Whole Foods Ordering System

Christian Taro Magpantay
Jerome Fulgado Jr.
Christian Alcalde

Introduction

Scope/Purpose

The purpose of this documentation is to provide the business and technical overview on the Whole Foods Ordering System. This application will give users easier and efficient control to order Whole Foods items from their local grocery store. Stores will provide information on what items are in stock and when a user's order is ready for pick up.

Definitions, acronyms

Term	Definition
Client/User	A person that uses the ordering system application to order products or groceries.
Command Line	Text based user interface that allows a client to interact with the application.
Database	Collection of data that manages information necessary for the application to function correctly. Some of this data includes user information, inventory, payment information, etc.
Foreign Key	In the context of databases, it is a field in a table(s) that refers to a primary key.
GUI	*Graphical User Interface.* Visual interface that allows users to interact with the application. From here, users can easily place orders and know when their orders are ready for pick up.
UML	*Unified Modeling Language.* A modeling language used to organize the project's code in an efficient manner and provide a visual representation of it.
Whole Foods	USA based supermarket chain with a focus on "healthier" food products.
Zone Controller	Software that allows users to interact indirectly with a store. Also known as the "infrastructure layer".

References

UML Design using draw.io
Images from a white board
Roles and Responsibilities

Overview

The Whole Foods Ordering System, is designed for users to select various products/groceries from any nearby Whole Foods store for pickup at a specified time. This application saves users time from having to search the store and waiting in line to pay for their goods. It will implement an account system that saves a user's information such as their name, address, payment information, etc. to speed up the check out process. Users will also be able to see nearby Whole Foods' inventory.

Overall Description

The goal of this product is to give users easier access to order whole foods items and pick up items without waiting in line. This application will ensure ease of use for those who need their items right away. This document will include all required items needed to successfully implement the application.

Product perspective

The Whole Foods Ordering System is an application that provides the functionality found in the User Requirements section. This application aims to fulfil the software requirements by implementing a GUI for the user, zone controller to help users interact with the stores, and several databases that will hold information that lets the application function properly. In addition to this, the built in account system gives users the luxury of saving their credentials and payment information to speed up the check out process. If they choose to register an account, users will also have the option to see their order history.

Product functions

The client is able to interact indirectly with the zone controller that feeds information to the client or user of the groceries or items that are needed for the user. The user is interacting with the Zone Controller which also interacts with all the stores that the user enters for a pickup location of their choice. The user is also allowed to pick another store if their preferred store does not have all the items in stock, desired by the user. Extra features would include but not limited to pre-ordering special items or regular items and delivery.

User characteristics

The client, if they wish to create an account, will be able to store all of the information such as credit card or preferred items that will be saved on their account. This also allows for speedier checkout and possibly add the characteristics of allowing the user to choose different cards or try different items that may be similar to their purchase history or have items that are on sale that can be recommended to them. The application is built around the user and how they use the application and creating an account that saves their history. The user is also allowed to perform a guest checkout.

Specific requirements

User Requirements

Here are the user requirements for the Whole Foods application:

- Create an easy to use GUI or Command Line UI with the following menu items
  - Order
    - Search/Select an item from a preferred store around them. They could add how much quantity they need for that item. Each item after adding will be added to their card. They could also deselect items and remove them from their cart as well. If an item is not available, prompt the user with “Item Not Available” for the correct store.
  - Review Cart
    - Print the users a list of all their items added to their cart.
  - Checkout
    - Go through the payment process. BEFORE PAYMENT, users are prompted to perform a ‘guest’ checkout or a user checkout. ‘Guest’ checkouts will require users to input their name, email address, and payment method. ‘Users’ will have a defined name, email, and payment method saved for future use. User info is tied to store accounts they create. Before confirming payment, users will be prompted with their billing information, payment methods, cart, and pick up time. After purchase, the cart should be emptied and a message is returned back to them saying that the order is complete.
  - Order History
    - ONLY USERS WILL HAVE ACCESS TO THIS FEATURE.This will keep track of all completed orders for a user. Users will be able to pull up their payment method and store information on their order receipt.
  - Exit Application
    - Exits the application.
- Outside of this menu, there will be an initial load of the food database and the store database inventory when the application starts. The initial load will connect to the

databases from the infrastructure layer. Check the system requirements below about the layers.

- Allow for user error check if an option is not valid.

System Architecture

To develop this application, we will need to have a “controller” or infrastructure layer to have the stores and the UI talk to each other. Each store will include a database that stores their whole food inventory. There will also be a central database, for the application, that will host users, order receipts, deals, and guest users (for temporary records). For this application, it is recommended to be familiar with database management; how to create, edit, and delete databases. It is also recommended to learn how to connect the database to the infrastructure layer. Refer to the system requirements below for detailed information.

System Requirements

Here are some of the system requirements that need to make this application function:

- Create a central database for Whole foods application
  - This includes the following tables: Users, Food Items, Guests (temporary records), Order Receipts, Payments, Deals, Stores
  - Users will have the following columns: id, username, password, email, mobile number
  - Food Items will include the following columns: id, name, calories
  - Guests will keep temporary records in for a certain amount of time to maintain table data. The columns are the same as the User table
  - Order Receipts will have the following columns: id, confirmation number, date of purchase, username (foreign key), payment (foreign key), store (foreign key)
  - Payments will have the following columns: id, card name, cvv, expiration date, username (foreign key)
  - Deals will have the following columns: id, code name, price deduction, store (foreign key)
  - Store will have the following columns: id, store id, address, zip code, est. date
- Create databases for 3 “test” stores. Each store will have their own individual database of items. The database will have the following tables: Food Items, Order Receipts, Users.
  - Food Items will have the following columns: id, name, quantity
  - Order Receipts will have the following columns: id, confirmation number, user (this will foreign key with the user table), store id
  - User’s will have the following columns: id, user (this will foreign key from the User table from the central database)
- Infrastructure Layer to have the UI and the Stores communicate

    - This “layer” will consist of methods and functions to retrieve/send data from the user to the store and vice versa. This will also retrieve/send data from the central application database.
    - This will be an essential piece for the store to also talk to the application database for food items that need to be added to each store and users to associate themselves with a desired store.
    - Here are some of the methods to create for the controller
        - Send data from the user to the store, essentially a user input of an order confirmation or when an item is added to their cart from one of the stores
        - Send data from the store to the user. This method will display to users what items are available from their location or from the user’s desired store.
        - Give stores/users what items are available for sale, meaning a method to connect the central database to the infrastructure layer and having the user/store retrieve the data from the central database.
        - Send data from the infrastructure layer to the database. When a user/store picks a command to send data to the database, it will send the data to the correct table in the database. An example of this is having the store confirm that the order was successful and sending a record to audit Order Receipts.
- User Interface Layer
    - GUI will be required for users to view the products available at their preferred Whole Foods and to interact with the application.
    - Allows users to login/create an account to save their information to speed up the checkout process in the future. This information is then saved into a “Users” table for the central database and the store database. Alternatively, users can opt for a guest check out.
        - Users will create an account by providing an email and password, consisting of alphanumeric strings from 5 to 15 characters. After creation, users can choose to store their home address, email, phone number, and payment methods if they desire.
    - Users will be able to set their preferred store and see available stock according to that particular store’s food database. If a desired item is unavailable at their store, the app will find the nearest Whole Foods with that item available. These actions are handled by the zone controller, previously explained above.
    - After choosing the desired items, users can set a quantity of each item (as long as there is enough inventory at the store) they would like to order and add it to their cart for check out.
    - Once all items have been selected and added to a user’s cart, they can now move on to check out their items.
    - A check is performed to see if a user is logged into an account

        - If logged in, zone controller pulls user info from the appropriate database, or if none is found, user is prompted to enter their credentials to be saved for future use.
        - If not logged in, user is prompted with one of three options: create an account, log in, or guest checkout (temporary records).
    - Users are then prompted to select (if saved) or input their payment information.
        - Users will select the credit card they have saved on their account. As an extra security measure, users are required to enter the CVV number to proceed with payment. This number will be sent to the zone controller and verified with the database. The option to add another card is there too.
        - If users have a coupon, they can enter it here. This coupon will be sent to the store's coupon database for verification.
    - Pick up time is then selected. Users will have three pick up options:
        - Earliest time possible
        - Within several hours
        - Next day/future date
    - They will then be brought to a final review page to confirm their order:
        - Cart
        - Credentials
        - Payment method
    - After confirming, users will be given an order number and an estimated pick up time.
    - All information is received by the zone controller and passed to the stores.
    - When a customer picks up their order, the store marks their order as "received" and their order is placed into an archive database.

Legal, Copyright and Other Notices

This product is mock up but will simulate as closely as possible to its planned construction. This product uses are by private entities including the public enterprise and data are only to be stored by the private entities and the public enterprise. This product uses the Java SE platform and other applications used to support this product. All entities using, downloading, installing, copying, or accessing agree to these rules and by doing so must abide by these rules.

System Evolution

As time goes by, our software will slowly start to become outdated and in need for updates to stay relevant and reliable for users. If bugs or errors are found while using the application normally, updates will be issued. Incremental maintenance updates will be performed on the app to avoid any breaches of user data whether that be their personal or payment information. We will also be releasing updates to ensure that the application works on the latest operating systems. This also includes extending compatibility of the software to other devices such as tablets, personal computers, web interface, etc. As users and inventory increases, maintenance will also be performed to expand database storage if needed.

Supporting info.

Extra features that may or may not have been mentioned about would include having a pre-order feature for clients and having the delivery aspect implemented so that users do not need to pick up their items. Users would be able to check on their deliveries using a tracking number and see how far the driver is from a map using a Google API.

To stay competitive with similar order system applications, we want to implement several new features in the future. Depending on the time of year, an algorithm to prepare inventory can be created to predict certain groceries only available during a specific season (ex: asparagus in the Spring, mangoes in the summer). This algorithm would take buying history into account as well. A machine learning algorithm would also be created to suggest popular groceries that a user may like based on their shopping history and interests.