**White box testing cases:**

**Presentation tier:**

This tier was tested with Bunit, xunit and Nunit.AutoFixture Libraries

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Scenario | Test Scenario | Test steps | Expected result | Actual result | Pass/Fail |
| 1 | User can be created | 1. Data required for registration is inputted 2. System calls create method for the user | An Id for the created user should be able to be checked | As expected | Pass |
| 2 | User should be able to login | 1. Data required for logging in is added 2. Authorisation is added 3. System calls LoginAsync method | Login page should be converted to Edit user page | As  expected | Pass |
| 3 | Item should be created | 1. Authorisation is added 2. Data required for item creation is inputted 3. The callouts are mocked 4. System calls create method for the item | An Id of the item should be able to be checked | As expected | Pass |

**Business logic tier:**

This tier was tested with xunit library, and the Data access tier has to be running for the tests to work

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Scenario | Test Scenario | Test steps | Expected result | Actual result | Pass/Fail |
| 1 | Create and get user | 1. Data needed for user creation is inputted. 2. System calls createUserAsync method 3. System calls getUsers method | The name of the created user matches the name of the user the system got | As expected | Pass |
| 2 | Create and get Item | 1. Data needed for item creation is inputted. 2. System calls createItemAsync method 3. System calls getItems method | The name of the created item matches the name of the user the system got | As expected | Pass |
| 3 | Edit profile | 1. Data needed for profile editing is inputted 2. System calls updateUserAsync method 3. System calls getUsers method | The name of the updated user matches the name of the user the system got | As expected | Pass |
| 4 | Delete user | 1. Id of the user that needs to be deleted is inputted 2. System calls deleteUserAsync method 3. System calls getUsers method 4. The system loops through existing users to check if the user with the matching id still exists | The user with the deleted id should not be in the database | As expected | Pass |
| 5 | Edit Item | 1. Data needed for Item editing is inputted 2. System calls updateItemAsync method 3. System calls getItemsByIdAsync method | The name of the updated Item matches the name of the Item the system got | As expected | Pass |
| 6 | Delete Item | 1. Id of the Item that needs to be deleted is inputted 2. System calls deleteItemAsync method 3. System calls getItems method 4. The system loops through existing items to check if the item with the matching id still exists | The item with the deleted id should not be in the database | As expected | Pass |