StakeFree Digital Casino

EECS Team 6

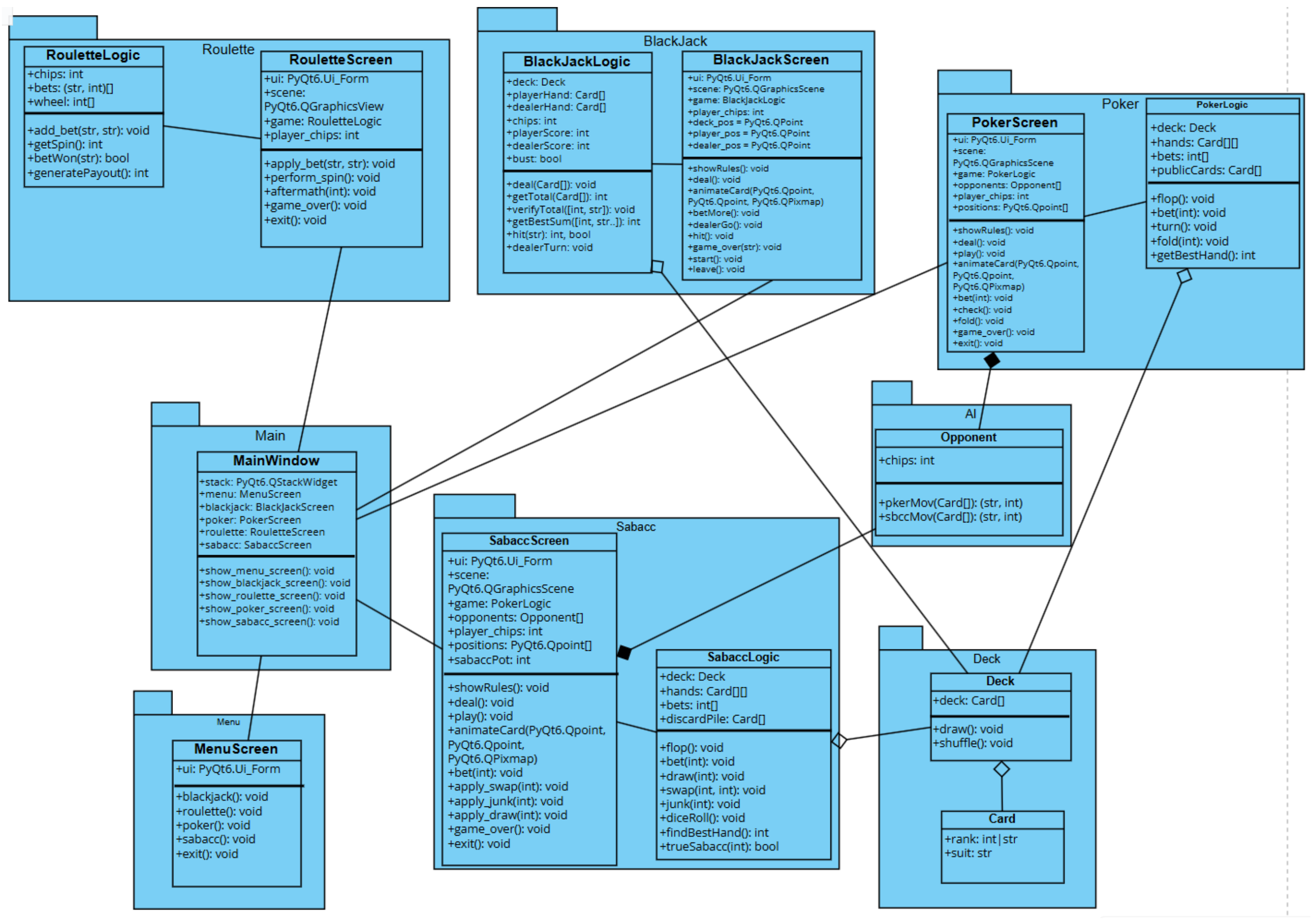
Bisshoy Bhattacharjee, Maximilian Biundo, Gavin Billinger, Mark Kitchin, Joshua Welicky

Project Synopsis

StakeFree Digital Casino provides gamblers with the opportunity to play games such as Roulette, Blackjack, Poker, and Sabacc without the risk of financial loss.

Architecture Description

The UML Class Diagram for the initial architecture of StakeFree Digitial Casino is shown below.



Dependencies

This project utilizes the PyQt6 libraries to implement GUIs. To install all required libraries, enter "pip install pyqt6-tools". This library provides a designer program that allows for the customization of a GUI in a user-friendly manner, saving the resulting GUI as a .ui file. This project also takes advantage of another tool provided by PyQt6, pyuic6, which converts a .ui file into a Python class for ease of use. Each element in the GUI is made available as a class attribute.

Components

MainWindow

- Description:
    - Core skeleton of the project. Manages switching between the menu GUI and game GUIs, as well as ultimate program termination.
- Attributes:
    - stack: A PyQt6.QStackedWidget instance that will store all of the GUI handlers, allowing switching.
    - menu: A MenuScreen instance which handles the main menu GUI.
    - blackjack: A BlackJackScreen instance which handles the blackjack GUI.
    - poker: A PokerScreen instance which handles the poker GUI.
    - roulette: A RouletteScreen instance which handles the roulette GUI.
    - sabacc: A SabaccScreen instance which handles the sabacc GUI.
- Methods:
    - show_[screen]_screen():
        - Simply switches to the desired screen (menu, blackjack, poker, sabacc, or roulette).

Card

- Description:
    - Basic representation of a playing card.
- Attributes:
    - rank: The integer value for a numbered card, the string name for a face card or ace.
    - suit: A string name of the suit of the card (spade, club, heart, diamond).

Deck

- Description:

    - A representation for a deck of cards.
- Attributes:
    - deck: An array of Card instances.
- Methods:
    - draw():
        - Randomly selects a Card instance and removes it from the deck attribute, returning the Card as well.
    - shuffle():
        - Reinitializes the deck attribute, clearing it and replacing every Card instance.

Opponent

- Description:
    - Represents an opponent in a game of Poker or Sabacc.
- Attributes:
    - chips: A randomly generated number of chips available for the opponent to bet.
- Methods:
    - pkerMov(Card[]):
        - Implements an opponent's decision on whether to bet, check, or fold during a turn in poker. Takes in an array of Card instances, which represents the opponent's hand. Returns a tuple containing a string choice(“bet”, “check”, “fold”) and an integer. The integer will represent the number of chips to bet if the choice is to bet. The number will be unused otherwise(set to zero).
    - sbccMov(Card[]):
        - Similarly implements the opponent’s decision in a sabacc game. Returns a similar tuple, containing the determined move and possible bet amount.

GAMEScreen

- Description:
    - Each game(GAME is here as a placeholder) implemented in this project will have a GAMEScreen class, which connects directly with the MainWindow instance and manages the GUI of the game.
    - For conciseness constraints, this will be a brief introduction to the core concepts of each GAMEScreen class. For more detail on each GAMEScreen class, refer to future sprint architecture documents.

- Attributes:
    - ui: A PyQt6.Ui_Form that generates the initial state of the GAME’s GUI.

    - scene: A PyQt6.QGraphicsScene that enables the addition of animated sprites to the GUI.
    - game: The corresponding GAMELogic class representing the actual actions of a game, such as drawing a card, betting more chips, determining a payout or loss, and more.
    - player_chips: Corresponds to the number of chips in the player's balance.
- Methods:
    - showRules():
        - Connects to a GUI button to show the rules of the game.
    - start():
        - Initiates the basic game flow of the game, leveraging the corresponding game class.
    - game_over():
        - Applies the determined chip payouts or deductions to the player's balance. Removes a player to the main menu if the player's chip balance reaches zero. Resets the game state to allow for replays.
    - leave():
        - Allows the player to return to the main menu. Implements forfeiting of chips that are currently bet if the user quits in the middle of a game.
    - Button handlers/helper functions:
        - Each game GUI will have slightly different buttons, and therefore slightly different handlers of the buttons.
        - Please refer to subsequent sprint architecture documents for more detailed information.

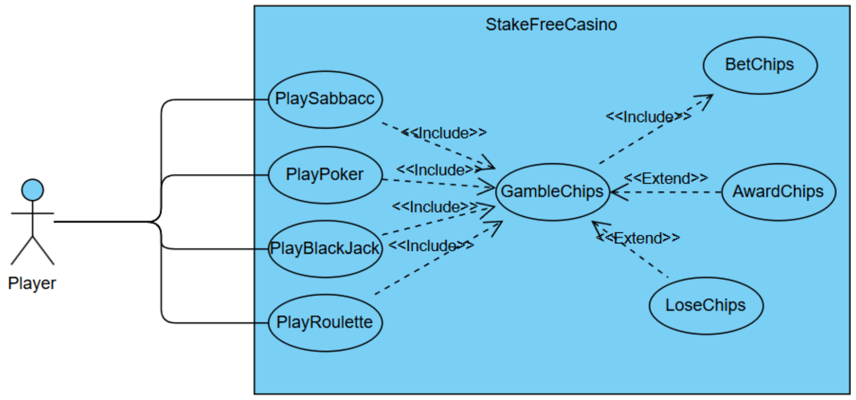
GAMELogic

- Description:
    - This class represents the core logic of a game. It is used by the GAMEScreen class to render the appropriate actions to the GUI. Depending on the specific game, the GAMELogic class will handle the logical actions like dealing a card, discarding a card, spinning a wheel, rolling dice, and others. It may also determine chip payouts upon victory and deductions upon failure.
    - It is crucial to note that the GAMELogic class makes NO DIRECT ALTERATIONS to the GUI.
- Attributes/Methods:
    - The attributes for each GAMELogic class are highly dependent on the particular game.
    - However, all GAMELogic classes will have some kind of attribute for **staked chips.** While the GAMEScreen class keeps track of what chips the user currently

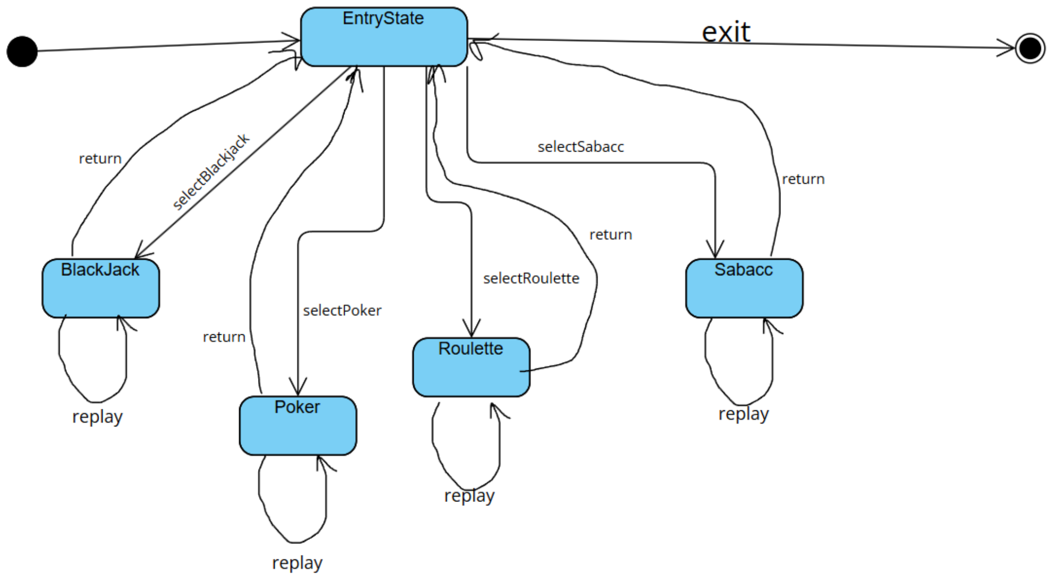
possesses, the GAMELogic class tracks the amount of chips a player stands to lose or win.

Appendix

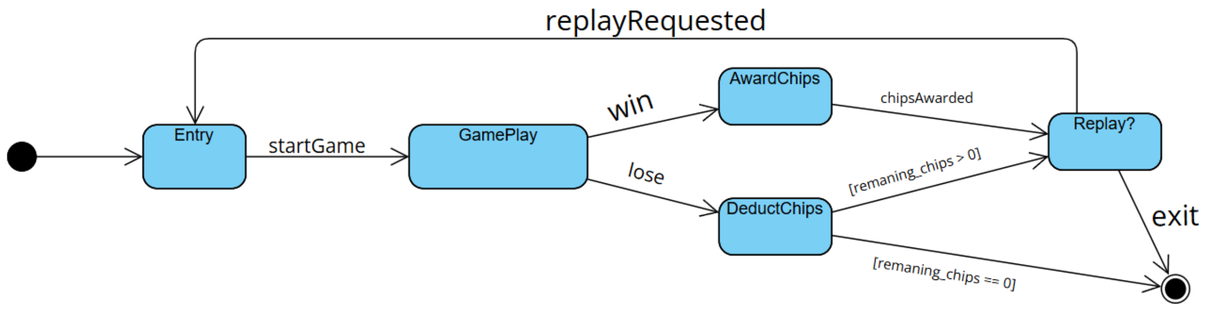
A UML Use Case Diagram for the project is shown below.



A basic UML State Chart Diagram used for initial program-flow modelling is shown below.



Another basic UML State Chart Diagram used to model the basic flow for each game.



replayRequested
Entry
startGame
GamePlay
win
lose
AwardChips
DeductChips
chipsAwarded
[remaning_chips > 0]
[remaning_chips == 0]
Replay?
exit