"*Sadguru's Amrit-Tulya - Tea Shop*" needs you to test their website that helps track the different items they have for sale.

The functional requirement includes Frontend and Backend specifications.

**Frontend Requirements:**

Create a page with a component at the top that can be used to add a new item to the inventory. The component should require a name, description, and price for the new item. You can upload a picture of the item as well. Below the component that adds new items to the inventory, display a list containing the name and price of each item in the inventory. If your app supports image uploads, you can display a thumbnail here as well. Each item in the list should have a button to remove it from the inventory. Clicking on an item should bring the user to a new page with all the information about the selected item (name, description, price, and optionally photo).

All API calls should be asynchronous and not cause any full-page reloads. The only full-page reloads the user should see are those caused by navigating between the two pages of the app

Regarding visual styles, the only requirement is that the content of both pages should be both horizontally and vertically centered.

**Backend Requirements:**

All of the functions of the frontend needs to be served by an API call and the store’s inventory should be persisted across restarts of the backend process. It will use a SQL database for any persistent data storage needs.

**Tracking:**

Please do your best to keep track of how long you spend on the functional analysis of each of the backend, frontend functionality, the frontend presentational aspects, and Test Case creation.

**QA Requirements:**

The deliverable shall be shared via **GitHub, BitBucket, SourceTree, or any versioning System**containing all relevant Test Cases in the correct sequence.

**Evaluation and Scoring:**

The design will be simple and minimal. You are free to ask any questions for clarification or help.

You will be evaluated on the below attributes -

- Test Case completeness of Frontend requirements

- Test Case completeness of Backend requirements

- Identify and define Test Cases for automation

**Next Steps:**

1. Read the requirement thoroughly. Please keep a copy of it for a reference if needed.

2. Send the deliverable to [***deep@thinkbridge.in***](mailto:deep@thinkbridge.in)***,***[***shivani@thinkbridge.in***](mailto:shivani@thinkbridge.in)in the **next 1 week**via **GitHub, BitBucket, or SourceTree**. If not received within the specified time, we will consider it as an un-attempted question.

3. Click on the Submit button of this test.

**Test Cases:**

**Frontend Testcases:**

1. Verify that username, password fields and Login button are displayed on Login page
2. Verify that user able to login only with valid user
3. Verify the login validation for invalid user
4. Verify the page title after successful login.
5. Verify the logos are present in header of the page.
6. Verify title text displayed in top-centre of the page.
7. Verify that add new item to inventory button is displayed.
8. Verify that user is able to add new item to inventory
9. Verify that name, description and price fields are displayed after clicking on button add new item to inventory button
10. Verify that validation is working properly for name, description and price text fields.
11. Verify that ‘Upload Picture’ button is displayed.
12. Verify that user is able to add/upload a picture for a new item
13. Verify the position of add component button in web application.
14. Verify the list containing the name and price of each item in the inventory is displayed on page.
15. Verify that items are displaying in the list after adding a new item
16. Verify thumbnail present on page for respective component (Thumbnail count equals to total number of components in database).
17. Verify the number of items in the inventory displayed when page loads.
18. Verify that user navigates to new page after clicking on respective thumbnail and display specified information (name, description, price, and optionally photo).
19. Verify that new page loads with name, description, price, and optionally photo match with items details on inventory page
20. Verify information displayed on the new page is accurate.
21. Verify each item in the list have a button to remove it from the inventory and is displayed.
22. Verify if the inventory list is empty then remove button is disabled.
23. Verify that items get removed after clicking on ‘Remove’ button
24. Verify if the back button navigate user to the homepage.
25. Verify API call does not cause full-page reloads and loss of information.
26. Verify the only full-page reload the inventory and cart component count is intact.
27. Verify the components on both pages are horizontally and vertically centred.

**Backend Testcases:**

1. Verify respective API call is getting triggered.
2. Verify the store’s inventory is persisted across restarts of the backend process.
3. Verify that all data entry is correct in respective tables and columns for all the fields
4. Verify that the data is retrieving properly and displayed on the page from correct tables

**Automation Testcases:**

As there is no such challenging scenario, I can see in the functional scenario list we can uptake all the functional scenarios for automation.

We can group the test case page wise, implement page object model and automate them.

Groups can be as follows:

1.Login Page

2.HomePage [Where user can see a button to add item to inventory]

3.Inventory Page

4.Item Description Page