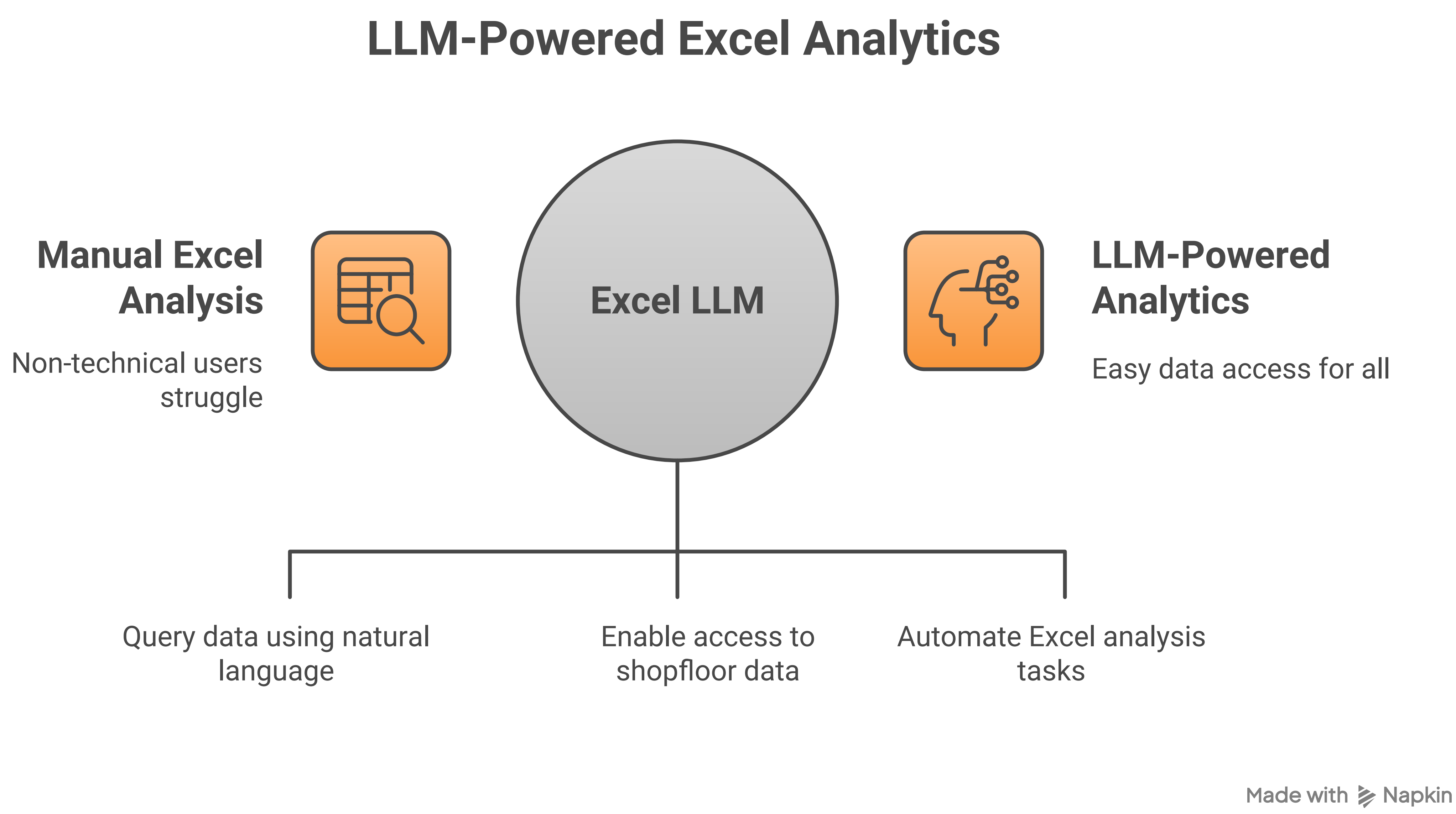
# Excel LLM

## Project Overview

I am building an **LLM-powered Excel analytics assistant** specifically for MSME manufacturing shopfloor data. The goal is to enable non-technical users to query production, quality, maintenance, and inventory data using natural language instead of manual Excel analysis.




## Key Requirements I've Noted

**1. Model Approach**

- Use open-source SLMs (Llama 2, Mistral, or Falcon)
- Fine-tune using LoRA/PEFT for efficiency
- Domain-specific training on manufacturing/shopfloor terminology

**2. Core Functionality**

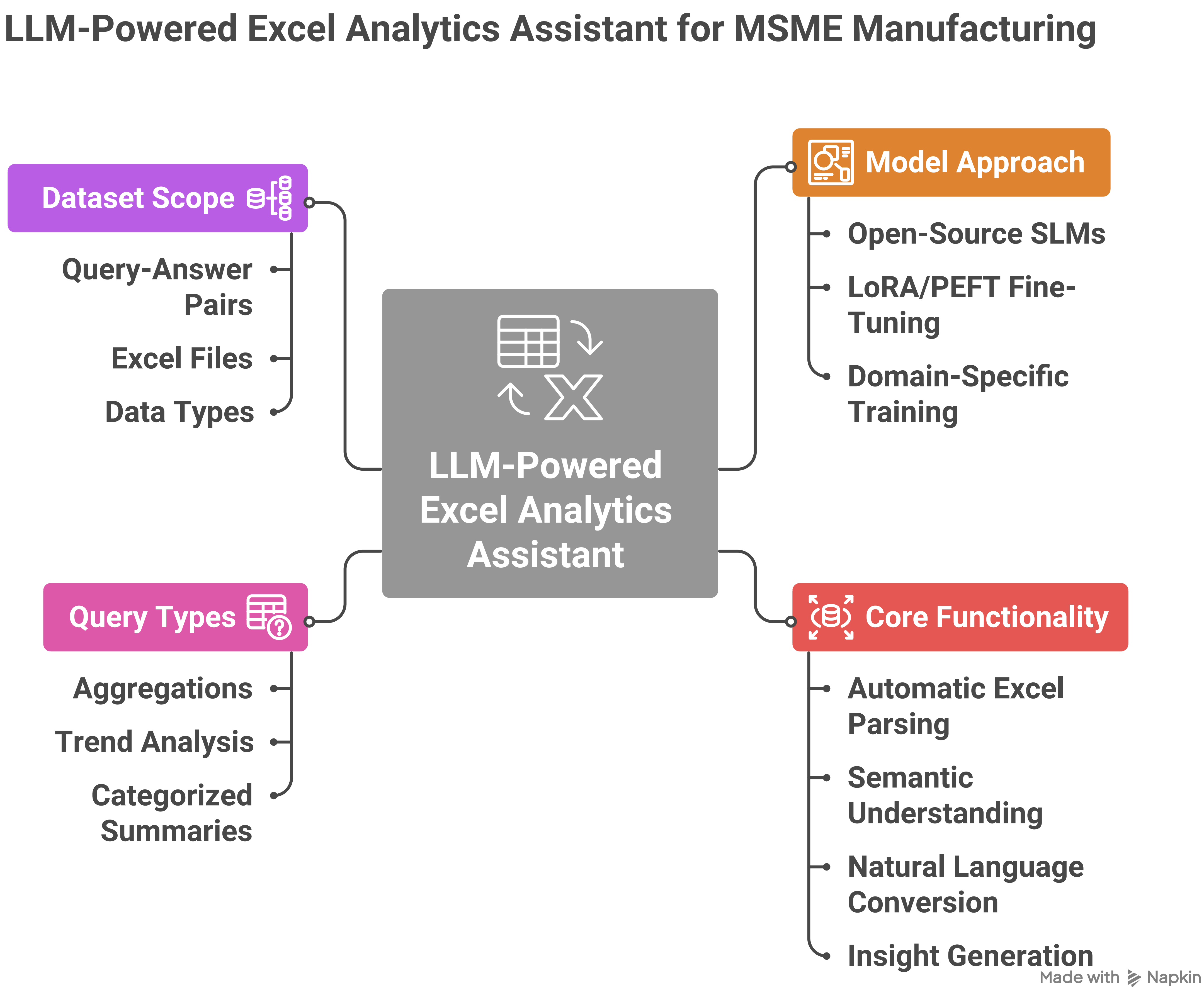
- Parse and normalize diverse Excel formats automatically
- Semantic understanding of manufacturing data structures
- Convert natural language queries to data operations
- Generate both textual insights AND visualizations

**3. Query Types to Handle**

- Aggregations ("most rework this quarter")
- Trend analysis ("daily efficiency trends")
- Categorized summaries ("rejected batches by defect type")

**4. Dataset Scope**

- ~2,000 query-answer pairs
- 50-100 Excel files
- Multiple data types: production logs, QC reports, inventory, maintenance




## Questions for Clarification

Before we dive deeper, I'd like to understand:

1. **Data Heterogeneity**: How varied are the Excel structures? (Different column names, formats, or relatively standardized templates?)
2. **Query Complexity**: Should the model handle multi-step reasoning (e.g., "Compare rework rates between products A and B, then show which defect types are common to both")?
3. **Deployment Context**: Will this run locally on MSME machines, or cloud-based? (Important for model size selection)
4. **Visual Output**: What chart types are priorities? (Line charts for trends, bar charts for comparisons, pie charts for distributions?)
5. **Real-time vs Batch**: Should this process queries in real-time or is batch processing acceptable?

## Understanding MSME Shopfloor Operations

### 1. Core Production Areas to Model

**Production/Manufacturing**

- What gets tracked: Output quantities, cycle times, machine utilization, shift performance
- Why it matters: Efficiency, capacity planning, bottleneck identification
- Typical questions: "Are we meeting targets?" "Which shift performs best?" "Where are delays happening?"

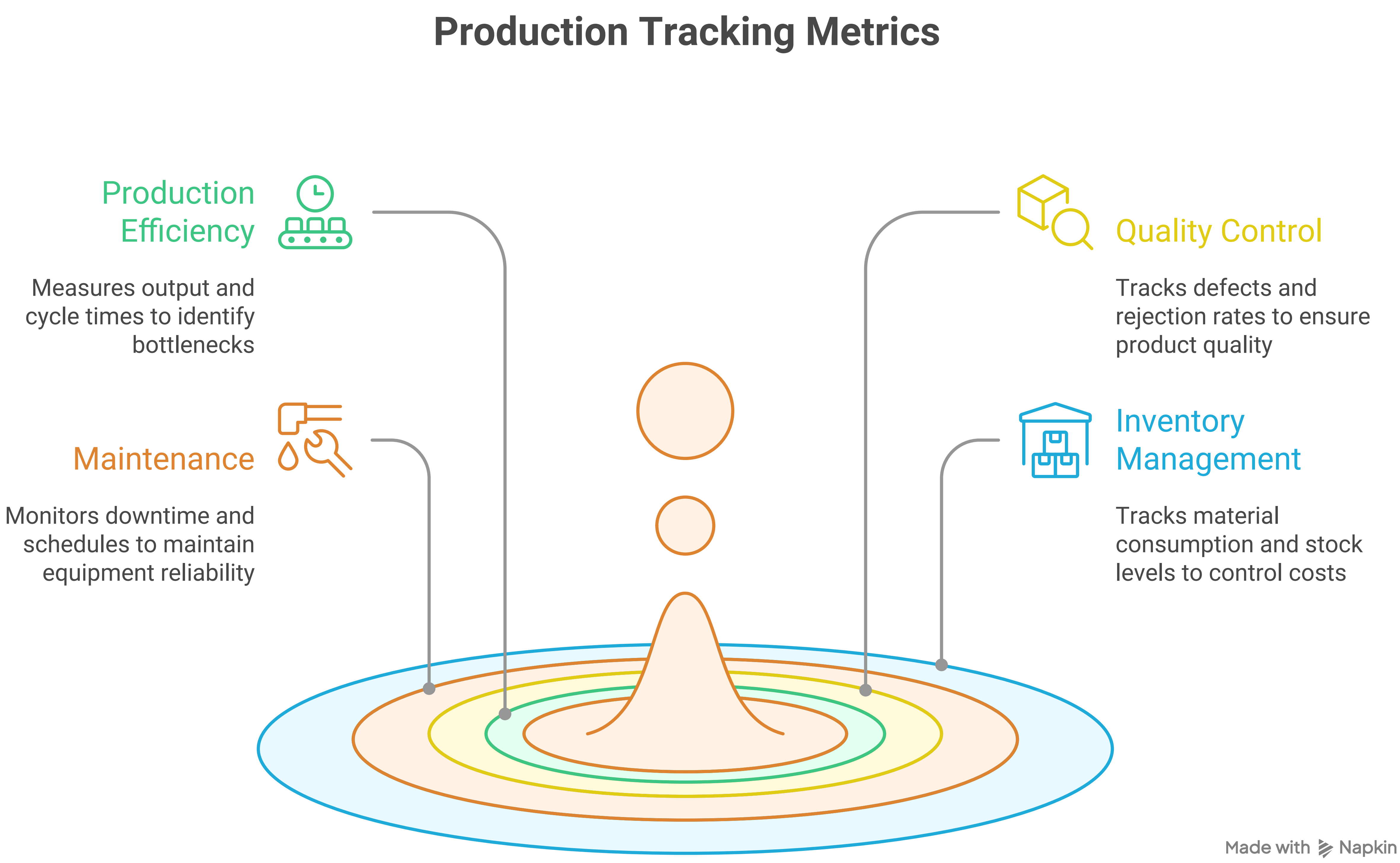
**Quality Control (QC)**

- What gets tracked: Inspection results, defect types, rejection rates, rework counts
- Why it matters: Product quality, cost of poor quality, compliance
- Typical questions: "What defects are most common?" "Which batches failed?" "Is quality improving?"

- What gets tracked: Machine downtime, breakdown frequency, preventive maintenance schedules
- Why it matters: Equipment reliability, production continuity
- Typical questions: "Which machines break down most?" "Are we maintaining on schedule?"

**Inventory/Materials**

- What gets tracked: Raw material consumption, stock levels, material wastage
- Why it matters: Cost control, preventing stockouts
- Typical questions: "Are we running low on materials?" "What's our wastage rate?"




## 2. Key Manufacturing Metrics (KPIs)

Before we create training data, we need to know what MSMEs actually measure:

**Efficiency Metrics**

- OEE (Overall Equipment Effectiveness) = Availability × Performance × Quality
- Production per shift/hour
- Cycle time vs. target time

**Quality Metrics**

- First Pass Yield (FPY): % of products passing without rework
- Defect rate per 1000 units (PPM - parts per million)
- Rework percentage

**Operational Metrics**

- Downtime hours
- Changeover time (switching between products)
- Resource utilization rates

## 3. Typical Excel Data Structures in MSMEs

Let me describe what these Excel sheets usually look like:

**Production Log Example Structure:**

```
Date | Shift | Line/Machine | Product | Target Qty | Actual Qty | Downtime
(min) | Operator
```

```
Batch ID | Product | Inspection Date | Inspected Qty | Passed | Failed | Defect
Type | Inspector
```

**Maintenance Log:**

```
Machine ID | Date | Issue Type | Downtime (hrs) | Repair Cost | Maintenance
Type (Breakdown/Preventive)
```

### 4. Industry-Specific Challenges

Understanding these helps us design better queries and responses:

**Data Quality Issues:**

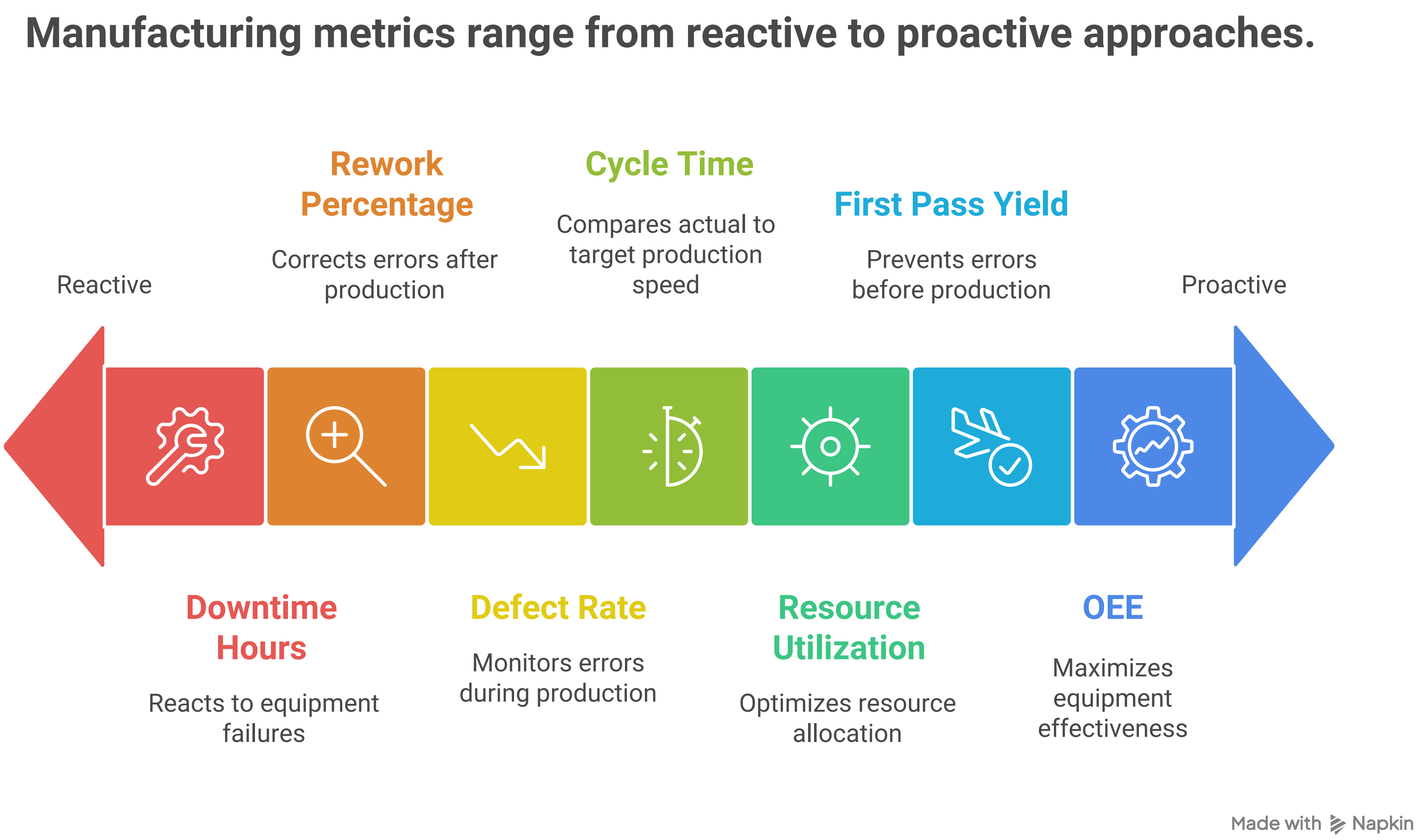
- Inconsistent naming (Product-A vs ProductA vs Prod_A)
- Missing entries
- Manual data entry errors
- Date format variations

**Domain Language:**

- Shopfloor terminology (rework, scrap, batch, lot, SKU)
- Abbreviations (FG = Finished Goods, WIP = Work In Progress)
- Machine-specific codes

**Decision Context:**

- MSMEs need quick, actionable insights (not complex statistical models)
- Visual dashboards matter more than raw numbers
- Comparison over time is crucial (this month vs last month)




---

**Refined Roadmap**

### Phase 1: Industry Selection & Intelligent Data Generation

Step 1.1: Choose Industry

- **Decision Point**: Walmart Retail/Logistics OR Manufacturing MSME?
- **Output**: Industry context document (terminology, KPIs, relationships)

**Step 1.2: Smart Data Generator (Gemini-Powered)**

**Architecture:**

```
Previous Data State → Gemini API → Generate New Related Data → Append to Excel/CSV
```

**Key Features :**

- **Stateful Generation**: Each new batch references previous data
- **Relationship Preservation**:
  - Customer "Shivam" who bought Product A → next time buys complementary Product B
  - Machine that broke down → needs maintenance → shows reduced efficiency
  - Seasonal patterns (Q4 sales spike)
- **Realistic Variations**: Not random, but story-driven

**Implementation Plan:**

- **Script**: data_generator.py
- **Input**: Industry type, row count, existing CSV (if continuing)
- **Output**: Incremental CSV files with relationship metadata
- **Gemini Call**: Batch generation (50-100 rows per call to maintain context)

## Phase 2: Intelligent Question Generator

### Step 2.1: Question Generation System

**Purpose**: Auto-generate diverse, realistic queries for training

**Features:**

- **Auto-answer generation**: Use Gemini to also generate ground truth answers
- **Metrics extraction**: Identify which KPIs each question targets
- **Benchmark categories**: Group questions by complexity for evaluation

## Phase 3: LLM Selection & Fine-Tuning Pipeline

### Step 3.1: Model Evaluation Matrix

**Benchmark Tests Before Fine-Tuning:**

| Model | Size | Speed | Zero-Shot Accuracy | SQL Generation | Reasoning |
|---|---|---|---|---|---|
| Llama 3.2 3B | 3B | Fast | ? | ? | ? |
| Mistral 7B | 7B | Medium | ? | ? | ? |
| Phi-3 Mini | 3.8B | Fast | ? | ? | ? |

**Test on:**

- 50 sample questions (before fine-tuning)
- Measure: Accuracy, hallucination rate, response time

**Step 3.2: Fine-Tuning Strategy**

————————————————————————————————

## Phase 4: LangChain Multi-Tool Agent System

### Step 4.1: Tool Architecture

```
tools = [
    # 1. Data Retrieval Tool
    Tool(
        name="ExcelDataRetriever",
        func=retrieve_relevant_rows,
        description="Fetch relevant rows from Excel based on semantic search"
    ),

    # 2. Calculation Tool
    Tool(
        name="DataCalculator",
        func=perform_calculations,
        description="Perform aggregations, averages, sums, etc."
    ),

    # 3. Trend Analysis Tool
```

```
        name="TrendAnalyzer",
        func=analyze_trends,
        description="Identify patterns over time, seasonality, outliers"
    ),

    # 4. Comparison Tool
    Tool(
        name="ComparativeAnalyzer",
        func=compare_entities,
        description="Compare products, time periods, categories"
    ),

    # 5. Visualization Recommender
    Tool(
        name="ChartRecommender",
        func=suggest_visualization,
        description="Recommend best chart type based on data and query"
    ),

    # 6. SQL Generator (optional, for complex queries)
    Tool(
        name="SQLGenerator",
        func=generate_sql,
        description="Convert natural language to SQL for complex operations"
    )
]
```

**Step 4.2: ReAct Agent with Industry Context**

```
system_prompt = f"""
You are an expert {industry} data analyst assistant.

Available tools: {tool_descriptions}

When answering:
1. Break down complex queries into steps
2. Use appropriate tools in sequence
3. Validate data before calculating
4. Provide context with industry benchmarks
5. Recommend visualizations

Industry Benchmarks:
{industry_kpis}

Respond in this format:
Thought: [reasoning]
Action: [tool_name]
Action Input: [tool_input]
Observation: [tool_output]
... (repeat as needed)
Final Answer: [comprehensive response with visualization spec]
"""
```

## Phase 5: Graph Generation System

**Step 5.1: LLM-Driven Visualization Specification**

**Step 5.2: Frontend Auto-Rendering**

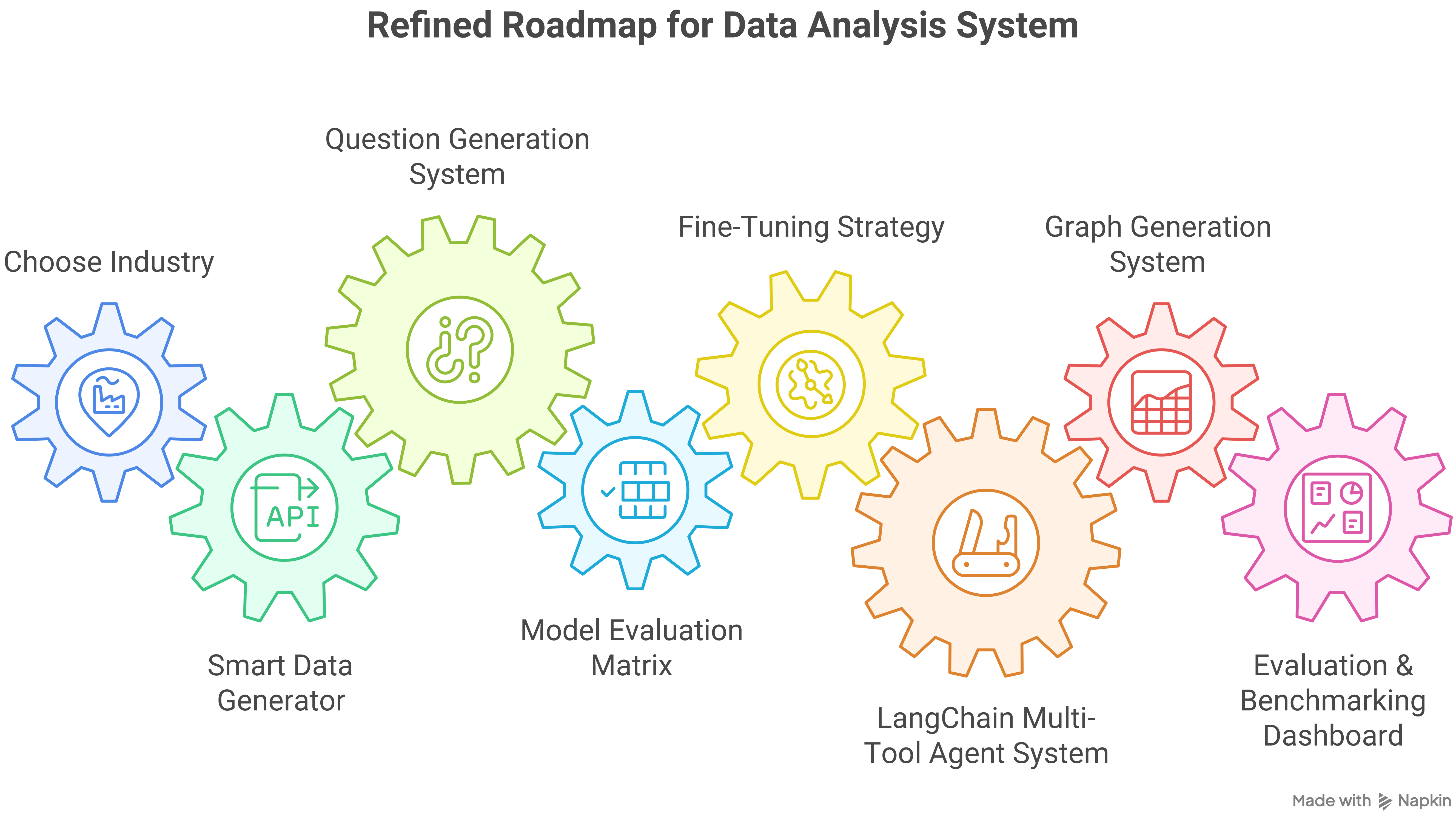
## Phase 6: Evaluation & Benchmarking Dashboard

**Step 6.1: Model Performance Metrics**

**Dashboard showing:**

- **Accuracy by Question Type** (simple vs complex)
- **Hallucination Rate**: Incorrect facts generated

- **Tool Usage Efficiency**: How often agent picks right tool first
- **Visualization Appropriateness**: Chart type matches data




## Revised Step-by-Step Execution Plan

### Foundation

- Choose industry (Walmart logistics recommended)
- Build data generator script with Gemini
- Generate initial 10,000 rows with relationships
- Build question generator script
- Generate 500 questions with ground truth

### Model Preparation

- Setup environment (Colab/Kaggle)
- Benchmark 3 base models (zero-shot)
- Select best performer
- Prepare fine-tuning dataset (questions + context + answers)
- Fine-tune with LoRA

### Embedding & Retrieval

- Build embedding pipeline
- Setup ChromaDB
- Test semantic retrieval accuracy
- Create metadata layer

### Agent System

- Implement 6 LangChain tools
- Build ReAct agent
- Test multi-step reasoning
- Create visualization recommender

### Web Application

- Setup Vite + React + Tailwind
- Build FastAPI backend
- Implement file upload & processing
- Create query interface
- Build chart auto-renderer

### Evaluation

- Build benchmarking dashboard
- Run 500 test queries
- Calculate all metrics
- Identify failure modes
- Iterate on prompts/tools

### Polish

- UI/UX improvements
- Add export functionality (PDF reports)
- Optimize performance
- Documentation
- Demo video

## Critical Improvements Needed

### 1. Missing: Data Preprocessing & Schema Normalization Layer

Roadmap jumps from data generation to embeddings, but MSMEs have messy Excel files. You need:

- **Schema detection & auto-mapping** (handling "Product_Name" vs "ProductName" vs "Prod")
- **Data cleaning pipeline** (null handling, date format standardization)
- **Column type inference** (is "123" a product code or quantity?)

**Add**: Phase 3.5 (between LLM selection and embeddings)

### 2. Missing: Ground Truth Generation for Evaluation

Mentioned "500 questions with ground truth" but don't detail HOW to generate accurate ground truth answers from synthetic data.

**Solution**: Use Gemini to generate query + execute pandas operations + verify = ground truth

### 3. Incomplete: Multi-Excel File Handling

Real MSMEs have 5-10 related Excel files (production, quality, inventory). Your roadmap doesn't clearly address:

- How to JOIN across multiple Excel files
- Which file contains what data (schema registry)
- How LLM knows which file to query

**Add**: Multi-file relationship mapping in Phase 1

### 4. Missing: Error Recovery & Fallback Mechanisms

What happens when:

- LLM generates invalid SQL?
- Visualization spec is malformed?
- User uploads corrupted Excel?

**Add**: Error handling strategy in each phase

### 5. Vague: "Fine-tuning with LoRA"

You need specifics:

- What training data format? (Instruction-tuning? QA pairs? SQL generation?)
- How many epochs? Validation strategy?
- How to handle domain-specific terminology (OEE, FPY, etc.)?

Shivam Tiwari
IIT Madras
DSAI