SmartKitchenFX — Next Steps

🔹 Step-by-Step Roadmap

1️⃣ Integrate Real Socket Layer

Goal: Replace simulated buttons with actual message passing.

```
ServerSocket server = new ServerSocket(5000);
while (true) {
    Socket client = server.accept();
    new Thread(() -> handleClient(client)).start();
}
```

2️⃣ Client Communication

Send order data through a socket when pressing Send Order.

```
Socket socket = new Socket("localhost", 5000);
PrintWriter out = new PrintWriter(socket.getOutputStream(),
true);
out.println(orderAsJson);
```

3️⃣ Shared Message Schema

Create a shared class for communication types:

```
enum MessageType { REQUEST, REPLY, RELEASE, ORDER }
```

Message format:

```
{
  "type": "ORDER",
  "client": "C1",
  "dish": "Pizza",
  "ts": 3
}
```

4️⃣ Server Listener Thread

Handle real incoming connections and update the UI via JavaFX threads.

```
Platform.runLater(() -> onOrderReceived(client, dish, ts));
```

5️⃣ Synchronize Multiple Clients

Each client has its own Lamport clock. Broadcast messages to all peers and handle deferred replies for Agrawala's algorithm.

6️⃣ Add State Machine (Agrawala)

Implement client states: - REQUESTING - HELD - RELEASED

Handle messages: REQUEST → REPLY → RELEASE.

7️⃣ UI Integration

Use Platform.runLater() to update: - Queue visuals - Clock labels - Logs & status badges

8️⃣ Testing

Run multiple clients:

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

Check message flow, timestamps, and synchronization.

9️⃣ Optional Add-ons

- REST API or WebSocket backend
- Data export (JSON, CSV)
- Visual network map
- Automatic simulation timer

🛠️Expected Deliverables

GOAL	DESCRIPTION
✅ REAL-TIME COMMUNICATION	Clients send live orders to the server
✅ SYNCHRONIZED LAMPORT CLOCKS	Distributed logical time maintained
✅ VISUAL QUEUE UPDATES	Server displays orders in causal order

GOAL	DESCRIPTION
✅ LOGS & METRICS	Live console + GUI updates

🎉 Outcome

After implementing the socket layer and linking your existing Agrawala logic: - The system will become a true distributed restaurant simulation. - The new UI will act as a visual dashboard for concurrency control. - Perfect for both academic defense and portfolio demonstration.