Designing Cloud-Based Gameplay Automation: Exploratory Software Testing, Game State-Analysis, and Test-Driven Development (TDD) Applied to Robotic Process Automation (RPA)

David Lebertus Andrade
Northcentral University (National University), USA

Abstract

Robotic Process Automation (RPA) technology has become a popular way to automate repetitive tasks. This exploratory study looks at how RPA could be applied to the automation of real-time cloud-based gameplay, interfaces, and scenarios. The techniques used to design a demonstration were drawn from Test Driven Development (TDD), exploratory software testing, and game-state analysis. The resulting use-case demonstrates how to design, implement, and test the automation of cloud-based gameplay in practice by utilizing cloud-based gaming platform Google Stadia, Microsoft Power Automate RPA, and a popular arcade style fighting game (Mortal Kombat 11). When broken down, gameplay consists of a series of states and state transitions [1]. Diagraming automatable scenarios can be effectively used to produce a model for understanding and automating in-game tasks. Just as software testers find patterns when creating test cases or testing paths, games often have these same repeatable sequences that can be identified and understood and have limited variability of scenarios as they have already been mapped out as part of the general testing before the software was released [2]. Thus, by applying exploratory testing and game-state analysis the patterns can be easily captured and reproduced in the RPA automation and applied to real-time gameplay. An analysis of the use-case indicated directions for future research that included: a) adding more automations paths to execute other gameplay scenarios, b) developing enhanced error handling and decision processing to avoid state-lock, c) evading known anti-cheat systems, and d) the implementation of adversarial Artificial Intelligence (AI) and Machine Learning (ML).

1. Introduction

Gameplay consists of a series of states and state transitions [1]. Many of these same principals are applied to the automation of software and user interface testing used in business software testing. In fact, in Exploratory Software Testing, applying scenario-based exploratory testing is one technique used by software testers because it mimics the way that a real user would behave [2]. However, instead of testing the software randomly to find bugs, this exploratory study discussed automating certain scenarios that contain very specific sequences of actions just as software testers would find when they create their test cases or testing paths. Just as Whittaker describes, "certain actions make sense to repeatedly test", and cloud-based games often have these "repeatable sequences that can be identified and understood and have a limited variability of scenarios because they had already been mapped out as part of the general testing scenarios before the software was released" [2]. These states in a game are observable and can be thought of as the different steps in a game's progression, such as "start", "play", "pause", "win", and "lose" [1]. Each state has its own set of conditions that must be met for a state transition to occur [1]. For example, to win a game, a player must first reach the end goal after which the gameplay transitions to the next game-state are invoked, like a cut scene, credit scene, sub-menu, or a main menu appearing. Often some kind of player input is required to get to the next state in the game and continue the game with a new set of conditions for gameplay.

Furthermore, the Test-Driven Development (TDD) technique of unit testing can also be utilized in the design of Robotic Process Automation (RPA), by testing for specific outcomes from the scenarios that are candidates for automation [4]. Thus, when applied to gaming, testing for resulting states that often occur in a game like "win" or "lose" can be the foundation for designing more complex and detailed automation code in the RPA. Automation can then be designed around the test cases to provide proper input to allow the game to continue or start over depending on the states that were reached, i.e. the end state was "lose", so the game now requires input to continue like "press 'a' to continue". Software with a known set of states and possible transitions between states is a strong candidate for this type of technique for Individual Specification Based Software Testing [3]. Using state transition diagrams allows a clear portrait of all the testing paths that must be exercised [3].

Diagraming automatable scenarios were used to produce a system for understanding and automating in-game tasks. By identify lower-level challenges for implementation, like challenges that carry rewards for the player when they are completed, was how those scenarios were chosen to be automation candidates. The cost of the time spent to build the automation vs

 | DOI: 10.20533/ijicr.2042.4655.2022.0137

the reward that the player normally receives for completing those scenarios was another way to determine which path to automate. There was the possibility that some of the pressure would be taken off the player who would otherwise have to perform the manual completion of these states to achieve specific goals in the game and, if successful, validate the time spent on building the automation with RPA. Therefore, this study demonstrated how breaking gameplay scenarios into repetitive tasks, applying TDD, exploratory testing, and game state-analysis for testing task-based scenarios were applied to the utilization of RPA to automate cloud-based gameplay demonstrated on cloud-based gaming platform Google Stadia, an arcade-style fighting game called Mortal Kombat 11 and Microsoft Power Automate, RPA.

2. Robotic Process Automation

Robotic Process Automation (RPA) is a technology that has been gaining popularity in recent years to automate repetitive tasks. RPA can be used for a wide variety of applications, from simple data entry to more complex process automation. This study investigated one area where RPA could potentially be utilized and that is for automating cloud-based gaming platform gameplay with RPA.

Most traditional gameplay typically requires the player to input a series of commands to navigate menus or perform movement within the game environment. This input can be very repetitive and time-consuming, particularly for skilled players who want to execute more complex sets of commands or reach more challenging levels later in the game. The outcome of a player executing these repetitive challenges are often some kind of reward, “rank”, “level-up”, “points”, “tokens”, or “experience(xp)” for performing a task, move, or macro-task within the game-state engine that is essentially a loop of obtainable progress from the repetition of the task being initiated, i.e. digging for gold, braking bricks to get to a secret level, or exploring a labyrinth with hidden rewards.

However, there are some potential challenges with using RPA for these purposes, let’s consider two things. Firstly, RPA is sophisticated with the right attention to detail but out of the box is not yet sophisticated enough to perfectly mimic human input in real-time, so there may be some errors in the execution of moves in a gaming if being automated without any testing or understanding of what is needed to accomplish the automation candidate process. Secondly, RPA would be reliant on the game-state itself remaining unchanged - if the game is updated or patched, or if there are a large amount of unforeseen input required and/or the baseline of the game-state is altered enough from the original baseline path then the RPA bot may no longer work correctly because it is being used to observe the state and then perform a task based on that task’s state’s original specified context. i.e. if the words appear on the screen “Press Enter to Continue” then the RPA bot will need to send “Enter” to progress to the next state. However, if the game input changes each time the state is encountered then it would require additional programming or attention to cover all of these test-case scenarios to provide adequate and reliable coverage in the event of changes. For example, if you have a 4-button controller that can input “A”, “B”, “C”, and “D” and depending on the result of the previous state the RPA needs to read “Press A to Continue” or “Press D to Continue”, recognize the state and apply the correct automation input either A or D.

RPA shows high potential in this area with the availability of computer vision and screen input readers to be applied to automating certain types of gameplay processes by recording the input of a human player and then playing it back at high speeds. Even having the feature in some RPA platforms to add Artificial Intelligence and Machine Learning to the automations makes this area of RPA a good area for further research. This could allow a player to automate non-repetitive gameplay states or even execute complex combinations without having to input any commands themselves. However, a much simpler approach to the automation of gameplay was explored in this study, focused on simple game-states and transitions as path-testing and looping through desired state operations to determine the automation paths taken to keep the game transitioning from state to state.

Overall, RPA has the potential to be a useful tool for automating gameplay in cloud-based games. It is also important to consider the limitations of the technology before using it for this purpose. This study explored how cloud-based platform gameplay could be automated using game state testing and Robotic Process Automation RPA tools, specifically with Microsoft Power Automate.

3. Microsoft Power Automate

Microsoft Power Automate is a low-code no-code platform that enables users to create automated workflows and processes [5]. With Power Automate, users can quickly and easily automate repetitive tasks, freeing up time for more important work [6]. Power Automate includes many powerful features, such as Robotic Process Automation (RPA), which can automate complex tasks and processes [5]. RPA is particularly useful for businesses that need to automate high-volume or time-consuming tasks. Overall, Microsoft Power Automate’s RPA is an extremely valuable tool for businesses of all sizes. It can help businesses save time and money by automating repetitive tasks and processes, freeing up

 | DOI: 10.20533/ijicr.2042.4655.2022.0137

human resources to turn their attention on more high value and more complex tasks. Additionally, Power Automate's RPA features can further boost efficiency by automating complex tasks that require a relatively lower level of effort or are repetitive in nature but produce a predictable set of results. There are a few different approaches that can be taken when it comes to automating business processes. One option is to use programmable RPA, which involves using software development style programming patterns to automate tasks that would otherwise be performed by humans. Another option is to use a low-code or no-code RPA platform like Microsoft Power Automate. With Power Automate, a users can create automation paths that include UI and interface interactions, without having to write any code, thus it being called a low-code-no-code solution [5].

Finally, creating automation paths in Microsoft Power Automate can help save time and improve efficiency by automating repetitive tasks. The demonstration in this study was made using Microsoft Power Automate to handle the automation of game-state transitions and testing gameplay paths running on the Cloud Gaming Platform by Google, Stadia.

4. Cloud Gaming Platform-Google Stadia

Google Stadia is a cloud gaming platform that allows you to play video games on any device with an internet connection [7]. Cloud gaming platforms have been gaining in popularity in recent years, as they offer gamers a more convenient and affordable way to play their favorite games without having to buy an expensive physical video game console. They also allow for the players to access their games from any computer or mobile device with an adequate internet connection. There are now several different cloud gaming services available, each with their own personalized technology, exclusive games, and user experience built in. Four of the most popular cloud gaming platforms include Microsoft's Xbox Cloud Gaming service, PlayStation's PS Now, Nvidia's GeForce Now, and Google Stadia.

Google Stadia was launched in November of 2019 and, at the time of this study, was available in 14 countries [7]. Google Stadia uses streaming technology to deliver games to any device, so a player can start playing instantly without having to download or install anything [8]. Players can also save their progress in the cloud and pick up where you left off on any other supported device [8]. Google Stadia supports 4K High-Dynamic-Range HDR gaming at 60 Frames Per Second FPS, and will eventually support 8K HDR gaming at 120 FPS [8]. The platform currently has over 30 games available, including popular titles like Red Dead Redemption 2, Assassin's Creed Odyssey, and Borderlands 3. Google Stadia was constantly adding new games and features, so there was always something new to explore with Google Stadia [7]. However, on September 29th of 2022, Google announced that it would be discontinuing the Stadia service and shutting down the servers January 18th, 2023 [9]. This study demonstrated the automation of game-state transitions by utilizing the Google Stadia platform to play a popular arcade style fighting game called Mortal Kombat 11, but theoretically the same concept could be applied to any cloud-based gaming platform or game.

5. Mortal Kombat 11

Mortal Kombat 11 is an arcade style fighting video game developed by NetherRealm Studios [10], published by Warner Bros [11]. Interactive Entertainment. It is the eleventh installment in the Mortal Kombat franchise and was released on April 23, 2019 [11]. Mortal Kombat 11 was the first game in the series to use Unreal Engine 4 [11], the 4th generation computer graphics game engine developed by Epic Games [12]. In Mortal Kombat two players fight against each other using various attacks, including special moves that are triggered by keyboard or controller button inputs. There are 37 playable characters in Mortal Kombat 11, 25 in the base game, 2 unlockable, and 12 in Downloadable Content which includes 5 guest characters from other popular movie franchises licensed by Warner Brothers [11]. Each character with their own unique move sets and fatalities. The game also introduces a system called "Custom Character Variations", which allows players to create their own version of existing fighters with their own special moves and equipment. Mortal Kombat 11 also introduced Custom Character Variations, which allows players to customize their characters' special moves, abilities, and outfits [11]. The game features a single-player story mode, as well as online and offline multiplayer modes, and an extensive story mode campaign that allows for many different outcomes and scenarios. The online multiplayer mode supports both ranked and casual matches, while the offline multiplayer mode supports local play with up to two players.

One of Mortal Kombat 11's most unique features is the AI Battles. AI Battles are essentially single-player matches against computer-controlled opponents. However, these AI Battles are not simply carbon copies of the multiplayer experience. Instead, they feature unique rulesets and objectives that make them distinct from traditional Mortal Kombat gameplay. In AI Battle, a player makes a team of three fighters from the entire roster, and then customize their gear and move sets [13]. Players also can customize which moves the AI will utilize in the fights, with various sliders focusing on Grappling, Combos, Zoning, and Rushdown [13]. For example, one AI Battle may task the player with defeating as many opponents as possible within a certain time

 | DOI: 10.20533/ijicr.2042.4655.2022.0137

limit. Another AI Battle may task the player with surviving against an endless wave of increasingly difficult opponents. There are also AI Battles that focus on specific characters or character types, such as "Only Klassic Fighters" or "Only Female Fighters."

The AI Battles are a great way to practice Mortal Kombat 11's mechanics and learn the ropes of the game. AI Battles provide a unique challenge for experienced players who are looking for something different from the usual multiplayer experience. Whether a player is a Mortal Kombat novice or a seasoned veteran, the availability of AI combatants presents unique automated gameplay scenarios with a variety of modes, customization, and rewards for continued gameplay. These features made the character specific in-game AI a perfect candidate for automating the game-states of Mortal Kombat 11. Specifically, to test if RPA can be applied, since the AI will play the actual arcade style fighting game by producing the adversarial gameplay. Then, the RPA will only need to work about analyzing the game-states and identifying the state transition from one battle to the next, and effectively "keep the game going" just as if a human player were interacting directly with the system. The AI combatants in the Klassic Towers game mode make the perfect candidate for applying game-state testing and applying these advanced automation techniques that we will explore further in the next section, Game State Testing.

6. Game-State Testing

What is gameplay? How does it work? Gameplay consists of a series of states and state transitions [1]. Game-state testing is a type of software testing where the game's state is checked for validity at specific points during gameplay. This can be done manually or automatically. Game state testing is a valuable tool for catching errors early in development [1]. Creating test-cases that exercise different paths through the UI elements can ensure that the game continues to work as intended and that all possible states are reachable [3].

Game states can be thought of as the different steps in a game, such as "start", "play", "pause", "win", and "lose". Each state has its own set of conditions that must be met for the state-transition to occur. For example, to win a game, the player must first reach the end goal by going from "play" to "win" or "lose" before continuing to the next sate and before reaching the "end" state as illustrated in Figure 1.

Figure 1. Example of a simple game state transition

The Mortal Kombat 11 game-state testing was interested in checking if certain states have been reached during gameplay and to let the RPA send the necessary game input based on the specific game-state transition reached. To demonstrate the game's state, test-paths and test-cases are created that exercise different paths through the UI elements to see if a certain state has occurred. If a certain state has occurred, then the automation would continue. In automation it is important to have a very clear understanding of the game's state-machine. The game's state-machine is a representation of all the possible states that the game can be in and the transitions between those states [1]. If a state was missed in the test coverage, then there is a risk that the automation will break and require a human to re-evaluate the state to make the decision on what input to send the game for it to progress.

Creating a successful use-case, making test cases that cause the game to continue its intended path, and recursively scanning the screen for UI Elements, messages, or certain images from the game to see if a state changing event has occurred was key to successfully capture state and allowed the automation to respond accurately with the correct input to continue through the state-transition when those certain scenarios/path conditions existed.

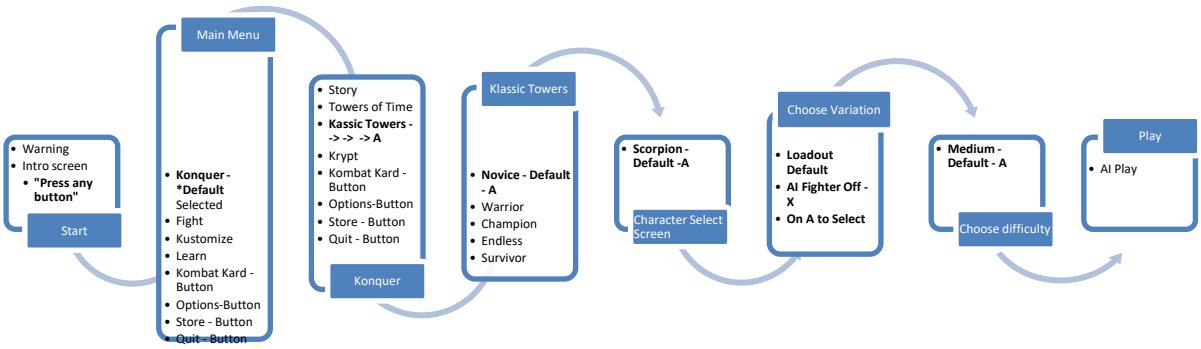


Figure 2. Player Manual Input required to launch "Klassic Tower" with menu options displayed

 | DOI: 10.20533/ijicr.2042.4655.2022.0137

Once in the tower "Play" game-state is reached through manual input the AI character will continue a battle until a "Win" or a "Lose" state is reached. To continue the game Power Automate will need to detect that the "Win" game-state has been reached and pass the correct input in-order for the game to progress to the next match in the tower. This can also be represented graphically using the screen captures from the game in its different observable states as seen below in Figure 3.

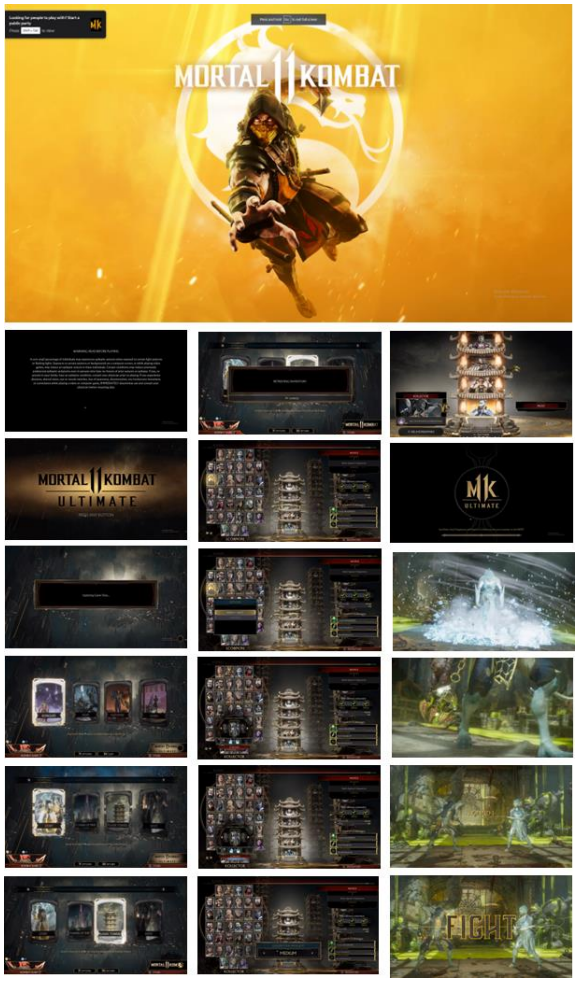


Figure 3. Mortal Kombat 11 game-state and transition image captures

Once the fighting in the tower begins, here is how that state diagram can be represented.

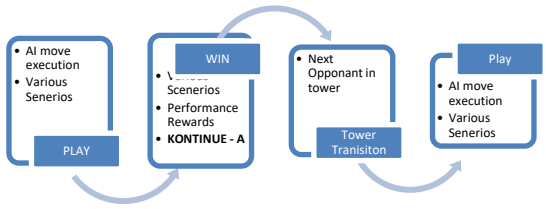


Figure 4. Game States for Play and Win.

Here are some of the corresponding observable states from the gameplay.

At this point in the game-state diagram the loop continues these iterations until the final state was reached which is the "end" state, this meant that the player won both rounds.

However, the automation had to also consider that the AI player might not win. So, the game-state diagram must incorporate the possibility of a loss as seen in Figure 7.

Figure 5. Play, Win, Transition and Play States Observed in gameplay

Figure 6. Round two play state to win state

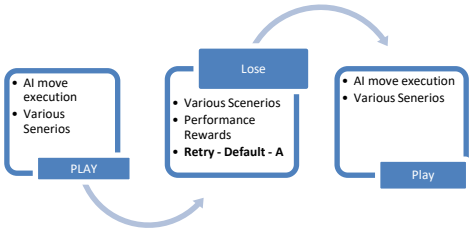


Figure 7. Lose State, reply until a win is obtained

 | DOI: 10.20533/ijicr.2042.4655.2022.0137

In this scenario the automation looped through the states until a win state is reached. This had the potential for becoming an endless loop, but in the use-case it was only interrupted with manual intervention if the AI fighter continued to lose. In order to switch characters and try again, a human player needed to select a different character and attempt the match again. For future research a path could be added that will recognize a certain threshold of losses and automatically go back to the main screen to choose a new character, or custom loadouts. This could be implemented using an accumulator variable that tracks the number of times the "lose" state is reached. Additionally, there are some other unpredictable states that would appear on the game-state transitions like "Congratulations" and "Rewards" however this was be covered in this test-case by setting up the RPA to also look for the word "KONTINUE" just as it would for the Win and Lose transitions and have it send the appropriate input to continue, which was always "Enter". After collecting this information, the next step was to begin building the automation. In the next section, a step-by-step guide is provided on how this basic automation was built.

7. Building the Automation

After identifying the automatable paths for "play" "win" and "lose" on the Klassic Tower gameplay building the automation began. Desktop flows in Microsoft Power Automate interacted with applications' UI elements and are also able to use visual indications to help users quickly recognize desktop and even web UI elements that may appear on the screen [14]. However, to identify the game-states and execute actions in the game it was necessary to identify images that appeared on the screen like menus, buttons, or specific inputs to trigger transitions to the next game state. This was achieved by using the "capture image" and "if image actions" in Power Automate [15]. After images are identified, Power Automate's conditionals, like if/else, can execute the keyboard input needed or any other associated actions that will need to take place to successfully automate a block of actions based on values or states being used [16].

The following figures and descriptions will demonstrate how to build the automation for the scenarios discussed in the previous section, for automating the continuous towers matches using images from the game, and if/else conditions in Power Automate to successfully loop through the game states.

Power Automate will create the Flow and bring you into the Developer Action stage where the automation is built (see Figure 10).

For the first automation function a loop was used. Search Loop on the left-hand Actions pane and select it. Then set "start from:" to 0 and "end to:" to 1000 and "Increment by:" to an integer value of 1. The variables produced are automatically set to "LoopIndex" and will be used as the accumulator for the loop (see Figure 11).

First, launch Power Automate

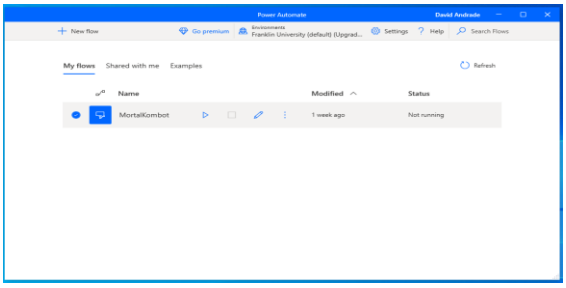


Figure 8. Launching Power Automate

User prompted to enter a Flow name, entered "Mortal Kombot"

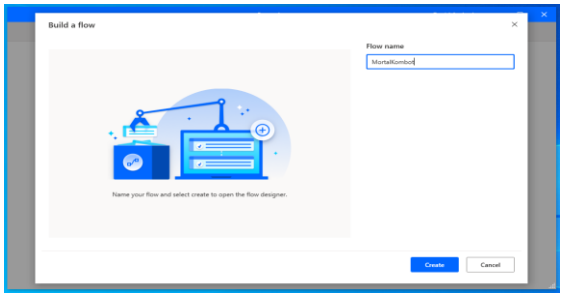


Figure 9. Build a new Flow

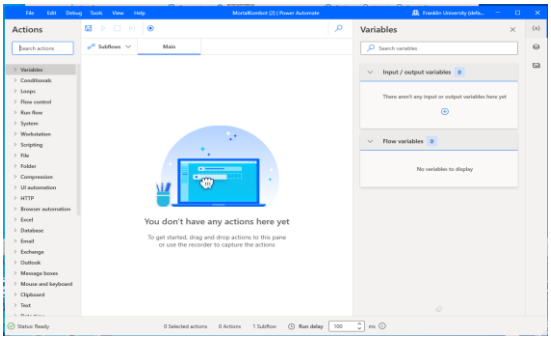


Figure 10. Power Automate Flow Developer Stage

Next, an if statement action is needed to check for images containing the game's UI elements that appear after a "Win" or "Loss" to trigger the automated keyboard input to allow the game to continue. In this case an image of the "KONTINUE" UI element will be used (see Figure 12). Because this element is not the only one that could appear, an alternate form of the KONTINUE UI element image as it appeared when a congratulations state was reached was added for additional test coverage (see Figure 13).

 | DOI: 10.20533/ijicr.2042.4655.2022.0137

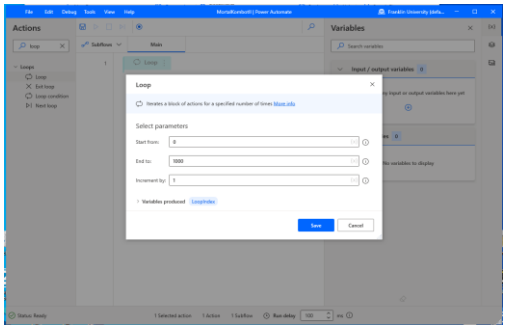


Figure 11. Loop action settings

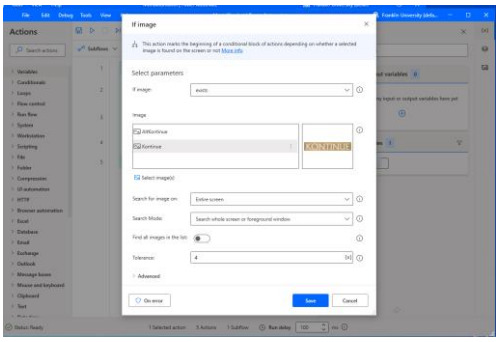


Figure 12. if image action with “KONTINUE” ui element image captured

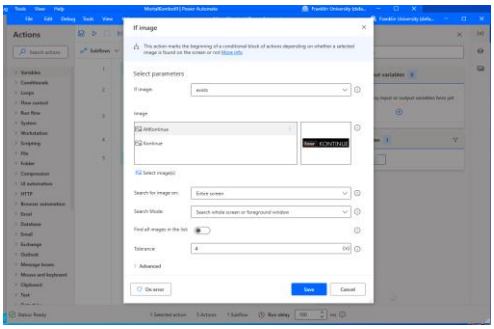


Figure 13. Adding additional image capture to identify when “Congratulations” state is reached

It is also necessary to capture the “RETRY” UI element image in case the “Lose” state was reached and in that case the automation needed to play again until the win state was reached. In that case “Enter” was also the same input so whether “Kontinue” or “Retry” occurred the input expected would always be the same “enter” to progress (see Figure 14).

Additionally, it was verified that the image search is set to search the whole screen, set the tolerance and, in Advanced, the Advanced image matching algorithm was used to enhance Power Automate’s ability to match the UI element being displayed on the screen with higher accuracy [15]. The Basic algorithm achieves better results with images less than 200x200 pixels, while the Advanced algorithm is more effective with bigger images and more robust to color changes [15].

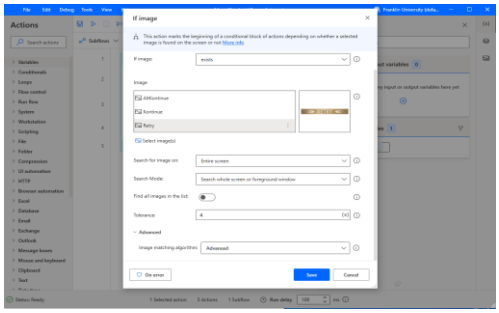


Figure 14. Retry image added to identify that a lose state has been reached

Now the automation had to send the keystrokes on either “Kontinue” or “Retry” appearing on screen. So inside of the If image action a “Send Keys” action was added (see Figure 15).

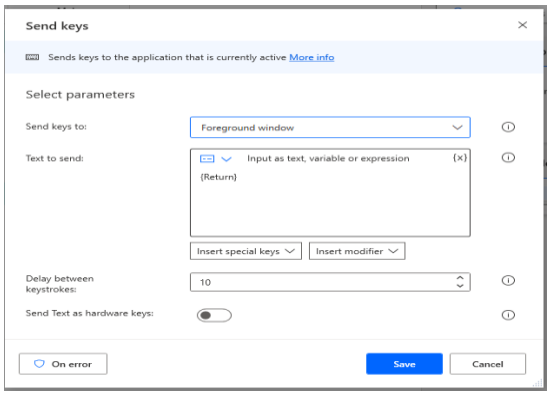


Figure 15. Adding send keys action to simulate keyboard input

To configure the simulation of keyboard input Sent Keys used “{Return}” to simulate someone hitting the Enter key on the keyboard and thus triggering the game state to move to the transition and on to the next state of “play”. Also, the “send text as hardware keys” switch needed to be enabled to mimic the keyboard sending the input rather than the Power Automate Desktop application.

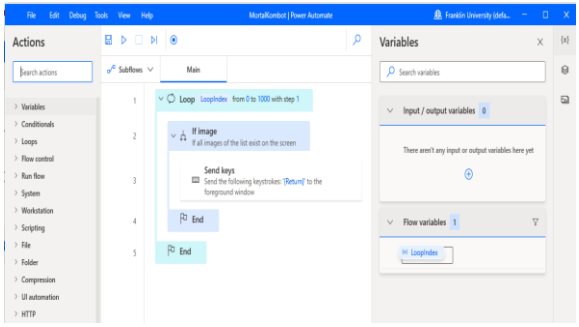


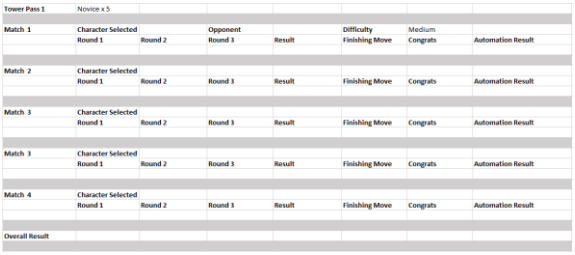
Figure 16. The completed flow

 | DOI: 10.20533/ijicr.2042.4655.2022.0137

The completed flow is given in Figure 16 which was the foundation in allowing the game to progress without the need for the player to interact after the tower selection, character selection, and AI player setting triggered. Thus, simple automation completed our automation for this test-case and allowed for the game to progress every time the player was presented with a prompt that required the player to press the "Enter" key to continue. In the next section we discuss how the automation's performance and reliability was tested.

8. Testing the Automation

Using path-testing tables to capture results of each of the initial tests helped to understand the results from each observable match (see Figure 17).



Tower Pass 1	Novice x 5						
Match 1	Character Selected		Opponent		Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
Match 2	Character Selected						
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
Match 3	Character Selected						
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
Match 3	Character Selected						
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
Match 4	Character Selected						
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
Overall Result							

Figure 17. Path Testing Table

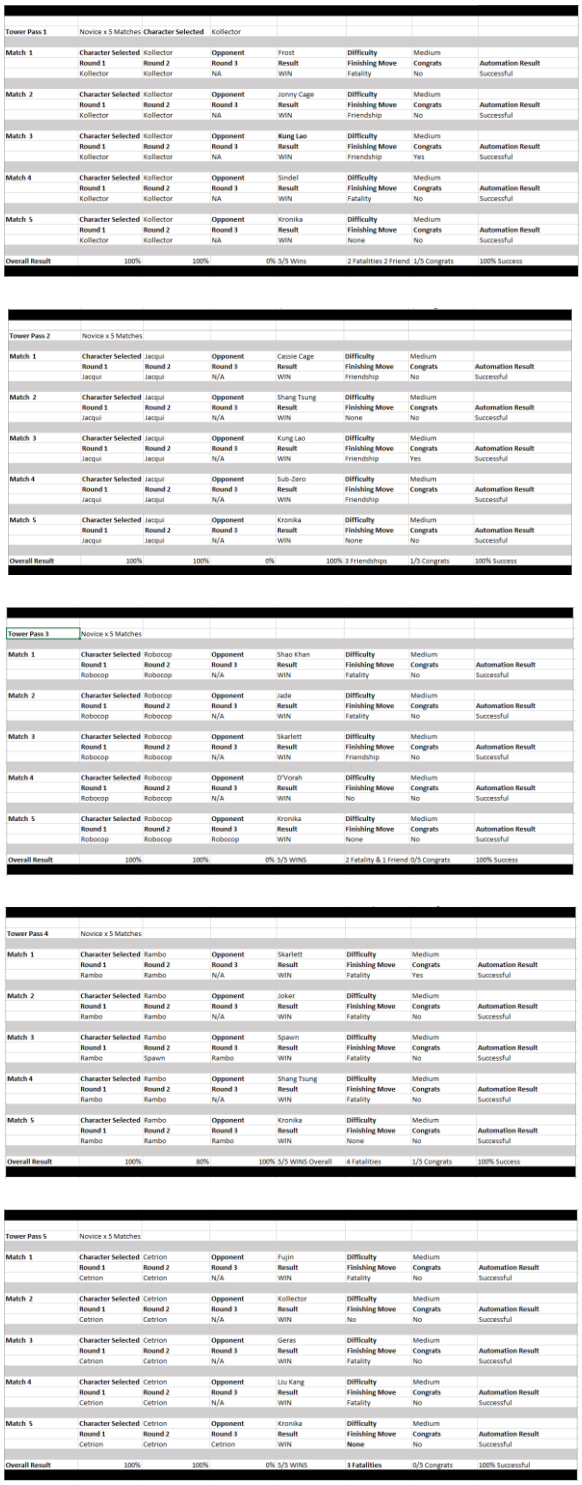
The tables cover the results of each round of 5 matches within the novice tower, the characters selected, difficulty, the results of the match [win or lose], and the result of the automation [either success or failure].

9. Results of Automation

By running the automation on the simplest (novice) tower, documenting the number of times that the game was looped through, and recording the number of times manual user input was required to progress the game, gave an estimate of how effective the automation was at handling the different game states. The Figure 18 shows the data collected from the initial tests of the simple automation and the completion of the Novice Tower with an AI combatant.

The automation was successful in progressing the game after each AI vs AI match through the Klassic Tower and even continued the game in the event of a loss. Because the custom characters selected had been modified, their chances of winning the matches on the Novice difficulty were already higher than an unmodified character load-out and that facilitated in the progression through the tower with little or no losses or repeated matches due to loss.

The automation executed successfully 100% of the time by identifying that the "Kontinue" button appeared and sending the "Enter" key input to progress regardless of a win or a loss state encountered.



Tower Pass 1	Novice x 5 Matches	Character Selected	Kollector				
Match 1	Character Selected	Kollector	Opponent	Frost	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Kollector	Kollector	NA	WIN	Fatality	No	Successful
Match 2	Character Selected	Kollector	Opponent	Jonny Cage	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Kollector	Kollector	NA	WIN	Friendship	No	Successful
Match 3	Character Selected	Kollector	Opponent	Kung Lao	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Kollector	Kollector	NA	WIN	Friendship	Yes	Successful
Match 4	Character Selected	Kollector	Opponent	Sindel	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Kollector	Kollector	NA	WIN	Fatality	No	Successful
Match 5	Character Selected	Kollector	Opponent	Kronika	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Kollector	Kollector	NA	WIN	None	No	Successful
Overall Result		100%	100%	0% 5/5 Wins	2 Fatalities 2 Friend	1/5 Congrats	100% Success

Tower Pass 2	Novice x 5 Matches						
Match 1	Character Selected	Jacqui	Opponent	Cassie Cage	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Jacqui	Jacqui	N/A	WIN	Friendship	No	Successful
Match 2	Character Selected	Jacqui	Opponent	Shang Tsung	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Jacqui	Jacqui	N/A	WIN	None	No	Successful
Match 3	Character Selected	Jacqui	Opponent	Kung Lao	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Jacqui	Jacqui	N/A	WIN	Friendship	Yes	Successful
Match 4	Character Selected	Jacqui	Opponent	Sub-Zero	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Jacqui	Jacqui	N/A	WIN	Friendship		Successful
Match 5	Character Selected	Jacqui	Opponent	Kronika	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Jacqui	Jacqui	N/A	WIN	None	No	Successful
Overall Result		100%	100%	0%	100% 3 Friendships	1/5 Congrats	100% Success

Tower Pass 3	Novice x 5 Matches						
Match 1	Character Selected	Robocop	Opponent	Shao Khan	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Robocop	Robocop	N/A	WIN	Fatality	No	Successful
Match 2	Character Selected	Robocop	Opponent	Jade	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Robocop	Robocop	N/A	WIN	Fatality	No	Successful
Match 3	Character Selected	Robocop	Opponent	Skarlett	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Robocop	Robocop	N/A	WIN	Friendship	No	Successful
Match 4	Character Selected	Robocop	Opponent	D'Vorah	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Robocop	Robocop	N/A	WIN	No	No	Successful
Match 5	Character Selected	Robocop	Opponent	Kronika	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Robocop	Robocop	Robocop	WIN	None	No	Successful
Overall Result		100%	100%	0% 5/5 WINS	2 Fatality & 1 Friend	0/5 Congrats	100% Success

Tower Pass 4	Novice x 5 Matches						
Match 1	Character Selected	Rambo	Opponent	Skarlett	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Rambo	Rambo	N/A	WIN	Fatality	Yes	Successful
Match 2	Character Selected	Rambo	Opponent	Joker	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Rambo	Rambo	N/A	WIN	Fatality	No	Successful
Match 3	Character Selected	Rambo	Opponent	Spawn	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Rambo	Spawn	Rambo	WIN	Fatality	No	Successful
Match 4	Character Selected	Rambo	Opponent	Shang Tsung	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Rambo	Rambo	N/A	WIN	Fatality	No	Successful
Match 5	Character Selected	Rambo	Opponent	Kronika	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Rambo	Rambo	Rambo	WIN	None	No	Successful
Overall Result		100%	80%	100% 5/5 WINS Overall	4 Fatalities	1/5 Congrats	100% Success

Tower Pass 5	Novice x 5 Matches						
Match 1	Character Selected	Cetrion	Opponent	Fujin	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Cetrion	Cetrion	N/A	WIN	Fatality	No	Successful
Match 2	Character Selected	Cetrion	Opponent	Kollector	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Cetrion	Cetrion	N/A	WIN	No	No	Successful
Match 3	Character Selected	Cetrion	Opponent	Geras	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Cetrion	Cetrion	N/A	WIN	Fatality	No	Successful
Match 4	Character Selected	Cetrion	Opponent	Liu Kang	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Cetrion	Cetrion	N/A	WIN	Fatality	No	Successful
Match 5	Character Selected	Cetrion	Opponent	Kronika	Difficulty	Medium	
	Round 1	Round 2	Round 3	Result	Finishing Move	Congrats	Automation Result
	Cetrion	Cetrion	Cetrion	WIN	None	No	Successful
Overall Result		100%	100%	0% 5/5 WINS	3 Fatalities	0/5 Congrats	100% Successful

Figure 18. Data Collected from the initial Tests

10. Challenges in Further Testing

Building more test-cases and testing them continued to see improvement in the coverage of

 | DOI: 10.20533/ijicr.2042.4655.2022.0137

scenarios that could be automated.

- Unplanned menu options that would require user input.
- Lack of character selection in the simple automation path.
- Noticed that enter command was being sent in some idol game-states without any on-screen ui element that was a known trigger for automation.

However certain challenges still existed with tuning the automation and covering all possible scenarios. Specifically, image recognition and the inclusion/omission of other on-screen elements in order to get more accurate results was necessary. A character selection automation was later added so that once the tower was completed a new character could be selected and another game be started without player input.

11. Anti-Cheat and Security

With the rise of online gaming, cheating has become an increasingly prevalent problem, and cheaters and gaming tend to go hand-in-hand [17]. A new question was raised during the study:

Is this type of automation considered cheating?

To combat cheating in the gaming industry, many gaming platforms have implemented anti-cheat systems that are designed to monitor and track suspicious behavior during gaming sessions. Google Stadia is no different, in-fact, Stadia had boasted one of the most effective anti-cheat systems in the industry [8]. Google Stadia used machine learning to identify and flag suspicious behavior. This can range from things like “aimbot” usage to more subtle forms of cheating like “scripting” or “boosting”. Once a player has been flagged, they can be banned from the platform entirely [7]. Many players want to get ahead in the competitive gaming leagues that they will go to great lengths and great sums of money to implement very sophisticated and technically complex cheats. In March of 2019, police in China busted the world's biggest gaming cheat manufacturer, seizing over $45 million in assets which was just a fraction of their revenue [17]. This ranged from things like aimbots, which are software that helps cheaters aim at targets in first-person shooting games with highly accurate precision, to even more subtle forms of cheating like scripting, where scripts are ran on the local computer to augment the game in favor of the layer, and boosting, where higher ranked players access lower-ranked players accounts to increase their rank for profit. Purchasable hacks don't come with a small price tag either, determined cheaters will often pay monthly or even daily for access to the latest cutting-edge aimbots for their games of choice [17]. Once a player has been flagged for cheating, they can be banned from a platform entirely. This system has proven to be quite effective in deterring cheaters. However, it's not perfect, and some clever players have found ways to circumvent the system. Nevertheless, Google Stadia was one of the best options for fair and clean online gaming [17].

By keeping the automation in the study very simple and limiting it to the automation of game transitions and game-state analysis to substitute for real-time player decision input in-game modes that were already automated helped to justify an ethical stance on how automation assisting the player is not cheating. However, the idea of creating more complex automation and automation that evades detection are topics that could be explored in future research which would directly relate to cheating and should be considered cheating if they are giving the player an unfair advantage in a one-on-one, one-on-many, multiplayer gaming or competitive scenarios.

Additionally, the use of RPA tools like Microsoft Power Automate did not seem to trigger any anti-cheat protection built into the platform and are likely not considered serious cheating tool themselves as they were not specifically designed to create cheatbots or have been proven method to give players an unfair advantage in game. This is likely because RPA is new or any overly technically complex solutions that cheaters would seek out would pose a barrier to deploy these types of solutions, requiring technical experience with the software, a testing background, and programming knowledge that would be better served in more established cheating software written in other mainstream programming languages.

12. Future State Discussion

The research study identified many ways that this type of research can be expanded in relationship to the foundations of the automation complexity and the directions for future research topics. The following are some of the opportunities for improvements:

- Automating other common functions:
  - i. Identification of additional automatable paths.
  - ii. Incorporate State Identification in complex automation.
- Enhancements of Mortal Kombat Characters with custom load-out using RPA and Machine Learning:
  - i. Collecting and leveraging data to make advanced decisions.
  - ii. Keeping matrix of player’s characters and customizations to make decision on what characters to play when different matches are

 | DOI: 10.20533/ijicr.2042.4655.2022.0137

paired or what character to try after a loss based on the character's customizations and probability of winning based on data collected in previous character vs character scenario.

- Match Pairing using Artificial Intelligence:
  i. Use match results to feed into adversarial gameplay algorithms like min-max and alpha-beta pruning, or Monte Carlo search to determine the best player to put into towers or individual matches.

Topics for future research included adding more complex automation paths to execute other gameplay scenarios, developing enhanced error handling and decision processing to avoid state-lock, evading known anti-cheat systems, and the implementation of strategies like the application of Artificial Intelligence (AI) and Machine Learning (ML) to further enhance the automation.

13. Conclusion

In conclusion, Robotic Process Automation was a powerful and highly customizable tool for automating various interactions and gameplay scenarios within a cloud-based game with little knowledge of programming and a low barrier to entry to get started. When designed and implemented correctly, RPA could take on many of the repetitive or mundane tasks that would otherwise require a human player's attention. This could free up the player's time by automating away repetitive aspects of games that require long time commitments but provide high levels of reward for playtime and repetitiveness. Yet players might still want to enjoy the gaming experience without having to worry about the more tedious elements of designing custom automations. Ultimately, RPA can help to improve the mundane aspects of certain gaming experiences for players looking to collect items or rewards who are knowledgeable and willing to commit some time to building the automation.

Designing automation for a video game requires careful consideration of the game's mechanics and flow. In particular, the identification of key gameplay states and transitions were essential to ensure that the automation worked as intended. Game-state testing was used to validate the automation and design the logic for progressively automating the transitioning of games states to the desired scenarios. RPA can absolutely be used to automate gameplay. However, was important to consider the design of the game when automating gameplay. The game's design should always be considered to ensure that the automated gameplay does not take away from the player's experience.

It was noted that when using automated gameplay with game-state automation, it is important to make sure that the automation does not interfere with the game's mechanics or cause any issues for other players as it could be considered cheating. As discussed, Google Stadia's anti-cheat system is a good example of how automated systems can be used effectively to monitor and track suspicious behavior on cloud-based gaming platforms [17]. Therefore, automating cloud-based gameplay can be a useful tool for gaming, but it is important to consider the design of the game and the effects that the automation has on the gameplay experience. Overall, this study examined how automation could be designed and implemented to execute various gameplay scenarios, how error handling and decision processing can be used to avoid state-lock, and how even well-known anti-cheat systems could be evaded by a product that was intended for enterprise or business use. We have also considered how strategies like the application of Artificial Intelligence (AI) and Machine Learning (ML) would further enhance the automation of gameplay.

In conclusion, RPA is a powerful tool that can be used to automate many different aspects of gameplay. With the right design and implementation, RPA can help players take their game to the next level.

14. References

[1] Nystrom, R. (2014). State – Game Programming Patterns / Design Patterns Revisited. Game Programming Patterns, Genever Benning, https://gameprogrammingpatterns.com/state.html, (Access Date: July 24, 2022).

[2] Whittaker, J. A. (2009). Exploratory Software Testing. Pearson Education.

[3] Bath, G., and McKay, J. (2014). The Software Test Engineer's Handbook. Rocky Nook.

[4] Muller, J. (3 February 2022). Test Driven Development (TDD) – An Approach to Automation.' Community Blog. UiPath. https://www.uipath.com/community/rpa-community-blog/understanding-test-driven-development-an-approach-to-automation., (Access Date: July 24, 2022).

[5] Microsoft. (2022). Take care of what's important. automate the rest. Power Automate. Microsoft Power Plat - form. https://powerautomate.microsoft.com/en-us/ (Access Date: July 10, 2022).

[6] Microsoft Power Automate. (25 May 2021). Microsoft Power Automate Overview. YouTube. https://www.youtube. com/watch?v=4z1A6YretuU (Access Date: July 10, 2022).

[7] Google. (n.d.). Stadia FAQ - Stadia help. Google. https: //support.google.com/stadia/answer/9338946?hl=en (Access Date: July 10, 2022).

[8] Google Developers. (8 May 2019). Stadia Streaming Tech: A deep dive (google I/O'19). YouTube.

 | DOI: 10.20533/ijicr.2042.4655.2022.0137

https://www.you tube.com/watch?v=9Htdhz6Op1I (Access Date: July 10, 2022).

[9] Stadia. (2022, November 9). Stadia Announcement FAQ - stadia help. Google. Retrieved November 27, 2022, from https://support.google.com/stadia/answer/12790109 (Access Date: November 27, 2022).

[10] NetherRealm Studios. (n.d.). Games. NetherRealm Studios. https://www.netherrealm.com/games/ (Access Date: July 10, 2022).

[11] Mortal Kombat. (n.d.). Game info. Mortal Kombat 11 Ultimate. https://mortalkombat.com/game-info (Access Date: July 10, 2022).

[12] Unreal Engine. (n.d.). The most powerful real-time 3D creation tool. Unreal Engine. https://www.unrealengine .com/en-US (Access Date: July 30, 2022).

[13] Williams, M. (2019, May 2). Mortal kombat 11's best feature takes the tedium out of the towers of Time Grind. USgamer.net. https://www.usgamer.net/articles/mortal-kombat-11s-best-feature-takes-the-tedium-out-of-the-towers-of-time-grind-#:~:text=In%20AI%20Battle%2C%20you%20make,Combos%2C%20Zoning%2C%20and%20Rushdown. (Access Date: July 10, 2022).

[14] Trantzas, G., Maniar, T., Leon, M., Melnykov, O., Rokontol., and Herbert. D. (16 December 2022.). UI elements and controls – power automate. UI elements and controls - Power Automate | Microsoft Docs. https://docs.microsoft.com/en-us/power- automate/desktop-flows/ui-elements (Access Date: July 24, 2022).

[15] Trantzas, G., Maniar, T., Leon, M., Melnykov, O., and Herbert. D. (n.d.). Images - power automate. Images - Power Automate | Microsoft Docs. https://docs.Microsoft .com/en-us/power-automate/desktop-flows/images (Access Date: July 31, 2022).

[16] Trantzas, G., Maniar, T., Leon, M., Melnykov, O., and Herbert. D. (n.d.). Using conditionals - power automate. Using conditionals - Power Automate | Microsoft Docs. https://docs.microsoft.com/en-us/power-automate/desktop-flows/use-conditionals (Access Date: July 31, 2022).

[17] DiBartolomeo, N. (2021, September 15). Stadia: The ultimate anti-cheat. Quit The Build. https://www.quitthe build.com/post/stadia-the-ultimate-anti-cheat (Access Date: July 10, 2022).

15. Special Note

More information about the Mortal Kombot RPA automation project and a video demo of the automation that was built from the information collected in this study can be found at www.mortalkombot.ai

 | DOI: 10.20533/ijicr.2042.4655.2022.0137