SODV1201

Final Group Project: Full Stack Web App

For the project, you are required to complete a full stack web application that includes both frontend and backend. You can use "Shared Work Space" or any other project (TODOAPP is not a valid project). The detailed description of Shared Worked Space project is available on next page. You are free to work on your own project idea. However, You should approve your project idea with me by Aug 1st, 2024. Your project should have the following minimum requirements.

1. **Part 1:**
2. You should first write user stories of your project. See examples of Shared WorkSpace project for writing user stories. Approve this from me by Aug 6th.
3. You should stor information in JS Object or arrays. You can use JSON file too once we study JSON file in coming lectures.
4. You application should include following four operations: Create, Read, Update, Delete and Search
5. **Create:** It will store data in the JS object or Array
6. **Read:** It will read data from the JS object or array and display on the webpage
7. **Delete:** It will delete the record
8. **Update:** It will update the record
9. **Seach:** It will search the data from the JS file/JS object and display the result on the webpage

10. **Part II**
11. You should implement API to fetch data from the file or database server.
12. Your server implementation should use Node.js and Express. If you would like to use a different framework, you need a permission from the instructor in advance.
13. Use the correct HTTP method (GET, POST, PUT, or DELETE).
14. For each web service/functionality in your API, return a flag that indicates the success or failure of the web request (error handling). In case of any failure, add a detailed error message.
15. Your server should be able to run locally and interact with a local file or a database that is hosted on your local MS SQL Server or on the cloud.

Project Overview

This project includes a number of different works. Make sure you form a group from the day it is assigned to you and start working on it. In this programming project, you are required to create a set of JavaScript functions and objects, then demonstrate that they operate correctly. You can use any editor to write the functions.

Coworking is a work style where different individuals or teams share a working environment. If you are a self-employed or a working-from-home professional, you would love to have the option to rent a meeting room, or a desk in an office equipped with high-speed internet, printers, copiers, stationaries, and office kitchen. As there is an increase in the number of coworkers and the number of sharable workspaces, we are looking to develop an application to connect both parties. The objective is very similar to how Uber connects riders with drivers and Airbnb connects tenants with landlords.

Instructions

You will work on this project as a group to develop this application. For this project, you must use the frameworks we cover in class. You will receive incremental grades for this project with the following two milestones:

Phase 1: Design and implement a standard web application for the given shared workspace project. The web application wireframe should have landing page, header and footer. For more functionalities refer to the user stories to figure out what pages do you need and what controls should each page has. Whenever you want to store any information, you can use array or Js object at this level. When you start working on phase two you will be working with .JSON file or a Database of your choice.

From your web application a user can do the following activities (Given user stories)

16. As a user, I can sign up for an account and provide my information (name, phone, and email). I can choose my role as an owner or a coworker.
17. As an owner, I can list a property with its address, neighborhood, square feet, whether it has a parking garage, and whether it is reachable by public transportation.
18. As an owner, I can select one of my properties and list workspaces for rent. Workspaces could be meeting rooms, private office rooms, or desks in an open work area. For each workspace, I can specify how many individuals it can seat, whether smoking is allowed or not, availability date, lease term (day, week, or month), and price.
19. As an owner, I can modify the data for any of my properties or any of my workspaces.
20. As an owner, I can delist any of my properties or any of my workspaces.
21. As a coworker, I can search for workspaces by address, neighborhood, square feet, with/without parking, with/without public transportation, number of individuals it can seat, with/without smoking, availability date, lease term, or price.
22. As a coworker, I can select a workspace and view its details.
23. As a coworker, I can get the contact information of a workspace's owner.

The following is a guideline from the user stories above:

User Story 1: Add a web page with controls to take the user information from keyboard and an Add button. On click on the Add button, make a call to the web service that adds users to the user object or array.

User Stories 2 and 3: similarly add web pages for the owner user to specify the property and workspace information and click a button to call the corresponding web service/ functionalities that adds them to the property/workspace object or array dataset.

User Stories 4 to 8:
Add a web page with controls to search and display the properties and their workspaces. Add a login page. If the login is successful, based on the user's role, modify the previous page as follows:

24. For the owner user, filter the page to show only the owner's properties and workspaces. Also, show controls that the user can use to sort, update and delete any of the items.
25. For the coworker user, show controls for the user to search and to sort properties using different criteria.

All pages should make requests to your locally defined array object or JS Object application. On phase two it should interact with the DB.

Phase 2: At this phase of your project, you need to start using the actual file system or a database server of your choice to store information:

26. Design and implement an API (backend application) to fetch data from a shared workspace file which can be local file or a database storage.
27. Your server implementation should use Node.js and Express. If you would like to use a different framework, you need a permission from the instructor in advance.
28. Use the correct HTTP method (GET, POST, PUT, or DELETE).
29. For each web service/functionality in your API, return a flag that indicates the success or failure of the web request (error handling). In case of any failure, add a detailed error message.
30. Your server should be able to run locally and interact with a local file or a database that is hosted on your local MS SQL Server or on the cloud.

Minimum Viable Product (MVP)

The MVP is the smallest conceivable list of features that fulfill the primary business goal of the software product. One way to summarize a feature is a user story-- a short sentence describing the feature from the perspective of the user. User stories frequently take the form: "As a <type of user>, I can <take some action> so I can <some reason>." You can start with the following user stories as a backlog for your MVP.

Bonus Features

You will receive a full mark if the features of your MVP are fully working. You learn a new concept with every feature you add. Adding the following features is highly recommended:

31. As an owner, I can add photos to any of my properties or workspaces.
32. As a coworker, I can rate a workspace on a one-to-five-scale. The workspace's listing should include the average rating.
33. As an owner, I can rate a coworker on a one-to-five-scale. The coworker's profile should include the average rating.
34. As a coworker, I can sort the workspaces' search-results by the average ratings, availability date, or price.
35. As a coworker, I can write a review about a workspace. The workspace's listing should include these reviews.

Submission Directions:

36. When you finish, compress all the files into one and upload to D2L Dropbox named **Final Group Project.**
37. Also add a readMe file, that explain how to start/run your application.
38. This is a coding assignment, so all the rules about best coding practice apply. Your code is evaluated for **correctness** (does it achieve the task it is supposed to) and for **hygiene** (is it clear, well-commented, and easy to follow). There is no point in writing accurate code that nobody else can understand.
39. Some tips to ensure good code hygiene: Add documentation about you on the very top of your code. Something like

```
/**
 * @name: Assignement1
 * @Course Code: SODV1201
 * @class: Software Development Diploma program.
 * @author: Your Full Name.
 */
```

40. Add intelligent comments that explain your logic for each code
41. Add intelligent comments before expressions, methods, functions and classes to outline what they do
42. Use sensible variable names that match the purpose of a variable
43. Use whitespace between functions and code blocks and indent consistently

Marking Criteria

Grading Items	**Points**
• Submission method & format • Code is readable and well organized with adequate whitespace • Author identification, course code, assignment title, date in web page • Code has detailed inline explanation/comment	10
• Interactive, responsive and styled front-end application which serve your backend server and API's using any of the following technology html, CSS, jQuery, js , .ejs ,bootstrap. Communicate with your instructor if you choose another - Working form - Proper form validation and warning for users	40
Proper incoming data completeness and correctness validation (partly it can be done on the front end) Response/Confirmation after form submission Clean code and detailed inline explanation/comments Proper naming, module, package and API usage	20
Working API to fetch the dataset based on CRUD principle	30
Total	100 points