CIS*3260 F22 – 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 username, 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 set to 0
    - User's Page
        - Displays the user's name, overall points, 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 signing out
            - If "Play a Game" is chosen, the "Play Game" page is displayed
            - If "Sign out" is chosen, the "Sign-in" page is displayed
        - The user can also choose to delete their account from this page
            - which also signs them out

- Play Game Page
  - Initial page setup
    - Displayed:
      - Player's bag contents
        - e.g., 3 x coins, 2 x 4-sided dice, 1 x 6-sided dice
  - Initial Loading
    - The player loads their cup from their bag with any die or coins they desire
    - The server loads its cup with the rest of the coins and dice from the bag
  - The server is then given the opportunity to switch cups
    - First the server sums the maximum points for each cup
      - E.g., if a cup has 2 coins and 2 six-sided dice, the maximum points is 1 + 1 + 6 + 6 = 14 points
    - If the total for the server is greater than that for the player
      i.e., `maxtotal(server) > maxtotal(player)`
      - the server does not switch cups
    - If the total of the server's cup is less than the players
      - The server computes a random number r between 0 and 1
      - If r `> maxtotal(server) / maxtotal(player)`
        - the server switches cups
    - If the computer switches cups, this is announced to the player
  - The player is given a chance to block the switch
    - If the player agrees to attempt the block,
      the following procedure is followed:
      - The player takes any item from the server's cup
      - The server takes the largest item from the player's cup
      - Both roll
      - If the player receives a higher value, the cups are not switched
      - The player then places the die they used into their cup
        (*which may or may not have been switched*)
      - The server places the die it used into its cup
        (*which may or may not have been switched*)
  - The player and server throw their cups and the results are displayed
  - If the Player wins,
    - The difference between the player's score and the server's score is computed
    - The result is added to the player's cumulative score,
    - A new die or coin, randomly determined, is added to the bag
  - 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

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
  - 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
  - **[2 pts bonus]** Full game
    - Demonstrates the ability to create an "application" <u>with proper OO interaction</u>, 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

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`"