# Assignment: Single Page Architecture – Part 2

In this assignment we add error trapping and an improved Bootstrap interface to our “to-do” list application.

Make note of the following URLs.

|  |  |
| --- | --- |
| **URL** | **Notes** |
| <http://ec2-3-15-38-21.us-east-2.compute.amazonaws.com/auth/>  | GET from this URL and send a “username” and a “password” as URL Encoded data. Each GET request results in either a JSON array of data or an error message in a string of text.  |
| <https://misdemo.temple.edu/todo/addtask/>  | POST to this URL to create a task in the to-do list. You must pass “task\_owner” and “task” as URL encoded data. This API feature returns confirmation data in a JSON array.**IMPORTANT:** In this specific assignment is important that you put the "/" at the end of the URL as shown here. |
| <https://misdemo.temple.edu/todo/tasks/>  | GET from this URL, sending “task\_owner” with each request. The result is a JSON array of tasks data.**IMPORTANT:** In this specific assignment is important that you put the "/" at the end of the URL as shown here. |

## Instructions

1. Start with unit6\_10\_spa2.zip. Set up your project in VS code.
2. Explore the readme folder. This folder contains same usernames and password that work with the “auth” API in the above table.
3. As you have done in an earlier assignment, find the hidden input tag with the value “Eustace”. Change the name “Eustace” to be your name, or any other unique value of your own choosing.
4. In the <script> portion of the page, find this line:

|  |
| --- |
| // this command runs when the page is ready.... $("#div-sample").show(); |

This line of code controls which of the 8 span columns gets displayed first. Try changing it to div-list, div-add, and/or div-login. Reload the page after each save and observe the change.

You should also note that each 8-span column is hidden by default with the in-line CSS statement “display:none;” That means that the only way these will appear after page load is if we me use JavaScript / jQuery to show them!

1. Find the click event for btn\_login. A few lines of code have already been provided there. These lines of code wipe out any content in the login\_message div, remove any CSS classes applied to the login\_message div, and create variables that will hold user input.

Add logic to the click event handler so that if username or password is blank, an error message gets put in to login\_message, and login\_message gets the CSS classes “alert” and “alert-danger”.

Otherwise, call the supporting function named “login\_controller”.

1. In the login controller, add logic so that if the data returned by the API call is of the type string, then the string is put in to the login\_message div. Again, give the login\_message div the CSS classes “alert” and “alert-danger”.

Otherwise, hide div-login and show the div tag with the id “div-list”.

## On your own

1. Now, in the click handler for btn\_add\_task… can you do the same thing?
	1. Check to make sure that the user has added text to the input tag “task”
	2. If they have not added text, put an error message into the confirmation div.
	3. If they have added text, call the supporting function named “add\_task”.
2. You may have notice that the logout link does not work. Add a click event handler that will show and hide div tags appropriately when the logout link is clicked.

## Optional

Deep thought: Wouldn’t it be good if the tasks were associated with the user who logged in successfully? (So, if I am logged in as “Joe” I see “Joe’s” tasks, and if I log in as “Jessica” I see “Jessica’s” tasks.)

A simple approach would be to manipulate the hidden task owner tag. Add logic to your application to support this.

## How will this assignment be graded?

|  |  |
| --- | --- |
| Item | Point Value |
| All files / folders uploaded correctly | 10 |
| Can I log in? | 50 |
| Can I add a task? | 20 |
| Can I log out? | 20 |