TECHNICAL DESIGN DOCUMENT

Project Title: SmartCart

Version 1.0

Created: 7/31/18

Last Updated: 8/3/18

Colantonio, Brian

1.1 Project

I am excited to design a web service that's service centered and uninterested in consumer data. That being stated, users will not need to create an account in order to utilize the basic functionality of the application.

I've developed a great appreciation for the minimalistic approach when it comes to user interface design and user experience. The SmartCart homepage will reflect that sentiment. Products will be searchable upon first load as a new user. The only benefits that version 1.0 will have to those who wish to create an account will be the ability to review their product search history, along with the verification of it's adherence with the user's dietary preferences. (version 1.0 will only verify whether a product is considered vegetarian or not)

1.2 Technical

Smart Cart version 1.0 will be a browser-based web application with a user interface which will be adaptive to the end device. PHP will be used for most of the project. Front-end development will be completed using HTML containing jquery accents for an enjoyable user interface/experience.

SmartCart will grant the user the ability to receive verification on whether consumer products that they search for are considered vegetarian or not. Ideally, this will be completed by making requests to a third-party API for product information, then filtering key ingredient data.

MySql will be used to maintain user data.

2.1 Software

Due to my previous development experience, Notepad++ will be utilized to write the majority of the code. Decision may be subject to change.

XAMPP will be utilized for database administration.

2.2 Programming

Pseudocode:
A connection class will be needed to save user searches. (connection.php)

Public getConnection()

A class is needed that will call the connection class. This class will need to include: a function that gets all user searches, a function that inserts one of the users searches, and maybe a product delete function.

Public function getProducts(userID){select * from products where userid = userID} | insert function would be similar but you would need the product ID as well.

Real time requests to a 3rd party API is the most desirable outcome for the finished product, if API requests compromise the integrity of the user experience, an option for beta will beto download a segment of the API and store it in my database for easier accessibility.

Database will consist of User_Products relational table. (select from * User_products where userID = userID | insert into User_Products where productid = productid, UserID = UserID,

//check if product exists | select * from products where id = productid | Delete would use the same parameters but utilize a generic delete syntax)

User.php | getUser(email, password) {SELECT * from users WHERE email = email & password = password}

InsertUsers(email & password){}

deleteUser(){}

PRODUCTS table | SELECT * products | getProduct(productID){SELECT * from products where ID = productID}

APIcall.php | getAPI{//in this function we are going to call api for product, //get product, insert into database}

Insert into products table.

Router.php | //we will receive calls from AJAX/JQUERY //assign a variable 'method' method = $_GET $_POST variable = $_GET[method] //create a switch, each case in this case switch on the method.

Html index and php index.

Jqery in html header, jquery CDN in header.

3.1 Anticipated date of completion

September 18, 2018.