Task 8.1P – Stem Math Game

Test Case:

Q1: Addition Test Cases

Test Case	**Description**	**Input**			**Expected Result**
testAdditionCorrectAnswer	Valid addition (Correct Answer)	10	5	15	Correct answer
testAdditionIncorrectAnswer	Invalid addition (Incorrect Answer)	10	5	16	Wrong answer
testAdditionInvalidInput	Invalid inputs for addition	abc	xyz	result	Invalid input
testAdditionEmptyInput	Empty input for addition	“”	“”	“”	Fields cannot be empty.

Q2: Subtraction Test Cases

Test Case	**Description**	**Input**			**Expected Result**
testSubtractionCorrectAnswer	Valid subtraction (Correct Answer)	20	8	12	Correct answer
testSubtractionIncorrectAnswer	Invalid subtraction (Incorrect Answer)	20	8	13	Wrong answer

testSubtractionInvalidInput	Invalid inputs for subtraction	twenty	eight	twelve	Invalid input
testSubtractionEmptyInput	Empty input for subtraction	“”	“”	“”	Fields cannot be empty.

Q3: Multiplication Test Cases

Test Case	**Description**	**Input**			**Expected Result**
testMultiplicationCorrectAnswer	Valid multiplication (Correct Answer)	3	4	12	Correct answer
testMultiplicationIncorrectAnswer	Invalid multiplication (Incorrect Answer)	3	4	16	Wrong answer
testMultiplicationInvalidInput	Invalid inputs for multiplication	three	four	twelve	Invalid input
testMultiplicationEmptyInput	Empty input for multiplication	“”	“”	“”	Fields cannot be empty.

Code:

RoutingServlet.java–

```
package web.handler;
import javax.servlet.http.HttpServletRequest;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
import org.springframework.web.servlet.view.RedirectView;
import web.service.LoginService;
import web.service.MathQuestionService;
@Controller
@RequestMapping("/")
```

```java
public class RoutingServlet {
        @GetMapping("/")
        public String welcome() {
                return "view-welcome";
        }
        @GetMapping("/login")
        public String loginView() {
                return "view-login";
        }
        @PostMapping("/login")
        public RedirectView login(HttpServletRequest request, RedirectAttributes redirectAttributes) {
                String username = request.getParameter("username");
                String password = request.getParameter("passwd");
                String dob = request.getParameter("dob");
                RedirectView redirectView;
                if (LoginService.login(username, password, dob)) {
                        redirectView = new RedirectView("/q1", true);
                } else {
                        redirectView = new RedirectView("/login", true);
                        redirectAttributes.addFlashAttribute("message", "Incorrect credentials.");
                }
                return redirectView;
        }
        @GetMapping("/q1")
        public String q1View() {
                return "view-q1";
        }
        @PostMapping("/q1")
        public RedirectView q1(HttpServletRequest request, RedirectAttributes redirectAttributes) {
                String number1 = request.getParameter("number1");
                String number2 = request.getParameter("number2");
                String resultUser = request.getParameter("result");
                Double calculatedResult = MathQuestionService.q1Addition(number1, number2);
                RedirectView redirectView;
                if (calculatedResult != null && calculatedResult.equals(Double.valueOf(resultUser))) {
                        redirectView = new RedirectView("/q2", true);
                } else {
                        redirectView = new RedirectView("/q1", true);
                        redirectAttributes.addFlashAttribute("message", "Wrong answer or invalid input. Try again.");
                }
                return redirectView;
        }
        @GetMapping("/q2")
        public String q2View() {
                return "view-q2";
        }
        @PostMapping("/q2")
        public RedirectView q2(HttpServletRequest request, RedirectAttributes redirectAttributes) {
                String number1 = request.getParameter("number1");
                String number2 = request.getParameter("number2");
                String resultUser = request.getParameter("result");
                Double calculatedResult = MathQuestionService.q2Subtraction(number1, number2);
                RedirectView redirectView;
                if (calculatedResult != null && calculatedResult.equals(Double.valueOf(resultUser))) {
                        redirectView = new RedirectView("/q3", true);
                } else {
                        redirectView = new RedirectView("/q2", true);
                        redirectAttributes.addFlashAttribute("message", "Wrong answer or invalid input. Try again.");
                }
                return redirectView;
        }
        @GetMapping("/q3")
        public String q3View() {
                return "view-q3";
        }
        @PostMapping("/q3")
        public RedirectView q3(HttpServletRequest request, RedirectAttributes redirectAttributes) {
                String number1 = request.getParameter("number1");
                String number2 = request.getParameter("number2");
                String resultUser = request.getParameter("result");
                Double calculatedResult = MathQuestionService.q3Multiplication(number1, number2);
                RedirectView redirectView;
                if (calculatedResult != null && calculatedResult.equals(Double.valueOf(resultUser))) {
                        redirectView = new RedirectView("/", true);
                        redirectAttributes.addFlashAttribute("message", "Congratulations! You completed the quiz.");
                } else {
                        redirectView = new RedirectView("/q3", true);
                        redirectAttributes.addFlashAttribute("message", "Wrong answer or invalid input. Try again.");
                }
                return redirectView;
        }
}
```

MathQuestionService.java

```java
package web.service;
```

```java
public class MathQuestionService {
		/**
		 * Safe addition for Q1.
		 */
		public static Double q1Addition(String number1, String number2) {
				try {
						double result = Double.valueOf(number1) + Double.valueOf(number2);
						return result;
				} catch (Exception e) {
						return null;
				}
		}
		/**
		 * Safe subtraction for Q2.
		 */
		public static Double q2Subtraction(String number1, String number2) {
				try {
						double result = Double.valueOf(number1) - Double.valueOf(number2);
						return result;
				} catch (Exception e) {
						return null;
				}
		}
		/**
		 * Multiplication for Q3.
		 */
		public static Double q3Multiplication(String number1, String number2) {
				try {
						double result = Double.valueOf(number1) * Double.valueOf(number2);
						return result;
				} catch (Exception e) {
						return null;
				}
		}
}
```

Unit Test(TestMathQuestionService.java)

```java
package web.service;
import org.junit.Assert;
import org.junit.Test;
public class TestMathQuestionService {
		@Test
		public void testTrueAdd() {
				Assert.assertEquals(3, MathQuestionService.q1Addition("1", "2"), 0);
		}
		@Test
		public void testAddNumber1Empty() {
				Assert.assertNull(MathQuestionService.q1Addition("", "2"));
		}
		@Test
		public void testAddInvalidInput() {
				Assert.assertNull(MathQuestionService.q1Addition("abc", "2"));
		}
		@Test
		public void testTrueSub() {
				Assert.assertEquals(2, MathQuestionService.q2Subtraction("5", "3"), 0);
		}
		@Test
		public void testSubInvalidInput() {
				Assert.assertNull(MathQuestionService.q2Subtraction("abc", "3"));
		}
		@Test
		public void testTrueMul() {
				Assert.assertEquals(6, MathQuestionService.q3Multiplication("2", "3"), 0);
		}
		@Test
		public void testMulInvalidInput() {
				Assert.assertNull(MathQuestionService.q3Multiplication("2", ""));
		}
}
```

Functional Testing(TestMathGameFunction.java)

```java
package web.service;
import org.junit.Assert;
import org.junit.Before;
import org.junit.After;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.support.ui.ExpectedConditions;
public class TestMathGameFunctional {
```

```java
private WebDriver driver;
private WebDriverWait wait;
@Before
public void setUp() {
    System.setProperty("webdriver.chrome.driver", "/Users/gagancheema/Documents/T12025/SIT333/chromedriver-mac-x64/chromedriver");
    driver = new ChromeDriver();
    wait = new WebDriverWait(driver, 10);
}
@After
public void tearDown() {
    if (driver != null) {
        driver.quit();
    }
}
private void login(String username, String password, String dob) {
    driver.get("http://localhost:8080/");
    driver.findElement(By.linkText("Login")).click();
    wait.until(ExpectedConditions.presenceOfElementLocated(By.id("loginBtn")));
    driver.findElement(By.id("username")).sendKeys(username);
    driver.findElement(By.id("passwd")).sendKeys(password);
    driver.findElement(By.id("dob")).sendKeys(dob);
    driver.findElement(By.id("loginBtn")).click();
    wait.until(ExpectedConditions.presenceOfElementLocated(By.id("number1")));
}
private void solveMathQuestion(String num1, String num2, String result) {
    driver.findElement(By.id("number1")).clear();
    driver.findElement(By.id("number2")).clear();
    driver.findElement(By.id("result")).clear();
    driver.findElement(By.id("number1")).sendKeys(num1);
    driver.findElement(By.id("number2")).sendKeys(num2);
    driver.findElement(By.id("result")).sendKeys(result);
    driver.findElement(By.cssSelector("input[type='submit']")).click();
}
private boolean pageContainsText(String text) {
    return driver.getPageSource().contains(text);
}
// 1. Addition Correct Answer
@Test
public void testAdditionCorrectAnswer() {
    login("gagan", "cheema22", "2000-01-01");
    solveMathQuestion("10", "5", "15");
    Assert.assertTrue(pageContainsText("Correct answer!"));
}
// 2. Addition Incorrect Answer
@Test
public void testAdditionIncorrectAnswer() {
    login("gagan", "cheema22", "2000-01-01");
    solveMathQuestion("10", "5", "16");
    Assert.assertTrue(pageContainsText("Wrong answer."));
}
// 3. Addition Invalid Input
@Test
public void testAdditionInvalidInput() {
    login("gagan", "cheema22", "2000-01-01");
    solveMathQuestion("abc", "xyz", "result");
    Assert.assertTrue(pageContainsText("Invalid input."));
}
// 4. Addition Empty Input
@Test
public void testAdditionEmptyInput() {
    login("gagan", "cheema22", "2000-01-01");
    solveMathQuestion("", "", "");
    Assert.assertTrue(pageContainsText("Fields cannot be empty."));
}
// 5. Subtraction Correct Answer
@Test
public void testSubtractionCorrectAnswer() {
    login("gagan", "cheema22", "2000-01-01");
    solveMathQuestion("20", "8", "12");
    Assert.assertTrue(pageContainsText("Correct answer!"));
}
// 6. Subtraction Incorrect Answer
@Test
public void testSubtractionIncorrectAnswer() {
    login("gagan", "cheema22", "2000-01-01");
    solveMathQuestion("20", "8", "13");
    Assert.assertTrue(pageContainsText("Wrong answer."));
}
// 7. Subtraction Invalid Input
@Test
public void testSubtractionInvalidInput() {
    login("gagan", "cheema22", "2000-01-01");
    solveMathQuestion("twenty", "eight", "twelve");
    Assert.assertTrue(pageContainsText("Invalid input."));
}
// 8. Subtraction Empty Input
@Test
public void testSubtractionEmptyInput() {
    login("gagan", "cheema22", "2000-01-01");
```

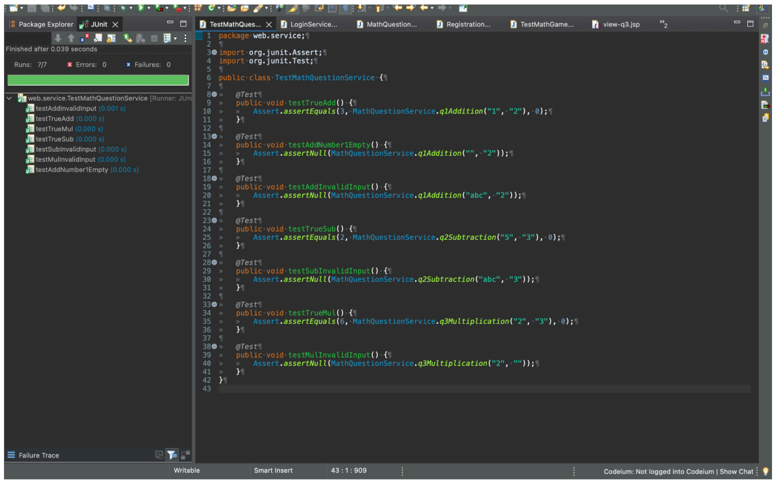
```
        solveMathQuestion("", "", "");
        Assert.assertTrue(pageContainsText("Fields cannot be empty."));
    }
    // 9. Multiplication Correct Answer
    @Test
    public void testMultiplicationCorrectAnswer() {
        login("gagan", "cheema22", "2000-01-01");
        solveMathQuestion("3", "4", "12");
        Assert.assertTrue(pageContainsText("Correct answer!"));
    }
    // 10. Multiplication Incorrect Answer
    @Test
    public void testMultiplicationIncorrectAnswer() {
        login("gagan", "cheema22", "2000-01-01");
        solveMathQuestion("3", "4", "14");
        Assert.assertTrue(pageContainsText("Wrong answer."));
    }
    // 11. Multiplication Invalid Input
    @Test
    public void testMultiplicationInvalidInput() {
        login("gagan", "cheema22", "2000-01-01");
        solveMathQuestion("three", "four", "twelve");
        Assert.assertTrue(pageContainsText("Invalid input."));
    }
    // 12. Multiplication Empty Input
    @Test
    public void testMultiplicationEmptyInput() {
        login("gagan", "cheema22", "2000-01-01");
        solveMathQuestion("", "", "");
        Assert.assertTrue(pageContainsText("Fields cannot be empty."));
    }
}
```

view-q3.jsp

```
<html>
<body>
<h2>Q3</h2>
<div>${message}</div><br/><br/>
<form action="/q3" method="post">
 <label for="number1">First number:</label><br>
 <input type="text" id="number1" name="number1"><br>
  <label for="number2">Second number:</label><br>
 <input type="text" id="number2" name="number2"><br>
  <label for="result">MULTIPLY:</label><br>
 <input type="text" id="result" name="result"><br><br>
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

Output:-

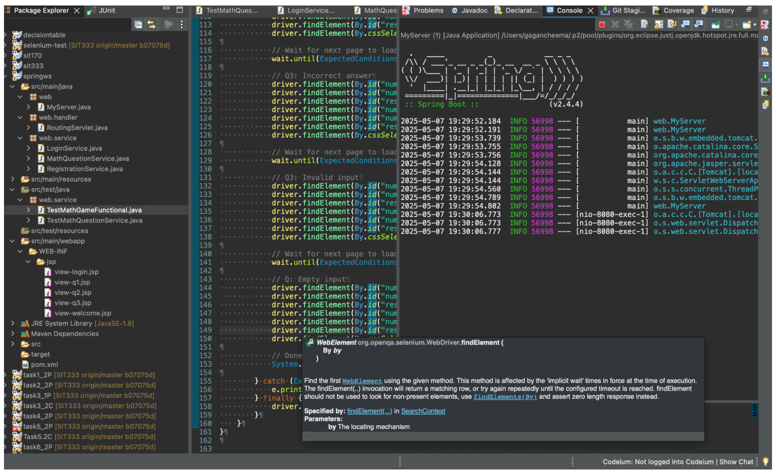
Unit Testing



```
Finished after 0.039 seconds
Runs: 7/7    Errors: 0    Failures: 0

web.service.TestMathQuestionService [Runner: JUni
  testAddInvalidInput (0.001 s)
  testTrueAdd (0.000 s)
  testTrueMul (0.000 s)
  testTrueSub (0.000 s)
  testSubInvalidInput (0.000 s)
  testMulInvalidInput (0.000 s)
  testAddNumber1Empty (0.000 s)
```

```java
package web.service;

import org.junit.Assert;
import org.junit.Test;

public class TestMathQuestionService {

	@Test
	public void testTrueAdd() {
		Assert.assertEquals(3, MathQuestionService.q1Addition("1", "2"), 0);
	}

	@Test
	public void testAddNumber1Empty() {
		Assert.assertNull(MathQuestionService.q1Addition("", "2"));
	}

	@Test
	public void testAddInvalidInput() {
		Assert.assertNull(MathQuestionService.q1Addition("abc", "2"));
	}

	@Test
	public void testTrueSub() {
		Assert.assertEquals(2, MathQuestionService.q2Subtraction("5", "3"), 0);
	}

	@Test
	public void testSubInvalidInput() {
		Assert.assertNull(MathQuestionService.q2Subtraction("abc", "3"));
	}

	@Test
	public void testTrueMul() {
		Assert.assertEquals(6, MathQuestionService.q3Multiplication("2", "3"), 0);
	}

	@Test
	public void testMulInvalidInput() {
		Assert.assertNull(MathQuestionService.q3Multiplication("2", ""));
	}
}
```

Functional Testing



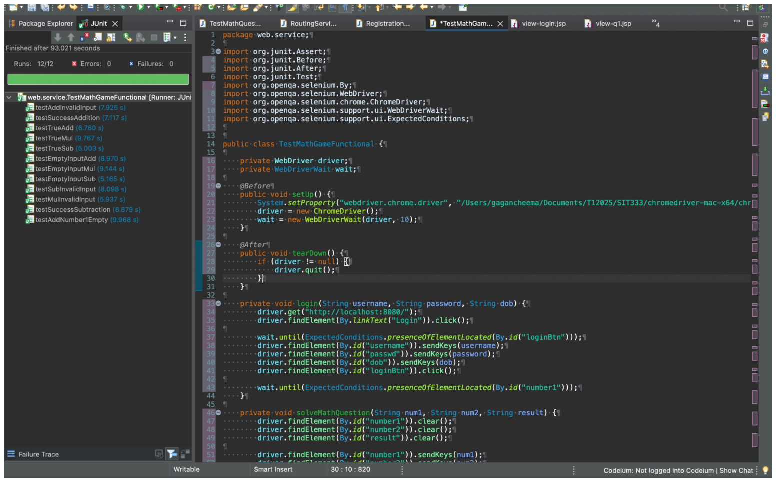
```java
            driver.findElement(By.id("res
            driver.findElement(By.cssSele

            // Wait for next page to load
            wait.until(ExpectedConditions

            // Q3: Incorrect answer
            driver.findElement(By.id("num
            driver.findElement(By.id("num
            driver.findElement(By.id("res
            driver.findElement(By.id("num
            driver.findElement(By.id("num
            driver.findElement(By.id("res
            driver.findElement(By.cssSele

            // Wait for next page to load
            wait.until(ExpectedConditions

            // Q3: Invalid input
            driver.findElement(By.id("num
            driver.findElement(By.id("num
            driver.findElement(By.id("res
            driver.findElement(By.id("num
            driver.findElement(By.id("num
            driver.findElement(By.id("res
            driver.findElement(By.cssSele

            // Wait for next page to load
            wait.until(ExpectedConditions

            // Q: Empty input
            driver.findElement(By.id("num
            driver.findElement(By.id("num
            driver.findElement(By.id("res
            driver.findElement(By.id("num
            driver.findElement(By.id("num
            driver.findElement(By.id("res
            driver.

            // Done
            System.

        } catch (Ex
            e.print
        } finally {
            driver.
        }
    }
}
```

```
MyServer (1) [Java Application] /Users/gagancheema/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.ma

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.4)

2025-05-07 19:29:52.184  INFO 56998 --- [           main] web.MyServer
2025-05-07 19:29:52.191  INFO 56998 --- [           main] web.MyServer
2025-05-07 19:29:53.739  INFO 56998 --- [           main] o.s.b.w.embedded.tomcat.
2025-05-07 19:29:53.755  INFO 56998 --- [           main] o.apache.catalina.core.S
2025-05-07 19:29:53.756  INFO 56998 --- [           main] org.apache.catalina.core
2025-05-07 19:29:54.128  INFO 56998 --- [           main] org.apache.jasper.servle
2025-05-07 19:29:54.144  INFO 56998 --- [           main] o.a.c.c.C.[Tomcat].[loca
2025-05-07 19:29:54.144  INFO 56998 --- [           main] w.s.c.ServletWebServerAp
2025-05-07 19:29:54.560  INFO 56998 --- [           main] o.s.s.concurrent.ThreadP
2025-05-07 19:29:54.789  INFO 56998 --- [           main] o.s.b.w.embedded.tomcat.
2025-05-07 19:29:54.802  INFO 56998 --- [           main] web.MyServer
2025-05-07 19:30:06.773  INFO 56998 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[loca
2025-05-07 19:30:06.773  INFO 56998 --- [nio-8080-exec-1] o.s.web.servlet.Dispatch
2025-05-07 19:30:06.777  INFO 56998 --- [nio-8080-exec-1] o.s.web.servlet.Dispatch
```

WebElement org.openqa.selenium.WebDriver.**findElement** (
  By *by*
)

Find the first WebElement using the given method. This method is affected by the 'implicit wait' times in force at the time of execution. The findElement(..) invocation will return a matching row, or try again repeatedly until the configured timeout is reached. findElement should not be used to look for non-present elements, use findElements(By) and assert zero length response instead.

Specified by: findElement(...) in SearchContext
Parameters:
by The locating mechanism



```
Finished after 93.021 seconds
Runs: 12/12    Errors: 0    Failures: 0

web.service.TestMathGameFunctional [Runner: JUni
  testAddInvalidInput (7.925 s)
  testSuccessAddition (7.117 s)
  testTrueAdd (6.760 s)
  testTrueMul (9.767 s)
  testTrueSub (5.003 s)
  testEmptyInputAdd (8.970 s)
  testEmptyInputMul (9.144 s)
  testEmptyInputSub (5.165 s)
  testSubInvalidInput (8.098 s)
  testMulInvalidInput (5.937 s)
  testSuccessSubtraction (8.879 s)
  testAddNumber1Empty (9.968 s)
```

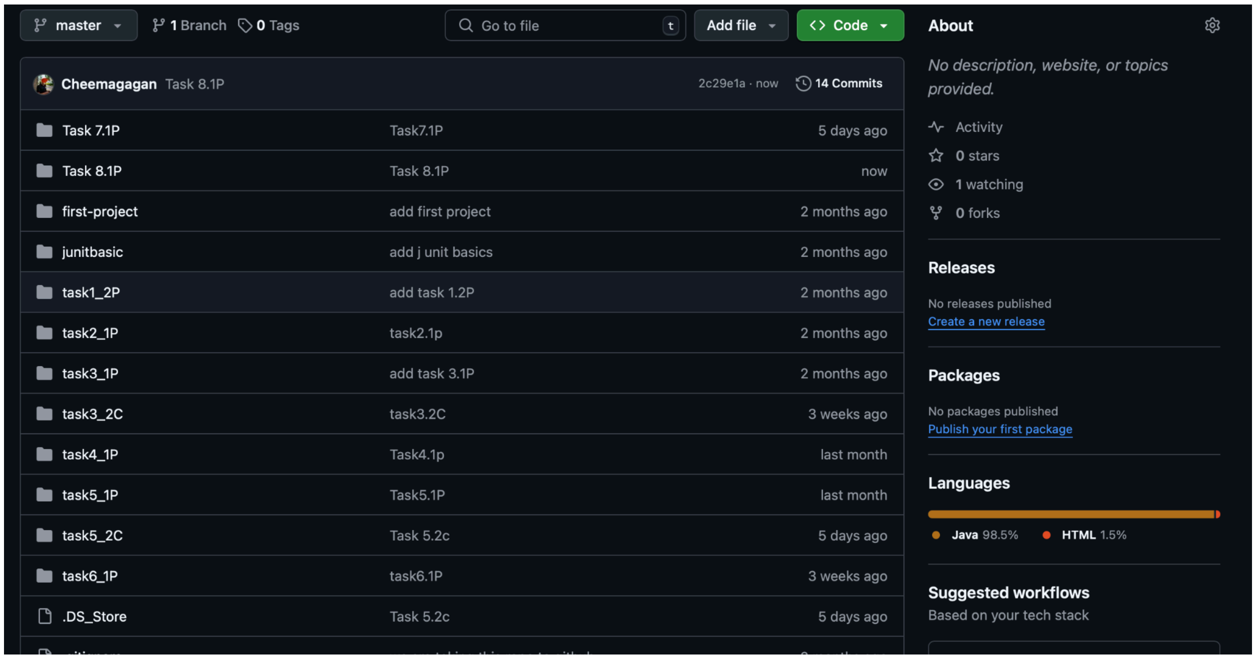
```java
package web.service;

import org.junit.Assert;
import org.junit.Before;
import org.junit.After;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.support.ui.ExpectedConditions;


public class TestMathGameFunctional {

    private WebDriver driver;
    private WebDriverWait wait;

    @Before
    public void setUp() {
        System.setProperty("webdriver.chrome.driver", "/Users/gagancheema/Documents/T12025/SIT333/chromedriver-mac-x64/chr
        driver = new ChromeDriver();
        wait = new WebDriverWait(driver, 10);
    }

    @After
    public void tearDown() {
        if (driver != null) {
            driver.quit();
        }
    }

    private void login(String username, String password, String dob) {
        driver.get("http://localhost:8080/");
        driver.findElement(By.linkText("Login")).click();

        wait.until(ExpectedConditions.presenceOfElementLocated(By.id("loginBtn")));
        driver.findElement(By.id("username")).sendKeys(username);
        driver.findElement(By.id("passwd")).sendKeys(password);
        driver.findElement(By.id("dob")).sendKeys(dob);
        driver.findElement(By.id("loginBtn")).click();

        wait.until(ExpectedConditions.presenceOfElementLocated(By.id("number1")));
    }

    private void solveMathQuestion(String num1, String num2, String result) {
        driver.findElement(By.id("number1")).clear();
        driver.findElement(By.id("number2")).clear();
        driver.findElement(By.id("result")).clear();

        driver.findElement(By.id("number1")).sendKeys(num1);
```

Self-reflection on Testing an Integrated Web Application:

The testing approach for Task 8.1P represents a more advanced stage of testing compared to Task 7.1P. While Task 7.1Pis focused on basic form validation and login functionality, Task 8.1P introduces complex interactions with a multi-step process involving user login, solving math questions, and receiving real-time feedback.

In essence, Task 8.1P provides a broader and more realistic testing scenario that mirrors a real-world web application with dynamic content and interactions. The testing environment is more sophisticated, requiring the use of waits, interactions with multiple UI elements, and validating multiple states and messages. This prepares for testing more feature-rich applications compared to simpler, static web pages.

Github Screenshot:



Cheemagagan Task 8.1P 2c29e1a · now 14 Commits

Task 7.1P	Task7.1P	5 days ago
Task 8.1P	Task 8.1P	now
first-project	add first project	2 months ago
junitbasic	add j unit basics	2 months ago
task1_2P	add task 1.2P	2 months ago
task2_1P	task2.1p	2 months ago
task3_1P	add task 3.1P	2 months ago
task3_2C	task3.2C	3 weeks ago
task4_1P	Task4.1p	last month
task5_1P	Task5.1P	last month
task5_2C	Task 5.2c	5 days ago
task6_1P	task6.1P	3 weeks ago
.DS_Store	Task 5.2c	5 days ago

About

No description, website, or topics provided.

Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● Java 98.5% ● HTML 1.5%

Suggested workflows
Based on your tech stack

Following Key Ideas:

Separating HTML/JSP and Java components improves maintainability and testability. Business logic can be unit tested, UI tested using tools like Selenium, and the entire flow can be tested through integration tests, ensuring a well-structured and testable web application.