1. System Context

- System type: Desktop application
- Technology: Java (NetBeans IDE)
- Mode: **Offline**
- Users:
    - Coffee shop **Owner/Staff**
    - **Customer** (indirectly, through actions handled by staff)
- Main goal: Handle drink/food ordering, order processing, billing, and basic records management for a coffee shop.

Assume the system supports at least:

1. Managing menu items (add/edit/remove items like coffee, pastries, etc.)
2. Taking customer orders and placing them in an order queue
3. Processing and completing orders (mark as served/paid)
4. Searching for orders or items (by order ID, customer name, or item name)
5. Sorting lists (e.g., menu by price or name, sales records by date or total amount)

---

2. Required Data Structures & Algorithms

Design the system so that it explicitly uses:

1. **Linked List + Queue**
    - Implement a **Linked List–based Queue** to manage customer orders (FIFO: first order in, first order out).
    - Clearly show:
        - Enqueue (adding a new order to the queue)
        - Dequeue (processing/serving the next order)
        - Traversal (listing all current orders)
2. **Searching Algorithm: Linear Search**
    - Use **Linear Search** to:
        - Search for a specific order in the queue or order history (by order ID, customer name, etc.)
        - Search for a specific menu item (by item code or item name)
    - Show where in the system flow the search is triggered and how the algorithm operates on the data.
3. **Sorting Algorithm: Insertion Sort**
    - Use **Insertion Sort** to sort:
        - Menu items (by price, name, or category), and/or
        - Daily sales records (by date/time or total amount)
    - Explain:
        - When sorting is triggered (e.g., when viewing sorted lists, generating reports)
        - How the data is prepared for sorting (array or list)
        - How the insertion sort steps apply to that data.

Scenario: “Rita Brew” Coffee Shop – Queuing, Searching, and Sorting in Action

Setting:
“Rita Brew” is a small coffee shop near a university in **Bustos**. It gets busiest during **7:00–9:00 AM** and **4:00–6:00 PM** when students and office workers line up for coffee and snacks.

The shop has:

- **1 cashier** (takes orders),
- **2 baristas** (prepare drinks),
- **1 owner/manager** (checks sales reports).

1) Linked List + Queue (FIFO Orders)

Morning Rush: Order Entry (Enqueue)

The cashier enters six orders in the exact sequence they arrive:

1. **Order #1007** – Albert orders a Hot Latte
2. **Order #1008** – Ryan orders a Hot Matcha
3. **Order #1009** – Nestor orders an Hot Chai Tea
4. **Order #1010** – Bernie orders a Iced Caramel Macchiato
5. **Order #1011** – Rommel orders a Iced Mocha
6. **Order #1012** – Thea orders a Hot Matcha Latte

Each order is **enqueued** into the **Linked List–based Queue** (FIFO):

Front -> #1007 -> #1008 -> #1009 -> #1010 -> #1011 -> #1012 -> null

Serving Orders (Dequeue)

The barista serves the **first order** in the queue:

- **Dequeue:** #1007 (Albert’s Hot Latte)
- Queue becomes:

Front -> #1008 -> #1009 -> #1010 -> #1011 -> #1012 -> null

This guarantees **first order in = first order out**.

Traversal (Viewing Queue)

When the cashier checks the queue list, the system **traverses** the linked list and displays:

- Ryan – Hot Matcha
- Nestor – Hot Chai Tea
- Bernie – Iced Caramel Macchiato
- Rommel – Iced Mocha
- Thea – Hot Matcha Latte

This allows the staff to see all **current waiting orders**.

2) Linear Search (Orders + Menu)

(A) Searching Orders

If **Ryan** returns and asks:

“Kuya, is my order ready?”

The cashier types **“Ryan”** into the search box.
Linear Search checks orders one by one in the queue:

- #1008 (Ryan) **Match found**
- #1009 (Nestor) not match
- #1010 (Bernie) not match
- #1011 (Rommel) not match
- #1012 (Thea) not match
- **Stops**

The cashier confirms Ryan’s order is **still in the queue**.

But if **Rommel** returns and asks:

“Kuya, is my order ready?”

The cashier types “Rommel” into the search box.
Linear Search checks orders one by one in the queue:

- #1008 (Ryan) not match
- #1009 (Nestor) not match
- #1010 (Bernie) not match
- #1011 (Rommel) **Match found**
- #1012 (Thea) not match
- **Stops**

The cashier confirms Rommel’s order is still in the queue.

(B) Searching Menu Items

A customer says:

“Do you have anything with caramel?”

The cashier types **“caramel”** in the menu search.

Linear Search scans each menu item:

- Latte → no
- Espresso → no
- Caramel Frappe → yes
- Caramel Macchiato → yes

Matches are displayed instantly.

3) Insertion Sort (Menu + Sales)

(A) Sorting Menu by Price

A student wants the **cheapest drink**.

The cashier selects **Sort by Price**, and **Insertion Sort** arranges menu items:

Before sorting: (example only)

- Matcha Latte – ₱170
- Espresso – ₱110
- Hot Latte – ₱150
- Iced Mocha – ₱180

After Insertion Sort: (example only)

1. Espresso – ₱110
2. Hot Latte – ₱150
3. Matcha Latte – ₱170
4. Iced Mocha – ₱180

(B) Sorting Sales Records

The owner wants to review sales in chronological order.

The system gathers all sales records for the day and **Insertion Sorts** them by time:

Example output:

- Add example here

This helps identify **peak sales date**.

How This Scenario Meets the Requirements

- **Linked List + Queue:**
  Orders are added (enqueue), served (dequeue), and listed (traversal) using FIFO order.
- **Linear Search:**
  Used when staff search for a specific order or menu item.
- **Insertion Sort:**
  Used when sorting menu items by price or daily sales by time