SFWRTECH 4WP3 – Advanced Web Programming – Project #1

Overview

This project involves designing, creating and presenting a web application. Both project milestones must receive a satisfactory grade for the project to receive a satisfactory grade, otherwise a grade of unsatisfactory will be awarded.

The flexibility of the requirements is intended as an opportunity to build your portfolio while creating something you find fun! Portfolios are important for job seeking... typically made up of a collection of projects hosted on GitHub, a portfolio demonstrates your practical skills to the world. The instructor also references projects when writing graduate school reference letters.

Git Repository

Before conducting any implementation work on your application for milestone #2, create a local git repository. As you implement the application and its features, make very frequent commits of the application code to the repository, i.e. at least 15 or preferably more commits total. Make multiple commits as you implement each feature. When you encounter any errors or logic bugs during implementation, make a commit to the repository before you have even fixed the bug to document the fact that you've encountered this issue (and of course, make a commit afterwards once you have fixed it). The commits will be overviewed during your milestone #2 presentation.

Overall Requirements

The web application developed must conform to the following minimum overall requirements:

- The MVC pattern as defined in the Week 5 MVC slides must be used.
    - i.e. controller, model and view files, actions, all conforming to those slides
- Node.js and Express must be used to handle requests.
- An SQLite database must be used to store records of data in at least one table.
    - The table must have at least 6 columns of data, including a column that is used as a primary key (the built-in rowid column could be used).
    - At least 3 different types of columns should be used, e.g. integer, real, text.
- Mustache must be used to generate a dynamic page (i.e. the view file).
    - At least one collection of records should be displayed on the page.
        - The word “collection” here is intentionally open-ended. The collection could be a table, but it could also be a “list” or 2D grid of items.
    - At least one template variable should be used.
    - At least one template loop should be used (e.g. to display the records in a table).
    - At least one template switch should be used.
    - At least one web form should be present as part of facilitating some action.

- At least 8 actions as defined in the Week 5 MVC slides must be present:
    - Examples of actions include...
        - Creating records, e.g. inserting a new record from form inputs
        - Reading records, e.g. all records, filtered records, sorted/ordered records
        - Updating records, e.g. liking, modifying multiple fields with a form
        - Deleting records, e.g. delete some records, delete all records
    - At least one action should update a record in the SQLite database.
    - At least one action should delete a record in the SQLite database.
    - At least one action should create a record in the SQLite database.
- Form validation must occur upon a form being submitted.
    - The validity of two inputs must be verified when a form is submitted.
        - This validity must not simply be that the inputs are not blank/empty.
        - i.e. something like checking that a value in within an allowed range or of the correct type (a number, a string, etc.) must be done.
    - This validation must occur on the backend, i.e. with Node.js and Express.
    - The results of the validation must be displayed to the user on the front-end.
        - i.e. if an input is invalid, display which input is invalid and how it is invalid
- The website should be responsive.
    - If the browser window width were to change from being above 768px to below 768px, at least 3 noticeable changes should explicitly be programmed to occur.
- The website should be neatly styled.
    - Tables, lists and forms should be styled.
    - Sections of content should be styled, e.g. borders, padding, margins, etc.
    - Bootstrap or another front-end framework can be used for this if you prefer. You can also use styles that you find online if you cite the source in the comments.
- At least one image should be displayed on the web application.

All the requirements must be satisfied by your own original work unless otherwise cited above. Generative AI such as ChatGPT must not be used for the project. The project that is submitted should not be a slight re-working of the MVC example in-class or any other examples/work.

Project Milestone #1 - Proposal

Initial Due Date: Tuesday February 25th at 11:59pm
Weight: 3% of course grade
Requirements:

Create a proposal for your application which must include the following:

- What is the purpose of the application? e.g. to track due dates...
- Who would be a potential user for the application? e.g. students, teachers
- What 8 actions will the application allow the user to perform?
- What explicit responsive changes will occur to the website at 768px?
- What columns will the SQLite table have, and what are their types?

    - If you have more than one table, provide this information for all tables
- What form validation will occur? i.e. what inputs will be checked for what properties?

Create a single UI prototype of your application, i.e. a sketch, wireframe or mockup illustrating what you intend your application to look like. At least a coupe important aspects of the application should be illustrated in the prototype... i.e. how the collection of data will be presented, where a form appears, how are some actions performed, etc. It could be hand-drawn or created with software like https://app.diagrams.net/ (see the Wireframe options). The finished application does not need to look *identical* to the UI prototype, but it should look similar.

Your proposal should be no more than 2 pages in length.

Submission:

Upload the document (PDF format) to the dropbox on Avenue.

Marking rubric:

Component	**Description**
Purpose, users	Application purpose and users
Actions	What actions will the application perform?
Responsive changes	What responsive changes will occur?
Database table(s)	Database table columns and types
Form validation	What form validation will occur?
UI prototype	Application layout and appearance

Each component will be graded as either unsatisfactory or satisfactory.

Grade	**Definition**
Unsatisfactory	The answer is fundamentally incorrect, incomplete, and/or contains at least one critically important mistake, or the answer does not demonstrate a reasonable understanding of the material.
Satisfactory	The answer is overall correct and largely complete, it may contain multiple minor mistakes or even a major mistake, and the answer does demonstrate a reasonable understanding of the material.

To achieve a grade of satisfactory on the milestone overall, 5 out of 6 components must receive a grade of satisfactory.

Otherwise, a failure grade of unsatisfactory will be assigned.

Please keep in mind you will be provided with feedback to make you aware of which components have what issues, if any, and you will have the ability to re-submit the milestone.

Project Milestone #2 – Implementation and Reflection Video

Initial Due Date: Tuesday March 11th at 11:59pm
Weight: 22% of course grade
Requirements:

Submit the implementation of your web application including:

- The git repository containing all source code files, commits and log messages
    - A package.json file should be present in the git repository
        - Running npm install should install any required packages for the app
        - Include an instructions.txt file with the command to run your application
            - e.g. npm start or node server.js, etc.
- The SQLite database file that the application uses should be present
    - The database table should have some records in it already

Submit a reflection video no longer than 12 minutes in length in which you do the following:

- Quickly demonstrate the actions of your application as it runs in the web browser
    - i.e. create a record, sort a column, etc.
- Next as part of the same video use the Timeline view or Source Control Graph view of VS Code to reflect on the changes you made to the code in the video. Discuss what you have learned, including at least two of the following:
    - Go over a bug that you encountered. Show the relevant version of your code. Discuss what the bug was and why it was a bug. Discuss how you fixed the bug and show the code that fixed it.
    - Go over something you learned OR had to figure out through experimentation while creating the application. Show the relevant version(s) of your code. By "experiment" we mean "trying different things to make something work or behave the way you intended". What did you have to spend some time figuring out? And/or what did you learn? Did you use any external sources, if so, how?
    - Go over a commit or multiple commits that involve implementing a feature. What did you achieve in the commit(s)? Describe and explain how you implemented the feature making reference to the commit(s).

Submission:

Upload the git repository and database in a file named source.zip. You can either upload your video as an mp4 file called video .mp4, or you can include a URL for an unlisted YouTube video

in a plaintext file called video.txt (the video cannot be set to 'private' or the marker cannot see it, it must be 'unlisted' so the marker can see it using the link provided).

Marking rubric:

Component Number	Component	Description
1	MVC	Is the MVC pattern used?
2	SQLite	SQLite database used according to overall requirements?
3	Mustache	Is mustache used according to the overall requirements?
4	Express	Express used to handle requests and responses?
5	Validation	Is form validation present according to overall requirements?
6	Responsive	Responsive design changes
7	Styles	Application styled nicely according to overall requirements
8	Actions	Are 8 actions present, conforming to overall requirements?
9	Git	Does git repository contain many commits with log messages?
10	Video	Demo + reflection on changes using git repository

Each component will be graded as either unsatisfactory or satisfactory.

Grade	Definition
Unsatisfactory	The answer is fundamentally incorrect, incomplete, and/or contains at least one critically important mistake, or the answer does not demonstrate a reasonable understanding of the material.
Satisfactory	The answer is overall correct and largely complete, it may contain multiple minor mistakes or even a major mistake, and the answer does demonstrate a reasonable understanding of the material.

To achieve a grade of satisfactory on the milestone overall, 5 out of 7 components 1-7 must receive a grade of satisfactory, and components 8, 9, and 10 must receive a grade of satisfactory.

Otherwise, a failure grade of unsatisfactory will be assigned.

Please keep in mind you will be provided with feedback to make you aware of which components have what issues, if any, and you will have the ability to re-submit the milestone.