**Objective:** Automation testing in JavaScript using BDD cucumber framework.

**Background:** Software being developed must be as per the need and requirements of the Problem domain being analyzed and design. An important phase of software development is testing. A Test Driven Development (TDD) is a technique to develop the code and test it for its correctness and quality.

Behavior Driven Development (BDD) is an extension to TDD. This tests the developed product instead of testing the code. A product developed must work according to specification and must be user friendly and robust.

**Introduction to Cucumber:** It cannot be simply downloaded and cannot directly start automating test cases. It is a process required to use Cucumber known as Behavior-Driven Development (BDD).

This process has three main people involved. One, who manages the prerequisites like a Business Analyst or a Product Owner, Second, an individual who designs executes and creates it like a developer. The third primary person test and guarantee it.

**Cucumber Basics:**

**1. Feature Files:**

These are the typical test case files in a BDD process. This is a living document to the business necessity and a bridge between the requirement documentation and implementation part of it.

The syntax/format in which these feature files are written is called Gherkin. These files should have an extension ‘.feature.’

**Scenario:** Here the expectations of the test case or scenario are specified. It clarifies what is being checked and in what conditions.

**Given:** This specifies the conditions required to perform the test. Multiple conditions are joined by the And Clause.

**When:** This statement begins with the when clause specifies the steps to execute specified in the test document.

**Then:** It specifies clause that represents the verification steps and the expected result of the scenario.

**URL to be tested:** <https://budget.modus.app>

**Requirement 1**: Write a test plan for the application. This plan should include appropriate test coverage with both negative and positive test scenarios. Be sure to include the plan in your `README.md` file and try to be as thorough as possible.

Feature: checking the Item Add for budget

When category, description and value is provided, add button click should add item to the list and update the budget

Scenario: Item added to list and budget is updated

Given I am on the site https://budget.modus.app

When clicked on add, Item is added to list

Then budget is not updated, It should Error

**Second Scenario:**

Feature: checking the Empty Items Add for budget

When category, description and value is not provided, one should be not able to add item to the list

Scenario: Empty Items added to list and budget is updated

Given I am on the site https://budget.modus.app

On should not be able to click the Add Button

**Requirement 2** Select \*\*3\*\* test scenarios from your test plan to automate. Be sure to include appropriate directions in your `README.md` file for running your tests.

Given(^/ I am on <https://budget.modus.app/#/budget/>)

Visit Home Page

End

When(^/I attempt to click button without entering Items description/)

On(Homepage).select\_Category)

Leave description and amount as empty

End

Then(^/Then I should not be able to enter Item Information /)

Add button is disabled

End

Given(^/ I am on <https://budget.modus.app/#/budget/>)

Visit Home Page

End

When(^/I attempt to click button after entering Items description/)

On(Homepage).select\_Category)

Enter description and amount

End

Then(^/Then I entered Item Information should be added to the list /)

Add button is clicked

Item should be added to the list

Budget should be updated

End

**Requirement 3**: Text execution should spin up a Chrome browser and include at least one scenario that fails. Don't forget to include appropriate commentary for the failing test.

The given URL is on a Chrome browser and has not come across any scenario that fails. All inputs are being validated and button enable only when input is provided. In the value textbox, only numbers can be entered which again ensures correctness. The budget is updated with each entry and reports displays updated data.

Requirement 4\* Incorporate Cucumber into your tests \* Include the ability to generate an HTML report of the test results \* Organize test execution using either `npm` or `rake` depending on the test framework chosen \* Use a design pattern to structure your tests.

**Requirement 4**: Incorporate Cucumber into your tests.

The following shows the output from eclipse when Cucumber is incorporated in the test.

