Fast Food Online Order Platform

By: Youssef Rafik and Kyle Gomes

Mission Statement:

To provide an all-inclusive food ordering application that maximizes the speed, convenience, and efficiency of ordering as well as managing inventory and locations and boosting profit.

Mission Objectives:

1. CRUD operations for Customers
    - Create a New Profile
    - Read Customer profile information
    - Update Customer Information
    - Delete Customer Profiles
2. CRUD operations for Locations
    - Create new locations (when new locations open)
    - Read Location information
    - Update Location information (hours, specials)
    - Delete Location information (when they close for good)
3. CRUD operations for Staff
    - Create new staff records
    - Read staff information
    - Update Staff information
    - Delete Staff records and information
4. CRUD operations for Menus
    - Create new menus (deal menus, Lunch, Breakfast, Dinner)
    - Read Menu information
    - Update Menu information
    - Delete Menu information
5. CRUD operations for Orders
    - Create New Orders
    - Read Order details
    - Update Order details
    - Delete/cancel Order Details
6. CRUD operations for Delivery Method
    - Create New Delivery method (Drive Thru, curbside Pick-up, Delivery)
    - Read Delivery Status/Information
    - Update Delivery details
    - Delete Delivery information
7. CRUD operations for Transactions
    - Create new bill and transaction

   - Read logged transactions
   - Update transaction information
   - Delete/clear logged transactions
8. CRUD operations for Order Progress
   - Create order progress
   - Read Order progress
   - Update Order progress
   - Delete order progress records
9. CRUD operations for Inventory management
   - Create items to add for inventory
   - Read Inventory items
   - Update Inventory items
   - Delete Inventory records
10. CRUD operations for Coupons and Discounts
    - Create new coupons or discounts for promotions.
    - Read coupons/discounts details and usage.
    - Update coupons/discounts validity.
    - Delete invalid/expired coupons/discounts.

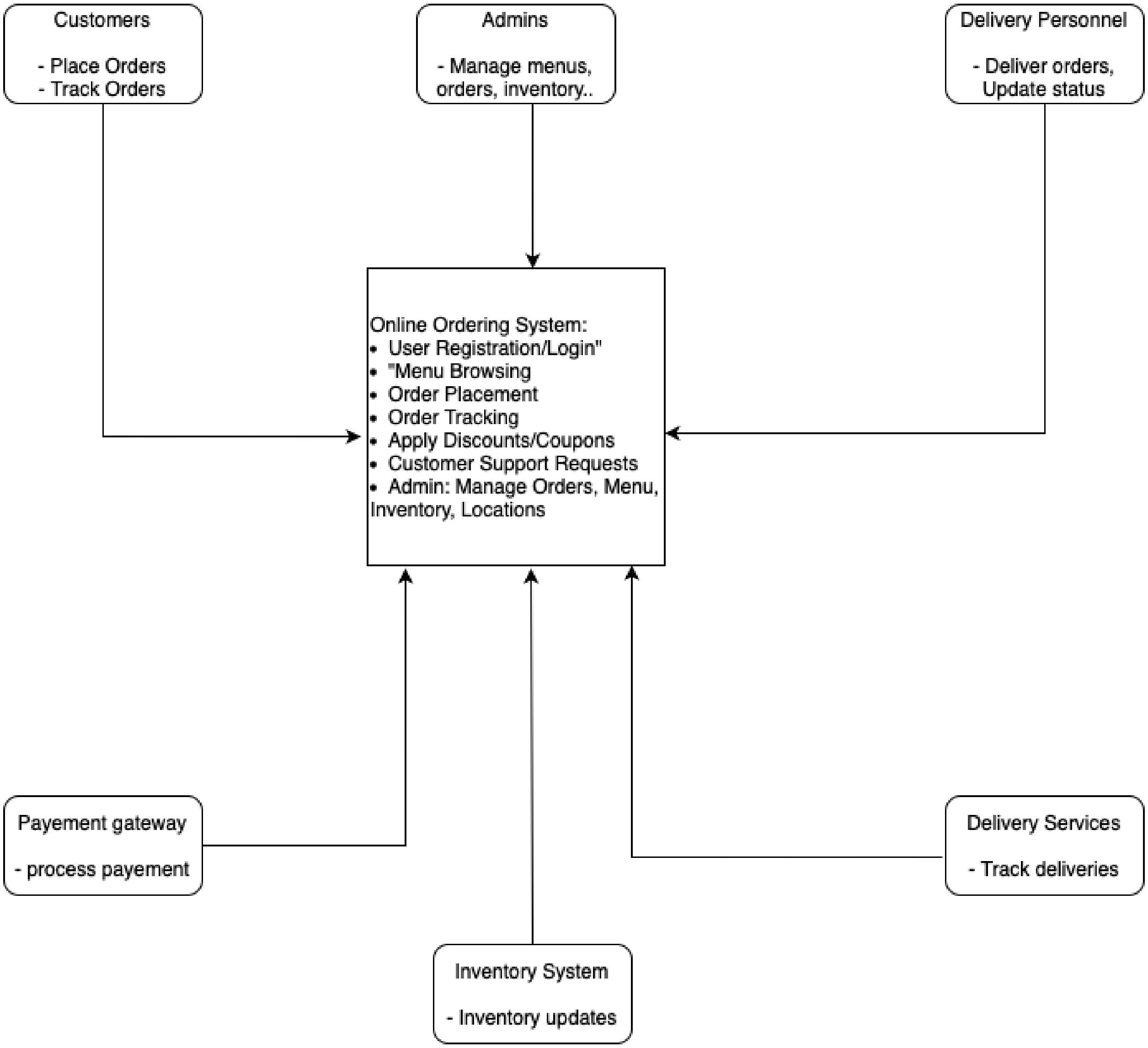
11. CRUD operations for customer feedback/reviews
    - Create new customer reviews for menu items or service.
    - Read customer feedback and ratings.
    - Update responses to customer reviews.
    - Delete inappropriate or outdated reviews.

12. CRUD operations for customer support request
    - Create new customer support requests.
    - Read details of support tickets.
    - Update the status of support requests (e.g. resolved, pending).
    - Delete resolved or invalid support tickets.

13. CRUD operations for refund requests
    - Create new refund requests for customers (e.g. for incorrect orders).
    - Read details of refund requests.
    - Update refund request status (e.g. approved, rejected).
    - Delete processed or invalid refund requests.

14. CRUD operations for subscription plan
    - Create new subscription plans for regular customers (e.g. meal delivery subscriptions).
    - Read details of active subscriptions.
    - Update subscription details (e.g. change plan, update billing).
    - Delete canceled or expired subscriptions.

System Boundary Diagram:



Samples of Responses from employed Fact Finding Techniques:

Interviews:

- Restaurant Owners/Managers: Learn about their needs for managing orders, staff, and inventory.

Manager Interview

Name: Jameson Schmidt ID: 10354326 Role: Employee Designation: Manager

Email: JShmidt0@company.com Location: 17 stopstreet rd Address: 133 fake street

Question 1: Why do you believe that an online food ordering system is beneficial for your company?

It helps us serve more customers and creates a convenient and easy way of ordering and getting food which in effect will boost our profits and our expected number of orders.

Question 2: What do you expect to be able to keep track of with a new online food ordering application?

We expect to be able to keep track of incoming orders, delivery progress, user information, payment information, location data, staff information, and Inventory data about stock of food.

Question 3: What are the biggest challenges you face with your current ordering system?

Our current fast food ordering system is great at displaying and collecting orders from customers but is not always accurate at telling the customer when certain items are out of stock at a certain location. The system should keep better track of stocked ingredients that we log and communicate that information to the customer.

- Customers: Gather feedback on what they expect from an online ordering system (e.g. ease of use, real-time order updates).

Customer Interview

Name: John Doe Email: JDoe@mail.com Phone #: (203)-434-4444

Customer ID: 70144239 Address: 123 Sesame street

Question 1: How often do you use mobile or online ordering methods when ordering fast food? Why?

I order fast food because I have a very busy work schedule and it is the quickest and most convenient option when I get out of the office late. On average I'd say that I order fast food three times a week for dinner but every morning for breakfast on my morning commute.

Question 2: What type of information do you expect to be able to see when ordering food with a mobile or online fast food ordering system?

I would expect to see an estimated time remaining for my order, a summary of my most recent order and updates to let me know if my order is completed, and a way to cancel my order if I changed my mind. If I am getting the food delivered to me at my office I expect to get updates about the progress of delivery as well and also updates from the person delivering it.

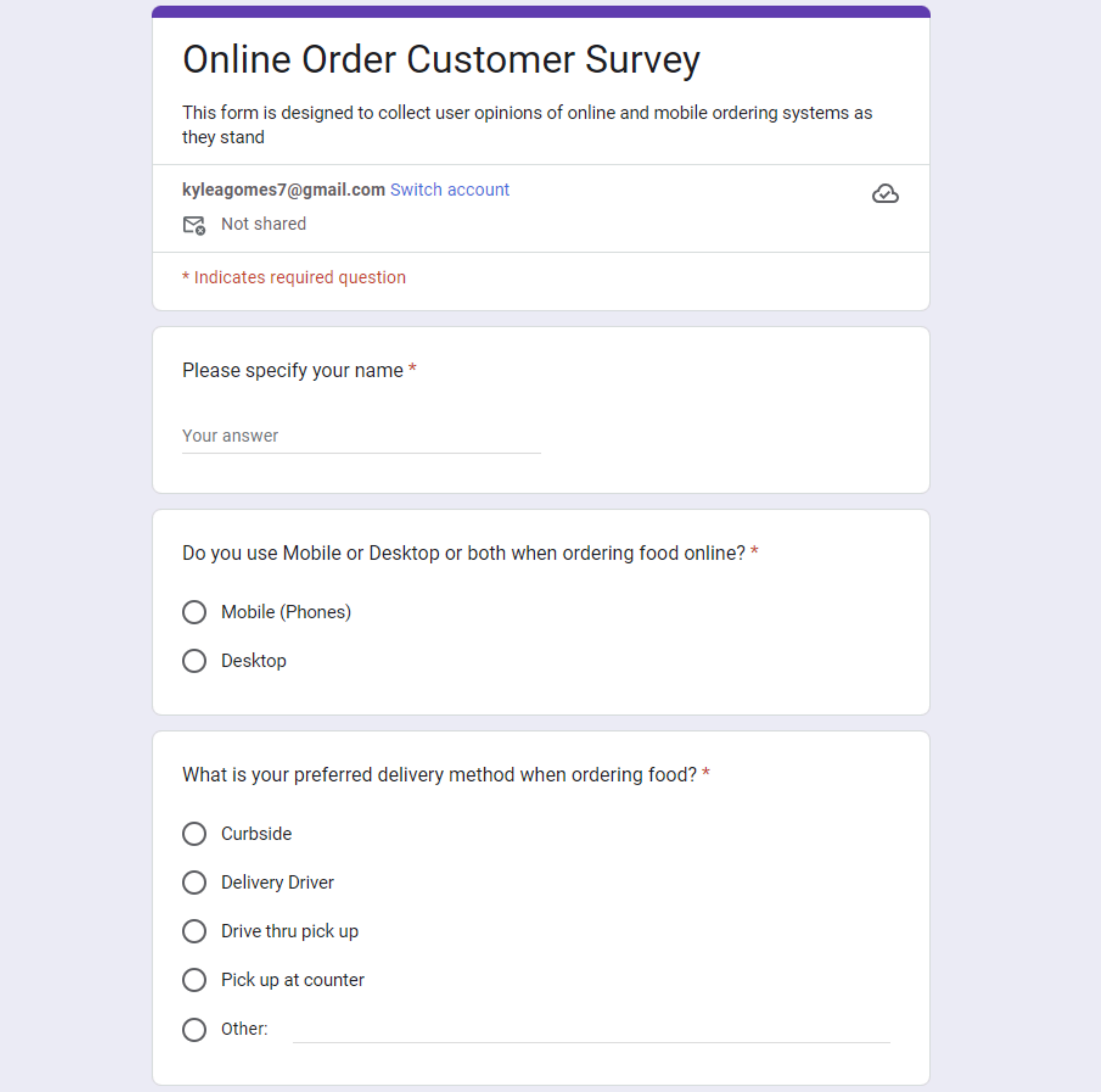
Question 3: What aspects of the current fast food ordering system do you find most helpful and which do you find most painful?

I like that all of the food items are grouped by category so that I can simply choose a descriptor and find the item that I want easily and quickly. I also really like that I can select a previous order and place it again by simply pressing a button, because it ensures that if I'm in a rush and I need to place a quick order then I can do so. One of the most painful aspects is that the time it tells me my order is expected to finish and the time that it ends up finishing is inaccurate and isn't updated at all. I get a notification that my order is done outside of the app but within the app the time is the only thing that is displayed for me to see.

Surveys:

- Customers: A survey about preferred delivery methods, mobile vs. desktop usage, and desired menu features.

Survey



Online Order Customer Survey

This form is designed to collect user opinions of online and mobile ordering systems as they stand

kyleagomes7@gmail.com Switch account

Not shared

* Indicates required question

Please specify your name *

Your answer

Do you use Mobile or Desktop or both when ordering food online? *

○ Mobile (Phones)

○ Desktop

What is your preferred delivery method when ordering food? *

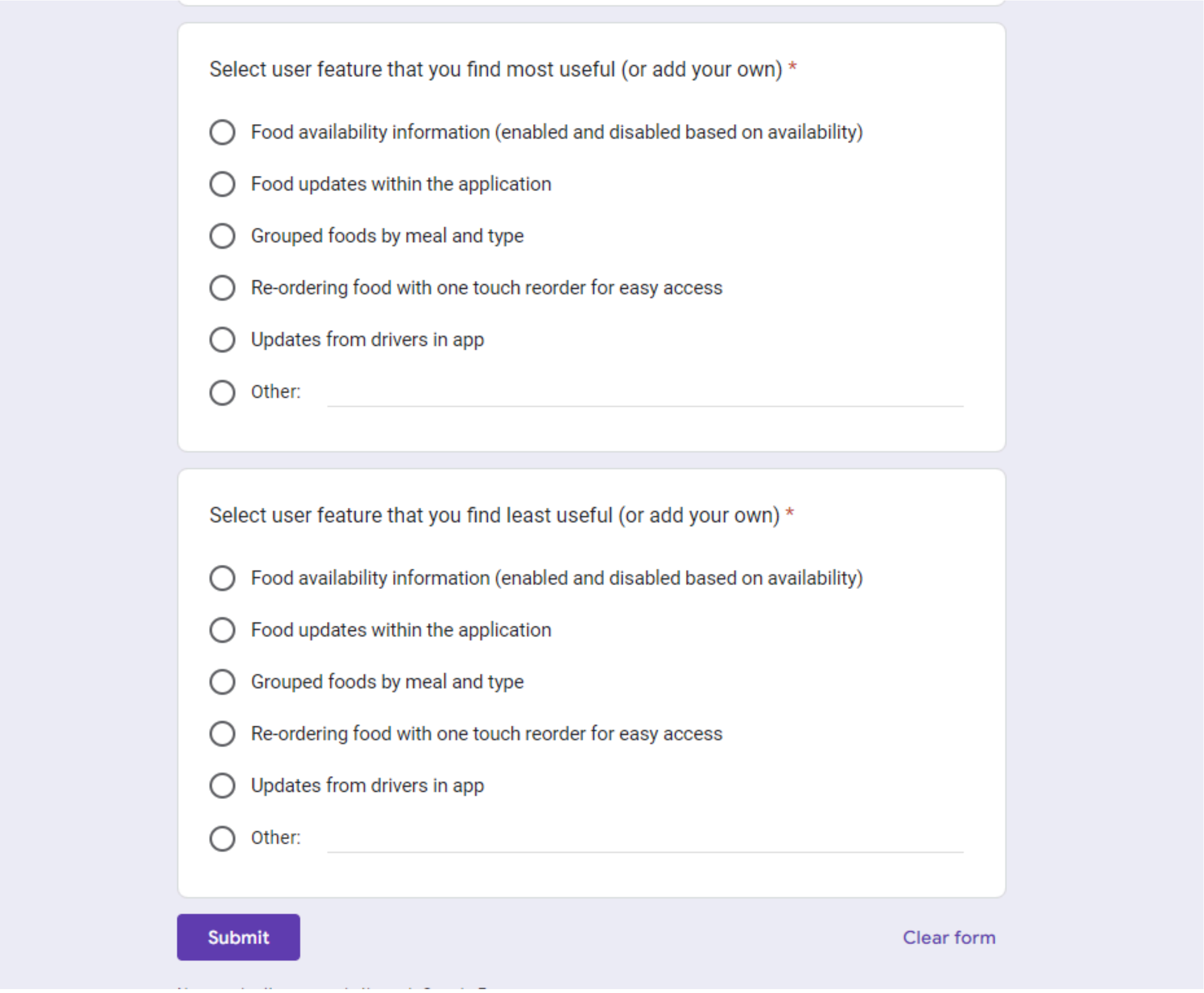
○ Curbside

○ Delivery Driver

○ Drive thru pick up

○ Pick up at counter

○ Other: ______



Select user feature that you find most useful (or add your own) *

○ Food availability information (enabled and disabled based on availability)

○ Food updates within the application

○ Grouped foods by meal and type

○ Re-ordering food with one touch reorder for easy access

○ Updates from drivers in app

○ Other: ____________

Select user feature that you find least useful (or add your own) *

○ Food availability information (enabled and disabled based on availability)

○ Food updates within the application

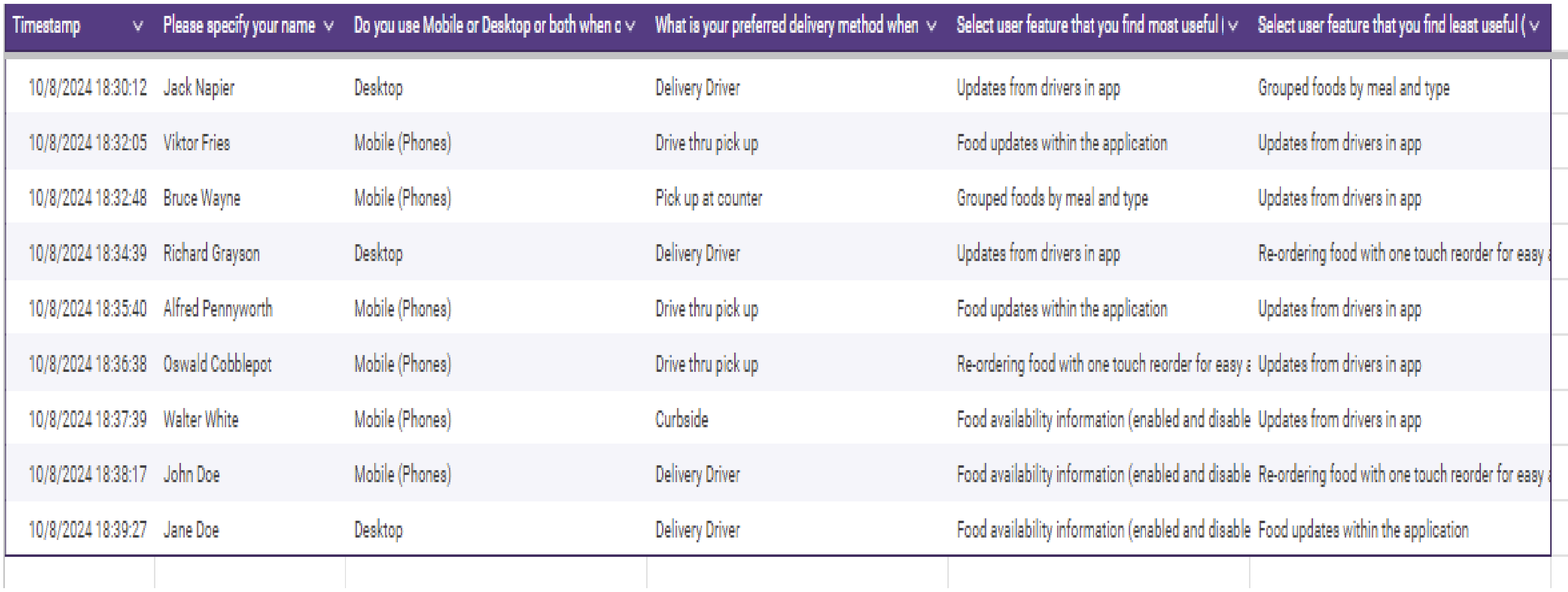
○ Grouped foods by meal and type

○ Re-ordering food with one touch reorder for easy access

○ Updates from drivers in app

○ Other: ____________

Gathered Responses



Timestamp	Please specify your name	Do you use Mobile or Desktop or both when o	What is your preferred delivery method when	Select user feature that you find most useful (	Select user feature that you find least useful (
10/8/2024 18:30:12	Jack Napier	Desktop	Delivery Driver	Updates from drivers in app	Grouped foods by meal and type
10/8/2024 18:32:05	Viktor Fries	Mobile (Phones)	Drive thru pick up	Food updates within the application	Updates from drivers in app
10/8/2024 18:32:48	Bruce Wayne	Mobile (Phones)	Pick up at counter	Grouped foods by meal and type	Updates from drivers in app
10/8/2024 18:34:39	Richard Grayson	Desktop	Delivery Driver	Updates from drivers in app	Re-ordering food with one touch reorder for easy a
10/8/2024 18:35:40	Alfred Pennyworth	Mobile (Phones)	Drive thru pick up	Food updates within the application	Updates from drivers in app
10/8/2024 18:36:38	Oswald Cobblepot	Mobile (Phones)	Drive thru pick up	Re-ordering food with one touch reorder for easy a	Updates from drivers in app
10/8/2024 18:37:39	Walter White	Mobile (Phones)	Curbside	Food availability information (enabled and disable	Updates from drivers in app
10/8/2024 18:38:17	John Doe	Mobile (Phones)	Delivery Driver	Food availability information (enabled and disable	Re-ordering food with one touch reorder for easy a
10/8/2024 18:39:27	Jane Doe	Desktop	Delivery Driver	Food availability information (enabled and disable	Food updates within the application

Procedure: Inventory is counted and logged at the end of every working day and restockings occur at the beginning or end of every day as needed.

1. Managers look at the incoming delivery information and mark items to be replaced
2. Employees count and mark food in the freezers and pantries on paper to hand to the manager
3. Employees count boxes, cups, and portion control devices and log them on paper to be given to the manager
4. Managers take logged information and input the details into the system which is sent to the headquarters and will schedule a delivery based on the projected date of projected run out of materials

Advantages	Disadvantages
Contacting and logging data directly to headquarters ensures stores always stocked	Hard copies from employees can lead to an incorrect count (loss, illegible hand writing etc)
Thorough employees can control portions well and ensure that stock is not leaving too quickly	Careless employees could lead to over served portions which could lead to stock selling out quickly
Menu items for consumers will always be in stock and can be logged by managers	

- Restaurants: Observe current ordering and inventory management processes to understand areas for improvement.

Ordering Observation

Date: 9/15/2024 Manager: Jameson Schmidt ID: 10354326

Procedure overview: customers can order food on their phones, computers and select delivery or pickup to receive the food.

1. Customer loads up app or website or drives to a restaurant location
2. Customer taps on the food item that they want, selects a quantity and then adds the item to cart
3. When the customer is ready to check out and complete the order then they can select a payment method
4. The customer selects a delivery method and can select between curbside, delivery, or drive through pick up
5. Customer is shown information on the completion process and is given updates via notification along the way
6. Upon completion the customer is told that the order is ready and/or delivered

Advantages	Disadvantages
All food is centralized and easy to choose from so	No way for cash users to pay for food over online

customers know exactly where it is/where to find it	order and mobile order
Many options for pick up allows for flexible, easy and quick ways for customers to pick up food	Updates are only given over notification and not within the application itself
Multiple ways to order adds flexibility for customers	Customers cant see which items are out of stock at the location until they try to place the order
	No way to keep tabs on drivers that mess with the order to make sure that they are reprimanded

Document Review:

- Review existing systems or manuals to ensure alignment with the required system functionalities.

List of Data Items:

Customer Data:

- Customer ID, Name, Email, Phone, Address
- Login Credentials (Username, Password)
- Order History, Payment Methods (Credit Card, Apple Pay, PayPal…)

Location Data:

- Location ID, Address, Operating Hours, Contact Info
- Specials, Location Status (Open/Closed)

Staff Data:

- Staff ID, Name, Position, Contact Info
- Shift Schedule

Menu Data:

- Menu ID, Menu Name (e.g. Breakfast, Lunch)
- Menu Item ID, Item Name, Description, Price, Category (Appetizer, Main Course...)
- Availability

Order Data:

- Order ID, Customer ID, Date/Time, Items Ordered
- Total Price, Order Status (Pending, Preparing, Delivered, Canceled)
- OrderItemID, Quantity

Delivery Method Data:

- Delivery Method ID, Type (Curbside, Drive-Thru, Home Delivery)

Transaction Data:

- Transaction ID, Order ID, Payment Method, Amount
- Status (Paid, Refunded), Date/Time

Inventory Data:

- Inventory Item ID, Name, Quantity, Supplier
- Expiration Date, Cost Price, Available Quantity

Customer Feedback/Review Data:

- Review ID, Customer ID, Order ID, Rating (1-5 stars)
- Review Text, Response from Admin

Customer Support Request Data:

- Request ID, Customer ID, Issue Type, Description
- Status (Pending, Resolved), Response

Refund Request Data:

- Refund ID, Order ID, Amount, Reason
- Status (Pending, Approved, Rejected)

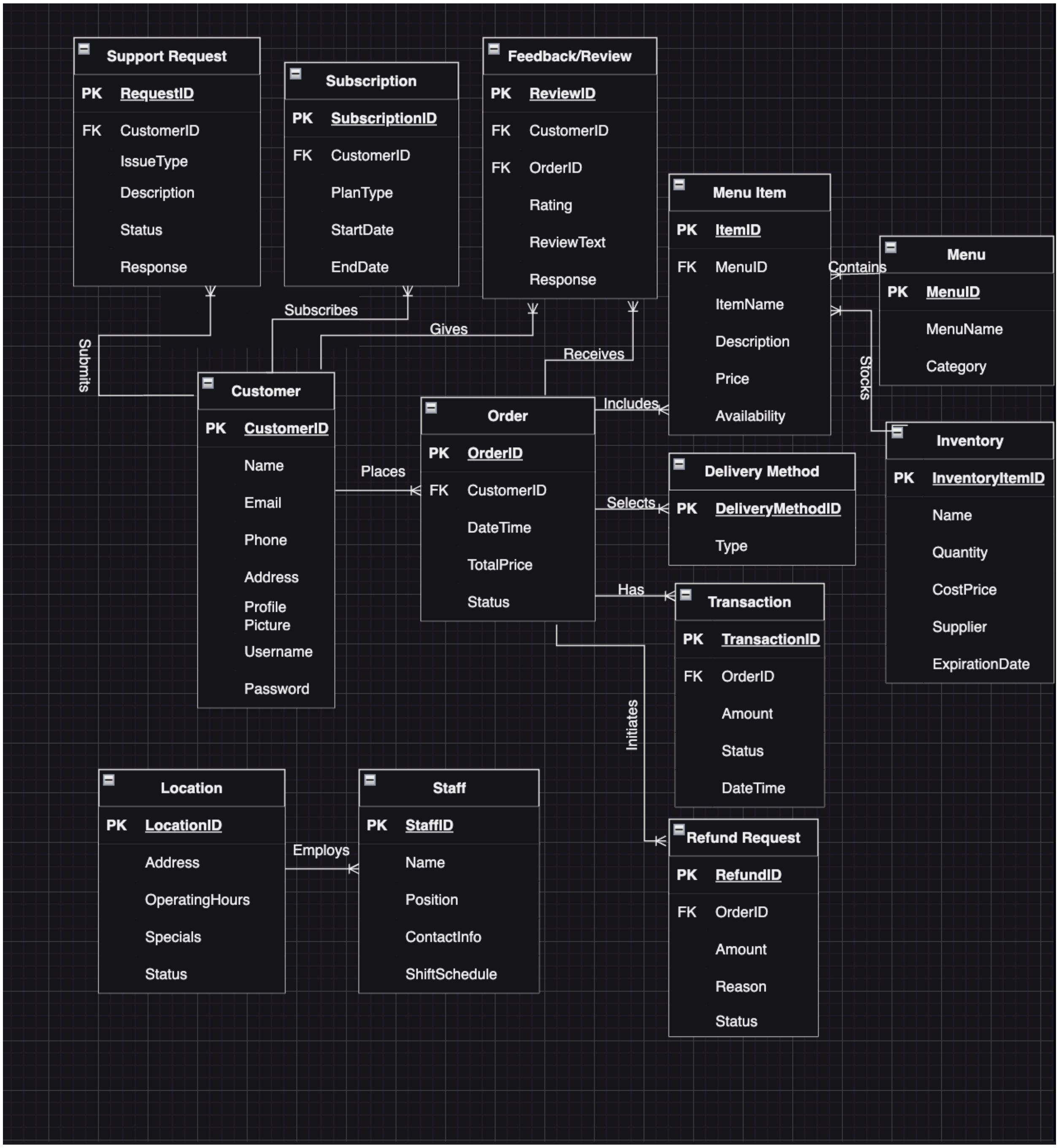
Subscription Plan Data:

- Subscription ID, Customer ID, Plan Type
- Start Date, End Date, Payment Information

<u>Complete E-R Diagram for the data model needed to support the proposed features (mission objectives):</u>

Assumptions:

- **Customer**: Each customer can place multiple orders, leave reviews, and submit support requests. Each customer may have a subscription plan.
- **Order**: Orders are placed by customers and may include multiple menu items. Each order is processed with a specific transaction and delivery method.
- **Menu and Menu Items**: Menus include multiple items (e.g., breakfast, lunch), and menu items are tracked in inventory.
- **Staff and Location**: Each location can have multiple staff members who manage inventory, orders, and support customer services.
- **Inventory**: Tracks each menu item's availability, supplier, and quantity, ensuring accurate stock management.


Support Request
PK RequestID
FK CustomerID
IssueType
Description
Status
Response
Subscription
PK SubscriptionID
FK CustomerID
PlanType
StartDate
EndDate
Feedback/Review
PK ReviewID
FK CustomerID
FK OrderID
Rating
ReviewText
Response
Menu Item
PK ItemID
FK MenuID
ItemName
Description
Price
Availability
Menu
PK MenuID
MenuName
Category
Contains
Stocks
Subscribes
Gives
Receives
Submits
Customer
PK CustomerID
Name
Email
Phone
Address
Profile Picture
Username
Password
Places
Order
PK OrderID
FK CustomerID
DateTime
TotalPrice
Status
Includes
Selects
Has
Initiates
Delivery Method
PK DeliveryMethodID
Type
Transaction
PK TransactionID
FK OrderID
Amount
Status
DateTime
Inventory
PK InventoryItemID
Name
Quantity
CostPrice
Supplier
ExpirationDate
Location
PK LocationID
Address
OperatingHours
Specials
Status
Employs
Staff
PK StaffID
Name
Position
ContactInfo
ShiftSchedule
Refund Request
PK RefundID
FK OrderID
Amount
Reason
Status

Schema definitions from ER diagram in 3NF:

The following schemas were derived from the E-R Diagram and normalized to 3NF, ensuring that all attributes are dependent only on the primary key without transitive dependencies.

Customer

- **Customer**(CustomerID PK, Name, Email, Phone, Address, Username, Password)

Location

- **Location**(LocationID PK, Address, OperatingHours, Specials, Status)

Staff

- **Staff**(StaffID PK, Name, Position, ContactInfo, ShiftSchedule)

Menu and Menu Item

- **Menu**(MenuID PK, MenuName, Category)
- **MenuItem**(ItemID PK, MenuID FK, ItemName, Description, Price, Availability)

Order

- **Order**(OrderID PK, CustomerID FK, LocationID FK, DateTime, TotalPrice, Status)

Delivery Method

- **DeliveryMethod**(DeliveryMethodID PK, Type)

Transaction

- **Transaction**(TransactionID PK, OrderID FK, Amount, Status, DateTime)

Inventory

- **Inventory**(InventoryItemID PK, Name, Quantity, CostPrice, Supplier, ExpirationDate)

Feedback/Review

- **FeedbackReview**(ReviewID PK, CustomerID FK, OrderID FK, Rating, ReviewText, Response)

Support Request

- **SupportRequest**(RequestID PK, CustomerID FK, IssueType, Description, Status, Response)

Refund Request

- **RefundRequest**(RefundID PK, OrderID FK, Amount, Reason, Status)

Subscription

- **Subscription**(SubscriptionID PK, CustomerID FK, PlanType, StartDate, EndDate, PaymentInfo)

Each schema was examined and refined to ensure 3NF compliance, eliminating partial and transitive dependencies where present.

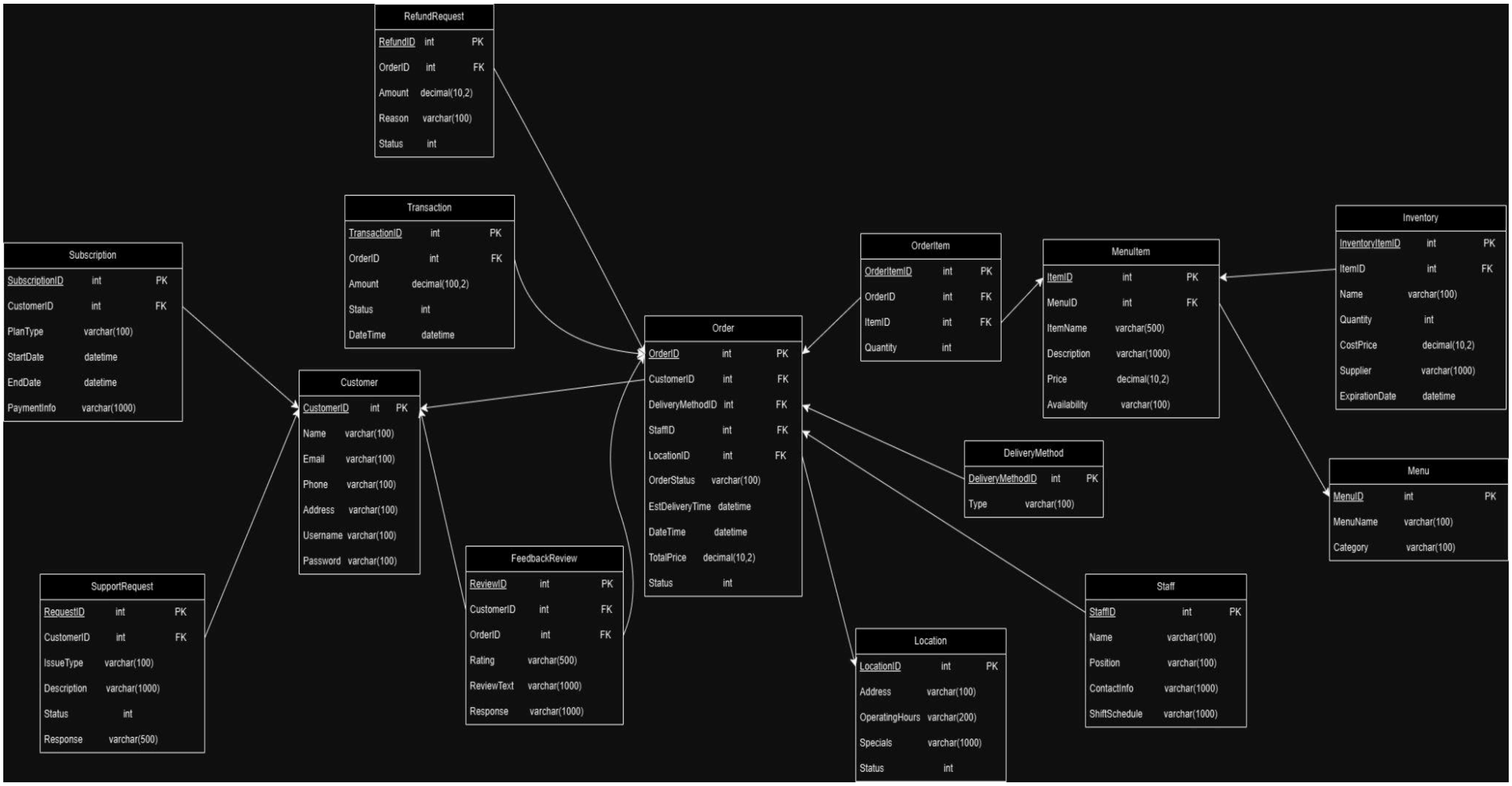
Functional Dependencies:

The functional dependencies for each schema are defined below, grouped by schema name:

- **Customer**: CustomerID → {Name, Email, Phone, Address, Username, Password}
- **Location**: LocationID → {Address, OperatingHours, Specials, Status}
- **Staff**: StaffID → {Name, Position, ContactInfo, ShiftSchedule}
- **MenuItem**: ItemID → {MenuID, ItemName, Description, Price, Availability}
- **Order**: OrderID → {CustomerID, LocationID, DateTime, TotalPrice, Status}
- **Transaction**: TransactionID → {OrderID, Amount, Status, DateTime}
- **Inventory**: InventoryItemID → {Name, Quantity, CostPrice, Supplier, ExpirationDate}
- **FeedbackReview**: ReviewID →{CustomerID, OrderID, Rating, ReviewText, Response}
- **SupportRequest**: RequestID →{CustomerID, IssueType, Description, Status, Response}
- **RefundRequest**: RefundID → {OrderID, Amount, Reason, Status}
- **Subscription**: SubscriptionID →{CustomerID, PlanType, StartDate, EndDate, PaymentInfo}

These functional dependencies were used to ensure each schema is normalized to 3NF.

Schema Diagram for schema definitions:



DDL Code for schema:

```
CREATE TABLE `customer` (
  `customerID` int(11) NOT NULL,
  `name` varchar(100) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  `phone` varchar(15) DEFAULT NULL,
  `address` varchar(255) DEFAULT NULL,
  `username` varchar(50) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  `role` enum('customer','admin') DEFAULT 'customer'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `delivery_method` (
  `deliveryMethodID` int(11) NOT NULL,
  `type` varchar(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `feedback` (
  `feedbackID` int(11) NOT NULL,
  `customerID` int(11) DEFAULT NULL,
  `orderID` int(11) DEFAULT NULL,
  `rating` int(11) DEFAULT NULL CHECK (`rating` between 1 and 5),
  `comment` text DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `feedback_review` (
  `reviewID` int(11) NOT NULL,
  `customerID` int(11) DEFAULT NULL,
  `orderID` int(11) DEFAULT NULL,
  `rating` int(11) DEFAULT NULL,
  `reviewText` text DEFAULT NULL,
  `response` text DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `inventory` (
  `inventoryItemID` int(11) NOT NULL,
  `itemID` int(11) DEFAULT NULL,
  `name` varchar(100) DEFAULT NULL,
  `quantity` int(11) DEFAULT NULL,
  `costPrice` decimal(10,2) DEFAULT NULL,
  `supplier` varchar(100) DEFAULT NULL,
  `expirationDate` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
CREATE TABLE `location` (
  `locationID` int(11) NOT NULL,
  `address` varchar(255) DEFAULT NULL,
  `operatingHours` varchar(100) DEFAULT NULL,
  `specials` varchar(255) DEFAULT NULL,
  `status` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `menu` (
  `menuID` int(11) NOT NULL,
  `menuName` varchar(50) DEFAULT NULL,
  `category` varchar(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `menu_item` (
  `itemID` int(11) NOT NULL,
  `menuID` int(11) DEFAULT NULL,
  `itemName` varchar(100) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `price` decimal(10,2) DEFAULT NULL,
  `availability` tinyint(1) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `orderItem` (
  `orderItemID` int(11) NOT NULL,
  `orderID` int(11) DEFAULT NULL,
  `itemID` int(11) DEFAULT NULL,
  `quantity` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `orders` (
  `orderID` int(11) NOT NULL,
  `customerID` int(11) DEFAULT NULL,
  `deliveryMethodID` int(11) DEFAULT NULL,
  `locationID` int(11) DEFAULT NULL,
  `dateTime` datetime DEFAULT NULL,
  `totalPrice` decimal(10,2) DEFAULT NULL,
  `status` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `report` (
  `reportID` int(11) NOT NULL,
  `customerID` int(11) DEFAULT NULL,
  `issueType` varchar(50) DEFAULT NULL,
  `description` text DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `staff` (
  `staffID` int(11) NOT NULL,
  `name` varchar(100) DEFAULT NULL,
  `position` varchar(50) DEFAULT NULL,
  `contactInfo` varchar(255) DEFAULT NULL,
  `shiftSchedule` varchar(100) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `support_request` (
  `requestID` int(11) NOT NULL,
  `customerID` int(11) DEFAULT NULL,
  `issueType` varchar(50) DEFAULT NULL,
  `description` text DEFAULT NULL,
  `status` int(11) DEFAULT NULL,
  `response` text DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `transaction` (
  `transactionID` int(11) NOT NULL,
  `orderID` int(11) DEFAULT NULL,
  `amount` decimal(10,2) DEFAULT NULL,
  `status` varchar(50) DEFAULT NULL,
  `dateTime` datetime DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Indexes for dumped tables
--



ALTER TABLE `customer`
  ADD PRIMARY KEY (`customerID`);


ALTER TABLE `delivery_method`
  ADD PRIMARY KEY (`deliveryMethodID`);


ALTER TABLE `feedback`
```

```
  ADD PRIMARY KEY (`feedbackID`),
  ADD KEY `customerID` (`customerID`),
  ADD KEY `orderID` (`orderID`);


ALTER TABLE `feedback_review`
  ADD PRIMARY KEY (`reviewID`),
  ADD KEY `customerID` (`customerID`),
  ADD KEY `orderID` (`orderID`);


ALTER TABLE `inventory`
  ADD PRIMARY KEY (`inventoryItemID`),
  ADD KEY `itemID` (`itemID`);


ALTER TABLE `location`
  ADD PRIMARY KEY (`locationID`);


ALTER TABLE `menu`
  ADD PRIMARY KEY (`menuID`);


ALTER TABLE `menu_item`
  ADD PRIMARY KEY (`itemID`),
  ADD KEY `menuID` (`menuID`);


ALTER TABLE `orderItem`
  ADD PRIMARY KEY (`orderItemID`),
  ADD KEY `orderID` (`orderID`),
  ADD KEY `itemID` (`itemID`);


ALTER TABLE `orders`
  ADD PRIMARY KEY (`orderID`),
  ADD KEY `customerID` (`customerID`),
  ADD KEY `deliveryMethodID` (`deliveryMethodID`),
  ADD KEY `locationID` (`locationID`);


ALTER TABLE `report`
  ADD PRIMARY KEY (`reportID`),
```

```sql
  ADD KEY `customerID` (`customerID`);


ALTER TABLE `staff`
  ADD PRIMARY KEY (`staffID`);


ALTER TABLE `support_request`
  ADD PRIMARY KEY (`requestID`),
  ADD KEY `customerID` (`customerID`);


ALTER TABLE `transaction`
  ADD PRIMARY KEY (`transactionID`),
  ADD KEY `orderID` (`orderID`);

--
-- AUTO_INCREMENT for dumped tables
--


ALTER TABLE `customer`
  MODIFY `customerID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;


ALTER TABLE `delivery_method`
  MODIFY `deliveryMethodID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;


ALTER TABLE `feedback`
  MODIFY `feedbackID` int(11) NOT NULL AUTO_INCREMENT;


ALTER TABLE `feedback_review`
  MODIFY `reviewID` int(11) NOT NULL AUTO_INCREMENT;


ALTER TABLE `inventory`
  MODIFY `inventoryItemID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;


ALTER TABLE `location`
  MODIFY `locationID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
```

```
ALTER TABLE `menu`
  MODIFY `menuID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;


ALTER TABLE `menu_item`
  MODIFY `itemID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;


ALTER TABLE `orderItem`
  MODIFY `orderItemID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=13;


ALTER TABLE `orders`
  MODIFY `orderID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;


ALTER TABLE `report`
  MODIFY `reportID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;


ALTER TABLE `staff`
  MODIFY `staffID` int(11) NOT NULL AUTO_INCREMENT;


ALTER TABLE `support_request`
  MODIFY `requestID` int(11) NOT NULL AUTO_INCREMENT;


ALTER TABLE `transaction`
  MODIFY `transactionID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

--
-- Constraints for dumped tables
--


ALTER TABLE `feedback`
  ADD CONSTRAINT `feedback_ibfk_1` FOREIGN KEY (`customerID`) REFERENCES `customer`
(`customerID`) ON DELETE CASCADE,
  ADD CONSTRAINT `feedback_ibfk_2` FOREIGN KEY (`orderID`) REFERENCES `orders`
(`orderID`) ON DELETE CASCADE;
```

```
ALTER TABLE `feedback_review`
  ADD CONSTRAINT `feedback_review_ibfk_1` FOREIGN KEY (`customerID`) REFERENCES
`customer` (`customerID`) ON DELETE SET NULL,
  ADD CONSTRAINT `feedback_review_ibfk_2` FOREIGN KEY (`orderID`) REFERENCES `orders`
(`orderID`) ON DELETE CASCADE;


ALTER TABLE `inventory`
  ADD CONSTRAINT `inventory_ibfk_1` FOREIGN KEY (`itemID`) REFERENCES `menu_item`
(`itemID`) ON DELETE SET NULL;


ALTER TABLE `menu_item`
  ADD CONSTRAINT `menu_item_ibfk_1` FOREIGN KEY (`menuID`) REFERENCES `menu`
(`menuID`) ON DELETE CASCADE;


ALTER TABLE `orderItem`
  ADD CONSTRAINT `orderitem_ibfk_1` FOREIGN KEY (`orderID`) REFERENCES `orders`
(`orderID`) ON DELETE CASCADE,
  ADD CONSTRAINT `orderitem_ibfk_2` FOREIGN KEY (`itemID`) REFERENCES `menu_item`
(`itemID`) ON DELETE CASCADE;


ALTER TABLE `orders`
  ADD CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`customerID`) REFERENCES `customer`
(`customerID`) ON DELETE SET NULL,
  ADD CONSTRAINT `orders_ibfk_2` FOREIGN KEY (`deliveryMethodID`) REFERENCES
`delivery_method` (`deliveryMethodID`) ON DELETE SET NULL,
  ADD CONSTRAINT `orders_ibfk_3` FOREIGN KEY (`locationID`) REFERENCES `location`
(`locationID`) ON DELETE SET NULL;


ALTER TABLE `report`
  ADD CONSTRAINT `report_ibfk_1` FOREIGN KEY (`customerID`) REFERENCES `customer`
(`customerID`) ON DELETE CASCADE;


ALTER TABLE `support_request`
  ADD CONSTRAINT `support_request_ibfk_1` FOREIGN KEY (`customerID`) REFERENCES
`customer` (`customerID`) ON DELETE SET NULL;
```

```
ALTER TABLE `transaction`
  ADD CONSTRAINT `transaction_ibfk_1` FOREIGN KEY (`orderID`) REFERENCES `orders`
(`orderID`) ON DELETE CASCADE;
COMMIT;
```

SQL Queries and DML Statements (lines in between indicate code from different files):

```
SELECT * FROM delivery_method
```

```
SELECT * FROM location
```

```
$stmt = $conn->prepare("SELECT customerID, password, role FROM customer WHERE username =
?");
$stmt->bind_param("s", $username);
$stmt->execute();
$result = $stmt->get_result();
```

```
query("SELECT m.menuName, mi.itemName, mi.description, mi.price, mi.availability
            FROM menu m
            JOIN menu_item mi ON m.menuID = mi.menuID");
```

```
foreach ($itemIDs as $key => $itemID) {
    $stmt = $conn->prepare("SELECT price FROM menu_item WHERE itemID = ?");
    $stmt->bind_param("i", $itemID);
    $stmt->execute();
    $result = $stmt->get_result();
    $item = $result->fetch_assoc();
    $totalPrice += $item["price"] * $quantities[$key];
  }
```

```
$stmt = $conn->prepare("INSERT INTO orders (customerID, deliveryMethodID, locationID, dateTime,
totalPrice, status)
              VALUES (?, ?, ?, NOW(), ?, 0)");
  $stmt->bind_param("iiid", $customerID, $deliveryMethodID, $locationID, $totalPrice);
  $stmt->execute();
  $orderID = $conn->insert_id;
```

```
foreach ($itemIDs as $key => $itemID) {
    $quantity = $quantities[$key];
```

```
        $stmt = $conn->prepare("INSERT INTO orderItem (orderID, itemID, quantity) VALUES (?, ?, ?)");
        $stmt->bind_param("iii", $orderID, $itemID, $quantity);
        $stmt->execute();
    }
```

```
SELECT itemID, itemName, price FROM menu_item WHERE availability = 1;
```

```
// Fetch customer details
$stmt = $conn->prepare("SELECT name, email, phone, address FROM customer WHERE customerID =
?");
$stmt->bind_param("i", $customerID);
$stmt->execute();
$result = $stmt->get_result();
$user = $result->fetch_assoc();

$stmt->close();
```

```
$stmt = $conn->prepare("INSERT INTO report (customerID, issueType, description) VALUES (?, ?, ?)");
    $stmt->bind_param("iss", $customerID, $issueType, $description);
```

```
// Update customer details
    $stmt = $conn->prepare("UPDATE customer SET email = ?, phone = ?, address = ? WHERE
customerID = ?");
    $stmt->bind_param("sssi", $email, $phone, $address, $customerID);
```

```
$result = $conn->query("
    SELECT
        o.orderID,
        c.name AS customerName,
        o.dateTime,
        o.totalPrice,
        d.type AS deliveryMethod,
        l.address AS location,
        o.status
    FROM orders o
    JOIN customer c ON o.customerID = c.customerID
    JOIN delivery_method d ON o.deliveryMethodID = d.deliveryMethodID
    JOIN location l ON o.locationID = l.locationID
    ORDER BY o.dateTime DESC
");
```

```
query("
    SELECT
        t.transactionID,
        o.orderID,
        c.name AS customerName,
        t.amount,
        t.status,
        t.dateTime
    FROM transaction t
    JOIN orders o ON t.orderID = o.orderID
    JOIN customer c ON o.customerID = c.customerID
    ORDER BY t.dateTime DESC
");
```

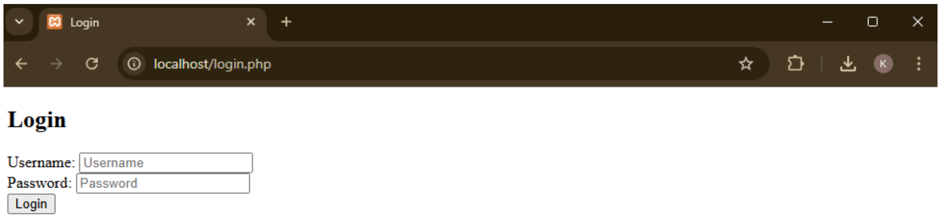
SQL Data Insertion Statements:

```
INSERT INTO `customer` (`customerID`, `name`, `email`, `phone`, `address`, `username`,
`password`, `role`) VALUES
(1, 'Alice Johnson', 'alice.johnson@example.com', '123-456-7890', '123 Maple St', 'alicej',
'$2b$12$x.sodpcpYLrwHWU7hN9UD.jH3qb1w8r0MIjapEtVNEusLvPLwvE1i', 'customer'),
(2, 'Bob Smith', 'bob.smith@example.com', '234-567-8901', '456 Oak St', 'bobsmith',
'$2b$12$7UvQY95qcmkDPZgfWSqtY.lip58xN7G4Rsb7tPA6BgyDVwujbfQ5m', 'customer'),
(3, 'John Doe', 'john.doe@example.com', '1234567890', '123 Elm St', 'johndoe',
'$2b$12$x.sodpcpYLrwHWU7hN9UD.jH3qb1w8r0MIjapEtVNEusLvPLwvE1i', 'customer'),
(4, 'Alice Johnson', 'alice.johnson@example.com', '123-456-7890', '123 Maple St', 'alicej',
'$2b$12$x.sodpcpYLrwHWU7hN9UD.jH3qb1w8r0MIjapEtVNEusLvPLwvE1i', 'customer'),
(5, 'Admin User', 'admin@example.com', '000-000-0000', 'Admin Office', 'admin',
'$2y$10$C2T3TN3xY5.onDiD0wjeEuSxi98xIDUjTnikTqxHSsZnfhunQR40m', 'admin');

INSERT INTO `delivery_method` (`deliveryMethodID`, `type`) VALUES
(1, 'Drive-thru'),
(2, 'Curbside'),
(3, 'Home Delivery');

INSERT INTO `inventory` (`inventoryItemID`, `itemID`, `name`, `quantity`, `costPrice`, `supplier`,
`expirationDate`) VALUES
(1, 1, 'Pancake Mix', 50, 2.99, 'FoodSupplier Inc.', '2024-12-31'),
(2, 2, 'Eggs', 30, 1.99, 'DairyBest', '2024-11-30'),
(3, 3, 'Tea Bags', 100, 0.99, 'TeaMasters', '2025-01-15'),
(4, 4, 'Beverage Cups', 200, 0.49, 'CupWorld', '2025-06-01');

INSERT INTO `location` (`locationID`, `address`, `operatingHours`, `specials`, `status`) VALUES
(1, '123 Main St, Cityville', '8 AM - 10 PM', 'Buy 1 Get 1 Free on Burgers', 1),
(2, '456 Elm St, Townsville', '9 AM - 9 PM', '20% off on Drinks during Happy Hour', 1);

INSERT INTO `menu` (`menuID`, `menuName`, `category`) VALUES
```

```
(1, 'Breakfast', 'Food'),
(2, 'Lunch', 'Food'),
(3, 'Drinks', 'Beverage');

INSERT INTO `menu_item` (`itemID`, `menuID`, `itemName`, `description`, `price`, `availability`)
VALUES
(1, 1, 'Pancakes', 'Fluffy golden pancakes served with syrup', 5.99, 1),
(2, 1, 'Scrambled Eggs', 'Classic scrambled eggs with a side of toast', 4.99, 1),
(3, 2, 'Burger Combo', 'Beef burger with fries and a drink', 9.99, 1),
(4, 3, 'Iced Tea', 'Freshly brewed iced tea', 2.99, 1);

INSERT INTO `orderItem` (`orderItemID`, `orderID`, `itemID`, `quantity`) VALUES
(1, 1, 1, 2),
(2, 1, 3, 1),
(3, 2, 2, 1),
(4, 2, 4, 2),
(5, 3, 1, 1),
(6, 3, 3, 1),
(7, 4, 1, 1),
(8, 4, 4, 1),
(9, 5, 1, 1),
(10, 5, 2, 1),
(11, 6, 1, 1),
(12, 6, 2, 1);

INSERT INTO `orders` (`orderID`, `customerID`, `deliveryMethodID`, `locationID`, `dateTime`,
`totalPrice`, `status`) VALUES
(1, 1, 1, 1, '2024-12-09 15:04:47', 17.48, 0),
(2, 2, 2, 2, '2024-12-09 15:04:47', 12.99, 1),
(3, 3, 3, 2, '2024-12-09 15:27:07', 15.98, 0),
(4, 3, 1, 2, '2024-12-09 15:27:53', 8.98, 0),
(5, 3, 2, 1, '2024-12-09 15:46:23', 10.98, 0),
(6, 3, 1, 1, '2024-12-09 16:43:55', 10.98, 0);

INSERT INTO `report` (`reportID`, `customerID`, `issueType`, `description`) VALUES
(1, 3, 'App Bug', 'fhfhfhf');

INSERT INTO `transaction` (`transactionID`, `orderID`, `amount`, `status`, `dateTime`) VALUES
(1, 1, 50.00, 'Completed', '2024-12-09 16:37:07'),
(2, 2, 75.00, 'Pending', '2024-12-09 16:37:07'),
(3, 3, 120.00, 'Cancelled', '2024-12-09 16:37:07');
```

Webpage UI Screenshots:

Login screen:

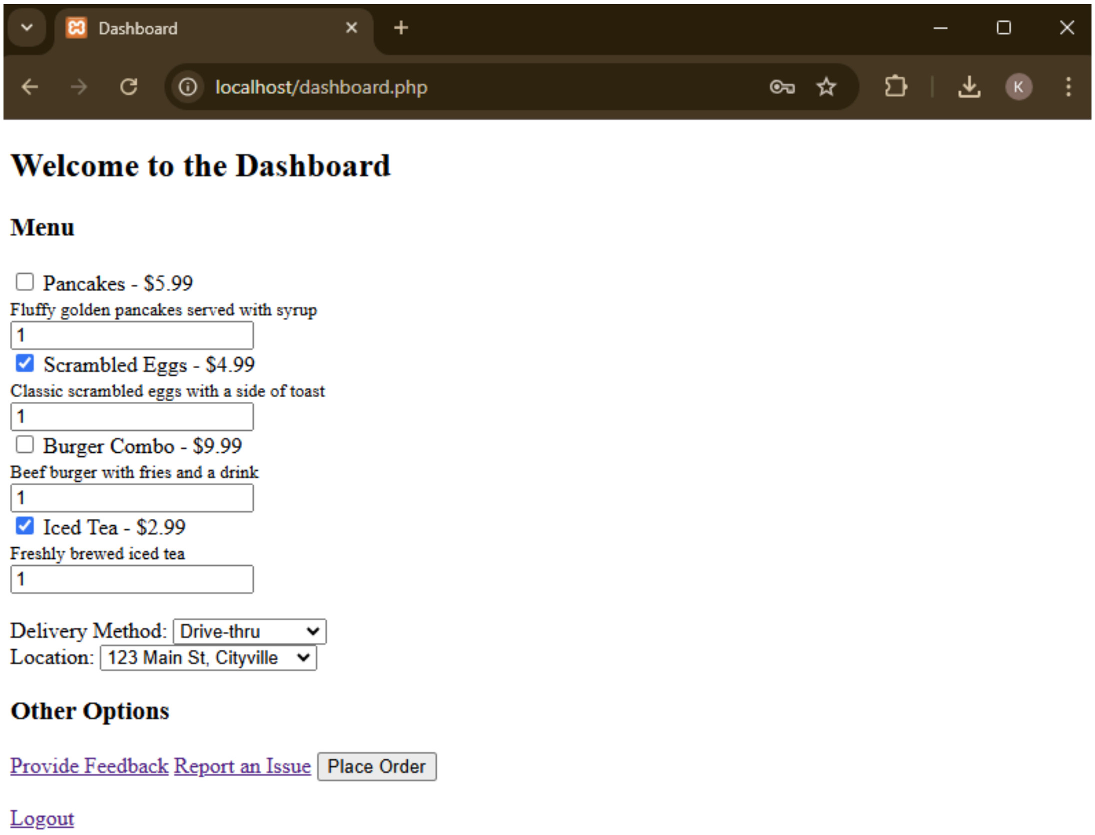


Login

Username: Username
Password: Password

(above) demonstrates common login page for both admin and customer user types.

User Dashboard Features



Welcome to the Dashboard

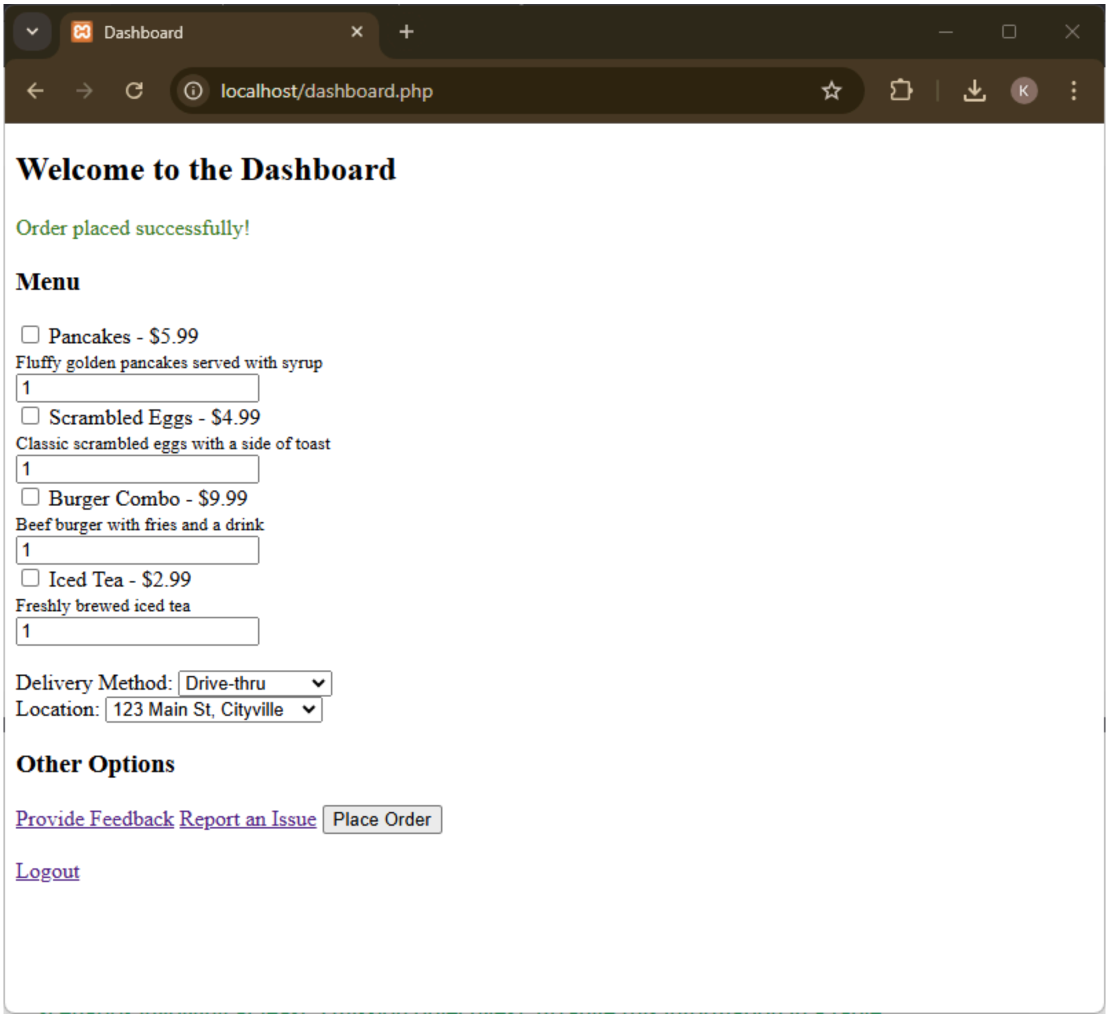
Menu

☐ Pancakes - $5.99
Fluffy golden pancakes served with syrup
1
☑ Scrambled Eggs - $4.99
Classic scrambled eggs with a side of toast
1
☐ Burger Combo - $9.99
Beef burger with fries and a drink
1
☑ Iced Tea - $2.99
Freshly brewed iced tea
1

Delivery Method: Drive-thru
Location: 123 Main St, Cityville

Other Options

(above) dashboard for users to be used for ordering, there are multiple mission objectives conveyed in the features presented including the menu, inputs for placing the order, delivery method selection, location selection, and finally the feedback and reports page hyperlinks.



Welcome to the Dashboard

Order placed successfully!

Menu

□ Pancakes - $5.99
Fluffy golden pancakes served with syrup
1

□ Scrambled Eggs - $4.99
Classic scrambled eggs with a side of toast
1

□ Burger Combo - $9.99
Beef burger with fries and a drink
1

□ Iced Tea - $2.99
Freshly brewed iced tea
1

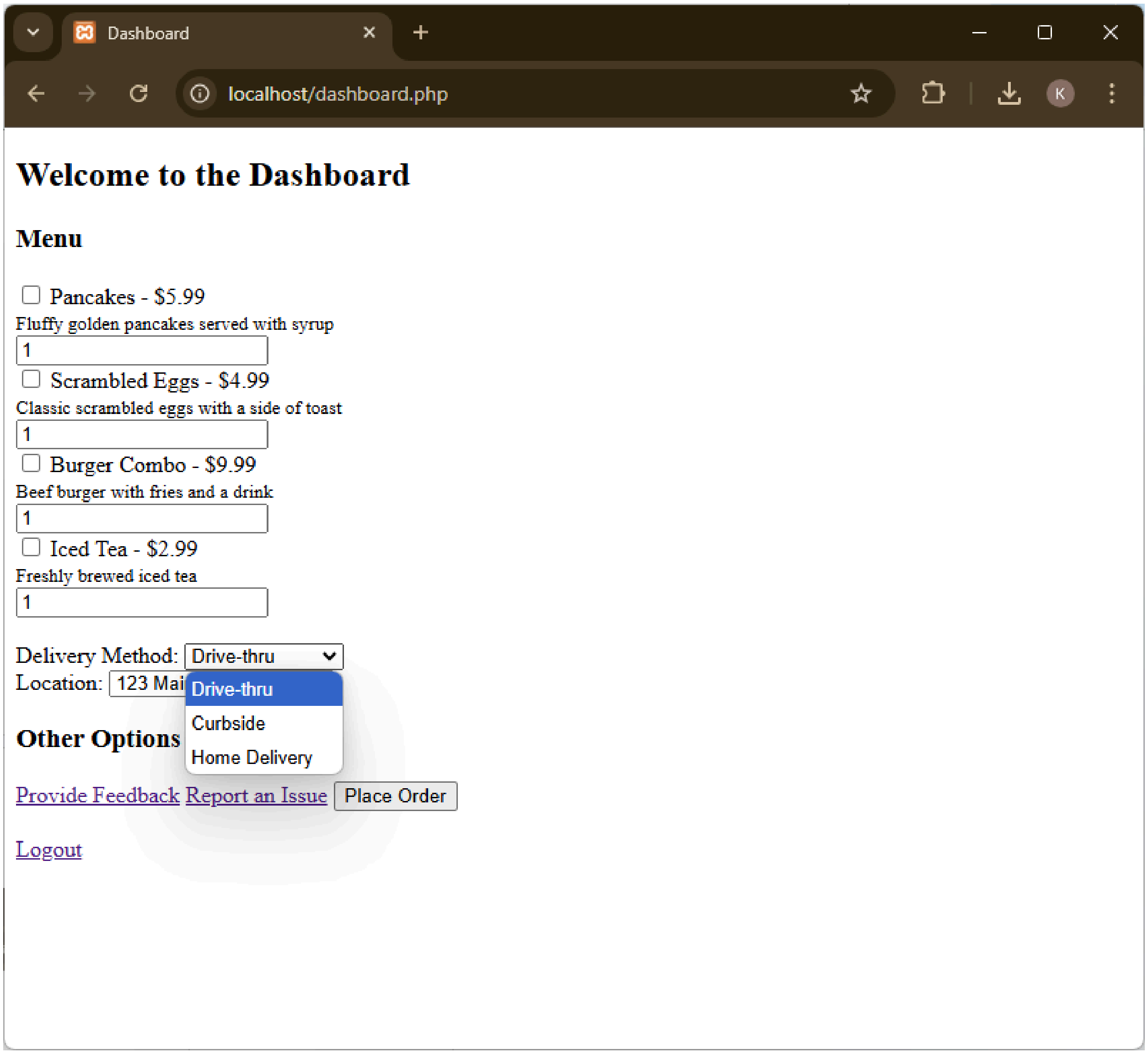
Delivery Method: Drive-thru
Location: 123 Main St, Cityville

Other Options

Provide Feedback Report an Issue Place Order

Logout

(above) user interaction showing that order information was submitted to the database successfully. This will be shown in both the database on the terminal and also on the order page in the administrator view.



Welcome to the Dashboard

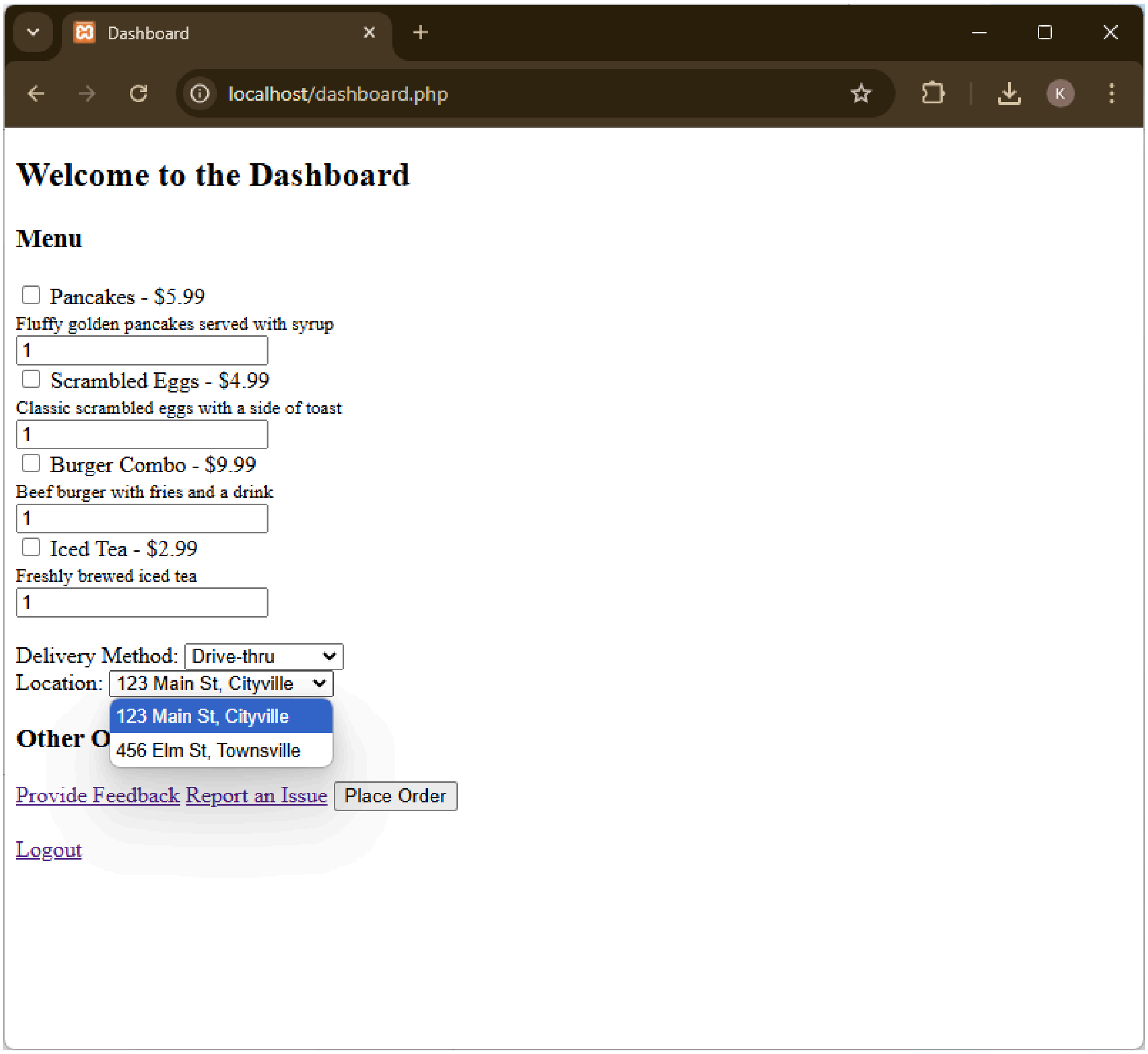
Menu

□ Pancakes - $5.99
Fluffy golden pancakes served with syrup
1
□ Scrambled Eggs - $4.99
Classic scrambled eggs with a side of toast
1
□ Burger Combo - $9.99
Beef burger with fries and a drink
1
□ Iced Tea - $2.99
Freshly brewed iced tea
1

Delivery Method: Drive-thru
Location: 123 Mai

Other Options

(above) shows functionality of the delivery methods dropdown, and shows communication between the website and database pulling information stored in the delivery_method table.



Welcome to the Dashboard

Menu

□ Pancakes - $5.99
Fluffy golden pancakes served with syrup
1

□ Scrambled Eggs - $4.99
Classic scrambled eggs with a side of toast
1

□ Burger Combo - $9.99
Beef burger with fries and a drink
1

□ Iced Tea - $2.99
Freshly brewed iced tea
1

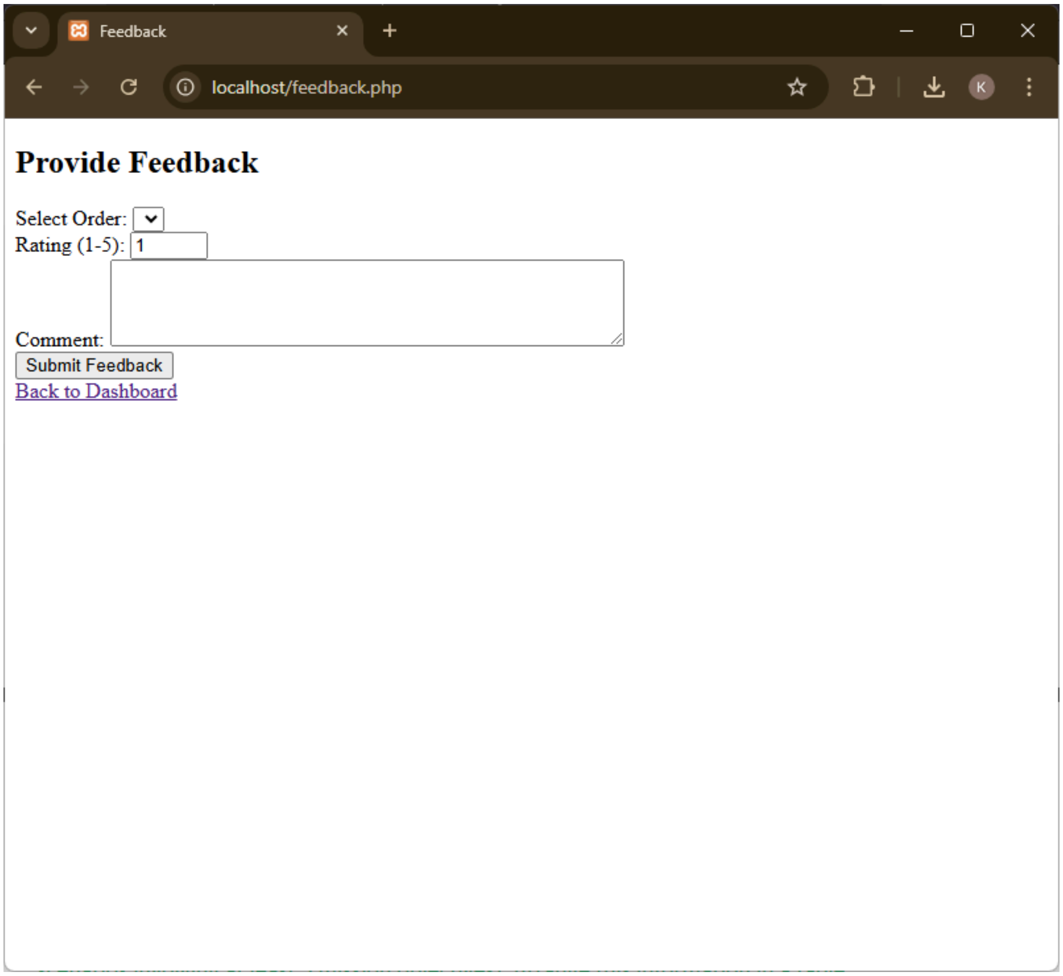
Delivery Method: Drive-thru
Location: 123 Main St, Cityville
123 Main St, Cityville
456 Elm St, Townsville

Other O

Provide Feedback Report an Issue Place Order

Logout

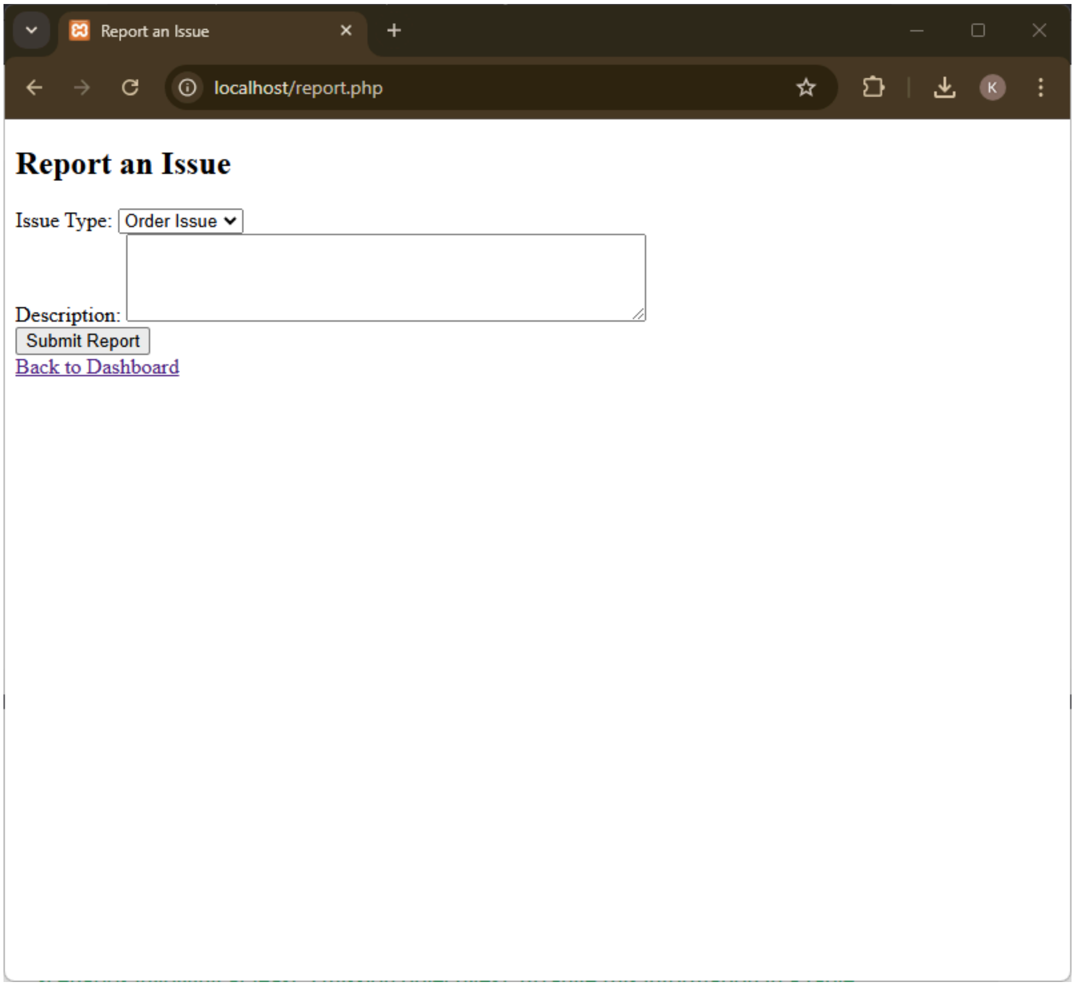
(above) shows functionality of the location dropdown, and shows communication between the website and database pulling information stored in the location table.



Provide Feedback

Select Order:
Rating (1-5): 1
Comment:

(above) UI for the feedback table presenting an opportunity for users to submit feedback for their order into the database.



Report an Issue

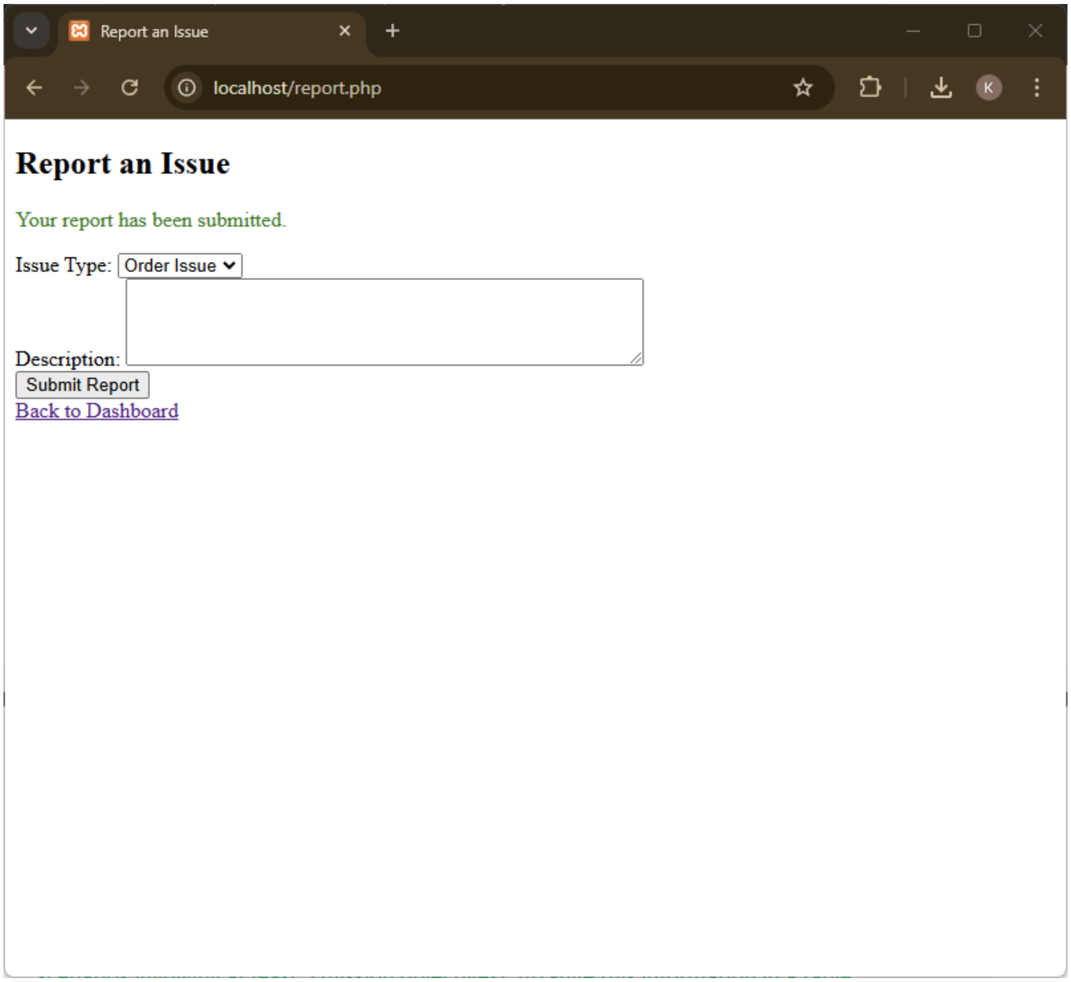
Issue Type: Order Issue

Description:

Submit Report

Back to Dashboard

(above) UI for the report table presenting an opportunity for users to submit a report on issues with orders, the user interface, or anything else into the database.



Report an Issue

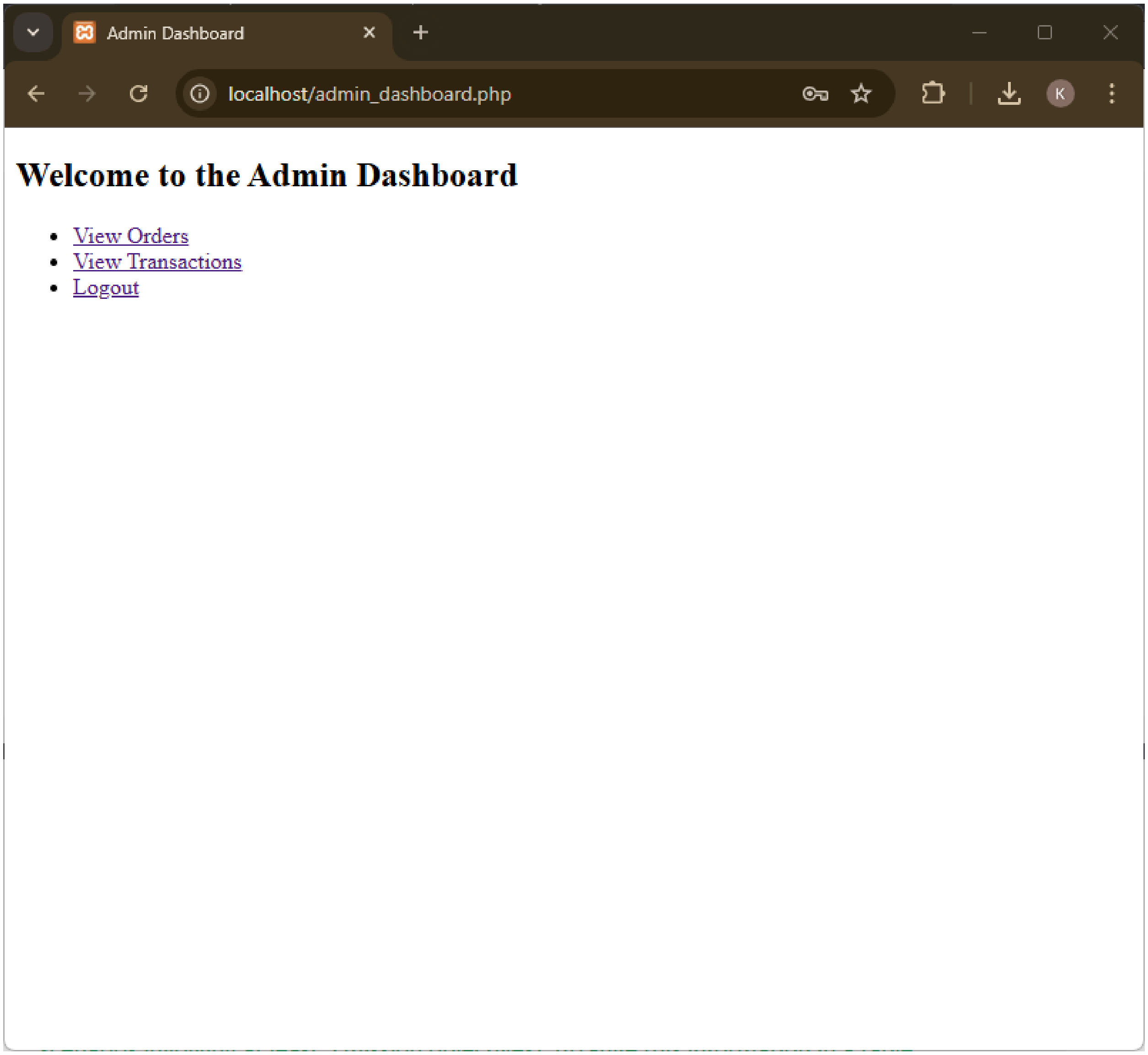
Your report has been submitted.

Issue Type: Order Issue

Description:

(above) shows the functionality of the report page and that it successfully communicates with the database by displaying a message to the user.

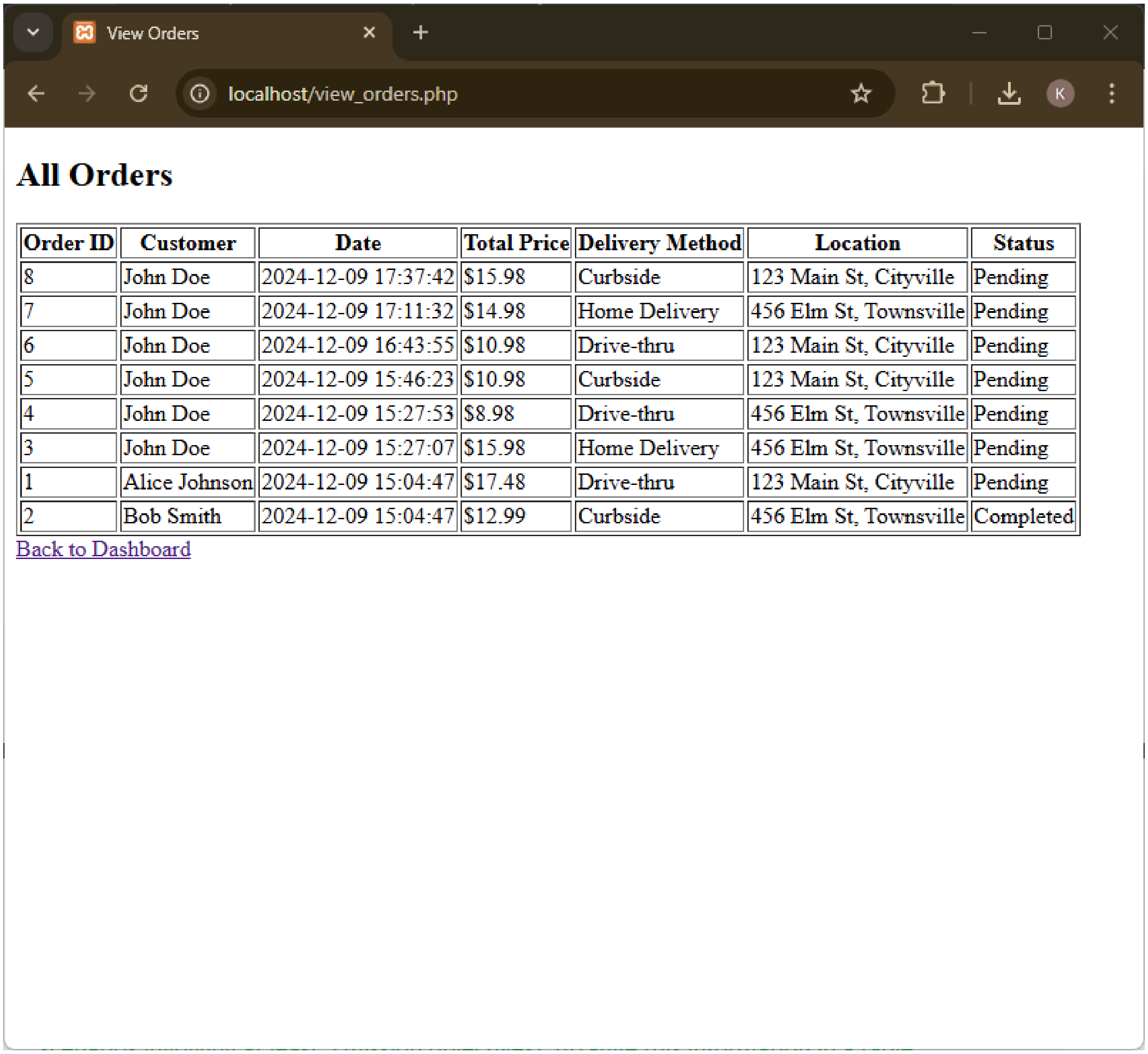
Admin Dashboard States and Features:



Welcome to the Admin Dashboard

- View Orders
- View Transactions
- Logout

(above) shows the dashboard for the administrative user, the admin has access to view orders and transaction data submitted to the database by the user.

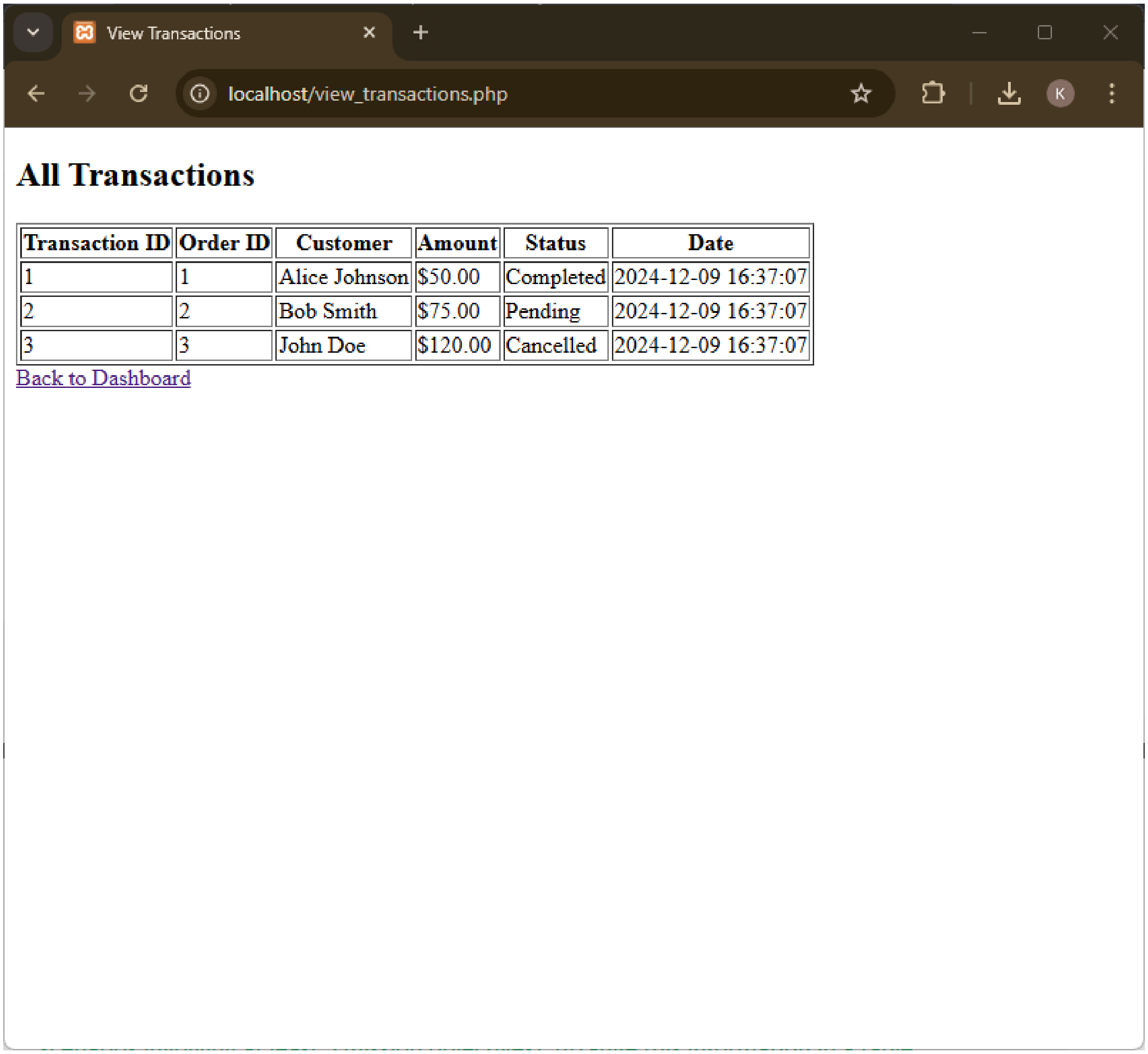


All Orders

Order ID	Customer	Date	Total Price	Delivery Method	Location	Status
8	John Doe	2024-12-09 17:37:42	$15.98	Curbside	123 Main St, Cityville	Pending
7	John Doe	2024-12-09 17:11:32	$14.98	Home Delivery	456 Elm St, Townsville	Pending
6	John Doe	2024-12-09 16:43:55	$10.98	Drive-thru	123 Main St, Cityville	Pending
5	John Doe	2024-12-09 15:46:23	$10.98	Curbside	123 Main St, Cityville	Pending
4	John Doe	2024-12-09 15:27:53	$8.98	Drive-thru	456 Elm St, Townsville	Pending
3	John Doe	2024-12-09 15:27:07	$15.98	Home Delivery	456 Elm St, Townsville	Pending
1	Alice Johnson	2024-12-09 15:04:47	$17.48	Drive-thru	123 Main St, Cityville	Pending
2	Bob Smith	2024-12-09 15:04:47	$12.99	Curbside	456 Elm St, Townsville	Completed

Back to Dashboard

(above) order page which pulls all aspects of the order table from the database and displays it to the administrative user so that they can complete the order.



All Transactions

Transaction ID	Order ID	Customer	Amount	Status	Date
1	1	Alice Johnson	$50.00	Completed	2024-12-09 16:37:07
2	2	Bob Smith	$75.00	Pending	2024-12-09 16:37:07
3	3	John Doe	$120.00	Cancelled	2024-12-09 16:37:07

Back to Dashboard

(above) shows all transaction data from customers to the administrator by communicating with the database, fetching the data and displaying it in the administrative view.

Test case descriptions, expected outcomes, and actual outcomes:

Test case #	**Test case description**	**Test Data**	**Expected outcome**	**Actual outcome**	**pass /fail**
1	Check response when valid username and password is entered for the customer end	Username: johndoe Password: password123	Login should be successful	Login successful	Pass
2	Check response when valid username and password is entered for the admin end	Username: admin Password: adminpassword	Login is successful	Login successful	Pass

3	**Mission obj**: read and display location information from the database to the menu page	Query: Select * from delivery_method;
		
		Data:
		(1, '123 Main St, Cityville', '8 AM - 10 PM', 'Buy 1 Get 1 Free on Burgers', 1),
		(2, '456 Elm St, Townsville', '9 AM - 9 PM', '20% off on Drinks during Happy Hour', 1);
			Data should be presented in a dropdown for users to select	Dropdown does display and allow Users to choose a location to order from	Pass

4	**Mission obj**: read and display menu items, descriptions and prices on the webpage for users to place orders with	Query: SELECT itemID, itemName, description, price, availability FROM menu_item WHERE availability = 1 Data: (1, 1, 'Pancakes', 'Fluffy golden pancakes served with syrup', 5.99, 1), (2, 1, 'Scrambled Eggs', 'Classic scrambled eggs with a side of toast', 4.99, 1), (3, 2, 'Burger Combo', 'Beef burger with fries and a drink', 9.99, 1), (4, 3, 'Iced Tea', 'Freshly brewed iced tea', 2.99, 1);	Data should be displayed on the screen with check boxes, descriptions , and item numbers	All items are presented on screen	Pass
5	**Mission obj**: allow users to create and customize new orders	Place order button sends data to the database when pressed	Data appears as a new order in the database	New order does appear in the database and on the admin order page	Pass
6	**Mission obj**: read and display delivery method information from the database to the menu page	Query: SELECT * FROM delivery_method; Data: (1, 'Drive-thru'),	Dropdown should allow the user to pick from 3	Dropdown appears and users can pick a delivery	Pass

		(2, 'Curbside'), (3, 'Home Delivery');	delivery methods	method before order processes	
7	**Mission obj:** create feedback for the restaurant for an order	Users can select an order, choose a rating and write a comment which upon clicking submit will send the response to a database	Upon clicking submit, feedback should appear in the database	Feedback appears in the table in the database	Pass
8	**Mission obj**: read and display new orders on the admin side	Query: SELECT o.orderID, c.name AS customerName, o.dateTime, o.totalPrice, d.type AS deliveryMethod, l.address AS location, o.status FROM orders o JOIN customer c ON o.customerID = c.customerID JOIN delivery_method d ON o.deliveryMethodID = d.deliveryMethodID JOIN location l ON o.locationID = l.locationID ORDER BY o.dateTime DESC	Display all order information and customer information in a table	Table appears and shows all orders	Pass
9	**Mission obj**: Show and update transaction information based on the customer on the admin side	Query: SELECT t.transactionID, o.orderID, c.name AS customerName, t.amount, t.status, t.dateTime FROM transaction t JOIN orders o ON t.orderID = o.orderID JOIN customer c ON o.customerID = c.customerID ORDER BY t.dateTime DESC	Show a table of fetched data on transactions from the database	Table appears with accurate informatio n and is updated with every new order from a specific user	Pass
10	**Mission obj**: create reports for issues on the computer or service end as well as for orders and anything else on the user end	Query: INSERT INTO report (customerID, issueType, description) VALUES (?, ?, ?)	Take data from user on reports and put it into the table in the database	When button is pressed, data appears in the table in the database	Pass

Contributions:

Youssef Rafik - Updated mission objectives, System Boundary Diagram (Draw.io), Samples of Responses from employed Fact Finding Techniques, List of Data Items, E-R diagram, Schema definitions, Functional dependencies, updated DDL statements for schema, SQL and DML features needed to support system features, SQL data insertion statements, UI design and development, Screenshots of webpages (UI) from your project, and test case descriptions, expected outcomes, and actual outcomes arranged in a table.

Kyle Gomes - Mission Statements, Mission Objectives, Samples of responses from employed fact finding techniques, Complete schema diagram for schema definitions, Updated mission objectives, updated list of data items, wrote DDL statements for schema, UI design and development, Screenshots of webpages (UI) from your project, and finally test case descriptions, expected outcomes, and actual outcomes arranged in a table.

.