**Postman Test Suite – DataUSA.io/api**

|  |  |  |
| --- | --- | --- |
| **Test Case ID: TC\_DATAUSA\_001** | | |
| Description: Get industry sectors, industry groups, state locations and detailed occupations information from datausa.io. | | |
| Preconditions:  1. Must have “[Data USA - Billy Chen](https://www.postman.com/bchenouyang/workspace/billy-chen-s-workspace/collection/40937109-984fcb0b-e2bf-43fc-b75f-7082b51fcbed?action=share&creator=40937109)” Collections in Postman. Highly recommended to fork the collection. | | |
| **Steps** | **Data** | **Expected Result** |
| 1. Go to ‘Data USA – Billy Chen’ collections folder > Variables |  | The only variables populated should be:  - baseUrl  - dataUrl  - nationID  - workforceStatus |
| 2. Go to Functional Testing > Extract Data folder |  | The following GET requests should be available:  - GetIndustryGroups  - GetIndustrySectors  - GetStateLocations  - GetCountyLocations  - GetDetailedOccupations  - GetMeasures |
| 3. Go to GetIndustryGroups and click ‘Send’ for the request |  | In the test results, verify that:  - status code is 200  - extracted data is successfully parsed  - data is saved in collections |
| 4. Go back to ‘Data USA – Billy Chen’ collections folder |  | Verify that listOfIndustryGroups variable was populated. |
| 5. Repeat step 3 and 4 with the other GET requests in “Extract Data” folder. The requests to be run are marked in Data. | - GetIndustrySectors  - GetStateLocations  - GetDetailedOccupations | All required GET requests were successfully run and their respective lists populated in the collections variables. |

|  |  |  |
| --- | --- | --- |
| **Test Case ID: TC\_DATAUSA\_002** | | |
| Description: Verify information about the employed population by a random state | | |
| Preconditions:  1. Must have run TC\_DATAUSA\_001 | | |
| **Steps** | **Data** | **Expected Result** |
| 1. Go to State Information Tests >  EmployedPopulationByState |  | The GET request exist within the folder. If the request is run for the first time, the variables should be highlighted in red. |
| 2. Send the GET request by clicking ‘Send’. |  | Should be able to send the request without errors. |
| 3. Verify the response body |  | Verify that:  - ID Year ranges from 2022 to 2014  - Age brackets are shown  - Total Population is shown  - Geography is populated by one state |
| 3. Switch to “Test Results” of the response. |  | Verify that:  - Status Code is 200  - Response time has passed  - Response body is JSON and its schema is valid |
| 4. Go to ‘Data USA – Billy Chen’ collections folder and verify the variables:  - stateID  - stateName  - drilldownsVal  - measurementVal |  | StateID, stateName, drilldownsVal and measurementVal variables should be populated automatically.  Additionally, the StateID should correspond to the stateName. |

|  |  |  |
| --- | --- | --- |
| **Test Case ID: TC\_DATAUSA\_003** | | |
| Description: Verify information about the workforce population in a specified industry sector. | | |
| Preconditions:  1. Must have run TC\_DATAUSA\_001 | | |
| **Steps** | **Data** | **Expected Result** |
| 1. Go to ‘Data USA – Billy Chen’ collections folder > Variables |  | The variable listOfIndustrySectors should be populated. |
| 2. Note down an ID of a specified industry sector, such as 31-33 for Manufacturing, and write this ID in the variable ‘industrySectorID’. Save the changes. | 54-56  928110  48-49,22 | IndustrySectorID should be manually populated. |
| 3. Go to Industry Tests >  WorkforceByIndustrySector |  | The GET request exist within the folder. If the request is run for the first time, some of the variables should be highlighted in red. |
| 4. Send the GET request by clicking ‘Send’. |  | Should be able to send the request without errors.  Please note that some industry sectors do not have data. Some of the ones that have data are presented in Step 2’s Data section. |
| 5. Verify the response body |  | The industry sector should match with the one specified from the ID and the Total Population number should be shown. |
| 5. Switch to “Test Results” of the response. |  | Verify that:  - Status Code is 200  - Response time has passed  - All StateIDs match the State Names  - Response body is JSON and its schema is valid |
| 6. Go back to ‘Data USA – Billy Chen’ collections folder and verify the variables:  - industrySectorName  - drilldownsVal  - measurementVal |  | industrySectorName, drilldownsVal and measurementVal variables should be populated automatically. |

|  |  |  |
| --- | --- | --- |
| **Test Case ID: TC\_DATAUSA\_004** | | |
| Description: Check information about the average wage by a specified occupation. | | |
| Preconditions:  1. Must have run TC\_DATAUSA\_001 | | |
| **Steps** | **Data** | **Expected Result** |
| 1. Go to ‘Data USA – Billy Chen’ collections folder > Variables |  | The variable listOfOccupation should be populated. |
| 2. Note down an ID of a specified occupation, such as 532010 for Pilots, and write this ID in the variable ‘occupationID’.  Save the changes. | 532010  254021  519010 | occupationID should be manually populated. |
| 3. Go to Occupation Tests >  AverageWagesByOccupation |  | The GET request exist within the folder. If the request is run for the first time, some of the variables should be highlighted in red. |
| 4. Send the GET request by clicking ‘Send’. |  | Should be able to send the request without errors.  Please note that some occupations do not have wage data. Some of the ones that have data are presented in Step 2’s Data section. |
| 5. Verify the response body |  | The detailed occupation name should match with the one specified from the ID and the Total Population number and Average Wage should be shown. |
| 5. Switch to “Test Results” of the response. |  | Verify that:  - Status Code is 200  - Response time has passed  - Response body is JSON and its schema is valid |
| 6. Go back to ‘Data USA – Billy Chen’ collections folder and verify the variables:  - occupationName  - drilldownsVal  - measurementVal |  | occupationName, drilldownsVal and measurementVal variables should be populated automatically. |

|  |  |  |
| --- | --- | --- |
| **Test Case ID: TC\_DATAUSA\_005** | | |
| Description: Verify that no information is received about household incomes of an invalid state. Please note that this test case seems to uncover a defect. | | |
| Preconditions:  1. Must have run TC\_DATAUSA\_001 | | |
| **Steps** | **Data** | **Expected Result** |
| 1. Go to ‘Data USA – Billy Chen’ collections folder > Variables |  | The variable listOfStates should be populated. |
| 2. Write an invalid State ID with correct structure in the variable ‘StateID’. Make sure to erase the value for stateName. Save the changes. | 04000US00  04000US57  04000US99 | StateID should be manually populated.  StateName has no current value. |
| 3. Go to State Information Tests >  HouseholdIncomeByState |  | The GET request exist within the folder. If the request is run for the first time, some of the variables should be highlighted in red. |
| 4. Send the GET request by clicking ‘Send’. |  | Should be able to send the request without errors. |
| 5. Verify the response body |  | Verify that the “data” from the json response is empty. |
| 5. Switch to “Test Results” of the response. |  | Verify that:  - Status Code is 200  - Response time has passed  - Response body is JSON and its schema is NOT passing. |
| 6. Go back to ‘Data USA – Billy Chen’ collections folder and verify the variables:  - stateName  - drilldownsVal  - measurementVal |  | drilldownsVal and measurementVal variables should be populated automatically.  StateName should not have any value. |

**Cypress Test Suite – NBA.com**

|  |  |  |
| --- | --- | --- |
| **Test Case ID: TC\_NBA\_001** | | |
| Description: New user should not be able to register without passing recaptcha. Associated automation test - RegisterNewUser.cy.js | | |
| Preconditions:  1. Must have a web browser: Chrome, Edge, Firefox... | | |
| **Steps** | **Data** | **Expected Result** |
| 1. Open browser and go to [www.nba.com](http://www.nba.com/) |  | The landing page should appear in the browser. |
| 2. Click ‘Sign In’ at the top navigation bar > Sign in with NBA ID |  | Sign in page should appear with title ‘Sign In With Your NBA ID’ |
| 3. Click ‘Don’t have an NBA ID?’ link |  | A registration page should appear with title ‘Create NBA ID’. |
| 4. Fill the required fields in the page with random information. Check to agree the Terms of Use. |  | All required information for registration are filled. |
| 5. Without checking the recaptcha ‘I’m not a robot’, click ‘Create Account’. |  | Registration should not pass and an error message ‘Captcha validation required.’ should appear. |

|  |  |  |
| --- | --- | --- |
| **Test Case ID: TC\_NBA\_002** | | |
| Description: New User should login before subscribing to NBA’s league pass. Associated automation test – SubscribeLeaguePass.cy.js | | |
| Preconditions:  1. Must have a web browser: Chrome, Edge, Firefox... | | |
| **Steps** | **Data** | **Expected Result** |
| 1. Open browser and go to [www.nba.com](http://www.nba.com/) |  | The landing page should appear in the browser. |
| 2. At the top bar, click the ‘League Pass’ button in yellow. |  | The league pass page should appear with endpoint ‘league-pass-purchase’. |
| 3. Scroll below to the subscription options. |  | Two options should appear for subscription with properties:  - League Pass for 57.99  - League Pass Premium for 79.99 highlighted as most popular  Both options have season and monthly choices. |
| 4. Click ‘Subscribe’ for League Pass Premium |  | Sign in page should appear with title ‘Sign In With Your NBA ID’ |
| 5. Go back to the league pass page and select ‘Subscribe’ for League Pass |  | Sign in page should appear with title ‘Sign In With Your NBA ID’ |

|  |  |  |
| --- | --- | --- |
| **Test Case ID: TC\_NBA\_003** | | |
| Description: Users should be able to add an item from the NBA Store. Associated automation test – StoreAddItems.cy.js | | |
| Preconditions:  1. Must have a web browser: Chrome, Edge, Firefox... | | |
| **Steps** | **Data** | **Expected Result** |
| 1. Open browser and go to [www.nba.com](http://www.nba.com/) |  | The landing page should appear in the browser. |
| 2. At the top bar, select ‘Store’ |  | The NBA Store page should appear with endpoint ‘store.nba.com/source/bm-nbacom-hp’ |
| 3. Go to T-Shirts |  | The T-Shirt product page should appear with endpoint ‘/t-shirts/’ |
| 4. Select a random shirt from the product page |  | A product description page should appear with size, quantity, and ‘add to cart’ option.  The product should also have details and a description. |
| 5. Select a random size and quantity and click ‘Add to Cart’ |  | A ‘Price Locked In’ loading frame should appear in the current page.  Shortly after, the store cart page should appear with an order summary and ‘Checkout’ button. Verify that the product selected and Cart Total matches the previous selection. |

|  |  |  |
| --- | --- | --- |
| **Test Case ID: TC\_NBA\_004** | | |
| Description: Users should be able to delete an item from cart in the NBA Store. Associated automation test – StoreDeleteItems.cy.js | | |
| Preconditions:  1. Must have a web browser: Chrome, Edge, Firefox... | | |
| **Steps** | **Data** | **Expected Result** |
| 1. Open browser and go to [www.nba.com](http://www.nba.com/) |  | The landing page should appear in the browser. |
| 2. At the top bar, select ‘Store’ |  | The NBA Store page should appear with endpoint ‘store.nba.com/source/bm-nbacom-hp’ |
| 3. Go to T-Shirts |  | The T-Shirt product page should appear with endpoint ‘/t-shirts/’ |
| 4. Select a random shirt from the product page |  | A product description page should appear with size, quantity, and ‘add to cart’ option.  The product should also have details and a description. |
| 5. Select a random size and quantity and click ‘Add to Cart’ |  | A ‘Price Locked In’ loading frame should appear in the current page.  Shortly after, the store cart page should appear with an order summary and ‘Checkout’ button. Verify that the product selected and Cart Total matches the previous selection. |
| 6. Click on the ‘X’ button at the top right of the product information to delete the item in the cart. |  | The order summary should disappear and a text ‘Your Shopping Cart Is Empty’ should be in the page. A ‘Continue Shopping’ button should also be present. |

|  |  |  |
| --- | --- | --- |
| **Test Case ID: TC\_NBA\_005** | | |
| Description: Users should not be able to see game scores once hidden. Associated automation test – HideGameScores.cy.js | | |
| Preconditions:  1. Must have a web browser: Chrome, Edge, Firefox... | | |
| **Steps** | **Data** | **Expected Result** |
| 1. Open browser and go to [www.nba.com](http://www.nba.com/) |  | The landing page should appear in the browser. |
| 2. At the top bar, click ‘Games’ |  | The game page with endpoint ‘games?date={today’s date}’, where {today’s date} is current date, should appear.  The scores of previous games should also appear. |
| 3. Toggle ‘Hide Scores’ at the head of page body |  | All scores, points and game leaders information should disappear except for the expected time of future games. |