VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering
ITE1002 – Web Technologies Laboratory
Assessment-3

Using JSON, AngularJS and AJAX

1. Develop a JSON program for creating 20 students personal information using the following fields as a inputs and print the outputs in table format (Use any Javascript/JQuery/AngularJS).
   student name, date_of_birth(date,month,year),parents(fathers_name,mothers_name), blood_group,email,phone(landline,mobile),address(door_no,street_name,place_name,pincode, country),degree(ug,pg and others).

2. Develop an angular shopping application to buy books and remove if not required.
   If the user enters a book in the text box and clicks addbook button, the book is added and if the user clicks on cross symbol next to each book given in the picture below, the book is removed from the list.
   If the book is selected twice, display an appropriate error message. Use angular animations to add and remove items.

   Books

   Web Technologies X

   Operating System X

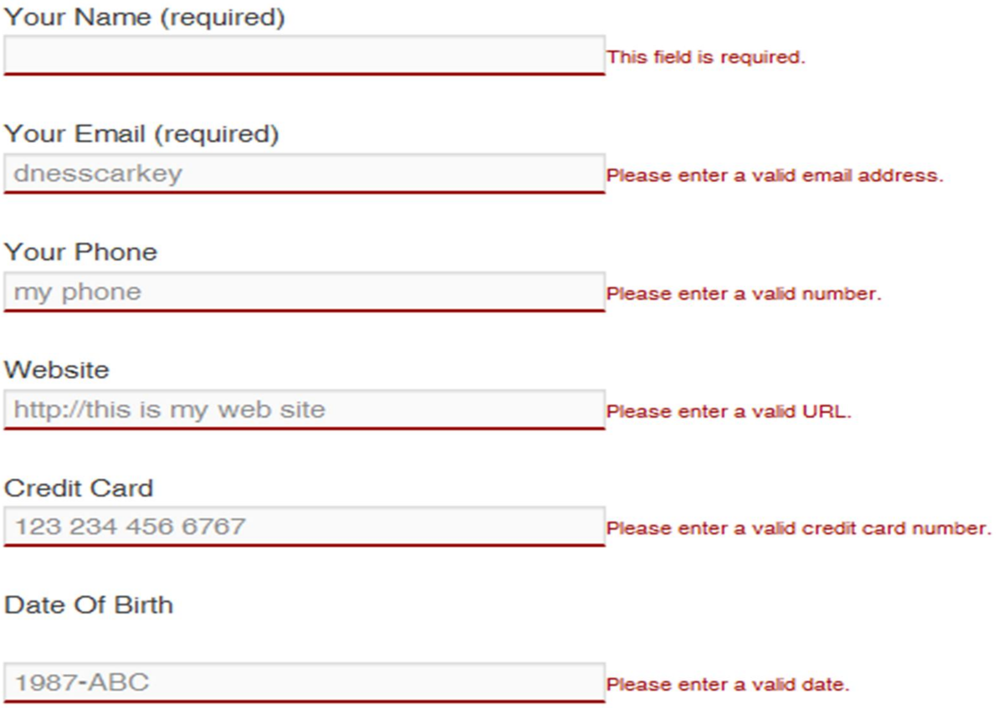
   Database systems X

   Computer Architecture X

   □ ADD BOOK

3. Product details are maintained in JSON file and gets it using AJAX. Develop the page contains a dropdown list which lists the value of product name. While clicking Availability button display the details of corresponding selected product as Product ID, name, price, stock and type.

4. Develop an angular application to validate the given form. Clearly mention the status of validation like required, invalid or valid message to the respective field.



Your Name (required)

This field is required.

Your Email (required)

dnesscarkey

Please enter a valid email address.

Your Phone

my phone

Please enter a valid number.

Website

http://this is my web site

Please enter a valid URL.

Credit Card

123 234 456 6767

Please enter a valid credit card number.

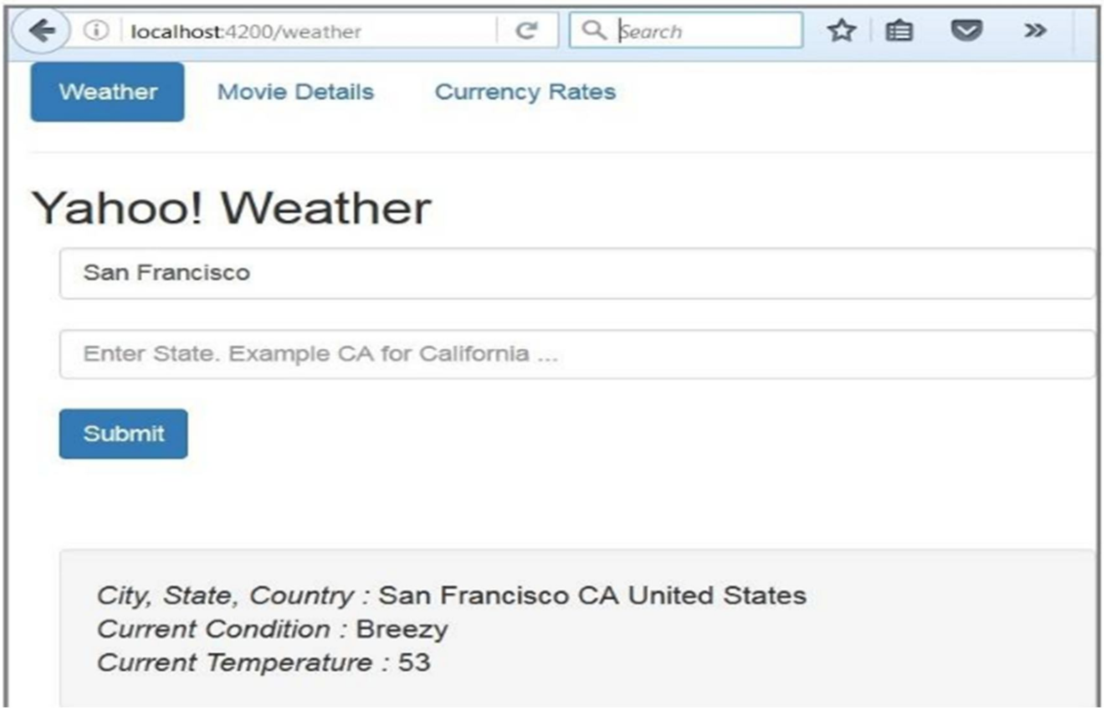
Date Of Birth

1987-ABC

Please enter a valid date.

5. Develop an Angular Single Page Application to show the details of weather, movie and currency rates of today. (Use templateURL)

Now, if you run the application and click the **Weather**, the app displays like below. Create a separate file weather.html and displayed it in a same page. Maintain weather details in JSON format. While clicking the submit button, your script display weather details for the corresponding city value which has entered in text field. Similar movie and currency details should be displayed in same page (Use template).



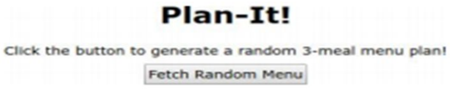
Yahoo! Weather

San Francisco

Enter State. Example CA for California ...

City, State, Country : San Francisco CA United States
Current Condition : Breezy
Current Temperature : 53

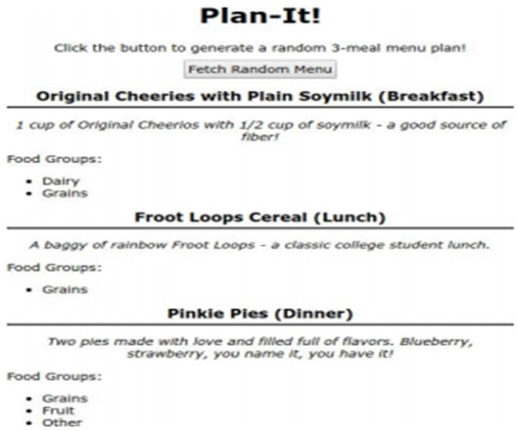
6. Develop an web application using AngularJS code for a small Meal Planner web page called Plan-It!. For simplicity, consider a ”full day meal plan” as a breakfast, lunch, and dinner meal types. Below are two screenshots of the Plan-It! page:



Plan-It!

Click the button to generate a random 3-meal menu plan!

Initial View



Plan-It!

Click the button to generate a random 3-meal menu plan!

Original Cheeries with Plain Soymilk (Breakfast)

1 cup of Original Cheerios with 1/2 cup of soymilk - a good source of fiber!

Food Groups:

- Dairy
- Grains

Froot Loops Cereal (Lunch)

A baggy of rainbow Froot Loops - a classic college student lunch.

Food Groups:

- Grains

Pinkie Pies (Dinner)

Two pies made with love and filled full of flavors. Blueberry, strawberry, you name it, you have it!

Food Groups:

- Grains
- Fruit
- Other

After Fetching Random Full-Day Menu

Provided HTML:

```
<body>
<h1>Plan-It!</h1>
<p>
Click the button to generate a random 3-meal menu plan!
<button id="day-btn">Fetch Random Menu</button>
</p>
<section id="day-results" class="hidden">
<article id="breakfast">
<h2><span class="name"></span> (Breakfast)</h2>
<p class="description"></p>
<p>Food Groups:</p>
<ul class="food-groups"></ul>
</article>
<article id="lunch">
<h2><span class="name"></span> (Lunch)</h2>
<p class="description"></p>
<p>Food Groups:</p>
<ul class="food-groups"></ul>
</article>
<article id="dinner">
<h2><span class="name"></span> (Dinner)</h2>
```

```
<p class="description"></p>
<p>Food Groups:</p>
<ul class="food-groups"></ul>
</article>
</section>
</body>
```

i. Clicking the #day-btn should fetch repeatedly a random full day meal plan (one option for each of breakfast, lunch, and dinner) from JSON object using **AngularJS.**

ii. The JSON format will be in the following format (example):

```
{ "breakfast" : {
       "name" : "Blueberry Oatmeal",
      "description" : "One cup of hot oatmeal with 1/4 cup of fresh blueberries.",
        "food-groups" : ["Grains", "Fruit"]   },
         "lunch" : {
         "name" : "Froot Loops",
          "description" : "A baggy of rainbow Froot Loops - a classic college
          student lunch.",
          "food-groups" : ["Grains"]      },
           "dinner" : {
           "name" : "Spaghetti with Tomato Sauce",
           "description" : "Two pies made with love and filled full of flavors.
             Blueberry, strawberry, you name it, you have it!",
             "food-groups" : ["Grains", "Fruit", "Other"] }}
```

iii. Use the above JSON to populate each of the #breakfast, #lunch, and #dinner elements on the page such that:

- The *span.name* is populated with the corresponding meal name.
- The *p.description* is populated with the corresponding meal description
- The *ul.food-groups* is populated with a list of the food groups for that item (one or more food groups may be in the ”food-groups” array in the JSON).

7. Develop a web application using AngularJS add behavior to the following HTML code. The page contains a text input box with an id of foodname, and a drop-down list with an id of foodgroup.
    i. The user types a name of a food item into the foodname text box, such as apple or Cookie, selects a food group from the drop-down list, such as dairy or fruit, and then clicks the button with an id of eat.
    ii. When the eat button is clicked, any element on the page that matches all of the following criteria will be removed from the page:
        a. The element is an img element that has a class of the food.
        b. The element's food group matches the group chosen in the foodgroup drop-down list. Food groups are represented as class attributes. This is in addition to the food class; recall that class attributes can specify multiple classes separated by spaces. For example, a jug of milk would have the following element:

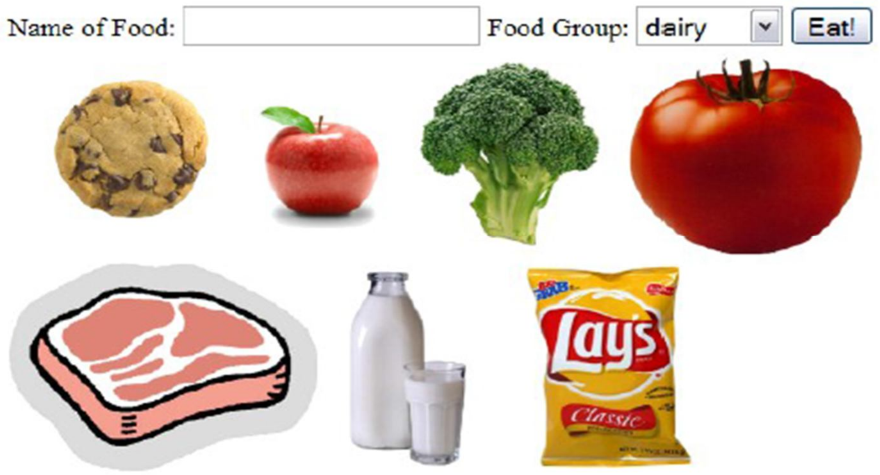
            <img src="milkjug.jpg" class="food dairy" alt="milk" />

        c. The food item's name is the same as the text in the foodname box. The food's name is stored as the image's alt attribute. For example, if the user types cookie, img elements with an alt of cookie will be removed. Your code should be case-insensitive; for example, coOKIe should match images with an alt of cookie.

The relevant HTML code for the page is the following.

```
<div>
Name of Food: <input id="foodname" type="text" />
Food Group:
<select id="foodgroup">
<option>dairy</option>
<option>fats</option>
<option>fruit</option>
<option>meat</option>
<option>veggies</option>
</select>
<button id="eat">Eat!</button>
</div>
<p>
<img src="cookie.jpg" class="food fats" alt="cookie" />
<img src="apple.jpg" class="fruit yummy food" alt="apple" />
<img src="broc.jpg" class="food veggies" alt="broccoli" />
<img src="tomato.jpg" class="fruit food" alt="tomato" />
<img src="beefsteak.jpg" class="food meat" alt="steak" />
<img src="milkjug.jpg" class="food dairy" alt="milk" />
<img src="potatochips.jpg" class="fats food other" alt="chips" />
</p>
```

Assume that Prototype is also included in the page.



Name of Food:

Food Group: dairy