Fullstack Sudoku App

Fullstack Development

Overview

For this final project, we will build upon project 2 to make a full stack version of Sudoku. This version of Sudoku will record user behavior and allow people to see high scores. When users enter your app, they should see a welcome page that contains nothing but the title of the game as well as ways to see the “rules and play the game page.

Reminder, for the rules of Sudoku: every column, row and sub-graph can only contain the numbers 1-X (where X is the height/width of the game board) exactly once.

For this assignment, you may work with up to two additional students, to make a group of 3. You can also tackle this work by yourself as well.

Finally, I recommend you use the code I provide in class as a basis for this project as much of the frustrating setup is handled there. There is a well setup template here: https://github.com/ajorgense1-chwy/cs5610_project3_template

Rubric

- Core Functionality - 25%
- Working Github and Deployed App - 5%
- Good Pages and Styling - 10%
- RESTful APIs - 20%
- MongoDB, Mongoose and Security Implementation - 20%
- Well Written JavaScript - 10%
- Writeup - 10%
- Bonus Points

Core Functionality and Pages

Here are the following pages we are expecting:

Home Page (/)

There should be a home page that includes the title of the game (you can change it from Sudoku if you wish) and any other art of your choosing.

Selection Page (/games)

This should be a game selection page. At the top of this page, there should be 2 buttons: “Create Normal Game” and “Create Easy Game”. When a user clicks on one of these, this will call the “POST” build game API (see below) to generate a new game and then redirect the user to that page with the game.

Additionally, every time a game is created, it is added to a list of games. This list of games is selectable from this page and a user may instead click one of these games to play instead of making a new one. The list will show the game name, difficulty, the username of the player who created it and the date it was created (in the format of Month Day, Year, like “Jan 1, 2020”.)

Game Page (/game/{gameid})

When a user clicks on an existing game or makes a new game, they are brought to this page. This should be able to handle both Normal and Easy Games and will allow users to play Sudoku nearly identical as in Project 2. The only difference is that there will not be a ~~“Reset” or~~ “New Game” button at the bottom of the page. There should be a button to reset the game at the bottom of the page.

If a user has already completed this specific game and returns to the page, the game should show as completed and have the solution on the page.

Rules Page (/rules)

As before, this page should have the rules of the game listed out. It should also include a “made by” or “credits” section that has a link to your email, Github, Linkedin, etc (you can use fake data if you prefer not to share these links.)

High Score Page (/scores)

~~This will be a list of all the users in the system and the number of games that they’ve won. The list should be ordered from most wins to least (ties based on username), and any user with 0 wins should not be shown on the screen.~~

List all the games in order by the number of players who have completed it. Any games where 0 people have completed it, should be ignored.

~~**Login Page** (/login)~~

~~There should be a login page that should allow a user to input a username and a password (which should obscure the password using the appropriate input type.) There should also be a submit button. While either input is blank, the submit button should be disabled. When the user clicks submit and if they are successfully logged in, then it should redirect to the /games page.~~

~~Please note that we will be using cookies to track user information and those should be set here.~~

~~**Register Page** (/register)~~

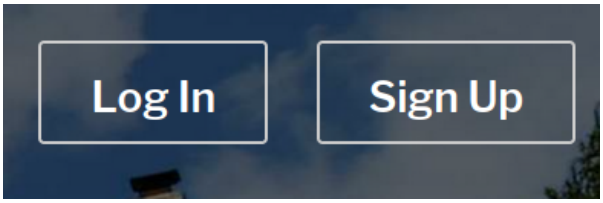
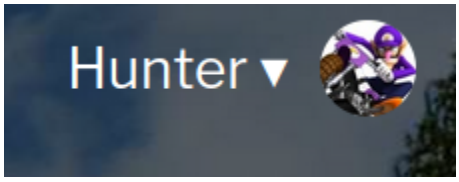
~~There should be a register page that includes a username field, password field, verify password field and a submit button. Passwords should be obscured (use the correct input field type) and while any field is blank, the submit button should not work. If a username already exists in the system, a user should not be able to create a new user. Please note that we will be using cookies to track user information and those should be set here.~~

~~**Logged Out Experience**~~

~~When a user first enters your website (even if they are not logged in), they should be able to see but not interact with all pages, including the games.~~

Navbar

At the top of every page, you should have a nav bar. This navbar should contain a link back to the home page and a link to a place where a user can login or register a new account. If the user is logged in, the navbar should still contain a link to the home page, a button to log out and a slightly stylized reference to their username. The following example is from Redfin.com and shows how the login/register buttons switch to the username when the user is logged in.

	
Logged out	Logged in (clicking will take me to my settings page)

Working Github and Deployed App Link

For this assignment, I recommend you use Render to host your code, but you are welcome to use any web hosting service you are comfortable with. Please follow the instructions from the lectures or contact any of the teaching staff to get this set up if you need help. Please be sure to add the TA's as collaborators to your Github repos.

Good Styling

As always, you should have a unique and consistent style across the different pages. There are no specific styling requirements, but make sure that your website looks good on mobile as well as desktop. You are welcome to use any 3rd party styling libraries, such as Tailwind, React Bootstrap, Material UI, etc. Ensure that this is something you would be proud to show an employer.

The expectation here is that we should see minimal “default” browser styling. Use different colors, fonts, sizes, backgrounds to make a website you would be comfortable showing off.

Well Written JavaScript

Now that we’re writing logic, you must start considering the quality of the code you’re writing. Functions should be simple, easy to read and avoid repetition. Make sure to make helper functions to simplify code in the backend and ensure that your React components are as simple as possible. We are not expecting you to use any “advanced” JavaScript functionality, but you should be writing code that you would be happy to show to a potential employer.
You are welcome to use or not use any library of your choice (this means that you are not expected to use Redux, for instance.)

RESTful APIs

Since this is a full stack app, you are expected to write backend Express APIs using each of the proper RESTful verbs that we learn in this course: POST, PUT, DELETE and GET. It is important that your code respects the promise made with these verbs so that there are no unexpected side effects.

Please note that it is up to you to determine how to store this data in memory. We will be looking for the following APIs, at least:

- ~~User APIs~~
  - ~~/api/user/isLoggedIn (GET)~~
    - ~~Checks cookie on user and returns username or error~~
  - ~~/api/user/login (POST)~~
    - ~~Request body contains password and username~~
    - ~~If successful, sets cookie~~
  - ~~/api/user/register (POST)~~
    - ~~Request body contains password and username~~
    - ~~If successful, creates users and sets cookie~~
  - ~~/api/logout (POST)~~

            - ~~Clears out cookies associated with logged in users~~
- Sudoku APIs
    - /api/sudoku (GET)
        - Returns list of all Sudoku games (including their name, ID, creation date and difficulty)
    - /api/sudoku (POST)
        - Request body should contain EASY or NORMAL as param
        - Returns game ID
        - A game should have a UNIQUE NAME. To do so, create a list of 1000+ words and randomly choose 3. (So a name may be something like Coconut Red House).
    - /api/sudoku/{game ID} (DELETE)
        - Will delete the game with the given ID
        - Not used on website for base project but will be checked for grading
    - /api/sudoku/{game ID} (PUT)
        - Will update the game with the given ID
        - Not used on website for project but will be checked for grading
- Highscores
    - /api/highscore (GET)
        - Returns list of sorted high score list for games ~~for users (list of username and game)~~
    - /api/highscore (POST)
        - Update the high score for a specific game
    - /api/highscore/:gameId (GET)
        - Return high score for specific game

You may use additional APIs as needed.

MongoDB, Mongoose and Security Implementation

You should correctly connect to MongoDB and Mongoose with your app. You should have at least 2 collections (i.e., MongoDB tables).
Additionally, your data should be secure in the ways we show in class.

Bonus Points

These tasks are OPTIONAL but will be good experiences if you're interested in exploring further into some harder ideas in web development, but remember that the teaching staff will not help you with this.

Submit Early - 3pts

Submit this assignment 48 hours before the submission time to receive extra points.

Password Encryption - 2pts

Ensure that your user passwords are encrypted in the database to add an extra level of security.

Delete Game - 5pts

When a user who has created a game goes to the game page, there should be a button labeled “DELETE.” This will delete the game from the system. Please note that this should update the high scores (every user that completed that game will now have one less win.)

Custom Games - 10pts

The ultimate challenge. This requires you to have completed the backtracking challenge in project 2 (or implement it now). On the game select page, add a new button labeled “Create Custom Game”. This will bring users to a new page “/custom” that shows a normal Sudoku board (9X9) but with all spaces empty. Users will be able to insert values here to create a new Sudoku challenge.

A button at the bottom of the page should be “Submit”. This will verify, on the backend, if the game is uniquely solvable (has one and only one valid solution). If it is accepted, the game is created and the user is redirected to the /game page for that game.

Writeup

With your submission, you must include a writeup that touches on the following points. You may discuss any other ideas that you deem salient to this work:

- What were some challenges you faced while making this app?
- Given more time, what additional features, functional or design changes would you make
- What assumptions did you make while working on this assignment?
- How long did this assignment take to complete?
- What bonus points did you accomplish? Please link to code where relevant and add any required details.

Deliverables

All members of a group must submit their projects individually.

Include all the following in your submission on Canvas:

1. A link to your Github repo. If you are working with a partner, you may submit a link to the same repo (for grading purposes, the TA's will likely only look at a single repo so make sure they are identical.) Please note that your Github repo should be named: {firstname}-{lastname}-project3, and if you're working with other people, all names should appear on the repo.
2. A video (uploaded or hosted anywhere you wish) of you or your collaborator walking through the project, talking about each page, anything you're particularly proud of, and playing a few rounds of the game. There is no time limit to this video but should typically be less than 5 minutes.
3. A link to your deployed app. As above, if you are working with one or more collaborators, please submit the same link.
4. Your writeup. If you are working with one or more collaborators, you may each write this together or individually, but please indicate this with your submission.
5. The name of your collaborator(s), if any

Academic Integrity

As always, all assignments are expected to uphold and follow NEU policies regarding academic integrity. Any students that violate these policies will be reported to OSCCR immediately.