CIS*3260 F21 – Assignment 2
Ruby on Rails

Overview

- Takes the Coin, Die, Player, etc. classes written in Ruby from A1 and use them to create a website that implements a single player game played against a server
    - You may use your own A1 code, or use one that will be provided on Courselink … your choice
        - No marks will be awarded or removed whether or not you use your code or the provided code
        - If you use your code, make sure that the functionality required to code the application for this assignment is properly implemented
            - i.e. your A1 code won't be marked, but a lack of functionality in the website will be
- Multiple users should be able to sign into the server, with the server playing against each user independently
- The server is to be implemented using Ruby-on-Rails

Details

- Implement the following "four" web pages:
    - Sign-in Page
        - Sign-in section of the page
            - The user is asked for their email address and password
            - If the user does not exist, they are informed that they must set up a new user
            - If the user exists but the password does not match, they are informed that the password was wrong
        - Sign-up section of the page
            - asks for a user name, email address and password (can be anything)
            - On signup, the user is …
                - provided with three quarters and three six-sided white dice
                - has their overall points and amount of gems both set to 0
    - User's Page
        - Displays the user's name, overall points, gem count and contents of their bag
            - A returning user should have the same name, points and content as they had when they last left the game
        - The user is given a choice of playing a game or buying coins/dice (for these pages the "user" is referred to as the "player")
            - If "Play a Game" is chosen, the "Play Game" page is displayed
            - If "Purchase Items" is chosen, the "Items" page is displayed
        - The user can choose to sign out from this page
            - This returns to the "Sign-in" page
        - The user can also choose to delete their account from this page
            - which also signs them out

- Items Page
  - On this page a player can buy a coin or die of any number of sides, colour (for dice) and denomination (for coins)
  - The cost is 1 gem per side … examples:
    - a coin cost 2 gems
    - a 10-sided die costs 10 gems
  - When a die or coin is purchased
    - The item it is added to the player's bag
    - The gem cost is deleted from the players gem count
  - If the player requests a coin/die of a given number of sides, but does not have the gems required for purchase, the purchase is refused
  - The player can exit the Items page and return back to the User's Page at any time
- Play Game Page
  - Initial page setup
    - A randomized set of description is generated by the server as the "goal" based on the contents of the player's bag
      - the goal is comprised of up to 3 parts
        - A description of the coins in the throw
        - A description of the dice in the throw
        - Whether the coins/dice should be summed or tallied (mandatory)
      - The goal can contain either a coins description, a dice description or both
        - If there are no coins in the player's bag, you should not generate a coin description
        - If there are no dice in the player's bag, you should not generate a dice description
        - If there is at least one coin and one die in the player's bag you can randomly determine whether to use one description or two
      - See A1 for a details of a description
    - Displayed:
      - The "goal"
      - Player's bag contents
        - e.g. 3 coins and 3 dice
  - Player's turn
    - Load and Throw
      - The player loads their cup with a chosen number of coins and dice based on a series of descriptions (again see A1's definition of a description)
        - Note: you have flexibility of design here
          - points will be awarded for design quality
      - The player then throws the cup and sees their throw
        - the coins and dice are automatically placed back into the player's bag
      - The result is scored against the goal, but is not displayed to the user
    - The player can load and throw up to three times
      - The total number of dice and coins used across all throws are stored
    - The player may choose which throw to use (if there is more than 1 throw)
  - Server's Turn
    - The server randomly chooses coins and dice from a "duplicate" bag to load into a cup
    - The cup is thrown only once
    - The result is scored against the description

- Play Game Page continued …
  - The result of the game is displayed on the page
    - The description, the computer's throw and the players chosen throw, and the score for each is displayed on the page and the winner declared
    - If the Player wins,
      - the difference between the player's score and the computer's score is computed
      - This result is multiplied by a fraction equal to the number of items used in the (winning) throw over the total number of items thrown by the player when determining the choice
      - This result is added to the total score, and half of the result (rounded down) is added to the user's gem count
    - An option to start a new game is also presented at this time
  - At any time, the player can exit the Play Game Page and return back to the User's Page
    - If the game was not completed at the time of exiting, it is considered forfeited

Example: A player generates two throws
- one with 2 coins and 2 dice (4 items)
- one with 5 coins and 1 die (6 items)

(note: a total of 4 + 6 = 10 items were used)
The second throw was chosen and received 20 points
The server generated a throw that received 10 points
The player wins and gets awarded: (20 – 10) * 6 / 10 = 6 points
The player also receives 6 / 2 = 3 gems

Note: while conceptually the "Play Game" page is a single page, it can be implemented as sequence of pages "behind the scenes"

How to approach the assignment - recommendations

- Read the Ruby-on-Rails guide
  - https://guides.rubyonrails.org/getting_started.html
  - Note: you do not have to set up Ruby, SQLite3, Node.js, Yarn or Rails as they have all been provided to you on the Linux server, so you can skip that section
- Start with a "hello world" page as recommended in the Rails guide
  - Marks will be awarded for a "hello world page" if nothing else works, but will be ignored o.w.
- Add one web page at a time in the following order: Sign-in, User's, Play Game, Items
  - Note: while the web page ordering presented here is recommended, other implementation orders can be effective as well
- When working on a web page, add functionality one feature at a time
  - Don't try to write an entire webpage all at once
  - Unlike typical compiled languages, Ruby and Rails are designed to allow for code to be written incrementally
- If you want to "skip" a page for time consideration, the "Items" page is recommended
  - the website can function well enough without it, and it's completion is awarded the least number of marks

Grading Rubric

- "Hello World" is working
  - [**3 pts]** Shows connectivity from client to server and the ability to modify the routes file and view
- Sign-in Page
  - **[3 pts]** Demonstrates ability to use scaffolding to obtain and store info from a user into the database through the manipulation of Control and Model and View code
- Users Page
  - **[1 pt]** Requires similar skills as the Sign-in page, but now with control over the extra logic needed to switch between webpages as well as allowing for more than one URL request by the client on the server
- Play Game
  - **[1 pt]** Simple "game"
    - i.e., some manipulation of a throwing dice and coins from a cup
      - You can make up the "rules", just inform the TA what they are in your read.me
    - Demonstrates the ability to manipulate instances of some of the classes obtained from the Rails model in an OO fashion
  - **[3 pts bonus]** Full game
    - Demonstrates the ability to create an "application" with proper OO interaction, using instances obtained from the Rails model and accessed from the controller and use it to process the game logic and create game pages from the view to display
- Items
  - **[0.5 pts bonus]**
    - No new skills demonstrated
    - Just here to complete the game and make it more fun

Submission

- Submit your source code in a single zipped (or tarred) directory through Courselink
- Make sure your Rails server is running your website either at the time of submission, or when informed through Courselink that the TAs have begun grading, until informed that grading has completed
- The Rails application should be called "`CupThrow`"