**1️⃣ Plan the Features (Requirements Gathering)**

Before coding, list out the **features** we want:

1. **Login** (via phone/email with OTP validation)
2. **Menu system** (veg, non-veg, snacks, desserts, drinks)
3. **Ordering items** (choose category, item, quantity)
4. **Cart system** (view items, total price, place order)
5. **Logout functionality**

This keeps our app **modular and realistic**.

**2️⃣ Break Down into Modules (Step-by-Step Flow)**

**🔹 Step 1: Menu Data**

* Use a **class-level dictionary** (Dominos.menu) to store categories, items, and prices.
* Example:

class Dominos:

    menu = {'veg': {'margerita': 129, 'cheese\_and\_corn': 169, 'peppi\_paneer': 260, 'veg\_loaded': 210, 'tomato\_tangi': 170},

            'non\_veg': {'pepper\_barbeque': 199, 'non\_veg\_loaded': 169, 'chicken\_sausage': 200},

            'snacks': {'garlic\_bread': 120, 'zingy': 59, 'chicken\_cheese\_balls': 170},

            'desserts': {'choco\_lava': 100, 'mousse cake': 169},

            'drinks': {'coke': 90, 'pepsi': 78, 'sprite': 50}}

This acts like a **database** for our app.

**🔹 Step 2: Class Initialization**

* Create a **Dominos class**.
* Inside \_\_init\_\_, define:
  + User details (name, email, phno)
  + login\_status (True/False)
  + cart (empty dict for orders)

def \_\_init\_\_(self,name,email,phno):

        self.name = name

        self.email = email

        self.phno = phno

        self.login\_status = False

        self.cart = {}

This sets up the **user session**.

**🔹 Step 3: OTP Validation (Login Security)**

* Write a **staticmethod** validate\_otp(value):
  + Generate random OTP (random.randint(1000,9999))
  + Take input from user
  + Compare OTP
  + Return True/False

@staticmethod

def validate\_otp(value):

        while True:

            print("-"\*50)

            og\_otp = random.randint(1000,9999)

            print(f"An otp has been sent to {value}...")

            print(f"your dominos otp is: {og\_otp}")

            otp = int(input("Enter otp: "))

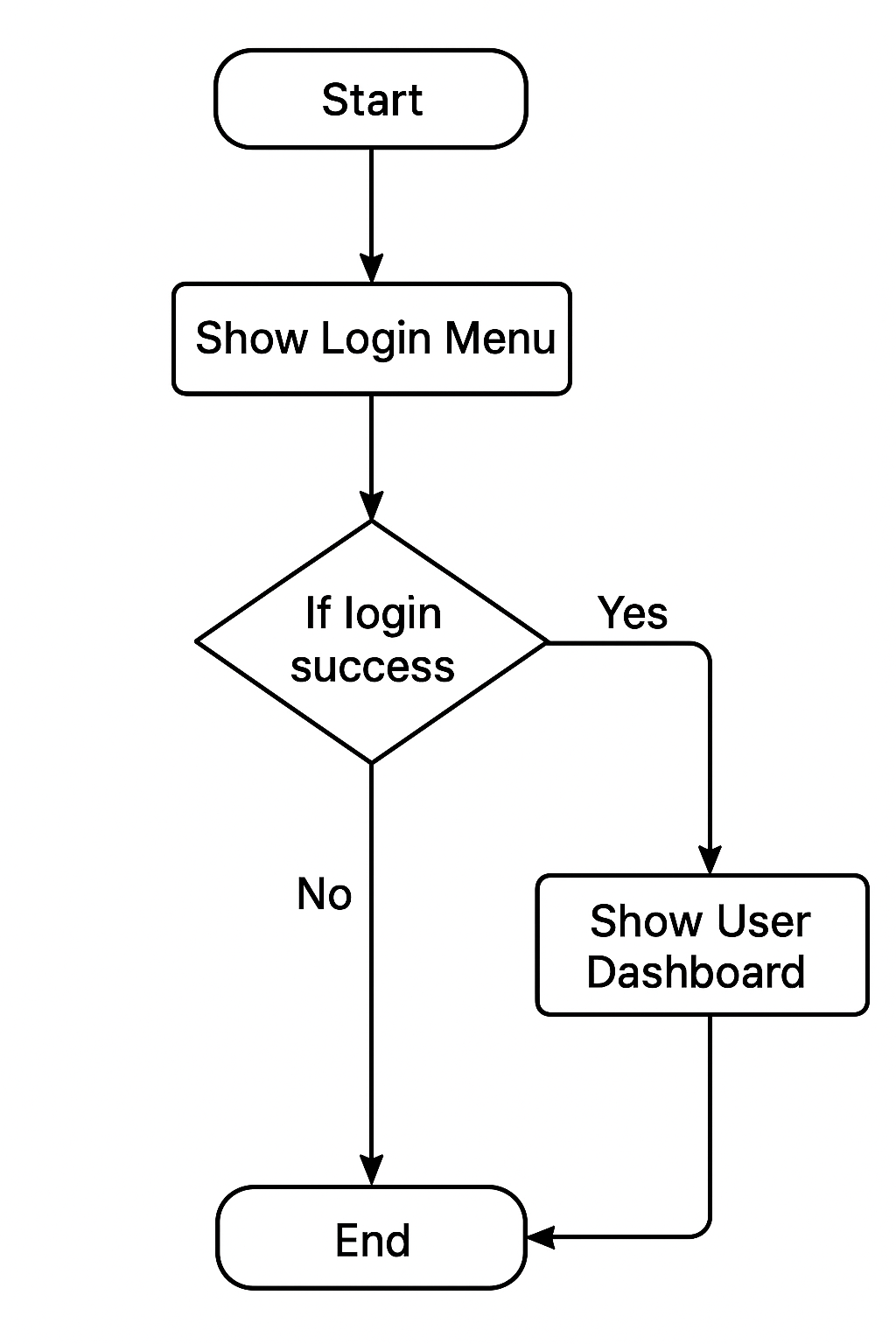
            if otp==og\_otp:

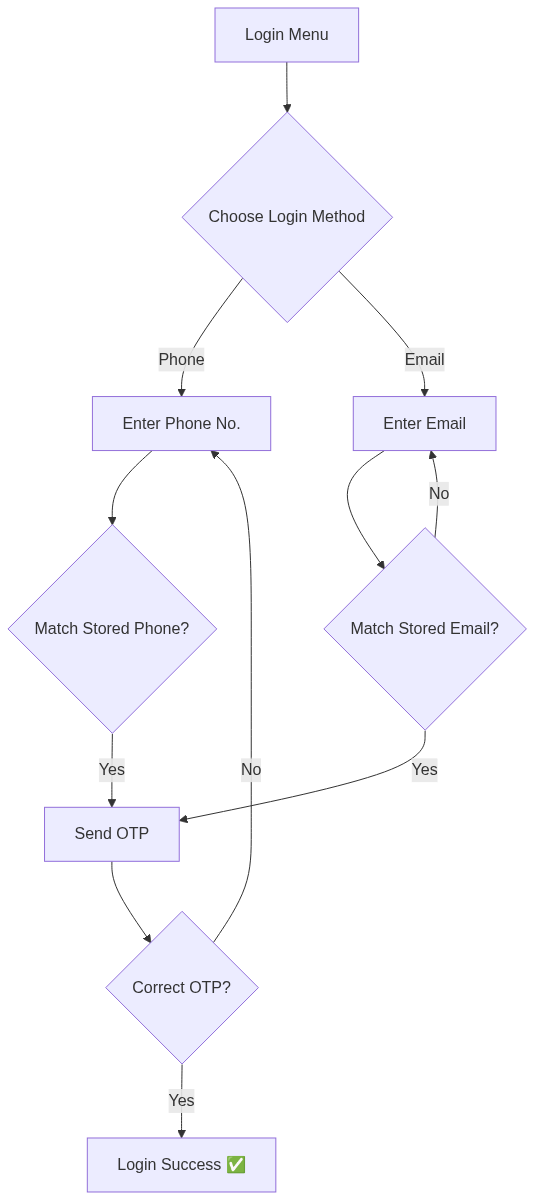
                print("Login successfull ✅")

                return True

            print("Invalid OTP enter correct otp")

This makes our login **feel real**.

****

**🔹 Step 4: Login Functionality**

* Ask user: Login with Phone or Email
* Validate entered phone/email with stored data
* If matched → Call validate\_otp()
* Set login\_status=True after success

def login(self):

        print("Enter 1: Login with Phone")

        print("Enter 2: Login with Email")

        ch = int(input("Enter choice: "))

        if ch==1:

            phno = int(input("Enter phno: "))

            if phno==self.phno:

                otp = self.validate\_otp(phno)

                self.login\_status = otp

            else:

                print("Phone no doesn't exist")

        elif ch==2:

            email = input("Enter Email: ")

            if email==self.email:

                print("Login successfull ✅")

                otp=self.validate\_otp(email)

                self.login\_status=otp

            else:

                print("Email does'nt exist")

        else:

            print("Invalid choice")

This ensures only **valid users** can access ordering.

**🔹 Step 5: Main Menu (User Dashboard)**

* Show options:
  1. Order food
  2. Show cart
  3. Logout
* Loop until user chooses to logout.

def \_\_init\_\_(self,name,email,phno):

        self.name = name

        self.email = email

        self.phno = phno

        self.login\_status = True

        self.cart = {}

        while True:

            if not(self.login\_status):

                print("-"\*50)

                print("-----------------Welcome to DOMINOS🍕 app-------------")

                print("Login")

                ch = input("Do you want to Login? (y/n): ").lower()

                if ch=='y':

                    self.login()

                else:

                    print("Login is required")

            if self.login\_status:

                print("-----------------Welcome to DOMINOS🍕 app-------------")

                print(f"User 👤: {self.name}")

                print("Enter 1: Order")

                print("Enter 2: Show cart")

                print("Enter 3: Logout")

                ch = int(input("Enter choice: "))

                if ch==1:

                    self.order()

                elif ch==2:

                    self.show\_cart()

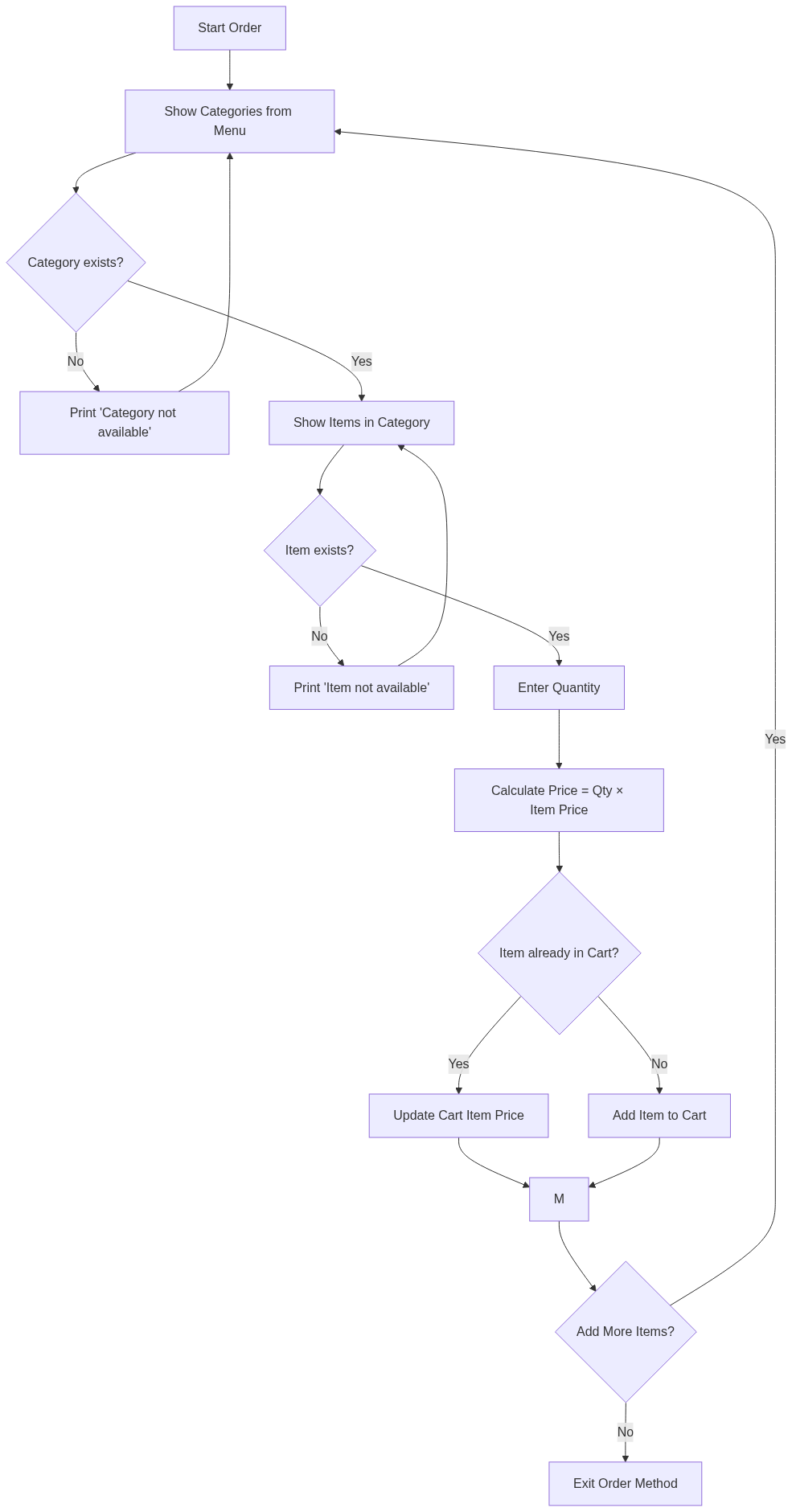
                elif ch==3:

                    self.logout()

                else:

                    print("Invalid choice")

This becomes the **central hub** of the app.

**🔹 Step 6: Order Function**

* Show available categories from Dominos.menu
* User selects category → display items with price
* User selects item → enter quantity
* Calculate price → add to cart
* Ask if they want to add more

def order(self):

        while True:

*#display menu*

            print("--------MENU-------")

            for category in Dominos.menu:

                print(category)

            print('-'\*50)

            cat = input("Enter category: ").lower()

            if cat in Dominos.menu:

*#Display items and price*

                for item in Dominos.menu[cat]:

                    print(f"{item} --------- Rs. {Dominos.menu[cat][item]}")

                item = input("Enter item name: ")

                if item in Dominos.menu[cat]:

                    q = int(input("Enter quantity: "))

                    price = q\*Dominos.menu[cat][item]

                    if item in self.cart:

                        self.cart[item] += price

                    else:

                        self.cart[item] = price

                    print(f"{item} added to the cart")

                    choice = input("Do you want to add more? (y/n): ").lower()

                    if choice=='n':

                        break

                else:

                    print(f"{item} not available")

            else:

                print(f"{cat} not available")

This mimics **real Dominos ordering flow**.

**🔹 Step 7: Cart Function**

* Print all items in cart with total price
* Ask if they want to place order
* If yes → Thank them, empty the cart

def show\_cart(self):

        print('---------Dominos Cart-----------')

        if self.cart!={}:

            total = 0

            for item in self.cart:

                price = self.cart[item]

                total+=price

                print(f"{item} ----- Rs. {price}")

            print("Total Price: Rs.",total)

            ch = input("place order ? (y/n): ").lower()

            if ch=='y':

                print("Thank you placing the order")

                print("your order will arrive soon ")

                self.cart = {}

        else:

            print("Your cart is empty")

This finalizes the **order placement**.

**🔹Step 8: Logout Function**

* Ask confirmation
* Set login\_status=False
* Redirect back to login menu

def logout(self):

        ch = input("Do you want to logout? (Y/N): ").lower()

        if ch=='y':

            self.login\_status=False

        print("Thank you For using our app")

Keeps session **clean**.

**Coding Flow**

Here’s how you should **develop step by step**:

1. Start with the **menu dictionary**.
2. Build the **class skeleton** with \_\_init\_\_.
3. Add **login system** (login, validate\_otp, logout).
4. Add **ordering system** (order method).
5. Add **cart system** (show\_cart method).
6. Wrap everything in a **main loop** inside \_\_init\_\_ so user keeps interacting.

**Possible Enhancements (Future Upgrades 🚀)**

* Add **signup functionality** (store multiple users in a dict).
* Store orders in a **text file** (file handling).
* Add **discount codes**.
* Track **order history**.
* Add **delivery status simulation**.

**Final Flow Diagram (Visual)**

Start

↓

Show Login Menu

↓

If login success → Show User Dashboard

┌─────────────────────────────┐

│ 1. Order → Select → Add to Cart │

│ 2. Show Cart → Place Order? │

│ 3. Logout → Back to Login │

└─────────────────────────────┘

↓

End