Magic Menu

Software Design Document

Nathan Jordan, Brandon Whitesides

03/27/2025

Intro

The Magic Menu project that we will be working on will be similar to the Dessert Shop project we've completed in class. This new project will have similar features as the Dessert Shop project, but also new ones. Instead of being for the customer's use, our project will be what the restaurant owner can use to organize everything neatly. It will use the topics we've learned in class to help restaurant owners and the customers they serve.

This project will be of interest to restaurant owners because of the flexibility that it gives them, and the convenience that it gives their customers. Many restaurants rely on physical menus, requiring a reprint of the menu to update prices. With our software, the restaurant owner will be able to easily update their menu and effectively advertise items with a high profit margin, and the customer will be able to easily compare items to come to a decision with ease.

System Architecture

Our program will implement CRUD operations (Create, Read, Update, Delete), and will have a simple GUI to see items on the menu, add items, update info about the items, and delete items from the menu. In this implementation, the restaurant owner will be able to manage the items on their menu easily via a TKINTER-based application. The items will be sent via JSON to a website that the customer will use as a menu, though we don't plan for that to be a part of this build.

Modules

1. main.py
    a. Calls the necessary functions from the other modules to bring the program together
2. gui.py
    a. Constructs the GUI that the restaurant owner uses to create, read, update, and delete items on their menu. Data is received from the process_items.py file
3. models.py
    a. Convert the feet to inches
    b. Add the user inputted inches to feet conversion to reach total inches
    c. Inches to hamster calculation
4. process_items.py
    a. Reads the items.csv file, and returns a dict object with {menu_item.name : menu_item object} that will be used to create the menu in the gui.py file
5. create_menu_item.py
    a. Adds an item to the items.csv file
6. update_menu_item.py
    a. Updates an attribute of the menu item in the items.csv file
7. delete_menu_item.py
    a. Deletes an item from the items.csv file

Classes & Methods

MODELS.PY

1. MenuItem (Base Class)
    a. Base class for all menu items (drinks, desserts, entrees)
    b. Attributes:
        i. name: Item name
        ii. price: Item price
        iii. description: Item description
        iv. image_path: Path to item's image
        v. Calories: Caloric Content
2. Drink (Inherits from MenuItem)
    a. Represents beverage items
    b. Attributes:
        i. size: Drink size
        ii. Inherits all attributes from MenuItem
3. Dessert (Inherits from MenuItem)
    a. Represents dessert items
    b. Attributes:
        i. Inherits all attributes from MenuItem
4. Entree (Inherits from MenuItem)
    a. Represents main course items
    b. Additional Attributes:
        i. Inherits all attributes from MenuItem
5. Appetizer (Inherits from MenuItem)
    a. Represents main course items
    b. Additional Attributes:
        i. Inherits all attributes from MenuItem
6. Order
    a. Represents a customer's order
    b. Attributes:
        i. items: List of ordered items
        ii. drink: Stores drink selection
        iii. dessert: Stores dessert selection
        iv. entree: Stores entree selection
        v. Appetizer: Stores appetizer selection
        vi. total: Total cost of the order
    c. Methods:
        i. add_item(item): Adds an item to the order and updates the total price
7. RestaurantApp (from gui.py)
    a. Main GUI application class
    b. Attributes:
        c. root: Main window of the application
        d. menu_items: Dictionary of menu items loaded from CSV

Data Storage

- Items.CSV: storage of the menu items.

Error Handling & Edge Cases to consider in testing

- Input validation to prevent invalid inputs for attributes (negative cost, negative calories, too many characters in the name, ect)

GUI

Because we are using Tkinter for our GUI, we will focus on overall composition in this mockup of the different screens. For that reason, we will use a table. This will help in our programming, because Tkinter relies heavily on gridview.

Main Screen				
MENU ITEMS		Click on item to see/edit info		
Appetizers	Appetizer1	Appetizer2	…	Add Appetizer
Entrees	Entree1	Entree2	…	Add Entree
Drinks	Drink1	Drink2	…	Add Drink
Desserts	Dessert1	Dessert2	…	Add Dessert

Item Screen (after clicking on an item)		
Attributes	Item Name here	Click to Delete {item name}
Name	item.name attribute	Text entry: enter new name:
Price	item.price attribute	Text entry: enter new price:
Description	item.description attribute	Text entry: enter new description:
Image path	item.imagepath	Text entry: enter new image path:
Calories	item.calories attribute	Text entry: enter new calories:
Size (if drink)	item.size attribute	Text entry: enter new size: