SmartKitchenFX Documentation

🔹 Project Overview

SmartKitchenFX is a distributed system simulation built with **JavaFX**, implementing **Lamport's Logical Clock** and **Agrawala's Mutual Exclusion Algorithm** in a restaurant-style environment.

Each client node represents a **cash desk** placing food orders concurrently, while the **central server** manages order sequencing and fairness through Lamport timestamps.

The latest **ModernFX Edition** revamps the UI with a polished, modular design — featuring **interactive dashboards**, **queue visualization**, and **real-time state logging**.

🔹 Architecture

```
SmartKitchenFX/
│
├── app/
│   ├── src/main/java/smk/
│   │   ├── client/
│   │   │   ├── ui/
│   │   │   │   ├── SmartKitchenClientModernApp.java
│   │   │   │   └── ClientTerminalController.java
│   │   │   └── CartRow.java
│   │   │
│   │   ├── server/
│   │   │   ├── ui/
│   │   │   │   ├── SmartKitchenServerModernApp.java
│   │   │   │   └── ServerDashboardController.java
│   │   │
│   │   └── shared/
│    │         ├── LamportClock.java
│    │         └── OrderRow.java
│    │
│    ├── resources/
│    │    ├── css/smk.css
│    │    ├── fxml/ClientTerminal.fxml
│    │    └── fxml/ServerDashboard.fxml
│    │
│    ├── build.gradle
│    └── settings.gradle
│
└── build/ (generated)
```

🔹 Requirements

REQUIREMENT	VERSION / DESCRIPTION
JAVA JDK	17+ (recommended 21)
GRADLE	Included wrapper (`./gradlew`)
JAVAFX SDK	Managed automatically via Gradle
IDE	IntelliJ IDEA / VS Code / Eclipse
OS	Windows, macOS, or Linux

🔹 Features

🔎 Client (SmartKitchenClientModernApp)

- Menu browsing with images and categories
- Add dishes to cart and calculate totals
- Search, filter, and checkout simulation
- Sends logical timestamped orders (Lamport)

🛠️ Server (SmartKitchenServerModernApp)

- Live Lamport queue visualization (modern cards instead of tables)
- Displays logical clock, queue head, and live stats
- Real-time log feed with timestamped actions
- Manual controls for **SimRecv**, **Start**, and **End**

⚙️ Shared Core

- `` `LamportClock.java` ``: Logical clock synchronization
- `` `OrderRow.java` ``: Comparable data structure for order sorting

🔹 Running the Project

Step 1 – Build

```
bash
./gradlew build
```

Step 2 – Run the Client

```
bash
./gradlew :app:run -
PmainClass="smk.client.ui.SmartKitchenClientModernApp"
```

Step 3 – Run the Server

```
bash
./gradlew :app:run -
PmainClass="smk.server.ui.SmartKitchenServerModernApp"
```

> 💡 Run them in **separate terminals** or IDE instances for best results.

🔹 How It Works

1. **Clients** simulate sending order requests, each tagged with their Lamport timestamp.
2. The **Server** receives these requests, updates its logical clock, and enqueues them.
3. Orders are displayed in causal order in the Lamport queue.
4. Manual or automatic operations simulate **Start/End** of meal preparation.

🔹 Communication Flow

```
Client → Server
   SEND(Order, ts=3)
        ↓
Server:
   L = max(Ls, ts) + 1
   Enqueue order
   Display in Lamport queue
```

🔹 UI Components

- **Sidebar:** Navigation panel, status indicator, logout button
- **Top Bar:** Clock display, queue size, search bar, and action buttons
- **Main Panel:** Lamport queue with color-coded cards
- **Right Column:** Stats panel and live log console

🔹 Example Log Output

```
[RECV] C1 Pizza ts=1 → L=3
[START] C1 Pizza (Lq=3) S(L)=4
[END]   C1 Pizza DONE. S(L)=5
```