@Test

public void testBuyProduct() {

// Create a new instance of the Product class

Product product = new Product("Laptop", "Dell Latitude", 750.00);

// Create a new instance of the User class

User user = new User("John Doe", "john@example.com", "password");

// Add the product to the database

ListingController listingController = new ListingController();

listingController.createListing(product, user);

// Create a new instance of the Cart class

Cart cart = new Cart(user);

// Add the product to the cart

cart.addProduct(product);

// Buy the product

cart.buyProducts();

// Verify that the product has been removed from the cart

assertTrue(cart.getProducts().isEmpty());

// Verify that the user's balance has been updated

assertEquals(250.00, user.getBalance(), 0.01);

// Verify that the product has been marked as sold

assertTrue(product.isSold());

}

@Test

public void testAskQuestion() {

// Create a new instance of the User class

User user = new User("John Doe", "john@example.com", "password");

// Create a new instance of the CustomerRep class

CustomerRep customerRep = new CustomerRep("Jane Smith", "jane@example.com");

// Create a new instance of the Question class

Question question = new Question("How can I return a product?", user);

// Ask the question to the customer representative

customerRep.answerQuestion(question, "You can initiate a return by clicking on the 'Return' button on the product page.");

// Verify that the question has been answered

assertEquals("You can initiate a return by clicking on the 'Return' button on the product page.", question.getAnswer());

}

@Test

public void testBrowseProducts() {

// Create a new instance of the Product class

Product product1 = new Product("T-Shirt", "A red T-shirt", 10.99, "clothing");

// Add the product to the database using the ListingController class

ListingController.addListing(product1);

// Create a new instance of the Product class

Product product2 = new Product("Coffee Mug", "A white ceramic coffee mug", 5.99, "home");

// Add the product to the database using the ListingController class

ListingController.addListing(product2);

// Retrieve all products from the database using the ListingController class

List<Product> products = ListingController.getAllListings();

// Verify that the correct number of products are returned

assertEquals(2, products.size());

// Verify that the correct products are returned

assertTrue(products.contains(product1));

assertTrue(products.contains(product2));

}

@Test

public void testRegisterUser() {

// Create a new instance of the User class with valid registration information

User newUser = new User("John", "Doe", "johndoe@example.com", "password");

// Register the user using the UserController class

boolean isRegistered = UserController.registerUser(newUser);

// Verify that the user is registered successfully

assertTrue(isRegistered);

// Attempt to register the same user again

boolean isRegisteredAgain = UserController.registerUser(newUser);

// Verify that the same user cannot be registered twice

assertFalse(isRegisteredAgain);

}

@Test

public void testCreateListingAndBuyItems() {

// create a new user account

User user = new User("JohnDoe", "password", "John", "Doe", "johndoe@email.com");

userRepository.save(user);

// create a new listing

Listing listing = new Listing("New Book", "This is a new book for sale", "books", 10.00, user);

listingRepository.save(listing);

// simulate a user buying the item

User buyer = new User("JaneDoe", "password", "Jane", "Doe", "janedoe@email.com");

userRepository.save(buyer);

// attempt to purchase the item

boolean purchaseSuccess = listing.purchase(buyer);

// check that the purchase was successful

assertTrue(purchaseSuccess);

// check that the item was removed from the listing after purchase

List<Listing> updatedListings = listingRepository.findByTitle("New Book");

assertTrue(updatedListings.isEmpty());

}

@Test

public void testViewCoursesAndTracker() {

// Create a new user and login

User user = new User("johnsmith", "password");

assertTrue(user.login());

// View all available courses and associated services

List<Course> courses = SC.getCourses();

assertNotNull(courses);

assertTrue(courses.size() > 0);

for (Course course : courses) {

assertNotNull(course.getTitle());

assertNotNull(course.getInstructor());

assertNotNull(course.getDepartment());

assertNotNull(course.getGroupMeLink());

assertNotNull(course.getRMPLink());

}

// Set up a course tracker for a closed course

Course closedCourse = courses.get(0); // assume first course is closed

assertTrue(closedCourse.isClosed());

// Add the course to the user's tracker

user.addCourseToTracker(closedCourse);

assertTrue(user.getTrackedCourses().contains(closedCourse));

}

import org.junit.Test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import static org.junit.Assert.assertEquals;

public class ViewLinksTest {

@Test

public void testViewGroupMeLink() {

// Set up the driver

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

WebDriver driver = new ChromeDriver();

// Navigate to the page with the GroupMe link

driver.get("https://example.com/course-info");

// Find the GroupMe link and click it

WebElement groupMeLink = driver.findElement(By.linkText("Join the GroupMe"));

groupMeLink.click();

// Verify that the GroupMe page is loaded

assertEquals("GroupMe | Course Name", driver.getTitle());

// Close the driver

driver.quit();

}

@Test

public void testViewRmpLink() {

// Set up the driver

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

WebDriver driver = new ChromeDriver();

// Navigate to the page with the RMP link

driver.get("https://example.com/course-info");

// Find the RMP link and click it

WebElement rmpLink = driver.findElement(By.linkText("View professor ratings on RMP"));

rmpLink.click();

// Verify that the RMP page is loaded

assertEquals("Professor Ratings | Course Name", driver.getTitle());

// Close the driver

driver.quit();

}

}

Python Selenium Functional Test Case Code:

**Register**

def test\_register\_account(self):

driver = self.driver

driver.get("http://localhost:8080/SWE-Amazon-2/register.jsp")

username = driver.find\_element\_by\_name("username")

email = driver.find\_element\_by\_name("email")

password = driver.find\_element\_by\_name("password")

confirm\_password = driver.find\_element\_by\_name("confirmPassword")

register\_button = driver.find\_element\_by\_xpath("/html/body/div/form/div[5]/input")

username.send\_keys("JohnDoe")

email.send\_keys("johndoe@example.com")

password.send\_keys("test\_password")

confirm\_password.send\_keys("test\_password")

register\_button.click()

success\_msg = WebDriverWait(driver, 10).until(

EC.visibility\_of\_element\_located((By.ID, "success-msg"))

)

self.assertEqual(success\_msg.text, "Account successfully created!")

**Sign In**

def test\_sign\_in(self):

driver = self.driver

driver.get("http://localhost:8080/SWE-Amazon-2/login.jsp")

username = driver.find\_element\_by\_name("username")

password = driver.find\_element\_by\_name("password")

login\_button = driver.find\_element\_by\_name("login")

username.send\_keys("JohnDoe")

password.send\_keys("test\_password")

login\_button.click()

welcome\_msg = WebDriverWait(driver, 10).until(

EC.visibility\_of\_element\_located((By.ID, "welcome-msg"))

)

self.assertIn("Welcome, JohnDoe", welcome\_msg.text)

**Create Listing**

def test\_create\_listing(self):

driver = self.driver

driver.find\_element\_by\_xpath("/html/body/nav/div/div/ul/li[1]/div/a[1]").click()

driver.find\_element\_by\_name("name").sendkeys(“item name”)

driver.find\_element\_by\_xpath("condition").select(1)

driver.find\_element\_by\_xpath("location").sendkeys(“item location”)

driver.find\_element\_by\_xpath("type").sendkeys(“item type”)

driver.find\_element\_by\_xpath("color").sendkeys(“item color”)

driver.find\_element\_by\_xpath("description").sendkeys(“item description”)

driver.find\_element\_by\_xpath("delivery").select(1)

driver.find\_element\_by\_xpath("price").sendkeys(“2500”)

driver.find\_element\_by\_xpath("photo").sendkeys(“/Users/afaqqamar/Downloads/image.jpg”)

driver.find\_element\_by\_xpath("listItem").click()

**Log Out**

def test\_sign\_in(self):

driver = self.driver

driver.find\_element\_by\_name("logout").click()

success\_msg = WebDriverWait(driver, 10).until(

EC.visibility\_of\_element\_located((By.ID, "login"))

)

self.assertEqual(success\_msg.text, "Signed Out")

**Buy Item**

def test\_buy\_listed\_item(self):

driver = self.driver

driver.get("http://localhost:8080/SWE-Amazon-2/login.jsp")

username = driver.find\_element\_by\_name("username")

password = driver.find\_element\_by\_name("password")

login\_button = driver.find\_element\_by\_name("login")

username.send\_keys("JohnDoe2")

password.send\_keys("test\_password")

login\_button.click()

Driver.find\_element\_by\_name

("/html/body/nav/div/div/ul/li[2]/div/a[1]").click()

driver.find\_element\_by\_id(‘item1’).click()

driver.find\_element\_by\_id(‘price’).send\_keys(“2500”)

driver.find\_element\_by\_id(‘submit’).click()

driver.find\_element\_by\_id(‘price’).send\_keys(“2500”)

driver.find\_element\_by\_id(‘street’).send\_keys(“street test”)

driver.find\_element\_by\_id(city).send\_keys(“city test”)

driver.find\_element\_by\_id(apartment).send\_keys(“apartment test”)

driver.find\_element\_by\_id(instructions).send\_keys(“instructions test”)

driver.find\_element\_by\_id(submit).click()