Test tasks for Automation Engineer

**Additional Information**: BASE\_URL for testing <https://cc.healthrecoverysolutions.com/login>

1. What page element selectors do you know?

**Answer : CSS Selector using name , id , class and attributes**

1. Rate your JS knowledge from 0 to 10 (or another language, just indicate)

**Answer : 6 , Since I have started implementing cypress recently so I have the basic prior knowledge of JS which will be required to code to make cypress test functionality work**

1. Rate your Cypress OR Selenium knowledge from 0 to 10

**Answer : 8 ,there are few features of cypress are left which I still need to be explored**

1. Use selectors to get a list of all admins  
    *<ul class="users\_list">*

*<li class="admin">one</li>*

*<li class="user">two</li>*

*<li class="admin">three</li>*

*<li class="user">four</li>*

*</ul>*

***Answer : ul.users\_list > li.admin***

1. Use selectors to list all users by data attribute name="user"  
   *<ul class="users\_list">*

*<li class="admin">one</li>*

*<li name="user"">two</li>*

*<li class="admin">three</li>*

*<li name="user">four</li>*

*</ul>*

***Answer : ul.users\_list > li[name=’user’]***

1. Filter the list by color attribute. The condition is that you first get the complete list of elements and then filter on the attribute. You need to get a list of items with a red marker.  
   *<ul class="users\_list">*

*<li class="user" color=”red”>test1</li>*

*<li class="user" color=”green”>test2</li>*

*<li class="user" color=”red”>test3</li>*

*<li class="user" color=”green”>test4</li>*

*<li class="user" color=”red”>test5</li>*

*<li class="user" color=”green”>test6</li>*

*<li class="user" color=”green”>test7</li>*

*<li class="user" color=”red”>test8</li>*

*<li class="user" color=”green”>test9</li>*

*<li class="user" color=”red”>test10</li>*

*<li class="user" color=”green”>test11</li>*

*</ul>*

***Answer : cy.get(‘ul.users\_list’).find(‘li’)***

***cy.get(‘li’).filter(:get(li[color=’red’])’)***

1. What actions for elements do you know?

**Answer : Clear , Type , Click , dblclick , rightclick , Check , invoke (text , show , val) , drag , attachFile , scrollIntoView, Trigger(‘mouseover’) ,select dropdown value and should for assertion**

1. How to get an element using Cypress OR Selenium?

**Answer :** **Using get method with CSS selector and using contains method**

1. How to work with select in Cypress OR Selenium?

**Answer :** **We can select a single value or multiple values from the dropdown using the select() command**

1. What actions can be applied to an element using Cypress OR Selenium?

**Answer : type() , focus() , blur() , clear() , submit() , click() , dblclick() , rightclick() , check() ,uncheck(),**

**select() , scrollIntoView() , scrollTo() ,trigger()**

1. Does Сypress OR Selenium support drag and drop?

**Answer : Yes It does support drag and drop, we have to install plugin for drag and dop and need to setup require(‘@4tw/cypress-drag-drop’) to command.js file under support file to make it functional**

1. Does Сypress OR Selenium support file uploads? How?

**Answer : Yes It does support file uploads using fixtures as directory for saving the files and we can use attachFile() to perform the upload action**

1. **Practical task[UI]**

Open the above link in your browser. Write test cases (cases must be complete in terms of assertions). Implement these cases with Cypress/Selenium and the Page Objects pattern.  
**For Cypress**: The weblink must be used with cypress.env.json

* Add some scripts to package.json:

1. Opening Cypress UI

2. Running tests in a specific browser.

* In on("before:browser:launch", (browser = {}, launchOptions) => {}); add additional launch options
  + incognito
  + fullscreen
* Write all possible test cases for the login page. There should be full assertions, without assertions the task will not be accepted.

**For Selenium**:  
\* Configure webdriver using capabilities  
\* Write all possible test cases for the login page. There should be full assertions, without assertions the task will not be accepted.

1. **Practical task[API-> Rest-Assured]**

Write tests for API based on previously created cases. Tests can be written linearly. Add data models and assertions to your tests. Take base URL, base path, and endpoints from the browser console.

In the ResponseSpecification, add a code status check and a content type check. Map the server response to the object.

1. Submit your test cases to GitHub as a project with package loaders. If this condition is not met, the task will not be accepted as done . Projects must be public.

**Note : Has a very good experience working with BDD framework with cypress**