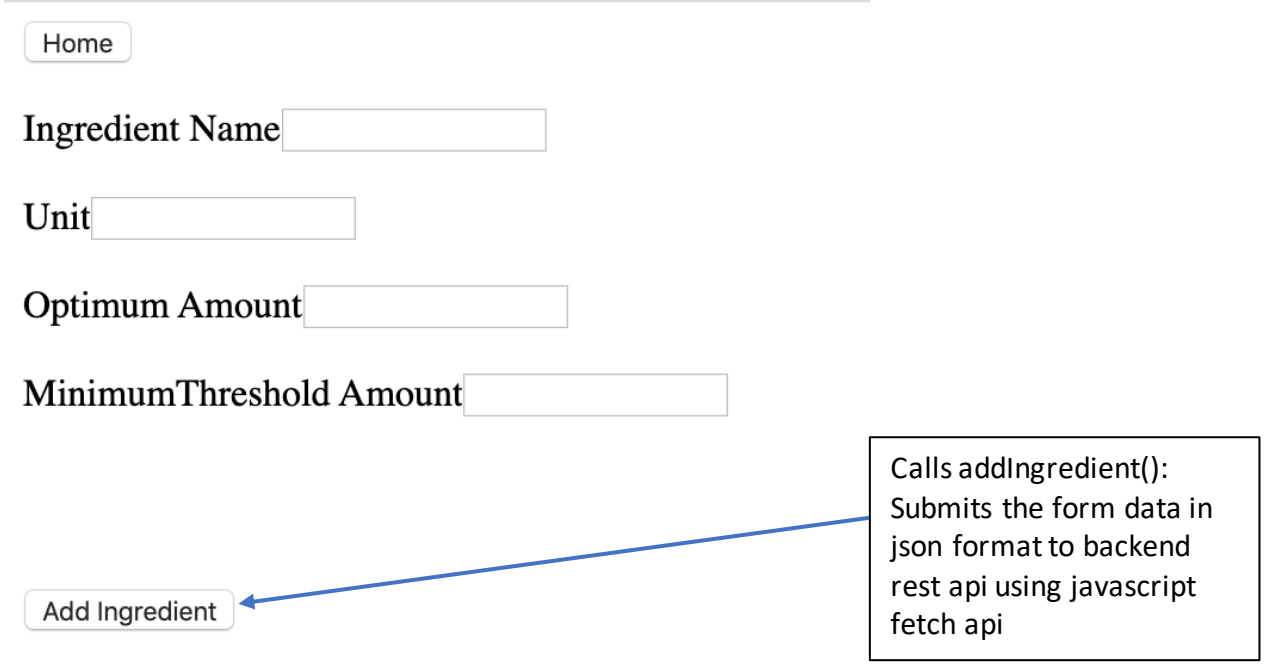
Criterion C: Development

List of techniques:

Data Entry forms

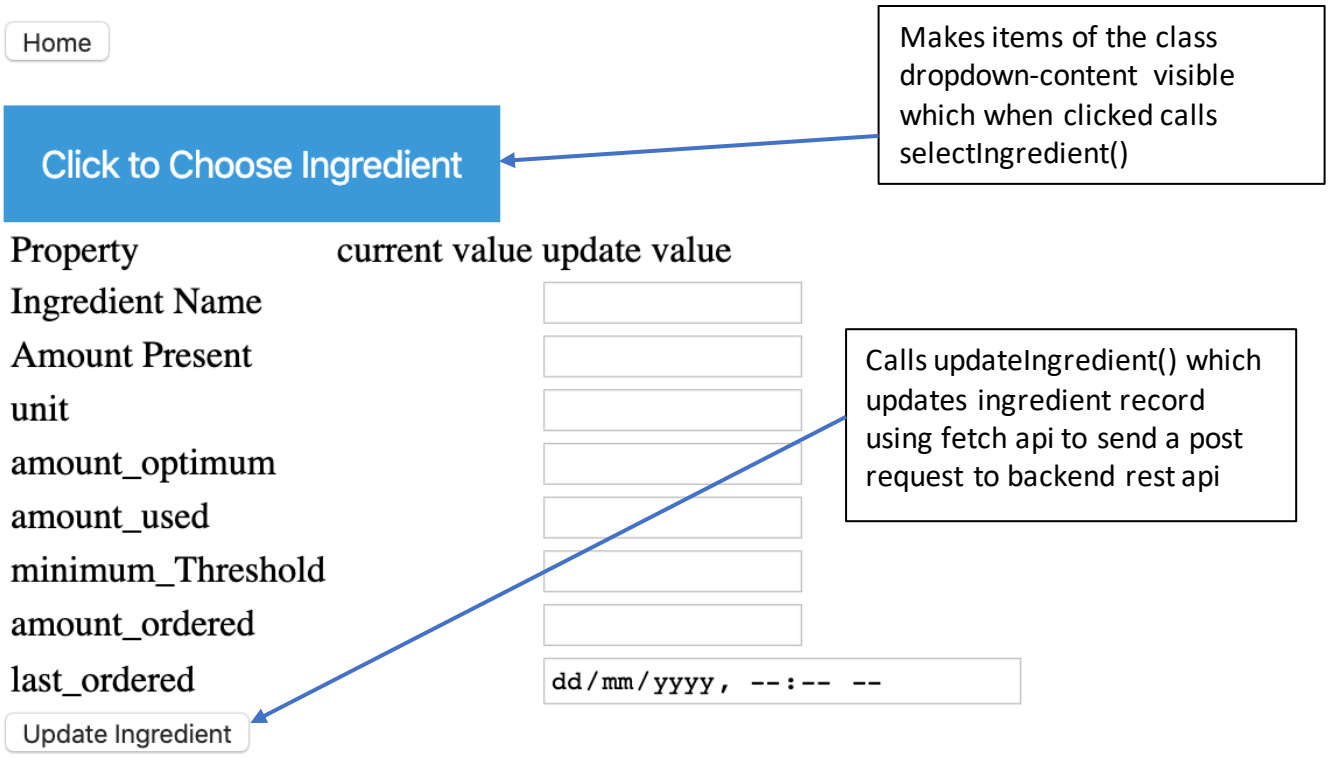
Make handling the inventory easier by making data entry forms using django html templates for webpage content and javascript for interactivity and form submission by fetch api which targets backend api developed using django rest framework.




```
{% load static %}
<a href="{% url 'home' %}"><button>Home</button></a><br><br>
<head>
<script src="{% static "InventoryManagementSystem/newIngredient.js" %}"
alt="newIngredient.js"></script>
</head>
<body>
Ingredient Name<input type="text" id="ingredient_name"><br><br>
Unit<input type="text" id="unit"><br><br>
```

```
Optimum Amount<input type="number" id="amount_optimum"><br><br>
MinimumThreshold Amount<input type="number" id="minimum_Threshold"><br><br>
<br><br><br>
<button onclick=addIngredient()>Add Ingredient</button>
```

```
async function addIngredient(){
        //get data from DOM
        var ingredient_name= document.getElementById("ingredient_name").value;
        var amount_optimum= document.getElementById("amount_optimum").value;
        var minimum_Threshold= document.getElementById("minimum_Threshold").value;
        var unit=document.getElementById("unit").value;
        //prepare json data
        data={
                "ingredient_name": ingredient_name,
                "amount_present": 0,
                "unit": unit,
                "amount_optimum": amount_optimum,
                "amount_used": 0,
                "minimum_Threshold": minimum_Threshold,
                "amount_ordered": 0,
                "last_ordered": new Date(),
                "category": "none"
        }
        //call backend rest api using fetch, show error if it fails
        const url = 'http://localhost:5000/manage/ingredients/';
        try {
                const response = await fetch(url, {
                        method: 'POST',
                        body: JSON.stringify(data),
                        headers: {
                                'Content-Type': 'application/json',
                        }
                });
                const json = await response.json();
                console.log('Success:', JSON.stringify(json));
                window.alert('Success:', JSON.stringify(json))
        } catch (error) {
                console.error('Error:', error);
                window.alert('Error:', error);
        }
}
```




```
<a href="{% url 'home' %}"><button>Home</button></a><br><br>
{% load static %}
<head>
<link rel=stylesheet href="{% static "InventoryManagementSystem/dropdown.css" %}">
<script src="{% static "InventoryManagementSystem/updateIngredient.js" %}"></script>
</head>
<body>
	<div class="dropdown">
		<button onclick="myFunction()"  class="dropbtn">
			Click to Choose Ingredient
		</button>
		<div id="ingredientDropdown"  class="dropdown-content">
			{% for ingredient in ingredients_list %}
				<div onclick=selectIngredient("{{ingredient.ingredient_name}}")>
					{{ingredient.ingredient_name}}
				</div>
			{% endfor %}
		</div>
	</div>
	<div id="selectedIngredient"></div>
	<table>
		<tr>
			<td>Property</td>
			<td>current value</td>
			<td>update value</td>
		</tr>
		<tr>
```

```
                        <td>Ingredient Name</td>
                        <td><div id="current_ingredient_name"></div></td>
                        <td><input type="text" id="new_ingredient_name"></td>
                </tr>
                <tr>
                        <td>Amount Present</td>
                        <td><div id="current_amount_present"></div></td>
                        <td><input type="number" id="new_amount_present"></td>
                </tr>
                <tr>
                        <td>unit</td>
                        <td><div id="current_unit"></div></td>
                        <td><input type="text" id="new_unit"></td>
                </tr>
                <tr>
                        <td>amount_optimum</td>
                        <td><div id="current_amount_optimum"></div></td>
                        <td><input type="number" id="new_amount_optimum"></td>
                </tr>
                <tr>
                        <td>amount_used</td>
                        <td><div id="current_amount_used"></div></td>
                        <td><input type="number" id="new_amount_used"></td>
                </tr>
                <tr>
                        <td>minimum_Threshold</td>
                        <td><div id="current_minimum_Threshold"></div></td>
                        <td><input type="number" id="new_minimum_Threshold"></td>
                </tr>
                <tr>
                        <td>amount_ordered</td>
                        <td><div id="current_amount_ordered"></div></td>
                        <td><input type="number" id="new_amount_ordered"></td>
                </tr>
                <tr>
                        <td>last_ordered</td>
                        <td><div id="current_last_ordered"></div></td>
                        <td><input type="datetime-local" id="new_last_ordered"></td>
                </tr>
        </table>
        <button onclick=updateIngredient()>Update Ingredient</button>
</body>
```

```
async function updateIngredient(){
//get data from DOM
  var current_ingredient_name=document.getElementById("current_ingredient_name").innerHTML;
  var ingredient_name= document.getElementById("new_ingredient_name").value;
  var amount_present=document.getElementById("new_amount_present").value;
```

```
  var unit=document.getElementById("new_unit").value;
  var amount_optimum=document.getElementById("new_amount_optimum").value;
  var amount_used=document.getElementById("new_amount_used").value;
  var minimum_Threshold=document.getElementById("new_minimum_Threshold").value;
  var amount_ordered=document.getElementById("new_amount_ordered").value;
  var last_ordered=document.getElementById("new_last_ordered").value;
//prepare JSON
  preData={
    "ingredient_name": ingredient_name,
    "amount_present": amount_present,
    "unit": unit,
    "amount_optimum":  amount_optimum,
    "amount_used": amount_used,
    "minimum_Threshold":  minimum_Threshold,
    "amount_ordered": amount_ordered,
    "last_ordered": last_ordered,
    "category": "none"
  }
//parse data into format accepted by the backend rest api
  postData={};
  for(key in preData){
    if (preData[key]===''){
      if (key === 'last_ordered'){
        d=document.getElementById(`current_${key}`).innerHTML
        ds=d.split(' ')
        date=ds[1]
        time=ds[0]
        da=date.split('/')
        ta=time.split(':')
        date=new Date(da[2],da[1]-1,date=da[0],hours=ta[0],minutes=ta[1])
        offset=(date.getTimezoneOffset())*60*1000
        newDate= new Date(date.getTime()+offset);
        postData[key]=date.toISOString();
      } else{
        postData[key]=document.getElementById(`current_${key}`).innerHTML;
      }
    }else{
      postData[key]=preData[key]
    }
  }
//post updated data to the backend rest api using javascript fetch show error if it fails
  const url = (
    `http://localhost:5000/manage/ingredients/${current_ingredient_name}/`);
  console.log({preData,postData});
  try {
    const response = await fetch(url, {
     method: 'PATCH',
     body: JSON.stringify(postData),
     headers: {
      'Content-Type': 'application/json',
     }
    });
    const json = await response.json();
    console.log('Success:', JSON.stringify(json));
    document.location.reload();
   } catch (error) {
       console.error('Error:', error);
```

```
    }
}

async function getIngredientDetails(ingredient_name){
  // get data using fetch
  const url = (
    'http://localhost:5000/manage/ingredients/?'
    + new URLSearchParams({
      ingredient_name: ingredient_name,
      })
    );
  try {
    const response = await fetch(url, {
     method: 'GET',
     headers: {
      'Content-Type': 'application/json',
     }
    });
    const json = await response.json();
    results=json['results'][0]
    // set data in the dom
    for(key in results){
      var el=document.getElementById(`current_${key}`);
      if(key==='last_ordered'){
        console.log(key)
        var d=new Date(results[key]);
        el.innerHTML=`${d.toLocaleTimeString()} ${d.toLocaleDateString()}`;
      } else if(!(key==='category' || el===null)){
        console.log('out',key);
        el.innerHTML=results[key];
      }
    }
   } catch (error) {
      console.error('Error:', error);
   }

}
async function selectIngredient(ingredient_name){
  await getIngredientDetails(ingredient_name);
 }
function myFunction() {
  document.getElementById("ingredientDropdown").classList.toggle("show");
 }
window.onclick = function(event) {
if (!event.target.matches('.dropbtn')) {
  var dropdowns = document.getElementsByClassName("dropdown-content");
  var i;
  for (i = 0; i < dropdowns.length; i++) {
  var openDropdown = dropdowns[i];
  if (openDropdown.classList.contains('show')) {
    openDropdown.classList.remove('show');
  }
  }
}
}
```

Gets ingredient record and updates the DOM

Used for implementing dropdown

Data tables

Keep detailed records of the inventory by having tables with all its properties

Details of the proportion of ingredients for each dish to be made available in the system

Home

Ingredients

Search...

Ingredient Name	Amount Present	Amount Optimum	Amount Used	Minimum Threshold	Amount Ordered	Last Ordered
Chicken	50.0 kg	25.0	10.0	5.0	5.0	Feb. 16, 2020, 7:29 p.m.
Potato	46.0 Kg	2.0	68.0	1.0	0.0	March 7, 2020, 3:11 p.m.
garlic	7.0 kgs	6.0	0.0	3.0	0.0	March 30, 2020, 9:02 p.m.
herbs	5.0 kgs	4.0	1.0	2.0	0.0	March 30, 2020, 9:27 p.m.
mushroom	8.0 kgs	6.0	0.0	3.0	0.0	March 30, 2020, 9:10 p.m.
onions	10.0 kgs	6.0	0.0	3.0	0.0	March 30, 2020, 9:14 p.m.
sweet potato	0.0 kgs	6.0	0.0	3.0	0.0	March 30, 2020, 9:06 p.m.
tomato	0.0 kg	50.0	0.0	25.0	0.0	March 30, 2020, 7:22 p.m.

For all purposes the webpages are prepared on the server side by using templates & data from the database. For example in the code below, the template is specified in an html document by using curly braces. Single curly braces allow for looping and branching, while double braces allow for using database record data. Similarly templates for the other webpages were prepared.

```
<a href="{% url 'home' %}"><button>Home</button>/a><br><br>
<h1>Ingredients</h1>
<form action="{% url 'ingredient_search' %}" method="get">
 <input name="q" type="text" placeholder="Search...">
</form>
<table style='border: 1px sloid black;'>
   <tr style='border: 1px solid black;'>
       <td style='border: 1px  black;'>Ingredient Name</td>
       <td style='border: 1px  black;'>Amount Present</td>
       <td style='border: 1px  black;'>Amount Optimum</td>
       <td style='border: 1px  black;'>Amount Used</td>
       <td style='border: 1px  black;'>Minimum Threshold</td>
       <td style='border: 1px  black;'>Amount Ordered</td>
       <td style='border: 1px  black;'>Last Ordered</td>
   </tr>
   {% for ingredient in ingredients_list %}
       <tr style='border: 1px solid black;'>
           <td style='border: 1px solid black;'>{{ ingredient.ingredient_name }}</td>
           <td style='border: 1px solid black;'>{{ingredient.amount_present}}</td>
           <td style='border: 1px solid black;'>{{ingredient.unit}}</td>
           <td style='border: 1px solid black;'>{{ingredient.amount_optimum}}</td>
           <td style='border: 1px solid black;'>{{ingredient.amount_used}}</td>
           <td style='border: 1px solid black;'>{{ingredient.minimum_Threshold}}</td>
           <td style='border: 1px solid black;'>{{ingredient.amount_ordered}}</td>
```

```
            <td style='border: 1px solid black;'>{{ingredient.last_ordered}}</td>
        </tr>
    {% endfor %}
</table>
```

Django Template for ingredient list

Keep detailed records of the orders tables with all its properties

Home

Orders

Search...

Order Id	Time	Name	Dish	Qty	Note
1585297315552-xsJ	March 27, 2020, 1:51 p.m.	ghxdfgdfg	potato stuffed chicken	1.0	fxgh
1585297315552-xsJ	March 27, 2020, 1:51 p.m.	ghxdfgdfg	potato chicken	2.0	fxgxfghfgh
1585577544823-NrA	March 30, 2020, 7:42 p.m.	ayaan	potato stuffed chicken	1.0	afdsfadsf
1585577544823-NrA	March 30, 2020, 7:42 p.m.	ayaan	potato chicken	2.0	afdsfadsfafsadsf
1585577577241-uco	March 30, 2020, 7:42 p.m.	ayaan	potato stuffed chicken	2.0	wdvcd
1585577577241-uco	March 30, 2020, 7:42 p.m.	ayaan	potato chicken	3.0	wdvczsvzdd
1585584033213-dFF	March 30, 2020, 9:30 p.m.	ayaan	mushroom stuffed chicken	2.0	note
1585584033213-dFF	March 30, 2020, 9:30 p.m.	ayaan	baked mushroom	1.0	note 1

A template similar to the one mentioned above for ingredient table was used to generate the order table and the corresponding search page. The order table is sorted according to order Id.

Maintain a list of recipes with tables

Home

Search...

Name:	Steps
baked mushroom	Slice mushrooms Place on baking tray and drizzle oil Sprinkle salt, pepper and herbs Bake for 20 minutes at 120 degrees Celsius
baked mushroom	Slice mushrooms Place on baking tray and drizzle oil Sprinkle salt, pepper and herbs Bake for 20 minutes at 120 degrees Celsius
baked mushroom	Slice mushrooms Place on baking tray and drizzle oil Sprinkle salt, pepper and herbs Bake for 20 minutes at 120 degrees Celsius
baked mushroom	Slice mushrooms Place on baking tray and drizzle oil Sprinkle salt, pepper and herbs Bake for 20 minutes at 120 degrees Celsius
mushroom stuffed chicken	Slice mushrooms Stuff chicken with mushroom filling Sprinkle salt, pepper and herbs Bake for 20 minutes at 120 degrees Celsius
potato chicken	boil potato boil chicken
potato stuffed chicken	grill chicken stuff potato
salsa	jdbheb

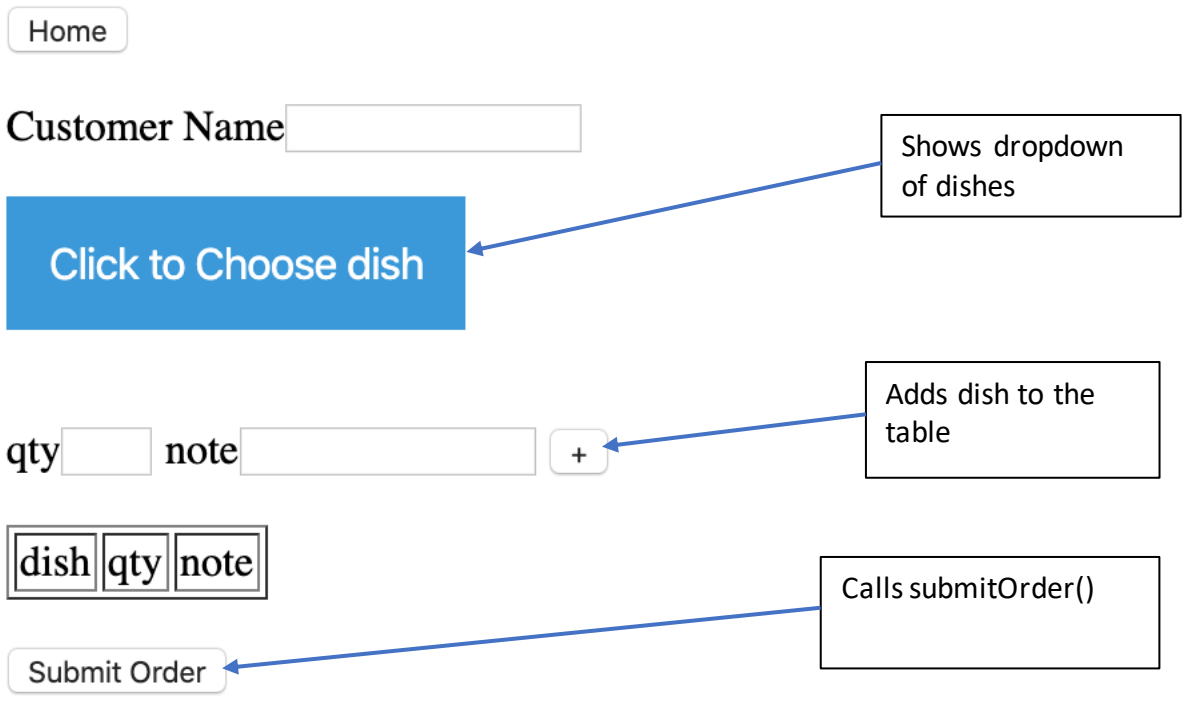
A template similar to the one mentioned above for ingredient table was used to generate the order table and the corresponding search page. The order table is sorted according to order Id.

Automatic update of inventory

Reduced need for human interference automatic update of inventory stock based on orders placed as when an order is successfully placed the ingredient record is updated automatically.

Placing an order

Provisions for making specifications for particular dish ordered is available through a form



The template for new order page has an empty table & a dropdown system as shown above for ingredient selection in the ingredient update page. All the functionality here is provided by the linked javascript files.

```
<a href="{% url 'home' %}"><button>Home</button></a><br><br>
{% load static %}
<head>
<link rel=stylesheet href="{% static "InventoryManagementSystem/dropdown.css" %}" >
<script src="{% static "InventoryManagementSystem/dropdown.js" %}"></script>
<script src="{% static "InventoryManagementSystem/AddOrder.js" %}"></script>
```

```
</head>
<body>
Customer Name<input type="text" id="name"><br>
<ul class="orderList"></ul>
<div class="dropdown">
<button onclick="myFunction()"  class="dropbtn">Click to Choose dish</button>
<div id="Recipe" class="dropdown-content">
        {% for recipe in recipe_list %}
                <div onclick="selectDishName('{{recipe.dish_name}}')">{{recipe.dish_name}}</div>
        {% endfor %}
</div>
</div><br><br>
<div id="selectedDish"></div><br>
qty<input type="number" min="1" max="15" id="qty">note<input type="text" id="note">
<button onclick="addOrder()">+</button><br><br>
<table id="orderTable"
border="solid"><tr><td>dish</td><td>qty</td><td>note</td></tr></table><br>
<button onclick="submitOrder()">Submit  Order</button>
</body>

function addOrder(){
  var note=document.getElementById("note").value;
  var qty=document.getElementById("qty").value;
  var dish=document.getElementById("selectedDish").innerHTML;
  var table=document.getElementById("orderTable");
  table.innerHTML=table.innerHTML+"<tr><td>"+dish+"</td><td>"+qty+"</td><td>"+note+"</td></tr>";
}
function makeId(size) {
 var text = "";
 var possible = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
 for (var i = 0; i < size; i++)
    text += possible.charAt(Math.floor(Math.random()  * possible.length));
 return text;
}
async function submitOrder(){
  const orderUrl = 'http://127.0.0.1:5000/manage/orders/';
  const recipeUrl = 'http://127.0.0.1:5000/manage/recipes/'
  var note;
  var dish;
  var qty;
  var table=document.getElementById("orderTable");
  var name= document.getElementById("name").value;
  if(name ===null || name===undefined){
   window.alert('Specify customer name')
   throw Error('specify customer name')
  }else if(name.trim()===''){
   window.alert('Specify customer name')
   throw Error('specify customer name')
  }
  //make order array
  var orderArray=[];
  for (i = 1; i < table.rows.length; i++) {
     var objCells = table.rows.item(i).cells;
```

Adds order to table using DOM manipulation

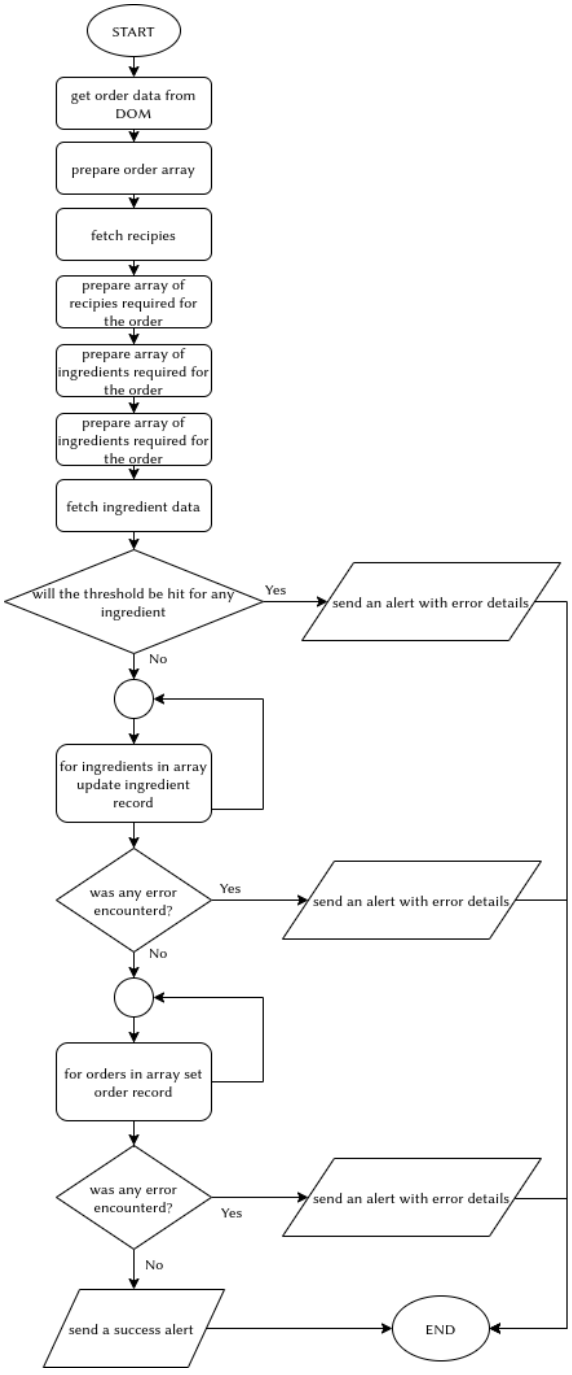
Makes a random alphanumeric order id

```
    dish=objCells.item(0).innerHTML;
    qty=objCells.item(1).innerHTML;
    note=objCells.item(2).innerHTML;
    orderArray.push({dish,qty,note,name});
  }
  //get recipie
  const recipeResponse = await fetch(recipeUrl);
  const recipeJson = await recipeResponse.json();
  recipeList= recipeJson['results'].filter((recipe)=>{
   for(order of orderArray){
    if (order.dish===recipe.dish_name){
     return true;
    }
   }
   return false;
  })
  //get ingredient list
  ingredientList=[]
  var valid;
  var ingredientListItem;
  var ingredientName;
  for(recipe of recipeList){
   for(i=1;i<=5;i++){
    ingredientName= recipe[`ingredient${i}_name`]
     valid = (ingredientName!==null && ingredientName!==undefined) &&  ingredientName.trim()!=='';
    if(valid){
     ingredientListed=false;
     index=null;
     for(j=0;i<ingredientList.length;i++){
      if(ingredientList[j].name===ingredientName){
       ingredientListed=true
       index=j;
      }
     }
     if(ingredientListed){
      ingredientList[index]={
       "name":ingredientList[index].name,
       "amount": ingredientList[index].amount +
                    (parseFloat(recipe[`ingredient${i}_amount`])*parseFloat(qty))
      };
     }else{
      ingredientListItem = {
          "name":ingredientName,
          "amount": parseFloat(recipe[`ingredient${i}_amount`])*parseFloat(qty)}
      ingredientList.push(ingredientListItem);
     }
    }
   }
  }
  ingredientResponses ={};
  ingredientJsons={};
  ingredientJsonsInitial={};
  ingredientDatas={};
  //check amount and prepare data for ingredient update
  for(ingredient of ingredientList){
   ingredientUrl= `http://localhost:5000/manage/ingredients/${ingredient.name}/`;
    initialData= await (await fetch(ingredientUrl)).json();
```

```
      if((initialData.amount_present-ingredient.amount)<initialData.minimum_Threshold){
        window.alert('not enough ingredients')
        throw Error('not enough ingredients')
      }else{
       ingredientDatas[ingredient.name]={
         "amount_present": initialData.amount_present-ingredient.amount,
         "amount_used": initialData.amount_used+ingredient.amount,
         };
      }
  }
  //update ingredients
  for (ingredient of ingredientList){
   try{
     ingredientResponses[ingredient.name] = await fetch(ingredientUrl,{
       method: 'PATCH',
       body: JSON.stringify(ingredientDatas[ingredient.name]),
       headers: {
         'Content-Type': 'application/json',
       }
     });
     ingredientJsons[ingredient.name] = await ingredientResponses[ingredient.name].json();
     if(ingredientResponses[ingredient.name].status===200
           ||ingredientResponses[ingredient.name].staus==201){
       console.log('Success:', JSON.stringify(ingredientJsons[ingredient.name]));
     }else{
       const json = ingredientJsons[ingredient.name] ;
       throw Error(ingredient.name+JSON.stringify(json))
     }
   } catch(error){
     window.alert(error);
     throw error;
   }
  }
  //add order
  var dateTime = new Date();
  var orderId = `${dateTime.getTime()}-${makeId(3)}`;
  for (var order of orderArray){
     try{
       orderData ={
         "dish_name": order.dish,
         "amount_ordered": order.qty,
         "order_time": dateTime,
         "note": order.note,
         "customer_name": name,
         "order_id": orderId
       };
       const orderResponse = await fetch(orderUrl, {
         method: 'POST',
         body: JSON.stringify(orderData),
         headers: {
           'Content-Type': 'application/json',
         }
       });
       if(orderResponse.status==200 || orderResponse.status==201){
         const json = await orderResponse.json();
         console.log('Success:', JSON.stringify(json));
         window.alert('order added');
```

```
        document.location.reload();
      }else{
        const json = await orderResponse.json();
        throw Error(JSON.stringify(json))
      }
    } catch (error) {
      window.alert(error);
      console.error('Error:', error);
    }
  }
}
```

Order submission flow chart



Searching

Enable easy searching of inventory, recipes and orders by providing a search field. On submitting the user is sent to a dedicated search page that has the table filtered according to the search term.

For sorting & filtering pythons built in functions were used. For sorting python uses Timsort which is a combination of merge sort & insertion sort. Certain merge criteria are used to decide when and where merge sort is used.
(source:https://svn.python.org/projects/python/trunk/Objects/listsort.txt)