Test plan for GUI Set game:

Coverage: This plan will directly test the GUI application (main_gui.rb). Since the GUI application ties each class together, all methods and classes will be indirectly tested by the test plan.

Method: This test plan is intended to be executed manually by a user. Since it is hard to unit-test a GUI, this plan aims to ensure that both the visual and functional aspects of the game are thoroughly tested.

Responsibility: This test plan was followed by everyone on the project team. Anyone else with access to the application may follow this test plan to double check the validity of our game.

Beginning of Game:

1. Start the game
2. Type in a name and press the enter button to ensure that the naming feature works
3. Type in Yes, No, or some other option to test text-cleaning in the AI prompt
4. Type in 1, 2, 3, or some other option to test text-cleaning in the difficulty prompt
5. Verify that at least 12 cards were dealt to the table
    a. If more than 12 cards were dealt, make sure that a prompt appears explaining that no sets were found
6. Inspect cards and buttons to ensure that they are properly sized and that any text is readable

Buttons:

1. Click on the left-most hint button to verify that it works
    a. The button should prompt the user with how many sets are possible on the board
2. Click on the second hint button to verify that it works
    a. The button should prompt the user with the index of a card on the board that belongs to an arbitrary set
3. Click on the third hint button to verify that it works
    a. The button should prompt the user with the indices of 2 cards on the board that belong to the same arbitrary set
    b. One of the indices displayed should match the index from the first hint button
4. Click on the fourth hint button to verify that it works
    a. The button should prompt the user with the indices of 3 cards on the board that make up a complete set
    b. Two of the three indices should match the indices from the previous 2 hints

Set Logic:

1. Click on a few known invalid sets to make sure that the sets are properly rejected
    a. The user should be prompted that their sets are incorrect
    b. The user should lose a point for each invalid set they make
2. Click on 3 cards that form a valid set to ensure that the set is accepted
    a. The user should be prompted that their set is valid
    b. The user should gain a point for each valid set they make
3. Verify that 3 new cards are added to the board and that the existing cards are not shifted around
    a. If no cards are added, ensure that the game ends within the next few found sets - this validates that no cards were added because the deck was empty
    b. If there are more than 12 cards on the board before making a set, gaps may appear on the board. This is a deliberate choice to ensure that cards aren't shuffled around.
4. Continue to make sets, miss sets, and click hint buttons throughout the game to reach the end

AI Logic (if playing an AI game):

1. Make sure you have entered "Yes" to the AI prompt at the beginning of the game
2. Wait 30, 60, or 120 seconds (hard, medium, or easy difficulties) for the AI to pick a set
    a. Make sure that the AI can choose both valid AND invalid sets
    b. AI's score should increase on a valid set and decrease on an invalid set

Timer Logic (if playing a non-AI game):

1. Make sure you have entered "No" to the AI prompt at the beginning of the game
2. Wait for the timer to count down
    a. Upon reaching 0, the player should lose a point
3. Now make some sets
    a. The timer should reset back to its starting value upon making a valid set

End Game:

1. As game approaches the end, make sure that new cards aren't added as the deck should be empty
2. Verify that the game ends when the deck is empty and no more cards on the board can form a valid set
    a. The user should be prompted that the game is over
3. Verify that the prompt tells the user who won and that the prompt is correct (in an AI game)
4. Verify that your name matches what you entered at the beginning

If a user follows this test plan and all verifications are successful, then we can consider that particular run a success. As more successful runs are conducted, we can build confidence that our application is stable and free of any *major* bugs.