Project Report

Art Dealer Game

Grade 3-5

Software Engineering

Submitted by:

Group 1

October 17, 2024

**Table of Contents**

[Introduction 3](#_Toc180017795)

[Objectives: 3](#_Toc180017796)

[Game Logic and Patterns 3](#_Toc180017797)

[User Interface and Interactivity 5](#_Toc180017798)

[Process Model 6](#_Toc180017799)

[Use Cases 7](#_Toc180017800)

[Use Case 1: Start a New Game 7](#_Toc180017801)

[Use Case 2: Select Cards 8](#_Toc180017802)

[Use Case 3: Submit Cards for Dealer Evaluation 9](#_Toc180017803)

[Use Case 4: Guess the Dealer's Pattern 9](#_Toc180017804)

[Use Case 5: View Feedback on Card Selection 10](#_Toc180017805)

[Use Case 6: Restart the Game 10](#_Toc180017806)

[UML Model 11](#_Toc180017807)

[Use Case Diagram: 11](#_Toc180017808)

[Deployment Diagram: 12](#_Toc180017809)

[Class Diagram: 13](#_Toc180017810)

[State Diagram: 15](#_Toc180017811)

[Activity Diagram: 16](#_Toc180017812)

[Customer Journey Map 17](#_Toc180017813)

[Persona 19](#_Toc180017814)

[Testing Strategies 19](#_Toc180017815)

[Scope of Testing 19](#_Toc180017816)

[Testing Types 20](#_Toc180017817)

[Functional Testing 20](#_Toc180017818)

[Non-Functional Testing 20](#_Toc180017819)

[User Acceptance Testing (UAT) 21](#_Toc180017820)

# Introduction

Third through fifth graders can practice pattern spotting using Python-based simulation called the Art Dealer Game. Acting as a gallery owner will help the player find out how an artificial intelligence art dealer spends their money. Using a conventional 52-card deck, the player tries to create combinations depending on known trends.

## Objectives:

The primary objective of the player is twofold:

* Lay out four cards that the art dealer will purchase.
* Correctly identify the pattern the dealer is following from a list of possible patterns.

This game will let students enjoy Python's Tkinter interface with pygame for sound effects. Thanks to the comments following every game, players can quickly modify their plans and projections. The game calls off whenever a player has utilized all of their guesses or effectively matched a pattern.

Unless all of the card numbers are odd or even, there can only be a set mix of suits. Participating in these trends would be a fantastic approach for children to acquire critical thinking and problem-solving in an interesting and enjoyable manner.

## Game Logic and Patterns

Our Art Dealer Game, designed for students in grades three through five, started with a card pattern. The game starts by creating a shuffled deck of 52 standard playing cards, consisting of four suits (Hearts, Diamonds, Clubs, and Spades) and values ranging from Ace (A) to King (K). At each turn, the player selects four cards from this deck and attempts to match a predefined pattern that the software (acting as the art dealer) is following.

The patterns used in the game are as follows:

1. All Even Numbers: The cards laid out must have even values (e.g., 2, 4, 6, 8, 10).
2. All Odd Numbers: The cards must have odd values (e.g., 1, 3, 5, 7, 9).
3. Sum of Values Greater than 20: The total value of the four cards must exceed 20 (with face cards counted as 10).
4. Two Red and Two Black Cards: The hand must consist of exactly two red-suited cards (Hearts, Diamonds) and two black-suited cards (Clubs, Spades).
5. All Cards of Different Suits: Each card must belong to a different suit (one from each of Hearts, Diamonds, Clubs, and Spades).
6. Two Picture Cards (Jack, Queen, King): The player’s hand must contain exactly two face cards (J, Q, K).
7. Prime Numbers and a Face Card: The hand must include at least one prime number (2, 3, 5, 7, 11) and at least one face card (J, Q, K).
8. Two Cards Add Up to 10: Two of the cards in the player’s hand must sum to 10 (e.g., 4 and 6, 3 and 7).
9. All Cards Greater than 5: The values of all cards must be greater than 5 (e.g., 6, 7, 8, 9, 10).
10. One Card of Each Rank (1-10): The hand must consist of cards that represent unique values from 1 to 10.

These patterns are checked using lambda functions that evaluate the selected cards. The player continues guessing until they correctly match the art dealer’s pattern or exhaust their guesses. The game provides immediate feedback after each round, helping players adjust their strategy for the next attempt.

## User Interface and Interactivity

The game is designed using the Tkinter library for a simple yet interactive graphical user interface (GUI). The GUI allows players to interact with the game smoothly by selecting cards and making guesses about the dealer's pattern. Here are the key features of the user interface:

* **Card Selection Area:** Randomly from the virtual deck of the game interface, the player selects four cards. The player can view all of their playing cards and choose the ones they choose to retain within this window.
* **Pattern Guess Area:** Players can check a list of possible patterns after every hand to observe the dealer's possible behavior. Should their predictions come true, players advance to the next round.
* **Feedback and Results:** The game will mark if your hand fits their pattern every time the dealer turns around. Successful pattern recognition could lead the player to be rewarded either visually or aurally.
* **Game Flow Control:** Starting the game, disable the UI, and guess well. Regardless of the result, players might decide to start over and hunt for another pattern.

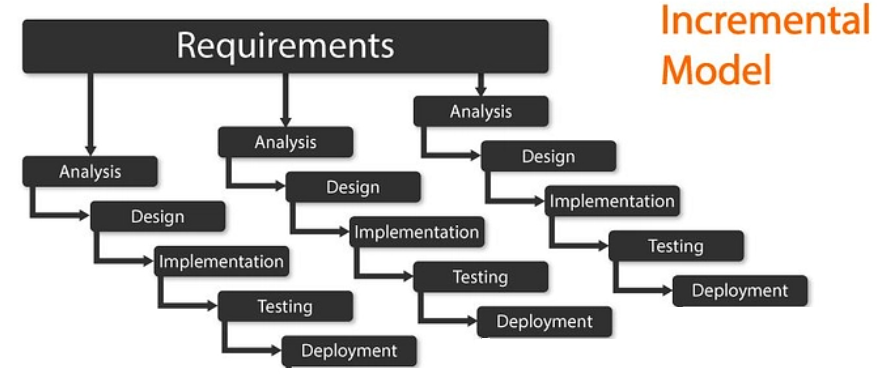
Children may focus on the areas of the game that solve difficulties as there are few buttons to press. Younger players will find the game more interesting as the pygame package uses audio signals. While appreciating this interactive feature that enhances the gaming experience, students could improve their computational and pattern-recognition skills.

# Process Model

Since there aren't many buttons to press, children may focus on the aspects of the game that fix problems. For younger players, using audio signals included in the pygame package makes the game more interesting. The interactive elements of the game let pupils have fun while learning arithmetic and pattern identification.

Steps of the Incremental Process Model in the Art Dealer Game:

1. **Requirements Gathering:**
   1. The specs, from what I could gather, were complete. A Python-based card game called the skill Dealer Game sought to teach third- through fifth-graders the skill of pattern recognition.
2. **Design:**
   1. The makers of the game focused especially on a user-friendly interface and many patterns. We worked our way up from the most basic elements—card selection and shuffling—while developing the framework—that is, the basis of pattern recognition.
3. **Implementation:**
   1. After a slow start, one could clearly see the most basic features of the game—building the deck and rearranging the cards. Using pattern-checking tools—those for even/odd numbers or sum-based patterns—you can progressively raise the difficulty of the game.
4. **Testing and Feedback:**
   1. With each increment, testing was likely done to ensure that the game logic was working correctly. For example, after implementing patterns like "All Even Numbers" or "Two Red and Two Black Cards," those specific features would have been tested before moving on to implement the next set of patterns.
5. **Deployment and Refinement:**
   1. Once all the patterns and game logic were implemented, the entire game was tested and refined. The user interface was likely adjusted and improved in this phase, making the game more user-friendly and engaging for students.



# Use Cases

## ****Use Case 1: Start a New Game****

|  |  |
| --- | --- |
| **Use Case ID** | UC-1 |
| **Use Case Name** | Start a New Game |
| **Primary Actor** | Student (Player) |
| **Goal** | Start a new game session |
| **Preconditions** | The game is installed and running |
| **Main Success Scenario** | 1. The player opens the game application. 2. The game interface loads with options to start a new game. 3. The player clicks the "Start New Game" button. 4. The game shuffles the deck of 52 cards. 5. A new game session begins. |
| **Postconditions** | A new game session starts with a shuffled deck of cards. |
| **Extensions** | None |

## ****Use Case 2: Select Cards****

|  |  |
| --- | --- |
| **Use Case ID** | UC-2 |
| **Use Case Name** | Select Cards |
| **Primary Actor** | Student (Player) |
| **Goal** | Select four cards to lay out |
| **Preconditions** | The game session is active, and the deck is shuffled |
| **Main Success Scenario** | 1. The player views the available cards. 2. The player selects four cards. 3. The selected cards are displayed in the player’s hand. |
| **Postconditions** | The selected cards are displayed as the player’s hand. |
| **Extensions** | If less than four cards are selected, the game prompts to select more cards. |

## ****Use Case 3: Submit Cards for Dealer Evaluation****

|  |  |
| --- | --- |
| **Use Case ID** | UC-3 |
| **Use Case Name** | Submit Cards for Dealer Evaluation |
| **Primary Actor** | Student (Player) |
| **Goal** | Submit selected cards for evaluation |
| **Preconditions** | The player has selected four cards |
| **Main Success Scenario** | 1. The player clicks the "Submit" button. 2. The game checks the selected cards against predefined patterns. 3. Dealer accepts or rejects the cards. 4. Feedback is given based on the result. |
| **Postconditions** | The player receives feedback on whether the cards match the dealer’s pattern. |
| **Extensions** | The player can select new cards if the cards do not match. |

## ****Use Case 4: Guess the Dealer's Pattern****

|  |  |
| --- | --- |
| **Use Case ID** | UC-4 |
| **Use Case Name** | Guess the Dealer's Pattern |
| **Primary Actor** | Student (Player) |
| **Goal** | Guess the correct pattern from the dealer |
| **Preconditions** | The player has submitted cards for evaluation and received feedback |
| **Main Success Scenario** | 1. The player guesses the dealer's pattern from a provided list. 2. If the guess is correct, the player wins. 3. If the guess is incorrect, the player is prompted to guess again (up to 3 times). |
| **Postconditions** | The game ends or prompts the player to make another attempt. |
| **Extensions** | After 3 incorrect guesses, the game stops and offers a new attempt. |

## ****Use Case 5: View Feedback on Card Selection****

|  |  |
| --- | --- |
| **Use Case ID** | UC-5 |
| **Use Case Name** | View Feedback on Card Selection |
| **Primary Actor** | Student (Player) |
| **Goal** | Receive feedback based on the selected cards |
| **Preconditions** | The player has submitted four cards for dealer evaluation |
| **Main Success Scenario** | 1. The game checks the selected cards against the dealer's pattern. 2. The player receives feedback on whether the cards match the pattern. 3. If the cards match, the game indicates success; if not, the player is prompted to try again. |
| **Postconditions** | The player receives feedback on the current selection, allowing them to adjust their next move. |
| **Extensions** | None. |

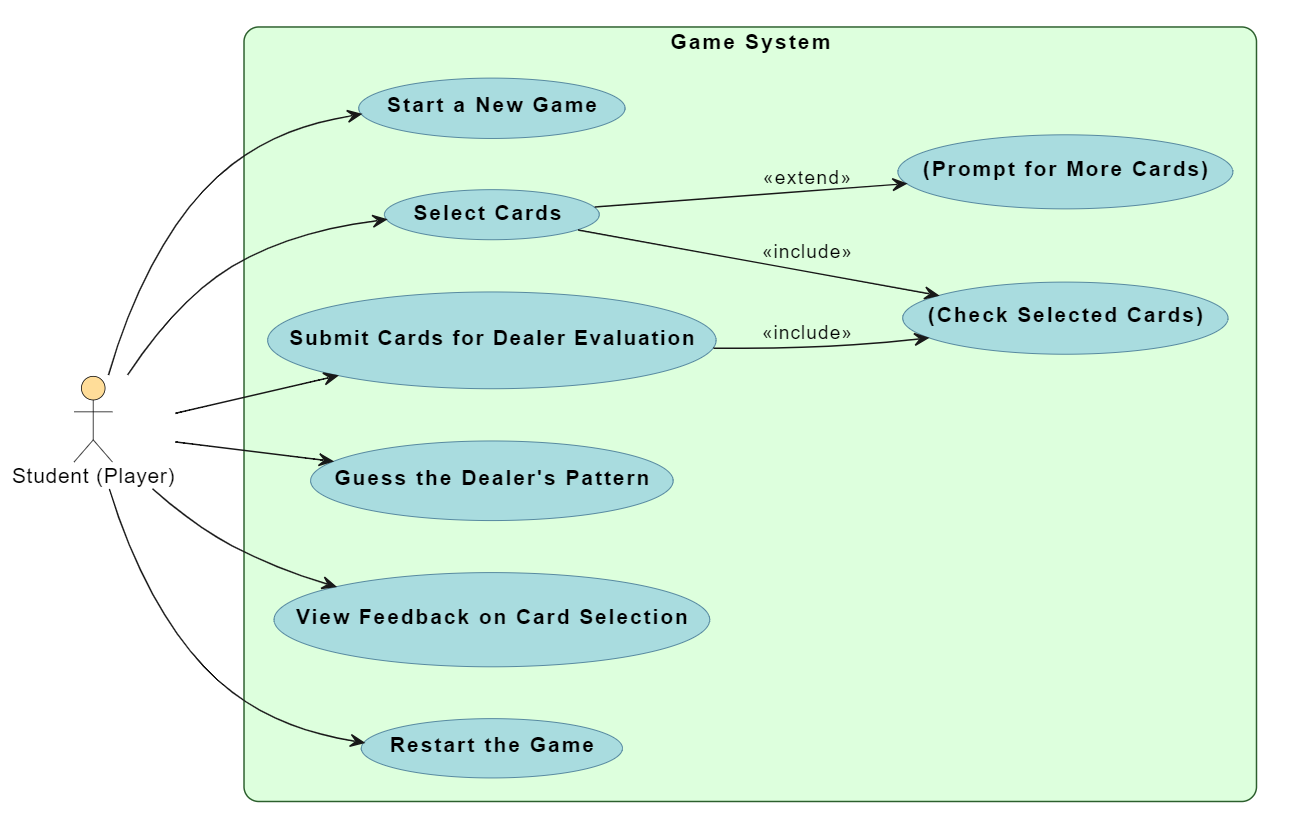
## ****Use Case 6: Restart the Game****

|  |  |
| --- | --- |
| **Use Case ID** | UC-6 |
| **Use Case Name** | Restart the Game |
| **Primary Actor** | Student (Player) |
| **Goal** | Restart the game after a round has ended |
| **Preconditions** | The player has completed or ended a game session |
| **Main Success Scenario** | 1. The player clicks the "Restart Game" button. 2. The game reshuffles the deck. 3. A new game session begins with the same difficulty level. |
| **Postconditions** | The game restarts with a fresh deck and a new pattern for the dealer. |
| **Extensions** | The player can choose to restart immediately after finishing the game or end the session. |

# UML Model

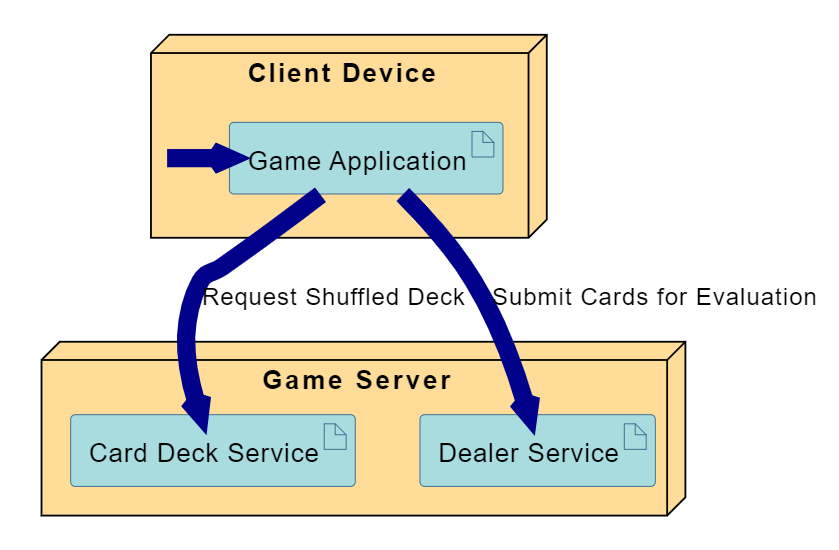
## Use Case Diagram:

A Use Case Diagram visually represents the interactions between a user (actor) and a system to capture its functionalities and behaviors. It identifies the different ways a user might interact with the system to achieve specific goals, such as starting a game, selecting options, or submitting actions. The diagram consists of actors (e.g., "Student (Player)") who engage with use cases (e.g., "Start a New Game"), and it can include relationships like <<include>>, which shows mandatory steps in a use case, and <<extend>>, which highlights optional behaviors. By outlining these interactions, the Use Case Diagram provides a high-level overview of the system’s functionality and the user's role within it.



## Deployment Diagram:

A Deployment Diagram illustrates the physical architecture of a system, showing how software components are deployed across hardware nodes. In this case, the diagram depicts the interaction between the Client Device (representing the player's device) and the Game Server, where the key services for the game are hosted. The Game Application runs on the client, and communicates with the Card Deck Service (responsible for shuffling and managing the deck) and the Dealer Service (which evaluates the player's card selections) on the server. The arrows indicate data flow anytime a player requests a fresh deck or turns in their cards for inspection. This graph shows how different system components interact with server-side services consumers use.



## Class Diagram:

In this Class Diagram of a card game players choose cards, show them for evaluation, and then get dealer remarks. Common subjects in the artwork are the Player, Game, CardDeck, Dealer, and Card classes. These classifications stand for qualities, deeds, and relationships.

Player Class:

* Attributes: playerId, playerName, and selectedCards (a list of Card objects).
* Methods:
  + startNewGame() initializes a new game session.
  + selectCard() allows the player to choose a card.
  + submitCards() submits the selected cards for evaluation.
* Relationship: A Player plays exactly one Game and selects zero or more Cards.

Game Class:

* Attributes: gameId, currentPlayer (a reference to a Player), dealer (a reference to a Dealer), and deck (a reference to a CardDeck).
* Methods:
  + start() begins a new game.
  + restart() restarts an existing game.
  + evaluatePlayerCards() evaluates the player's selected cards.
* Relationships: The Game interacts with both the CardDeck and Dealer. The game is responsible for shuffling the deck and interacting with the dealer to evaluate the player's selected cards.

CardDeck Class:

* Attribute: cards, a list of Card objects representing the full deck.
* Methods:
  + shuffle() randomizes the order of cards in the deck.
  + dealCard() returns a card to the player.
* Relationship: The Game manages a CardDeck and calls on it to shuffle and deal cards.

Dealer Class:

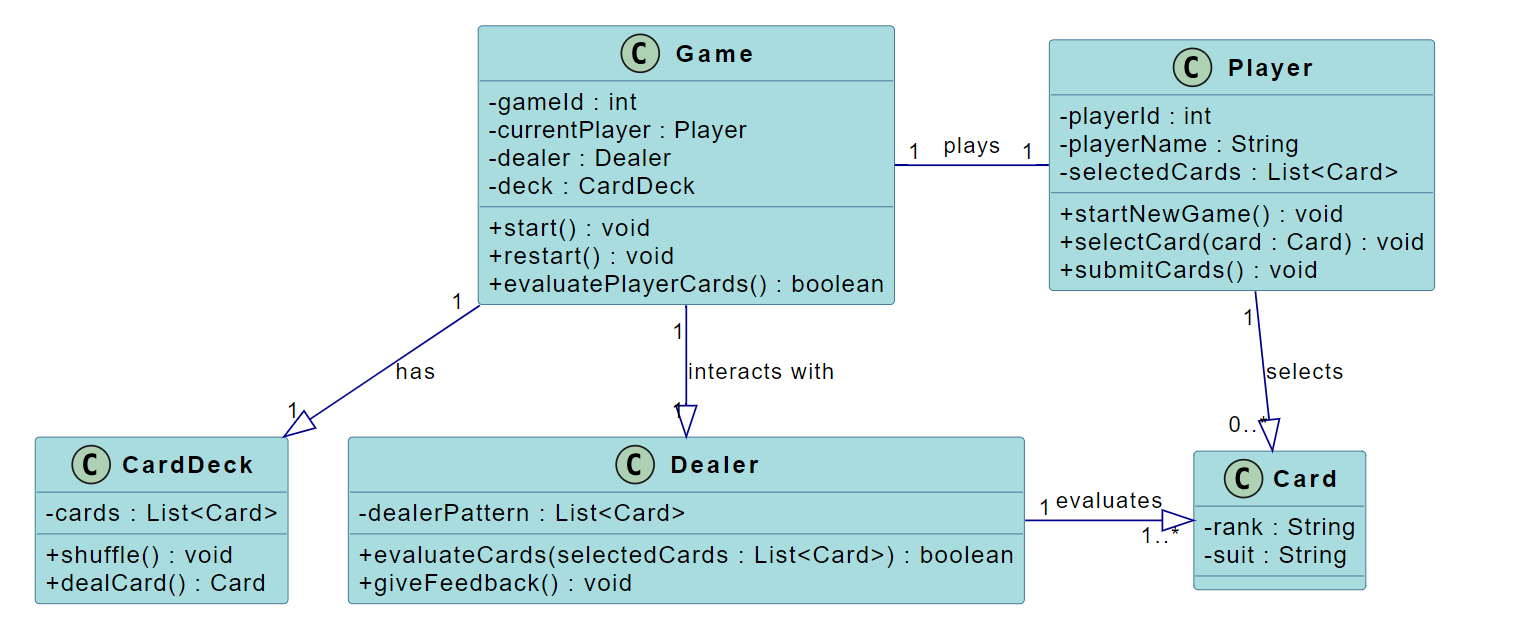
* Attribute: dealerPattern, a list of Card objects representing the pattern against which the player's cards are evaluated.
* Methods:
  + evaluateCards() checks whether the player's selected cards match the dealer's pattern.
  + giveFeedback() provides feedback based on the evaluation.
* Relationship: The Dealer evaluates the player's selected cards and interacts with the Game to process the results.

Card Class:

* Attributes: rank and suit, representing the value and type of a card (e.g., Ace of Spades).
* Relationship: The Player selects one or more Cards, and the Dealer evaluates these cards.

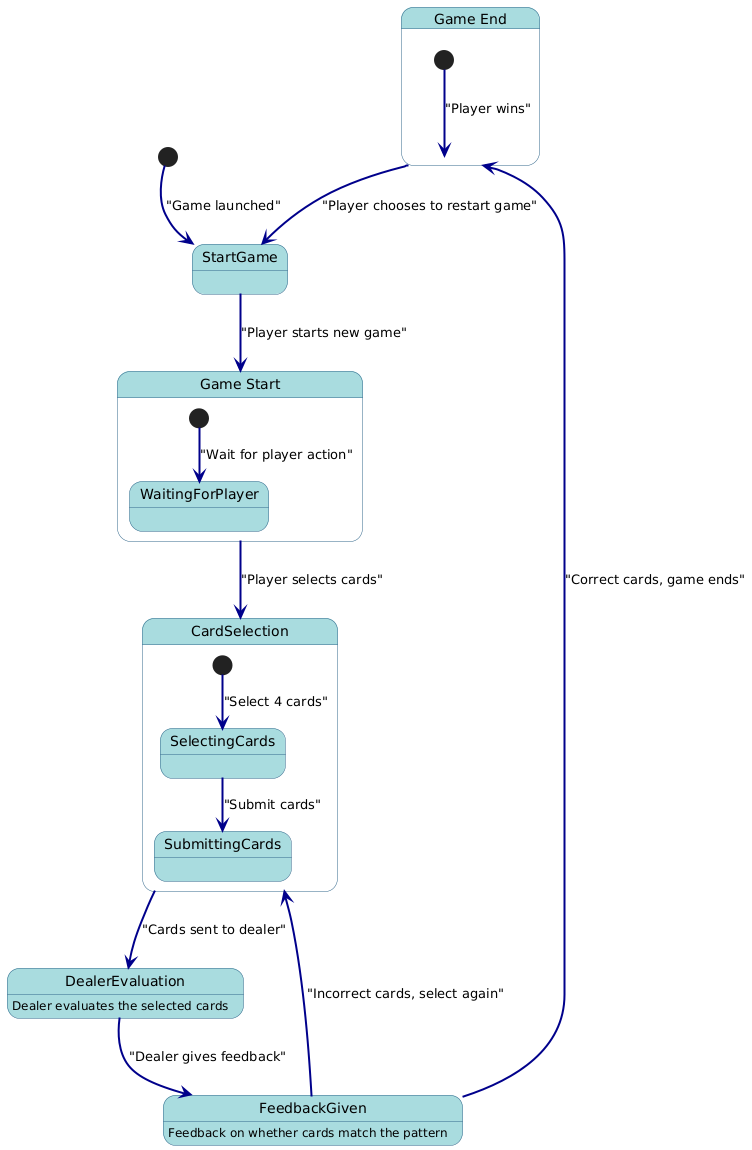
Relationships:

* Multiplicity:
  + A Player plays exactly one Game.
  + A Game interacts with one Dealer and manages one CardDeck.
  + A Player can select zero or more Cards (0...\*), while the Dealer evaluates one or more cards (1...\*).



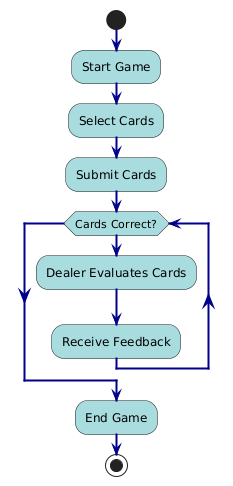
## State Diagram:

This State Diagram represents the flow of a card-based game. It starts with the game being launched (StartGame), transitioning to the Game Start state where the system waits for the player to take action. The player can then move to the Card Selection state to select and submit cards for dealer evaluation. The game transitions to the Dealer Evaluation state, where the dealer checks the player's cards. After the evaluation, feedback is provided in the FeedbackGiven state. If the cards are incorrect, the player returns to the Card Selection state; if correct, the game ends, reaching the Game End state. The player can choose to restart the game from there.



## Activity Diagram:

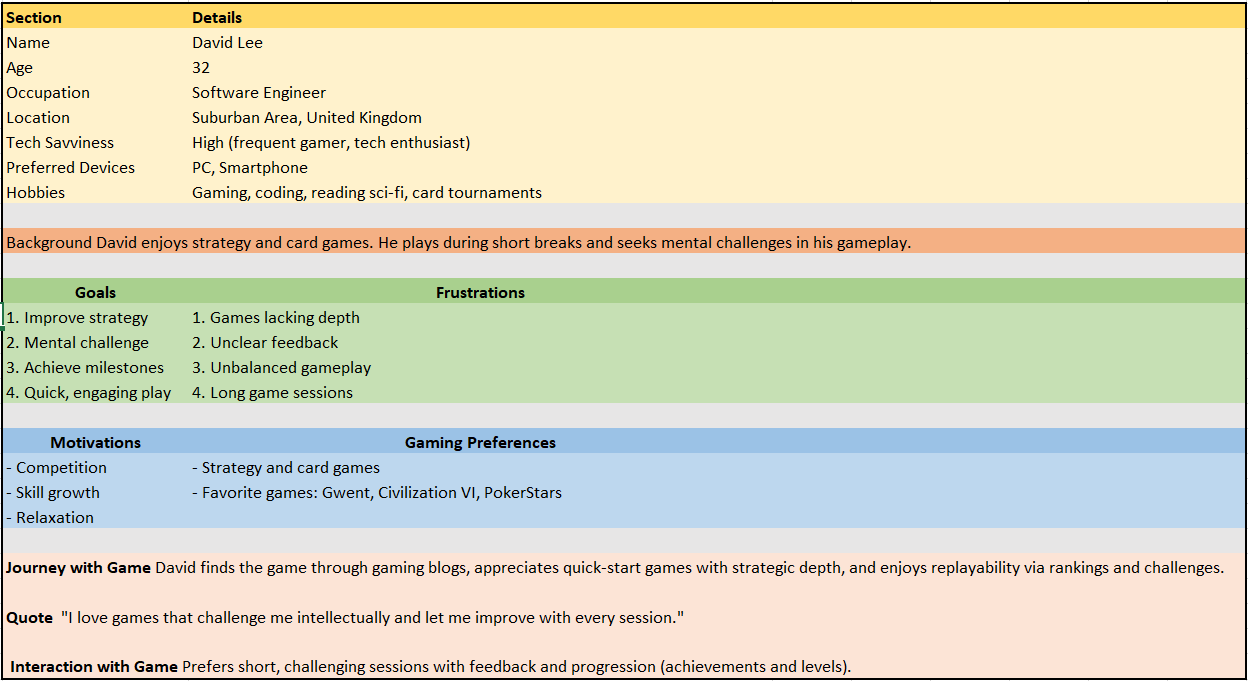
This Activity Diagram represents the flow of a card-based game where the player selects and submits cards for evaluation. The game starts with "Start Game", followed by the player selecting and submitting cards. A while loop checks if the submitted cards are correct. If the cards are incorrect, the loop repeats, allowing the player to resubmit cards until they are correct. Once the correct cards are submitted, the game ends with the "End Game" activity. The loop ensures continuous evaluation until the correct result is achieved.



# Customer Journey Map

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stage | Player Actions | Player Thoughts | Player Emotions | Touchpoints |
| Discovery | Finds the game (through app store, recommendation, etc.) | "This game looks interesting!" | Excited, Curious | App Store, Game Website, Ads |
| Installation | Installs the game on their device | "Let me see how it works." | Enthusiastic, Eager | Game download/install process |
| Game Start | Launches the game and starts a new session | "I hope this is fun and easy to play." | Curious, Hopeful | Game interface (main menu) |
| Card Selection | Selects cards to play in the game | "Which cards should I choose?" | Engaged, Focused | Game screen (card selection) |
| Submit Cards | Submits the selected cards for dealer evaluation | "Will these cards be accepted?" | Anticipation, Slightly Anxious | Submit button (game interface) |
| Dealer Feedback | Receives feedback on the card selection (correct/incorrect) | "Did I make the right choice?" | Excited (if correct), Frustrated (if wrong) | Dealer feedback screen |
| Game End | Game ends (either win or lose) | "That was fun! Should I play again?" | Happy (if win), Determined (if lost) | Game completion screen (result) |

# Persona



# Testing Strategies

The primary objective of this testing strategy is to:

Ensure the game meets functional and non-functional requirements. Identify and resolve any defects in gameplay mechanics, user interface, and backend systems. Validate the game’s performance, security, and user experience across multiple platforms and devices.

## Scope of Testing

The scope of testing will include:

* Functional Testing: Testing the core functionalities of the game, including card selection, dealer evaluation, feedback mechanisms, and game progression.
* Non-functional Testing: Testing the performance, security, usability, and compatibility across various devices.
* User Acceptance Testing (UAT): Verifying that the game meets the expectations of end users.

## Testing Types

### Functional Testing

This type of testing will focus on validating each feature of the game.

* + Unit Testing: Verifying individual functions, such as shuffling cards, evaluating selected cards, and submitting feedback.
  + Integration Testing: Ensuring modules (e.g., card selection and dealer evaluation) work together smoothly.
  + End-to-End Testing: Testing the complete game flow from the start of a new game to card submission, feedback, and game conclusion.
  + Smoke Testing: Performing a basic test to verify that the game’s key functions (e.g., card selection, game start, dealer evaluation) work after each build.

### Non-Functional Testing

These tests validate the game's performance, scalability, and user experience.

* + Performance Testing: Ensuring the game runs smoothly under normal and peak loads (handling multiple players simultaneously without lag).
  + Security Testing: Checking the game for vulnerabilities such as unauthorized access, data security issues, and payment system flaws (if any).
  + Usability Testing: Validating the user interface for ease of use, accessibility, and intuitiveness.
  + Compatibility Testing: Ensuring the game works across different devices (smartphones, tablets, PC) and operating systems (iOS, Android, Windows).

### User Acceptance Testing (UAT)

* + End users or focus groups will test the game to ensure it meets their expectations in terms of gameplay, difficulty, and enjoyment.