

# 🎨 Visual Workflow Creation Activity

## Part 3 of Workshop: Design Agent Logic Visually

### 🎯 What You'll Create

A **visual flowchart** showing how your insurance agent thinks and makes decisions.

**Time:** 20 minutes

**Tool:** Mermaid Live Editor (<https://mermaid.live/>)

**No coding required!**

### 💻 The Visual Workflow You're Building

```
graph TD
    Start([User Message]) --> Gather[Gather Information]
    Gather --> HasInfo{Has age, vehicle, history?}
    HasInfo -->|No| AskMore[Ask More Questions]
    AskMore --> Gather
    HasInfo -->|Yes| Calculate[Calculate Premium]
    Calculate --> Explain[Explain Quote]
    Explain --> Done([End])

    Gather --> IsQuestion{User asking What is X?}
    IsQuestion -->|Yes| SearchKB[Search Knowledge Base]
    SearchKB --> Answer[Provide Answer]
    Answer --> Gather
```

**This shows:**

- 🟢 **Green ovals** = Start/End points
- 🟢 **Blue rectangles** = Actions the agent takes
- ⚡ **Orange diamonds** = Decisions the agent makes
- ➔ **Arrows** = Flow from one step to another

### 🚀 Step-by-Step: Create Your Visual Workflow

#### Step 1: Open Mermaid Live Editor (1 min)

1. Go to: <https://mermaid.live/>

2. You'll see:

- **Left side:** Code editor

- **Right side:** Visual preview

3. Clear the default code

---

## Step 2: Start with the Template (2 min)

**Copy this template into the left side:**

```
graph TD
    Start([User: I need insurance]) --> Node1[Agent does something]
    Node1 --> Decision{Agent decides}
    Decision -->|Option 1| Node2[Action 1]
    Decision -->|Option 2| Node3[Action 2]
    Node2 --> End([Done])
    Node3 --> End
```

You should see a flowchart appear on the right!

---

## Step 3: Design the Insurance Agent Flow (10 min)

**Replace the template with the insurance agent logic:**

```
graph TD
    Start([User Message]) --> Gather[Gather Information]

    Gather --> HasInfo{Has enough info?}

    HasInfo -->|No - Missing details| AskMore[Ask Questions]
    AskMore --> Gather

    HasInfo -->|Yes - Have everything| Calculate[Calculate Premium]
    Calculate --> Explain[Explain Quote]
    Explain --> Done([Conversation Complete])

    Gather --> IsQuestion{Is it a question?}
    IsQuestion -->|Yes - User asks What is X| SearchKB[Search Knowledge Base]
    SearchKB --> Answer[Provide Answer]
    Answer --> Gather

    IsQuestion -->|No - Providing info| HasInfo
```

**What this shows:**

1. **Agent gathers information** from user
2. **Decision 1:** Does agent have enough info?
  - **No** → Ask more questions (loop back)
  - **Yes** → Calculate quote
3. **Decision 2:** Is user asking a question?

- **Yes** → Search knowledge base
  - **No** → Continue gathering
- 

## Step 4: Customize Your Design (5 min)

**Add your own logic!** Examples:

**Add insurance type detection:**

```
graph TD
    Start([User Message]) --> DetectType{Auto or Home?}
    DetectType -->|Auto| GatherAuto[Gather Auto Info]
    DetectType -->|Home| GatherHome[Gather Home Info]
    DetectType -->|Unknown| AskType[Ask Insurance Type]
```

**Add error handling:**

```
Calculate --> CheckValid{Valid data?}
CheckValid -->|Yes| Explain>Show Quote
CheckValid -->|No| Error>Show Error Message
Error --> AskMore
```

**Add human review (HTL):**

```
Calculate --> CheckConfidence{Confidence > 95%?}
CheckConfidence -->|Yes| AutoApprove[Auto-approve]
CheckConfidence -->|No| HumanReview[Send to Human]
```

---

## Step 5: Export Your Design (2 min)

1. **Click** the download icon (top right)
  2. **Choose** "PNG" or "SVG"
  3. **Save** the image
  4. **Share** with implementers!
- 

## Visual Design Patterns

### Pattern 1: Simple Linear Flow

```
graph LR
    A[Start] --> B[Gather]
    B --> C[Calculate]
```

```
C --> D[Explain]
D --> E[End]
```

**Use for:** Simple, straightforward processes

---

## Pattern 2: Decision Tree

```
graph TD
    A[Start] --> B{Question?}
    B -->|Yes| C[Search KB]
    B -->|No| D[Gather Info]
    C --> E[Answer]
    D --> F{Has Info?}
    F -->|Yes| G[Calculate]
    F -->|No| D
```

**Use for:** Agents that make decisions

---

## Pattern 3: Loop Back

```
graph TD
    A[Ask Question] --> B[Get Answer]
    B --> C{Complete?}
    C -->|No| A
    C -->|Yes| D[Done]
```

**Use for:** Iterative information gathering

---

## 💡 Test Your Visual Design

### Walk Through Scenarios

#### Scenario 1: Happy Path

User: "I'm 28, drive a 2020 Honda Civic, licensed 10 years, no accidents"

Follow the arrows:

Start → Gather → HasInfo? → Yes → Calculate → Explain → Done ✓

#### Scenario 2: Missing Info

User: "I need car insurance"

Follow the arrows:

Start → Gather → HasInfo? → No → AskMore → Gather (loop)

### Scenario 3: Knowledge Question

User: "What is collision coverage?"

Follow the arrows:

Start → Gather → IsQuestion? → Yes → SearchKB → Answer → Gather

**Does your design handle all scenarios?** If not, add more nodes/decisions!

---

## 💡 Tips for Great Visual Designs

### 1. Keep It Simple

- Start with 3-5 nodes
- Add complexity later
- Each node = one clear action

### 2. Use Clear Labels

Bad: "Process"

Good: "Calculate Premium"

Bad: "Check"

Good: "Has age, vehicle, history?"

### 3. Show Loops

```
graph TD
    A[Ask] --> B[Answer]
    B --> C{Done?}
    C -->|No| A
    C -->|Yes| D[End]
```

### 4. Use Colors (optional)

```
graph TD
    A[Start]:::green --> B[Process]:::blue
```

```
B --> C{Decision}:::orange  
  
classDef green fill:#90EE90  
classDef blue fill:#87CEEB  
classDef orange fill:#FFB347
```

---

## From Visual to Code

### Your Visual Design:

```
graph TD  
    Gather --> HasInfo{Has info?}  
    HasInfo -->|No| AskMore  
    HasInfo -->|Yes| Calculate
```

### Becomes Code (Implementers do this):

```
# Nodes  
workflow.add_node("gather", gather_node)  
workflow.add_node("calculate", calculate_node)  
  
# Decision  
def has_enough_info(state):  
    if state["user_info"]["age"] and state["user_info"]["vehicle"]:  
        return "calculate"  
    else:  
        return "gather"  
  
# Edges  
workflow.add_conditional_edges(  
    "gather",  
    has_enough_info,  
    {  
        "calculate": "calculate",  
        "gather": "gather" # Loop back  
    }  
)
```

---

### You designed it visually, they coded it!

## Workshop Activity Checklist

### Orchestrators:

- Open Mermaid Live Editor

- Copy insurance agent template
- Customize for your use case
- Test with 3 scenarios
- Export as PNG
- Share with implementers
- Explain your logic

### **Implementers:**

- Receive visual design
- Understand the flow
- Code nodes and edges
- Test with orchestrator's scenarios
- Show orchestrators it works!

## ⌚ Example: Complete Insurance Agent Workflow

```
graph TD
    Start([User Message]) --> Gather[Gather Information]

    Gather --> Type{Insurance Type Known?}
    Type -->|No| AskType[Ask: Auto or Home?]
    AskType --> Gather

    Type -->|Yes| Question{Is it a question?}

    Question -->|Yes - What is X| Search[Search Knowledge Base]
    Search --> Provide[Provide Answer]
    Provide --> Gather

    Question -->|No - Providing info| Extract[Extract Information]
    Extract --> Store[Store in Memory]
    Store --> Check{Has all required info?}

    Check -->|No - Missing details| AskMore[Ask for Missing Info]
    AskMore --> Gather

    Check -->|Yes - Complete| Calculate[Calculate Premium]
    Calculate --> Confidence{Confidence > 95%?}

    Confidence -->|Yes| AutoApprove[Show Quote]
    Confidence -->|No| HumanReview[Flag for Human Review]

    AutoApprove --> Explain[Explain Breakdown]
    HumanReview --> Explain

    Explain --> Upsell{Offer Additional Coverage?}
    Upsell -->|Yes| Suggest[Suggest Add-ons]
    Upsell -->|No| Done([Conversation Complete])
```

Suggest --> Done

### This shows:

- Information gathering loop
  - Knowledge base search
  - Decision making
  - Human-in-the-loop
  - Upselling logic
  - Complete conversation flow
- 

## Resources

### Tools:

- **Mermaid Live:** <https://mermaid.live/>
- **Mermaid Docs:** <https://mermaid.js.org/syntax/flowchart.html>
- **Draw.io:** <https://app.diagrams.net/> (alternative)

### Workshop Materials:

- **Visual LangGraph Guide:** [docs/VISUAL\\_LANGGRAPH\\_GUIDE.md](#)
  - **Workshop Guide:** [docs/WORKSHOP\\_90MIN\\_GUIDE.md](#)
- 

## Success Criteria

### Your visual design should:

- Show all major actions (gather, calculate, search)
- Include decision points (has info? is question?)
- Have clear flow with arrows
- Handle loops (ask more → gather)
- Be understandable by implementers

### Implementers should be able to:

- Look at your design
  - Understand the logic
  - Code it in 10 minutes
  - Run it successfully
- 

## You're a Visual Designer!

### Remember:

- Visual design is just as important as code
- Your flowcharts guide the implementation

- Good design = better agent
  - Collaborate with implementers
  - Iterate and improve!
- 

**Next:** Implementers will turn your visual design into working LangGraph code! 