

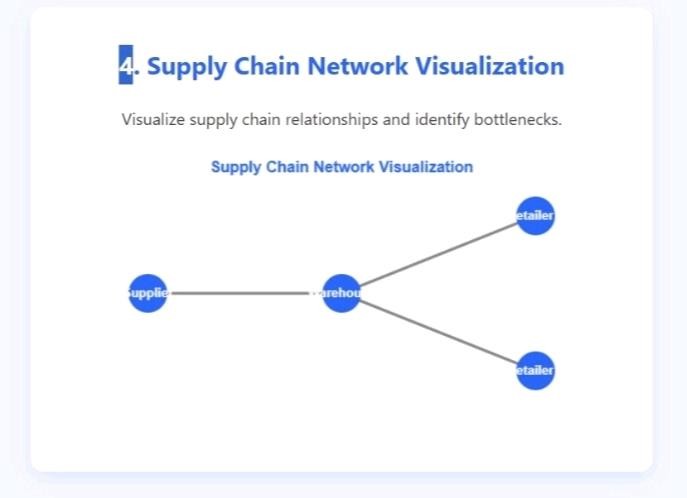
**Description**: This bar chart displays the optimized inventory levels for five products based on forecasted demand.

**Purpose:** It helps ensure stock availability while minimizing excess inventory, leading to efficient supply chain operations.



**Description:** This line chart shows monthly sales demand trends based on historical data to forecast future demand.

**Purpose:** It supports proactive planning in the supply chain by predicting upcoming product needs accurately.



**Description**: This diagram illustrates the flow of goods from suppliers to a central warehouse and then to multiple retailers.

**Purpose**: It helps visualize supply chain connectivity and pinpoint areas of potential delays or inefficiencies.

**12. Final Deliverables**

**1. Project Report (Documentation)**

**Executive Summary**

**Problem Statement and Objectives**

**Methodology (Data Collection, Cleaning, SQL Queries for Forecasting & Optimization) Tools and Technologies Used (e.g., SQL Server, PostgreSQL)**

**Results and Insights Recommendations for Business Impact**

ctx.fillText('Delivery Time (hrs)', paddingLeft + w\*0.5 + 20, 52);

})();

// Topic 4: Network Visualization (Nodes and Edges) (function drawNetworkChart() {

const canvas = document.getElementById('networkChart'); const ctx = canvas.getContext('2d');

ctx.clearRect(0, 0, canvas.width, canvas.height);

// Title

ctx.fillStyle = '#2a66f7'; ctx.font = 'bold 16px Arial'; ctx.textAlign = 'center';

ctx.fillText('Supply Chain Network Visualization', canvas.width / 2, 25);

// Draw edges ctx.strokeStyle = '#888';

ctx.lineWidth = 3;

for(let edge of networkEdges) {

const fromNode = networkNodes[edge.from]; const toNode = networkNodes[edge.to]; ctx.beginPath();

ctx.moveTo(fromNode.x, fromNode.y); ctx.lineTo(toNode.x, toNode.y); ctx.stroke();

// Arrowhead

const angle = Math.atan2(toNode.y - fromNode.y, toNode.x - fromNode.x); const arrowLength = 10;

const arrowAngle = Math.PI / 7;

let arrowX = toNode.x - arrowLength \* Math.cos(angle - arrowAngle); let arrowY = toNode.y - arrowLength \* Math.sin(angle - arrowAngle); ctx.beginPath();

ctx.moveTo(toNode.x, toNode.y); ctx.lineTo(arrowX, arrowY); ctx.stroke();

arrowX = toNode.x - arrowLength \* Math.cos(angle + arrowAngle);