The Burger Bar Web App

Project Description

Your collaborative project team is making a bid to create an online web app for The Burger Bar food truck. They are looking to streamline their ordering process and they think a new digital menu web application will make a big impact on sales. As a part of your bid, your team needs to submit a functional prototype to The Burger Bar to demonstrate your team's talent and hopefully win the bid. Your job is to create a 3-tier architecture composed of:

1. SQL database
2. PHP middleware
3. Web client

They understand that your bid is only a **functional prototype** so they are NOT expecting your application to look like a finalized product. However, they are expecting your team to meet all project requirements and, at a minimum, implement their rough sketch.

Project Requirements

From the website, users will be able to perform the following tasks:

Unregistered Customer User

1. Create a user account. *See UI Requirements section for more details.*

Registered Customer User

1. Ability to reorder the last order they placed (if one exists). This is useful for frequent customers that always order the same thing.

All Customer Users

1. Create an order of burgers
    a. An order consists of 1 or more burgers.
2. Create a burger
    a. Choose a type of burger
    b. Choose a bun
    c. Choose 0-1 type of cheese
    d. Choose 0-N toppings
    e. Choose 0-N sauces
    f. Choose 0-1 side item
    g. Select quantity
    h. Add to order
3. Have an order menu and can see their order while creating burgers
    a. Customers can easily and cleanly see what they've ordered and what options they've chosen for each burger
    b. Customers can add/remove and update the quantity of a burger
    c. Customers can see the price of a burger
    d. Customers can see the total price of their order
4. Submit order
5. Review and pay for order

DB Requirements

Create a database:

Create a database in MySQL that is sufficiently robust to store all data needed for the application. Minimally, it should handle customers (user accounts), orders, and saved orders. Where appropriate, add code to enforce primary keys. You may add foreign key constraints as well.

Populate your database:

Use the included menu.json file to extract data to populate your database for burger types, buns, cheeses, toppings, etc. You need to be able to parse the .json file to collect the proper information for populating your database.

SQL Queries:

You're responsible for creating the SQL queries which provide the UI developers with the burger types, buns, cheese, toppings, etc. When returning the result of your queries, you'll need to work with the UI developers for outputting the desired HTML using the PHP *echo* command.

UI Requirements

The following requirements will be used to inform the UI development and the database design. Your UI will need to handle everything here and the database will need to store it.

Login:

Since both guests and unregistered users are allowed to make orders, registered users should be able to login from any point during the ordering process.

1. Allows a registered user to enter an email address and password.
2. Before submitting information to the server, client-side validation will occur to ensure the following:
    a. Email address is not empty and is actually a valid email address
    b. Password field is not empty and is greater than or equal to 8 characters
3. If user authentication **fails**, the user should be given indication why authentication failed. If the user chooses to not re-authenticate, they should still be able to complete their order as a guest.
4. Upon **successful** user authentication, the user will be offered the ability to select their last order. This is useful for frequent customers that order the same items. Also, if a user is logged in, their payment information should auto-populate on the payment screen.

Account Creation:

Allows an unregistered user to create an account.

1. The registration form includes the following fields and validation:
    a. First name – cannot be empty
    b. Last name – cannot be empty
    c. Email – must match the format of a traditional email address; cannot be empty
    d. Password – must be greater than or equal to 8 characters; cannot be empty
    e. Credit card provider: Visa, MasterCard, or American Express
    f. CC number – must match this regular expression; cannot be empty
2. If any of the fields fail to validate, provide a meaningful error message to the user which describes what the problem is.

Order Screen:

Provides a customer the ability to create a burger and add it to their order, remove items from their order, see the price of each burger and the total order, and select/reorder the last burger order (for logged in users).

1. **Last order:** A menu that shows a registered user's last order (if one exists).
2. **Burger Builder:** Allows the user to select everything they want on their burger by specifying the meat, bun, cheese, toppings, sauces, and side item. Functionality includes:
    a. Allow the user to select a meat, bun, cheese, toppings, sauces, and side item.
    b. Clear all selected toppings
    c. Reset the entire burger back to default
    d. Select burger quantity
    e. Add the burger and quantity to the order
    f. All burger items (meat, bun, cheese, etc.) will be populated using PHP by reading these items from the database and then echoing out HTML.
3. **Order Menu:** Allows the user to view their current order, remove a burger from the order, edit a burger, and/or submit their order. Functionality includes:
    a. When a user adds a burger via the **Burger Builder**, the burger and quantity will be added.
    b. Allows the user to delete/remove a burger.
    c. Allows the user to change the quantity of a burger.
    d. Allows the user to submit their order where it will be processed by the server.
    e. Pressing the "Pay" or "Order" button shows the **Payment Screen**.

Payment Screen:

This screen should appear as a modal dialog box and overlay the entire screen. It will allow guest users to enter their payment information and will auto-populate with payment information for logged-in users.

Required Fields (See required validation in **Account Creation**):

1. First name
2. Last name
3. Credit card provider
4. Credit card number

Additional Requirements

- This project is designed to give you experience for your collaborative project, so you are required to use git, GitHub, and your production server to host this project. This means you need to create a new repository for this assignment.
- Document your PHP interface thoroughly. While you don't need to expose a public API, your private API should be sufficiently documented so that a new developer to your company could get up to speed very quickly. Documentation exists inside and outside of the actual PHP files. You could research a PHP API documentation engine that could extract comments in the code into a beautiful web-based API.
- Provide an ER diagram for the database schema. Include relationships as needed.

Deliverables

Submit all deliverables by Friday, October 17th @ 11:59 PM to the Burger Bar link on the CSE 3345 Blackboard course. Each team only needs one submission, so a single team member can upload for the entire team. You will need to provide the following:

1. A README file that includes:
    a. A link to your GitHub repo (we will be examining your code so make sure it is pretty and well documented).
    b. A link to your live website on the production server
2. ER Diagram
3. A SQL script file to create your database and tables. It should also include other constraints such as primary key and foreign key constraints.