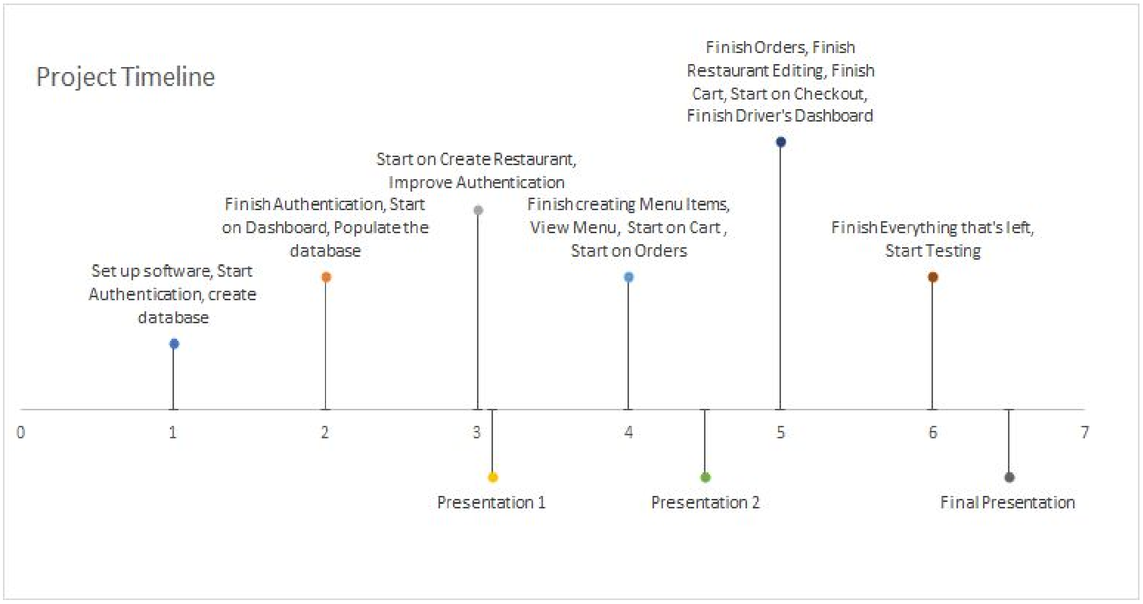
FOODMachine

Presented by The Powerpuff Spartans

Authors: Rohan Agrawal, Alexis Butler, De’terris Fox, Harsh Nagargar, and Adam Whitten

Process Documentation:

1. Vision Statement:
    a. Our vision is to provide customers the ease of getting their favorite restaurants to deliver to them without the hassle of leaving home. We believe in convenience, so the development of FOOD Machine helps with that. Our goal is to create a fully functional websites designed around food delivery. This website would be open to customers, owners of restaurants and other food stores, and food deliverers.
2. Iteration Timeline:
    a. Our timeline over the past few months was broken down into several sprints. Each sprint lasted two weeks, so when we met with our client, we always showed her newly added features of our project. Our tasks were created on cards through Trello and then distributed to each partner to complete by the end of the sprint. The timeline below shows our projection over the course of six to seven sprints.

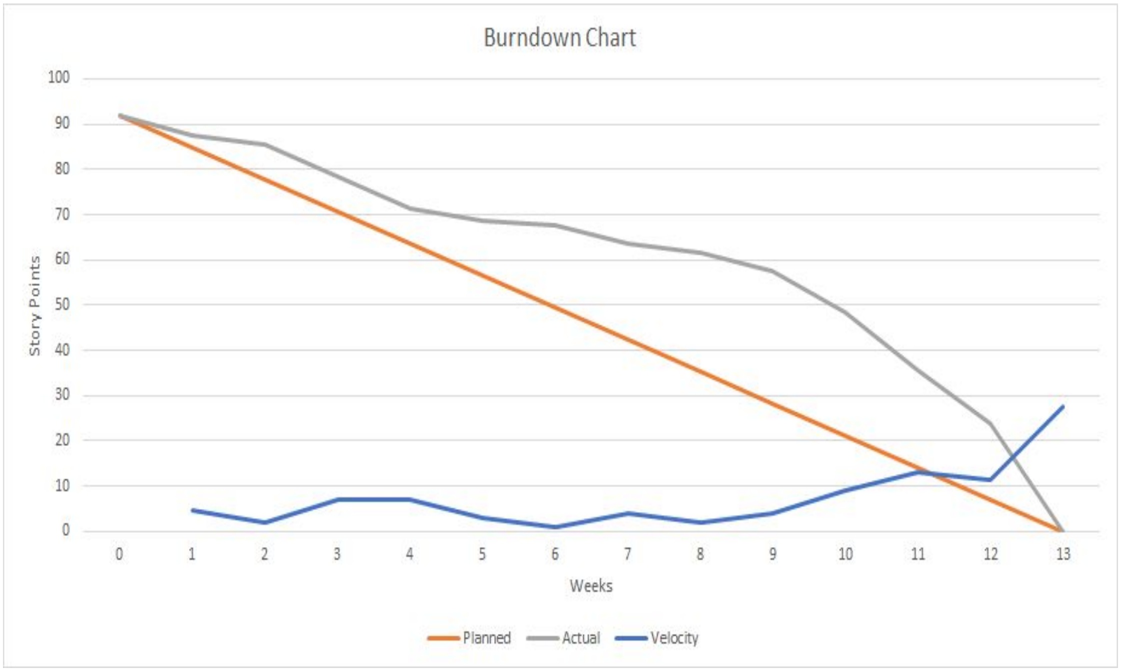


3. Software Project Management Plan:
    a. Using Agile Methodology, our highest priority was keeping the client updated on the progress of our project. After every meeting, our goal was always to deliver new implementations to keep the client interested in our delivery applications, reflecting on our success and failures based on feedback from our client. It ensures that we fix and correct everything needed for our next meeting. By meeting face-to-face to discuss anything involving our applications, we were able to work more efficiently and help one another when stuck on a given task.

Another method used was extreme programming. Paired programming was very helpful. Rather than going solo, we would split into pairs based on experience. Pairs varied by difficulty of task. Also, we always had visuals on what everyone was doing through git. Members posted updates or fixes almost daily to track progress or backtrack. Frequent commits helped us to stay on track with our sprints.

As for SCRUM practices, we tracked every task on Trello and prioritized based on a difficulty from 1-5 story points with 5 being the most complex. We also met for an hour twice a week to evaluate and update one another on anything important. Our story points correlated with our meetings with the client, so we only had two weeks to complete each set of story points.

Burn down charts also played a huge role to track our status on reaching the end. We would sometimes find ourselves falling behind on certain story points, but quickly caught back on track just in time for the final presentation.



Team Roles: FOODMachine was created by a five person team. Even though each team member was given a specific role (Project Manager, Front End, Back End, Server side), each team member worked in multiple areas of the software. In other words, we were not tied to our role. We created our components from the bottom up starting with the backend database and working towards the frontend.

4. Software Requirements Specification:

a. Requirements Engineering
    i. Functional Requirements: All users shall be able to search for any of the restaurants and menu items via filters that are available on our site. The site shall list the menu items for each restaurant. Only the customer shall be able to add menu items to his or her cart and checkout. Only the restaurant owner shall be able to create their restaurant and add items to their menu.

b. How the requirements were collected?
    i. The requirements for this project were collected by looking at the project requirements, otherwise known as “MVPs”, and discussing as a group on how we will be able to successfully implement these features.

<u>Product Documentation</u>:

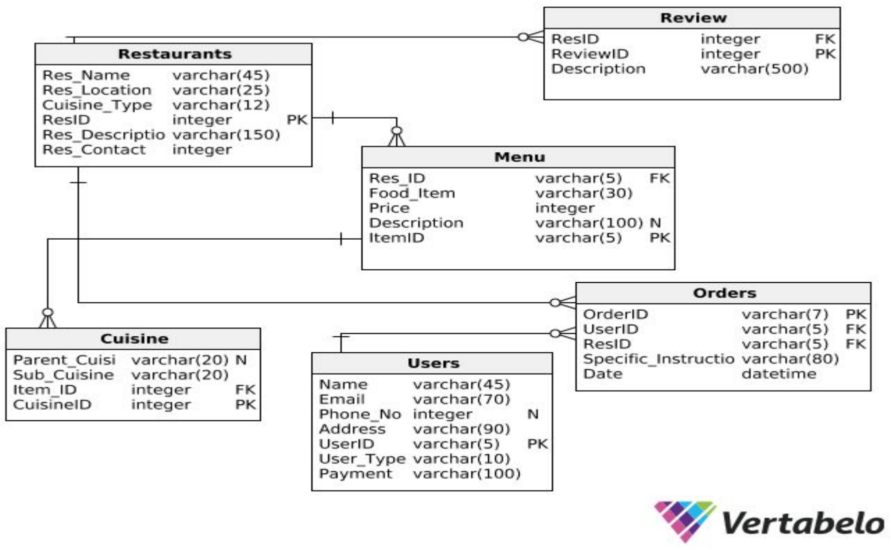
1. Architecture and Software Design Principles
   Architecture of the software is MVC, which comprises of two main processing layers, client side processing and server side processing. On client side, all data is processed using javascript. This processing mainly deals with cart engine management. On server side processing, we write functions to check and interact with the database for limited access. For further detail on the two:
   - Server side
     Here we used authentication method using prebuilt in libraries for better security. This allowed us to create a system wide single use token for each session. We used django forms whenever data is posted to server to ensure there are no XSS attacks and database leaks for unknown queries running.
     The only times where we didn’t run django forms were when with ajax calls done using “post” method to get the cart where we manually created checks to ensure robust performance. All the queries for our database were ran using django models. The orders table was the only one to use Json String data type for compressed processing. One of the major features that our web app has is the ability to send emails to users when needed. For that, we use sendgrid api and registered a domain name foodmachine.ml (.ml stands for mailing list). This allows us to keep the client updated for the action he was taking during the whole journey of ordering.
   - Client Side
     For Client side processing, we use Local storage and Web storage APIs. With these APIs, we were to able to avoid constant ajax calling back and forth between server and client. In Local storage, we store Json objects where we had price quantity and Menu item name for storage. In this, we update, delete, and add functionality were easy to implement. All price tax and total calculations are done on client side where that information is shown and checked back again in

the server before placing order. Until the order is placed, we use server session to store the ajax call done by the client.

Software Design Principles: We designed software in groups using the actions methodology. We would sit down as a group and discuss all the action that needed to be done before a particular task needed to be accomplish. We would then break down these action in individual story points and assign the full stack or role based on expertise to that person with a tag teammate as a helper. We did not use a UML diagram mainly because we were not able to handle the complexity and details of an action at the micro level. We need a micro analysis way where at least two members should be able to understand the complexity of the program. Since all group members have various but limited backgrounds, this complexity was hard to achieve. Using actions methodology, we got the job done and kept at least one person other than the leader for the action to know the implementation for the technology.

Database Schemas:

1. First Iteration:



Review

ResID	integer	FK
ReviewID	integer	PK
Description	varchar(500)	

Restaurants

Res_Name	varchar(45)	
Res_Location	varchar(25)	
Cuisine_Type	varchar(12)	
ResID	integer	PK
Res_Descriptio	varchar(150)	
Res_Contact	integer	

Menu

Res_ID	varchar(5)	FK
Food_Item	varchar(30)	
Price	integer	
Description	varchar(100)	N
ItemID	varchar(5)	PK

Orders

OrderID	varchar(7)	PK
UserID	varchar(5)	FK
ResID	varchar(5)	FK
Specific_Instructio	varchar(80)	
Date	datetime	

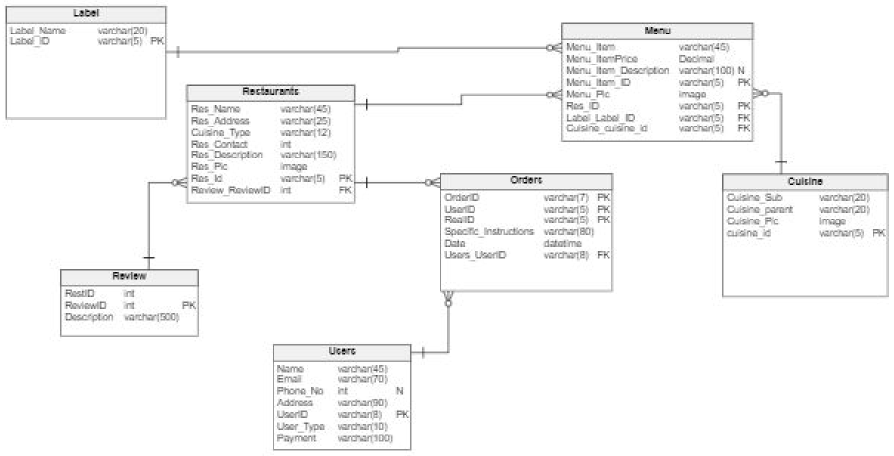
Cuisine

Parent_Cuisi	varchar(20)	N
Sub_Cuisine	varchar(20)	
Item_ID	integer	FK
CuisineID	integer	PK

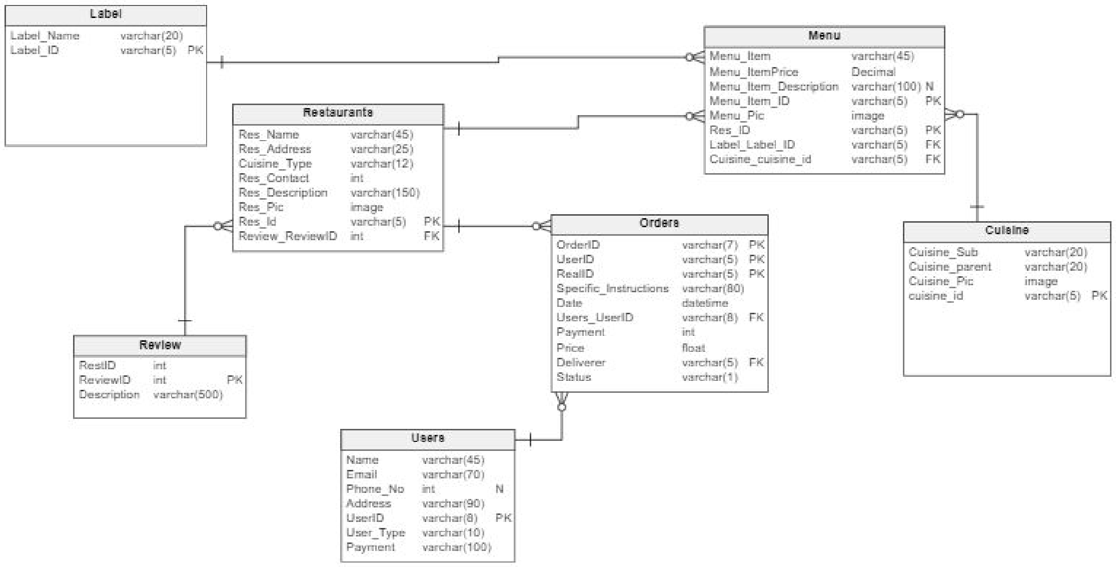
Users

Name	varchar(45)	
Email	varchar(70)	
Phone_No	integer	N
Address	varchar(90)	
UserID	varchar(5)	PK
User_Type	varchar(10)	
Payment	varchar(100)	

2. Second Iteration: In the second iteration, we added the Label table and Image Fields



3. Third Iteration: On the final iteration, we added Status, Price, Payment, and Deliverer attributes to the Orders table.



2. Software Testing/Plan:

   Our group focused on validation techniques to make sure we were building the right product. We inspected each feature functionality as we were building the software. Each method was tested individually during the process with dynamic data to check its functionality.
   No automated testing methods were used during the process due to the time frame of the project. Instead, we relied on dynamic testing methods to feed different data into the software and test its functionality.

Alpha testing methods were utilized during the process where our group would meet with Madeline every alternate week to report our progress and get feedback on our current work. This customer testing method proved to be very effective in terms of validation and verification.

3. User Guide:

This software works differently for each user type: Restaurant Owner, Customer, and Deliverer. First: Go to the Sign up page to create an account. In this process, you will choose a user type .

If you are a Restaurant Owner:

1. Click on your profile name on the navigation bar to go to your user dashboard.
2. On the user dashboard, there will be a button that says “Create Restaurant”. This will redirect you the Create Restaurant page.
3. Fill out the Create Restaurant form and hit submit. You now have a Restaurant. You can only own one restaurant per owner account.
4. After you have created your restaurant, go your menu. It will start out empty with a “Create Menu Item” button, which will open up a separate window with the create menu item form.
5. Fill out the form and submit..
    a. “Label” represents what kind of item you are creating, such as an appetizer or beverage. You are given a list of pre existing labels. Sometimes, however, items are called by different labels depending on region such as “drink” and “beverage”. If the label that you want does not exist already, type in the label that you want into the field. Our system will create it in our database.
    b. “Cuisine” represents food item’s style based on cultural regions/nations. We have a list of 25 different cuisines to choose from. This field is not like the “Labels” field.
6. Once you hit the submit button to create an item, your menu page will automatically reload with the new menu item. You are able to edit or delete the item. The only thing to wait for now is for a customer to make an order.
7. If a customer makes an order from your restaurant, go to your orders tab from your user’s dashboard. You will see the customer’s order listed with buttons indicating whether or not to accept it. If you accept it, the order will be added to a queue for the deliverer.

If you are a Customer:

1. There are two tabs on the navigation bar: Restaurants and Food Listings. Each will take you to a page that will automatically list out all restaurants and menu items, respectively.
    a. In Restaurants, you are given two filters: Filter by Cuisine and Sort All Alphabetically. “Filter by Cuisine” allows the user to filter out Restaurants based on Cuisines. “Sort All Alphabetically” sorts all Restaurants in alphabetical order (A - Z).

   b. If you want to view the menu on a specific restaurant, click the “View Menu” button.
   c. In Food Listings, you are given three filters: Filter by Label, Filter by Cuisine, and a Search bar. “Filter by Label” and “Filter by Cuisine” function the same way as Restaurants “Filter by Cuisine”. The search bar allows you to find any item that contains the words or letters that you have searched; it is case insensitive.
2. Under each menu item, there is an “Add to Cart” button. Clicking this will add the item to the cart.
3. When you are ready to checkout, click on the cart icon in the upper right. If you do not have your payment information already in place on your profile, you will be asked to update your information. If your payment information is already there, go to your orders tab on your dashboard and checkout. Your order should be submitted for review for the Restaurant Owner.

If you are a Deliverer:

1. Go to your dashboard and navigate to your Orders tab. There, you can accept to deliver any orders that the Restaurant Owner has accepted from the Customer.

4. Limitations:

   Despite being able to successfully finish the MVPs, our group felt that more functionalities could have been added to the existing MVPs, including:

   - The user cannot signup/login through Google or any other social media platforms and would have to create an account on our website.
   - The restaurant cannot contain any special characters in its name
   - The checkout cart does not allow customers to put additional information to the order
   - The customer does not know which deliverer has picked up their order