**1. Flow Overview (Step by Step)**

**Step 1: Publisher**

* Your .NET app sends a message like "Order Shipped" or "Out for Delivery".
* This message is sent to **RabbitMQ** queue.

**Step 2: Docker RabbitMQ**

* RabbitMQ running on your machine (localhost:5672) receives the message.
* Queue stores messages until a subscriber reads them.

**Step 3: Subscriber**

* Another .NET app listens to RabbitMQ queue.
* When a new message arrives, the subscriber **reads it**.

**Step 4: SignalR Hub**

* Subscriber then sends the received message to **SignalR Hub** in your ASP.NET Core app.
* SignalR hub **broadcasts the message** to all connected frontend clients in real-time.

**Step 5: Frontend**

* Your frontend (React + TypeScript + Tailwind) connects to the **SignalR hub** via WebSocket.
* When SignalR sends a message, the frontend **updates the UI**, e.g., shows a progress bar for “Order Placed → Shipped → Delivered”.

**2. Flowchart Diagram**

Here’s a **text-based flowchart** to visualize:

Publisher (.NET App)

|

| sends "Order Shipped" message

v

Docker RabbitMQ

|

| stores message in queue

v

Subscriber (.NET App)

|

| receives message from queue

v

SignalR Hub (ASP.NET Core)

|

| broadcasts message to clients

v

Frontend (React + TypeScript + Tailwind)

|

| UI updates in real-time (progress bar / order tracking)

v

User sees order status update