Flowable Hands-on Exercise: Message Event

Scenario: Customer Support Ticket Processing

A company handles **customer support tickets**. When a **ticket is created**, it is assigned to an **agent** for resolution. However, if the customer provides **additional information**, the process should handle the update before proceeding.

This exercise will demonstrate how to:
✅ Use a **Message Start Event** to trigger a new process.
✅ Use a **Message Intermediate Catch Event** to wait for external input.
✅ Use a **Message Boundary Event** to handle updates dynamically.

Step 1: Create a New BPMN Process in Flowable

1. Open **Flowable Modeler**.
2. Create a new **BPMN Process Model** named **Support Ticket Process**.
3. Set **Process ID** as `supportTicketProcess`.

Step 2: Define the Process Flow

1️⃣ Start Event

- Drag a **Start Event** onto the canvas.
- Name it **New Support Ticket Received**.

2️⃣ User Task (Assign Ticket)

- Drag a **User Task** and name it **Assign to Support Agent**.
- Assign it to the `support_agents` group.

3️⃣ Message Intermediate Catch Event (Waiting for Customer Update)

- Drag an **Intermediate Catch Event** onto the canvas.
- Name it **Wait for Customer Update**.
- Set it as a **Message Catch Event** listening for `"ticketUpdateMessage"`.

4️⃣ User Task (Resolve Ticket)

- Drag another **User Task** named **Resolve Ticket**.

- Assign it to `support_agents`.

5️⃣ Message Boundary Event (Customer Update During Resolution)

- Drag a **Message Boundary Event** onto the **Resolve Ticket** task.
- Name it **Customer Sent an Update**.
- Set it as **Non-Interrupting** (so it can handle multiple updates).
- Connect it back to **Wait for Customer Update**, allowing multiple updates.

6️⃣ End Event

- Connect **Resolve Ticket** to an **End Event** named **Ticket Resolved**.

Step 3: Define Message Events

- **Message Catch Event** (`ticketUpdateMessage`) waits for an update.
- **Message Boundary Event** (`ticketUpdateMessage`) listens for updates during resolution.

Step 4: Deploy and Test

1. Deploy the **Support Ticket Process**.
2. Start a new process instance with:

```
{
  "ticketId": "TCKT123",
  "customerName": "Alice"
}
```

3. Simulate a **customer update** using the **Flowable REST API**:

```
POST http://localhost:8080/flowable-rest/service/runtime/executions
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
{
  "messageName": "ticketUpdateMessage",
  "processInstanceId": "12345",
  "variables": [
    {
      "name": "additionalInfo",
      "value": "Customer provided new details."
    }
  ]
}
```

4. Observe that the process updates dynamically, allowing customer input before resolution.

Expected Behavior

Scenario	Message Event Triggered	Outcome
No customer update	✗ No	Ticket follows normal resolution flow
Customer updates ticket	✓ Yes	Process waits for update before resolving