IRSTI 50.53.15

A. Talasbek [1]
[1]Suleyman Demirel University, Kaskelen, Kazakhstan

ARTIFICIAL AI IN TEST AUTOMATION: SOFTWARE TESTING OPPORTUNITIES WITH OPENAI TECHNOLOGY - CHATGPT

Abstract. One of the most important and significant stages of the software development life cycle is software testing. Automated testing reduces testing costs and increases productivity, resulting in a high-quality and stable end product. To ensure that software is bug-free and delivers the desired user experience, test automation is critical. As technology advances, test automation becomes more complex. However, advanced AI technologies will soon become commonplace thanks to powerful tools like ChatGPT that, among countless other things, can chat with you and teach you how to read and write code like a human. In this article, we will try to consider the possibilities of automation with chatgpt. Starting with writing test plans, and test scripts by using Python and Selenium WebDriver. How ChatGPT can improve our tasks and make them more feasible. By leveraging the capabilities of ChatGPT, software testing engineers can enhance their skills, improve testing capabilities, and achieve better quality and accuracy of test results. The opportunities presented by ChatGPT can lead to improved efficiency, productivity, and overall performance in the field of software testing engineering.
Keywords: Chatgpt, OpenAI, test automation, test plan, test case, test scripts, Selenium WebDriver

***

Аңдатпа. Бағдарламалық жасақтаманы әзірлеудің өмірлік циклінің маңызды және маңызды кезеңдерінің бірі бағдарламалық жасақтаманы тестілеу болып табылады. Автоматтандырылған тестілеу сынақ шығындарын азайтады және өнімділікті арттырады, нәтижесінде жоғары сапалы және тұрақты соңғы өнім алынады. Технология дамыған сайын сынақты автоматтандыру күрделене түседі. Дегенмен, озық AI технологиялары ChatGPT сияқты қуатты құралдардың арқасында көп ұзамай қарапайым болады, олар сансыз басқа нәрселермен қатар сізбен сөйлесе алады және адам сияқты кодты оқу және жазуды үйретеді. Бұл мақалада біз ChatGPT көмегімен автоматтандыру мүмкіндіктерін қарастыруға тырысамыз. Python және Selenium WebDriver көмегімен сынақ жоспарларын, сынақ жағдайларын және сынақ сценарийлерін жазудан бастаңыз. ChatGPT мүмкіндігін пайдалана отырып, бағдарламалық жасақтаманың сынақ инженерлері өз дағдыларын жақсарта алады, тестілеу мүмкіндіктерін жақсарта алады және сынақ нәтижелерінің жоғары сапасы

мен дәлдігіне қол жеткізе алады. ChatGPT ұсынатын мүмкіндіктер бағдарламалық жасақтаманы сынау дизайны саласындағы тиімділікті, өнімділікті және жалпы өнімділікті арттыруға әкелуі мүмкін.
Түйінді сөздер: ChatGPT, OpenAI, тестілеуді автоматтандыру, сынақ жоспары, сынақ жағдайы, сынақ сценарийлері, Selenium WebDriver.

***

Аннотация: Одним из наиболее важных и значимых этапов жизненного цикла разработки программного обеспечения является тестирование программного обеспечения. Автоматизированное тестирование снижает затраты на тестирование и повышает производительность, в результате чего получается высококачественный и стабильный конечный продукт. По мере развития технологий автоматизация тестирования становится все более сложной. Однако передовые технологии искусственного интеллекта скоро станут обычным явлением благодаря мощным инструментам, таким как ChatGPT, которые, помимо бесчисленного множества других вещей, могут общаться с вами и учить вас читать и писать код, как человек. В этой статье мы постараемся рассмотреть возможности автоматизации с помощью ChatGPT. Начиная с написания планов тестирования, тест кэйсов и тестовых сценариев с использованием Python и Selenium WebDriver. Используя возможности ChatGPT, инженеры по тестированию программного обеспечения могут повысить свои навыки, улучшить возможности тестирования и добиться более высокого качества и точности результатов тестирования. Возможности, предоставляемые ChatGPT, могут привести к повышению эффективности, производительности и общей производительности в области проектирования тестирования программного обеспечения.
Ключевые слова: ChatGPT, OpenAI, автоматизация тестирования, план тестирования, тестовый пример, тестовые скрипты, Selenium WebDriver.

I. Introduction

Software testing is the first and most important step in software development. By providing certain inputs and comparing them to the desired output, testing evaluates the system or module to find and correct any inconsistencies between the desired and actual outputs. We can conditionally distinguish two forms of software testing. These are software testing automation and manual testing. Without the use of software automation tools or test scripts, software or applications are tested manually in a manual testing process. Software or applications are manually tested to find issues and ensure they meet standards. This method is known as manual estimation. The tester must act as an associated user and use all the options available in the program to ensure proper behavior.

Before automated testing, any new application must first undergo manual testing. While manual testing requires more work, it's important to determine if automation is worthwhile. Manual testing is important and essential because automated testing is impossible. Automated testing is more efficient and takes less time than human testing. More tests can be run faster, and tests can be reused across different versions of the application. There are a number of things to consider when choosing a testing tool. It is easy to implement and compatible with application testing, maintenance, and design. Most problems with manual testing are solved with automated testing. The goal of automated testing is to automate the testing process as much as possible using the least number of scenarios. Tests can be run with automated testing tools, which can also provide results and compare them to past test runs. Although automation does not completely replace manual testing, it continues it and aims to improve the accuracy and speed of testing.

There are several stages of software testing, such as creating test plans, test scenarios, test cases, and test scripts with different scripts. Modern language model called ChatGPT was created by OpenAI. Artificial intelligence that is based on natural language processing enables the automation of jobs requiring human-like interactions, such as those carried out by an AI chatbot or virtual assistant. With the aid of the web application ChatGPT, users may communicate naturally and in-depth with OpenAI's AI. It can comprehend natural language input, respond to it, and remember previous discussions to give more precise and pertinent responses. Text in a variety of formats, including structured data, code snippets, and annotations, can be produced using ChatGPT and used to automate processes like code development and testing. Testing ChatGPT may have a lot of ways for testing approaches.
Unit Testing: For front-end applications created in a variety of languages, including Javascript, Python, Java, Swift, etc., unit testing is recommended. To create useful syntactic references for writing unit tests correctly, utilize ChatGPT.

Integration: With the aid of a framework like Selenium, ChatGPT can be used to create test scripts for integration testing. Front-end web applications created with Javascript, HTML, and CSS may be tested with this.

API Testing: Using a framework like Rest Assured or Postman, ChatGPT may be used to produce test scripts for API testing. Backend APIs created using languages like Java, Python, NodeJS, etc. can be tested using this.

Mobile Testing: Using a framework like Appium, ChatGPT may be used to create test scripts for mobile testing. This may be used to test mobile applications created using languages like Swift and Java, among others.

End-to-end testing: Using a framework like Selenium, Puppeteer, or another, ChatGPT can be used to create test scripts for end-to-end testing. This can be used to test full web apps that were created using Javascript, HTML, and CSS.

First, let's get acquainted with the phases of testing, in Figure 1 you can show the test automation lifecycle. Let’s discuss and develop it by chatGPT.

II. Methodology

The methodological part of the research paper we implement our software testing life cycle step by step by ChatGPT. As I said before, there are several stages in software testing, first is detecting which tests need to be automated. After defining which are for automation testing, the next stage is selection tool and customization. The next step will be writing test scripts and a testing process using these automation scripts.

2.1. Detecting which tests need to be automated.

The process of test automation is comparable to that of software development. The work required to automate a test is comparable. It follows the same cycle as a software development product's development process. The plan is created by taking into account the amount of time needed, the overall quantity of resources required, and who will be doing it.

Automation test mostly selected by this criterias:

- Repetitive testing carried out across several releases
- Tests with a high risk of human error
- Tests that demand numerous data sets
- Common characteristics that lead to dangerous situations
- Tests that can't be performed by hand
- Tests that are carried out using various hardware and software combinations and platforms.
- Tests that are labor-intensive and time-consuming to run manually

2.2.Selection tool and customization

A portable software testing tool for automation testing is Selenium. It is a framework made up of numerous web application testing tools. With Selenium, you can create tests without having to learn a test scripting language using a record/playback tool (Selenium IDE). In order to construct test cases in a variety of well-known programming languages, such as C#, Java, Groovy, Perl, PHP, Python, and Ruby, it has a test domain-specific language (Selenese). The majority of contemporary web browsers may then be used to run all these written test cases. Any operating system, including Windows, Linux, and Macintosh, may execute Selenium.

Selenium WebDriver is a cross-browser testing tool to check how websites render in different browsers.It is part of the Selenium automated testing suite. There are three components in Selenium: Selenium WebDriver, Selenium IDE and Selenium Grid. Testers use Selenium to automate the actions of browsers, check the functionality of the program and receive data from sites. Selenium tools have different approaches to test automation. WebDriver is needed to control the browser, both local and remote. It is a flexible tool: it easily integrates

with test frameworks and other tools. Based on it, you can create narrowly focused tools for cross-browser testing.

2.3. Writing test plan

For this phase to test plan , we need to create objectives, test environments, test cases. A test case is a document that contains the testing steps to perform a procedure, set of test inputs, conditions for their execution, and expected results that are created with a specific goal in mind, such as checking a certain program route or ensuring that a given input will produce the desired result. While a test case should not follow a certain format, it must include input, expected behavior, and expected output.

2.4. Automated test scripts

Step 1:

Selenium supports several programming languages - Java, PHP, Python, JavaScript, C#. Python is used in this example of automation. Go to the illustration web application. Installing or updating the Google Chrome driver comes first. The Chrome browser window is then maximized to take up the entire window. The website whose user interface will be tested in this example is "https://www.bstackdemo.com," which we finally navigate to.

Step 2:

The sign-in button element is located using the ID locator, which is then clicked. The textbox for entering the username is then located using the XPath locator. The textbox for the username is made the driver's active element, and the letters "demouser" are typed into it.

Once more, the textbox for entering the password is located using the XPath locator. The textbox for the password is made the driver's active element, and the letters "testingisfun99" are typed into it. To make the software wait until the login and password text fields are clicked, explicit wait is once more used with an expected condition. The ID locator is then used to locate the login button element and click it.

III. Results

Test Planning with ChatGpt.

Consider that you wish to plan the test automation for a website called bstackdemo.com that you have developed some features for. In these situations, you can use GPT by providing it with information about the functional components and how they are used, then asking it to produce a test plan. Prompt: Prepare a test plan with objectives, test environments, test case tables to enter Jira tickets,test data, test execution and test conclusion with following validations on website:

- Open browserstack.com. Success if page loads with title ‘StackDemo’
- Click the sign in button. Success if page redirects to ‘/signin?checkout=true’
- Use username 'userdemo' and password 'testingisfun99'. Success if fields are filled out.

- Click to login. Success if a user logged into the website.

ChatGPT:

data, test execution and test conclusion

Test Plan for Automation Testing of bstackdemo.com:

Objectives:

- To ensure that the website functions as intended and meets the requirements of the business and users.
- To identify and report defects or issues that affect the usability, performance, and security of the website.
- To improve the quality and reliability of the website by implementing automated tests that cover the critical and high-risk areas of the application.

Test Environment:

- Operating System: Windows 10
- Browser: Chrome, Firefox, Safari, Edge
- Test Framework: Selenium WebDriver
- Programming Language: Java
- Test Management Tool: Jira

Test Case Table:

Test Case ID	Test Case Description	Test Data	Expected Result
TC01	Verify Page Title	N/A	The page title should be 'StackDemo'.
TC02	Verify Sign In Page	N/A	The current URL should be 'bstackdemo.com/signin?checkout=true'.
TC03	Verify User Login	username: userdemo, password: testingisfun99	The username and password fields should be filled out with the correct values.
TC04	Verify Successful Login	username: userdemo, password: testingisfun99	The user should be logged into the website successfully.

Test Data:

- Valid username: 'userdemo'
- Valid password: 'testingisfun99'

Test Execution:

1. Set up the test environment by installing the required software, libraries, and drivers.
2. Create test cases in Jira and link them to the corresponding requirements or user stories.
3. Create test scripts using Selenium WebDriver and Java.
4. Execute the test cases in different browsers and report any defects or issues found during testing.
5. Repeat the test cycle until all the critical and high-risk areas of the website are covered and the defects are fixed or mitigated.
6. Generate test reports and share them with the development team, project stakeholders, and quality assurance team.

Test Conclusion:

The test plan for automation testing of bstackdemo.com covers the critical and high-risk areas of the website and ensures that the website functions as intended and meets the requirements of the business and users. By implementing automated tests, the quality and reliability of the website can be improved, and defects or issues can be identified and fixed in a timely and efficient manner. The test reports generated during testing provide valuable insights and feedback that can be used to enhance the website and improve the user experience.

Web testing with Selenium Webdriver

You may now utilize ChatGPT to create test automation code using the test plans that were previously generated.

Write a BrowserStack Automate script in Python with Selenium to

1. Open the Bstackdemo.com website in the Chrome browser.
2. Verify that the title of the webpage is “StackDemo”.
3. Verify that the homepage loads correctly with all elements displayed as expected.
4. Click to id = 'signin'. Success if page redirects to ‘/signin?checkout=true’
5. Fill out the field id 'username' as 'userdemo'
6. Fill out the field id 'password' as 'testingisfun99'
7. Click to button with id = 'login'
8. Verify that user is logged into system with username 'userdemo'

ChatGPT:

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities

# Set BrowserStack credentials and desired capabilities
bs_username = 'your_username'
bs_access_key = 'your_access_key'
bs_capability = DesiredCapabilities.CHROME.copy()
bs_capability['os'] = 'Windows'
bs_capability['os_version'] = '10'
bs_capability['browser'] = 'Chrome'
bs_capability['browser_version'] = 'latest'

# Initialize the WebDriver with BrowserStack capabilities
driver = webdriver.Remote(
    command_executor=f'http://{bs_username}:{bs_access_key}@hub-cloud.browserstack.
    desired_capabilities=bs_capability
)

# Navigate to Bstackdemo.com and verify the title
driver.get("https://bstackdemo.com")
assert "StackDemo" in driver.title

# Verify that the homepage loads correctly
expected_elements = ['signin', 'features', 'pricing', 'support']
for element_id in expected_elements:
    assert driver.find_element_by_id(element_id).is_displayed()
```

```
# Click the Sign In button and verify the redirect URL
sign_in_button = driver.find_element_by_id('signin')
sign_in_button.click()
assert "/signin?checkout=true" in driver.current_url

# Fill out the login form and submit it
username_field = driver.find_element_by_id('username')
password_field = driver.find_element_by_id('password')
login_button = driver.find_element_by_id('login')
username_field.send_keys('userdemo')
password_field.send_keys('testingisfun99')
login_button.click()

# Verify that the user is logged in
assert "Welcome, userdemo" in driver.page_source

# Quit the WebDriver session
driver.quit()
```

This script uses BrowserStack credentials to initialize a remote WebDriver session with Chrome on Windows 10. It then navigates to the Bstackdemo.com website, verifies the title and that the homepage loads correctly, clicks the Sign In button and fills out the login form with the specified credentials, and finally verifies that the user is logged in with the expected username.

IV. Conclusion

ChatGPT, a large language model trained by OpenAI, is a complex system that requires a comprehensive software testing approach to ensure its quality, reliability, and usability. Testing provides an opportunity to identify and fix defects or issues that may affect the performance or functionality of the system, and to validate that the system meets the requirements and expectations of the users. To test ChatGPT effectively, a test plan must be developed that outlines the objectives, scope, and approach of the testing process. The test plan should define the test environments, tools, and resources required for testing, and should identify the test scenarios and test cases to be executed. The test plan should also establish the test data and test schedules, and should define the roles and responsibilities of the testing team. The test case design is another critical aspect of software testing in ChatGPT. Test cases should be designed to cover all the functional and non-functional requirements of the system, and should be written in a clear, concise, and unambiguous manner. Each test case should specify the preconditions, input data, expected results, and post-conditions, and should be linked to the corresponding requirements or user stories. Automated testing is an essential component of software testing in ChatGPT, as it allows for the efficient and effective execution of test cases, reduces the time and cost of testing, and enhances the reliability and repeatability of the testing process. Automated test scripts can be developed using a variety of tools and frameworks, such as Selenium WebDriver and Java. The test scripts should be designed to cover the critical and high-risk areas of the system, and should be executed in different test environments and configurations.

In conclusion, software testing plays a critical role in ensuring the quality, reliability, and usability of ChatGPT. A comprehensive software testing approach should be developed that includes a test plan, test case design, and automated test scripts. Testing provides an opportunity to identify and fix defects or issues that may affect the performance or functionality of the system, and to validate that the system meets the requirements and expectations of the users. By implementing an effective and efficient software testing strategy, ChatGPT can continue to deliver high-quality and reliable services to its users and maintain its reputation as a leading language model.

References:

1 Deepthi Wilson. R PG Student Dept.Computer Science & Engineering GSSSIETW Mysuru, India "A Comprehensive Review on Selenium Automation Testing Tool", vol. 5, no. 2, 2017, pp. 1-4.

2 Vishwajyoti and Sachin Sharma "STUDY AND ANALYSIS OF AUTOMATION TESTING TECHNIQUES" Deptt of Computer Applications, Manav Rachna International University, Faridabad, vol. 3, no. 12, 2012, pp. 36-43.

3 Mubarak Albarka Umar "A Study of Automated Software Testing: Automation Tools and Frameworks" , Vol. 8 No.06 Nov-Dec 2019, pp. 217-224.

4 Dudekula Mohammad Rafi; Katam Reddy Kiran Moses; Kai Petersen; Mika V. Mäntylä "Benefits and limitations of automated software testing: Systematic literature review and practitioner survey", 2012, pp. 10-15

5 Dudekula Mohammad Rafi Katam Reddy Kiran Moses "Automated Software Testing: A Study of State of Practice",2017, 3-5.

6 Divya Kumara, K. K. Mishrab, "The Impacts of TestAutomation on Software's Cost, Quality and Time to Market",7th International Conference on Communication, Computing and Virtualization 2016 (Procedia Computer Science 79 (2016) 8 – 15)