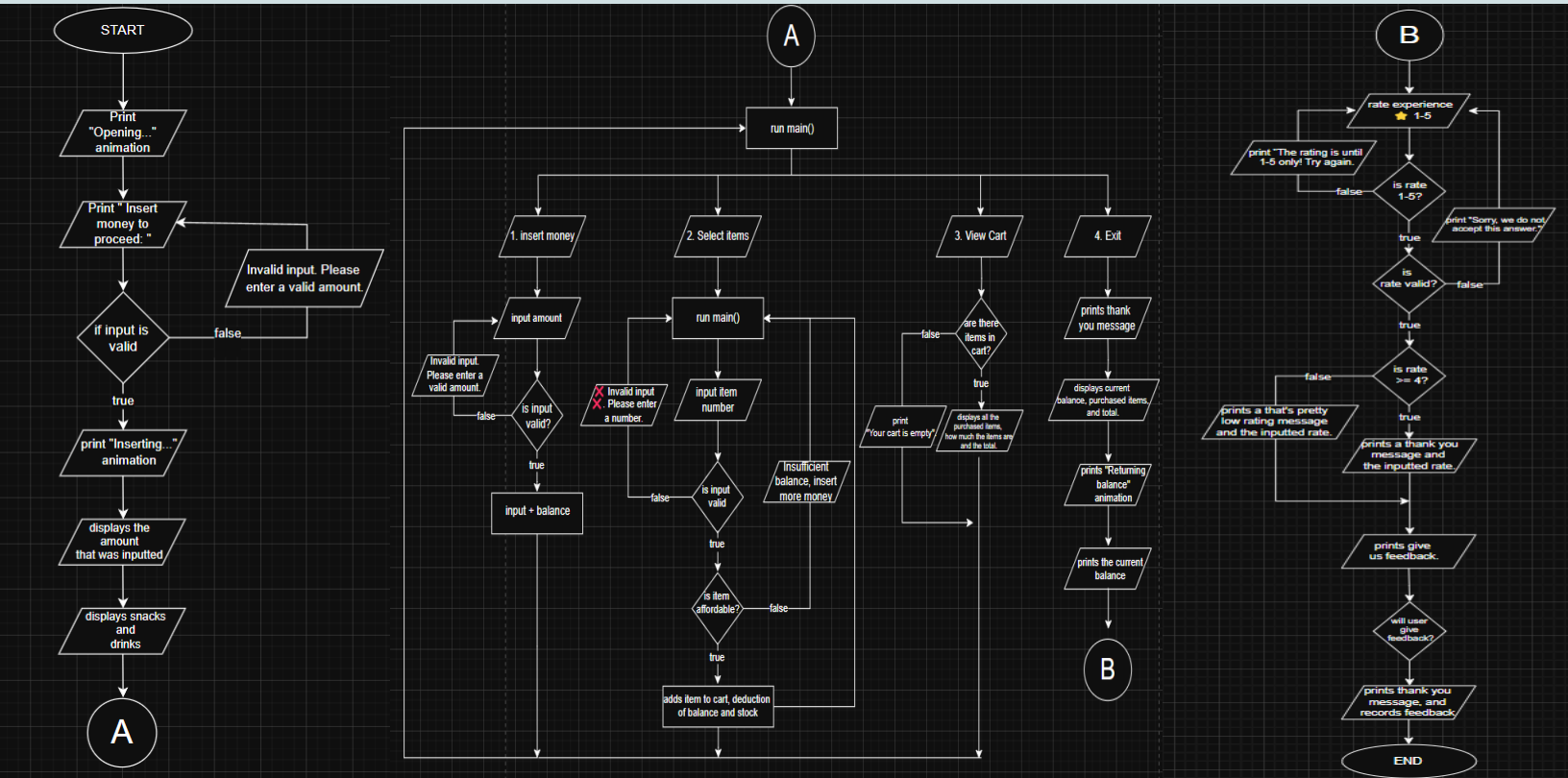
VENDING MACHINE DOCUMENTATION

Specification

As the first semester is about to end, in intro to programming, we were told to make a vending machine using the coding language python. We learnt how to use the Python language from our beloved teacher, Mr. Oliver. He began by teaching us the fundamentals of coding, and as the weeks passed by, the lessons also got more complicated which led us on making a vending machine on our own. My vending machine is now quite basic. Although it sells usual snacks and beverages, it has certain special features that we were told to include in order to gain extra points. The only thing that makes my vending machine unique is that it contains features like the ability to view the cart, insert money at any moment, select items at any time, receive the current balance after leaving, and, finally, a feedback feature that allows me to make improvements and add more special features.

Flow Chart Illustration

For a clear view of the illustration click here!



Flow Chart Explanation

This vending machine is created by the Python coding language. First, food and drinks are defined together with their costs, stocks, and labels. Through a menu-driven interface, consumers may engage by choosing any of the items, adding money, seeing their cart, and exiting. The application verifies the accuracy of user inputs and offers feedback, such as items that are out of stock or have an insufficient balance. As users add payments or make purchases, their current balance is kept and updated.

Technical Description

Introduction:

Just like any other vending machine in real life, additional features include the ability to add any amount of money, receive a refund, examine the cart, receive a receipt, and receive feedback as the user is about to exit the vending machine program. The key features of the vending machine I made for this project are listed below this introduction.

1. Program Structure

- Import statements of time and sys for flashy and extra design animations.
- A list of dictionaries regarding the snacks, price, and stocks
- Data definitions of the cart to hold the items that have been purchased, and balance which is a global variable to track the user's balance.

2. Core Functions

- The **opening()** function displays a simple animated message once the program starts functioning.
- The **proceed()** function will ask the user to insert money, update the balance data and then display an animation afterwards. You can not proceed unless you input a valid amount of money.

- The **display_item()** function displays the list of snacks and drinks, along with the prices and stocks available. This function organizes and displays all snacks, drinks, prices, and stocks everytime it has been updated.
- The **insert_money()** function asks the user to input a valid amount of money, this will play an animation after inputting the amount and will then update the balance and play a little animation that the user has input an amount.
- The **select_items()** function is one of the main functions that makes the program usable. The features of this function validates item selection, checks the stocks, deducts balance once an item has been purchased, and updates the stocks and adds the item to the cart.
- The **display_cart()** function displays all the items that have been added to the cart and shows all the items then calculates the total cost.
- The **main()** function is one of the most important functions. It provides 4 options which are to insert money, select items, view the cart, or exit the program.
- The **experience()** function collects a rating from the user's experience with the vending machine of 1-5 stars.
- The **feedback()** function collects and saves the user's feedback in a file named "Feedback.txt"

3. Key Features

- **Inventory Management** - The program tracks the stock level for each snack and drinks. Once an item is out of stock, the program will then notify the user that the selected item is out of stock.
- **Balance Management -** Once the user has inputted a certain amount it will be then updated and kept for future use. If the balance is insufficient for a selected item, the program will tell the user that the user has to add more money in order to purchase the selected item.
- **Cart System** - Selected items that have been purchased will be added to the cart and keeps track of the total cost, the users can also view their cart once they choose the option to view the cart.
- **Error Handling** - Checks the user input to make sure that the incorrect numbers/inputs or menu choices are handled properly. It also prevents unnecessary crashes
- **User Feedback Collection** - Collects the user's opinions about the vending machine program and stores it in a file "Feedbacks.txt" for future references.

4. Technical Breakdown

- **Data Initialization:** The program analyzes the stock and price details for the snacks and drinks using a list of dictionaries.
- **Displaying items:** The program will use the function **display_items()** to show all the available snacks and drinks with their prices and stocks.
- **Inserting money:** The **proceed()** and **insert_money()** are similar which prompt the user to add money, validates the input and then updates the balance. A short animation will play every time the user adds money.
- **Selecting items:** The **select_item()** function makes the user choose and purchase a selected item by their corresponding item code. The program will then check if the user has the right amount of money, deducts the balance once an item has been purchased, and updates the stock.
- **Feedback system:** The program will ask the user to put their experience and feedback regarding the vending machine program. The program will then collect the feedback and put it in a file "Feedbacks.txt"

Critical Reflection

As our final task for this semester, we were given the task to create a vending machine that meets specific requirements in order to receive a point. On top of that, in order to get additional points, we had to add unique features that set our vending machine program apart from other typical vending machines in the real world. As I started this project, I was already looking forward to learning how to build my own vending machine. As a matter of fact I had another idea on how my vending machine would look and how it would function, which is to make a vending machine all about cars and car parts.

The idea was to make a GUI vending machine that looks like Carvana. Now if you do not know what Carvana is, according to google, "Carvana is an online-only used-car retailer that performs almost all the functions a physical dealer would offer: buying and selling cars, accepting trade-ins, and financing purchases. Naturally, the company's site contains a thorough FAQ page, but here's a primer on how it works."

When I tried to make a GUI vending machine, my mind was like this looks like it's too complicated and it would take me weeks, probably months to finish it since it does not look like a normal vending machine and the function is very different. So I had to rethink the whole idea

and it came to the conclusion that I had to make a normal Vending Machine yet with unique features instead to make things easier.

Overall, I am proud of how my Vending Machine has turned out. It may look like a normal vending machine, but some of my functions are unique, which is the feedback and thinking about it makes me want to learn coding more. I could do better if I actually have to implement it like a normal vending machine. Watching more tutorial videos and asking professional coders would make me improve more than I am right now.

Codes

```
import time
import sys

# List/items of snacks and drinks.
snacks = {
    1: {"name": "Cheetos", "price": 6.50, "stock": 5},
    2: {"name": "Doritos", "price": 8.75, "stock": 3},
    3: {"name": "Maltesers", "price": 8.20, "stock": 4},
    4: {"name": "Snickers", "price": 6.15, "stock": 6},
    5: {"name": "Takis", "price": 6.80, "stock": 2},
}
drinks = {
    6: {"name": "Monster", "price": 8.00, "stock": 7},
    7: {"name": "Mountain Dew", "price": 7.00, "stock": 5},
    8: {"name": "Water", "price": 3.00, "stock": 4},
    9: {"name": "Vento", "price": 5.00, "stock": 2},
    10: {"name": "Orange Juice", "price": 4.00, "stock": 3},
}

cart = []
balance = 0.0

# Opening and granted animation.
def opening():
    message = "opening", ".", ".", ".\n"
    for i in message:
        sys.stdout.write(i)
```

```
        sys.stdout.flush()
        time.sleep(0.6)
opening()

def proceed():
    global balance
    while True:
        try:
            # Asks the user how much money to input to proceed.
            amount = float(input("\nInsert money to proceed: "))
            # inserts the user's input and shows the balance.
            if amount > 0:
                balance += amount
                display = "\nInserting",".",".","."
                for i in display:
                    sys.stdout.write(i)
                    sys.stdout.flush()
                    time.sleep(0.5)
                print(f"\nYou have inserted ${amount:.2f}. Current
balance: ${balance:.2f}")
                message2 = "Granted!!!"
                for char in message2:
                    sys.stdout.write(char)
                    sys.stdout.flush()
                    time.sleep(0.09)
                break
            else:
                print("Please insert a valid amount.")
        except ValueError:
            print("Invalid input. Please enter a valid amount.")
proceed()

# Displays the list of items.
def display_item():
    print("\n\nWelcome to Dale's Vending Machine.")
    print("—" * 52)
    print("Here are the list of Snacks 🍫:")
    for number, item in snacks.items():
        print(f"{number}. {item['name']:<25} - ${item['price']:.2f}  -
Stocks: {item['stock']}")
```

```
    print("—" * 52)
    print("Here are the list of Drinks 🧃:")
    for number, item in drinks.items():
        print(f"{number}. {item['name']:<25} - $ {item['price']:.2f}  -
Stocks: {item['stock']}")
    print("—" * 52)
display_item()

# Insert money function.
def insert_money():
    global balance
    while True:
        try:
            # Asks the user how much money to input.
            amount = float(input("\nInsert money (or type 0 to go back):
"))
            # Exits out the insert money function.
            if amount == 0:
                break
            # inserts the user's input and shows the balance.
            elif amount > 0:
                balance += amount
                display = "\nInserting",".",".","."
                for i in display:
                    sys.stdout.write(i)
                    sys.stdout.flush()
                    time.sleep(0.5)
                print(f"\nYou have inserted $ {amount:.2f}. Current
balance: $ {balance:.2f}")
                break
            else:
                print("Please insert a valid amount.")
        except ValueError:
            print("Invalid input. Please enter a valid amount.")

# Selection of items function.
def select_items():
    global balance
    while True:
        try:
```

```
            display_item()
            # asks the user to input what item to purchase.
            user_input = int(input("\nPlease enter an item number to
purchase (or 0 to go back): "))
            # exits out the selections of items.
            if user_input == 0:
                break
            # checks if the user chose a snack or a drink.
            if user_input in snacks:
                item = snacks[user_input]
            elif user_input in drinks:
                item = drinks[user_input]
            else:
                print("❌ Invalid item number ❌. Please try again.")
                continue
            # checks if there are still items in stock and then adds to
the cart.
            if item['stock'] > 0:
                if balance >= item['price']:
                    cart.append(item)
                    item['stock'] -= 1
                    balance -= item['price']
                    print(f"\nAdded {item['name']} to cart. Remaining
stock: {item['stock']}. Current balance: $ {balance:.2f}")
                # executes when the user has low balance.
                else:
                    print(f"Insufficient balance. {item['name']} costs
$ {item['price']:.2f}, but your balance is $ {balance:.2f}. Insert more
money to purchase this item.")
            # Displays that a certain item has ran out of stock.
            else:
                print(f"Sorry, {item['name']} is out of stock 🥺👉👈 pick
something else.")
        except ValueError:
            print("❌ Invalid input ❌. Please enter a number.")

# Shows what is in the cart.
def display_cart():
    # Shows that the cart is empty
    if not cart:
```

```
        print("\nYour cart is empty...")
    else:
    # Shows all the items that have been purchased.
        print("\nHere are the items in your cart:")
        total = 0
        for item in cart:
            print(f"{item['name']} - $ {item['price']:.2f}")
            total += item['price']
        # shows how much all the items are.
        print(f"Total: $ {total:.2f}")

# Options function for user to pick.
def main():
    while True:
        # Options to select on what the user would want to do.
        time.sleep(1.5)
        print(f"\nYour current balance is: 💵 {balance:.2f}")
        print("—" * 52)
        print("\nWhat would you like to do?")
        print("1. Insert more money")
        print("2. Select items")
        print("3. View cart")
        print("4. Exit")
        try:
            # If the user chooses one of the options then one of these
will be called out depending on what the user chooses.
            choice = int(input("Enter your choice: "))
            # Calls out the insert money function.
            if choice == 1:
                insert_money()
            # Calls out the Select items function.
            elif choice == 2:
                select_items()
            # Calls out what is inside the cart function.
            elif choice == 3:
                display_cart()
            # Ends the purchasing and choosing options. This will then
show how much money you have left
            elif choice == 4:
```

```
                print(f"\nThank you for using Dale's Vending Machine!
😊.")
                print("—" * 52)
                print("\nHere is your receipt:")
                print(f"Current balance: $ {balance:.2f}")
                total = 0
                for item in cart:
                    print(f"{item['name']} - $ {item['price']:.2f}")
                    total += item['price']
                # shows how much all the items are.
                print(f"Total: $ {total:.2f}\n")
                print("—" * 52)
                if balance > 0:
                    display = ("Returning balance",".",".",".", f"\nYour
returning balance is 💵 {balance:.2f}. Have a nice day! 😊")
                    for i in display:
                        sys.stdout.write(i)
                        sys.stdout.flush()
                        time.sleep(0.5)
                break
            else:
                print("❌ Invalid choice ❌. Please try again.")
        except ValueError:
            print("❌ Invalid input ❌. Please enter a number.")
main()

# Rating of experience function for user to rate how good or bad the
vending machine is.
def experience():
    while True:
        try:
            user_input = int(input("\nKindly rate your experience ⭐ 1-5:
"))
            # if the user's rate is less than or equal to 3 then this will
be executed.
            if 1 <= user_input <= 3:
                print(f"\nOh wow thats pretty low 😢! Thank you for rating
our Vending Machine. You rated us {user_input}⭐.")
                break
```

```
            # if the user's rate is more than or equal to 5 then this will
be executed.
            elif 4 <= user_input <=5:
                print(f"\nHappy to hear that you like the experience 🤩!
Thank you for rating our Vending Machine. You rated us {user_input}⭐.")
                break
            # this will execute if the user inputs more than 5 or less
than 0.
            else:
                print("The rating is until 1-5 only! Try again.")
        # Executes if the user input is not an integer.
        except ValueError:
            print("Sorry, we do not accept this answer.")
experience()

def feedback():
    # Get user input for feedback
    user_feedback = input("\nPlease help us improve more by giving us
feedbacks (Keep empty if none): ").strip().lower()

    # Try to read existing feedbacks from the file.
    try:
        with open("Feedbacks.txt", "r") as file:
            feedback_list = file.readlines()  # Read all lines from the
file.
    except FileNotFoundError:
        # If the file doesn't exist, start with an empty list.
        feedback_list = []

    # Count how many times the user's feedback has appeared in the file.
    feedback_counts = sum(1 for line in feedback_list if
line.strip().lower() == user_feedback)

    # Append the user's feedback to the file.
    with open("Feedbacks.txt", "a") as file:
        file.write(user_feedback + "\n")

    # Provide feedback to the user about their entry.
    print(f"Thank you for your feedback ❤️! Your feedback has been
recorded {feedback_counts + 1} time(s).")
```

```
feedback()
```