Challenge
Interact With Application Servers Using HTTP Protocol

Challenge

- Persist the order details using Axios API for the Veggie Pizza Outlet

Points to Remember

- Apply form validation using HTML5 built-in validations for form field elements used inside the customer registration form.
- Use a custom function for validating the password and confirm password form fields in the **customer.js** file:
  - on input event attribute can be used with confirm password field to call the function.
- Validation error messages should be simple, crisp, clear and in red fonts.
- Menu items should be filtered when a particular category is selected while viewing the menu details.
- Errors can be caught using catch block.

Instructions for Accessing the Boilerplate

- Click here for the boilerplate.
- Please read the README.md file provided in the boilerplate for further instructions about the practice exercise.
- Fork the boilerplate into your own workspace.
- Clone the boilerplate into your local system.
- Open command terminal run the command **`npm install`** to install `Mocha` and `Chai` as dependencies.
- Open the folder containing the boilerplate code in VS Code.
- Provide the solution code in the files specified with the task details.

Instructions for the Challenge

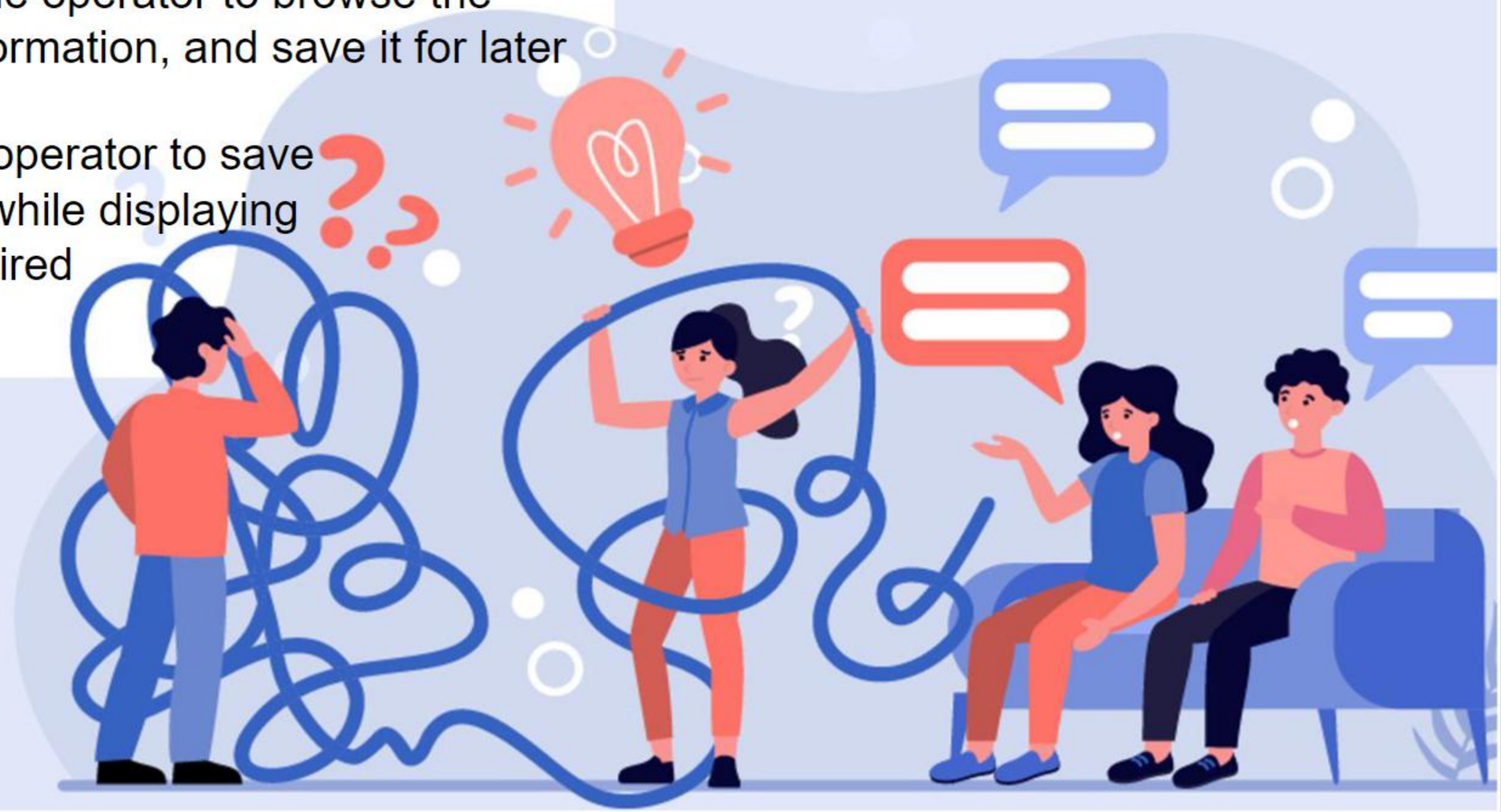
- The Challenge of this sprint is in continuation to the Challenge of the previous sprint `Sprint 5 - Develop Interactive Web Pages Using DOM and DOM Events`.
- Modify the existing code to fulfil the requirements stated with the tasks.
- The unzipped code contains **index.html, customer.html, menu.html and order.html** file which contains the partial design code of the `Veggie Pizza` web application.
- Write CSS code in the files located inside **css** folder of the boilerplate to style the web page to get the expected output.
- Write JavaScript code in the files located inside **js** folder of the boilerplate as per the requirements stated in the upcoming slides.
- Start json-server for menu, customer and order to manipulate data from **menu.json**, **customer.json, order.json** respectively located under json folder.
- Use different port numbers for each `json-server`. Eg. 3000, 3001 and 3002.
- Open the **index.html** file using Live Server and test the output.
- Submit the files for evaluation.

Persist the Order Details Using Axios API for Veggie Pizza Outlet

Veggie Pizza, a pizza delivery outlet, became popular for its custom-made vegetarian pizzas. By calling the pizza outlet, a consumer can get pizza and other products. The retailer sends the order to the address that each consumer specifies. The outlet has a new software that enables the phone operator to browse the menu information, enter the order information, and save it for later use.

Write a JS program that enables the operator to save client information and ordered items while displaying the total amount the customer is required to pay.

CHALLENGE



Tasks

- The challenge can be performed by the steps given below:
  - Step 1: View Menu Details
  - Step 2: Register a new customer
  - Step 3: Persist the Order

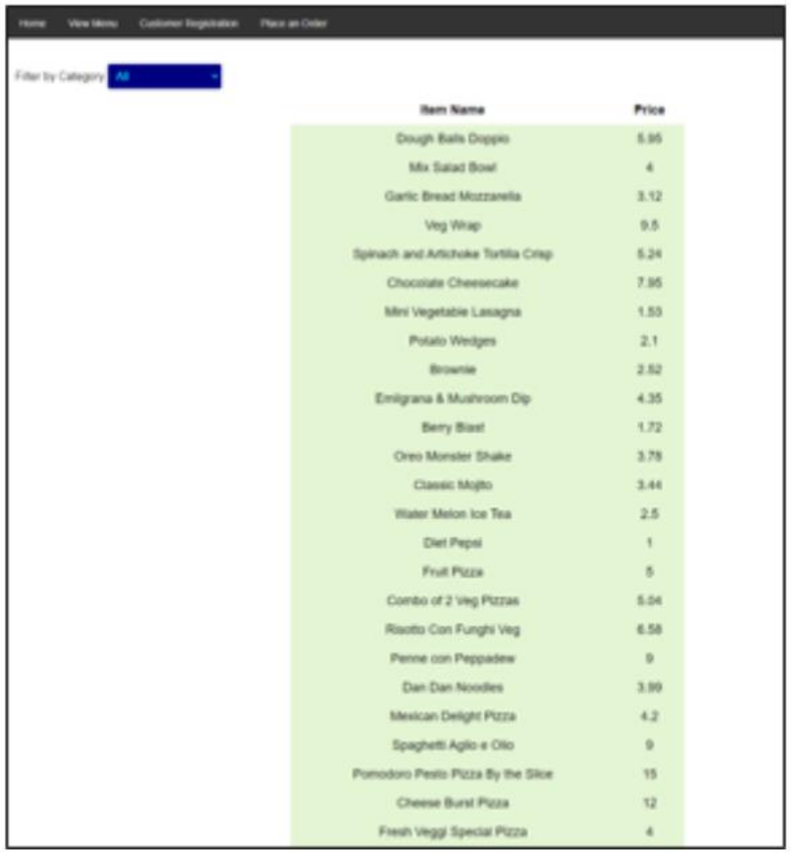
Note: Details about these steps are given in the upcoming slide.

Step 1 – View Menu Details

- The outlet team should be able to view the entire menu details to place an order.
- Open the file **menu.html** and add HTML code to display the menu items in a tabular format.
- On clicking the "View Menu", the menu details should be fetched from the server using Axios API.

  Note: As per test requirement, the Menu API should be running on port 3000.
- Selected menu items should be listed when a particular category is selected, e.g., starters:
- Edit **menu.js** file to
  - add a function which fetches the menu data from the server and display it.
  - add a function to filter the menu items using Array's `filter()` method based on a particular category and display them.

Expected Output: For Reference



Step 2 – Register a New Customer

- The pizza outlet team should register a new customer as per the details listed in the table.
- Edit the file **customer.html** to apply the listed HTML5 validations on the registration form fields which are available in the boilerplate code.

Input Field	Validation Requirement
Customer Id	Should not be left blank
Customer Name	Should not be left blank
Password	Should not be left blank and have min 8 characters
Confirm Password	Should be same as password value
Customer Email	Should not be left blank and should allow input of type email
Customer Phone	Should not be left blank and should allow valid phone number pattern
Customer Address	Should not be left blank

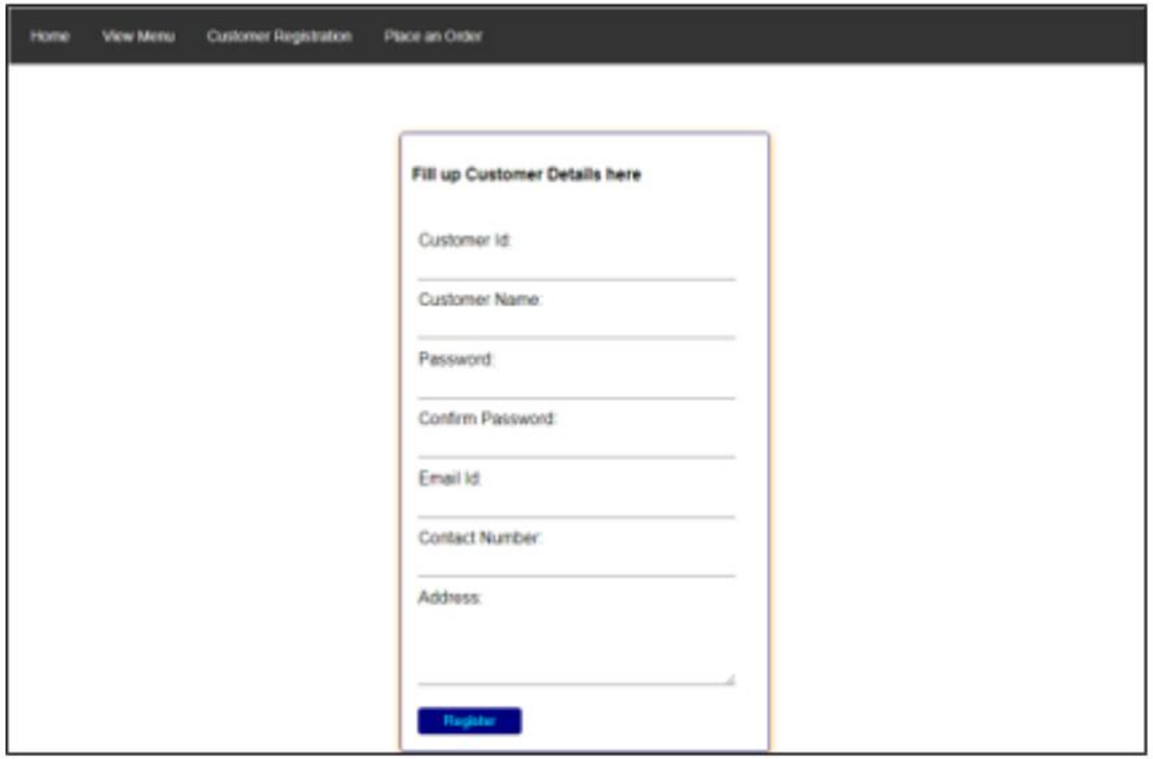
Step 2 – Register a New Customer (Cont'd)

- Edit **customer.js** file to
  - add custom validation functions which displays custom error messages for the field inputs that do not fulfil the validation criteria.
  - save valid customer details in the `json-server` using Axios API.

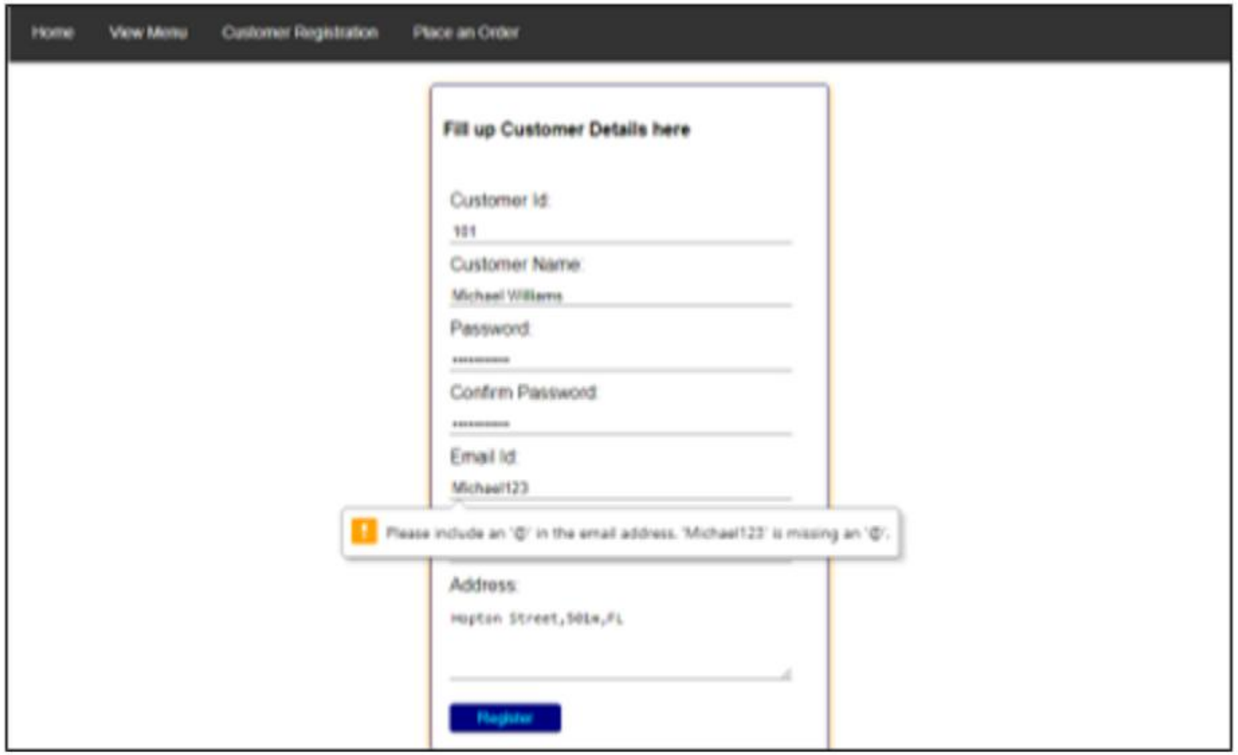
    Note: As per test requirement, the Customer API should be running on port 3001.
  - Once the customer details are saved, it should display the text `"You have successfully registered !"` on the page.

    Note: This text message will be used to evaluate test case and hence any mismatch between the actual and expected values will result into test case failure.

Expected Output: For Reference



Registration Form



Registration Form With Validations

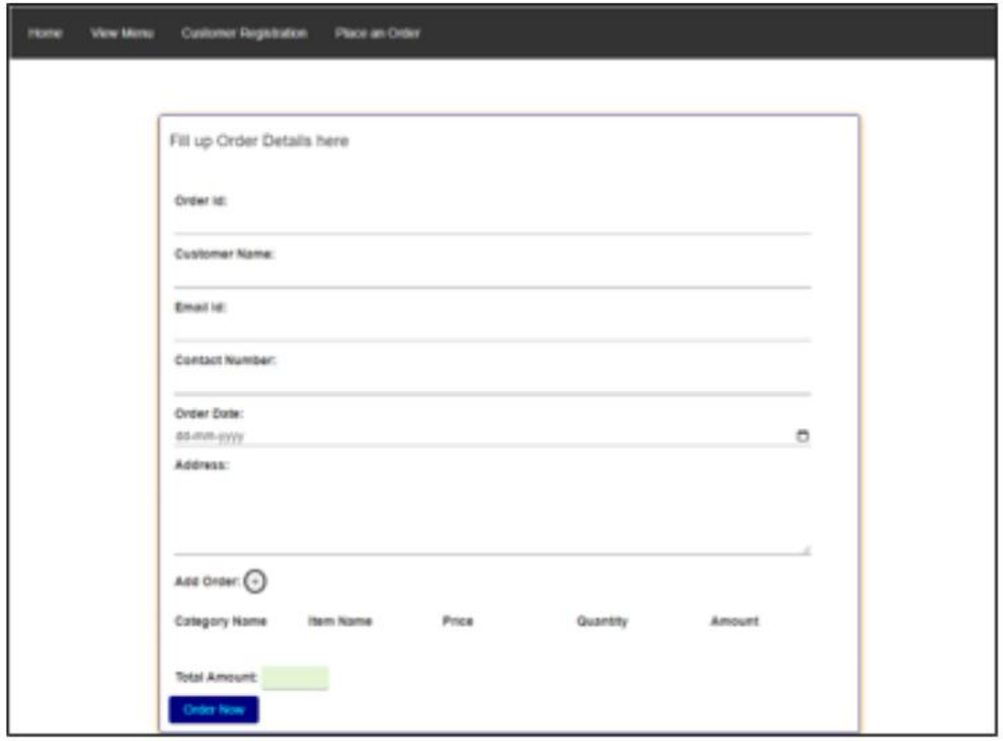
Step 3 – Persist the Order

- The order form created in the previous Sprint challenge should be reused in the file **order.html** of the boilerplate code.
- Edit the file **order.js** to reuse the solution developed in the previous sprint challenge and modify the code to persist the order details.
- As the user enters order items, the text field for “Total Amount” should get continuously updated with the total for all order items.
- When the user clicks on “Order Now” button, the complete order with all the details should be captured.
- Persist the captured order details in the `json-server` using Axios API calls.

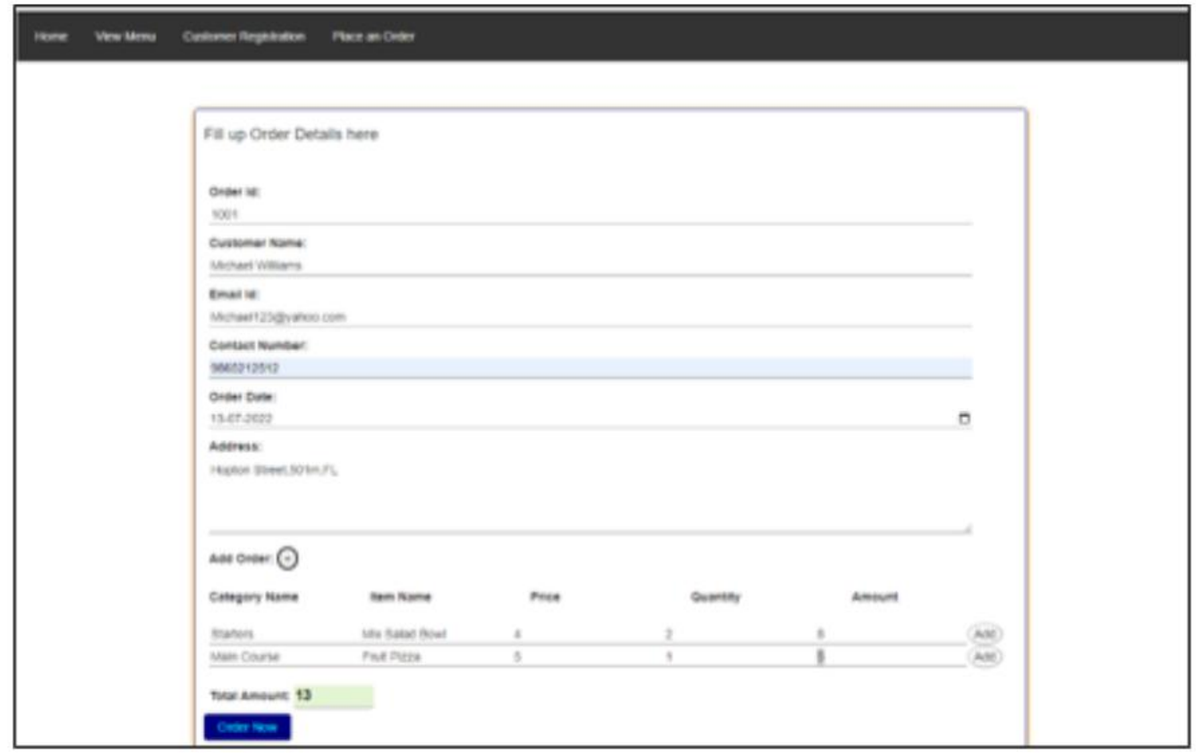
  Note: As per test requirement, the Order API should be running on port 3002.
- Once the order details are saved, the app should display the text `"Total amount to be paid: $<total-amount-value>"` on the page.

Note: This text message will be used to evaluate test case and hence any mismatch between the actual and expected values will result into test case failure.

Expected Output: For Reference



Order Form



Order Form With Items