



Test Vest

work smarter, not harder

Work smarter, not harder.

Why?

Code testing is:

- Skill requiring
- Time consuming
- Technology specific
- Quite frankly, boring
- **But extremely necessary**

Code tests?

SecondAssign.java

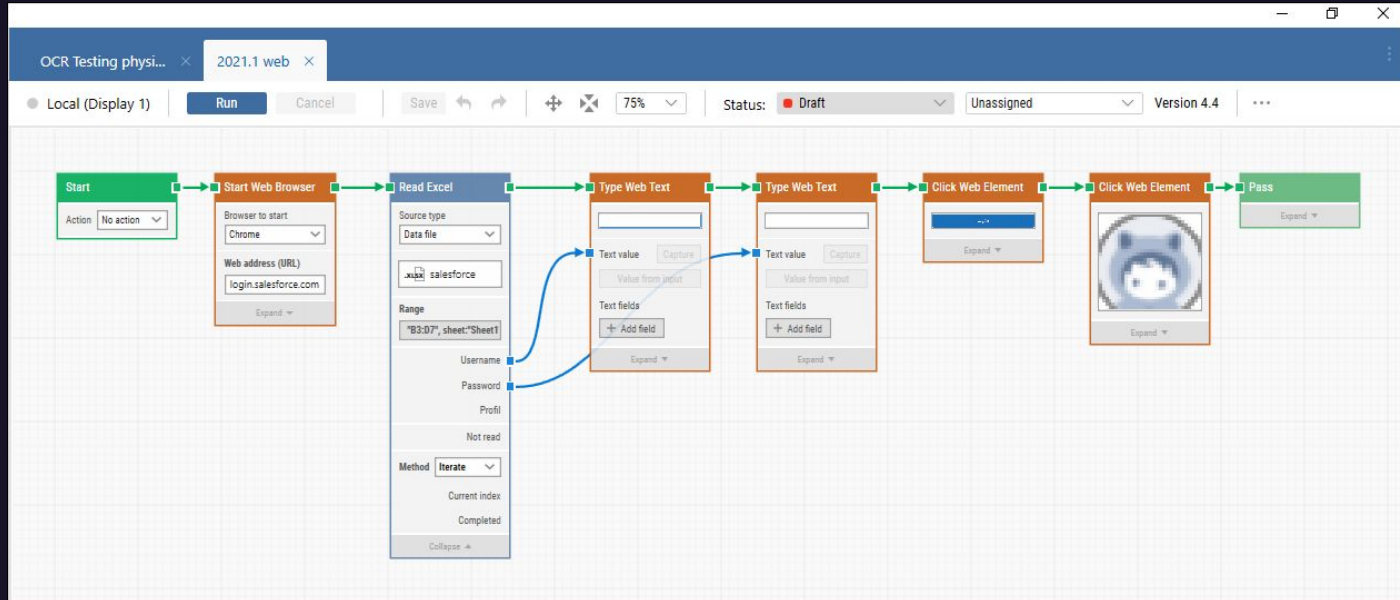
```
1 import org.openqa.selenium.By;
2 import org.openqa.selenium.WebDriver;
3 import org.openqa.selenium.chrome.ChromeDriver;
4 import java.util.concurrent.TimeUnit;
5
6 public class SecondAssign {
7     public static void main(String[] p) {
8
9         System.setProperty("webdriver.chrome.driver", "chromedriver");
10        WebDriver brw = new ChromeDriver();
11        brw.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
12        brw.get("https://www.selenium.dev/");
13        String text = brw.findElement(By.tagName("h1")).getText();
14
15        if (brw.getPageSource().contains("Selenium automates browsers")) {
16            System.out.println("The searched text: " + text + " exists.");
17        } else
18            System.out.println("The searched text: " + text + " does not exists.");
19        brw.quit();
20    }
21
22 }
23 |
```

We'd rather not!

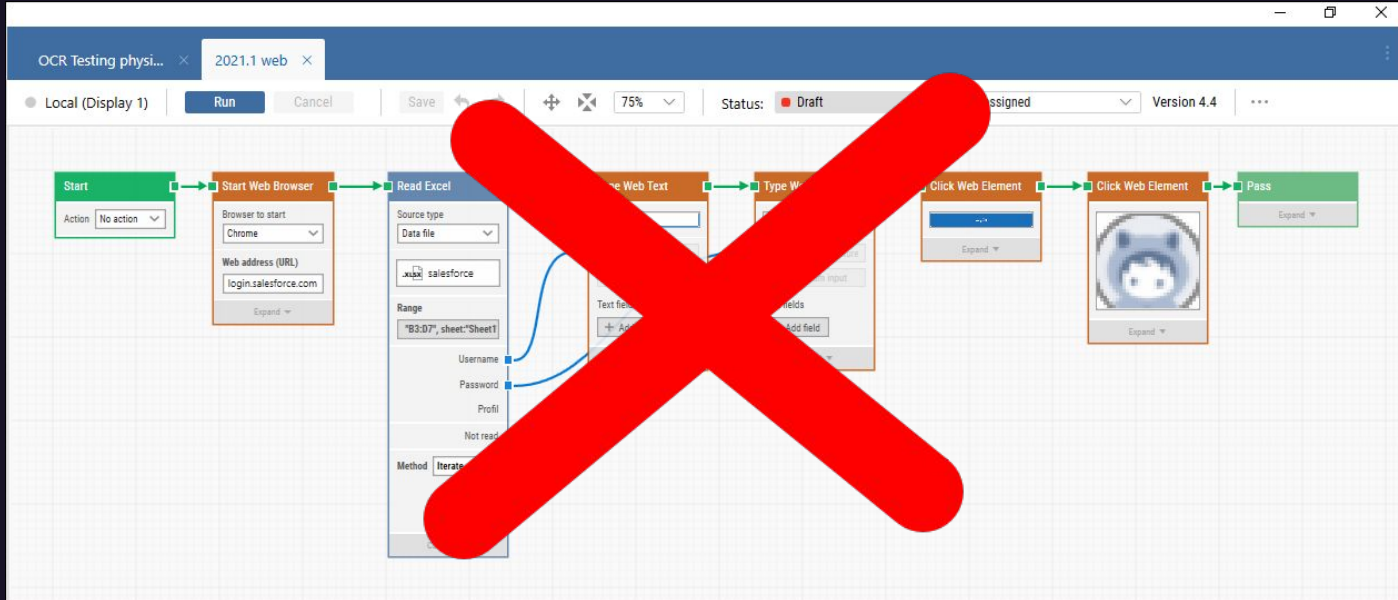
```
SecondAssign.java
1 import org.openqa.selenium.By;
2 import org.openqa.selenium.WebDriver;
3 import org.openqa.selenium.chrome.ChromeDriver;
4 import java.util.concurrent.TimeUnit;
5
6 public class SecondAssign {
7     public static void main(String[] p) {
8
9         System.setProperty("webdriver.chrome.driver", "chromedriver");
10        WebDriver brw = new ChromeDriver();
11        brw.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
12        brw.get("https://www.selenium.dev/");
13        String text = brw.findElement(By.tagName("h1")).getText();
14
15        if (brw.getPageSource().contains("Selenium automates browsers")) {
16            System.out.println("The searched text: " + text + " exists.");
17        } else {
18            System.out.println("The searched text: " + text + " does not exists.");
19        }
20    }
21 }
22 }
23 |
```

Codeless testing!

We are not the first ones...

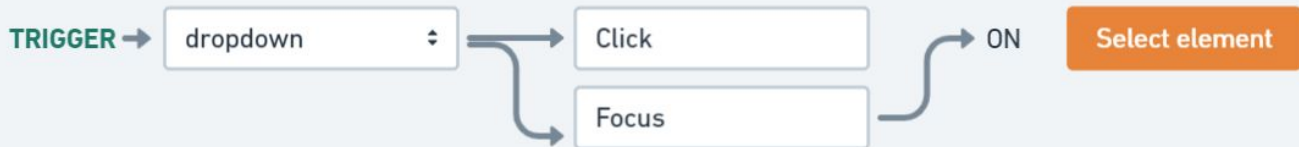
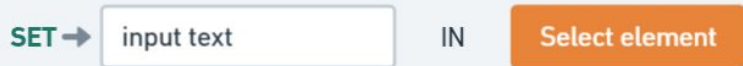


But this isn't very user friendly



Test Vest to the
rescue!

Concept



Landing page



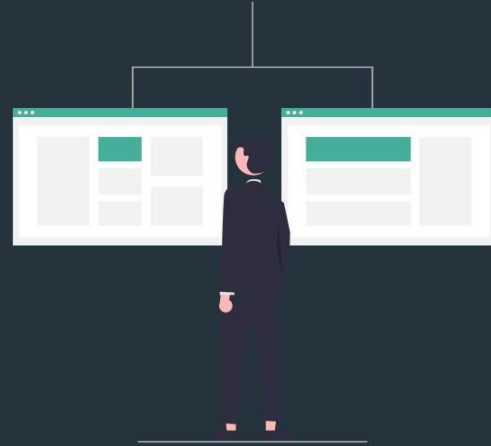
[Home](#) [Start](#)

[Sign Up](#)

TEST VEST

Work smarter, not harder

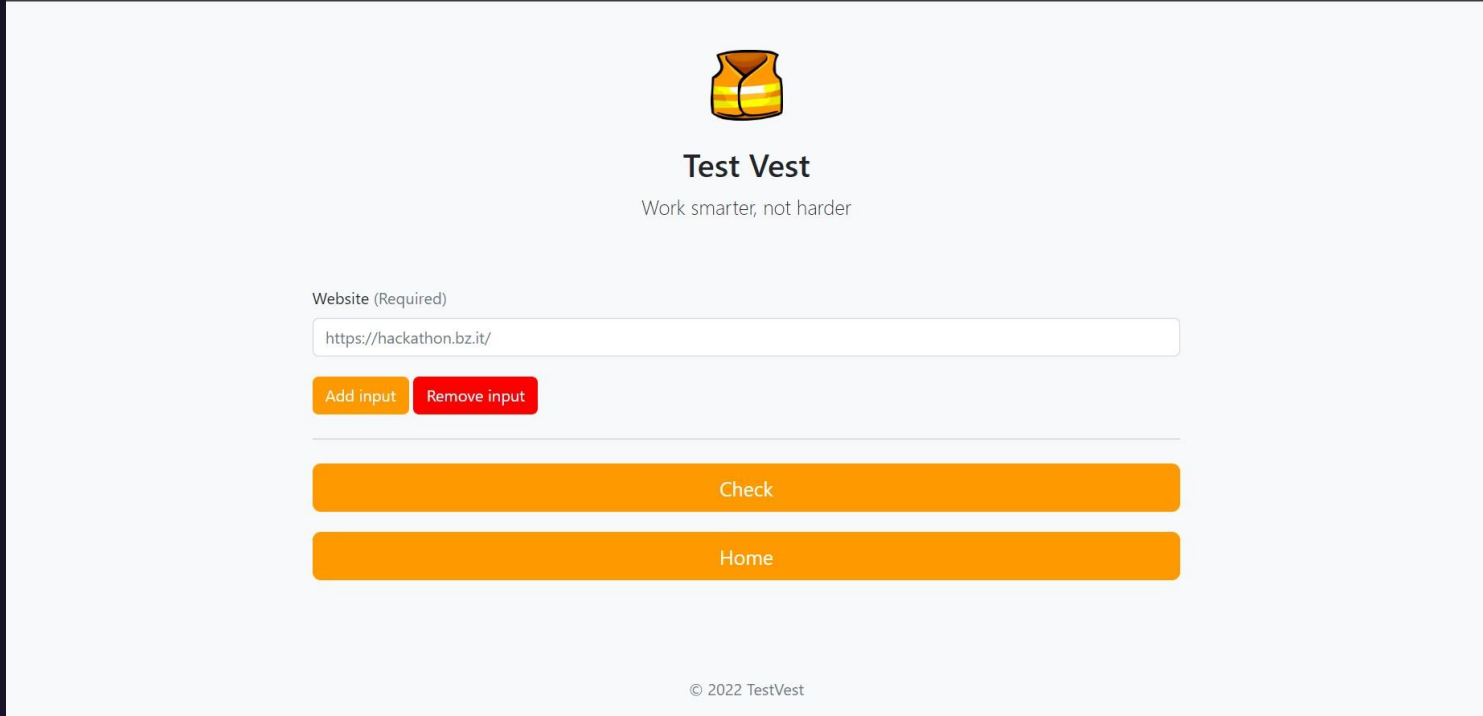
[Get Started](#)



Make testing speak
your language

Simple and intuitive UI

The App



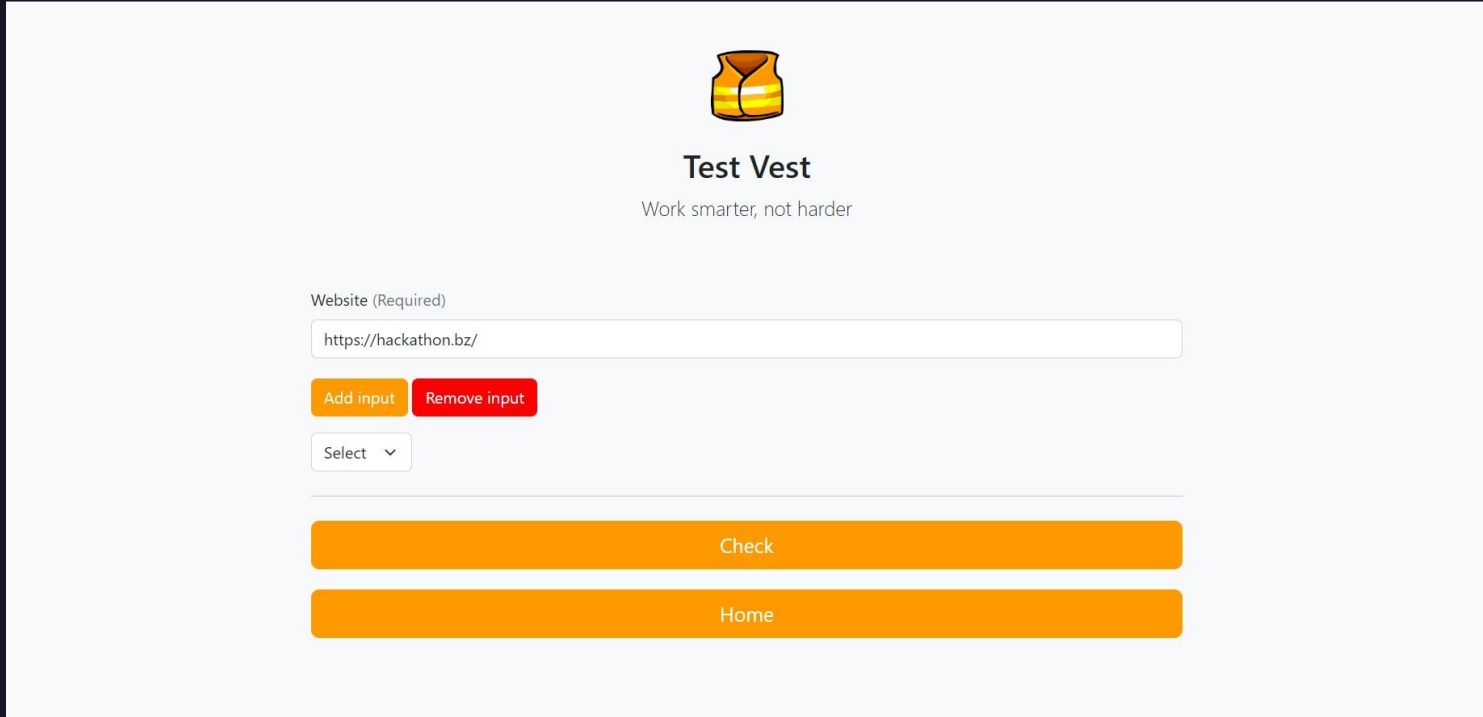
The screenshot shows a web application interface for 'Test Vest'. At the top center is a yellow and orange life vest icon. Below it, the text 'Test Vest' is displayed in a bold, black font, followed by the tagline 'Work smarter, not harder' in a smaller, grey font. A form section is titled 'Website (Required)' and contains a single text input field with the URL 'https://hackathon.bz.it/'. Below the input field are two buttons: 'Add input' (orange) and 'Remove input' (red). A horizontal line separates the input section from the navigation area, which consists of two large, orange buttons labeled 'Check' and 'Home'. At the bottom center of the page, there is a small copyright notice: '© 2022 TestVest'.

Simple enough for your grandma! (maybe)

Full use-case video on our project page.

Simple and intuitive UI

Add a test element



The screenshot shows a web application titled "Test Vest" with the tagline "Work smarter, not harder". At the top center is an icon of a yellow and orange safety vest. Below the title is a form with a label "Website (Required)" and a text input field containing "https://hackathon.bz/". Underneath the input field are two buttons: "Add input" (orange) and "Remove input" (red). Below these buttons is a dropdown menu with the text "Select" and a downward arrow. At the bottom of the form are two large orange buttons: "Check" and "Home".

Simple enough for your grandma! (maybe)

Full use-case video on our project page.

Simple and intuitive UI

Testing types

The screenshot displays the 'Test Vest' web application interface. At the top, the title 'Test Vest' is centered, with the tagline 'Work smarter, not harder' below it. The main section is a form for configuring a test. It starts with a 'Website (Required)' label above a text input field containing 'https://hackathon.bz/'. Below this are two buttons: 'Add input' (orange) and 'Remove input' (red). The test configuration is organized into three rows. Each row has a dropdown menu on the left (labeled 'Has', 'Set', and 'Trigger' respectively), a text input field in the middle (containing 'something...' or 'click'), a preposition in the center ('in' or 'on'), and another dropdown menu on the right (labeled 'whole page' or 'custom element'). At the bottom of the form are two large orange buttons: 'Check' and 'Home'. A copyright notice '© 2022 TestVest' is visible at the very bottom of the page.

Simple enough for your grandma! (maybe)

Full use-case video on our project page.

Simple and intuitive UI

Insert your test input and
click "Check"

Website (Required)

Add input Remove input

Has	<input type="text" value="Hackathon"/>	in	whole page
Set	<input type="text" value="Confirm"/>	in	Button
Trigger	<input type="text" value="click"/>	on	Link

Check

Home

Simple enough for your grandma! (maybe)

Full use-case video on our project page.

Simple and intuitive UI

Get clear responses:
did it work?

Test Vest

Work smarter, not harder

Website (Required)

Add input Remove input

✓	Has	<input type="text" value="Hackathon"/>	in	<input type="text" value="whole page"/>
✗	Set	<input type="text" value="Confirm"/>	in	<input type="text" value="Button"/>
✓	Trigger	<input type="text" value="click"/>	on	<input type="text" value="Link"/>

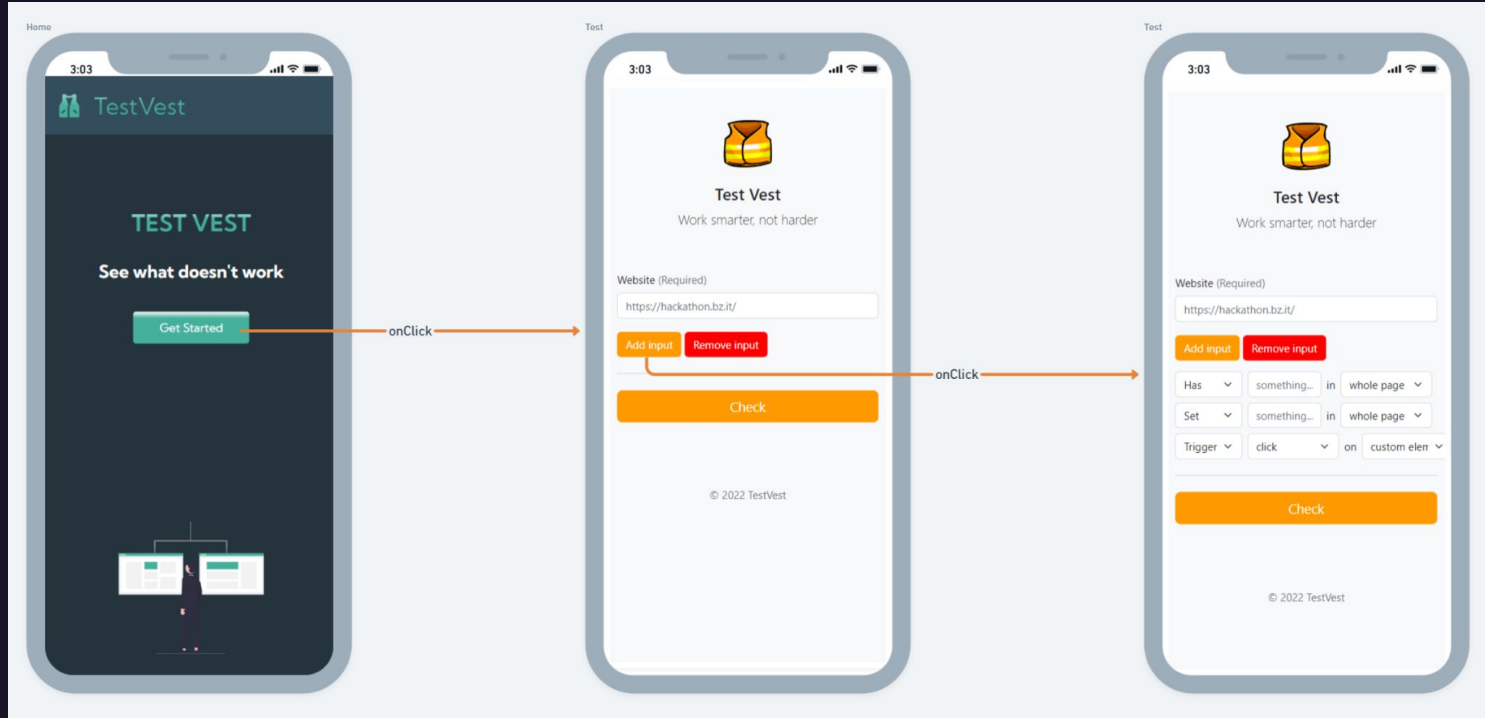
Check

Home

Simple enough for your grandma! (maybe)

Full use-case video on our project page.

Testing on the go? No prob!



Same interface for an even softer learning curve.

Thank you for
your attention.