**Casino Sim - Project Description**

**<names>, Zach Ray**

**Applied Programming Concepts (ELEC-3225-01)**

**July 24th, 2022**

**Synopsis [from project description on brightspace]:**

Simulate a casino environment from the standpoint of the casino management. Track money, games, users, etc. Will need many casino games (and multiple versions of the same games) to be run in a statistically accurate environment (blackjack, poker, roulette, craps, slots, etc.). Must have multiple users of different strategies and bankrolls. Track user gambling history (wins, losses, games). Record the amount lost/won individually and for the entire casino.

**Introduction**

Many components will be required to function together to simulate a casino environment. Just to name a few: we will need a database with multiple relations (games, users, money, etc.), classes to control user functionality as well as access to the database, and a clean and intuitive user interface. At the end of the semester, there should be multiple playable games. We’re shooting for 5 games – we can also create multiple versions of the same game. Users should be able to sign in, view their winnings, play games, and bet in the casino. There should be a separate view for admins which allows one to see how the casino is performing. Admins should have a neat menu, be able to display info from the database, and update / remove entries in the database.

**Current Functionality (6/15/22)**

Functionality so far has been added in unison with the lecture topics. As of now, we have a database with two relations: player and slots. Each game will have their own relation – there is a base game model and table which is created when a game is added. There is an abstract base class – user. Player is inherited from user. Players currently store a unique id, user id, name, password, and winnings. Player also stores an integer to identify if they are banned or if they are an administrator. Players can be caught cheating and will be banned from playing games. Administrator functionality will give access to all of the back-end. It is easier to store an administrator, or few administrators, inside of the player relation – rather than create a whole new table just to store a few admins.

**Requirements analysis and definition**

**Major Components**

* Database with multiple tables
  + Player table (store user id, name, winnings, password, ban state, admin state)
  + Game table (store history of games, amount won, who won, when they won)
  + We should be able to track all player’s gambling history, gains/losses for each game, and be able to view different gains / losses over different time intervals.
* Multiple Interfaces – can be text or graphical
  + Player interface – access games, winnings
  + Game interfaces – each game will have their own unique interface
  + Admin interface – access and views on back-end
* Object Oriented Programming (OOP)
  + Base class User – derived classes for players and administrators
  + Games will be controlled through classes
* Playable games
  + Players should be able to sign into the casino and play games
  + Poker, roulette, blackjack, craps, and slots
* Simulation
  + Should be able to simulate the casino as if 100’s of players were playing games and betting
  + Mimic real random games by the 100s
  + Integrate / change games and determine efficacy
  + Determine best number of each game to maximize profit
* Casino Environment
  + “Will need many casino games, and multiple versions of the same game”

**Interface**

Multiple interfaces will need to be created. There should be a player interface – a menu which allows the user to choose from playing games, checking their account balance, and making bets. Each game will need to have their own interface for playing the game. Some may be graphical, such as our slots game. There should also be an admin interface. A menu which allows us to add / remove players, track user gambling history, track dealers / players / games over some time period, a way to update rules and more.

Class Functionality

Player

**User** is an abstract superclass. **Player** and **Admin** are both inherited from User. Each user in the casino will have a player class created for them. The player class will control how users can interact with the casino. Currently players can check their winnings and play games. In the future, players may be able to check their winnings over some time period, or from some specific game. It should be noted that management side will also keep track of specific players (high rollers, cheaters). Players should have the ability to make bets as well. In some games, like craps for example, anyone around the table can make bets on the current shooter. So, players don’t necessarily have to be the ones rolling the die / playing the game to make bets.

Admin

Admin is inherited from User. Admin is stored in the same relation as player. We do this for several reasons. First, there will be a limited number of administrators, so it doesn’t make sense to create a whole new table just to store few entries. Secondly, administrators will have functionality which allows them to add / remove / update players. Instead of having to interact with two separate relations – all of it can be done inside a single table.

Admins will have an interface separate from the player interface. The management side is probably more important than the user side. At the end of the day, we should be able to access all sorts of back end info – and display it in neat and easy-to-understand ways. Whether it be formatted text inputs or graphical displays – it should display information neatly. When we simulate the casino, we will need to add many players (100s), which can all be done through the administrator class. We should generate many players with different strategies, betting habits, and bankrolls. The admin should be able to test how the casino performs over different time intervals. The admin should be able to simulate new rules to our games and test to see how these changes impact the casino.

**ADMIN METHODS**

1. Add Player – the admin should have a choice in their menu which allows them to add a player. Adding a player will continuously prompt the admin for the various attributes player contains. To assist with simulation, we should make a function which generates attributes of new players (random username, password, winnings etc.)
2. Remove Player – the admin should have a choice in their menu which allows them to remove a player by inputting the id of the player. We will still store the removed players gambling history, as it will be stored in separate game tables.
3. Update Player info – admin should be able to input an id of a player - change username, password, adjust winnings.
4. Add Game **–** In addition to adding full-fledged games, we should be able to add a new version of the same game – with slightly altered rules / odds.
5. Remove Game
6. Update game info
   1. Not all games will have rules which can be changed
   2. Some games, like slots for example, will be easy to adjust the rules / odds
   3. Each game will have their own unique prompts for the admin. Different games = different rulesets = different prompts
7. Print gains / losses of games – the admin should have a choice in their menu to allow them to print history of game info. When selected, they will be prompted for which game they want the history of – and the time interval for which they want the data from.
8. Print gains / losses of players - the admin should have a choice in their menu to allow them to print history of player info. When selected, they will be prompted for which player they want the history of – and the time interval for which they want the data from. This method will be more complex than game info – it will require querying multiple databases to find which games the player has played – and then gathering that info.

**Playable Games**

Multiple games are being created right now. We’re aiming to have five games – slots, poker, blackjack, craps, and roulette. Each game will have their own table. History of all the games will be stored here, and accessible from the admin. Each entry will store the player, how much they won, and a time stamp. Most games should have rules / variables which can be altered to test how profits change. The casino should eventually be able to be simulated with 100s of players, and we should be able to identify how changing rules impacts the casino. One view which the admin should have is gain / loss for any game over specific time intervals. Some games will involve betting which will either need to be an additional column in the games relation or a separate entity itself.

Poker – add descriptions

Blackjack

Craps

Roulette

Slots

**Simulation**

For simulation, we decided on using a package called Faker. Faker generates fake data. There is a list of standard and community “providers” which have different properties to generate types of data such as names, addresses, birthdays, and many more. One can also create their own provider based off other providers. We will use faker to generate thousands of players and test the user functionality (login, update info, check winnings). We can also create thousands of entries inside the games database – including over specific time intervals. It’s not the same as user’s inputting data and playing the games, but with restrictions on what can appear in the entries, the database will look as if thousands of users are playing the games. This will allow us to test the functionality of our graphs and the management views of the casino.