

SWEN 502 Group Assignment

# Design and Planning Report

Group 3 - Bespoke Burgers

Callum, Danielle, Guy, Julian, Lian

## Table of Contents

<b>System Description</b>	<b>3</b>
Customer UI	3
Restaurant UI	3
Server/DB requirements	4
Additional/Nice to have	4
<b>Use Cases</b>	<b>5</b>
Ordering a Burger	5
Receiving an order	5
<b>Use Case Diagram</b>	<b>6</b>
<b>Class Diagrams</b>	<b>7</b>
<b>Class Responsibilities</b>	<b>8</b>
<b>UI MockUp (Customer UI)</b>	<b>10</b>
<b>UI Mockup (Operator UI)</b>	<b>13</b>
<b>Server Architecture</b>	<b>15</b>

## User / System Requirements

### Customer UI

*Cust UR1:* To add their name

*Cust UR2:* To create custom burger combinations from stocked products via an intuitive GUI

*Cust UR3:* To select quantity of ingredients

*Cust UR4:* To see the price as the order is updated

*Cust UR5:* To submit order

*Cust UR6:* To select option for payment (Pay now, or in-store)

*Cust SR1:* Order must be sent to the restaurant via the server

*Cust SR2:* User notified if the chosen ingredients are no longer available

*Cust SR3:* Be notified of their order number once order submitted

*Cust SR4:* Cost of each ingredient must be displayed to the user prior to selection

*Cust SR5:* All orders must have exactly one bun selection

### Restaurant UI

*Rest UR1:* Ability to update ingredients (quantity / price)

*Rest UR2:* Ability to change order status

*Rest UR3:* Must be able to alter the list of ingredients types to add/remove ingredient types

*Rest SR1:* Retrieves current orders from the database for production

*Rest SR2:* Updates the current order list once a burger has been requested

*Rest SR3:* The restaurant app will accept only valid orders

*Rest SR4:* Orders will be displayed to all workers

*Rest SR5:* Order must have a status (pending, in progress, complete, collected)

*Rest SR6:* Ingredients must be automatically updated as orders come in

*Rest SR7:* Ingredients have minimum holding levels that when reached will notify manager for reorder

*Rest SR8:* Order removed from display once collected

### **Server/DB requirements**

*SR1:* DB will hold accurate quantities and costs for all ingredients

*SR2:* Must update ingredient levels at customer submission

*SR3:* All communication between customer and restaurant must be processed through the server

*SR4:* Must store completed orders to permanent storage for future reference

*SR4:* Orders stored in the database will use the timestamp combined with the 'orderID' as a unique ID

*SR5:* Order ID numbers will restart from 0 at the commencement of each day

*SR6:* Robust against SQL injection attacks

### **Additional/Nice to have**

- Ability to order multiple different burgers per order
- User privilege levels based on job title e.g. manager has authentication to adjust stock levels
- Ingredients on customer UI will be updated every 1 minute
- Suggested burger combinations
- Opening hours displayed on website
- Menu
- Contact us page
- Language compatibility
- Delivery options
- User can specify pick up time
- Time bound progress for burger cooking, if burger in-progress for too long then operator is notified
- Order page resize nicely (e.g. for mobile)
- Web display status of order

## Use Cases

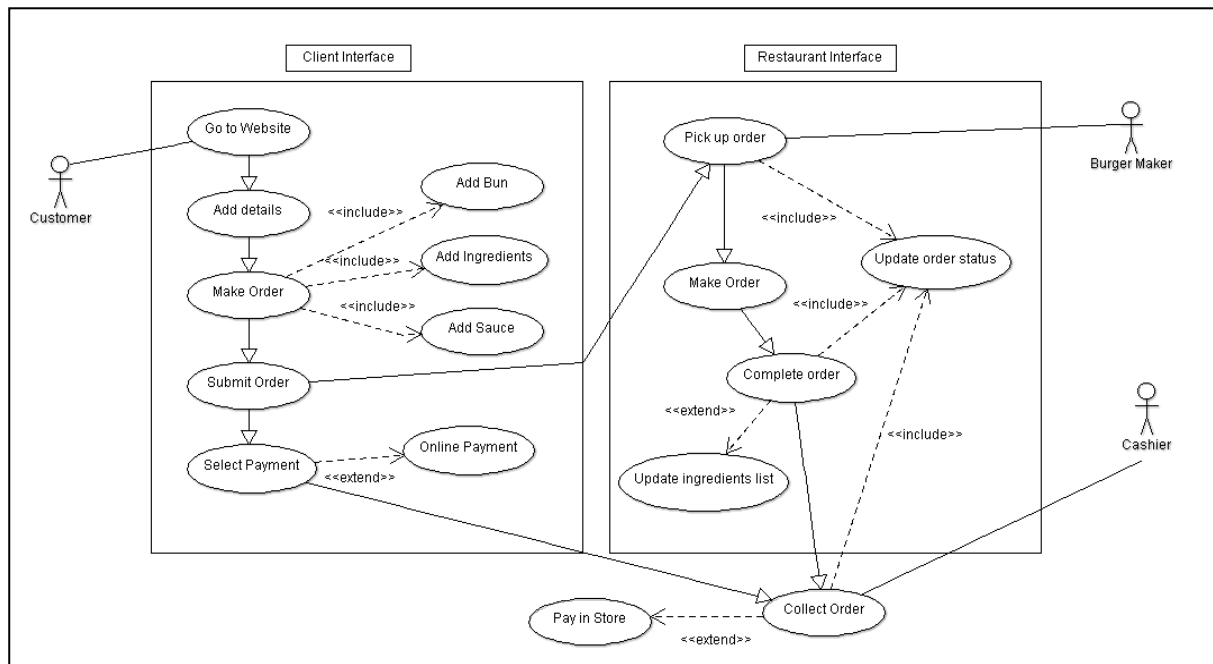
### Ordering a Burger:

- User goes to website
- User adds details/name
- User Makes Order
  - Select type of bun
  - Select ingredients
  - Select sauce
- User submits order
- User selects payment method
  - Pay online (if selected)
- User receives order number
- User collects order from store
  - Pay in store (if selected)

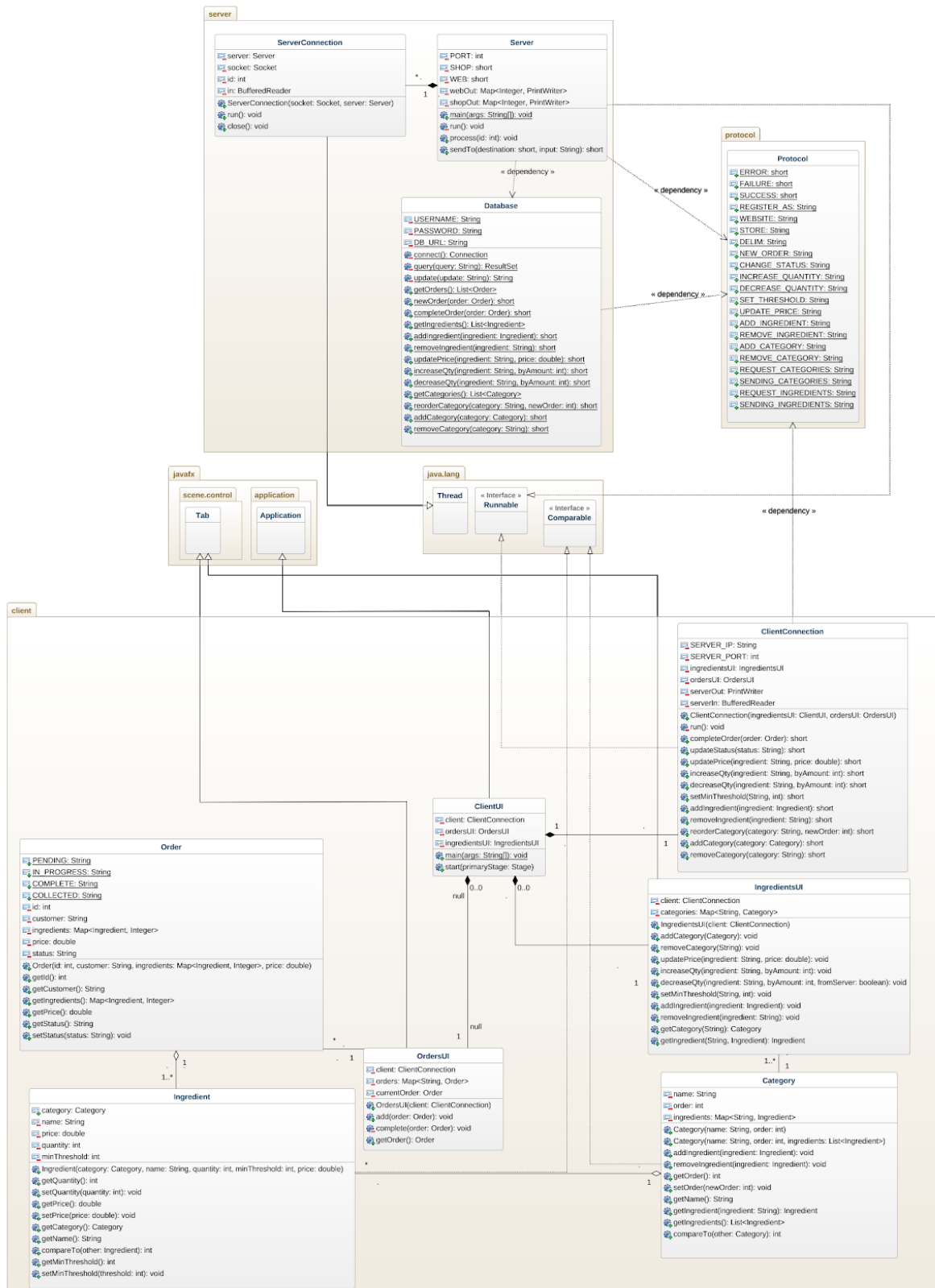
### Receiving an order:

- System picks up order
  - Worker changes order status from “Pending” to “In Progress”
- Worker makes burger
- Worker completes order
  - Worker changes order status to “Complete”
  - Worker makes updates to ingredients stock list (if required)
- Order collected by user
  - Take in store payment (if selected by user)
  - Worker changes order status to “Collected”

## Use Case Diagram



# Class Diagrams



## Class Responsibilities

Server:

- Main server-side entry point
- Collects incoming connections from web server and shop clients
- Processes incoming data
- Forwards information to web server or shop clients as appropriate
- Forwards information to Database class as appropriate

ServerConnection:

- Forwards incoming data to Server object

Database:

- Processes queries, updates, and table alterations via JDBC

Protocol:

- Holds public static final fields for consistent communication between server and clients

ClientConnection:

- Processes incoming data and calls appropriate methods on IngredientsUI, OrdersUI, Ingredient, Category, and Order objects
- Receives information from IngredientsUI, OrdersUI, Ingredient, Category, and Order objects and forwards that information through the server to the database and web server

ClientUI:

- Main client-side entry point and application window
- Overall GUI layout including TabPane
- Instantiates ClientConnection object, IngredientsUI object, and OrdersUI object



IngredientsUI:

- GUI layout for ingredients tab
- Holds and displays a Category object for each category of ingredients
- Displays ingredients as per GUI requirements
- Sends information about changes to categories and ingredients to ClientConnection object
- Receives instructions from server via ClientConnection object and forwards them to the appropriate Category and Ingredient objects

OrdersUI:

- GUI layout for Orders tab
- Holds and displays Order objects for all orders with status other than COLLECTED
- Sends completed orders to server for storage in database
- Receives incoming orders from server via ClientConnection object

Category:

- Holds an Ingredient object for every ingredient belonging to that category
- Provides easy way to order and separate ingredients via compareTo method
- Provides ingredients to IngredientsUI object

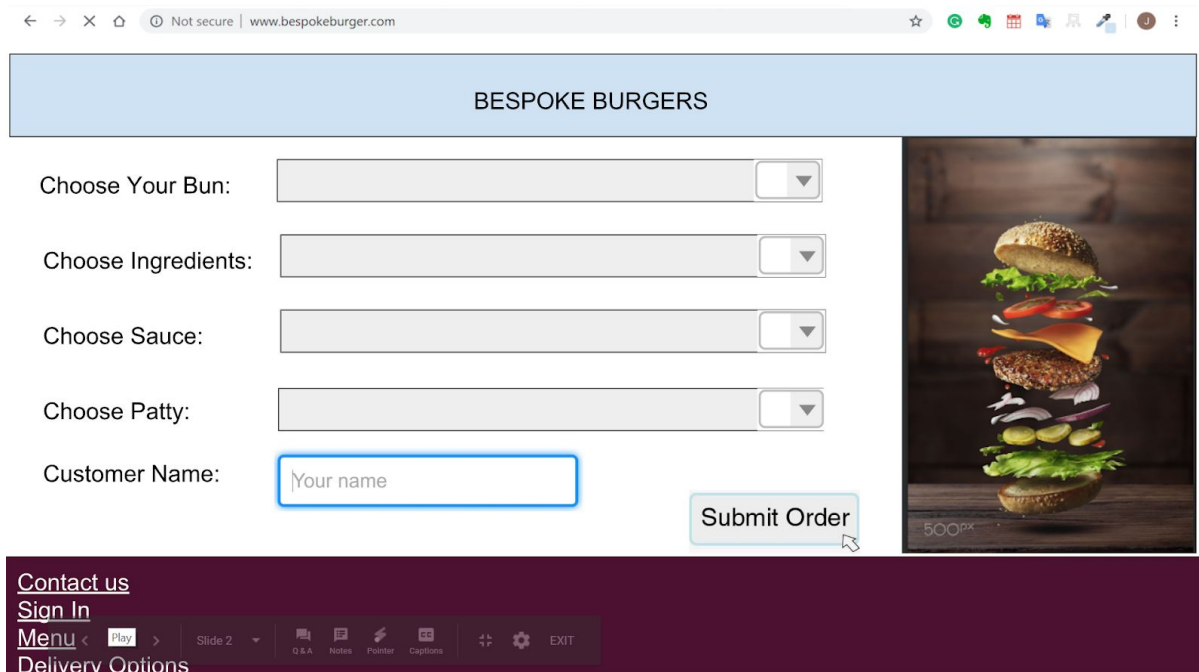
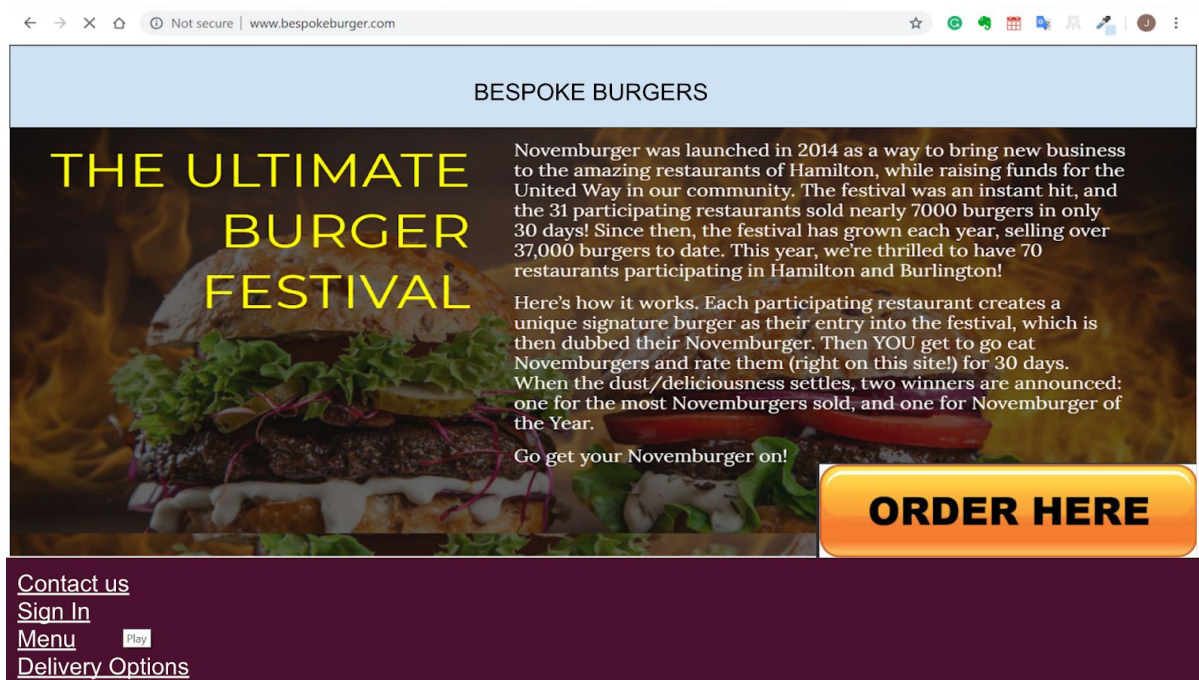
Ingredient:

- Holds specific information about ingredient state (e.g. quantity and price) for display in IngredientsUI object

Order:

- Holds specific information about order state (e.g. ingredients, price, and status) for display in OrdersUI object

## UI MockUp (Customer UI)



← → × ⌂ ⓘ Not secure | www.bespokeburger.com ☆ 📧 📅 📁 📎 📌 📌 📌 ⋮

## BESPOKE BURGERS

Choose Your Bun:


Sesame	-	1	+
Plain	-	1	+
Wholemeal	-	1	+

Choose Ingredients:

Choose Sauce:

Customer Name:

[Submit Order](#)



500px

[Contact us](#)

[Sign In](#)

[Menu](#) Play

[Delivery Options](#)

← → × ⌂ ⓘ Not secure | www.bespokeburger.com ☆ 📧 📅 📁 📎 📌 📌 📌 ⋮

## BESPOKE BURGERS


Choose Your Bun:

Choose Ingredients:

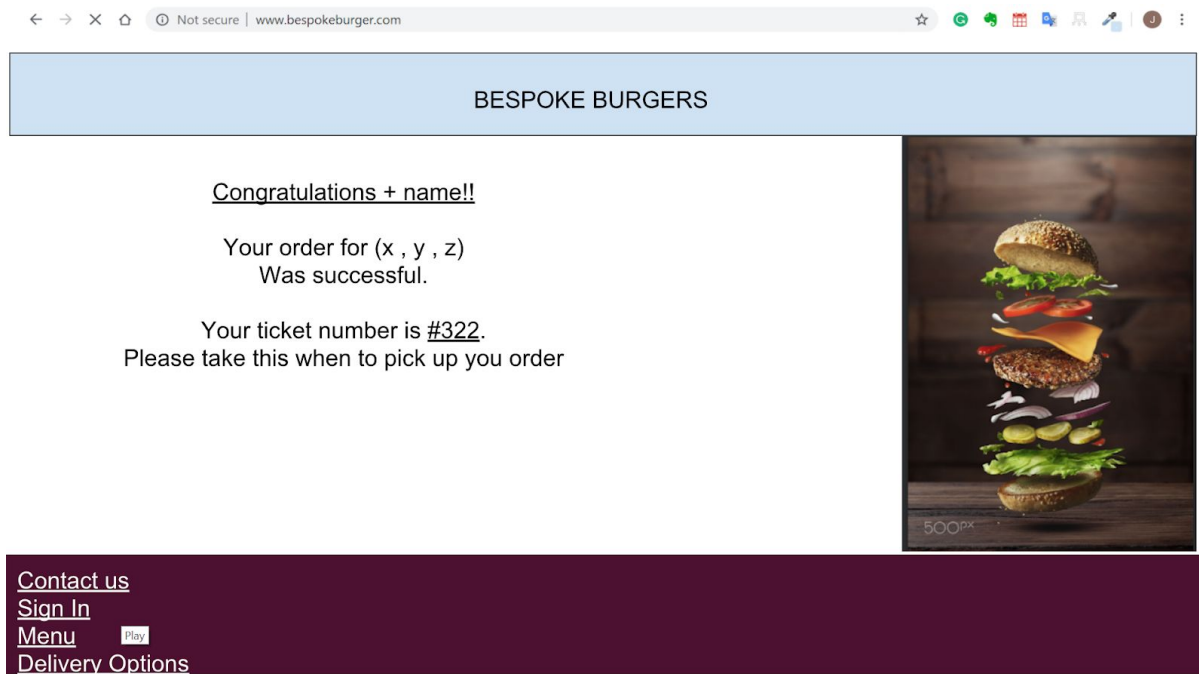
Lettuce	-	1	+
Cheese	-	1	+
Tomato	-	1	+
Beetroot	-	1	+
Gherkin	-	1	+
Pineapple	-	1	+

Choose Sauce:

Choose Play Patty:



500px



## UI Mockup (Operator UI)

### Operator UI - Orders.


**Orders** **Ingredients** **Cook** ▼

Order #	Status	Name	Bread	Patty	Ingredients	Sauces	Action
001	In-progress	Johnny	Sesame	Falafel	Lettuce x 1 Tomato x 2 Onions x 1	BBQ - Tomato	DONE
003	PENDING	Sally	Wholemeal	Chicken	Lettuce x 3 Onions x 1 Jalapeno x 1 Coleslaw x 1	Ranch - Tomato	
004	PENDING	Donald	Plain	Beef x 2	Cheese x 1 Olives x 1	BBQ	

Oldest ————— Newest


- Operators able to select which orders to see based on their role. For example, those making the burgers would see all orders PENDING and orders that **they** have chosen to make (IN\_PROGRESS).
- Tap an order to choose a burger to make, changing its status from PENDING to IN\_PROGRESS. This will update the database and remove the order from other operators' screens.
- Tap again to hand order back to pool of orders in case of user error/circumstances preventing the operator from completing the order.
- Tap DONE button to set status to READY when they've finished making the order.

**Operator UI - Ingredients.**

Orders		Ingredients	
Ingredient	Current Stock	Update Stock	
Beef	5	<input type="button" value="+"/> <input type="button" value="-"/> <input type="text" value="20"/> <input type="button" value="OK"/>	<div>ORDER </div>
Chicken	305	<input type="button" value="+"/> <input type="button" value="-"/> <input type="text" value="0"/> <input type="button" value="OK"/>	
Falafel	300	<input type="button" value="+"/> <input type="button" value="-"/> <input type="text" value="0"/> <input type="button" value="OK"/>	
Lettuce	25	<input type="button" value="+"/> <input type="button" value="-"/> <input type="text" value="0"/> <input type="button" value="OK"/>	
Tomato	40	<input type="button" value="+"/> <input type="button" value="-"/> <input type="text" value="0"/> <input type="button" value="OK"/>	
Onion	45	<input type="button" value="+"/> <input type="button" value="-"/> <input type="text" value="0"/> <input type="button" value="OK"/>	
Jalapeno	30	<input type="button" value="+"/> <input type="button" value="-"/> <input type="text" value="0"/> <input type="button" value="OK"/>	
Coleslaw	95	<input type="button" value="+"/> <input type="button" value="-"/> <input type="text" value="0"/> <input type="button" value="OK"/>	
Beetroot	80	<input type="button" value="+"/> <input type="button" value="-"/> <input type="text" value="0"/> <input type="button" value="OK"/>	
Olives	80	<input type="button" value="+"/> <input type="button" value="-"/> <input type="text" value="0"/> <input type="button" value="OK"/>	
Cheese	35	<input type="button" value="+"/> <input type="button" value="-"/> <input type="text" value="0"/> <input type="button" value="OK"/>	
...	...	...	

+

ORDER



- List all ingredients with their current stock levels.
- Add a new ingredient or add/remove ingredient classes with the plus button.
- Arrange by ingredient class, however highlight items that are below a certain threshold by colour and position (in this example, beef is low so it is red and at the top).
- Access the settings of each ingredient (set threshold, change name, remove) via the info button.
- Order more of a particular ingredient from the supplier.
- Add/remove quantities of stock, confirming the change with the tick button.
- The order and settings buttons will appear next to the currently selected ingredient.

**Operator UI - Ingredient Modals**

X

**Order**

Ingredient: Beef

Order Quantity:

Wholesale Price/unit: \$1  
Total cost: \$100

X

**Settings**

Ingredient:

Class:

Threshold:

X

**New Ingredient**

Ingredient:

Class:

Threshold:

Price/unit:

X

**Edit Classes**

Current Classes:

Patty

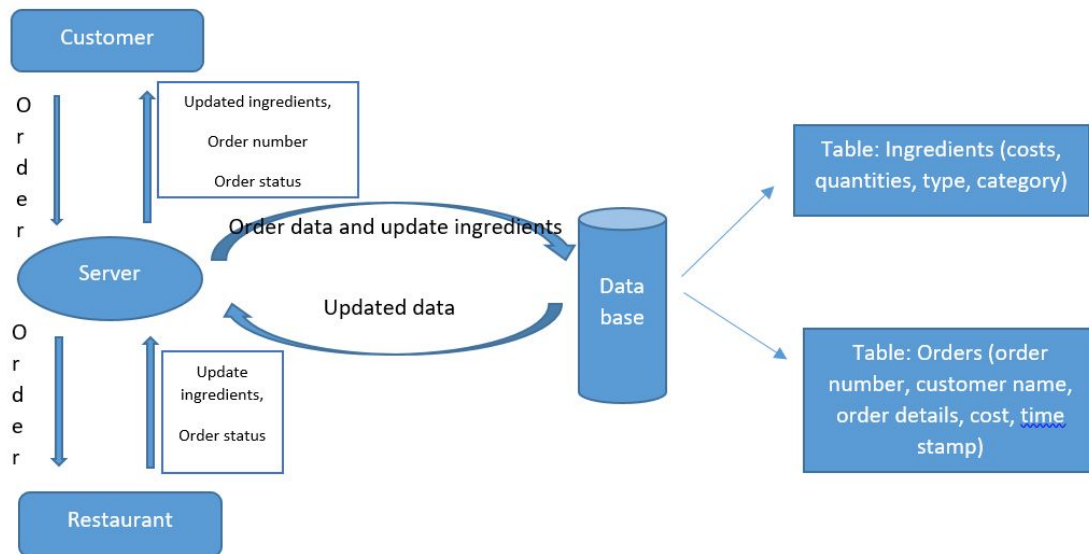
Bun

Salad

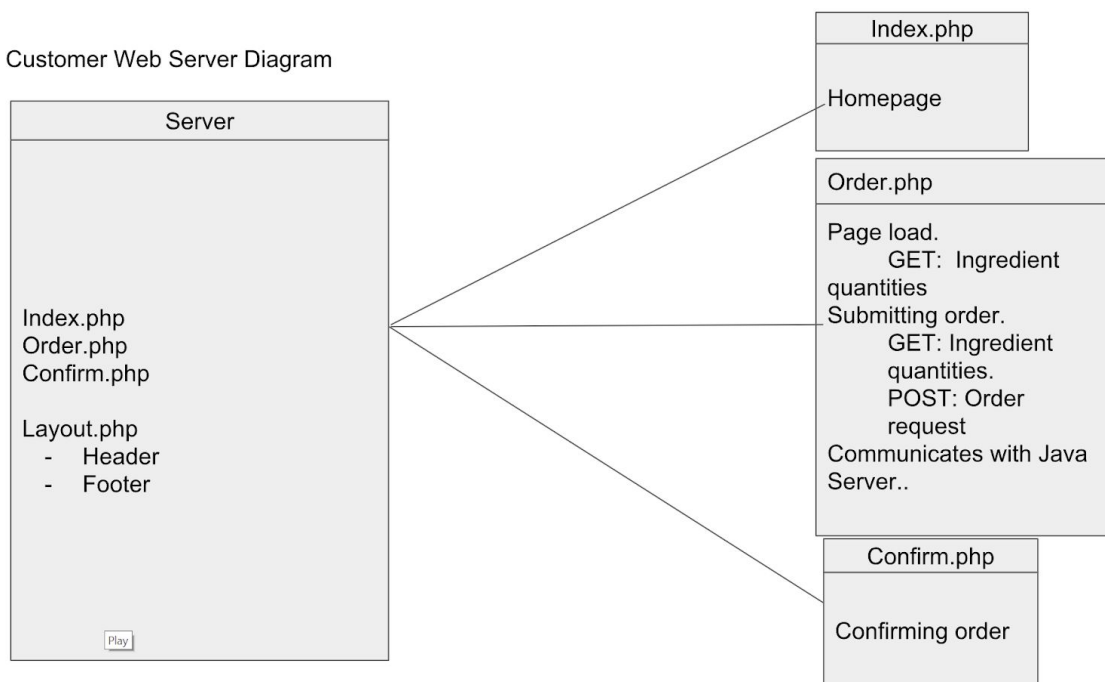
Sauce

Add New:

## Server Architecture



Customer Web Server Diagram



- When the web client requests a page from the web server, the web server will query the database (via the java server) for what ingredients exist and what quantity is in stock. The server will populate the webpage with the appropriate information and send it to the client. Only ingredients with quantity  $> 0$  will be displayed.
- The customer will compile their order using the webApp UI. Once submit button is clicked, the order will be sent to the store via the java server.



- The web server will check if this order is valid or not. If valid, the web server will send the order to the restaurant and database, then the restaurant will process this order, and the database will be updated accordingly. If the order was not valid, the user will be notified and their page refreshed with the updated ingredients.

*Information sent from server to client:*

- Updated ingredients (when requested)
- Order number (after submission)
- Order status (after submission)

*Information sent from client to server:*

- Customer name
- Order details (bun, ingredients, sauce, patty)

*Information sent from server to database and restaurant:*

- Order number
- Customer name
- Order details(bun, ingredients, sauce, patty)
- Time stamp
- Update ingredients

*Information sent from restaurant to server:*

- Order status
- Update ingredients

*Information shared inside the restaurant:*

- Order number
- Customer name
- Order details (bun, ingredients, sauce, patty)
- Time stamp
- Order status
- Ingredients which reach minimum holding levels



