Test Cases

Black-Box Test Cases

These test cases are defined based on the available documentation and execution of the program. The code was not inspected.

#	**Test case (very brief description)**	**Preconditions (any required setup)**	**Test steps (steps executed during testing)**	**Expectation**	**Observation (“pass” or failure description)**
1	Create Account - account creation with invalid input (spaces only)	Launch App	1. Select ‘Create New Account’ button 2. Enter only spaces in the username and password fields	An error message "Username or password is invalid" should be displayed, and the account should not be created	Fail - No error message is displayed, and the account is created
2	Create Account - account creation with an already existing username/password	Username and password should be already existed in the database	1. Select ‘Create New Account’ button 2. Create new account with already existed username/password	An error message “Account is already existing" should be displayed, and the account should not be created	Fail - No error message is displayed, and the account is created
3	Create Account - account creation with non-existing username/password with valid input	Username and password should not be already existed in the database	1. Select ‘Create New Account’ button 2. Enter the valid input	User should be able to create the account	Pass
4	Log in - without username and password	Launch App	Press the "Login" button without any input in both the username and password fields	An error message “Incorrect username/ password“ should be displayed.	Fail - No error message is displayed, and the user can log in

5	Log in - With correct username and incorrect password	Username and password should be already existed in the database	Press the "Login" button with correct username and incorrect password	An error message “Incorrect username/ password“ should be displayed.	Pass
6	Log in - With incorrect username and correct password	Username and password should be already existed in the database	Press the "Login" button with incorrect username and correct password	An error message “Incorrect username/ password“ should be displayed.	Pass
7	Log in - With correct username and password	Username and password should be already existed in the database	Press the "Login" button without incorrect password	User should be able to log in	Pass
8	Forgot password - reset password with unregistered username	Username should not exist in the database	1. Select ‘Forgot password’ button 2. Enter unregistered username in the username field 3. Put the new password in the password field	An error message “Incorrect username/ password“ should be displayed.	Pass
9	Selling Item - without any input	Selling page is loaded	Click ‘Submit’ button without any input	An error message should be displayed and item should not be added.	Pass
10	Selling Item - leave one field empty (except the comment field) and fill the other fields with valid input	Selling page is loaded	1. Leave one of the required fields (item name, category, year of purchase, actual price, selling price, image) empty 2. Fill in the other required fields with valid input 3. Click ‘Submit’ button	An error message should be displayed and item should not be added.	Pass
11	Selling Item - leave the comment field empty and fill the other fields with valid input	Selling page is loaded	1. Leave the comment field empty 2. Fill in the other required fields with valid input 3. Click ‘Submit’ button	The item should be successfully added to the database	Pass

12	Selling Item - fill the ‘Item name’ field with only spaces and fill the other fields with valid input	Selling page is loaded	1. Fill the Item name field with only spaces 2. Fill in the other required fields with valid input 3. Click ‘Submit’ button	An error message should be displayed and item should not be added.	Fail - No error message is displayed, and the item is added
13	Selling Item - with an unsupported image format	Selling page is loaded	1. Click the "Import Image" link 2. Select an unsupported image file format 3. Click ‘Submit’ button	An error message should be displayed and item should not be added.	Pass : Only able to select supported image file in the first place
14	Selling Item - ‘Year of Purchase’ field is negative	Selling page is loaded	1. Fill in the other required fields with valid input 2. Fill the Year of Purchase field with a negative number(ex. -200) 3. Click ‘Submit’ button	An error message should be displayed and item should not be added.	Fail - No error message is displayed, and the item is added
15	Selling Item - Price fields are negative	Selling page is loaded	1. Fill in the other required fields with valid input 2. Fill the Actual price/Selling price fields with a negative price (ex. -200) 3. Click ‘Submit’ button	An error message should be displayed and item should not be added.	Fail - No error message is displayed, and the item is added
16	Selling Item - Put characters in the number field (year, price)	Selling page is loaded	1. Fill in the other required fields with valid input 2. Fill the Year of Purchase/Actual price/ Selling price fields with characters not numbers. 3. Click ‘Submit’ button	An error message should be displayed and item should not be added.	Pass
17	Selling Item - With valid input	Selling page is loaded	1. Fill all the fields with valid input 2. Click ‘Submit’ button	Item should be added without an error	Pass

18	Edit the detail of Item	At least one item should exist in the database	Navigate to the item detail page Click on the 'Edit' button Make changes to the item details Click 'Save'	The item details should be updated in the database	Fail - Feature not existing
19	Delete the item	At least one item should exist in the database	Navigate to the item detail page Click on the 'Delete' button Confirm the deletion	The item should be removed from the database	Fail - Feature not existing
20	Buying the item	At least one item should exist in the database	Navigate to the item detail page Click on the 'Delete' button Confirm the deletion	The item should be marked as sold or removed from the listing	Fail - Feature not existing
21	Reload the list of items	Item should be newly added	Click on the 'Refresh' button to update the list	The list of items should be updated	Pass
22	Sign out	User should be logged in	Click on the ‘Sign out' button to log out	User should be able to log out	Pass

White-Box Test Cases

These additional test cases were defined during inspection of the code.

#	**Test case (very brief description)**	**Preconditions (any required setup)**	**Test steps (steps executed during testing)**	**Expectation**	**Observation (“pass” or failure description)**
1	Test item name property (Product class)	Initialize JavaFX platform by using JavaFXTestSetup class	1. Create Product object with item name “Laptop". 2. Retrieve item name property.	The item name property should be "Laptop".	Pass

2	Test year of purchase property (Product class)	Initialize JavaFX platform by using JavaFXTestSetup class	1. Create Product object with year of purchase 2021. 2. Retrieve year of purchase property.	The year of purchase property should be 2021.	Pass
3	Test actual price property (Product class)	Initialize JavaFX platform by using JavaFXTestSetup class	1. Create Product object with actual price 1000.0. 2. Retrieve actual price property.	The actual price property should be 1000.0.	Pass
4	Test selling price property (Product class)	Initialize JavaFX platform by using JavaFXTestSetup class	1. Create Product object with selling price 800.0 2. Retrieve selling price property.	The selling price property should be 800.0.	Pass
5	Test category property (Product class)	Initialize JavaFX platform by using JavaFXTestSetup class	1. Create Product object with category Electronics. 2. Retrieve category property.	The category property should be Electronics.	Pass
6	Test image property (Product class)	Initialize JavaFX platform by using JavaFXTestSetup class	1. Create Product object with an image. 2. Retrieve image property.	The image property should match the provided image.	Pass
7	Test loading data when the database is empty (ProductPage class)	Mock the database to return no results, initialize JavaFX platform by using JavaFXTestSetup class	1. Initialize JavaFX environment. 2. Mock DataStoreSql to return an empty result set. 3. Instantiate ProductPage. 4. Load product data into tableView.	TableView should be empty when there are no products in the database.	Pass
8	Test filtering with a non-existent category (ProductPage class)	Mock the database to return no results, make variable TableView protected instead of private, make filterProductsByCategory method protected, initialize JavaFX platform by using JavaFXTestSetup class	1. Initialize JavaFX environment. 2. Mock DataStoreSql to return an empty result set. 3. Instantiate ProductPage. 4. Filter products by a non-existent category.	The TableView should be empty as this category doesn't exist	Pass

9	Test adding a product without an image (handleSubmitButtonAction method in SellingPage class)	Mock DataStoreSql methods to control behavior, set private fields using reflection, initialize JavaFX platform by using JavaFXTestSetup class	1. Initialize JavaFX environment. 2. Mock DataStoreSql to control database interactions. 3. Instantiate SellingPage and set private fields using reflection. 4. Set text fields for item details but leave image null. 5. Call handleSubmitButtonAction method on SellingPage. 6. Wait for JavaFX operations to complete. 7. Verify that DataStoreSql.addProduct was not called with a null image. 8. Load product data into ProductPage. 9. Verify that no products are loaded into the TableView.	The product without an image should not be added to the database, and the TableView should remain empty.	Failed: Product without image was loaded into the table.
10	Test adding a product with negative selling price (handleSubmitButtonAction method in SellingPage class)	Mock DataStoreSql methods to control behavior, set private fields using reflection, initialize JavaFX platform by using JavaFXTestSetup class	1. Initialize JavaFX environment 2. Mock DataStoreSql to control database interactions. 3. Instantiate SellingPage and set private fields using reflection. 4. Set text fields for item details with negative selling price. 5. Call handleSubmitButtonAction method on SellingPage. 6. Wait for JavaFX operations to complete. 7. Load product data into ProductPage. 8. Verify that no products are loaded into the TableView.	The product with negative selling price should not be added to the database, and the TableView should remain empty.	Failed: Product with negative selling price was loaded into the table

11	Test adding a product with negative actual price (handleSubmitButtonAction method in SellingPage class)	Mock DataStoreSql methods to control behavior, set private fields using reflection, initialize JavaFX platform by using JavaFXTestSetup class	1. Initialize JavaFX environment. 2. Mock DataStoreSql to control database interactions. 3. Instantiate SellingPage and set private fields using reflection. 4. Set text fields for item details with negative actual price. 5. Call handleSubmitButtonAction method on SellingPage. 6. Wait for JavaFX operations to complete. 7. Load product data into ProductPage. 8. Verify that no products are loaded into the TableView.	The product with negative actual price should not be added to the database, and the TableView should remain empty.	Failed: Product with negative actual price was loaded into the table
12	Test adding a product with future year of purchase (handleSubmitButtonAction method in SellingPage class)	Mock DataStoreSql methods to control behavior, set private fields using reflection, initialize JavaFX platform by using JavaFXTestSetup class	1. Initialize JavaFX environment. 2. Mock DataStoreSql to control database interactions. 3. Instantiate SellingPage and set private fields using reflection. 4. Set text fields for item details with future year of purchase. 5. Call handleSubmitButtonAction method on SellingPage. 6. Wait for JavaFX operations to complete. 7. Load product data into ProductPage. 8. Verify that no products are loaded into the TableView..	The product with future year of purchase should not be added to the database, and the TableView should remain empty.	Failed: Product with future year of purchase was loaded into the table

13	Test different entries in Password and Re-Enter Password fields (NewUserForm class)	Add a method public static boolean createUser(String username, String password, String reEnterPassword) in NewUserForm class, initialize JavaFX environment, use JavaFXTestSetup class	Create new user with different entries in Password and Re-Enter Password fields.Check if user can be added.	Get false result, user will not be added in the database.	Pass
14	Test username is empty (NewUserForm class)	Add a method public static boolean createUser(String username, String password, String reEnterPassword) in NewUserForm class, initialize JavaFX environment, use JavaFXTestSetup class	Create new user without entries in Username field.Check if user can be added.	Get false result, user will not be added in the database.	Pass
15	Test password is empty (NewUserForm class)	Add a method public static boolean createUser(String username, String password, String reEnterPassword) in NewUserForm class, initialize JavaFX environment, use JavaFXTestSetup class	Create new user without entries in Passwords field.Check if user can be added.	Get false result, user will not be added in the database.	Pass

16	Test username includes only spaces (NewUserForm class)	Add a method public static boolean createUser(String username, String password, String reEnterPassword) in NewUserForm class, initialize JavaFX environment, use JavaFXTestSetup class	Create new user with spaces only in Username field.Check if user can be added.	Get false result, user will not be added in the database.	Failed: the result was true, so user was added in the database.
17	Test passwords include only spaces (NewUserForm class)	Add a method public static boolean createUser(String username, String password, String reEnterPassword) in NewUserForm class, initialize JavaFX environment, use JavaFXTestSetup class	Create new user with spaces only in Passwords fields.Check if user can be added.	Get false result, user will not be added in the database.	Failed: the result was true, so user was added in the database.