Project Report: Cafe Order Management System

University: VIT Bhopal
Course: CSE1021 - Introduction to Problem Solving and Programming
Name: Aryan Arora
Reg No: 25BAS10008

1.
For my project I chose to build a Cafe Order Management System. I live on the VIT Bhopal campus. I have seen the canteens become crowded. The staff shout orders. Write orders, on scraps of paper. That is messy. I want to write a Python script that helps a cashier take orders quickly. The Python script will calculate the bill without the cashier doing math in the head. The Python script will also print a receipt.

2. Problem Statement

I was annoyed by the mess at the checkout counter. The mess, at the checkout counter was the issue I wanted to solve. I wanted to fix the mess at the checkout counter.

Math Mistakes: Math Mistakes appear when you try to add ₹85 + ₹120 + ₹50, in your head. The result can be wrong.
No Proof: I often see that there is no bill. Usually there is no record of when an order was placed.
Solution: I built a console app that does the math for you. I built the console app to keep the work organized.

3. Features

I have the input, from the user. I wrote the code. The code can do the following:
Shows the Menu: The menu shows a list of things you can buy such, as Coffee, Tea, Sandwich and more. The menu also shows the price of each thing, in Rupees.
Takes Orders: I built Takes Orders to be simple. You type the name of the item. Takes Orders is smart. If you type coffee or COFFEE Takes Orders still understands you.

Calculates Bills: Calculates Bills adds up the total as you go. You can watch the change.
Checkout: Checkout gives you an Order ID, for example #4921. Checkout puts a timestamp, on the receipt.

Manager Stats now includes an option.
The small option shows the price of items, on the menu.

4. How I Built It

I keep the project in a file. The file is called cafe_order_system.py.
Storing Data:
I used a dictionary, for the menu. A dictionary is the way to link an item name to a price.
I used a List. The List kept track of the customer order.

Modules:
datetime: To get the actual date and time for the receipt.
random: To generate random order numbers and pick random staff names.
array: I used the array to handle the prices. All the prices are numbers so the array is a way to store the prices.

5. Code Details

Using the Array Module
In class we learned that arrays are good, for data that's all the type. I did not just use a list, for prices. I changed the prices into a float array. Then I added the prices up.

```
# I turn the price list into an array (type 'd', for decimals)
prices_array = array.array('d', prices_list)
total = sum(prices_array)
```

Fixing User Input
I realized that people might type inputs. I used.strip().title() on whatever the user types. The.strip().title() removes spaces. The.strip().title() also fixes the capitalization so the.strip().title() output matches the menu.

6. Challenges & Learnings

Clearing the Order:
I had a bug. The bug made the next person see the items, in the cart after the previous person finished ordering. I fixed the bug by adding current_order.clear() after the checkout step.
Alignment: The receipt was annoying to make straight. The receipt kept looking crooked. I

played with the f-strings (, like :<15). The f-strings forced the prices to line up.
Real-time Data: Using datetime was cool because it makes the project feel like a real software used in a shop.