SOFTWARE TESTING AND QUALITY ASSURANCE

EXPERIMENT 10

NAME: Shashwat Shah

SAP ID : 60004220126

BRANCH: COMPUTER ENGINEERING

C22

AIM: To study software Automation Testing tool WinRunner Creating Data-Driven Tests.

THEORY:

Data-Driven Testing (DDT) is a powerful automation testing methodology where the same test script is executed multiple times with different sets of input data. Instead of hardcoding test inputs into test scripts, the data is stored externally (in files like Excel, CSV, or databases), and the script reads from these data sources during execution.

This is especially helpful in repetitive testing scenarios, such as:

- Logging in with multiple usernames and passwords
- Form submissions with different input combinations
- Validating business logic across various edge cases

Manually testing applications for different data combinations is **tedious and error-prone**. Data-driven testing helps address these issues by automating the variation in data, allowing for a **wider coverage of test cases with minimal effort**.

Data-driven testing offers numerous benefits that enhance the efficiency, reliability, and maintainability of automated testing. Below is a detailed explanation of key advantages:

1. **Reusability**
   Data-driven testing promotes reusability as the same test script can be executed multiple times using different sets of test data. This eliminates the need to write separate test cases for each input scenario.

2. **Scalability**
   This approach is highly scalable because adding more test cases simply involves updating or extending the external data file (such as an Excel sheet), without needing to modify the test script itself.

3. **Accuracy**
   By automating the process of feeding different inputs, data-driven testing reduces the chances of human error. This results in more reliable and consistent test outcomes.

4. **Efficiency**
   It saves significant time and manual effort, especially when testing the same functionality repeatedly with various data combinations. Automation of repetitive tasks makes the entire testing process more efficient.

5. **Maintainability**
   Since the data is stored separately from the script, updating or changing test values becomes straightforward. This separation of logic and data enhances the maintainability of the test suite.

WinRunner supports **data-driven testing** using its **built-in Data Table** feature, or through external files such as Excel sheets. It uses **Test Script Language (TSL)** to read, iterate, and apply test data dynamically.

CODE:

1. **LOGIN PAGE**

```
<!DOCTYPE html>
<html>
<head><title>Login Page</title></head>
<body>
  <h1>Login Page</h1>
  <form>
    <label for="username">Username:</label>
    <input type="text" name="username" /><br><br>
    <label for="password">Password:</label>
    <input type="password" name="password" /><br><br>
    <input type="submit" id="submit-button" value="Login">
  </form>
</body>
</html>
```

2. **LOGIN DATA CSV FILE**

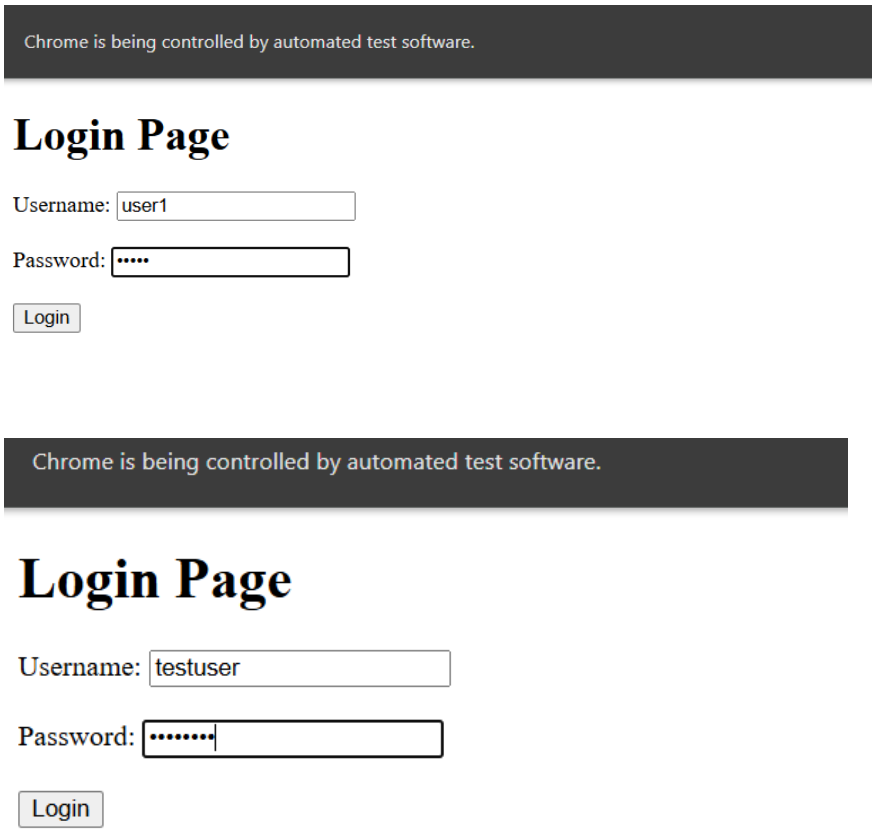
```
username,password
user1,pass1
user2,pass2
testuser,testpass
```

3. **DATA DRIVEN TEST:**

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
import time
```

```
import pandas as pd

# Load test data from CSV
data = pd.read_csv("D:\\hi\\STQA\\exp_10\\login_data.csv")

# Setup WebDriver
service = Service("D:\\hi\\STQA\\exp_10\\chromedriver-
win64\\chromedriver.exe")
driver = webdriver.Chrome(service=service)
driver.maximize_window()

# Run test for each set of credentials
for index, row in data.iterrows():
    driver.get("file:///D:/hi/STQA/exp_10/login.html")

    try:
        # Fill username
        username_field = driver.find_element(By.NAME, "username")
        username_field.clear()
        username_field.send_keys(row['username'])

        # Fill password
        password_field = driver.find_element(By.NAME, "password")
        password_field.clear()
        password_field.send_keys(row['password'])

        # Pause for screenshot
        print(f"Testing with Username: {row['username']}, Password:
{row['password']}")
        time.sleep(3)

        # Click submit
        submit_button = driver.find_element(By.ID, "submit-button")
        submit_button.click()

        # Wait and simulate a result message check
        time.sleep(3)
        print(f"Test case {index + 1} executed.\n")

    except Exception as e:
        print("Error during test execution:", e)

# Close browser after all tests
driver.quit()
```

OUTPUT:


Chrome is being controlled by automated test software.
Login Page
Username: user1
Password:
Login

Chrome is being controlled by automated test software.
Login Page
Username: testuser
Password:
Login