**Software Requirements Specification for the MultiGame System**

**A system to support recreational video game play and educational games and quizzes**

**Team Members: Phil Maalouf**

**0. Table Of Contents**

0. Table Of Contents

1. Introduction

2. User Requirements

Functional User Requirements

Non-Functional User Requirements

Diagrams

3. System Architecture

4. System Requirements

Functional System Requirements

Non-Functional System Requirements

System Requirements Tests

Tests For Functional Requirements

Tests For Non-Functional Requirements

5. System Models

**1. Introduction**

The MultiGame system is intended to fulfill the need for interactive graphical entertainment systems which, in addition to serving this role also provide for educational needs including interactively, graphically entertaining educational games and additional educational resources such as quizzes for vocabulary development, or quizzes for a course of study in Anatomy and Physiology. The system provides a main user interface which will display multiple game buttons, other buttons for altering basic system functionalities, and a central, detailed display which will show the details of a selected game (or quiz). Users will navigate to explore the various games, select and play games or take quizzes, peruse game play/quiz statistics, send messages to other players, post to their wall, etc.

The only other system that this version of the MultiGame system will interact with is the host operating system. Future versions, however, may be expanded to take advantage of other systems which offer APIs for messaging, game play, educational resources, geographical services, or social media such as photo sharing platforms.

The system is developed as a term project for a graduate level software engineering course but can be thought of as a prototype for an entrepreneurial endeavor into the video game/educational resource space. As such, developers may gain a feel for application development process applicable to such systems, from which they may be better poised to elicit and analyze system requirements for future increments. For example, some of the games included in the MultiGame system involve simple interactive animations. This prototype will serve as a window to the costs and benefits associated with such games. Quizzes developed for the system will shed light on the costs and benefits of compiling information from academic knowledge bases (e.g., textbooks) to serve as material for the quizzes, as well as those for quiz design and implementation. Overall user interface design and functionality tradeoffs can be postulated and deliberated over given the prototype as a jump-off point.

**2. User Requirements**

**Functional User Requirements**

**FUR-1:**

Priority: 1

**Story:**

A user does not have a MultiGame account and would like to play. She opens the application and clicks the "register" button. A registration page appears and she enters her information to register for a new account. She clicks the "create account" button to create the account and is then returned to the start page where she can log in.

**Scenario:**

**A description of what the system and users expect when the scenario starts:** The user expects to be able to open the application and to be able to see and click the "register" button.

**A description of the normal flow of events in the scenario:** In a normal flow of events, she will see the registration page open with fields to enter her information, such as email and password, and a "create account" button which, when clicked, will create her account and display a success message before returning her to the start page so she can log in.

**A description of what can go wrong and how resulting problems can be handled:** If the user enters an invalid email address or password, the system will not create a new account. Instead it will prompt the user for acceptable credentials.

**Information about other activities that might be going on at the same time:** Other players may be using MultiGame on their own computers, and other applications may be running on this user's computer.

**A description of the system state when the scenario ends:** This scenario completes at the start page where it began but this user will now be registered with a username and password and will be able to log in.

**FUR-2:**

Priority: 1

**Story:**

A user has a MultiGame account and would like to play. He opens the application and logs in by clicking the "login" button on the start page.

**Scenario:**

**A description of what the system and users expect when the scenario starts:**

When the user opens the application he expects to see and be able to click the "login" button.

**A description of the normal flow of events in the scenario:**

Normally, when he clicks "login" the system will bring up the "login" page where the user can enter his username and password and press "enter" to navigate to the main page where he can select a game to play, view stats, etc.

**A description of what can go wrong and how resulting problems can be handled:**

If there is a network problem, data corruption, or the user enters the wrong password it is possible that the user could be locked out of his account. If he can't log in after he tries again with the correct password Bob can contact the system administrator to get help in resolving the issue.

**Information about other activities that might be going on at the same time:**

**A description of the system state when the scenario ends:**

The state of the system can now be described as that state in which the application's main page is displayed and the application awaits input from the user.

**FUR-3:**

Priority: 0

**Story:**

A user logs into MultiGame to play SuperMath but she forgot to boost her play points after playing CoolPies the last time she was logged in. She clicks the "boost play points" button and increases her play points. Now she can play SuperMath and she gets a pretty good score.

**Scenario:**

**A description of what the system and users expect when the scenario starts:**

The user expects to be able to play SuperMath, but the system expects the user to have play points. Since the user is low on (or completely out of) play points she cannot play SuperMath. She used up too many play points playing CoolPies.

**A description of the normal flow of events in the scenario:**

Each game requires play points, and when a game is played the player's play points are decremented. The main page has a "boost play points" button. When clicked the system navigates to a "boost play points" page, where the user can choose how many play points she wishes to add. After adding the play points, the user is returned to the main page where she can use the play points to play games.

**A description of what can go wrong and how resulting problems can be handled:**

If there is a network issue, or perhaps a problem with the user's credit card the user will be informed of the issue. The user can then return to the main page and play "basic" grade games until she can boost her play points.

**Information about other activities that might be going on at the same time:**

**A description of the system state when the scenario ends:**

**FUR-4:**

Priority: 2

**Story:**

User 1 earns his all-time ultimate low score in SimpleSpaceship. In frustration he X's out of the application window without hitting the "logout" button. The system automatically logs him out so that play point pilfering user 2 can't open MultiGame and pilfer his play points.

**Scenario:**

**A description of what the system and users expect when the scenario starts:**

User 1 expects to do well in SimpleSpaceship and to be able to X out of the app in a blind fury in case he doesn't live up to his own lofty expectations. However, he would like to know that he will not remain logged in.

**A description of the normal flow of events in the scenario:**

When user 1 X's out of the app, the system logs him out.

**A description of what can go wrong and how resulting problems can be handled:**

If something goes wrong (e.g., a power outage occurs and the system loses power after user 1 clicks the "X" and before the system logs him out) user 2 will be able to open the app (when the power comes back on) and play games on user 1's account, both using up his play points and potentially sabotaging user 1's play stats. There's no recourse for this, so the implementation should preclude this.

**A description of what can go wrong and how resulting problems can be handled:**

**Information about other activities that might be going on at the same time:**

**A description of the system state when the scenario ends:**

Therefore, after user 1 has X'd out of the app the system will be off and user 1 will be logged out.

**FUR-5:**

Priority: 1

**Story:**

Use 2 discovers that stingy user 2 has left open and paused a MultiGame game of QuickJump with a near record-breaking score. Not wanting to be shamed, she clicks the "discard game" button before the score can be logged into User 1's stats page, writes "User 1 is a doofus" on his wall, then logs him out and logs in with her own account to try and improve her own high score in QuickJump.

**Scenario:**

**A description of what the system and users expect when the scenario starts:**

User 2 expects to be able to tinker with User 1's stats and wall since he is logged in and the game is paused. The system expects that the user at the UI will be authorized.

**A description of the normal flow of events in the scenario:**

Since the user is not authorized but still has access, sabotage ensues. User 1 can place a request to have the insult removed from his wall, but these things take time. As far as his high-scoring game goes, well, it's a quiet piece of history that nobody may believe. User 2 will be able to access and write to User 1's wall (since she's on his account) and that post will be made public.

**A description of what can go wrong and how resulting problems can be handled:**

**Information about other activities that might be going on at the same time:**

There's a network connectivity activity occurring concurrently.

**A description of the system state when the scenario ends:**

After this scenario User 1's account will be vandalized by User 2.

**FUR-6:**

Priority: 2

**Story:**

User 3 logs into MultiGame and navigates to the "stats" page, then clicks the "share stats" button and enters User 2's name in the "recipients" field to prove that he's the ultimate QuickJumper.

**Scenario:**

**A description of what the system and users expect when the scenario starts:**

User expects to be able to navigate to the "stats" page and to find a functioning "share stats" button which...

**A description of the normal flow of events in the scenario:**

... introduces a "recipients" field where he can enter User 2's name so that he can send her his stats.

**A description of what can go wrong and how resulting problems can be handled:**

If User 2 no longer has an account User 3 won't be able to send her his stats, and the system should inform User 3 of the issue.

**Information about other activities that might be going on at the same time:**

**A description of the system state when the scenario ends:**

After taking these steps User 2 will be able to view User 3's stats.

**FUR-7:**

Priority: 1

**Story:**

A user downloads, opens, and registers with MultiGame on good reference from a friend. She logs in and views the home page, where there are buttons for games, 5 along the top, 5 along the bottom, one on the right and one on the left. As she clicks the different game buttons the larger center image in the window shows the details of the game as a preview. The user double clicks 2DMatchstickBowling. She quickly gets bored and hits the "back" button to go to the main page and chooses a different game.

**Scenario:**

**A description of what the system and users expect when the scenario starts:**

The system expects the user's computer to have the minimum computer system requirements necessary to install and run MultiGame. The user expects to be able to log in and view the home page where there will be buttons for games, 5 along the top, 5 along the bottom, one on the right and one on the left.

**A description of the normal flow of events in the scenario:**

As the user clicks the different game buttons the larger center image in the window shows the details of the game as a preview. She double clicks 2DMatchstickBowling.

**A description of what can go wrong and how resulting problems can be handled:**

She quickly gets bored and hits the "back" button to go to the main page and chooses a different game.

**Information about other activities that might be going on at the same time:**

**A description of the system state when the scenario ends:**

After this scenario is complete the user will be back at the main page of the application where she can browse the different game options.

**FUR-8:**

Priority: 3

**Story:**

A user logs into MultiGame to play a round of DynaMaze. As he navigates the maze with acute arrow-key dexterity the maze changes. Not dissuaded, he uses his super-charged maze blaster to strategically blast a hole in a wall to grab a battery pack. Now recharged, his dynaCompass guides him toward the future location of the room's exit.

**Scenario:**

**A description of what the system and users expect when the scenario starts:**

The system expects the user to be up for a good challenge. The user expects to impress his real estate agent by posting a high DynaMaze score on his stats page.

**A description of the normal flow of events in the scenario:**

In a normal game of DynaMaze the user enters from a random side and as the player progresses through the maze the walls move, and various obstacles and power-ups hinder and help. The user can pick-up a maze blaster but can only blast through green walls, unless it's super-charged, in which case he can blast through green or blue walls. Super-chargers are rewarded for defeating enemies, and enemies are awarded super-chargers for defeating the user. The ultimate goal is to make way to the moving exit unscathed, or with as few scathes as possible, using the dynaCompass, which is obtained by making it to the core of the maze and defeating the dynaCompass goblin. It takes batteries.

**A description of what can go wrong and how resulting problems can be handled:**

The user may not do well in the game. If the user lacks DynaMaze skills the game can detect this and dynamically lower the difficulty level.

**Information about other activities that might be going on at the same time:**

**A description of the system state when the scenario ends:**

At the conclusion of this scenario the user will be able to choose between playing another round of DynaMaze or going back to the main MultiGame page.

**FUR-9:**

Priority: 0

**Story:**

A user joins MultiGame and explores her options by clicking the "<" and ">" buttons on either side of the large center display on the home page to view the game details. Optionally, the system could be designed so that the game button is highlighted when the corresponding game's details populate the details pane.

**Scenario:**

**A description of what the system and users expect when the scenario starts:**

The user expects to be able to view all the different game details without clicking on each game image.

**A description of the normal flow of events in the scenario:**

This way, she can hold the mouse in one position and repeatedly click until she finds a game she would like to play, then click the details pane to play the game. The "<" and ">" (from here on, "arrows") appear on the left and right of the details pane, respectively, either just inside or just outside the respective edge of the pane. As the arrows are clicked the image in the pane cycles through the details for the different games.

**A description of what can go wrong and how resulting problems can be handled:**

If the user cycles through all the games and continues clicking the same arrow the images will just cycle again restarting from the first one.

**Information about other activities that might be going on at the same time:**

**A description of the system state when the scenario ends:**

When the user has finished exploring the different games using the "arrows" the MultiGame main page will have the details for one game in the details pane, and possibly have one game highlighted.

**FUR-10:**

Priority: 2

**Story:**

User 1 logs in and discovers AandPQuiz. In reading the details of the game he decides he will spend the afternoon playing it so he can learn enough A and P to impress his classmate User 2, who's studying to be a physician. User 1 double clicks the details pane and chooses to start taking quizzes in successive order, starting with the quiz on body systems.

**Scenario:**

**A description of what the system and users expect when the scenario starts:**

AandPQuiz starts with a page where the user can choose among different topics and possibly subsequently subtopics. Alternatively, the user can choose "beginner","intermediary", or "expert". In User 1's case, he chooses to start taking all the beginner level quizzes in successive order. The system expects the user to choose a quiz, or choose a skill level and then choose a quiz, or choose a skill level and then choose to take some set of successive quizzes.

**A description of the normal flow of events in the scenario:**

Once this choice has been made the user proceeds with the quiz(s) having his stats page updated at the completion of each quiz. A user completing a beginner sequence will be given the option to advance to the corresponding intermediary sequence, and likewise from intermediary to expert.

**A description of what can go wrong and how resulting problems can be handled:**

It is possible that the answers for the quizzes are incorrect. If this is found to be the case the user should contact the system developers to inform them.

**Information about other activities that might be going on at the same time:**

**A description of the system state when the scenario ends:**

When the user chooses to stop playing the game he returns to the main application page by clicking the "back" button.

**FUR-11:**

Priority: 2

**Story:**

User 2 chooses to play NotTooShabby Vocab to get the wheels turning so she can better expressively draft a note to her classmate User 1. She starts with the synonyms quiz for the word "impressed".

**Scenario:**

**A description of what the system and users expect when the scenario starts:**

User 2 expects to be able to improve her vocabulary by taking various types of quizzes.

**A description of the normal flow of events in the scenario:**

She can choose fill-the-blank, multiple-choice, or crossword style quizzes with challenges such as synonyms, antonyms, or definitions over sets such as math, biology, adjectives, verbs, or foods.

**A description of what can go wrong and how resulting problems can be handled:**

If the app loses its connection to the network it won't be able to load the quiz libraries, though it may keep some favorites in memory. In this case the user will be prompted to connect to a network to play any un-cached quizzes.

**Information about other activities that might be going on at the same time:**

**A description of the system state when the scenario ends:**

**Non-Functional User Requirements**

**NFUR-1:**

Priority: 2

The system should have a fast response time. User's will not want to wait for the system to respond.

**NFUR-2:**

Priority: 1

The system should not run up excessive network traffic. Networks tend to be taxed already so our application shouldn't contribute unduly to the issue.

**NFUR-3:**

Priority: 2

The system should support enhanced security. We do not want our product to be a security liability.

**NFUR-4:**

Priority: 0

The system should be designed to improve our bottom line. While remaining sensitive to broader social concerns it is necessary that we retain a viable business model. Therefore, care should be taken to establish sources of revenue through MultiGame without compromising our mission statement.

**NFUR-5:**

Priority: 1

The system should not crash or unduly tax the host operating system. If it can be achieved, we would like our system to perform with high responsivity without hanging up or crashing the host OS.

**Diagrams**

A picture containing text, map

Description automatically generated

Figure 1: use case diagram for the MultiGame system.

**4. System Architecture**

This section shows the working system's architcture at a high level. The speed of development is not necessarily synchronized with the speed of these high-level documentations. Therefore, there may be minor discrepancies between these views and the latest system version or versions. Nonetheless, they should provide a decent picture of the architectural design choices made in the progressing implementation.

A screenshot of a cell phone

Description automatically generated

Figure 2: MultiGame Package Diagram

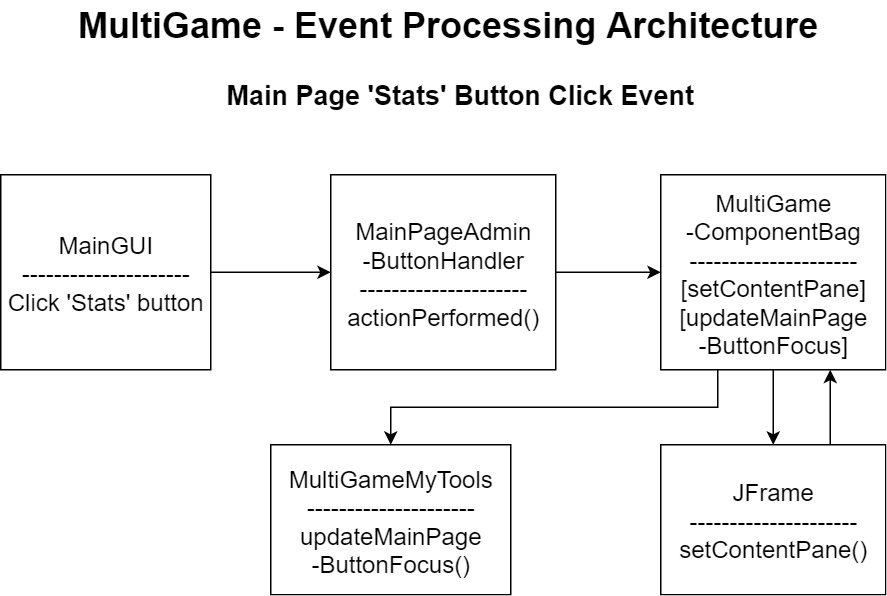


Figure 3: MultiGame - Event Processing Architecture

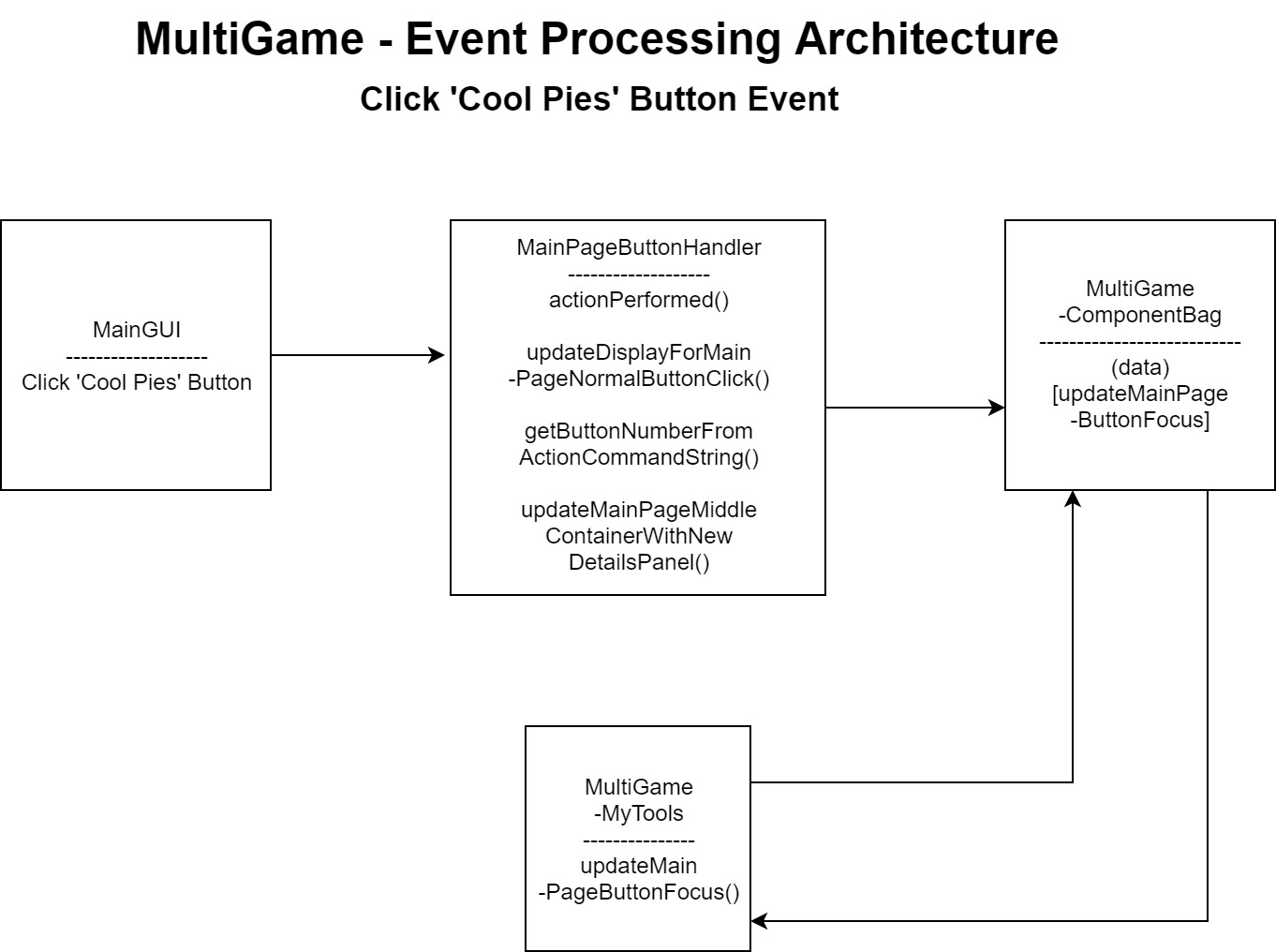


Figure 4: MultiGame - Event Processing Architecture

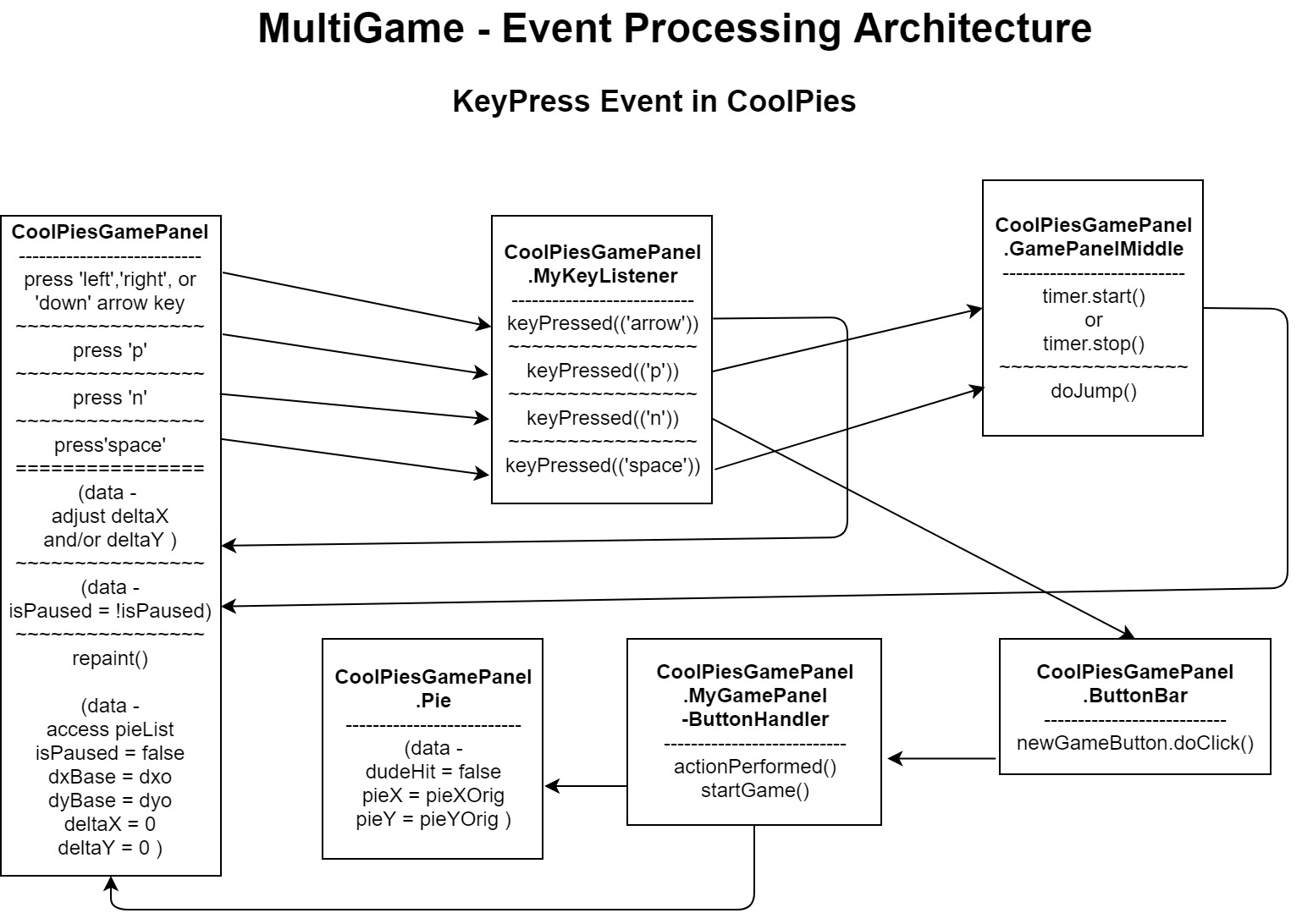


Figure 5: MultiGame - Event Processing Architecture - KeyPress Event in CoolPies

A screenshot of a cell phone

Description automatically generated

Figure 6: MultiGame Package Diagram

A picture containing screenshot

Description automatically generated

Figure 7: MultiGame Architecture using the Model-View-Controller Pattern

A screenshot of a cell phone

Description automatically generated

Figure 8: MultiGame - Observer Design Pattern

**4. System Requirements**

**Functional System Requirements**

FSR-1.1 (priority 1):The system shall have a registration page which will include fields to enter an email, password, and username.

FSR-1.2 (priority 1): The registration page shall have a "create account" button.

FSR-1.3 (priority 2): When clicked, the "create account" button shall check the email, password, and username entered for validity. If not valid it shall inform the user and instruct to retry, showing the "rules" for properness. Otherwise the account shall be created so the user will be able to log in with his new credentials.

FSR-2.1 (priority 1): The system shall have a start page with a "login" button.

FSR-2.2 (priority 3): When clicked, the "login" button shall bring the user to a "login" page where he can enter his credentials and press "enter" to navigate to the main UI page.

FSR-2.3 (priority 1): If the user enters the wrong credentials or another error occurs, preventing the user from logging in, the user shall be informed of the nature of the issue and allowed to retry.

FSR-3.1 (priority 0): The system shall contain games SuperMath. SuperMath shall yield a score.

FSR-3.2 (priority 1): The system shall contain game CoolPies. CoolPies shall yield a score.

FSR-3.3 (priority 1): The system shall incorporate play points. These shall be allocated to each user, with current values displayed on the main UI page.

FSR-3.4 (priority 1): When a game is selected and played the user's play points shall be decremented by an appropriate value, possibly depending on the game selected.

FSR-3.5 (priority 1): The main UI page shall have a "boost play points" button which, when clicked, will navigate the user to a page for handling the boosting of play points. The page shall allow for the user to choose how many play points to add and include fields and buttons for simulating credit card processing. Since this system is intended as a prototype the credit card processing will not require real functionality.

FSR-3.6 (priority 1): In the case of a network issue or problem with the user's credit card the user shall be informed of the nature of the issue and allowed to retry. If it is not possible to boost a user's play points the user shall be allowed to play "basic" grade games until she can boost her play points.

FSR-4.1 (priority 2): The system shall contain game SimpleSpaceship which, when played, shall yield a score.

FSR-4.2 (priority 2): The system shall have a "logout" button.

FSR-4.3 (priority 2): A first user shall be able to easily X out of the application window at any time and be automatically logged out of the system so that no second user will be able to open the app and play games on the first user's account using up his play points or sabotaging his play stats.

FSR-5.1 (priority 1): The system shall contain game QuickJump which, when played, shall yield a score.

FSR-5.2 (priority 1): The MultiGame system shall allow a user to pause a live game. Un-pausing the game shall resume gameplay.

FSR-5.3 (priority 1): The MultiGame system shall have a user stats page which will contain the stats for the different games which can be played. The stats page should highlight high scores for each game played.

FSR-5.4 (priority 1): QuickJump shall have a "discard game" button which will discard the current game so that the current score cannot be logged into the user's stats page.

FSR-5.5 (priority 2): The system shall have a "wall" for each user, where the user will be able to post public updates.

FSR-5.6 (priority 2): Users shall be able to place a request to have an item removed from their wall

FSR-6.1 (priority 2): The "stats" page shall have a "share stats" button which will enable User A to enter a second User B's username in a "recipients" field and then send User B User A's stats for viewing.

FSR-6.2 (priority 2): In the case where the username entered in the "recipients" field does not correspond to an active user account the initiating user should be informed of the issue.

FSR-7.1 (priority 1): The system shall contain game 2DMatchstickBowling.

FSR-7.2 (priority 1): User's shall be able to download, open, and register with the MultiGame system.

FSR-7.3 (priority 1): The MultiGame home page shall have buttons for the different games/quizzes, five along the top, five along the bottom, one on the right and one on the left.

FSR-7.4 (priority 1): The home page shall have a larger center image which shows the details of the currently highlighted game as a preview.

FSR-7.5 (priority 1): When the user clicks the different game buttons the larger center image shall update to show the details for the corresponding game/quiz.

FSR-7.6 (priority 1): When the user double-clicks a game button the corresponding game shall open so that the user can play.

FSR-7.7 (priority 1): The game 2DMatchstickBowling (and, likely, all games) shall have a "back" button which, when clicked, will navigate the user back to the main page.

FSR-8.1 (priority 3): The system should contain a game DynaMaze which will yield a score. DynaMaze should be a challenge for the user.

FSR-8.2 (priority 3): DynaMaze should allow for navigation of a maze by arrow-key commands.

FSR-8.3 (priority 3): As the user's character navigates the maze the walls change position.

FSR-8.4 (priority 3): The user's character should be able to use a super-charged maze blaster to blast a hole in a wall.

FSR-8.5 (priority 3): The user's character should be able to pick up items such as battery packs.

FSR-8.6 (priority 3): The user's character should be able to use an item, dynaCompass, which will guide him toward the future location of a room's exit. The dynaCompass should run on batteries.

FSR-8.7 (priority 3): The user's character should enter a DynaMaze room form a random side.

FSR-8.8 (priority 3): Various obstacles and power-ups should hinder and help the user's character.

FSR-8.9 (priority 3): The user's character should be able to pick-up a maze blaster. The maze blaster should be able to blast through green walls. The maze blaster should also be able to blast through blue walls if and only if it is super-charged.

FSR-8.10 (priority 3): Super-chargers should be rewarded to the user's character for defeating enemies, and enemies should be awarded super-chargers for defeating the user's character.

FSR-8.11 (priority 3): The goal of DynaMaze game play should be to make way to the moving exit while avoiding being harmed as much as possible using an item, the dynaCompass.

FSR-8.12 (priority 3): The user's character should be subject to injuries, called scathes. There should be a scathe-meter to indicate the current health of the user's character.

FSR-8.13 (priority 3): The dyna-compass should be obtained by reaching the core of the maze and defeating a dynaCompass goblin. The goblin should be defeated by excessive scathing.

FSR-8.14 (priority 3): If the user is not doing well in the game the system should detect this and dynamically lower the difficulty level.

FSR-8.15 (priority 3): After the game has been completed (successfully or unsuccessfully) the user should be able to choose to play another round of DynaMaze or return to the MultiGame main page.

FSR-9.1 (priority 0): The user should be able to explore game/quiz options by clicking left and right arrow ("<" and ">") buttons. The buttons should be located on the left and right of the details pane, respectively, either just inside or just outside the respective edge of the pane.

FSR-9.2 (priority 0): When clicked, the arrows should change the image in the large, center details pane to show the details of a different game. The user should be able to hold the mouse in one position and keep clicking to cycle through all the available games. Once all games have been cycled through the cycle should start back from the first game again.

FSR-9.3 (priority 2): The game/quiz button for the currently selected game/quiz should be highlighted.

FSR-10.1 (priority 2): The system shall contain a game AandPQuiz.

FSR-10.2 (priority 2): The AandPQuiz should start with a page where the user can choose among different topics and possibly subsequently subtopics or choose between different skill levels. The user should be able to choose, for example, to take all the "beginner" level quizzes, one for each topic, in successive order, or choose to take the "beginner" level quiz for a specific topic.

FSR-10.3 (priority 2): At the completion of each quiz the user's stats should be logged in the user's stats page.

FSR-10.4 (priority 3): After completing one level for a sequence or single quiz, the user should be given the option to advance to the next difficulty level for the corresponding sequence or quiz.

FSR-10.5 (priority 2): AandPQuiz should have a "back" button which, when clicked, will bring the user back to the main MultiGame page.

FSR-11.1 (priority 2): The system shall contain a game NotTooShabby Vocab.

FSR-11.2 (priority 2): The game should have various types of quizzes, including a synonyms quiz for the word "impressed".

FSR-11.3 (priority 2): The user should be able to choose among fill-the-blank, multiple-choice, or crossword style quizzes with challenges such as synonyms, antonyms, or definitions over sets such as math, biology, adjectives, verbs, or foods.

FSR-11.4 (priority 3): The game should load quiz libraries over a network connection but should also keep some favorites in memory.

FSR-11.5 (priority 3): If the app loses its network connection the user should be prompted to connect to a network to play any un-cached quizzes.

**Non-Functional System Requirements**

NFSR-1.1 (priority 2): Button clicks should resolve within 128 ms.

NFSR-1.2 (priority 2): Page transitions should resolve within 128 ms.

NFSR-1.3 (priority 2): Animated gameplay commands such as arrow-key presses and mouse-clicks should have response times less than or equal to 32 ms.

NFSR-1.4 (priority 2): Network requests should be resolved without excessive delay. If delays of more than 512ms occur because of network limitations the user should be informed that the system is waiting for the network so they don't keep pressing buttons, wondering what's going on.

NFSR-2.1 (priority 2): For any set of resources which may be accessed via network the system should implement some sort of caching policy so that some of these resources will be stored on the device to avoid excessive network traffic. This should be verified by disconnecting from the network and using these common resources.

NFSR-2.2 (priority 1): Any game stats archived via network should only be uploaded at the completion of a game. No intra-game stats archiving should occur.

NFSR-2.3 (priority 1): The MultiGame system should remain usable in the absence of a network connection.

NFSR-3.1 (priority 2): Users should need to enter credentials, including username and password, in order to login to MultiGame.

NFSR-3.2 (priority 2): Username and password should be authenticated via network and checked against a database of hashed user credentials.

NFSR-4.1 (priority 0): Revenue should be elicited from users in one or more ways. Possibilities include requiring play points to play certain games, requiring users to pay to download the system, or charging extra for extra levels, items, etc.

NFSR-5.1 (priority 1): The system should not use an excessive amount of processes/threads that could slow down the host OS. The limitation on thread usage should not exceed 40 for a Windows system, 80 for a Mac system, or 160 for a Linux system.

**System Requirements Tests**

**Tests For Functional Requirements**

FSR-1.1: Observe and verify that there is a registration page. Observe and verify that the registration page has fields to enter an email, password, and username.

FSR-1.2: Observe and verify that the registration page has a "create account" button.

FSR-1.3: Observe and verify that proper email, password, and username will cause the system to create a new user account and that it is possible to log in with these credentials. Observe and verify that any improper email, password, or username entered will cause the system to instruct the user to retry and show the "rules" for what is acceptable values for these fields.

FSR-2.1: Observe and verify that the system has a start page with a "login" button.

FSR-2.2: Observe and verify that, when clicked, the "login" button will bring the user to a "login" page where he can enter his credentials and press "enter". Observe and verify that pressing "enter" here navigates the user to the main UI page.

FSR-2.3: Observe and verify that the user will be informed of any issue which has prevented him from logging in. Examples to check for are incorrect username, incorrect password, and network connection issue.

FSR-3.1: Simply observe and verify that the system contains game SuperMath.

FSR-3.1: Simply observe and verify that the system contains game CoolPies.

FSR-3.2: Simply observe and verify that the system uses play points which are displayed in the main UI page.

FSR-3.3: Observe and verify that a user's play points are in fact decremented by the appropriate value when a game is played.

FSR-3.4: Verify there is a "boost play points" button. Verify it navigates to the page for boosting play points. Verify correct behavior for "corner cases", adding 0, negative, or astronomical play points.

FSR-3.5: Verify faulty credit card info will not work for play points. Verify that "basic" grade games can still be played when the user does not have play points.

FSR-4.1: Observe and verify that the system contains a game called SimpleSpaceship and that it will yield a score when played.

FSR-4.2: Observe and verify that the system has a "logout" button which, when clicked, will log the current user out of the system.

FSR-4.3: Observe and verify that the a user (User 2) cannot open the system and use another user's (User 1's) account after User 1 has X'd out of the main application window without logging out.

FSR-5.1: Observe and verify that the system contains a game called QuickJump and that it will yield a score when played.

FSR-5.2: Observe and verify that there is a way to pause any live game and a way to un-pause to resume gameplay.

FSR-5.3-a: Observe and verify that the system has a user stats page accessable from the main UI page and that it does indeed contain the stats for the different games which can be played.

FSR-5.3-b: Observe and verify that the stats page has highlighted high scores for each game played.

FSR-5.4: Observe and verify that the QuickJump game has a "discard game" button which, when clicked, discards the current game. Verify that the current score is not logged into the user's stats page.

FSR-5.5: Observe and verify that each user has a "wall" and that the user is able to post public updates on his own wall.

FSR-5.6: Observe and verify that a user can place a request to have an item removed from their wall.

FSR-6.1: Verify that the "stats" page has a "share stats" button and that clicking the button gives the user the opportunity to enter a second user's username in a field labeled "recipients" and then send stats for viewing.

FSR-6.2: It's unfeasible to test all possible cases of this. Test several known usernames and then several known non-usernames to verify that the non-usernames will effect the informing of the user. Also, test any corner-cases, such as blank usernames, usernames of users whose accounts have been deleted, usernames which are a concatenation of a current username and some other string, etc.

FSR-9.1: Observe and verify the arrows are in their correct locations, and the correct behavior effected when clicking these.

FSR-9.2: Observe and verify that the arrows change the image in the large, center details pane to show the details of a different game. Verify the user can hold the mouse in one position and cycle through all the games by clicking the mouse button. Verify the cycling continues when the user cycles past the last game.

FSR-9.3: Verify the correct game/quiz button is highlighted (that it matches the game in the details pane).

**Tests For Non-Functional Requirements**

NFSR-2.1: Verify by disconnecting from the network and using these common resources.

NFSR-2.2: Invoke operating system services to monitor network traffic during game play to verify that no intra-game stats archiving occurs during gameplay. Alternatively, add a debugging tag to any functionality involved in stats archiving and verify that these tags do not print during gameplay.

NFSR-2.3: Verify that the system is available, and any non-network dependent functionality is working while the network is disconnected.

**5. System Models**

This section presents select Use Case Diagrams, Use Case Descriptions, Sequence Diagrams, and Class Diagrams from the MultiGame system.

A picture containing text, map

Description automatically generated

Figure : Major Use Cases for the MultiGame System

A close up of text on a white background

Description automatically generated

Figure : Select Use Cases for the MultiGame System

|  |  |
| --- | --- |
| Use Case Name:  Arrow Button Click |  |
| Actors | MultiGame user |
| Description | The user 'scrolls' to display a different game details panel by clicking an arrow ('<' or '>' ) button. The main GUI's large central details panel is updated to reflect the details of a new game as 'scrolling' is effected by clicking the arrow button. Also, the large game button associated with the current game in the details panel will be highlighted accordingly. This use case involves interactions among object classes including JButton, ButtonHandler, ActionEvent, ComponentBag, JPanel, and MyTools in both the mainPackage and mainPageListeners packages. |
| Stimulus | The user clicks an arrow ('<' or '>') button on the large, central details panel of the main GUI page. |
| Response | The large, central panel of the main GUI page is upadated to display the appropriate game details panel. |
| Comments | As the user 'scrolls' through the different games' details the details panel is updated and the corresponding game button is highlighted. If the last game (Game 10) is currently displayed (with the corresponding button highlighted) another right-arrow button click will result in the first game being displayed (and the first game buttton highlighted), and the analogous case for the first game (SuperMath) and the left-arrow button click. This way the 'scrolling' will cycle through all the game details. |

A screenshot of a cell phone

Description automatically generated

Figure : Sequence Diagram for the Arrow Button Click Use Case

|  |  |
| --- | --- |
| Use Case Name:  Navigate to Game Page |  |
| Actors | MultiGame user |
| Description | The user clicks a particular game's details which are displayed in a large central display on the main page GUI to navigate to the corresponding game page. For example, clicking the 'SuperMath' details will take the user away from the main page to the 'SuperMath' game page. Once at the game page the user will be able to play the game (if it has been implemented). This use case involves interactions among the DetailsPanel, MouseHandler, MouseEvent, ComponentBag, GameDetailPanel, Map<Integer,JPanel>, JFrame, and JPanel classes. In addition, the game pages which may be loaded are themselves classes which extend the JPanel class and (as of the time of this writing) these include the SuperMathPanel, CoolPiesPanel, NotTooShabbyVocab, and GamePanelTop classes. There are dependancies among the games, mainPackage, and mainPageListeners packages for this use case. |
| Stimulus | The user clicks the details panel for a particular game. This is found in the large central panel of the main page. |
| Response | The main application window will navigate away from the main GUI page and instead show the appropriate game with it's GUI where the user will be able to play the game (if it has been implemented). |
| Comments | The image in the game details panel shown in the main GUI is supposed to represent the details of the associated game so that the user can decide if she would like to play that game. So far, however, these images have instead mostly only shown the name of the associated game and possibly some graphics. The SuperMath details panel shows some equations and the NotTooShabby Vocab panel indicates the idea of that game as well, but the other games' details panels are not so informative. Also, the games all use a very similar GUI. The details panel click functionality has been implemented and will navigate the user to the associated game page. |

A screenshot of a cell phone

Description automatically generated

Figure 12: Sequence Diagram for the Navigate to Game Page Use Case

A close up of a map

Description automatically generated

Figure 13: Class Diagram - Relationships Drawn from Arrow Button Click and Navigate to Game Page Use Case Sequence Diagrams

A close up of a map

Description automatically generated

Figure : MultiGame High Level Relationships - Class Diagram. Any class which contains (CB) also aggregates Component-Bag. I did not make these explicit with arrows for brevity.

A screenshot of a cell phone

Description automatically generated

Figure 15: State Diagram for MultiGame System Window

A picture containing screenshot

Description automatically generated

Figure 16: State Diagram for the MultiGame CoolPies Dude

**State Diagram for the Main Page Details Panel**

A close up of a map

Description automatically generated

Figure 17: State Diagram for the Main Page Details Pane. Messages on transition lines indicate button clicked events.l