**TopCard – A Fun Three Card Java Game**

A close-up of a deck of cards

Description automatically generated

**Revision History:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Revision** | **Author** | **Description** | **Document Tracking** |
| 10-23-2024 | 0.1 | Rajesh Rajchal | Game description | **Document ID:** TopCard  **Version:** 0.1 |
| 11-01-2024 | 0.2 | Rajesh Rajchal | Added Domain | **Document ID:** TopCard  **Version:** 0.2 |
| 11-08-2024 | 0.3 | Rajesh Rajchal | Added Interface | **Document ID:** TopCard  **Version:** 0.3 |

Table of Contents

[1. Introduction 1](#_Toc181979112)

[1.1. Intent 1](#_Toc181979113)

[1.2. High Level Requirements 1](#_Toc181979114)

[1.2.1. Registration 1](#_Toc181979115)

[1.2.2. Authentication 1](#_Toc181979116)

[1.2.3. Profile 2](#_Toc181979117)

[1.2.4. Update 2](#_Toc181979118)

[1.2.5. Point Purchase 2](#_Toc181979119)

[1.2.6. Checkout 2](#_Toc181979120)

[1.2.7. Administration 2](#_Toc181979121)

[1.2.8. Key Technology Standards 2](#_Toc181979122)

[1.2.9. Future Enhancements 2](#_Toc181979123)

[2. TopCard Use Cases 3](#_Toc181979124)

[2.1. Intend 3](#_Toc181979125)

[2.2. Player Interaction Use Cases 3](#_Toc181979126)

[2.2.1. Register 3](#_Toc181979127)

[2.2.2. Login 3](#_Toc181979128)

[2.2.3. Authenticate 3](#_Toc181979129)

[2.2.4. Play Game 3](#_Toc181979130)

[2.2.5. View Profile 3](#_Toc181979131)

[2.2.6. Update Profile 3](#_Toc181979132)

[2.2.7. Purchase Points 3](#_Toc181979133)

[2.2.8. Bet 4](#_Toc181979134)

[2.2.9. Logout 4](#_Toc181979135)

[2.2.10. Invalid Login 4](#_Toc181979136)

[2.2.11. Insufficient Points 4](#_Toc181979137)

[2.2.12. Profile Update Error 4](#_Toc181979138)

[2.3. Use Case Diagram 5](#_Toc181979139)

[2.3.1. Use Case: Primary Use Case 5](#_Toc181979140)

[2.3.2. Use Case: Admin Interaction 6](#_Toc181979141)

[3. Fully Dressed Use Case 7](#_Toc181979142)

[3.1. Intent 7](#_Toc181979143)

[3.2. Player Use Case 7](#_Toc181979144)

[3.3. Flow of Events: 7](#_Toc181979145)

[4. UML 8](#_Toc181979146)

[4.1. UML Components 8](#_Toc181979147)

[4.1.1. Player: 8](#_Toc181979148)

[4.1.2. Card: 8](#_Toc181979149)

[4.1.3. Deck: 8](#_Toc181979150)

[4.1.4. GameService: 8](#_Toc181979151)

[4.1.5. BettingService: 8](#_Toc181979152)

[4.2. UML Diagram 9](#_Toc181979153)

[4.3. Relationships 9](#_Toc181979154)

[4.3.1. Deck and Card: Composition 9](#_Toc181979155)

[4.3.2. Player and Card: Aggregation 9](#_Toc181979156)

[4.3.3. GameService and Deck: Association 9](#_Toc181979157)

[4.3.4. GameService and Player: Association 10](#_Toc181979158)

[4.3.5. BettingService and Player: Aggregation 10](#_Toc181979159)

[5. Testing 11](#_Toc181979160)

[5.1. System Test: 11](#_Toc181979161)

[5.2. Card Distribution Test: 11](#_Toc181979162)

[5.3. Point Calculation Test: 11](#_Toc181979163)

[5.4. Betting Logic Test: 11](#_Toc181979164)

[5.5. User Interface Test: 11](#_Toc181979165)

[5.6. Performance Test: 11](#_Toc181979166)

# Introduction

## Intent

This document outlines the requirements for the card game Java application. The name of the game is TopCard. TopCard is a digital card game designed to be engaging and user-friendly, leveraging the power of Java to create a seamless and interactive gaming experience. The implementation focuses on providing a robust and enjoyable game environment, ensuring that all components, including the card management and player interaction features, are well-integrated and function smoothly together. The aim is to deliver a high-quality gaming experience that is both entertaining and technically sound.

Details of this report and code can be found in my repo: <https://github.com/rrajchal/MSSE670_OOSoftwareConstruction>

## High Level Requirements

This is a GUI-based card game using a standard 52-card deck, supporting 3 to 5 players. Before each round begins, players place their bets and the deck is shuffled. Each player is then dealt three cards. Based on the cards they receive players can win or lose money.

**Points Calculation:**

* Ace is worth 1 point.
* Cards 2 to 10 are worth their face value.
* Face cards (J, Q, K) are each worth 10 points.

The player with the highest total points wins. In the event of tied point totals, specific card rankings determine the winner: K > Q, Q > J, J > 10, 10 > 9, and so on. If multiple players have the highest point with the same total and card rankings, they both win and receive the betting amount from the other players.

### Registration

Players must register before entering the game, providing their name, date of birth, and email. The login information is securely stored in the database.

### Authentication

Players must log in each time before playing, providing correct login information. Players must be 18 years old or older to play.

### Profile

Players can view their profile information, including name, age, and points.

### Update

Players can update their profile information.

### Point Purchase

Players receive 100 points upon registration. Once all points are spent, they will be prompted to purchase more via debit or credit card.

### Checkout

After selecting points to purchase, players will go through the checkout process. After checkout is successful, the point will be added to their account.

### Administration

The admin user can update the profile of any player and add points to their account.

### Key Technology Standards

The registration, login, and credit card transactions will occur on a secure channel.

### Future Enhancements

Future enhancements will include online gameplay and additional card games, allowing players to choose from a variety of games.

# TopCard Use Cases

## Intend

This section outlines the use cases for the player.

## Player Interaction Use Cases

### Register

* The player provides their name, date of birth, and email.
* The system securely stores the login information.

### Login

* The player provides their login credentials.
* The system verifies the credentials and age (18+).
* The player is granted access to the game.

### Authenticate

* The player will be authenticated against the username and password stored in the database.

### Play Game

* The player places a bet.
* The deck is shuffled.
* The player is dealt three cards.
* The player checks their card points.
* The winner(s) is determined and announced.

### View Profile

* The player navigates to their profile.
* The player views their name, age, and points.

### Update Profile

* The player updates their profile information (name, email).
* The system saves updated information.

### Purchase Points

* The player opts to buy more points.
* The player enters debit/credit card information.
* The system processes the transaction securely.
* Points are added to the player’s account.

### Bet

* The player initiates the game by placing a bet.
* The amount of the bet is decided by the player and will be used in the betting round against other players.

### Logout

Players can choose to log out at any time, returning to the login screen.

### Invalid Login

* The player enters incorrect credentials.
* The system displays an error message and prompts for re-entry.

### Insufficient Points

* The player tries to place a bet but has insufficient points.
* The system prompts the player to purchase additional points.

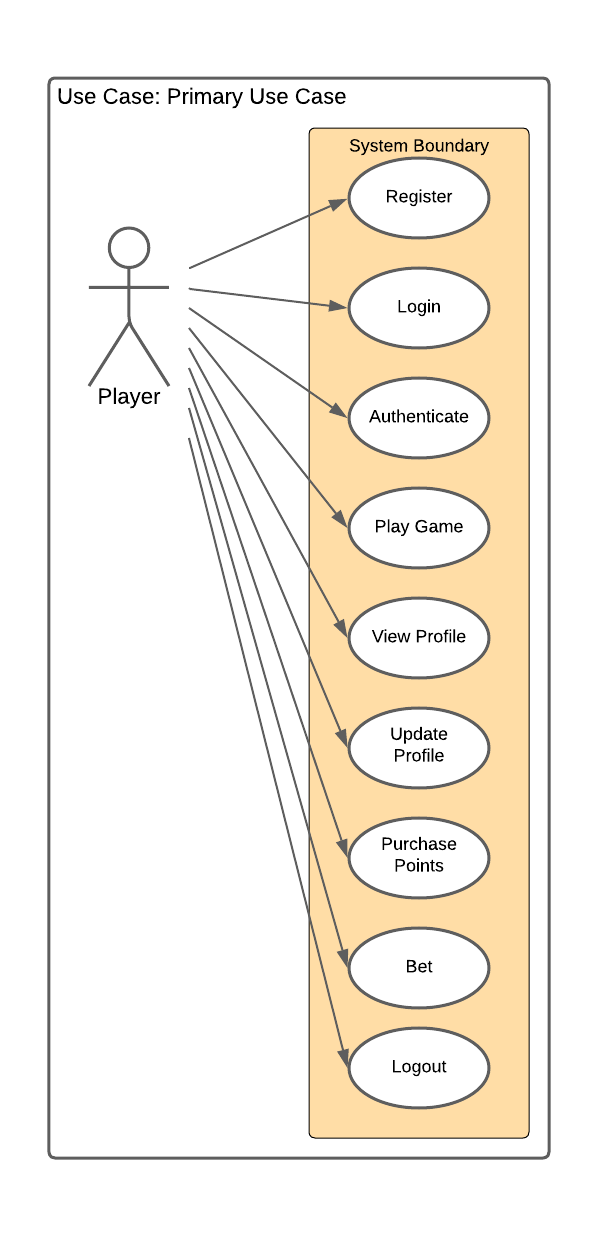
### Profile Update Error

* The player enters invalid data while updating their profile.
* The system displays an error message and prompts for correction.

## Use Case Diagram

### Use Case: Primary Use Case

This use case outlines the primary interactions of the player within the TopCard game. It covers essential activities such as registration, login, playing the game, viewing and updating the player’s profile, purchasing points, and logging out. These actions ensure that the player can engage with the game smoothly, manage their account, and continue playing without interruptions.



### Use Case: Admin Interaction

This use case details the responsibilities and activities of the admin within the TopCard system. It includes managing player accounts, viewing player statistics, adding points, and updating player profiles. The admin’s role is crucial in maintaining the system’s integrity, supporting players, and ensuring a fair and enjoyable gaming experience.

A diagram of a person

Description automatically generated

# Fully Dressed Use Case

## Intent

This section outlines a fully dressed use case for players.

## Player Use Case

**Primary Actor:** Player

**Assumption:** Player already existed in the TopCard database

**Stakeholders and Interests:**

* **Player:** Wants to play the game, manage their profile, and purchase points.
* **Admin**: Wants to manage player accounts and monitor activity.

**Pre-condition:**

* The player has successfully registered and logged in.

**Post-condition:**

* The player can play the card game, view and update their profile, and purchase additional points if necessary.

## Flow of Events:

* + 1. When this application is run, players are prompted to either enter their login username and password or register.
    - Login will take the player to start playing the game.
    - Register will take the player to registration page. After registration, they will be taken to start playing the game.
    1. Purchase will take the player to the purchase page followed by checkout page.
    2. Logout will take the player to 3.3.1.

# UML

The Unified Modeling Language Diagram provides an overview of the system by displaying its classes and relationships.

## UML Components

### Player:

Represents a player with attributes such as name, date of birth, admin, points, and cards.

### Card:

Represents a card with attributes like suit and rank.

### Deck:

Represents 52 cards, manages the collection of cards and shuffling mechanism.

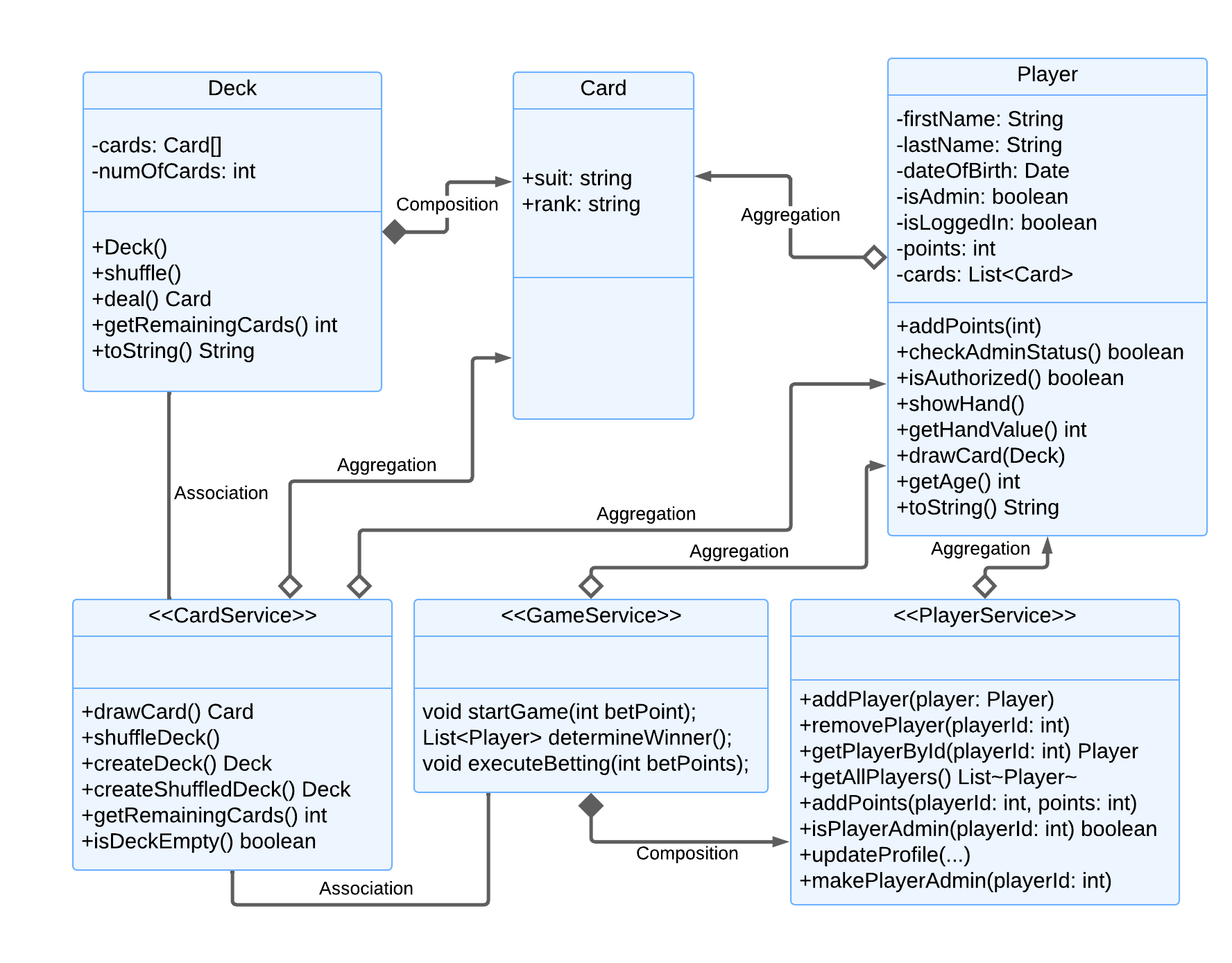
### GameService:

Contains the business logic for the game, including card distribution, point calculation for individual and three cards, and determining the winner.

### BettingService:

Manages the betting logic and calculations.

## UML Diagram



## Relationships

### Deck and Card: Composition

A Deck is composed of multiple Card objects. The Card objects cannot exist independently of the Deck. The Deck manages the lifecycle of Card objects, including shuffling and dealing. Cards are integral parts of the deck, and their existence is dependent on it.

### Player and Card: Aggregation

A Player aggregates multiple Card objects in their hand, meaning the cards can exist independently but are aggregated within the player’s context.

### GameService and Deck: Association

The GameService interacts with the Deck to manage the flow of the game, such as shuffling and dealing cards. The GameService uses the Deck to control game mechanics like dealing cards to players but does not own the deck.

### GameService and Player: Association

The GameService manages interaction with players, such as updating scores and managing turns. The GameService handles game logic and interacts with multiple players. Players participate in the game through the GameService.

### BettingService and Player: Aggregation

The BettingService aggregates Player objects, managing their bets during the game. The BettingService manages bets placed by players. Players can exist independently of the BettingService, but their bets are aggregated by the service during gameplay.

# Testing

This section outlines the testing strategies to validate the game’s functionality and performance. Unit Testing will involve thorough testing involving the following tests.

## System Test:

Verifies the core functionalities of the game, ensuring methods return expected results.

## Card Distribution Test:

Confirms that the deck shuffling and card distribution work correctly.

## Point Calculation Test:

Ensures that points are calculated accurately based on the cards dealt.

## Betting Logic Test:

Validates that the betting mechanism functions as intended and calculates wins and losses correctly.

## User Interface Test:

Ensures that the GUI elements are displayed correctly and respond to user interactions as expected.

## Performance Test:

Measures the game’s performance under various loads to ensure it remains responsive and stable.