System and Unit Test Report

SlugSense WebApp

WebSwag

7/23/17

System Test Scenarios

A. User story 1 from sprint 1: As a user I would like to be able to clearly read a chart with useful data presented without confusion or obstruction.
B. User story 2 from sprint 1: As a user, I would like to have options for looking at data from days and weeks.
C. User story 3 from sprint 1: As a user, I would like to easily distinguish different data graph such as sunlight, humidity, temperature, and moisture.
D. User story 1 from sprint 2: As a user, I want a search menu so that I can search information for my device.
E. User story 1 from sprint 3: As a user, I want to be welcomed by nice and well designed and functional login page.
F. User story 2 from sprint 3: As a user, I would like the web application to be simple, clear and uniform in regards to design and visualizing data.
G. User story 3 from sprint 3: As a user, I want to see a functional website so that I can count on the product's reliability.
H. User story 4 from sprint 3: As a user, I want alerts shown on the dashboard so I will have a clear idea of the data in my device if it goes wrong.

Scenario:

➔ Start SlugSense; check the design and simplicity of the login-page.

- ◆ Check the vivid display and clear lines.
- ◆ Check if login box can be found easily.
- ◆ See if it is eye-catching.
- ◆ See if the login page is responsive

➔ Login into the web-page.

- ◆ ID: sustainability
- ◆ Password: sustainability
- ◆ Press Login
- ◆ User should access into the first front page of the web with graph and numbers displayed already.

➔ Hover the mouse on the graph to see information about data.

- ◆ Hover the mouse over the graph
- ◆ Displays a box containing the data of the data point being hovered over.
- ◆ Move the mouse off the graph.
- ◆ The box display on the graph disappears.

➔ Click the buttons day and week on top of the graph to see different data.

- ◆ Click day

- Displays a day of data; X-axis label and unit of measurement is now hours
- Click week after clicking days
- Displays data in weeks; X-axis label and unit of measurement is now days

➔ Click buttons for temperature, humidity, moisture, and sunlight on the top right side of the graph to see different data.

- Click temperature
- Data for Y-axis changes to temperature
- Click moisture
- Data for Y-axis changes to moisture
- Click humidity
- Data for Y-axis changes to humidity
- Click sunlight
- Data for Y-axis changes to sunlight
- Buttons for unselected data also get blurred and dimmed in order to communicate that they are not being displayed.

➔ Try to search a keyword in the search bar to look for specific device that represents the keyword or key number.

- Click the search bar and type in the number or keyword.
- Type number 50 to see any device or data that is related to 50.
- Search result contained device with ID 50.
- Type number 51 to see the result.

- Got result that is a device with ID 51. Also got result from multiple devices that have value 51 in their data within one of the following categories: temperature, humidity, moisture, or sunlight.
- Search 0 to see what happens; got zero results from it meaning it is working well.

➔ Check the alert; click the dashboard and see the notification.

- Click the flag on the top right side of the web-page.
- Navigates to the new page that displays a list of alerts.
- Click on each alert to see if the number of notifications goes down.
- Initial notification=2; every click, the notification goes down by one.
- Click on one notification, total notification alert is 1.
- Click all of the notification, the alert number displays on the flag disappears.

➔ Click one of the boxes below the graph to see status and alerts from last 24 hours for all devices

- Select one device; click an alert category at the bottom of the page.
- Displays a list of all alerts for that data category that happened within the last 24 hours
- The text that shows on the box is clear and easy to understand
- Click another box that is different; click the box.
- Displays a list of all alerts within the last 24 hours for the selected data at the selected node again.
- The text is clear and easy to understand.

- ◆ Click the box again to make the list of alerts disappear.

➔ Click the button on the left menu bar to see different data from each different device

- ◆ Click one of the devices; check the graph to see if it comes out well.
- ◆ Click another device; check if the graph and sensor box data changes to the device that I clicked.
- ◆ It displays each device's data correctly
- ◆ Check functionality of day and week buttons.
- ◆ Hovering mouse over graph works as well.

Graph:

Function: Display data on the graph upon loading; clicking day button; clicking week button.

Function: Display data on the graph upon clicking each category.

Equivalence Class	Input	Expected Output	Tested by
$EC_{negative\ data}$	-3	Graph will not display this data point.	T.W.
$EC_{temperature\ data}$	10	Y-axis' label unit should be ℃. Both x and y axis data range should change accordingly.	T.W.
$EC_{humidity/moisture/sunlight}$	20	Y-axis' label unit should	T.W.

data		be in %, both x and y axis data ranges should change accordingly	
$EC_{\text{user changes webpage size}}$	Web page size changes	Graph should change size accordingly	T.W.
$EC_{\text{user clicks different nodes}}$	Different node selected	Graph should change to selected node's day data	T.W.
$EC_{\text{user load webpage}}$	Load web page	Entire graph should show without user scrolling down	T.W.
$EC_{\text{Week button}}$	Week button being clicked.	X-axis display should change to seven days. The data is the average of each day.	D.A.R
$EC_{\text{Day button}}$	Day button being clicked.	The x-axis should change to show the data for each hour.	D.A.R
$EC_{\text{Legend Functionality}}$	Different legend fields being clicked.	The graph should have the proper color coordinated display. The color should coordinate with the legend color.	D.A.R

		The y-axis labeling should change between percentage and temperature.	

Data:

Function: Make calls to backend API to get data for certain days and weeks.

Equivalence Class	Input	Expected Output	Tested by
$EC_{\text{date with null data}}$	“4/19/2017”	Graph displays no data, y-axis disappears, 400 responses from API calls	H.B.B.
$EC_{\text{date with no data}}$	“1/20/2017”	Graph displays no data, y-axis disappears, 400 responses from API calls	H.B.B.
$EC_{\text{date with data}}$	“4/30/2017”	Graph displays data, no errors	H.B.B.

Login(Design):

Function: The login page can decide what to do depending on localStorage; if the user has already logged in, it should re-direct the browser, and display other things based on the user not having the API token (or not being able to store one)

Equivalence Class	Input	Expected Output	Tested by
$EC_{YesAPIToken}$, $EC_{YesLocalStorage}$	API token in local storage	Redirect to webapp	Z.D.H.
$EC_{noLocalStorage}$	No localStorage	#errMess says update browser	Z.D.H.
$EC_{noAPIToken}$	No API token	Redirects work with valid input	Z.D.H.
$EC_{Consistent\ Display}$	Resize the page to make it small.	Grid layout should stay the same. Even when screen size shrinks down.	D.A.R
$EC_{Initial\ Display}$	Launch the webpage.	All divs and images should be formatted correctly. Everything should be readable and in full view.	D.A.R

Notifications:

Recent are scraped from the server based on per-node environmental limits, and immediately available environmental data. If we can make requests to the server, we should be able to scrape them based on the format we are assured the API responds with. If an API request fails, the user is informed as such.

Equivalence Class	Input	Expected Output	Tested by
$EC_{\text{no internet, valid token}}$	[no internet access]	[{ type: 'error', value: true, content: 'Failure to POST to database API.' }]	ZDH
$EC_{\text{internet, invalid token}}$	[faulty API token]	[{ type: 'error', value: true, content: 'Failure to POST to database API.' }]	ZDH
$EC_{\text{internet, valid API token}}$	[valid API token]	True notifications	ZDH

Login Function:

Returns API token and a list of all nodes as well as username, save token to local storage which allows us to make API request per user based on the data required by the webapp.

Equivalence Class	Input: Username	Input: Password	Expected Output	Tested by
$EC_{UID,Invalid}$ and $EC_{PW, Valid}$	“fakeid”	“sustainability”	“Invalid login”	D.B.R
$EC_{UID,Valid}$ and $EC_{PW, Invalid}$	“sustainability”	“fakepw”	“Invalid login”	D.B.R.
$EC_{UID, Valid}$ and $EC_{pw, Valid}$	“sustainability”	“sustainability”	Redirected to webapp	D.B.R.
$EC_{UID, Invalid}$ and $EC_{PW, Invalid}$	“fakeid”	“fakepw”	“Invalid login”	D.B.R.
$EC_{UID, invalid/Valid}$ and $EC_{PW, None}$	“fakeid”/ “sustainability”	“”(empty string)	“Fill out all fields”	D.A.R
$EC_{UID, None}$ and $EC_{PW, Invalid/Valid}$	“”(empty string)	“fakeid”/ “sustainability”	“Fill out all fields”	D.B.R.

Equivalence Class	Input:	Expected Output:	Tested By:
EC_{Logout}	Click the logout icon in the top right corner of the dashboard.	The user should be taken back to the login page.	D.A.R
$EC_{Social\ media\ links}$	Click the Instagram or Facebook icon.	Takes the user to the facebook or instagram login page.	D.A.R
$EC_{About\ page}$	Click the SlugSense logo.	Takes you to the “about” page, which has info on the members of the project.	D.A.R