Los Pollos Hermanos

Fast Food Point of Sales (POS) System

Professor Damavandi - CS 3560 Section 2

Kyle Just
Computer Science Department
Cal Poly Pomona
Pomona, California
kjust@cpp.edu

Russell Rickards
Computer Science Department
Cal Poly Pomona
Pomona, California
rdrickards@cpp.edu

Joshua Jenkins
Computer Science Department
Cal Poly Pomona
Pomona, California
jejenkins@cpp.edu

Taft Ring
Computer Science Department
Cal Poly Pomona
Pomona, California
tcring@cpp.edu

Abdul Kalam Syed
Computer Science Department
Cal Poly Pomona
Pomona, California
syed@cpp.edu

***Abstract*— For this project, we implemented an upgraded drive-through POS system for a fast-food restaurant named Los Pollos Hermanos. This system can create orders upon request, edit orders, delete orders, display orders to the chef, update order status, and manage existing orders. To create this system we used Python, Qt Designer, and Azure SQL. Throughout this paper, we will discuss the thought process, design work, and implementation of our project. We will also show the various diagrams used to construct and connect our classes.**

***Keywords*— Customer, Cook, Manager, Database, GUI**

I. Introduction

A drive-through POS system can be used to create orders, edit existing orders, update existing orders, display active orders, and manage previous orders in a more effective and efficient manner compared to the pen-and-paper method. This system will increase productivity and efficiency while also providing a secure database for managers to analyse their day-to-day operations.

A. Problem Description

Los Pollos Hermanos started out as a mom-and-pop restaurant that uses pen and paper to take orders. This system is advantageous for setting up a business because of the low initial cost. However, this system does not adapt well to modern fast-food restaurants. Customers are impatient, hungry, and can very easily become dissatisfied. When taking an order through a drive through, customers must speak through a microphone system which is played over a headset to the cashier. These systems are usually incredibly low quality and can lead to confusion between cashier and customer. This confusion exacerbates the chances of an error when recording an order with pen and paper. Additionally, it is much more likely for a mistake to occur when reading handwriting, especially if employees are writing quickly to process orders quickly. Order slips can also be physically lost in the chaos of a lunch or dinner rush. For a drive-through, an error is fatal. Orders must be served in the order that they are received because the cars are locked in position one after another. If an order is incorrect at the order window, the entire line must wait for that one car's order to be resolved. Therefore, for a business such as Los Pollos Hermanos that is looking to implement a drive-

through, a digital POS system is a necessary acquisition.

B. Proposed Solution

Our software reduced the frequency and severity of these errors by introducing a digital POS. This POS allows for orders to be taken quickly and accurately. Instead of cashiers having to remember what customers are saying through the microphone as they write down the order, they can instead quickly press the buttons on the screen. This allows cashiers to keep up with whatever speed the customer speaks at, reducing the number of times that a cashier must ask the customer to repeat themselves. The order will also be much easier for kitchen staff to read, with absolutely no confusion as to what each specific order item is. Additionally, the POS does not require that the cashier memorise the menu, nor a specific format for writing to avoid confusion with the kitchen staff. The POS cannot lose orders, further reducing the chances of an error in the drive through line. The POS also enables live management of orders that have already been recorded, something that can be particularly challenging with paper tickets. This system improves on every aspect of taking orders from customers and will enable Los Pollos Hermanos to perform better during periods of high traffic.

II. Analysis

A. System Requirements

When analysing the current drive-through system, there were three major users that needed to interact with it. The first is the cashier. The cashier will need to create a new order for every customer coming in. To do this a cashier should be able to create a new ID for the order, select the items the customer wants, increase or decrease the quantity per item, and add any notes to the item upon the customer's request. The cashier should also be able to have an overview of the entire order including the total price, and the option to continue to edit the order if needed. Finally, the cashier should be able to submit the finished order. The next user of the POS system is the cook. The cook should be able to access and view all the active orders that the cashier submits. This view includes the Items per order, the quantity per item, and any notes attached to the item. Once the cook is done, they prepare the order, they should be able to update the status of the order. Then the view should continue to show all active orders until there are none left. The final user of the drive-through POS system is the manager. The manager will have the same rights and abilities as the cashier. The only addition is the ability to edit menu items. The manager can add new menu items, delete existing menu items, edit the price of menu items, edit the description of menu items, and the name of the menu item. On top of all the abilities each user can do, the system should be able to self-update and record all orders.

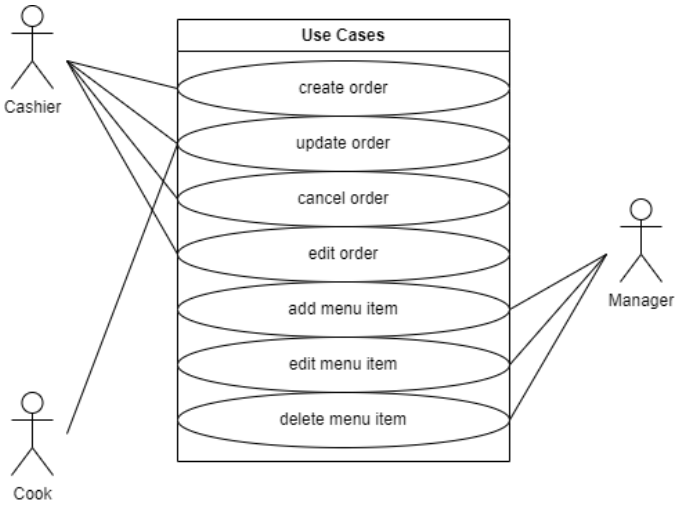
B. Actors and Stakeholders

While examining the idea of this system, we realized that there would really be three actors and three types of stakeholders. The first actor would be the cashier. This person's duty is to create the instance of the order, add items to it as the customer pleases, and then, when finished, submit to the database for the order to continue throughout the rest of the process. The second actor would be the cook. Here, the order is passed onto a screen that only the cook sees. This screen shows three orders. On each order there are order items, any notes for the items and the order position. After the cook finishes preparing the order, they would mark it prepared by clicking "done" on their screen. The third and last actor would be the manager. The manager would have a separate application on their computer that would have administrative access to items on the menu. In this specific application, they could add, edit, and delete menu items. As for the three types of stakeholders, there would be operational, management, and, of course, the customers. Operational stakeholders are the stakeholders who directly use the system and do not have administrative access. The cashier and cook would all fall under this role. The customer stakeholder is self-explanatory. Any consumer of Los Pollos Hermanos that indirectly uses the point of sales system (put simply, a customer that orders through the drive-thru). The management stakeholders would be the manager. This is, because the manager would be the one in charge of the operation and have administrative power.

C. System Analysis

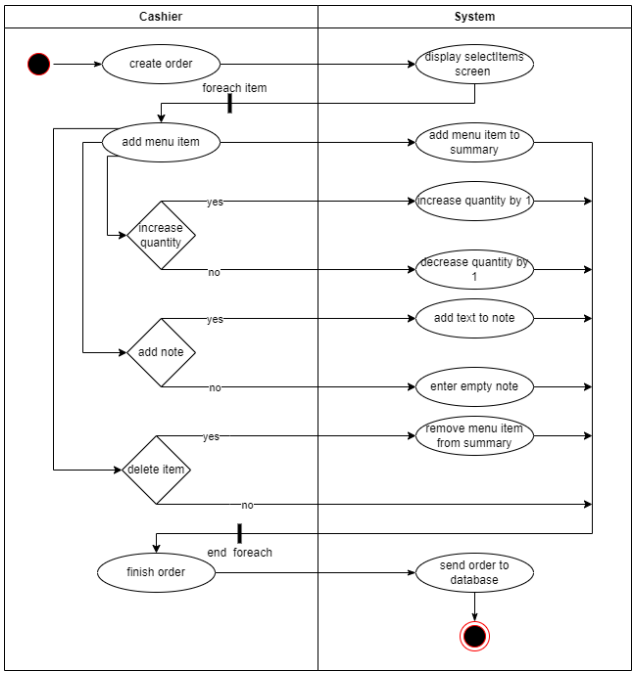
System Analysis is a crucial step of any system design. The programmers must understand the scope of their program, and what functionalities their system must have. We started by envisioning a fast-food worker servicing a drive through during a lunch rush. What actions would they need to perform? What are the possible requests from a customer that the system needs to be able to handle? What capabilities should this worker have to improve the flow of operations within the business? These are the kinds of questions that we posed to each other as we designed our system. The result was a fully flushed out system that meets the demands of a fast-food POS, within the capabilities of a CS3560 student.

D. Use Cases



Above is our use case diagram that displays which actors in our system will interact with which classes or functionalities of the program. The cashier is the only one that interacts with the create order class, which is the first point of contact once the customer comes in to order. The cashier can also update, cancel, and edit the order after submitting it to the database if the customer wants to change the order. They would do this by going to the manage order screens and making the changes. The cook can also update the order because they can mark active orders as prepared. The manager can interact with the manager UI, which is separate from the UIs for cashier and cook, this includes add menu item, edit menu item, and delete menu item use cases. A manager can add new items to the menu, and edit or delete existing items, like changing the price, name, and description of the menu item.

E. "Create Order" Activity Flow Diagram



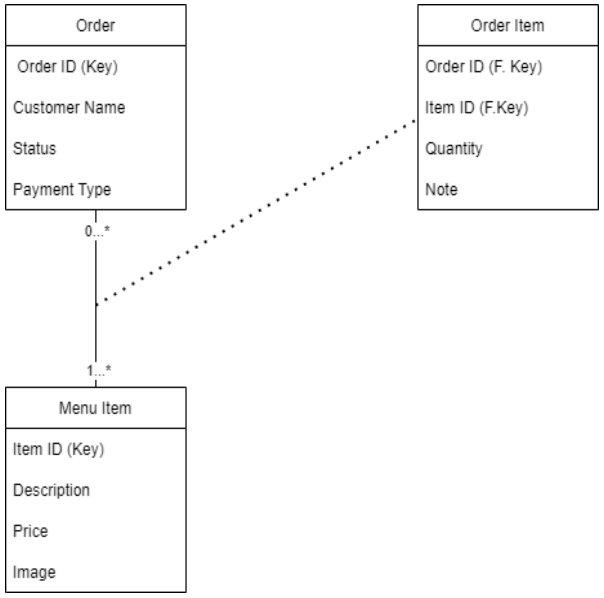
Above is the "create order" activity flow diagram. This diagram shows the flow of the system when the cashier part takes in the "create order" use case. To elaborate the system flow of creating an order, first the cashier creates an instance of an order by clicking the create order button. This then opens a selection screen where the user is shown all the menu items able to be added to the order. Here, the cashier adds as many menu items as the customer wants -- which explains the foreach loop. For each menu item, the cashier can increase or decrease the quantity, add a note, or remove it off the order entirely. At the end of the order, the cashier finishes the order. Here, the cashier enters the name of the individual and submits the order to the database.

III. Design

The use of a design model is to work out all the details and minor adjustments we want to implement before we begin coding the project. Properly implementing the design models will make the

overall project go by smoother since we have already considered all outcomes and capabilities of the project.

A. Domain Class Diagram



For the domain class diagram, we have three classes. The first class is order, its attributes are Order ID which is the key, customer name, and the status of the order. During implementation, we decided to leave the payment type out since it is not really related to our project. The next class we have is the Menu Item. Its attributes include the Item ID which is the key, a description, a price, and an image. W then has an association class called Order Item which uses the Order ID and Item ID as foreign keys. This class also includes the quantity per item and the notes.

B. Graphical User Interface

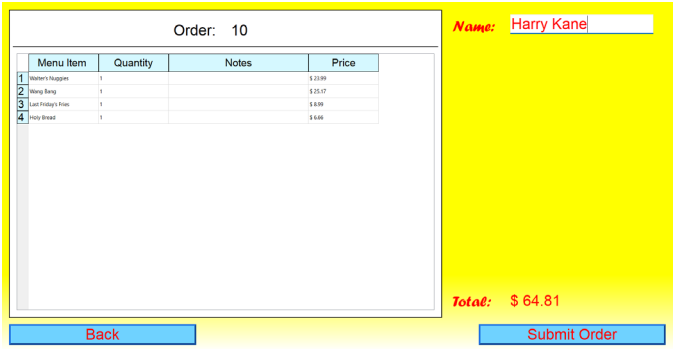
The graphical user interface is the interface that the users will see when they run the application. It is important to build an intuitive GUI, or else users will not be able to make full use of the backend.

Our welcome screen has a clean welcome message and allows the user to immediately begin working by managing a current order or creating a new order.

This display opens when the user selects “Create Order” from the home screen. Cancel order closes the window and goes back to the welcome screen. Each image as well as each title of every item is a clickable button that will add an order item to the order summary. (see below)

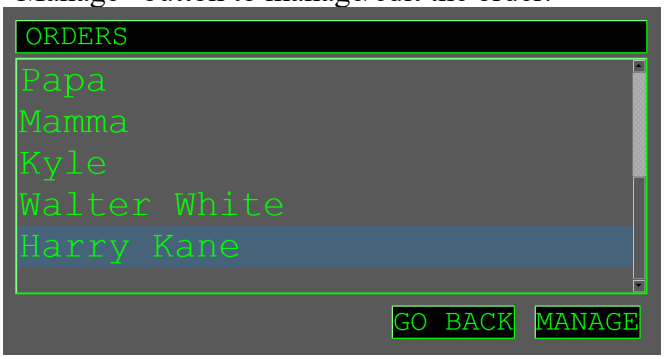
Once the order is added to the order summary, the cashier can change the quantity, which initially comes up as one, by using the up and down arrows or by directly entering the desired quantity in the section. They can add any notes, and they can remove the item from the order if the customer no longer wants it. Once the customer is satisfied with

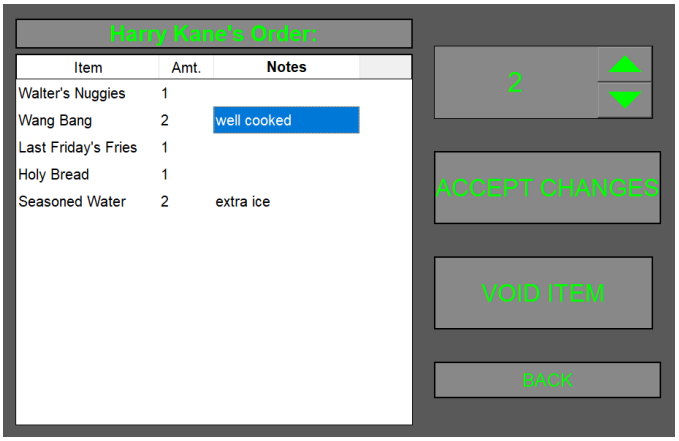
the order, the cashier will click the finish order button, which will bring up a new window that will display the price of each item and the total price, as well as, have an input text field to enter the customer's name.



This window has two buttons, the “Back” button and the “Submit Order” button. The Back button will take the cashier back to the create order screen, which will have all the current items already listed and the cashier can make any changes that the customer wants. The Submit Order button will submit the order to the database and close the current instance of creating an order and take the cashier back to the welcome screen.

After submitting an order to the database, the cashier can still update or edit an order by clicking the “Manage Order” button on the welcome screen which will prompt up a new screen (which you can see on the right side of the page). In the manage order window, the cashier can see all the active orders in the database by the name of the customers, they can select a name and click the “Manage” button to manage/edit the order.





Once the cashier clicks the manage button, the above screen shows up. The cashier has the option to change the quantity of an item, add/edit notes for an item, and void an item, which can be done by selecting an item and clicking the “Void Item” button. Once the customer is satisfied, the cashier can click the “Accept Changes” button which will update the order in the database. The cashier can click the back button to go back to the manage order window to select any other orders or click the back button on the manage order button to go to the welcome screen.

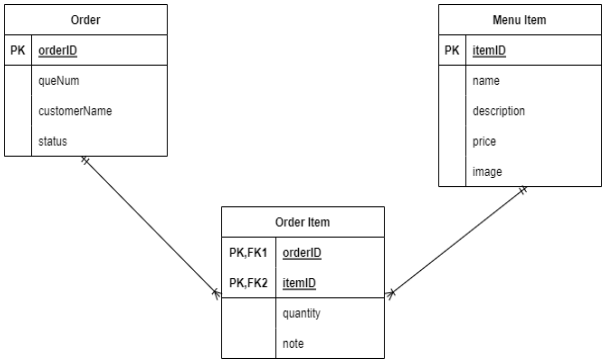
This display is the cook GUI. In this GUI, we display all the active orders in the database. The oldest active order displays on the far right and the newer active orders come in from the left. Once the cook is done making an order, they can click done which marks the order as complete and deletes it from the cook GUI. The orders then shift over.

C. Development Model

Our development of this software followed a waterfall-like development cycle, with distinct design, development, testing and release phases. The project itself was broken up into key elements in the design phase: the database controller class and each of the graphical interfaces that were used in the system. The database served as the spine that much of the rest of the development extended from. Often the most progress that we made was in our bi-weekly extreme programming meetings in which we all sat in the same room to debug and collaborate on bugs and solutions for an evening during the development and testing phases.

D. *Database Schema*

The database used MySQL for our relational database management system. The MySQL server was hosted with Microsoft's Azure Cloud services. Our database schema included all of the attributes of our domain class model. The orders and menu items had an order id and menu id as their respective primary keys, while the order items table had 2 foreign keys from the two associated classes as its primary keys. I put no "ON" keywords relating the tables together, instead opting to control the deletion of order items to orders with the database controller rather than the database itself.



Database Schema

IV. Findings and Suggestion

A. *Findings*

We found that our software effectively addressed all use cases that we identified in our design analysis. We eliminated use cases that would be managed by other systems, such as payment and inventory. Throughout the project, we discovered many challenges that we had to come together to solve. What we expected to be a "simple" program ended up being much more involved than we thought.

We discovered difficulties with the software that we were using. Namely, QT designer was an entirely new software for us and finding out which widgets to use to represent our ideas took quite some time. We also had to reference QT documentation to learn how to manipulate the widgets with code.

We now understand that we need to expect problems and errors, even with the simplest tasks. Sometimes, simple errors can take the longest to resolve. If we had committed ourselves to working on it earlier, we would have avoided many headaches and long nights.

B. *Future Development*

For future development, we think that an online portal that allows customers to place orders remotely would be a useful tool for a restaurant. Additionally, promotions and other advertising could be run on that website, drawing more customers to the restaurant. Time did not allow for a web portal in our case.

A fast-food restaurant would also need to have inventory and payment systems integrated with their POS to work securely and efficiently. Credit payment would need to be validated before issuing an order to reduce the chance that the restaurant is taken advantage of by fraudsters. The restaurant should also manage its inventory so that it does not run out of food mid service. Our current system does not accomplish these tasks, but we feel that our design would for integration with these systems.

While QT designer served its purpose and we were able to build a unique and aesthetically pleasing UI, we found the software to be clunky and restrictive. In the future, a new GUI builder would have allowed us to focus more time on the code rather than GUI design.

Finally, we would have liked to implement user credentials so that we could authenticate a manager before they use administrative functions. Currently, we have that program running independently, as it would only be accessed on a manager's computer and not the main POS. However, credentials are a standard form of added

security that would have been added to our system.

V. Conclusion

Overall, we are happy with the end game and what we have produced. We used this project to learn plenty of new skill that will help us in future projects. We learned to create a Graphical User Interface and how to implement them using Python and PyQt5. We also learned how to create a database, as well as, linking the database to and application and moving that data to a front-end interface, which proved to be a remarkably interesting part of the project. Additionally, we learned the common workflow within a GitHub repository. Some of us had experience with the platform beforehand, however, this was the first time many of us had used it so extensively, and it made our work simple to collaborate on and construct.

VI. Recap

To revisit, the reason for this project was to solve the inefficiencies of the rudimentary pen and paper order taking method. Our software addresses these inefficiencies with features that allow the quick creation and editing of orders, as well as the capability to update orders and menu items. With the upgrade to our POS, Los Pollos Hermanos can become a thriving business.

References

[1] *Python.org*, 2001. [Online]. Available: http://www.python.org/. [Accessed: 09-Dec-2022].

[2] "Cross-platform software design and Development Tools," *Qt*. [Online]. Available: https://www.qt.io/. [Accessed: 09-Dec-2022].

[3] "GitHub," *GitHub*. [Online]. Available: https://github.com/. [Accessed: 09-Dec-2022].