**Test plan and Results**

**Overall Test Plan**

Our testing plan is to test each system component individually using mock data. Each of these will have expected and unexpected conditions covered for each component. This will consist mostly of Unit tests that will indicate the purpose of the test as well as the expected result of the test. The other portion of our testing will be integration testing where we will test the components working together which will make sure that the connection between these components is functionally working properly.

**Test case Descriptions**

|  |  |
| --- | --- |
| LT1.1 | **Login Test 1** |
| LT1.2 | This test will ensure that the user will be logged in with valid credentials. |
| LT1.3 | This test will run mock data that will send a fake request to the backend and if the backend were to return a 200 indicating that the user’s credentials are valid it will successfully route to the correct page. |
| LT1.4 | Inputs: A email and password used as valid credentials. |
| LT1.5 | Outputs: A navigation to a list designated by the users unique ID. |
| LT1.6 | Normal |
| LT1.7 | Blackbox |
| LT1.8 | Functional Test |
| LT1.9 | Unit Test |

|  |  |
| --- | --- |
| LT2.1 | **Login Test 2** |
| LT2.2 | This test will ensure that invalid credentials will not log in. |
| LT2.3 | This test will run mock data that will send a fake request to the backend and if the backend were to return a 401 Unauthorized error indicating that the user’s credentials are invalid it will display a validation error to the user. |
| LT2.4 | Inputs: A email and password used as invalid credentials. |
| LT2.5 | Outputs: Validation Error on screen indicating invalid credentials. |
| LT2.6 | Abnormal |
| LT2.7 | Blackbox |
| LT2.8 | Functional Test |
| LT2.9 | Unit Test |

|  |  |
| --- | --- |
| LT3.1 | **Login Test 3** |
| LT3.2 | This test will check that the backend can handle login. |
| LT3.3 | This test will run mock data from the frontend to test that the backend will encrypt and send the encrypted data to the database successfully. |
| LT3.4 | Inputs: Database Query using Email and Password for user. |
| LT3.5 | Outputs: 1 Matched row in database returned. |
| LT3.6 | Normal |
| LT3.7 | Whitebox |
| LT3.8 | Functional Test |
| LT3.9 | Unit Test |

|  |  |
| --- | --- |
| IA1.1 | **Image Api Test 1** |
| IA1.2 | This test will verify that we can pull image data from the Serp Api on the back end. |
| IA3 | This test will run mock data that will query the serp api and check if the query has been successful. |
| IA1.4 | Inputs: Item name |
| IA1.5 | Outputs: 200 Okay, Table rows returned |
| IA1.6 | Normal |
| IA1.7 | Blackbox |
| IA1.8 | Functional test |
| IA1.9 | Unit Test |

|  |  |
| --- | --- |
| IA2.1 | **Image Api Test 2** |
| IA2.2 | This test will verify that the front end recieves the data from the backend request. |
| IA2.3 | This test will run data that will query the serp api and check if the query has returned images from the api. |
| IA2.4 | Inputs: Item name |
| IA2.5 | Outputs: 200 Okay, list of images/urls |
| IA2.6 | Normal |
| IA2.7 | Blackbox |
| IA2.8 | Functional test |
| IA2.9 | Unit Test |

|  |  |
| --- | --- |
| IS1.1 | **Image Search Test 1** |
| IS1.2 | This test will verify that the user has entered valid information. |
| IS1.3 | This test will run data that will show the intended outcome |
| IS1.4 | Inputs: Item name |
| IS1.5 | Outputs: 200 Okay, list of images/urls |
| IS1.6 | Normal |
| IS1.7 | Whitebox |
| IS1.8 | Functional test |
| IS1.9 | Unit Test |

|  |  |
| --- | --- |
| IS2.1 | **Image Search Test 2** |
| IS2.2 | This test will verify that the application returns an error when they do not put valid information in the front end. |
| IS2.3 | This test will run an invalid string into the item text entry and ensure that the user is notified that they should enter a word. |
| IS2.4 | Inputs: invalid Item name |
| IS2.5 | Outputs: error |
| IS2.6 | Abnormal |
| IS2.7 | Whitebox |
| IS2.8 | Functional test |
| IS2.9 | Unit Test |

|  |  |
| --- | --- |
| AI1.1 | **Add Item Test 1** |
| AI1.2 | This test will navigate to add item page. |
| AI1.3 | This test will test the navigation of the application from the home page to the add item page based on a click event of a button. |
| AI1.4 | Inputs: Button on Click event. |
| AI1.5 | Outputs: Navigation to Add Item Page. |
| AI1.6 | Normal |
| AI1.7 | Whitebox |
| AI1.8 | Functional Test |
| AI1.9 | Unit Test |

|  |  |
| --- | --- |
| AI2.1 | **Add Item Test 2** |
| AI2.2 | This test will add a single item to list. |
| AI2.3 | This test will use mock data for each required input then send a request to the backend with a 200 okay response. After the response the UI will navigate back to the users list. |
| AI2.4 | Inputs: Name, Price, URL, Description |
| AI2.5 | Outputs: 200 Okay and navigation back to home screen. |
| AI2.6 | Normal |
| AI2.7 | Blackbox |
| AI2.8 | Functional Test |
| AI2.9 | Unit Test |

|  |  |
| --- | --- |
| AI3.1 | **Add Item Test 3** |
| AI3.2 | This test will fail to add a single item to the list. |
| AI3.3 | This test will use mock data for some of the required fields but not all of them, which will result in error validation to respond to fix the issue. No data will be sent to the backend. |
| AI3.4 | Inputs: Name, Price only |
| AI3.5 | Outputs: Error validation on missing fields |
| AI3.6 | Abnormal |
| AI3.7 | Whitebox |
| AI3.8 | Functional Test |
| AI3.9 | Unit Test |

|  |  |
| --- | --- |
| IL1.1 | **Item List Test 1** |
| IL1.2 | This test will ensure the database is functioning |
| IL1.3 | This test will run mock data that will query our azure database and check if the query has been successful. |
| IL1.4 | Inputs: Database connection information, |
| IL1.5 | Outputs: 200 Okay, Table rows returned |
| IL1.6 | Normal |
| IL1.7 | Blackbox |
| IL1.8 | Functional test |
| IL1.9 | Unit Test |

|  |  |
| --- | --- |
| IL2.1 | **Item List Test 2** |
| IL2.2 | This test will ensure the data from the items data is shown in the front end. |
| IL23 | This test will check if the items show in on the list screen. |
| IL2.4 | Inputs: Mock data returned from backend |
| IL2.5 | Outputs: 200 Okay, Table rows returned |
| IL2.6 | Normal |
| IL2.7 | Blackbox |
| IL2.8 | Functional test |
| IL2.9 | Unit Test |

|  |  |
| --- | --- |
| EI1.1 | **Edit Item Test 1** |
| EI1.2 | This test will be used to test the navigation to edit page. |
| EI1.3 | This test will test the navigation of the application from the home page to the edit item page based on a click event of a button labeled edit. |
| EI1.4 | Inputs: Button on Click event. |
| EI1.5 | Outputs: Navigation to Edit Item Page. |
| EI1.6 | Normal |
| EI1.7 | Whitebox |
| EI1.8 | Functional Test |
| EI1.9 | Unit Test |

|  |  |
| --- | --- |
| EI2.1 | **Edit Item Test 2** |
| EI2.2 | This test will check if the edit page fetches data correctly. |
| EI2.3 | This test will test that the edit page is populated after fetching mock data. |
| EI2.4 | Inputs: Data to populate fields. |
| EI2.5 | Outputs: All fields will be populated with appropriate data. |
| EI2.6 | Normal |
| EI2.7 | Whitebox |
| EI2.8 | Functional Test |
| EI2.9 | Unit Test |

|  |  |
| --- | --- |
| EI3.1 | **Edit Item Test 3** |
| EI3.2 | This test will check navigation back from edit page no updates. |
| EI3.3 | This test will test the navigation back to the home screen using the on click event  From the “Cancel” Button which will navigate to the home screen with no updates. |
| EI3.4 | Inputs: Cancel click event. |
| EI3.5 | Outputs: All fields will be populated with appropriate data. |
| EI3.6 | Normal |
| EI3.7 | Whitebox |
| EI3.8 | Functional Test |
| EI3.9 | Unit Test |

|  |  |
| --- | --- |
| EI4.1 | **Edit Item Test 4** |
| EI4.2 | This test will check navigation back from edit page after updating. |
| EI4.3 | This test will test when the user clicks the “Submit” button all changes applied to  every field will be sent to the backend to be updated in the database. |
| EI4.4 | Inputs: Data in populated fields. Submit on click event. |
| EI4.5 | Outputs: Request to backend with 200 response assuming database is updated accordingly. |
| EI4.6 | Normal |
| EI4.7 | Whitebox |
| EI4.8 | Functional Test |
| EI4.9 | Unit Test |

|  |  |
| --- | --- |
| DI1.1 | **Delete Item Test 1** |
| DI1.2 | This test will be used to check the deletion feature. |
| DI1.3 | This test will check that the click event of deleting an entry sends an Api request to the backend. After which it will refresh the page once a 200 response is returned. |
| DI1.4 | Inputs: On click event for Delete button. |
| DI1.5 | Outputs: Removed entry from list. |
| DI1.6 | Normal |
| DI1.7 | Blackbox |
| DI1.8 | Functional Test |
| DI1.9 | Unit Test |

|  |  |
| --- | --- |
| FS1.1 | **Full System Test 1** |
| FS1.2 | This test will be used to see that the full system works well together. |
| FS1.3 | We will deploy the application on an active server where we will setup and monitor a user being created and make sure that it is added to the database and able to access their account after creation. |
| FS1.4 | Inputs: Account creation email/password. |
| FS1.5 | Outputs: Appropriate database entries based on data input etc. |
| FS1.6 | Normal |
| FS1.7 | Blackbox |
| FS1.8 | Performance |
| FS1.9 | Integration Test |

|  |  |
| --- | --- |
| FS2.1 | **Full System Test 2** |
| FS2.2 | This test will be used to see that the full system works well together. |
| FS2.3 | We will deploy the application on an active server where we will access a user’s account and add/edit/delete items on their list. As well as test the URL of our linked items in order to purchase. |
| FS2.4 | Inputs: Logging into account, adding items and updating accordingly. |
| FS2.5 | Outputs: Appropriate database entries based on data inputs/requests. |
| FS2.6 | Normal |
| FS2.7 | Blackbox |
| FS2.8 | Functional |
| FS2.9 | Integration Test |

**Test Case Matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LT1** | Normal | Blackbox | Functional | Unit |
| **LT2** | Abnormal | Blackbox | Functional | Unit |
| **LT3** | Normal | Whitebox | Functional | Unit |
| **IA1** | Normal | Blackbox | Functional | Unit |
| **IA2** | Normal | Blackbox | Functional | Unit |
| **IS1** | Normal | Whitebox | Functional | Unit |
| **IS2** | Abnormal | Whitebox | Functional | Unit |
| **AI1** | Normal | Whitebox | Functional | Unit |
| **AI2** | Normal | Blackbox | Functional | Unit |
| **AI3** | Abnormal | Whitebox | Functional | Unit |
| **IL1** | Normal | Blackbox | Functional | Unit |
| **IL2** | Normal | Blackbox | Functional | Unit |
| **EI1** | Normal | Whitebox | Functional | Unit |
| **EI2** | Normal | Whitebox | Functional | Unit |
| **EI3** | Normal | Whitebox | Functional | Unit |
| **EI4** | Normal | Whitebox | Functional | Unit |
| **DI1** | Normal | Blackbox | Functional | Unit |
| **FS1** | Normal | Blackbox | Integration | Integration |
| **FS2** | Normal | Blackbox | Integration | Integration |