Restaurant Ordering System – Components, Design, and Process

Drafted by Daniel Szelogowski, 02/08/2022

Component	Description	Languages/Platforms
Hosting	Serving web pages and database queries	Linux + Apache2 or Flask (Python)
Relational database	User accounts, transaction ledger, items	MySQL – easy to setup and interface with, doesn't require learning another language like NoSQL databases (Mongo, PostreSQL, DynamoDB, etc.)
Login/Registration	Create and parse user accounts – hash passwords, encrypted session login (Common cypher suite: RSA/ECB/OAEPWithSHA1AndMGF1Padding)	Same as CMS (usually PHP) – also needs a secure protocol to encrypt account information and session
Website front-end	Webpages	HTML5/CSS + JS or Python + Django or Flask (more difficult, requires routing, Flask is often easier)
Website back-end	Querying databases for items	Same as CMS
Content Management System (CMS)	Modify items (price, description, name/title), view transaction history, profit	Anything that can perform REST methods (get, post, put, delete) – PHP most common, or Python if using Flask
Shopping Cart and Delivery/Pickup Time Estimate (ETA)	Perform user purchase transactions. Calculate estimated prep time for food before pickup is ready (plus driving time for delivery)	Databases (food prep time, user address) + Google Maps API for travel time
Commission Script	Check commission table to see if money has met a threshold – if so, inform cart to split payment to developer, subtract from commission until 0, then tell cart to stop splitting and subtracting. Send transaction to restaurant client program for processing	Any language that interfaces with cart system (JS, Python)
Restaurant Client Program	Program to notify restaurant when a new transaction appears (like TCP message)	Any good software language (C#/Java)
Mobile app	Optional – same features as website. Divide into front-end and back-end if desired	Either wrap website or use C# -- compiles to iOS and android with Xamarin and MAUI

* See toast (toasttab.com) and EatStreet (eatstreet.com) for website examples.

Structure:

- Website
    - **Home**
        - Display restaurant info (type of products, photos, etc.), contact info (bottom of page), direct user to correct page, promos/deals, advertising
    - **Register/Login**
        - Push/Pull in SQL table
        - Customers (and guest users) are given the role "customer", developer gives the owner "administrator" role (can modify other user roles)
    - **Menu**
        - EatStreet-like item hierarchy? Sandwiches, drinks, desserts, etc.
        - Allow users to add items to cart – OPTIONAL modify ingredients of item (no X, extra Y, etc.)
        - Admin Content Management System (CMS) – edit items (pencil click to edit item name, description, price, ingredients, update preparation time, etc.)
            - OPTIONAL – display "slash price" alongside normal price for items on sale (ability to add coupons/deals)
    - **Shopping Cart**
        - Update item quantities, remove items, clear cart, checkout (as user or guest)
        - Save credit/debit card to account or remove card (OPTIONAL)
        - Pay with [Toast, Cake, Square, etc.] – enter delivery/pickup details, perform transaction, push transaction to transaction and admin tables
            - Use Luhn Algorithm to verify if card numbers are valid (in real-time)
            - If successful, add transaction to transaction table and sale price to commission table (incoming_amount) and run commission script (check if incoming_amount equals threshold, if so, set outgoing_amount = incoming_amount, reset incoming_amount, and subtract percentage of sale from outgoing_amount to pay developer, pay the rest to restaurant – otherwise pay all to restaurant), send to **Receipt** page
                - Show option to print, delivery/pickup ETA [using sum of preparation time of each item + distance travel time for delivery], pickup/delivery information, add to user's purchase history if logged in. Otherwise, return user to cart if the transaction is unsuccessful
    - **Admin Page**
        - (FOR ADMIN ROLE ONLY) View transaction history, profit, modify user roles, modify home page (display promos/deals, add advertisements, etc.)
- Database System
    - **Account table**
        - Stores all information related to user account
        - Possible structure:
            - index, role (CUSTOMER/ADMIN), email, phone, first_name, last_name, password, address [street, city, state, zip], billing address (SAME_AS_ADDRESS) OR [street, city, state, zip], purchase_history [[product, modifications, quantity, price, date, time, transaction_id]], saved_card_info [[number, exp_date, cvv]]

- **Transaction table**
    - Stores all transactions and related information
    - Possible structure:
        - index, transaction_id, username (OR GUEST), revenue, items_sold {item [ingredients], cost, quantity}, eta, date, time, completed (OPTIONAL, true/false)
- **Commission table**
    - Stores ongoing profit until a threshold is reached
        - When incoming_amount hits a certain threshold, add to outgoing_amount and set to 0
    - Possible structure:
        - index, incoming_amount, outgoing_amount, total_amt_paid
- **Item table:**
    - Stores all information related to the restaurant products
    - Possible structure:
        - index, name, price, description, (OPTIONAL) image, (OPTIONAL) ingredients, preparation_time (IN MINUTES, 5 BY DEFAULT)

- Serving System
    - Hosting Server
        - Stores and hosts web pages and database systems
        - Runs commission script for each transaction
    - Commission Script
        - Automatically ran during each transaction, check commission table, send transactions to restaurant client program to notify restaurant of a new order. Update total_amt_paid each time a commission is subtracted from outgoing_amount (add commission amount to total, subtract from outgoing)
    - Restaurant Client Program
        - Acts as a server waiting for the commission script to notify of a new transaction, display transaction information for owner (from transaction table) to prepare
        - OPTIONAL – Make “order ready for pickup” button for restaurant to message server to update delivery ETA to ready – Set “completed” to true; receipt page will have to automatically refresh every ~60 seconds to parse transaction from table to see if the order is completed yet or not

Process:

1. Decide on implementation technologies (starting with server/host and database management system) and delegate initial tasks. Start working on components in order on components table
2. Databases should be made before anything to allow for dummy scripting to test SQL queries
3. Login/registration system can be made in any language for proof-of-concept, then translate into language for website (usually the same as CMS, typically PHP)
4. Develop skeleton of website – create a file for each page, and a separate file for the header and footer (these can be done dynamically, especially if all files are PHP or Python)
5. Fill databases with dummy information to test item queries and flow results onto page, make sure each feature from the table goes to the correct page element
6. Start working on cart system (adding items to cart, modifying items, removing, updating quantity) – start with only allowing users with an account, then allow guest transactions after accounts work. Modified ingredients should be stored in an array to be displayed to the restaurant (ADD X, REMOVE Y) or as a dictionary {X: REMOVE, Y: ADD)
7. Simultaneously start working on CMS – all developers should have admin role, so no one has to directly modify the SQL tables
8. Make a mockup transaction system and concurrently develop commission script and restaurant client program. Use Luhn Algorithm to validate card numbers, but don't perform transaction on a real card – (temporarily) randomly fail transactions to make sure that works, then remove
9. Send successful transactions to database and call commission script, make sure commission script can modify the commission table and message the restaurant client program
10. Develop delivery ETA system for receipt page, make a mockup payment script that prints the amount paid to the developer ($0 unless at threshold) and restaurant for each transaction
11. Develop admin page and home page simultaneously
12. Furnish the designs of the website and restaurant client program, add the ability for guest transactions
13. If an actual client is found, wipe the databases, and get product information from restaurant to populate item table. Make a developer account and owner account with role ADMIN so no dummy accounts are leftover or necessary
14. Perform a few dummy transactions to ensure all components are working successfully. If so, decide on a transaction system (Square, PayPal, Stripe, etc.) and implement restaurant and developer payment systems (use commission script to help)
15. Optionally, design mobile app using the same components as the website – either using HTML5 or a web wrapper, or a software language like C# (most backend scripts will be easy to re-write, so delegating development to front-end and back-end will be necessary since the front-end will likely take more time)