**Test cases of Gerkin language in Java**

**Test cases of Gui in Java:**

|  |
| --- |
| import org.junit.jupiter.api.Test;  import static org.junit.jupiter.api.Assertions.\*;  public class SystemLoginTest {  @Test  public class SystemUnderTest {  private static boolean systemOperational = true;  private static boolean userLoggedIn = false;  public static boolean login(String username, String password) {    if (!systemOperational) {  throw new IllegalStateException("System is not operational");  }  if (isValidCredentials(username, password)) {  userLoggedIn = true;  return true;  } else {  return false;  }  }  private static boolean isValidCredentials(String username, String password) {  return "user".equals(username) && "pass".equals(password);  }  public static boolean isUserLoggedIn() {  return userLoggedIn;  }  }  @Test  public class SystemUnderTest {  private static boolean userLoggedIn = true;  private static boolean browseDataSectionDisplayed = false;  public static void navigateToHomeScreen() {  browseDataSectionDisplayed = true;  }  public static boolean isBrowseDataSectionDisplayed() {  if (!userLoggedIn) {  throw new IllegalStateException("User is not logged in");  }  return browseDataSectionDisplayed;  }  }  @Test  public class UITest {  private static boolean systemLoaded = true;  public static void main(String[] args) {  if (!systemLoaded) {  throw new IllegalStateException("System is not loaded");  }  testUIResponsiveness();  }  public static void testUIResponsiveness() {  if (simulateUserInteractions()) {  System.out.println("UI is responsive");  } else {  System.out.println("UI is not responsive");  }  }  private static boolean simulateUserInteractions() {  return true;  }  }  @Test  public class BrowserCompatibilityTest {  private static boolean applicationRunning = true;  public static void main(String[] args) {  if (!applicationRunning) {  throw new IllegalStateException("Application is not running");  }  testCompatibilityOnBrowsers("Chrome", "Firefox", "Safari");  }  public static void testCompatibilityOnBrowsers(String... browsers) {  for (String browser : browsers) {  if (testCompatibilityWithBrowser(browser)) {  System.out.println("Compatibility test passed for " + browser);  } else {  System.out.println("Compatibility test failed for " + browser);  }  }  }  private static boolean testCompatibilityWithBrowser(String browser) {  return true;  }  }  @Test  public class AccessibilityTest {  private static boolean accessibilitySupported = true;  public static void main(String[] args) {  if (!accessibilitySupported) {  throw new IllegalStateException("Accessibility is not supported");  }  testAccessibilityFeaturesWithScreenReader();  }  public static void testAccessibilityFeaturesWithScreenReader() {  if (simulateScreenReaderInteractions()) {  System.out.println("Accessibility features function correctly");  } else {  System.out.println("Accessibility features do not function correctly");  }  }  private static boolean simulateScreenReaderInteractions() {  return true;  }  } |

**Test cases for API:**

|  |
| --- |
| import io.restassured.RestAssured;  import io.restassured.http.ContentType;  import io.restassured.response.Response;  import org.junit.Assert;  public class APITestMenu {  public static void main(String[] args) {  RestAssured.baseURI = "https://api.testsite.org";  String createUserPayload = "{\"username\":\"newUser\",\"email\":\"newUser@example.com\",\"password\":\"password123\"}";  Response createUserResponse = RestAssured.given()  .contentType(ContentType.JSON)  .body(createUserPayload)  .post("/api/users");  Assert.assertEquals("User is created successfully", 201, createUserResponse.getStatusCode());  int userId = 1;  Response getUserResponse = RestAssured.get("/api/users/" + userId);  Assert.assertEquals("User details are successfully retrieved", 200, getUserResponse.getStatusCode());  String updateUserPayload = "{\"email\":\"updatedEmail@example.com\"}";  Response updateUserResponse = RestAssured.given()  .contentType(ContentType.JSON)  .body(updateUserPayload)  .put("/api/users/" + userId);  Assert.assertEquals("User information is successfully updated", 200, updateUserResponse.getStatusCode());  Response deleteUserResponse = RestAssured.delete("/api/users/" + userId);  Assert.assertEquals("User is deleted successfully", 204, deleteUserResponse.getStatusCode());    String invalidUserPayload = "{\"username\":\"\",\"email\":\"invalidEmail\",\"password\":\"asadsabir227\"}";  Response createInvalidUserResponse = RestAssured.given()  .contentType(ContentType.JSON)  .body(invalidUserPayload)  .post("/api/users");  Assert.assertEquals("Error is returned for invalid user data", 400, createInvalidUserResponse.getStatusCode());  }  } |