Project Description

This is a Ruby on Rails application that consists of two pages. Page 1 allows uploading of images. Page 2 is a simple game that consists of showing a timer, an image, a button and an HTML table. When the button is clicked, the current timer value and image url is saved to a database table named "plays".

Specifications

1. Page 1: Allows upload of multiple images at one time (one must be able to multi-select the images to be uploaded using the system's Open File dialog).
2. Page 2 contains:
    1. An H1 element: This element displays a countdown timer that counts down from 10 to 1. When the timer reaches "1", it starts over from 10 again. The timer has a per-second resolution.
    2. An IMG element: The application will choose a random selection of up to 10 images from all player-uploaded images and store the url's in an array. With each tick of the countdown timer, this element will display an image from the array, in order. The set of selected images must not change unless the page is refreshed.
        1. Example: If 7 images were randomly chosen by the application (because the player uploaded only 7 images), the images displayed for each tick will be as follows:

Tick	Index of image in the array
10	0
9	1
8	2
7	3
6	4
5	5
4	6
3	0
2	1
1	2

    3. A BUTTON element: When the button is clicked, the current timer value and current image url, should be saved to a database table called "plays". At the same time, update the HTML table described in #4 below with this latest entry. All of this should be done via Javascript/Ajax etc. (the page must not be reloaded)
    4. An HTML table: On page load, this HTML table should be populated with all the entries in the "plays" database table (no pagination needed - show all records at once). The

HTML table has two columns: one shows the timer value and the other shows, in an IMG tag, the image associated with that record.

Notes

1. Use Ruby on Rails (any version)
2. Use vanilla JS only
3. App does not need any sort of user authentication or any other features other than those described above.
4. Tests are optional, but if you do write them, write tests only for what you think are the parts that would most benefit from tests (no need for Javascript unit tests).
5. Create a separate commit for the initial Rails project (this makes it easy for the reviewer to view each commit and be able distinguish between your code and third party code)
6. Zip the project and send it back via email or post it to your github account.
7. Write production quality code