select "Easy" as the level of difficulty.
    2. Start the game.
    3. Verify that the gameplay instance starts with £100,000 and that only positive and neutral events occur during gameplay.
- **Expected Outcome:** The gameplay instance starts with the easy gameplay conditions; the player receives £100,000 at the start of the game and only experiences positive or neutral events during the course of the 5 minute window.
- **Actual Outcome:** **PASS - The player started with £100000, and no negative events occurred (ran the game three times).**

TC_DIFF_03

- **Objective:** Test that the "Hard" level of difficulty is implemented when the user selects such in the settings menu, and verify that the gameplay conditions reflect the user's selection.
- **Linked Requirements:** UR_DIFFICULTY_SETTINGS, FR_DIFFICULTY_ SELECTION, FR_DIFFICULTY_SETTINGS

- **Steps:**
    1. Select “Hard” as the level of difficulty.
    2. Start the game.
    3. Verify that the gameplay instance starts with £50,000 and that events of any type occur except for the Building Discount event.
- **Expected Outcome:** The gameplay instance starts with the hard gameplay conditions; the player receives £50,000 at the start of the game and experiences events of any type except for the Building Discount event.
- **Actual Outcome:** **PASS - The player started with £50000, and no building discount event occurred - negative events did (ran the game twice).**

Achievement Tests

TC_ACH_01

- **Objective:** Verify that the “Entrepreneur” achievement unlocks correctly.
- **Linked Requirements:** UR_ACHIEVEMENTS, FR_ACHIEVEMENTS_SYSTEM, FR_ACHIEVEMENTS_EFFECT, NFR_ACHIEVEMENTS_NOTIFICATION
- **Steps:**
    1. Start the game.
    2. Accumulate over £500,000 and observe the gameplay window.
    3. Finish the game, and observe the GameOver window.
- **Expected Outcome:** The user should be informed they have unlocked the “Entrepreneur” achievement the moment they have more than £500,000. The GameOver window should mention the achievement and the impact it had on their final score.
- **Actual Outcome:** **PASS - The banner “Achievement Unlocked: Entrepreneur” appeared once the amount of money rose above £500000. It also appeared on the Game Over screen and impacted the final score.**

TC_ACH_02

- **Objective:** Verify that the “I Heart Uni” achievement unlocks correctly.
- **Linked Requirements:** UR_ACHIEVEMENTS, FR_ACHIEVEMENTS_SYSTEM, FR_ACHIEVEMENTS_EFFECT, NFR_ACHIEVEMENTS_NOTIFICATION
- **Steps:**
    1. Start the game.
    2. Keep a level of satisfaction of above 75% for more than 2 minutes and observe the gameplay window.
    3. Finish the game, and observe the GameOver window.
- **Expected Outcome:** The user should be informed they have unlocked the “I Heart Uni” achievement once they have retained a satisfaction level of more than 75% for 2 minutes. The GameOver window should mention the achievement and the impact it had on their final score.
- **Actual Outcome:** **PASS - The banner “Achievement Unlocked: I Heart Uni” appeared once the satisfaction score had remained above 75% for more than 2 minutes. It also appeared on the Game Over screen and impacted the final score.**

TC_ACH_03

- **Objective:** Verify that the “One Hundredth Student” achievement unlocks correctly.
- **Linked Requirements:** UR_ACHIEVEMENTS, FR_ACHIEVEMENTS_SYSTEM, FR_ACHIEVEMENTS_EFFECT, NFR_ACHIEVEMENTS_NOTIFICATION
- **Steps:**
    1. Start the game.
    2. Accumulate a population of over 100 students and observe the gameplay window.
    3. Finish the game, and observe the GameOver window.
- **Expected Outcome:** The user should be informed they have unlocked the “One Hundredth Student” achievement once they have a student population of over 100 students. The GameOver window should mention the achievement and the impact it had on their final score.
- **Actual Outcome:** **PASS - The banner “Achievement Unlocked: One Hundredth Student” appeared once the student count reached 100. It also appeared on the Game Over screen and impacted the final score.**

Leaderboard Tests

TC_LEAD_01

- **Objective:** Ensure that when the game first loads the leaderboard has no entries.
- **Linked Requirements:** UR_LEADERBOARD, FR_LEADERBOARD, NFR_LEADERBOARD_ VISIBILITY
- **Steps:**
    1. Load the game.
    2. Check the leaderboard.
- **Expected Outcome:** The leaderboard should be blank and contain no entries when the game first loads, and inform the user that there are no entries.
- **Actual Outcome:** **PASS - The LeaderboardScreen just displays a back button and a title stating “There are no entries yet”.**

TC_LEAD_02

- **Objective:** Ensure that if the user chooses not to provide their name, their score is not added to the leaderboard.
- **Linked Requirements:** UR_LEADERBOARD, FR_LEADERBOARD, NFR_LEADERBOARD_ VISIBILITY
- **Steps:**
    1. Start the game.
    2. Reach the game over screen and do not provide a name.
- **Expected Outcome:** The player should be able to not provide a name, return to the main menu, and not see their score on the leaderboard.
- **Actual Outcome:** **PASS - The LeaderboardScreen continues to display just a back button and the title stating “There are no entries yet”.**

TC_LEAD_03

- **Objective:** Ensure that if the user provides their name, their score is added to the leaderboard provided their score is higher than other records.

- **Linked Requirements:** UR_LEADERBOARD, FR_LEADERBOARD, NFR_LEADERBOARD_VISIBILITY
- **Steps:**
    1. Start the game.
    2. Reach the game over screen and provide a name.
    3. Ensure the record has been added to the leaderboard.
- **Expected Outcome:** The player's score is added to the leaderboard.
- **Actual Outcome:** **PASS - The player's score and name has been added to the leaderboard.**

TC_LEAD_04

- **Objective:** Ensure that a player's score updates if they play the game and get a better score, assuming they use the same name.
- **Linked Requirements:** UR_LEADERBOARD, FR_LEADERBOARD, NFR_LEADERBOARD_VISIBILITY
- **Steps:**
    1. Start the game.
    2. Reach the game over screen and provide a name.
    3. Repeat steps 1 and 2 again and get a higher score than the original game, and then provide the same name and view the leaderboard.
- **Expected Outcome:** If the player plays with the same name, and receives a higher score, their position and record on the leaderboard gets updated.
- **Actual Outcome:** **PASS - The leaderboard correctly updated the player's record with their new score. It does not with a lower score. It also correctly orders the new record in a situation where the leaderboard has more than one score.**

TC_LEAD_05

- **Objective:** Ensure that the leaderboard can handle more than 5 entries correctly.
- **Linked Requirements:** UR_LEADERBOARD, FR_LEADERBOARD, NFR_LEADERBOARD_VISIBILITY
- **Steps:**
    1. Play the game multiple times to add at least 6 entries to the leaderboard.
    2. Observe how the scores are displayed each time and confirm that only the top 5 scores remain visible.
- **Expected Result:** The leaderboard should only display the top 5 scores and update if there is a new entry that has a better score than an existing entry.
- **Actual Outcome:** **PASS - The leaderboard correctly orders records as they are added in descending order of score, and only displays the top 5 once a sixth record is added.**

Non-Functional Requirement Tests

TC_PERF_01

- **Objective:** Ensure that the gameplay runs smoothly on a wide range of platforms and devices, especially on low-spec devices.
- **Linked Requirements:** NFR_PERFORMANCE, CR_LOW_SPEC
- **Steps:**
    1. Run the game on a wide variety of devices and platforms.

2. Monitor the games performance for the duration of gameplay on each.

- **Expected Outcome:** The game should run smoothly on a wide variety of operating systems and devices and consistently maintain frame rate of at least 30 frames per second, and different menus should load within 5 seconds of being requested i.e. button clicked.
- **Actual Outcome:** **PARTIAL PASS - The game runs smoothly on three different devices; A Linux Laptop, a Windows Laptop, and a MAC Laptop.**

TC_USE_01

- **Objective:** Ensure that first time players can play with minimal tutorials.
- **Linked Requirements:** NFR_OPERABILITY, UR_TIPS
- **Steps:**
    1. Load the game and view the help page.
- **Expected Outcome:** The instructions on the help page should be short enough to read within approx. 1 minute but detailed enough to provide an understanding of the game to a new player.
- **Actual Outcome:** **PASS - The help screen contains text that is easy to read and can be read in less than 1 minute.**