Software Requirement Document

Overview:

This document outlines the software requirements for the "Refactored Exploding Kittens" game in the form of detailed use cases that reflect the code's execution flow.

Functional Requirement/ Use cases:

Use Case 1: Setup Game

Scope: The entire process of a user starting the application, configuring a new game, and the system dealing the initial cards.
Primary Actor: Player
Preconditions: The application is not yet running.
Basic Flow:

1. The Player runs the application, which executes Main.main().
2. Main instantiates Game, GameUI, and GameController.
3. GameController creates a GameSetupUI to manage the configuration process.
4. The System (GameSetupUI) displays: "Please choose a language: 1. English 2. German". The Player enters a valid number (1 or 2).
5. The System (GameSetupUI) displays the game mode options. The Player enters a valid number to select a mode.
6. The System (GameSetupUI) prompts for the number of players (between 2 and 4). The Player enters a valid number.
7. The GameController calls setupGame(), which uses the collected information to initialize the Game object, including populating the Deck and creating Player objects.
8. The setup phase concludes, and the GameController proceeds to the main game loop.

Alternative Flow (Invalid Input):

- At steps 4, 5, or 6, if the Player enters text, a number outside the allowed range, or any other invalid input, the GameSetupUI will display a descriptive error message (e.g., "Invalid input. Please enter a number between 2 and 4.").
- The system will then re-display the exact same prompt from the step where the error occurred.
- This loop continues until the Player provides a valid input.

Success Condition (Postconditions): A game is fully initialized. The deck is populated and shuffled, players have their starting hands, and the game is ready for the first turn.

Use Case 2: Play Game

Scope: The high-level, automated loop that governs the flow of the game from the first turn until a winner is declared.
Primary Actor: System (GameController)
Preconditions: A game has been successfully set up.
Basic Flow:

1. The GameController initiates its run() method, starting the game loop.
2. For each iteration, the GameController identifies the current player via the TurnManager .
3. It then calls GameUI.startTurn() to begin the current player's turn (see Use Case: Take a Turn).
4. After the turn concludes, the TurnManager advances to the next player.
5. The GameController checks the game-over condition by asking the PlayerManager for the number of active players.
6. The loop continues as long as more than one player is active.
7. Once the loop terminates, GameUI.endGame() is called to display the winner.

Alternative Flow:

- This use case is fully automated and driven by the system's state. It does not involve direct user input, so there are no alternative flows based on invalid input. Specific outcomes of player actions are handled in their respective use cases.

Success Condition (Postconditions): The game has concluded, and a single winner is announced.

Use Case 3: Take a Turn

Scope: A single player's turn, involving choices to play cards or to end their action phase by drawing a card.
Primary Actor: Player
Preconditions: It is the specified Player's turn.
Basic Flow:

1. GameUI.startTurn() displays the current player's name, their hand of cards, and a menu of primary actions (e.g., "1. Play Card", "2. Draw Card").
2. The Player enters a valid command to select an action.
3. If "Play Card" is chosen, the system prompts for the specific card to play. This flow then

continues in a more specific use case (e.g., Play a Card Combo , Respond with Nope).
4. If "Draw Card" is chosen, GameUI calls game.drawCard(). The result of this action is handled by another use case (e.g., Handle Exploding Kitten).
5. The Player's turn ends, assuming they were not eliminated by drawing a card.

Alternative Flow (Invalid Command):

- At step 2, if the Player enters a command that is not a valid menu option (e.g., "3", "abc"), the GameUI displays an error message ("Invalid command, please try again.").
- The system then re-displays the turn menu from step 1, prompting for a correct choice.

Success Condition (Postconditions): The Player completes their action phase and successfully draws a card, ending their turn without being eliminated.

Use Case 4: Respond with Nope

Scope: The sequence allowing any player to cancel another player's action card by playing a NOPE card.
Primary Actor: Player (any player except the one who played the original action)
Preconditions: An action card (that is not a Kitten or Defuse card) has just been played.
Basic Flow:
1. Player A plays an action card (e.g., ATTACK).
2. The GameController cycles through all other players, prompting each one: "[Player Name], do you want to play a NOPE card? (y/n)".
3. Player B responds with "y". The system validates they have a NOPE card, which is then discarded.
4. The GameController immediately restarts the cycle, asking all players (including Player A) if they want to NOPE Player B's NOPE.
5. All players respond with "n".
6. Player B's NOPE resolves, canceling Player A's ATTACK card. The turn proceeds.
Alternative Flow (Invalid Input):

- At step 3 or any subsequent NOPE prompt, if a Player enters a command other than "y" or "n", the system displays an error message ("Invalid input. Please enter 'y' or 'n'.") and re-issues the same prompt to that same player.

Success Condition (Postconditions): The original action is either definitively canceled or definitively proceeds.

Use Case 5: Handle Exploding Kitten

Scope: The entire lifecycle of an EXPLODING_KITTEN encounter, from drawing it to its resolution or the consequences of holding it.

Primary Actor: Player

Preconditions: A Player either draws an EXPLODING_KITTEN or loses their STREAKING_KITTEN while holding one.

Basic Flow:

Flow A: Drawing an Exploding Kitten

1. The Player draws an EXPLODING_KITTEN. The system immediately analyzes their hand for a solution.
2. Path 1: Player has STREAKING_KITTEN. The EXPLODING_KITTEN is added to their hand. They are safe, and their turn ends. The state described in Flow B now applies.
3. Path 2: Player has DEFUSE (but no STREAKING_KITTEN). The DEFUSE is automatically played and discarded. The system prompts the player to enter a numeric position to re-insert the EXPLODING_KITTEN . The player enters a valid number, the card is re-inserted, and their turn ends.
4. Path 3: Player has neither. The Player is immediately eliminated from the game. The EXPLODING_KITTEN is discarded.

Flow B: Losing a Streaking Kitten while Holding an Exploding Kitten

1. Player A holds both a STREAKING_KITTEN and an EXPLODING_KITTEN.
2. An external event (e.g., another player's steal action) removes the STREAKING_KITTEN from Player A's hand.
3. The system immediately re-evaluates Player A's hand and finds the now-unprotected EXPLODING_KITTEN.
4. The logic reverts to Flow A, step 3. The system searches for a DEFUSE. If found, it's used. If not, Player A is eliminated.

Alternative Flow (Invalid Input for Re-insertion):

- In Flow A, Path 2, if the player enters non-numeric text, a negative number, or a number greater than the deck size, the GameUI displays an error ("Invalid position. Please enter a number from 0 to [deck size].").
- The system then re-prompts for a valid position. The flow continues until valid input is provided.

Success Condition (Postconditions): The EXPLODING_KITTEN is neutralized (defused or taken into hand) or the Player is eliminated.

Use Case 6: Handle Imploding Kitten

Scope: The specific two-stage sequence for drawing and handling the IMPLODING_KITTEN card.
Primary Actor: Player
Preconditions: The Player has just drawn an IMPLODING_KITTEN card.
Basic Flow (Drawing Face-Down):

1. The Player draws the IMPLODING_KITTEN. The system checks its isFacedUp status, which is false.
2. The card's state is flipped to true. The Player's turn ends instantly.
3. The GameUI prompts the Player: "You drew the Imploding Kitten! Choose a position (0-[deck size]) to re-insert it, face-up."
4. The Player enters a valid numeric position.
5. The card is placed back in the Deck, and the game continues.

Alternative Flow (Drawing Face-Up):

- A Player draws the IMPLODING_KITTEN. The system checks its isFacedUp status, which is true.
- The Player is immediately eliminated. This cannot be Noped or Defused.

Alternative Flow (Invalid Input for Re-insertion):

- At step 4, if the Player enters text or a number outside the valid range, GameUI displays an error and re-issues the prompt from step 3 until valid input is received.

Success Condition (Postconditions): The Player either re-inserts the card face-up and survives or is eliminated if drawing it face-up.

Use Case 7: Play a Card Combo

Scope: A Player uses a two-card or three-card combination to steal a card from another player.
Primary Actor: Player
Preconditions: It is the Player's turn, and they have the necessary matching cards in their hand.
Basic Flow:

1. The Player chooses to play a "Special Combo" from the turn menu.
2. The system prompts for the target player: "Select a target player (by number)." The Player enters the number of another active player.

3. The system prompts for combo type: "Select combo type (2 or 3 cards)." The Player enters a valid number.
4. If 3-card combo: The system prompts: "Name the card you wish to steal." The Player enters a valid card name (e.g., "Defuse").
5. The appropriate steal action (stealRandomCard or stealTypeCard) is executed.
6. The GameUI displays the result (e.g., "You stole a Defuse card!" or "Steal failed, target does not have that card.").

Alternative Flow (Invalid Input):

- At step 2, if the Player enters their own number, the number of an eliminated player, or invalid text, the system shows an error and re-prompts for a valid target.
- At step 3 or 4, if the Player enters an invalid option, the system shows an error and re-prompts.
- The flow resumes at the point of error once valid input is provided.

Success Condition (Postconditions): The steal action is resolved (successfully or not), and the combo cards are discarded.