PolyShop OSS Test Cases

Group 41
COMP2411 Database Systems (Fall 2023)
Dominicus Dylan HARYOTO, Manish RAI, Sidharth SREEKUMAR, Shusen ZHENG, Muze XIANG, Syed Muhammad Askar Hussain ZAIDI

TEST 1: Testing the existence of a user

```
@Test
public void testUserExist() throws SQLException {
        //Test query user exist or not
        boolean queryResult = OSS.userExist("test1");
        assertTrue("User Exists", queryResult);
}
```

Tests whether a user with ID, test1, exists in the Online Shopping System as a user.

TEST 2: Tests to check password authentication of user accounts

```
@Test
public void testGetUserPwd() throws SQLException {
        // Test for user password
        String pwd = OSS.userPwd("test1");
        assertEquals("test", pwd);
}
```

Retrieves a user's (with ID test1) password and compares with the expected value

TEST 3: Tests to see existence of admin account

```
@Test
public void testAdminExist() throws SQLException {
        // Test successful product search
        boolean queryResult = OSS.adminExist("rai");
        assertTrue("Admin Exists", queryResult);
}
```

Asserts a check to verify the existence of an admin user with ID: rai

TEST 4: Tests to check password authentication of admin accounts

```
@Test
public void testSelectAdminPwd() throws SQLException {
        // Test query admin exist or not
        String pwd = OSS.selectAdminPwd("rai");
        assertEquals("rai", pwd);
}
```

Retrieves the password for admin user (ID:rai) and verifies with expected password.

TEST 5: Tests functionality of creating new users in the system

```
@Test
public void testCreateUser() throws SQLException {
        // Test create a user
        boolean addResult = OSS.createUser("testUser", "test","test","test",
            "2023-12-2","test@test.com","1234578","testing address");
        assertTrue("Added a user successfully", addResult);
}
```

Attempts to create a new user with ID: testUser. Attempt is a success

TEST 6: Tests functionality of creating new admins in the system

```
@Test
public void testCreateAdmin() throws SQLException {
        // Test create a admin
        boolean addResult = OSS.createAdmin("tester7", "test","test","test",
            "2023-12-02","testA@test.com","12345678");
        assertTrue("Added an admin successfully", addResult);
}
```

Attempts to create a new admin user with ID: tester7. Attempt is a success

TEST 7: Tests the Analysis Report Generation for a single criterion

```
@Test
public void testSelectReport() throws SQLException {
        // Test query report
        List<Map<String, Object>> resultList = OSS.selectReport("VIEWS", true);
        assertNotNull(resultList);
}
```

Create an analysis report of the products sorted by increasing order of most viewed.

TEST 8: Tests the implementation of our promotions using relation joins

```
@Test
public void testSelectReportUnion() throws SQLException {
        // Test query report union info
        List<Map<String, Object>> resultList = OSS.selectReportUnion();
        assertNotNull(resultList);
}
```

Creates a join between the product and promotion tables by joining using the promotionID attribute

TEST 9: Tests the implementation of our search and filtering conditions

```
@Test
public void testSelectProductBylike() throws SQLException {
        // Test query products by keyword
        List<Map<String, Object>> resultList = OSS.selectProductBylike("chicken");
        assertNotNull(resultList);
}
```

Attempts to select products who's name or description contains an inputted keyword using the LIKE operator

TEST 10: Tests the display of all products in the system

```
@Test
public void testSelectAllProduct() throws SQLException {
        // Test query all products
        List<Map<String, Object>> resultList = OSS.selectAllProduct();
        assertNotNull(resultList);
}
```

Selects all the products in the system and returns true if the selection works as intended

TEST 11: Tests the exact search feature of our database that enables search by productID

```
@Test
public void testSelectProductById() throws SQLException {
        // Test query product by id
        Map<String, Object> resultList = OSS.selectProductById("1");
        assertNull(resultList);
}
```

Attempts to select a product with ID: 1, which does not exist in our database and hence evaluates to false

TEST 12: Tests our product inventory feature

```
@Test
public void testSelectStockQtyById() throws SQLException {
        // Test query product stock
        int stock = OSS.selectStockQtyById("chicken");
        assertEquals(936, stock);
}
```

Selects the stock of a particular product, "chicken" within our database. Evaluates to true if the stock is 936. Will become false once a transaction is carried out in which the chicken is bought (as quantity will decrease)

TEST 13: Tests the implementation of our shopping cart addition

```
@Test
public void testAddOrUpdateCart() throws SQLException {
        // Test add or update to cart
        String operation = OSS.addOrUpdateCart(1, "test", "chicken");
        assertEquals("add", operation);
}
```

Attempts a transaction by user test who wants to add 1x chicken into the cart. It will evaluate to true of the attempt is successful

TEST 14: Tests the implementation of our shopping cart deletion

```
@Test
public void testdeleteFromCart() throws SQLException {
        // Test add or update to cart
        String testProduct = OSS.deleteFromCart( "test","chicken");
        assertEquals("chicken", testProduct);
}
```

Attempts to delete the item “chicken” from user “test” ‘s shopping cart. Evaluates to true if the attempt is successful

TEST 15: Tests the selection of a cart using the userID

```
@Test
public void testSelectCartByUserId() throws SQLException {
        // Test query cart by userId
        List<Map<String, Object>> resultList = OSS.selectCartByUserId( "test");
        assertNotNull(resultList);
}
```

Attempts to select the cart of a particular user from our user relation. This is essential for the checkout process and evaluates to true if successful

—---------- THE END —----------