CptS 489—Web Development Spring 2020

**Individual Assignment #3: More Advanced App to Support Editing and Deleting Data Records**

*Last modified 30 January 2020*

|  |  |
| --- | --- |
| Worth: | 5% of overall course grade (plus extra credit opportunities) |
| Assigned: | 30 Jan 2020 |
| Due Dates: | * GitHub repository: 6 Feb 2020 at 9:10 a.m. * OSBLE post-mortem reflection post: 6 Feb 2020 at 11:59 p.m. * OSBLE post-mortem response post: 7 Feb 2020 at 11:59 p.m. |
| Late Policy: | * 10% deduction if submitted up to 24 hours late * Assignments more than 24 hours late will *not* be accepted * Late post-mortem posts and responses will *not* be accepted |
| GitHub invite: | <https://classroom.github.com/a/d9_P2F01> |

# Overview

In this assignment, you will expand single page web app you created in IA#2 by adding the ability to view, edit and delete data records. As the user adds data records, they will appear in a data table. Clicking on the “view/edit…” button in a given row of the data table takes the user to the data entry page, with the fields pre-filled with the corresponding data. Here, the user can view the full data record (if it is impractical for all data to be displayed in the data table), and update values as desired. Similarly, clicking on the “delete” button in a given row of the data table takes the user to a modal dialog box that asks them to confirm the deletion. If confirmed, the data record is deleted, removed from local storage and removed from the data table.

As before, your app will be required to have the same basic design we’ve been using throughout this course, with a fixed top bar, along with a fixed bottom bars and a dynamic side menu. You are required to use CSS, BootStrap and FontAwesome to style your app and enable it to gracefully scale to any device size.

# Learning Objectives

Through completing this individual assignment, you will learn and/or reinforce your ability to

* use HTML, CSS and JavaScript to get data from a user.
* use HTML, CSS and JavaScript to display data entered by the user.
* use HTML, CSS and JavaScript to allow the user to view, edit and delete data.
* set up and use a GitHub project (a.k.a. Kanban) board to track task completion and identify issues and questions that arise as you work.
* use the GitHub to commit code to a local Git repository, and push updates to a remote Git repository on GitHub.com.
* use an IDE of your choice ([Visual Studio Code](https://code.visualstudio.com/) recommended) to write and deploy web programming projects.
* properly submit an assignment by (a) creating a new *branch* for your assignment work, (b) publishing your web app to GitHub Pages, and (c) issuing a *pull request* when your assignment is ready to be graded.

# Requirements

You will expand on the app created in IA#2 by supporting the ability to view, edit and delete the multi-field records that can be entered in your IA#2 app. As such, the requirements below designate which requirements are carryovers from IA#2 and which requirements are new.

### Visual Design

1. [IA#2] Your website has a login screen with username and password fields and a login button. See requirements for IA#2 for more details.
2. [IA#2] Your website has a fixed top bar, fixed bottom bar, and side (hamburger) menu.
3. [IA#2] The bottom bar has two buttons that toggle between modes where the user may (a) add data and (b) view/edit data.
4. [IA#2] The side menu has items for Adding a Data Record, About and Log Out.
5. [IA#2] When an app page is displayed, the fixed top bar displays an appropriate label for the screen (e.g., “Welcome to MyApp” for the login page, “My Data” for the Data Table page).
6. [IA#2] The app has a Data Table page that displays data records. A floating button (e.g., ) appears in the lower right-hand corner of this page.
7. [IA#2] The app has an Add Data Record page that presents a form that the user can fill in to enter a new data record.
8. The app has a View/Edit Data Record page that presents the form in (V7) pre-populated with the data of the record to be viewed/edited.



### Functionality

1. [IA#2] **Login Screen**. When the app starts, it displays a login screen with a username (email address) field, a password (obscure text) field, and a “Log In” button.
   1. The cursor is initially be focused on the username field.
   2. When the user clicks the “Log In” button, the app ensures that the username and password are valid (use template regular expressions).
      1. If they are not, the app displays and error message and requires the user to re-enter the username.
      2. If they are, the “Log In” button displays a spinner for one second and then the app switches to the “Data Table” screen.
2. [IA#2] **About Box**. When the user chooses “About,” a *modal* dialog box is presented with info on the app. The user can dismiss the dialog box by clicking an “x” in the top-right of the box and/or by clicking an “OK” button in the bottom-right of the box.
3.  **Data Table page**. Each data record must be displayed as a row of a table, with columns for each data item. (Note: If your data records contain many items, then you can present a summary of each record in approximately 3-4 columns). Two additional buttons are available in the last two columns of each row: *edit data* and *delete data*. Here is an example:



* 1. When the “Delete” button is clicked, a modal dialog box must appear asking the user to confirm the deletion. If the user confirms the deletion, then the row must be removed from the table and also removed from localStorage.
  2. When the “View/Edit” button is clicked, the user must be taken to the Add/Edit Data Record page (see next item), where they can edit the data and click a button to update the data.
  3. A floating button (e.g., ) appears in the lower right-hand corner of the screen. When clicked, the user is taken to the Add/Edit Data Record page.

*Note: In your IA#3 app, you must remove the alert box in your IA#2 app that displayed all app data stored in localStorage each time a data record was saved. That alert box was intended only for debugging purposes and is no longer needed in IA#3.*

1. **Add/Edit Data Record page**. As in IA#2, this page displays some combination of input fields, buttons, sliders or other input elements to enable the user to enter data records. When the user is adding a new record, the submit button is appropriately labeled “Save Data.” When the user is editing an existing record, the form is pre-populated with the existing data and the submit button is appropriately labeled “Update Data.”.



* 1. The bottom bar buttons are disabled when the Add/Edit Data record page is open.
  2. The side menu button displays a left arrow (ß), which the user can click on to exit the page without saving.
  3. Reasonable measures are be taken to ensure only valid data are entered. Use checkboxes, dropdown menus and other input elements to constrain input choices as appropriate. Use form patterns to ensure textual data are valid and to report invalid data.
  4. When the user clicks on the button to add or update data, the app writes the new or updated data to localStorage and provides feedback that the data has been saved by switching to the Data Table page, which now has a row containing the newly-entered or newly-updated data.

1. **Extra Credit (up to 5%)**: **User account management.** Add the following features to your app to support user account management:



* 1. *Create user account* (worth 2% extra credit). Create a page that allows the user to create a user account with username, password, password hint, challenge question and challenge answer. Display the password in obfuscated text and require the user to enter it twice. On login, validate both username and password. Store username and password in localStorage. (This is totally insecure, but a good exercise to begin thinking about user management, which we will learn how to do securely later in the course.)
  2. *Password hint (1% extra credit, must implement (a) also)*. On the login page, provide a “Password Hint” link that brings up a modal dialog box that asks the challenge question. If the user provides the correct answer, then the modal dialog box displays the password hint. Dismissing the dialog takes the user back to the login screen.
  3. *Password reset* (*2% extra credit; must implement (a) and (b) also*). On the login page, provide a “Reset Password” link that brings up a modal dialog box that asks the challenge question. If the user provides the correct answer, then the page allows the user to reset their password. As before, display the pasword as obfuscated text and require the user to enter it twice. Update the password in localStorage. Dismissing the dialog takes the user back to the login screen.

### Code Structure and Contents

1. Your web app uses the same single page app template used in IA#2. The template includes
   1. a fixed top bar with app icon and page title
   2. a fixed bottom bar with at least two buttons that can be used to navigate to the data table page and the Add Data Record Screen.
   3. A hamburger (side) menu.
   4. A floating button on the Data Table page.
2. Your project has three subdirectories, as in the starter code: html, styles, and scripts. Only index.html is at the top level, outside of these folders.
3. As appropriate write HTML snippets in separate files (in the html directory) and include them into index.html using the w3-include-html tag, as illustrated in class.
4. All scripts that handle user interaction and page navigation reside in scripts/eventHandlers.js. These must include a header documentation block that describes what they do.
5. All scripts that should execute at startup reside in scripts/startup.js.
6. All user data must be stored in localStorage, with data records associated with the specific user who created/edited them. Thus, user data must persist locally across app sessions.

### Programming Process

1. In your GitHub repository, you create a project (“Kanban”) board with at least three columns: “To Do,” “Doing,” and “Done.”
2. You divide your project into bite-size tasks and create a card or issue for each task. Cards contain checkboxes as appropriate to denote subtasks.
3. You move the cards between columns and add documentation to your cards to indicate the issues you encounter and your progress.
4. You commit code regularly throughout the assignment period. This means that you start the project early and commit at least five times over three days. Avoid making all or most of your commits on the same day right before the deadline!
5. You use descriptive commit messages to clearly describe the code changes you are committing.
6. You post questions and issues that arise to OSBLE activity feed. You attempt to respond to questions/issues on activity feed

# Assignment Submission

Here’s how to work on and submit this assignment using GitHub and OSBLE:

1. Use your IA#2 repository as the starting point for this code. Clone it over to your IA#3 repository before starting on IA#3.
2. When you start working on your new repo, fork a new branch gh-pages and work in that branch.
3. When you are done with the assignment, issue a *pull request* to indicate that you are done with the assignment and it is ready for review. Your pull request must be issued before the assignment deadline.
4. Publish your gh-pages branch to GitHub Pages. This will allow us to execute your completed assignment in a web browser. Here’s how:
   1. From your repo on GitHub, choose “Settings.”
   2. Scroll down to the “GitHub Pages” section.
   3. Under “Source,” choose the gh-pages branch.
   4. Confirm that your web app is accessible at <https://Cpts489-WSU.github.io/>..., where … will depend on the assignment number and your repo name.
   5. Place the URL at which your web app is accessible in the README file in your repo. We will use that URL to access your assignment for grading purposes.
5. Submit the URLs of your GitHub repo and your app through the IA#5 assignment in OSBLE.

# Post-Mortem Reflection and Response Posts

After you complete a programming task, a best practice is to take a step back and reflect on how you did and what you learned. You are required to write a “post-mortem” reflection in the OSBLE reflection assignment associated with this assignment. Y**our reflection post must be at least 200 words and should make an earnest attempt to reflect on how you did, what you learned, and what you could do better next time**. The following prompts are intended to help get you started by thinking about what you did more deeply:

* Describe a struggle that you overcame when working on this assignment.
* Describe an issue with your assignment that you were unable to resolve.
* Assess your process: What did you do well and what could you improve on to be more effective?
* Provide advice to a future student on how he or she might succeed on this assignment.
* Describe the most fun aspect of the assignment.
* Describe the most challenging aspect of the assignment.
* Describe the most difficult aspect of the assignment to understand.
* Provide any suggestions for improving the assignment in the future.

Once you have posted your reflection, you are required to post a reply of at least **50 words** to the reflection of one other peer. *When selecting a post to reply to, you must choose a post no one else has replied to*. You may not reply to your own post!

At a minimum, your reply post should compare and contrast what you did with what the author of the post did. Here are some prompts to guide your reply to another student’s post:

* What did the student do or learn that was similar to or different from what you did or learned?
* What did the student struggle with that was similar to or different from what you struggled with?
* What can you learn from the student’s reflection?
* What advice do you have for the student?

# Assessment

We will grade your submission according to the rubric on the following page. The grading rubric for your post-mortem reflection posts and responses appears on the page after that. These rubrics also appear in the corresponding assignments in OSBLE, where we will complete the rubrics and provide feedback.

**Grading Rubric for Individual Assignments**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criterion | Weight | D-F Level: Emerging (0-6 pts) | B-C Level: Developing (7-8 pts) | A Level: Mastering (9-10 pts) |
| 1. Visual Design | 25 | Your app fails to address two or more “Visual Design” requirements adequately. | Your app fails to address one “Visual Design” requirement adequately. | Your app fully addresses all “Visual Design” requirements defined for this assignment. |
| 1. Functionality | 35 | Your app does not display when launched and/or fails to address two or more “Functionality” requirements adequately. | Your app displays when launched but fails to address one “Functionality” requirement adequately. | Your app displays when launched and correctly and fully implements all “Functionality” requirements defined for this assignment. |
| 1. Code | 25 | Your app fails to address one “Code Structure and Content” requirement adequately. | Your app fails to address one “Code Structure and Content” requirement adequately. | Your code base fully addresses all “Code Structure and Content” requirements defined for this assignment. |
| 1. Process | 15 | Your development process fails to address two or more “Process” requirements adequately. | Your development process fails to address one “Process” requirement adequately. | Your development process fully addresses all “Process” requirements defined for this assignment. |

**Grading Rubric for Post-Mortem Reflection Posts and Responses**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criterion | Weight | D-F Level: Emerging (0-6 pts) | B-C Level: Developing (7-8 pts) | A Level: Mastering (9-10 pts) |
| 1. Reflection Post | 80 | Post is on time, is at least 200 words and appears to be hastily written; in many places, it is off-topic, difficult to understand, is superficial and/or it has grammar or spelling errors that are distracting. (*Late posts or posts that are not at least 200 words receive a 0*.) | Post is on time, is at least 200 words and makes attempt to reflect on programming process; however, it may be difficult to understand or superficial in a few places and/or have clearly noticeable grammar or spelling errors. | * Post is on time and at least 200 words * Post makes an earnest attempt to reflect on programming process: what you did, what you learned, and what you can do differently next time. * Post is easy to read and follow; grammar and spelling are mostly correct. |
| 1. Reply to Reflection Post | 20 | Reply is on time, at least 50 words and appears to be hastily written; in many places, it is off-topic, difficult to understand, is superficial and/or has grammar or spelling errors that are distracting. (*Late posts or posts that are not at least 50 words receive a 0*.) | Reply is on time, at least 50 words and makes attempt to compare/contrast own experiences with those of person to whom replying; however, it may reply to a post. with other replies; be difficult to understand or superficial in a few places; and/or have clearly-noticeable grammar or spelling errors. | * Reply is on time and at least 50 words. * Reply makes an earnest attempt to compare/contrast own experiences with those of person to whom replying. * Post is easy to read and follow; grammar and spelling are mostly correct. |