Project Summary

Project Title

“Terminal Casino”

Team members

- Jay Seabrum
- Weitung Liao
- Dananjay Srinivas

Overview and ambition

Terminal Casino is a simple console based computer-game, where a user can experience the thrills of a casino in their preferred terminal window.

We hope to build an engaging game, where a user will start with a given amount of cash - which they have to maximise by making prudent bets. The user can make these bets by playing their pick of 3 games-of-chance : BlackJack, Slots, Roulette.

As mentioned earlier, the game will be served via text terminal and will be built with Python.

Project Requirements

Responsibilities

Project responsibilities can be trifurcated into 3 major constituents:

1. **User management**

    a. **Managing user data, user profile**

The system should allow the user to access their profile and edit information like their display name and password. The user should also be able to access their statistics such as win/loss ratio and cash in hand.

    b. **Implementing authentication, and session management**

The system should allow connections via password authentication, and should be able to manage a user session with proper logout. Since the application will not be online, we do not expect to design for network discrepancies (persistence of session is not a constraint).

    c. **Providing forms for user registration, validating user credentials**

There must be a proper login form that validates user credentials such as the integrity of their password. Care must also be taken to only allow qualifying individuals to play the game (persons below the age of 18 may not be allowed, for instance.)

2. **UI Responsibilities**

    There are 3 kinds of overarching UI themes:

    a. **Display Game Screens**

Given that text terminals are usually quite drab, we need to produce a quality view that represents the game that is being played in an intuitive and engaging way. We hope to achieve this by incorporating ASCII art and unicode symbols (for instance, cards can be represented by their deck like so : A♠)

b. Display Segues, Transitions

This responsibility primarily corresponds to displaying and navigating menus. Again, because text terminals limit our ability to navigate easily - we expect to use Gameboy-esque navigation where each menu option will have a key binding.

This responsibility also takes care of the user journey - which screen to load given a particular event and so on (example : a different screen will be presented for a victory and defeat).

c. Display User Info

Users must have a lucid and intuitive experience when navigating their profile. Options such as editing, reversals etc must be accessible.

3. Games and Control Logic

This section is slightly self explanatory. We need to ensure that the games are implemented well; and the rules are true to the game. There must not be weird edge cases (example: two participants at the BlackJack table cannot be dealt the same card).

a. Serving BlackJack

A player may choose between playing vs the dealer (compulsory) + 2 other AI participants (optional). Each round has a buy-in and a bet that is determined by the player before the cards are dealt. The player goes first - and can choose between “Hit” (drawing a card from the deck) or “Stay” (finalizing the total).

The objective of the game is to get a set of cards that sums as close to 21 without going over. Face-Cards (K, Q, J) are considered 10 points. Ace is considered 1 or 11, whichever allows the sum to be legal.

The entity closest to the sum of 21 is the winner. The winning pot will be distributed amongst tied highscores.

The AI will bet on the basis of expected value of winning. There will be inefficiencies introduced into the AI - it will have a chance of making a substandard bet. This chance of inefficiency will be tuned to ensure the game is captivating for the player.

b. Serving Slots

A player comes to a 3x3 grid of slots.
They must input between $5-$60 (inclusive) before they are allowed to spin the slots.

The house keeps 20% of the original bet, meaning $4-$48 actually passes on to the next step.
If a row, column, or diagonal has all equal inputs (ie: 7-7-7, or B-B-B, etc.) then that constitutes a win for the user.
The user will then gain 1.5 times the amount of the remaining bet that came through, resulting in a win of $6-$72 for the user (which ends up being a net gain of $1-$12 for the user).
If there is not a matching row, column, or diagonal, then the entire rest of the bet is lost.
If there are multiple row, column, diagonal matches, each additional match over the first match will result in an additional 10% increase to the winnings that are given to the user.
The maximum number of matches in slots is 8, meaning that if a user gets a board that looks like this:
7 7 7
7 7 7
7 7 7
That means they got 1 match plus 7 additional matches.
They will receive (1.5 + 7*0.1) = 2.2 times the amount of the remaining bet.
Maximum earnings will then be $8.80-$105.60, which is a net gain for the user of $4.80-$57.60.

c. Serving Roulette

A player will start with 2500 credits, he can put a number of tokens as many as he wants in each bet.

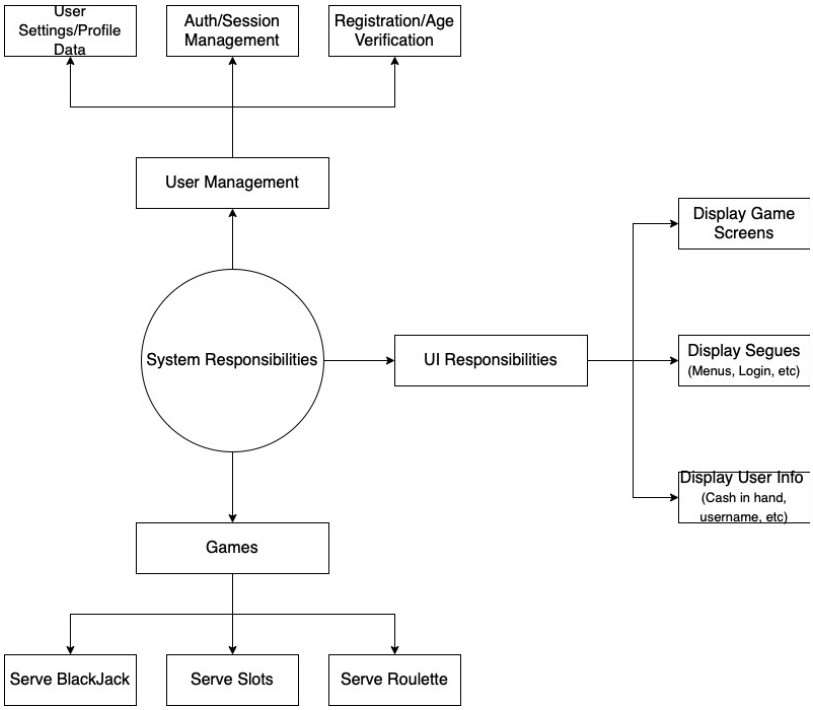
There will be a spin table, which will randomly generate a number between “1” to “36" or “0”, “00". And come with a color in red, black or green.

Player can bet a number between 1 to 36 or 0, 00. He/She can bet a range of numbers such as 1-12 13-24 25-36 1-18 or 19-36. He/She can also bet the number as even or odd, bet the color as black or red.

Player can put different credit on each bet and can make different sets of bets. For example “bet 5 credit for even number, bet 10 credit for number 18” is possible in one round.

Every type of bet comes with a different return rate. For example, bet even number could win back double of the credit put, bet range 1-12 could win back triple of the credit put.

Game ends when the player's credit is down to zero, or the player chooses to quit the game.
The system would save the highest credit player win during one game as highest score.



Requirements

1. **Functional Requirements**
    a. **User needs to track, edit and see their profile**
    b. **User should be able to create a new profile, if they don't have one**
    c. **Game should be able to verify identity, disallow duplicate accounts and underage users**
    d. **User should be able to access main menu, and quit with ease**
    e. **User should be able to select the game they want to play**
    f. **User should be able to play choice of game without any ambiguity**
    g. **Game should automatically add the results of the game to the W/L of the User**
    h. **Game should adjust "in-game Cash" the user has based on their winnings**

2. **Constraints**
    a. **Game can play on Localhost and LAN only (Network)**
    b. **OS Agnostic if >=Python3.8 present (Software and Platform)**
    c. **64 bit architecture preferred (Hardware)**
    d. **Admin/SU privileges may be required to build from source (Permissions)**
    e. **Write access to the disk is required (Permissions)**
    f. **2 Sprints (Scheduling)**
    g. **Test Coverage (Class Policy)**

3. **Non-Functional / Interfaces**

    a. **Database ORM Interface**

Since all the games share a common database, there must be a backend API where we can push and retrieve game information from the user.

    b. **View/Animation Control**

Most terminals are limited in what they can display, and since we are only using text - we need to have a view module that manages the animations.

Additionally, certain elements of the UI may be shared across games and menus (such as time, display name and cash in hand).

Hence it is good to have an Animation control unit.