1. **User Input:**
   * Collect customer details:
     + First name (up to 30 characters, non-empty)
     + Last name (up to 30 characters, non-empty)
     + Phone number (in format "+\_\_\_ (\_\_\_) \_\_\_\_\_\_\_", non-empty)
     + Email address (valid format, non-empty)
     + Street address (for delivery, up to 30 characters, non-empty)
     + Town/City (for delivery, up to 30 characters, non-empty)
     + County (for delivery, up to 30 characters, non-empty)
     + Eircode (for delivery, in format "XXX XXXX", non-empty)
   * Collect pizza order details:
     + Pizza type, size, toppings, dips from flat-file database (txt/csv/json)
     + Display options to the operator
   * Allow the operator to add:
     + Any number of pizzas or none
     + Any number of extras or none
2. **Calculations:**
   * Calculate total cost of the order:
     + Sum up the cost of selected pizzas
     + Add the cost of selected extras
   * Apply 10% discount for orders of €35 or more
     + If applicable, subtract 10% from the total cost
   * Add delivery fee (if applicable) to the discounted total
   * Calculate VAT (23%):
     + Apply VAT to the new total (pizza cost + extras cost + delivery fee)
3. **Display Information:**
   * Display a summary to the operator:
     + Customer details
     + Number of pizzas with type and size
     + Number and type of extra toppings/dips
     + Discounts and VAT
4. **File Handling:**
   * Generate a unique receipt number for each order
   * Write details to a text file for receipt:
     + Include customer details, order details, discounts, VAT
   * Update flat-file databases:
     + Orders database:
       - Use receipt number as the key
       - Include details of the order (pizzas, extras, discounts, VAT)
     + Customers database:
       - Assign a unique ID to each client
       - Include customer details
5. **Modular Coding:**
   * Use modular principles with dedicated files for:
     + Databases (orders, customers)
     + Processes (calculations, file handling)
     + GUI
6. **Coding Best Practices:**
   * Consistent and efficient coding structures
   * Meaningful naming conventions for classes, sub-classes, properties, methods
   * Relevant and useful comments throughout the code
7. **User Interface:**
   * Create a user-friendly interface for reliability and ease of use
   * Ensure GUI allows smooth interaction for entering details and viewing summary
8. **External Sources:**
   * If any external sources are used/adapted, provide references in the code
9. **Additional Functionality:**
   * Encouraged to add any additional functionality deemed necessary to improve the system and demonstrate coding skills
10. **Testing:**
    * Implement testing procedures to ensure the reliability of the system
    * Check for various scenarios (e.g., different order sizes, combinations, discounts, delivery options)