CMR INSTITUTE OF TECHNOLOGY

(UGC AUTONOMUS)

(Approved by AICTE, Affiliated to JNTU, Kukatpally, Hyderabad)

A Mini Project On

Automated Testing for Employee Salary Threshold

Done by : Nandikasi Hemanth Kumar

(21R01A05P5)

CONTENTS

Title : Automated Testing for Employee Salary Threshold

Introduction :

In the realm of software development, ensuring the correctness and reliability of applications is paramount. One critical aspect of testing is verifying that functionalities related to data processing, such as filtering and querying, operate accurately. In this mini project, we focus on automating the testing process for determining a list of employees whose salaries exceed a specified threshold.

Traditional manual testing methods can be time-consuming and error-prone, especially when dealing with large datasets. Automation offers a systematic approach to efficiently validate these functionalities, saving time and reducing the risk of human error. By automating the testing of salary threshold checks, software teams can ensure that the application accurately identifies employees meeting certain salary criteria, thereby enhancing the overall quality and reliability of the software.

Abstract :

The objective of this mini project is to develop an automated testing framework for verifying the functionality of determining employees whose salaries surpass a predefined threshold. The automation process involves creating test cases that simulate various scenarios, such as different salary thresholds, empty employee lists, and boundary cases. These test cases will be executed automatically using testing frameworks like Selenium or Appium, coupled with scripting languages such as Python or Java.

The automation script will interact with the application's user interface or API endpoints to input different salary thresholds and validate the resulting list of employees against expected outcomes. Additionally, assertions will be implemented to ensure the correctness of the retrieved data. The testing framework will generate detailed reports highlighting passed and failed test cases, aiding in identifying and resolving any issues promptly.

By implementing automated testing for salary threshold functionality, software development teams can streamline the testing process, increase test coverage, and enhance the reliability of their applications. This mini project serves as a practical demonstration of the benefits of automation in ensuring the accuracy and robustness of data processing functionalities within software systems.

REQUIREMENTS :

Recommended Operating Systems

• Windows: 11

• MAC: OS X v10.7 or higher

• Linux: Ubuntu

Hardware Requirements :

• Processor: Minimum 1 GHz; Recommended 2GHz or more

• Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)

• Hard Drive: Minimum 32 GB; Recommended 64 GB or more

• Memory (RAM): Minimum 1 GB; Recommended 4 GB or above

• Sound card w/speakers

• Some classes require a camera and microphone

Software Requirements :

Supported Browsers

• Internet Explorer • Safari

• Firefox • Chrome

Tools Required

• Install Java JDK • Install Eclipse

• Install Selenium Webdriver Files

Source Code :

```
import java.util.*;
import java.util.stream.Collectors;

public class Program2 {
   public static void main(String[] args) {
      List<Employee> employees = Arrays.asList(
            new Employee("Emp1", "CS", 10000),
            new Employee("Emp2", "CS", 20000),
            new Employee("Emp3", "CS", 15000),
            new Employee("Emp4", "IT", 30000),
            new Employee("Emp5", "IT", 18000),
            new Employee("Emp6", "IT", 40000)
      );

      System.out.println("All employees:");
      System.out.printf("%-5s%-10s%-15s%-5s%n", "S.no", "Employee", "Department",
                        "Salary");
      int count = 1;
      for (Employee emp : employees) {
         System.out.printf("%-7s%-12s%-11s%-15s%n", count++, emp.getName(),
                           emp.getDepartment(), emp.getSalary());
      }
      System.out.println();

      Map<String, List<Employee>> bySalary = employees.stream().filter(e -> e.getSalary()
      > 15000).collect(Collectors.groupingBy(Employee::getDepartment));

      System.out.println("Salary greater than 15,000 are");
```

```
        System.out.printf("%-5s%-10s%-15s%-5s%n", "S.no", "Employee", "Department",
                                    "Salary");
        count = 1;
        for (Map.Entry<String, List<Employee>> entry : bySalary.entrySet()) {
            for (Employee emp : entry.getValue()) {
                System.out.printf("%-7s%-12s%-11s%-15s%n", count++, emp.getName(),
                                        emp.getDepartment(), emp.getSalary());
            }
        }
    }
}
class Employee {
    private String name;
    private String department;
    private int salary;
    public Employee(String name, String department, int salary) {
        super();
        this.name = name;
        this.department = department;
        this.salary = salary;
    }
    public String getName() {
        return name;    }
    public String getDepartment() {
        return department;    }
    public int getSalary() {
        return salary;    }
    public void setSalary(int salary) {
        this.salary = salary;    }
}
```

Results :

```
All employees:
S.no Employee  Department     Salary
1       Emp1         CS          10000
2       Emp2         CS          20000
3       Emp3         CS          15000
4       Emp4         IT          30000
5       Emp5         IT          18000
6       Emp6         IT          40000

Salary greater than 15,000 are
S.no Employee  Department     Salary
1       Emp2         CS          20000
2       Emp4         IT          30000
3       Emp5         IT          18000
4       Emp6         IT          40000
```

Conclusion :

In summary, automated testing for determining employee salaries exceeding a specific threshold offers a robust solution for ensuring software accuracy and reliability. By leveraging automation, we streamline testing processes, reduce manual effort, and enhance overall software quality. Embracing automation in testing represents a proactive step towards delivering high-quality software solutions efficiently and effectively.