

## 1. Problem Statement

The pizza business is facing a shift in how customers place orders.

Traditional web and phone-based ordering systems are no longer sufficient because AI agents (ChatGPT, Copilot, etc.) are becoming the new interface layer.

However, existing pizza APIs are built using traditional REST/OpenAPI formats, which are not directly usable by AI agents.

Manually rewriting every API into Model Context Protocol (MCP) is slow, error-prone, and not scalable.

### The Core Problem

- REST APIs exist but AI agents cannot directly reason over them
- Manual MCP conversion does not scale
- Multiple agents need to coordinate, not just call APIs

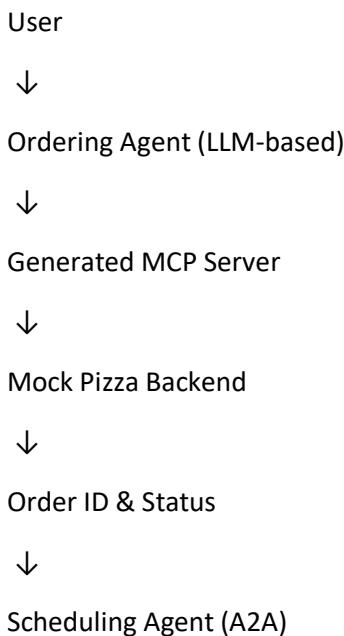
## 2. Solution Overview

This project solves the problem by building:

- An automated OpenAPI → MCP Server generator
- A Pizza Ordering AI Agent that interacts with the MCP server
- A Scheduling / Coordination Agent that communicates agent-to-agent (A2A)

Together, these components demonstrate how a traditional pizza API can become AI-ready with minimal manual effort.

## 3. Architecture Overview



## 4. OpenAPI to MCP Server Design

OpenAPI specification (pizza\_openapi.json)

Defines:

- Menu endpoint
- Order placement
- Order tracking
- Processing

The MCP generator:

- Parses OpenAPI paths
- Converts endpoints into MCP tools
- Adds structured context for AI agents

Automatically exposes:

- get\_menu
- place\_order
- track\_order

Output

A fully functional MCP-compliant server

AI agents can:

Discover tools

Understand required inputs

Execute actions without hardcoded logic

## 5. Ordering Agent Design

The Ordering Agent handles natural language input from the user.

Responsibilities

- Understand user intent (pizza type, toppings)
- Validate selections using MCP tools
- Place orders via the MCP server

Store and manage:

- order\_id
- pizza details
- order status

## 6. Scheduling / Coordination Agent (A2A)

Demonstrates agent-to-agent communication.

Responsibilities

- Receives order details from Ordering Agent
- Simulates interaction with an external MCP service
- Schedules delivery or confirms completion
- Sends updates back to the Ordering Agent

### Why A2A Matters

- Real-world AI systems rarely operate alone
- Demonstrates coordination, delegation, and autonomy
- Matches real enterprise AI workflows
- Scalability & Extensibility

### This architecture allows:

- Plugging in real REST APIs later
- Replacing mock backend with a database
- Adding new agents (payment, feedback, analytics)
- Reusing the OpenAPI → MCP generator for other domains

### Conclusion

This project proves that existing pizza APIs can be made AI-ready without rewriting everything manually.

By combining MCP, agent reasoning, and A2A communication, the system represents a future-proof approach to AI-driven commerce.