|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Plan ID:** 1 | | **Test Plan Name:** Main Menu Test Cases | | | | | |
| **Test Plan Description:**  Ensure that the intended functionality implemented in the Main Menu Scene acts as intended. | | | | | | | |
| **Pre-Conditions:** Open two clients to enter the GameScene, with one being on the Unity Editor. | | | | | | | |
|  | | | | | | | |
| **Test ID** | **Test Type** | | **Test Description** | **Purpose of the Test** | **Expected Outcome** | **Actual Outcome** | **Status (Pass, Failed, Deferred)** |
| 1. | Visual UI | | Check the visual components of the **MainMenuUI** **Canvas** *displays* correctly. | UI components for the **MainMenuUI** should be correctly *displayed*. | Displays:   * GameTitle * Play Button * Options Button * Quit Button | All *four* components are visually *displayed* on the scene as UI components. | Pass |
| 2. | Button Functionality | | Clicking the **Play Button** changes the UI canvas to the **PlayMenuUI**. | Ensures that the UI components correctly *display* when transitioning from the **MainMenuUI** to **PlayMenuUI**. | *Clicking* the **Play button** will *‘hide’* the **MainMenuUI** and *‘show’* the **PlayMenuUI**. | Clicking the **Play button** *displays* the **PlayMenuUI**. | Pass |
| 3. | Visual UI | | Check the visual components for **PlayMenuUI** **Canvas** *displays* correctly. | UI components for the **PlayMenuUI** should be correctly *displayed*. | Displays:   * Create Session Button. * Create Session Text Input * Join Session Button * Join Session Text Input | All *four* components are visually *displayed* on the scene as UI components. | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4. | Button Function | Buttons for **Create Session** and **Join Session** can be interacted with | Check buttons can be interacted with by Players | Clicking the **Create Session Button** displays a debug log of ‘*Create Session Clicked’*.  Clicking the **Join Session Button** displays a debug log of *‘Join Session Clicked’*. | *‘Create Session Clicked*’ is displayed on the *console* when **Create Session** is clicked.  *‘Join Session Clicked’* is displayed on the *console* when **Join Session** is clicked. | Pass |
| 5. | Text Input Functionality | Text input fields can be interacted with to input text via keyboard. | Ensures characters are correctly *displayed* when attempting to input text into the text field. | Typing in *‘H3ll0’* in the **Create Session Text Input** *displays* *‘H3ll0’*.  Typing in *‘H3ll0’* in the **Join Session Text Input** *displays* *‘H3ll0’*. | *‘H3ll0’* is *displayed* in **Create Session Input Field** when *‘H3ll0’* is typed in.  *‘H3ll0’* is *displayed* in **Join Session Input Field** when *‘H3ll0’* is typed in. | Pass |
| 6. | Text Input Functionality | You can correctly *copy* and *paste* text from input fields. | Ensures that input fields allow for text to be *copied* if text is contained inside. | Copying a typed input field of *‘Hello’* in **Create Session Text Input** will return *‘Hello’* when pasted in **the Join Session Input Field**, and vice versa. | *‘Hello’* is returned when pressing CTRL-C for the text field for **Create Session Input Field** and then CTRL-V for the **Join Session Input Field**. The opposite is also true. | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7. | Text Input Validation | Input Fields *rejects* less than *5* characters and more than *8*. | Restrict the Session keys to be in the range of *5-8* characters long. | Input Field *returns a validation error* when clicking the Create Session Button with *1, 2, 3, 4, or 9* characters in the Input Field. | Typing in *‘a’* in **Create Session Input Field** generates a red validation message under the **Create Session Input Field** when clicking the **Create Session Button**. This also applies to *‘aa’, ‘aaa’*, and *‘aaaaa’*. Same applies for the **Join Session Input Field**.  It is impossible to type more than *8* characters in both Input Fields. | Pass |
| 8. | Text Input Validation | Input Fields can take between *5 to 8* characters. | Restrict the Session keys to be in the range of *5-8* characters long. | Input Field doesn’t return a validation error when clicking the Create Session Button with *5, 6, 7, 8* characters in the Input Field. | Typing in *‘aaaaa’* in **Create Session Input Field** doesn’t generate red validation when *clicking* the **Create Session Button**. This also applies to *‘aaaaaa’, ‘aaaaaaa’*, and *‘aaaaaaaa’*. Same applies for the **Join Session Input Field**. | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 9. | Text Input Validation | A red Text Field for validation messages is displayed when the session key of an Input Field is not in the range of *5-8* characters. | *Displays* correct validation message to inform Players of the validation issue. | Typing *‘aaaa’* in the **Create Session Input Field** and clicking the **Create Session Button** returns a *‘MUST BE 5 – 8 CHARACTERS’* validation message underneath the **Create Session Input Field**. | *‘MUST BE 5 – 8 CHARACTERS’* is *displayed* when attempting to create a session using *‘aaaa’*. Same applies for Join Session as well. | Pass |
| 10. | Text Input Validation | A red Text Field for validation messages is displayed for Create Session when a room already exists using the Input Field text value as the key. | Prevents a session from being created if a room already exists using the same key as the Input Field. This enforces unique session keys for Players. | Using *‘aaaaa’* as a create session key generates a *‘ROOM CANNOT BE CREATED’* validation message when another client has already created a session with *‘aaaaa’*. | *‘ROOM CANNOT BE CREATED’* is returned as a validation message when two or more clients use *‘aaaaa’* as the create session key. | Pass |
| 11. | Text Input Validation | A red Text Field for validation messages is displayed for Join Session when using a session key that doesn’t have a corresponding session. | Prevents Players from joining sessions that do not exist. | Typing *‘aaaaa’* as a join session key returns *‘INVALID SESSION’* validation message if there exists no session with the key *‘aaaaa’*. | *‘INVALID SESSION’* is returned when using *‘aaaaa’* without a *‘aaaaa’* session already existing. Similarly, the same validation message appears when a room with the session key exists but is full due to two clients who have already joined the room. | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 12. | Button Functionality | Clicking the **Option Button** in **MainMenuUI** changes the UI **Canvas** to the **OptionsMenuUI**. | Ensures that the UI components correctly display when transitioning from the **MainMenuUI** to **OptionsMenuUI.** | Clicking the **Play Button** *‘show’* the shows the **OptionsMenuUI.** | Clicking the **Options button** *displays* the **OptionsMenuUI** as a pop-up. | Pass |
| 13. | Visual UI | Check the visual components for **OptionsMenuUI** **Canvas** displays correctly. | UI components for the **OptionsMenuUI** should be correctly displayed. | Displays:   * Options Title * Close Icon Button * Fullscreen Label * Fullscreen Toggle * Resolution Label * Resolution Dropdown * Volume Label * Volume Slider * Apply Button | All *nine* components are visually displayed on the scene as UI components. | Pass |
| 14. | Button Functionality | Clicking the **Close Button** or the **Apply Button** will hide the **OptionsMenuUI.** | The **Close** or **Apply Button** being clicked should remove the options pop-up as intended. | Clicking the Close Button should hide **OptionMenuUI.**  Clicking the **Apply Button** should hide **OptionMenuUI.** | Clicking close, closes the options pop-up.  Clicking apply, closes the options pop-up. | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 15. | Toggle Function | Clicking the **Toggle** for fullscreen; toggles between windowed mode and fullscreen mode based on current application state. | Toggling on fullscreen mode should change the application display to fullscreen. Whilst toggled off should oppositely change the application display to windowed mode. | Clicking the **Fullscreen Toggle** whilst it is currently false will convert the application to fullscreen mode.  Clicking the **Fullscreen Toggle** whilst it is currently true will convert the application to windowed mode. | When the **Fullscreen Toggle** is checked, fullscreen mode is enabled.  When **Fullscreen Toggle** is not checked, it defaults to windowed mode. | Pass |
| 16. | Dropdown Function | Clicking the **Resolution Dropdown** *displays* a list of resolutions native to the device. | The **Dropdown** menu should be populated with fields for resolution for changing application resolution. | Clicking the **Dropdown** menu displays a list of resolutions for the client device. | The **Dropdown** displays a filtered list of resolutions for their given device frame rate when clicked. | Pass |
| 17. | Dropdown Functionality | Clicking an option from the list of resolutions in the **Dropdown** menu will change the application resolution to the one chosen. | Player can change resolution of the application based on the **Dropdown** item that was selected. | Clicking *1600x900* in the Dropdown menu, will change the initial *1920x1080* resolution to 1600x900. | Application resizes to *1600x900* from initial *1920x1080* when selecting *1600x900* from the Dropdown list items. | Pass |
| 18. | Slider Functionality | Dragging the handle of the **Volume Slider** will reposition the visual value of the **Slider.** | Ensures that the Player can interact with the slider as intended. | When the **Volume Slider** is initially set to max (right most), it is possible to drag it the min (left most). | Able to drag the handle from right to left, thus it’s possible to change the slider value. | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 19. | Slider Functionality | Changing the **Slider** value can change the volume of the **AudioMixer object**. | Player can change the master volume of the game by dragging the **Slider** value to a volume best suiting them. | When **Volume Slider** is initially max (right most), dragging the Slider handle left, will decrease the AudioMixer volume. | Dragging the handle left and right currently does nothing because there is no audio or **AudioSource** for the **Main Menu Scene.** | Deferred |
| 20. | Button Functionality | Clicking the **Quit Button** closes the application. | Player can close the application by pressing the **QuitButton** in the **MainMenuUI.** | Clicking the **Quit Button** in the **MainMenuUI** exits the game. | The application is closed when clicking the **Quit Button**. | Pass |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Plan ID:** 2 | | **Test Plan Name:** Pause & Options Menu Test Cases | | | | | |
| **Test Plan Description:**  Ensure that the intended functionality implemented in the PauseMenuUI in the Game Scene acts as intended. | | | | | | | |
| **Pre-Conditions:** Open two clients to enter the GameScene, in a playable state. | | | | | | | |
|  | | | | | | | |
| **Test ID** | **Test Type** | | **Test Description** | **Purpose of the Test** | **Expected Outcome** | **Actual Outcome** | **Status (Pass, Failed, Deferred)** |
| 1. | Visual UI | | Check the visual components of the **PauseMenuUI Canvas** displays correctly. | UI components for the **PauseMenuUI** should be correctly displayed. | Displays:   * Resume Button * Options Button * Give Up Button * Quit Button | All *four* components are visually displayed on the scene as UI components in a pop-up. | Pass |
| 2. | Button Functionality | | Clicking the **Resume Button** hides the UI **Canvas** of **PlayMenuUI.** | Ensure that the **PauseMenuUI** is not displayed when clicking resume, so that the Player can return to the game. | Clicking the **Resume Button** hides the **PlayMenuUI.** | The **PauseMenuUI** pop-up is not displayed when the **Resume Button** is clicked. | Pass |
| 3. | Button Functionality | | Clicking the **PlayButton** shows the **OptionsMenuUI**. | Ensure that the **OptionsMenuUI** is displayed when clicking options, so that the Player can open the **OptionsMenuUI**. | Clicking the **Options Button** displays the **OptionsMenuUI**. | A pop-up for **OptionsMenuUI** is displayed when the **Options Button** is clicked. | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4. | Visual UI | Check the visual components for **OptionsMenuUI Canvas** displays correctly. | UI components for the **OptionsMenuUI** should be correctly displayed. | Displays:   * Options Title * Close Icon Button * Fullscreen Label * Fullscreen Toggle * Resolution Label * Resolution Dropdown * Volume Label * Volume Slider   Apply Button | All nine components are visually displayed on the scene as UI components. | Pass |
| 5. | Button Functionality | Clicking the **Close Button** or the **Apply Button** will hide the **OptionsMenuUI.** | The **Close** or **Apply Button** being clicked should remove the options pop-up as intended. | Clicking the **Close Button** should hide **OptionMenuUI**.  Clicking the **Apply Button** should hide **OptionMenuUI.** | Clicking close, closes the options pop-up.  Clicking apply, closes the options pop-up. | Pass |
| 6. | Toggle Function | Clicking the Toggle for fullscreen; toggles between windowed mode and fullscreen mode based on current application state. | Toggling on fullscreen mode should change the application display to fullscreen. Whilst toggled off should oppositely change the application display to windowed mode. | Clicking the **Fullscreen Toggle** whilst it is currently false will convert the application to fullscreen mode.  Clicking the **Fullscreen Toggle** whilst it is currently true will convert the application to windowed mode. | When the **Fullscreen Toggle** is checked, fullscreen mode is enabled.  When **Fullscreen Toggle** is not checked, it defaults to windowed mode. | Pass |
| 7. | Dropdown Function | Clicking the **Resolution Dropdown** displays a list of resolutions native to the device. | The **Dropdown** menu should be populated with fields for resolution for changing application resolution. | Clicking the **Dropdown** menu displays a list of resolutions for the client device. | The **Dropdown** displays a filtered list of resolutions for their given device frame rate when clicked. | Pass |
| 8. | Dropdown Functionality | Clicking an option from the list of resolutions in the **Dropdown** menu will change the application resolution to the one chosen. | Player can change resolution of the application based on the **Dropdown** item that was selected. | Clicking *1600x900* in the **Dropdown** menu, will change the initial 1920x1080 resolution to 1600x900. | Application resizes to *1600x900* from initial *1920x1080* when selecting *1600x900* from the **Dropdown** list items. | Pass |
| 9. | Slider Functionality | Dragging the handle of the **Volume Slider** will reposition the visual value of the **Slider**. | Ensures that the Player can interact with the slider as intended. | When the **Volume Slider** is initially set to max (right most), it is possible to drag it the min (left most). | Able to drag the handle from right to left, thus it’s possible to change the slider value. | Pass |
| 10. | Slider Functionality | Changing the **Slider** value can change the volume of the **AudioMixer object**. | Player can change the master volume of the game by dragging the **Slider** value to a volume best suiting them. | When **Volume Slider** is initially max (right most), dragging the Slider handle left, will decrease the AudioMixer volume. | The music volume is slowly decreased as the max volume is reduced via dragging the **Slider** handle to the left (min). | Pass |
| 11. | Button Functionality | Clicking the **Give Up Button** hide the **PauseMenuUI** and display the **GameOverUI.** | When a Player gives up, a game over screen should be generated for the winner and loser. | Clicking the **Give Up Button** should show the **GameOverUI**, whilst hiding the **PauseMenuUI.** | The **LoseScreen** is displayed when clicking the **Give Up Button**. The **PauseMenuUI** is also hidden. | Pass |
| 15. | Button Functionality | Clicking **the Quit Button** closes the application. | Player can close the application from the **PauseMenuUI** by clicking the **Quit Button.** | Clicking the **Quit Button** in the **PauseMenuUI** exits the game. | The application is closed when clicking the **Quit Button.** | Pass |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Plan ID:** 3 | | **Test Plan Name:** TurnSystemUI Test Cases | | | | | |
| **Test Plan Description:**  Ensure that the TurnSystemUI correctly updates visuals when applicable. | | | | | | | |
| **Pre-Conditions:** Open two clients to enter the GameScene, in a playable state. | | | | | | | |
|  | | | | | | | |
| **Test ID** | **Test Type** | | **Test Description** | **Purpose of the Test** | **Expected Outcome** | **Actual Outcome** | **Status (Pass, Failed, Deferred)** |
| 1. | Visual UI | | Check the visual components of the **TurnSystemUI Canvas** displays correctly. | UI components for the **TurnSystemUI** should be correctly displayed. | Displays:   * Turn Counter * Timer * End Turn Button | All three components are visually displayed on the scene as UI components. | Pass |
| 2. | Button Functionality | | Clicking the **End Turn Button** immediately hides End Turn Button. | A Player shouldn’t be able to end the turn of another Player. | Clicking the **End Turn Button** conceals the **End Turn Button.** | **End Turn Button** is not visibility when clicked. | Pass |
| 3. | Button Functionality | | When another Player clicks the **End Turn Button** immediately show **End Turn Button.** | When the other Player ends their turn, you should be allowed to have your turn, and subsequently end your turn. | The **End Turn Button** is shown when the other Player clicks the **End Turn Button** on their client. | When clicking the **End Turn Butto**n on another client, the **End Turn Button** is hidden for the clicker client and is displayed for the other Player. | Pass |
| 3. | Text Fields | | **Turn Counter** correctly displays what Player turn it is, and the turn number. | Players are visibility aware whose turn it currently is, and what number turn they are on, so far in the session. | From immediately after the game starts. Clicking the **End Turn Button**, causes the **Turn Counter** Text to display *‘[Player 2] Turn 1’*, and if clicked again on the other client to display *‘[Player 1] Turn 2’.* | *‘[Player 2] Turn 1’* is displayed when clicking end turn immediately after the game starts, and then changes to *‘[Player 1] Turn 2’*, when clicked once again on the other client. | Pass |
| 4. | Text Fields | | **Timer** counts down every second and reduces the **Timer** value. | Ensures that the **Timer** is being updated every second and is correctly displaying their value in seconds. | Immediately after the game starts, the **Timer Text** displays *120*. After *10* seconds it displays *110*. | The **Timer Text** starts off as *120* at the start of the game and reduces by *1*, every second, and after *10* seconds it reaches *110*, and continues to countdown. | Pass |
| 5. | Text Fields | | **Timer** resets after reaching *0*. | Ensures that the **Timer** resets for the next Player when the timer elapses. | Once the **Timer** reaches *0*, it changes to *120*, after *1* second. | **Timer** immediately changes to *120* after the **Timer** reaches *0*. | Pass |
| 6. | Text Fields | | The **Turn Counter** is updated when the **Timer** reaches *0*. | Ensures that the Player turn is correctly changed when the **Timer** reaches *0*, and the Player turn changes. | From immediately after the game starts, once the **Timer** reaches *0*, the Turn Counter displays *‘[Player 2] Turn 1’.* | *‘[Player 2] Turn 1’* is displayed on both clients when the **Timer** reaches *0*, on the first turn of the game for Player 1. | Pass |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Plan ID:** 4 | | **Test Plan Name:** UnitUI & ToolTips Test Cases | | | | | |
| **Test Plan Description:**  Ensure the the UnitUIs and the ToolTips correctly update visual UI components on the Screen Space. | | | | | | | |
| **Pre-Conditions:** Open two clients to enter the GameScene, in a playable state. | | | | | | | |
|  | | | | | | | |
| **Test ID** | **Test Type** | | **Test Description** | **Purpose of the Test** | **Expected Outcome** | **Actual Outcome** | **Status (Pass, Failed, Deferred)** |
| 1. | Visual UI | | Check the visual components of the **PlayerUnitUI** **Canvas** displays correctly. | UI components for the **PlayerUnitUI** should be correctly displayed. Being aligned to the left-hand side. | Displays:   * Unit Name * Icons for the 6 Unit parameters * Text fields for the 6 Unit parameters * Weapon Type * Armour Type * Player TOTAL\_GOLD Icon * Player TOTAL\_GOLD Text | *Seven* icons are displayed, *6* for the Unit parameter types, and a *7th* for the *TOTAL\_GOLD*. **Text** fields are displayed parallel to the *seven* icons.  Name is also shown at the top of the UI with *Weapon* and *Armour Type* being displayed below the Unit parameters.  These components are displayed as a UI section aligned on the left. | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2. | Visual UI | Check the visual components of the **EnemyUnitUI Canvas** displays correctly. | UI components for the **EnemyUnitUI** should be correctly displayed. Being aligned to the right-hand side. | Displays:   * Unit Name * Icons for the 6 Unit parameters * Text fields for the 6 Unit parameters * Weapon Type * Armour Type * Player TOTAL\_GOLD Icon * Enemy TOTAL\_GOLD Text | *Seven* icons are displayed, *6* for the Unit parameter types, and a *7th* for the *TOTAL\_GOLD*. Text fields are displayed parallel to the seven icons.  Name is also shown at the top of the UI with *Weapon* and *Armour Type* being displayed below the Unit parameters.  These components are displayed as a UI section aligned on the right. | Pass |
| 3. | Visual UI | Check the **UnitUI** background **Colo(u)rs** for Player and Enemy are correctly defined based on the Player and their client. | Helps Player distinguish which Player they are in the session, and what Units they own based on the designated Color based on Player team. | If the Player client is Player 1, the background Color for **PlayerUnitUI** is blue, with the **EnemyUnitUI** being red. The opposite should be true for Player 2. | A *blue* background is shown for Player 1’s **PlayerUnitUI** and the **EnemyUnitUI** being red.  The Player 2 client shows the opposite result. | Pass |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4. | Visual UI | | | | The Text fields for the **UnitUIs** are correctly updated when a new Unit is selected. The Text fields should display the parameters of the newly selected Unit. | Players should be able to identify the statistical parameter values of a Unit by clicking on them, whether they own the Unit or if it belongs to the enemy. | Clicking an **Archer** Unit should display the following parameters in the UnitUI:   |  |  | | --- | --- | | HP | 100 | | ATK | 35 | | DEF | 5 | | MOVE | 2 | | ATK\_RANGE | 2 | | GOLD\_COST | 15 | | WEAPON TYPE | PIERCE | | ARMOUR TYPE | UNARMOURED | | | The initial Unit parameter values of **PlayerUnitUI** of *[100, 30, 10, 3, 1, 15, SLASH]* and Light are replaced with *[100, 35, 5, 2, 2,* *15, PIERCE, UNARMOURED]* when clicking a Player 1 owned **Archer,** as Player 1.  The same occurs when clicking a non-Player 1 owned **Archer** from a non-Player 1 owned **Militia** for **EnemyUnitUI**. | | Pass |
| 5. | Visual UI | | | | The HP value is reduced in the **UnitUI** when a selected Unit takes damage. | Ensures that the HP parameter values for **UnitUI** is always accurate for Players. | As Player 1, with a Player owned **Militia** selected and shown on the **PlayerUnitUI** (at *100* health). When said Militia is attacked by an Enemy **Militia,** update the **PlayerUnitUI** to display *65*. | | *100* reduces to *65* on the **PlayerUnitUI**, when a selected Militia at full HP takes damage from an Enemy **Militia**.  The **EnemyUnitUI** also updates this the HP to *65*, if Player *2* has also selected the Player 1 owned **Militia.** | | Pass |
| 6. | Visual UI | | | | If a Unit that has been displayed on the **UnitUI** for either Player 1 or 2 dies, hide the **UnitUI** displaying the Unit parameters of the dead Unit. | All references of a dead Unit should be destroyed, including UI components. This prevents erroneous data and bugs from occurring, (e.g. displaying *-3 HP*). | With a Player 1 owned **Militia** selected and displaying on one of the **UnitUIs.** Hide said **UnitUI** when the mentioned **Militia** Unit is killed when attacked by an Enemy Unit. | | The **PlayerUnitUI** is hidden as Player 1, when the **Militi**a Unit dies.  For Player 2, the **EnemyUnitUI** is hidden instead. | | Pass |
| 7. | Visual UI | | | | The *TOTAL\_GOLD* Text fields for both Players are updated when a turn ends. | **UnitUIs** should be updated to reflect the generation (*25* per turn) of Gold when a turn ends. | During *‘[Player 1] Turn 1’* without performing any actions. When Player 1 ends the turn, the *TOTAL\_GOLD* for **PlayerUnitUI** and **EnemyUnitUI** increases from *25* to *50*. | | The *TOTAL\_GOLD* Text field changes from *25* to *50*, when Player 1 clicks the **End Turn Button** after the game starts.  Similarly, when Player 2 subsequently ends their turn without any actions, the *TOTAL\_GOLD* increases from *50* to *75* for both **UnitUIs** on both clients. | | Pass |
| 8. | Visual UI | | | | Hovering over a Text or Image (icons) field for **PlayerUnitUI** for either clients will generate a visual pop-up on the lower right-hand side of the screen. | ToolTips are displayed for contextual help when hovering of fields in the **UnitUI.** | When mousing over any of the following fields will generate a ToolTip pop-up.   * Unit name * HP icon * ATK icon * DEF icon * MOVE icon * ATK\_RANGE icon * GOLD\_COST icon * WEAPON TYPE * ARMOUR TYPE * TOTAL\_GOLDicon | | A pop-up is displayed for the client of the Player who hovers over any of the mentioned Image or Text fields.  ToolTip pop-ups appears to be independent of the client rather than shared, which is intended. | | Pass |
| 9. | Visual UI | | | | The ToolTip text is correctly displaying the corresponding description of the hovered over field. | Players are given Tooltip descriptions based on what field they hovered over. | When Player 1 hovers over the HP icon, the ToolTip should display the following:  *‘The amount of HP a Unit has. Having 0 HP results in the Unit dying’* | | *‘The amount of HP a Unit has. Having 0 HP results in the Unit dying’*, is shown on the ToolTip Text field of the ToolTip pop-up for Player 1. Same occurs for Player 2 but on their client instead. | | Pass |
|  |  | | | |  |  |  | |  | |  |
| **Test Plan ID:** 5 | | | **Test Plan Name:** UnitShopUI Visual Test Cases | | | | | | | | |
| **Test Plan Description:**  Ensure that visual components of the UnitShopUI are correctly updated, based on Player interactions. | | | | | | | | | | | |
| **Pre-Conditions:** Open two clients to enter the GameScene, in a playable state, with one of the clients being opened in the Unity Editor. | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **Test ID** | | **Test Type** | | **Test Description** | | **Purpose of the Test** | | **Expected Outcome** | **Actual Outcome** | **Status (Pass, Failed, Deferred)** | |
| 1. | | Visual UI | | Check the visual components of the **UnitShopUI** **Canvas** displays correctly. | | UI components for the **UnitShopUI** should be correctly displayed. Being aligned at the bottom-centre of the screen as a pop-up. | | Displays:   * Close Button * Visual icon for Militia * Visual icon for Archer * Visual icon for Light Cavalry * Visual icon for Catapult * Visual icon for Heavy Cavalry | A **Close Button** is displayed at the top-centre of the **UnitShopUI** pop-up. Underneath containing the icons of the *six* respective Unit types mentioned.  This applies for both Players. | Pass | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2. | Button Functionality | The **Close Button** hides the **UnitShopUI** pop-up when clicked. | Allows for the Player to conceal the **UnitShopUI** whenever they desire. | With the **UnitShopUI** opened, clicking the Close Button will close the pop-up. | Pop-up is not shown when the **Close Button** is clicked.  This appears to be independent of each client, so Player 1 can’t close the **UnitShopUI** of Player 2 from their screen. | Pass |
| 3. | Button Functionality | Selecting a Player owned **Fortress** will open the **UnitShopUI** pop-up. | **UnitShopUI** can be accessed from the **Fortress** Unit where the Player will be able to purchase (summon) additional Units to control. | As Player 1, when selecting a Player 1 owned **Fortress**, the **UnitShopUI** should be displayed as a pop-up appearing at the bottom-centre of the screen. | A **UnitShopUI** is shown when clicking the **Fortress** owned by Player 1, as Player 1.  This does not occur when clicking the **Fortress** of Player 2. This is intended, as Player 1 shouldn’t be able to summon Units from Player 2s **Fortress**. | Pass |
| 4. | Button Functionality | Clicking one of the Unit icons (Buttons), will set the Unit as selected in the **UnitShopUI.** | Players are able to select what type of Unit they wish to summon from the available list of Units. | On the *Unity Editor* client, when clicking the Catapult icon, display a debug log stating, *‘Catapult Set’.* | *‘Catapult Set’* is shown in the console, when clicking on the Catapult icon in **UnitShopUI**.  This also occurs for Player 2, when they are the one selecting it. | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5. | Visual UI | Selecting a Unit in the **UnitShopUI** updates the outline visual Color. | Players are visually aware of what Unit has been or is currently selected in the **UnitShopUI**. | With **Archer** currently not selected (white) in the **UnitShopUI**, clicking **Archer** will turn the outline Color to red. | The outline Color changes to red when **Archer** is clicked when originally not selected.  This also occurs for Player 2, when they are the one selecting it. | Pass |
| 6. | Visual UI | Hovering over a Unit icon in the **UnitShopUI** updates the **PlayerUnitUI** Text fields. | Players can view the statistical parameters of each Unit so that they are aware of what Unit they are purchasing as well as their cost. | With a **Militia** currently displayed in the **PlayerUnitUI**. When hovering over the **Light Cavalry** icon, display the following text values.   |  |  | | --- | --- | | Name | Light Cavalry | | HP | 100 | | ATK | 40 | | DEF | 10 | | MOVE | 4 | | ATK\_RANGE | 1 | | GOLD\_COST | 25 | | WEAPON TYPE | PIERCE | | ARMOUR TYPE | LIGHT | | The **PlayerUnitUI** is updated to display *[Light Cavalry, 100, 40, 10, 4, 1, 25, PIERCE, LIGHT]* when the **Light Cavalry** icon is hovered over.  This occurs for both Players respectively on their own clients. | Pass |