Integration test plan

Test 1 – Account Creation Check

Go to the create account screen at the **Frontend** and the Database at the **Backend** and enter the following tests.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Expected Result | Actual Result | Succes? |
| F\_ACC\_0.1 | In username type “UserName1”, in password type “Password1” and in email  type “dummy@fontys.nl”. | The screen will confirm creation of account (“Account created”) and an entry of the user account info will be added to the Database file in the Backend. |  | NOK |
| F\_ACC\_0.2 | After completing ACC\_0.1 reopen the create account screen. In username type “UserName1”, in password type “Password1” and in email  type “dummy2@fontys.nl”. | The backend will verify that this username is already included in the Database and trigger an error message for the frontend.  The screen will show an error message “This username is already taken, please try again”. |  | NOK |
| F\_ACC\_0.3 | After completing ACC\_0.1 reopen the create account screen. In username type “UserName2”, in password type “Password1” and in email  type “dummy@fontys.nl”. | The backend will verify that this email is already included in the Database and trigger an error message for the frontend.  The screen will show an error message “This email adress is already taken, please try again”. |  | NOK |

Test 2 – Account Login Check

Go to the login screen at the **Frontend** and the Database at the **Backend** and enter the following tests.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Expected Result | Actual Result | Succes? |
| F\_AL\_01 | Correct data is entered in the fields and the login button is pressed | The login menu goes away. And you will be redirected to another page | Works from the backend only | NOK |
| F\_AL\_02 | The right Login Name is entered in the login field, but the wrong password | There is an error showing, that the login name is incorrect | Works from the backend only | NOK |
| F\_AL\_03 | The wrong Login Name is entered in the login field, but the right password | There is an error showing, that the password is incorrect | Works from the backend only | NOK |
| F\_AL\_04 | The wrong Login Name is entered in the login field, but the wrong password | There is an error showing, that the login name is incorrect | Works from the backend only | NOK |

Test 3 – Current Skittle Count Check

Go to the sorting request screen at the **Frontend** and the Database at the **Backend** enter the following tests.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Expected Result | Actual Result | Succes? |
| F\_CSCC\_0.1 | While the sorting request screen is active, check the number of available skittles under the color Yellow.  Go to the Backend and open the Database and check the amount of Skittles under the color Yellow. | Both Skittle counts have the same amount of available Skittles under the color Yellow. |  | OK |
| F\_CSCC\_0.2 | While the sorting request screen is active, check the number of available skittles under the color Red.  Go to the Backend and open the Database and check the amount of Skittles under the color Red. | Both Skittle counts have the same amount of available Skittles under the color Red. |  | OK |
| F\_CSCC\_0.3 | While the sorting request screen is active, check the number of available skittles under the color Purple.  Go to the Backend and open the Database and check the amount of Skittles under the color Purple | Both Skittle counts have the same amount of available Skittles under the color Purple. |  | OK |
| F\_CSCC\_0.4 | While the sorting request screen is active, check the number of available skittles under the color Green.  Go to the Backend and open the Database and check the amount of Skittles under the color Green | Both Skittle counts have the same amount of available Skittles under the color Green. |  | OK |
| F\_CSCC\_0.5 | While the sorting request screen is active, check the number of available skittles under the color Orange.  Go to the Backend and open the Database and check the amount of Skittles under the color Orange | Both Skittle counts have the same amount of available Skittles under the color Orange. |  | OK |

Test 4-Order Skittles

Go to the sorting request screen at the **Frontend** and the Database at the **Backend** enter the following tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Expected Result | Actual Result | Succes? |
| F\_OS\_01 | All the selectors are correctly filled in (greater than zero), then we press ok | A JSON message goes to the backend with the correct amount of chosen skittles. |  | OK |
| F\_OS\_02 | An incorrect JSON string is send from the frontend to the backend | The backend gives you a warning that the JSON string send is not correct | Incorrect data cannot be entered in the Frontend app | OK |
| F\_OS\_03 | Backend does not respond to the send message (no connection). | The message should be saved for when the connection comes back online. After that the backend should redo the request. | In this case error 404 is thrown. Data is not saved | NOK |

Test 5 –Skittle Count Check After Sort

Go to the Database at the **Backend** and the sorting reservoir at the **Arduino**, and enter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Expected Result | Actual Result | Succes? |
| B\_SCCAS\_0.1 | At the Database check and note the current amounts of Yellow skittles at each color.  Then take two (2) Yellow Skittles, add them to the sorting reservoir of the Arduino sorter.  Start the Arduino sorting process. After the process is finished, check and note the current amount of Yellow Skittles at the Database | The amount of Yellow Skittles has incremented by two (2). |  | OK |
| B\_SCCAS\_0.2 | At the Database check and note the current amounts of Red skittles at each color.  Then take two (2) Red Skittles, add them to the sorting reservoir of the Arduino sorter.  Start the Arduino sorting process. After the process is finished, check and note the current amount of Red Skittles at the Database. | The amount of Red Skittles has incremented by two (2). |  | OK |
| B\_SCCAS\_0.3 | At the Database check and note the current amounts of Purple skittles at each color.  Then take two (2) Purple Skittles, add them to the sorting reservoir of the Arduino sorter.  Start the Arduino sorting process. After the process is finished, check and note the current amount of Purple Skittles at the Database. | The amount of Purple Skittles has incremented by two (2). |  | OK |
| B\_SCCAS\_0.4 | At the Database check and note the current amounts of Green skittles at each color.  Then take two (2) Green Skittles, add them to the sorting reservoir of the Arduino sorter.  Start the Arduino sorting process. After the process is finished, check and note the current amount of Green Skittles at the Database. | The amount of Green Skittles has incremented by two (2). |  | OK |
| B\_SCCAS\_0.5 | At the Database check and note the current amounts of Orange skittles at each color.  Then take two (2) Orange Skittles, add them to the sorting reservoir of the Arduino sorter.  Start the Arduino sorting process. After the process is finished, check and note the current amount of Orange Skittles at the Database. | The amount of Orange Skittles has incremented by two (2). |  | OK |

the following tests.

Test 6 –Skittle Count Check After Order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Expected Result | Actual Result | Succes? |
| SCCAO\_0.1 | While the sorting request screen is active, check and note the number of available Yellow Skittles.  Enter an order of two (2) Yellow Skittles and click the order button.  After the order has been sorted, check and note the number of available Yellow Skittles again.  Check the order receptacle of the Arduino. | The amount of available Yellow Skittles at the sorting request screen has decreased by two (2).  The order receptacle at the Arduino now holds two (2) Yellow Skittles. |  | OK |
| SCCAO\_0.2 | While the sorting request screen is active, check and note the number of available Red Skittles.  Enter an order of two (2) Red Skittles and click the order button.  After the order has been sorted, check and note the number of available Red Skittles again.  Check the order receptacle of the Arduino. | The amount of available Red Skittles at the sorting request screen has decreased by two (2).  The order receptacle at the Arduino now holds two (2) Red Skittles. |  | OK |
| SCCAO\_0.3 | While the sorting request screen is active, check and note the number of available Purple Skittles.  Enter an order of two (2) Purple Skittles and click the order button.  After the order has been sorted, check and note the number of available Purple Skittles again.  Check the order receptacle of the Arduino. | The amount of available Purple Skittles at the sorting request screen has decreased by two (2).  The order receptacle at the Arduino now holds two (2) Purple Skittles. |  | OK |
| SCCAO\_0.4 | While the sorting request screen is active, check and note the number of available Green Skittles.  Enter an order of two (2) Green Skittles and click the order button.  After the order has been sorted, check and note the number of available Green Skittles again.  Check the order receptacle of the Arduino. | The amount of available Green Skittles at the sorting request screen has decreased by two (2).  The order receptacle at the Arduino now holds two (2) Green Skittles. |  | OK |
| SCCAO\_0.5 | While the sorting request screen is active, check and note the number of available Orange Skittles.  Enter an order of two (2) Orange Skittles and click the order button.  After the order has been sorted, check and note the number of available Orange Skittles again.  Check the order receptacle of the Arduino. | The amount of available Orange Skittles at the sorting request screen has decreased by two (2).  The order receptacle at the Arduino now holds two (2) Orange Skittles. |  | OK |

Go the sorting request screen at the **Frontend** and the order receptacle of the **Arduino**, then enter the following tests.

Test 7-Sorted Skittles - back-end

Go to the Database at the **Backend** and the sorting reservoir at the **Arduino**, and enter

the following tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Expected Result | Actual Result | Succes? |
| B\_SK\_01 | The backend should store the sorted values of the color when the MQTT server sends a JSON text. | When the JSON string is acquired. The backend should store the given values in a list. |  | OK |
| B\_SK\_02 | When incorrect data is send from the MQTT server. The back-end should respond with a message. | When the incorrect data is catched. The backend should return a message to the frontend that something went wrong. | When incorrect Data is send backend loses connection with the broker | NOK |
| B\_SK\_03 | When a request from the front-end comes to get the correct amount of skittles left. The backend makes a get request to the MQTT server to give the data, and sends that data back to the front-end | The data send by the MQTT server is stored in a repository. This data is send via JSON to the front-end who then stores it in the amount available on the page. |  | OK |

Test 9-Sorted Skittles – Arduino

**Arduino pre-sort** with **Backend**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Expected Result | Actual Result | Succes? |
| A\_SK\_01 | When the sorting machine has finished sorting. Data needs to be send to an MQTT Server. | When finished sorting. The Arduino sends the values to an MQTT server who then waits from a request from the backend to get the information. |  | OK |
| A\_SK\_02 | When the servo motor is not running so that the machine stops sorting. That information has to be stored into a logfile | As soon as the machine stops working via a busted servo. A log file should be updated or created. |  | NOK |
| A\_SK\_03 | When there are no skittles left to be sorted. There should be a message saying the bin is empty. | As soon as the machine detects there are no more skittles left to sort. A message needs to be send to the backend informing the problem. | In this case Arduino sends last known status via MQTT and prints a message that the bin is empty in IDE Serial Monitor | OK |