Logic

1. User Input:

Collect customer details: This involves a user-friendly interface which allows users to input their personal information, such as their first and last name, phone number, email address, home address, city, county and Eircode. This information is needed for processing orders and ensuring accurate and efficient delivery.

e.g. To collect the customer's first name, we might use a message such as "Enter first name here" and then use the 'input()' function in Python to attain their input.

2. Calculations:

Calculate total cost of the order: This step involves computing the total cost of the customer's order. To do this, we can sum up the costs of the selected pizzas, add any additional costs for toppings or dips, apply any discounts if the order meets certain criteria (e.g. €35 or more), add a delivery fee if applicable, and calculate the VAT based on the final total.

e.g. Using Python, we can sum up the costs of the selected items using the 'sum()' function and apply discounts based on conditional statements (e.g. 'if total_cost >= 35: '). We can then calculate the VAT by multiplying the total cost by the VA rate (0.23)

3. Display Information:

Display a summary to the customer: After collecting the necessary customer and order details and performing calculations, it is important to present a clear summary to the user for verification. This summary will include the customer details such as name and contact information, as well as the order details (the number of pizzas, selected toppings and dips if necessary, any discounts applied, and the total cost including VAT)

e.g. We can use Python's 'print()' function to display the summary in a structured format, including the relevant information gathered from the user input and calculations performed.

4. File Handling:

Generate a unique receipt number for each order: Each order should have a unique identifier or receipt number associated with it. This number is used for tracking purposes and for referencing the order in the system.

Write details to a text file for receipt: Once the order details have been collected and processed, they should be stored in a text file for record-keeping. This file typically includes information such as the receipt number, customer details, order items, discounts applied and total cost.

e.g. In Python, you can generate a unique receipt number using libraries such as 'uuid'. To write details to a text file, we can use the 'open()' function with the

correct/ appropriate file mode (e.g. 'w' for writing) and then use methods like 'write()' to write the necessary information to the file.

5. **<u>Modular Coding:</u>**

   Organise our code into separate files for databases, processes, and GUI. Use classes and functions to modularize your codebase effectively.

6. **<u>Coding Best Practices:</u>**

   Refactor Regularly: Review and refactor code regularly to maintain readability and performance.
   Code Reviews: Conduct regular code reviews to gather feedback and ensure adherence to coding standards.
   Error Handling: Ensure that our code handles unexpected situations smoothly and gracefully.

7. **<u>User Interface:</u>**

   Intuitive Design: Create a user-friendly interface which is easy to use but also attracts the eye of the user.
   Validation: Implement input validation to ensure correctness and to prevent errors.
   Responsive Design: Design the interface to adapt seamlessly across various devices and screen sizes.

8. **<u>External sources:</u>**

   Documentation: Provide clear documentation or references to external sources used in the project.
   Acknowledgements: Acknowledge the contributions of external sources in project documentation.

9. **<u>Testing:</u>**

   Unit Testing: Write and execute unit tests to validate individual component's functionality.
   Integration Testing: Test interactions between different components to ensure seamless integration.
   User Acceptance Testing: Test the system to validate system functionality and usability.