Blue Badge Casino

Blue Badge
Team 2
v2 (wireframe)

Goal / Purpose of the App

Build an online casino model which demonstrates the use of an API structured with n-tier architecture and REST principles to access a relational database. The business logic for the necessary tasks resides in the API data layer. Assigning user roles will separate players, admins, and a master “house” account.

The casino gives a user/player the option to play up to five games. Initially, the user may create a player account, make a deposit, then choose from three games and place a bet. If the player wins, the funds are immediately added to his balance. Otherwise, the account is decreased by the bet amount. After placing enough bets or wagering a certain amount, the player may increase his status to a higher tier, and access the SilverLevel or HighRoller game with altered parameters and greater risk/reward mechanics.

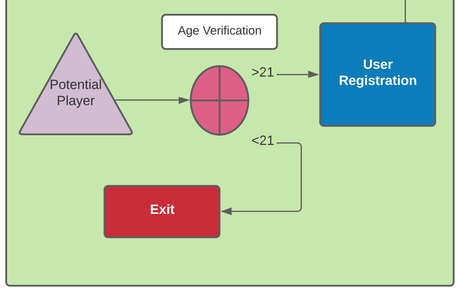
Utilizing user roles, a player has the ability see her transaction history, bet history, and profile. An admin has the ability to generate reports for the individual games showing the profit for each game by day, week, month, etc. The admin can also see reports for each player, all players, all bets ever, all bets in a day, etc. This allows him to analyze the data to make decisions or changes for the casino. The house account encompasses the house bank balance, filling the role of the casino bank secured by cash.

User Story

Samson is an entrepreneur that has been involved in several online startup projects, including a rideshare app, a lead generator program for small service business, and an online auto parts store. He is currently partnering with a small group of investors on two additional projects; a food delivery app and cryptocurrency app. He likes to jump on the latest trend, so with more and more states legalizing online gambling, the online casino was the obvious next project idea.

He came to us here at MigrationStation Development with his requirements and requested we build his program. He wants to keep it simple with only a few games. He wants our initial working project to only simulate the casino experience; he will build his following by providing an ad-free, “play money” version, and after its success he will go live and start taking actual bets for real money.

Pre-access and Basic Info Capture



DATA TABLES | **COLUMNS IN TABLE W/ DATA TYPES** | **END POINTS** | **NOTES**

Player

Captured at account creation, flows into Player

- int PlayerID [KEY]
- GUID PlayerNumber
- string PlayerFirstName
- string PlayerLastName
- string PlayerPhone
- string PlayerEmail

Basic User Properties

- string PlayerAddress
- enum PlayerState [state1-50]
- DateTime PlayerDob
- DateTime Account Created

Additional Player Properties

- bool IsActive
- enum TierStatus [1-3]
- bool HasAccessToHighLevelGame

Tier Properties

- double CurrentBankBalance
- virtual <List> BankTransaction

Bank Property

- Bets
- virtual <List> Bets

Bet Property

Create (POST)
Get ALL Players (GET)
Get Player by ID (GET)
Get Players by TierStatus (GET)
Get Players w/ + Balance (GET)
Get active Players (GET)
Update Player by ID (PUT)
Delete Player (DELETE) (admin)

Age verifcation and basic account setup gathered at login, passed into Player setup

Tier/Status as a function of age of account, number of bets placed, and total amount of bets placed. Tier 3 = bool HighRollerGameAccess true

Bet

- int BetId [KEY]
- [FK] GUID PlayerNumber
- [FK] int GameId
- virtual Game
- dateTime TimeOfBet
- double BetAmount
- bool AccountHasBalanceToCoverBet
- bool PlayerWonGame
- double BetPayoutAmount (can be negative if lose. use if/else statement to determine whether the payout will be +/win and use the game's multiplier, or -/lose and use a multiplier of -1)

Logic

- Players
- virtual <Player>

Post new Bet (POST)
Get ALL Bets (GET)
Get Bet by ID (GET)
Get Bets by PlayerID (GET)
Get Bets by GameID (GET)
Get Bets by date/range (GET)
Get Bets by amount/order by amount (GET)
Delete Bet (DELETE) (admin)

Bet as primary vehicle to carry information

Game

- int GameId [KEY]
- string GameName
- bool HighLevelGame (needs access granted by tier)
- Bets
- virtual List <Bet>
- double MinBet
- double MaxBet
- float OddsToWin (?)
- float PayoutMultiplier

Properties from ABSTRACT game class

- float InternalGameLogic (change for each game : formula / random num generator / game sim / etc to determine win / loss)

Game mechanics for each game implemented with interface

Post new Game (POST) (admin)
Get ALL Games (GET)
Update Game (UPDATE) (admin)
Delete Game (DELETE) (admin)

~~**Individual Game inherits from abstract Game class. Use of interface to ensure new Games have all the methods to be a functional game.**~~ **Those methods will hold game logic, and will determine the value of many of the properties.**

BankTransaction

- int BtId [KEY]
- [FK] GUID PlayerNumber / int PlayerId ?
- dateTime TimeOfBT
- double BtAmount (can be negative for withdrawal)

BT = BankTransaction

Post new BT (POST)
Get ALL BT (GET)
Get BT by BtID (GET)
Get BT by PlayerID (GET)
Get BT by date/range (GET)
Get BT by amount/order by amount (GET)
Delete BT(DELETE) (admin)

Log of Deposits (+) and Withdrawals (-) by POST

Informs Player Bank Balance

MVP "70%"

User roles / admin / security
- Seed database
- Age verification
- User creation, player will inherit demographics from user signup
- Update player info/demographics
- Players: deposit / withdraw money from account (tokens / security)
- Choose a game
- Place a bet, check for adequate funds
- Return win / lose status, amount won/lost, change bank balance
- Player GET bet history
- Player GET transaction history
- Player tier adjusts based on history of bets
- Access to highRoller games
- House/Admin GET all bet history
- House/Admin GET history by date/game/player
- House/Admin GET all players or players by tier
- See endpoints section for additional features
- ~~The admin has the power to transfer funds from each game when it reaches a certain point (or this could be automatic) and to add funds to a game if it has taken a few losses and needs to be replenished~~
- All casino winnings are held in house account
- Logic in service layer to determine win/loss

Stretch Goals

- Remote hosted db
- **Integrating 3rd party API to fund account**
- **Parameters for endpoints (one GET that uses parameters to filter results)**
- Credit
- Multiple Games
- Player vs Player
- Credit (tied to tiers?)
- Average bet: bank -> high roller / rewards
- **Additional logic for games**
- **Additional games**
- **Admin ability to create new game**
- **Swagger**