

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Институт Информатики и кибернетики   
Кафедра Программных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
  
к лабораторной работе №5 по дисциплине «Автоматизация тестирования»

Обучающийся группы 6231-020302D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гижевская В.Д.

Обучающийся группы 6231-020302D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алкеев М.Г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лобанков А.А.

Самара 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc146727791)

[2 Результаты работы 4](#_Toc146727792)

[ПРИЛОЖЕНИЕ А Листинг программы 8](#_Toc146727796)

1. Постановка задачи

В процессе выполнения заданий ознакомиться с библиотеками для сквозного тестирования front-end приложений. За основу взять приложение из 5 лабораторной работы. В качестве фреймворка можно использовать Protractor (selenium) или cypress.

Необходимо описать несколько тест-кейсов с использованием Gherkin нотации. Для обращения к одним и тем же элементам страницы рекомендуется использовать паттерн Page Object.

1. Проверить, что у нас на странице существуют инпуты, дропдаун и кнопка для получения результата.
2. Проверить, что корректно работают все арифметические действия
3. Проверить, что в поля ввода можно ввести только цифры.
4. Проверить, что при выборе деления во второй инпут нельзя ввести 0
5. Проверить, что при выборе шестнадцатеричной системы счисления в инпуты можно вводить соответствующие буквы
6. Проверить, цвета результата в зависимости от знака
7. Результаты работы

На рисунке 1 представлен список тестов.

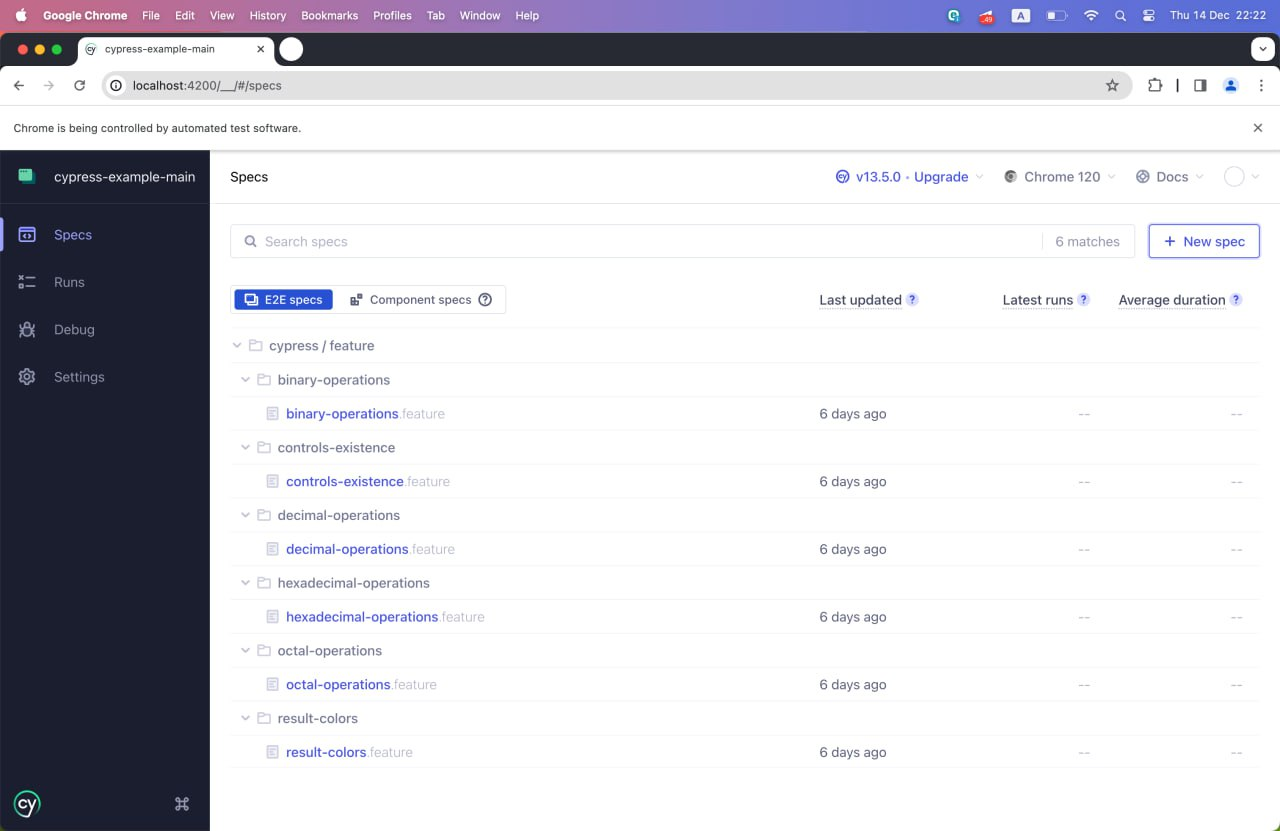


Рисунок 1 – Список тестов

На рисунках 2-7 представлены успешные результаты тестов по каждому из необходимых кейсов.

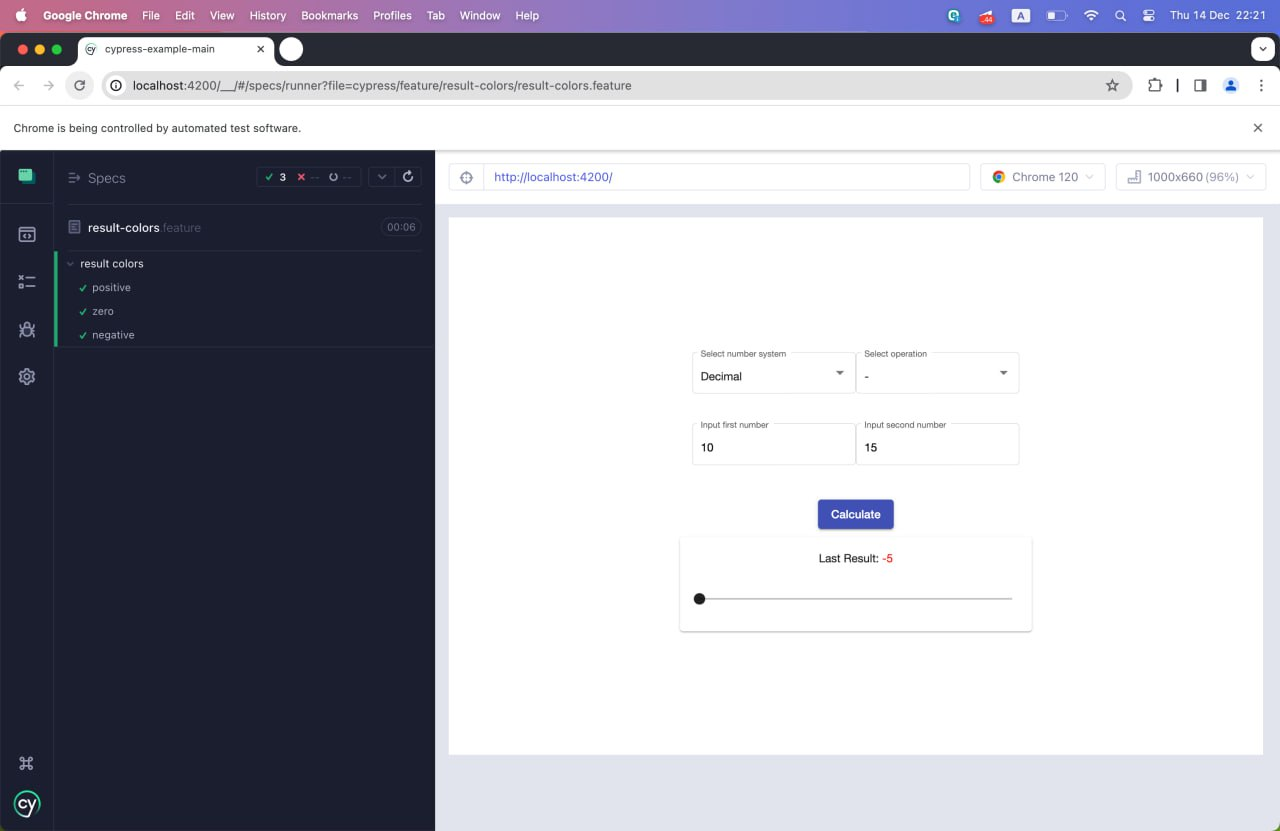


Рисунок 2 – Тесты на цвета результатов

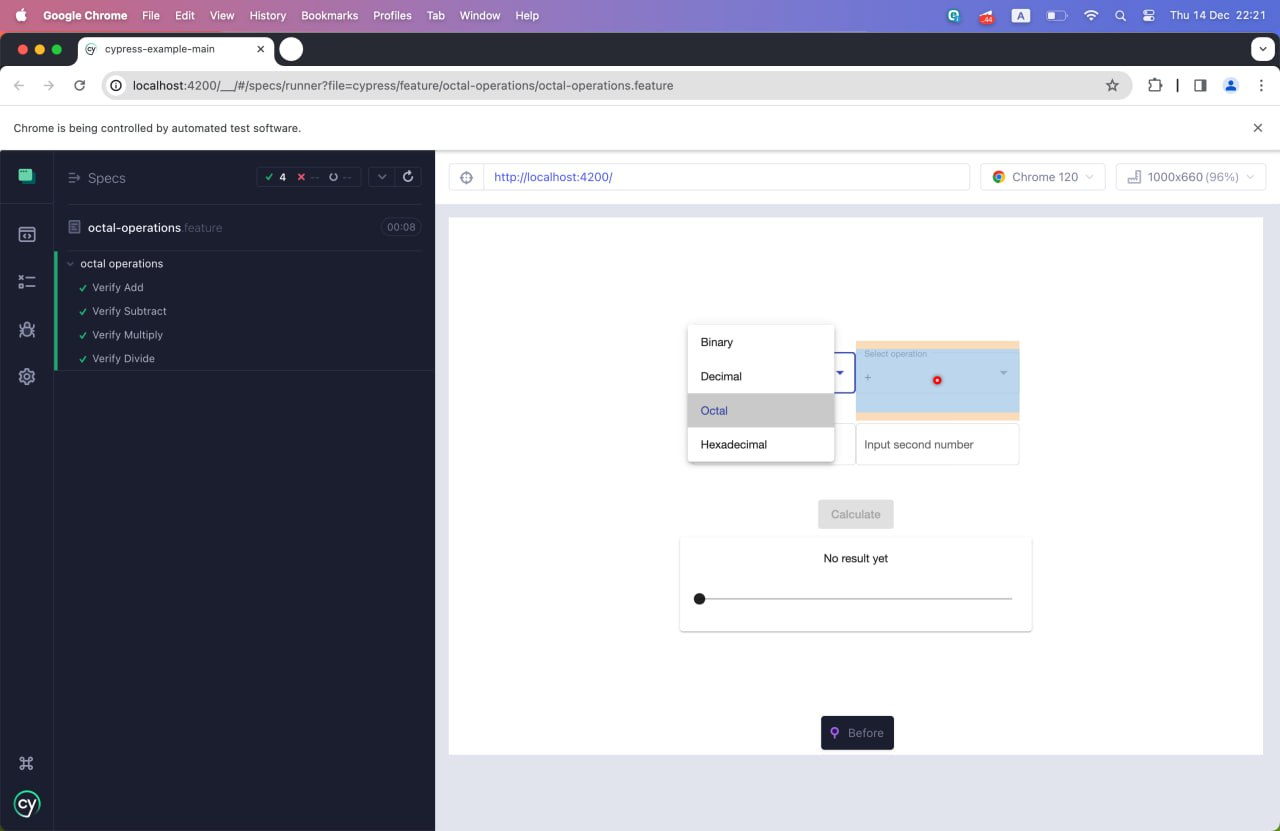


Рисунок 3 – Тесты на восьмиричный калькулятор

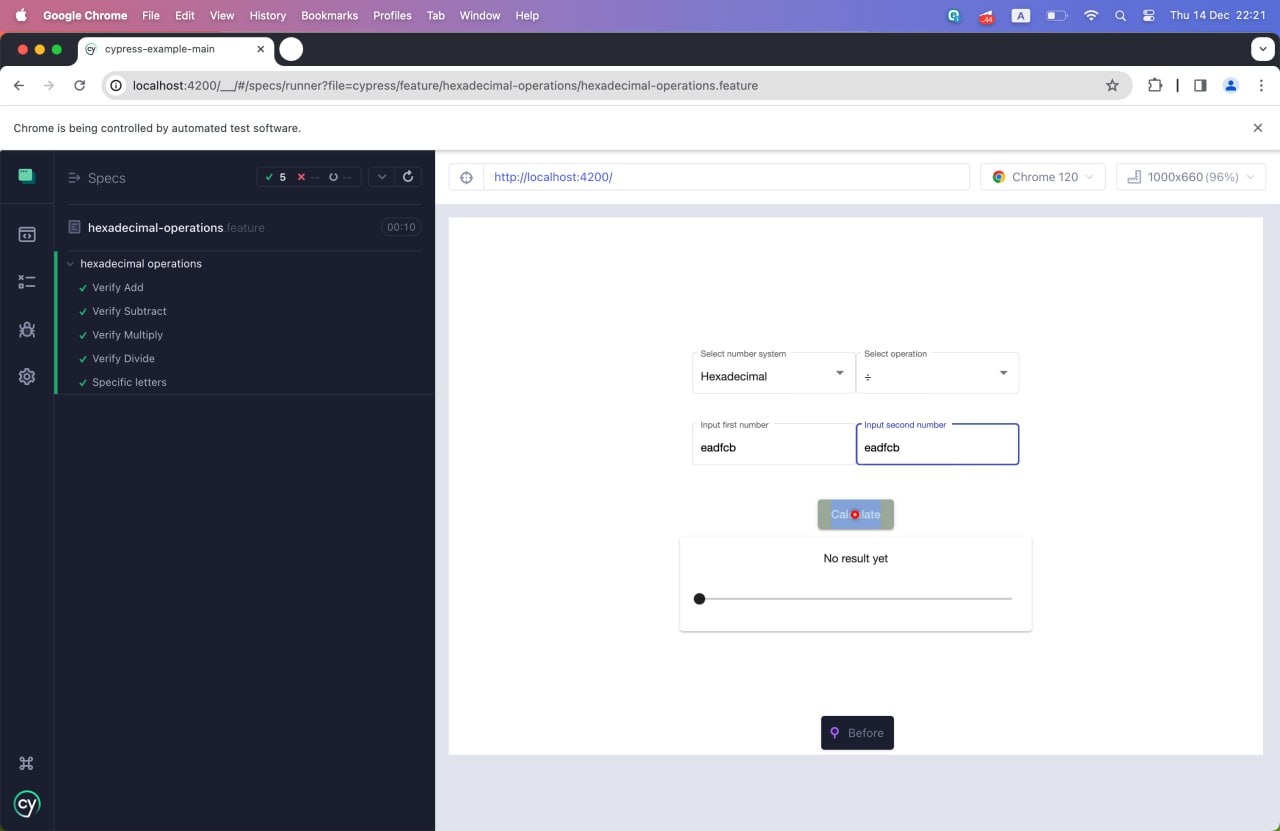


Рисунок 4 – Тесты на шестнадцатеричный калькулятор

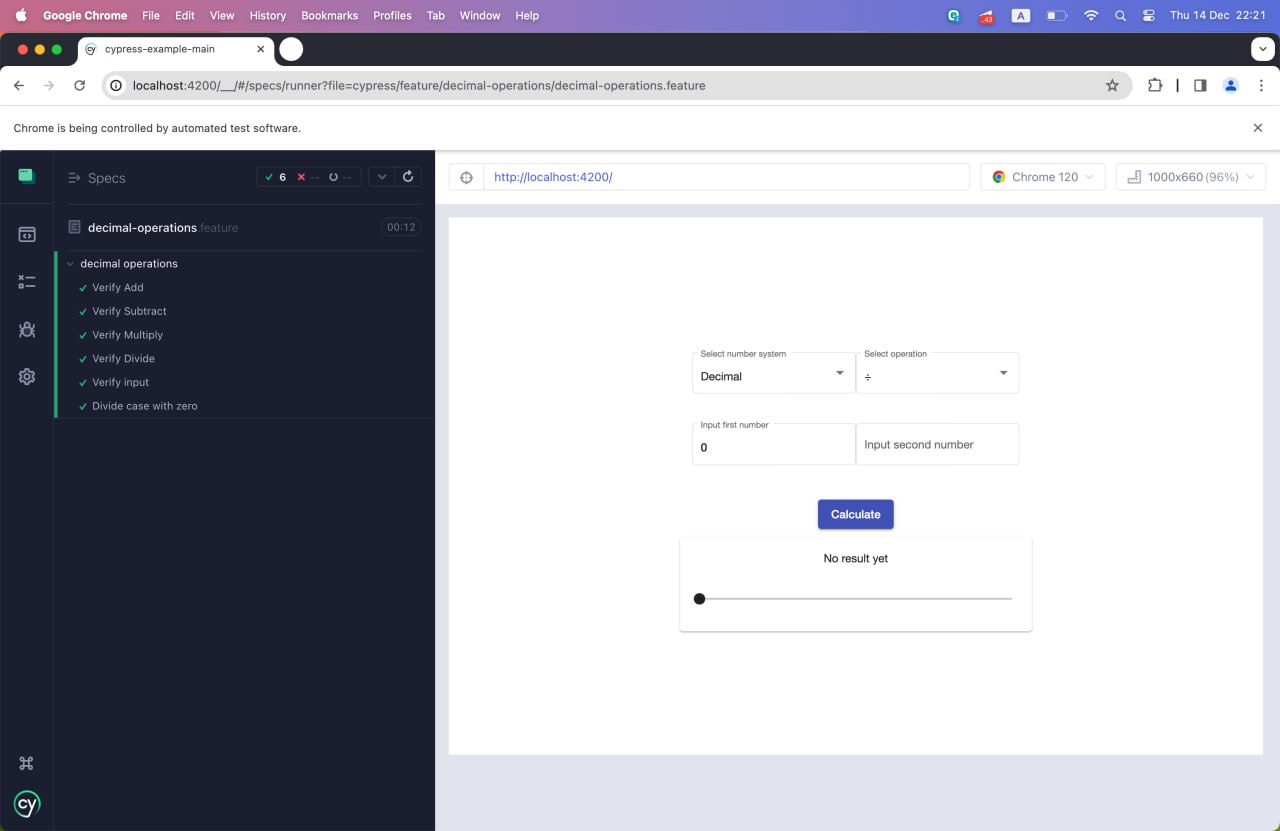


Рисунок 5 – Тесты на десятичный калькулятор

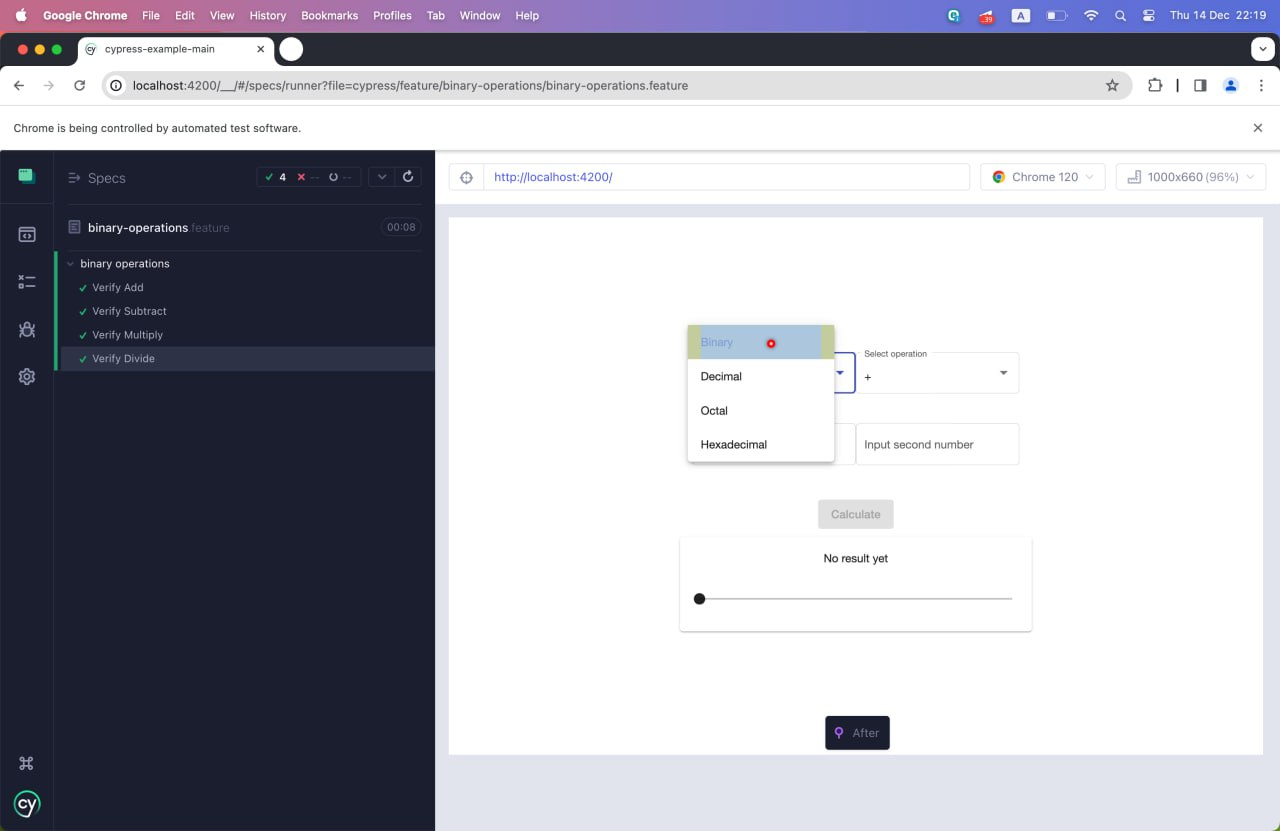


Рисунок 6 – Тесты на двоичный калькулятор

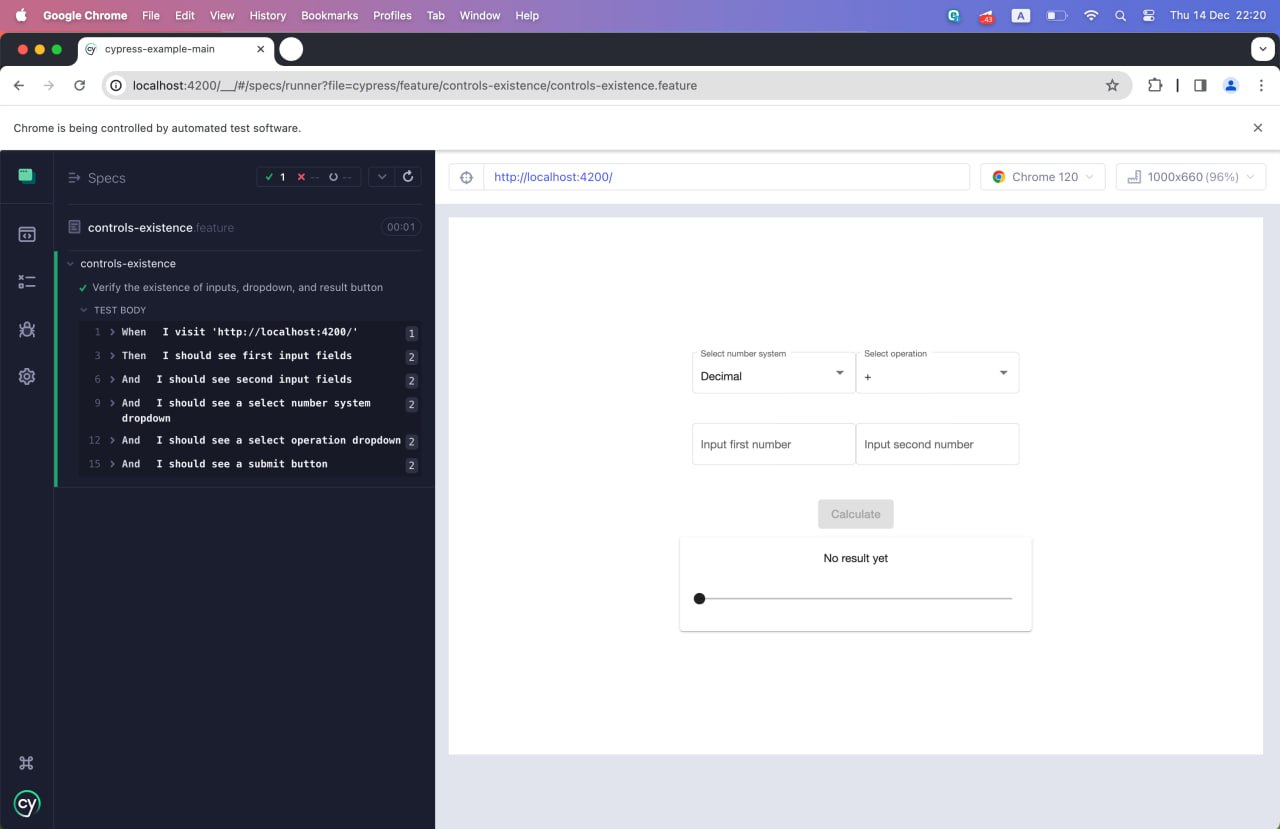


Рисунок 7 – Тесты на наличие контролов

ПРИЛОЖЕНИЕ А  
Листинг программы

class MainPage {

static visitPage(url: string) {

cy.visit(url);

}

static getNumberInput(label: string) {

return cy.get(app-number-input[label="${label}"]);

}

static getSelect(label: string) {

return cy.contains("mat-form-field", label);

}

static getSubmitButton() {

return cy.contains("button", "Calculate");

}

static performArithmeticOperation(

numberSystem,

operation,

firstNumber,

secondNumber

) {

MainPage.getSelect("Select number system").click();

cy.get(mat-option[ng-reflect-value="${numberSystem}"]).click();

MainPage.getSelect("Select operation").click();

cy.get(mat-option[ng-reflect-value="${operation}"]).click();

MainPage.getNumberInput("Input first number").type(firstNumber);

MainPage.getNumberInput("Input second number").type(secondNumber);

MainPage.getSubmitButton().click();

}

static verifyOperation(expectedResult) {

return cy.get(".result-card").contains(Last Result: ${expectedResult});

}

static getResult(value: string) {

return cy.get(".result-card").find("span").contains(value);

}

static input(string: string): void {

MainPage.getNumberInput("Input first number").type(string);

MainPage.getNumberInput("Input second number").type(string);

}

}

export default MainPage

Feature: binary operations

Scenario: Verify Add

Given I visit 'http://localhost:4200/'

When I input data to form, set add operation and submit

Then I should get right add result

Scenario: Verify Subtract

Given I visit 'http://localhost:4200/'

When I input data to form, set subtract operation and submit

Then I should get right subtract result

Scenario: Verify Multiply

Given I visit 'http://localhost:4200/'

When I input data to form, set multiply operation and submit

Then I should get right multiply result

Scenario: Verify Divide

Given I visit 'http://localhost:4200/'

When I input data to form, set divide operation and submit

Then I should get right divide result

import { Given, Then, When } from "@badeball/cypress-cucumber-preprocessor";

import { ENumberSystemType } from "../../../../contracts/number-system-type.enum";

import { EOperation } from "../../../../contracts/operation.enum";

import MainPage from "../../e2e/main-page";

const bunaryNumberSystemType = ENumberSystemType.Binary;

Given("I visit {string}", (url: string) => {

MainPage.visitPage(url);

});

When("I input data to form, set add operation and submit", (url: string) => {

MainPage.performArithmeticOperation(

bunaryNumberSystemType,

EOperation.Add,

"1",

"1"

);

});

Then("I should get right add result", () => {

MainPage.verifyOperation("10").should("exist");

});

When(

"I input data to form, set subtract operation and submit",

(url: string) => {

MainPage.performArithmeticOperation(

bunaryNumberSystemType,

EOperation.Subtract,

"1000011",

"10001"

);

}

);

Then("I should get right subtract result", () => {

MainPage.verifyOperation("110010").should("exist");

});

When(

"I input data to form, set multiply operation and submit",

(url: string) => {

MainPage.performArithmeticOperation(

bunaryNumberSystemType,

EOperation.Multiply,

"10001",

"101"

);

}

);

Then("I should get right multiply result", () => {

MainPage.verifyOperation("1010101").should("exist");

});

When("I input data to form, set divide operation and submit", (url: string) => {

MainPage.performArithmeticOperation(

bunaryNumberSystemType,

EOperation.Divide,

"11111",

"1011"

);

});

Then("I should get right divide result", () => {

MainPage.verifyOperation("10").should("exist");

});

Feature: controls-existence

Scenario: Verify the existence of inputs, dropdown, and result button

When I visit 'http://localhost:4200/'

Then I should see first input fields

And I should see second input fields

And I should see a select number system dropdown

And I should see a select operation dropdown

And I should see a submit button

import { Given } from "@badeball/cypress-cucumber-preprocessor";

import MainPage from "../../e2e/main-page";

const { When, Then } = require("@badeball/cypress-cucumber-preprocessor");

When("I visit {string}", (url: string) => {

MainPage.visitPage(url);

});

Then("I should see first input fields", () => {

MainPage.getNumberInput("Input first number").should("exist");

});

Then("I should see second input fields", () => {

MainPage.getNumberInput("Input second number").should("exist");

});

Then("I should see a select number system dropdown", () => {

MainPage.getSelect("Select number system").should("exist");

});

Then("I should see a select operation dropdown", () => {

MainPage.getSelect("Select operation").should("exist");

});

Then("I should see a submit button", () => {

MainPage.getSubmitButton().should("exist");

});

Feature: decimal operations

Scenario: Verify Add

Given I visit 'http://localhost:4200/'

When I input data to form, set add operation and submit

Then I should get right add result

Scenario: Verify Subtract

Given I visit 'http://localhost:4200/'

When I input data to form, set subtract operation and submit

Then I should get right subtract result

Scenario: Verify Multiply

Given I visit 'http://localhost:4200/'

When I input data to form, set multiply operation and submit

Then I should get right multiply result

Scenario: Verify Divide

Given I visit 'http://localhost:4200/'

When I input data to form, set divide operation and submit

Then I should get right divide result

Scenario: Verify input

Given I visit 'http://localhost:4200/'

When I input data to form

Then I should get only numbers

Scenario: Divide case with zero

Given I visit 'http://localhost:4200/'

When I try input zero to second input

Then I should not be able to input zero

import { Given, Then, When } from "@badeball/cypress-cucumber-preprocessor";

import { ENumberSystemType } from "../../../../contracts/number-system-type.enum";

import { EOperation } from "../../../../contracts/operation.enum";

import MainPage from "../../e2e/main-page";

const decimalNumberSystemType = ENumberSystemType.Decimal;

Given("I visit {string}", (url: string) => {

MainPage.visitPage(url);

});

When("I input data to form, set add operation and submit", (url: string) => {

MainPage.performArithmeticOperation(

decimalNumberSystemType,

EOperation.Add,

"10",

"5"

);

});

Then("I should get right add result", () => {

MainPage.verifyOperation("15").should("exist");

});

When(

"I input data to form, set subtract operation and submit",

(url: string) => {

MainPage.performArithmeticOperation(

decimalNumberSystemType,

EOperation.Subtract,

"10",

"5"

);

}

);

Then("I should get right subtract result", () => {

MainPage.verifyOperation("5").should("exist");

});

When(

"I input data to form, set multiply operation and submit",

(url: string) => {

MainPage.performArithmeticOperation(

decimalNumberSystemType,

EOperation.Multiply,

"10",

"5"

);

}

);

Then("I should get right multiply result", () => {

MainPage.verifyOperation("50").should("exist");

});

When("I input data to form, set divide operation and submit", (url: string) => {

MainPage.performArithmeticOperation(

decimalNumberSystemType,

EOperation.Divide,

"10",

"5"

);

});

Then("I should get right divide result", () => {

MainPage.verifyOperation("2").should("exist");

});

When("I input data to form", (url: string) => {

MainPage.input("1234FRG@%^&\*&U^YTutyrg4546");

});

Then("I should get only numbers", () => {

MainPage.getNumberInput("Input first number")

.find("input")

.invoke("val")

.should("eq", "12344546");

MainPage.getNumberInput("Input second number")

.find("input")

.invoke("val")

.should("eq", "12344546");

});

When("I try input zero to second input", (url: string) => {

MainPage.performArithmeticOperation(

decimalNumberSystemType,

EOperation.Divide,

"0",

"0"

);

});

Then("I should not be able to input zero", () => {

MainPage.getNumberInput("Input first number")

.find("input")

.invoke("val")

.should("eq", "0");

MainPage.getNumberInput("Input second number")

.find("input")

.invoke("val")

.should("eq", "");

});

Feature: hexadecimal operations

Scenario: Verify Add

Given I visit 'http://localhost:4200/'

When I input data to form, set add operation and submit

Then I should get right add result

Scenario: Verify Subtract

Given I visit 'http://localhost:4200/'

When I input data to form, set subtract operation and submit

Then I should get right subtract result

Scenario: Verify Multiply

Given I visit 'http://localhost:4200/'

When I input data to form, set multiply operation and submit

Then I should get right multiply result

Scenario: Verify Divide

Given I visit 'http://localhost:4200/'

When I input data to form, set divide operation and submit

Then I should get right divide result

Scenario: Specific letters

Given I visit 'http://localhost:4200/'

When I input data to form with specific lettrs

Then I should get see a b c d e f

import { Given, Then, When } from "@badeball/cypress-cucumber-preprocessor";

import { ENumberSystemType } from "../../../../contracts/number-system-type.enum";

import { EOperation } from "../../../../contracts/operation.enum";

import MainPage from "../../e2e/main-page";

const hexadecimalNumberSystemType = ENumberSystemType.Hexadecimal;

Given("I visit {string}", (url: string) => {

MainPage.visitPage(url);

});

When("I input data to form, set add operation and submit", (url: string) => {

MainPage.performArithmeticOperation(

hexadecimalNumberSystemType,

EOperation.Add,

"c",

"1"

);

});

Then("I should get right add result", () => {

MainPage.verifyOperation("d").should("exist");

});

When(

"I input data to form, set subtract operation and submit",

(url: string) => {

MainPage.performArithmeticOperation(

hexadecimalNumberSystemType,

EOperation.Subtract,

"32",

"6"

);

}

);

Then("I should get right subtract result", () => {

MainPage.verifyOperation("2c").should("exist");

});

When(

"I input data to form, set multiply operation and submit",

(url: string) => {

MainPage.performArithmeticOperation(

hexadecimalNumberSystemType,

EOperation.Multiply,

"7",

"6"

);

}

);

Then("I should get right multiply result", () => {

MainPage.verifyOperation("2a").should("exist");

});

When("I input data to form, set divide operation and submit", (url: string) => {

MainPage.performArithmeticOperation(

hexadecimalNumberSystemType,

EOperation.Divide,

"f",

"f"

);

});

Then("I should get right divide result", () => {

MainPage.verifyOperation("1").should("exist");

});

When("I input data to form with specific lettrs", (url: string) => {

MainPage.performArithmeticOperation(

hexadecimalNumberSystemType,

EOperation.Divide,

"qwertyuiopasdfghjklzxcvbnm",

"qwertyuiopasdfghjklzxcvbnm"

);

});

Then("I should get see a b c d e f", () => {

MainPage.getNumberInput("Input first number")

.find("input")

.invoke("val")

.should("eq", "eadfcb");

MainPage.getNumberInput("Input second number")

.find("input")

.invoke("val")

.should("eq", "eadfcb");

});

Feature: octal operations

Scenario: Verify Add

Given I visit 'http://localhost:4200/'

When I input data to form, set add operation and submit

Then I should get right add result

Scenario: Verify Subtract

Given I visit 'http://localhost:4200/'

When I input data to form, set subtract operation and submit

Then I should get right subtract result

Scenario: Verify Multiply

Given I visit 'http://localhost:4200/'

When I input data to form, set multiply operation and submit

Then I should get right multiply result

Scenario: Verify Divide

Given I visit 'http://localhost:4200/'

When I input data to form, set divide operation and submit

Then I should get right divide result

import { Given, Then, When } from "@badeball/cypress-cucumber-preprocessor";

import { ENumberSystemType } from "../../../../contracts/number-system-type.enum";

import { EOperation } from "../../../../contracts/operation.enum";

import MainPage from "../../e2e/main-page";

const octalNumberSystemType = ENumberSystemType.Octal;

Given("I visit {string}", (url: string) => {

MainPage.visitPage(url);

});

When("I input data to form, set add operation and submit", (url: string) => {

MainPage.performArithmeticOperation(

octalNumberSystemType,

EOperation.Add,

"5",

"5"

);

});

Then("I should get right add result", () => {

MainPage.verifyOperation("12").should("exist");

});

When(

"I input data to form, set subtract operation and submit",

(url: string) => {

MainPage.performArithmeticOperation(

octalNumberSystemType,

EOperation.Subtract,

"10",

"6"

);

}

);

Then("I should get right subtract result", () => {

MainPage.verifyOperation("2").should("exist");

});

When(

"I input data to form, set multiply operation and submit",

(url: string) => {

MainPage.performArithmeticOperation(

octalNumberSystemType,

EOperation.Multiply,

"18",

"4"

);

}

);

Then("I should get right multiply result", () => {

MainPage.verifyOperation("4").should("exist");

});

When("I input data to form, set divide operation and submit", (url: string) => {

MainPage.performArithmeticOperation(

octalNumberSystemType,

EOperation.Divide,

"10",

"4"

);

});

Then("I should get right divide result", () => {

MainPage.verifyOperation("2").should("exist");

});

Feature: result colors

Scenario: positive

Given I visit 'http://localhost:4200/'

When I get result that grater than zero

Then I should see green result

Scenario: zero

Given I visit 'http://localhost:4200/'

When I get result that equals zero

Then I should see black result

Scenario: negative

Given I visit 'http://localhost:4200/'

When I get result that less than zero

Then I should see red result

import { Given, Then, When } from "@badeball/cypress-cucumber-preprocessor";

import { ENumberSystemType } from "../../../../contracts/number-system-type.enum";

import { EOperation } from "../../../../contracts/operation.enum";

import MainPage from "../../e2e/main-page";

const decimalNumberSystemType = ENumberSystemType.Decimal;

Given("I visit {string}", (url: string) => {

MainPage.visitPage(url);

});

When("I get result that grater than zero", (url: string) => {

MainPage.performArithmeticOperation(

decimalNumberSystemType,

EOperation.Add,

"10",

"5"

);

});

Then("I should see green result", () => {

MainPage.getResult("15")

.invoke("css", "color")

.should("eq", "rgb(0, 128, 0)");

});

When("I get result that equals zero", (url: string) => {

MainPage.performArithmeticOperation(

decimalNumberSystemType,

EOperation.Subtract,

"10",

"10"

);

});

Then("I should see black result", () => {

MainPage.getResult("0").invoke("css", "color").should("eq", "rgb(0, 0, 0)");

});

When("I get result that less than zero", (url: string) => {

MainPage.performArithmeticOperation(

decimalNumberSystemType,

EOperation.Subtract,

"10",

"15"

);

});

Then("I should see red result", () => {

MainPage.getResult("-5")

.invoke("css", "color")

.should("eq", "rgb(255, 0, 0)");

});

import { defineConfig } from "cypress";

import { addCucumberPreprocessorPlugin } from "@badeball/cypress-cucumber-preprocessor";

import { preprocessor } from "@badeball/cypress-cucumber-preprocessor/browserify";

async function setupNodeEvents(

on: Cypress.PluginEvents,

config: Cypress.PluginConfigOptions

): Promise<Cypress.PluginConfigOptions> {

// This is required for the preprocessor to be able to generate JSON reports after each run, and more,

await addCucumberPreprocessorPlugin(on, config);

on(

"file:preprocessor",

preprocessor(config, {

typescript: require.resolve("typescript"),

})

);

// Make sure to return the config object as it might have been modified by the plugin.

return config;

}

export default defineConfig({

e2e: {

specPattern: "\*\*/\*.feature",

setupNodeEvents,

supportFile: false

},

});