# API Integration of PartOrder System and Mechanic Supplies System

Table of Contents

[API Integration of PartOrder System and Mechanic Supplies System 1](#_Toc177942277)

[1.1 System Design and Component Breakdown 1](#_Toc177942278)

[1.1.1 Components: 1](#_Toc177942279)

[1.1.2 Technology Choices: 2](#_Toc177942280)

[1.2 . Requirements, Trade-offs, and Assumptions 2](#_Toc177942281)

[1.2.1 Requirements Met: 2](#_Toc177942282)

[1.2.2 Trade-offs Considered: 2](#_Toc177942283)

[1.2.3 Assumptions Made: 2](#_Toc177942284)

[1.3 Implementation Time Estimate 3](#_Toc177942285)

[1.3.1 Estimated Time: 3](#_Toc177942286)

## System Design and Component Breakdown

### Components:

OrderService: Handles the business logic for sending and receiving orders.

MechanicSuppliesClient: A client to communicate with the Mechanic Supplies API.

OrderRepository: Manages order data storage and retrieval.

Controllers: Expose endpoints for sending and receiving orders.

DTOs: Data Transfer Objects for transferring data between client and server.

Models: Data models representing the structure of the data.

Middleware: Custom middleware for error handling and logging.

Message Queue (Optional): Handles asynchronous communication and ensures reliable message delivery.

Technology: Azure Service Bus

Logging and Monitoring: Logs important events and monitors system health.

Technology: Serilog, Application Insights

### Technology Choices:

ASP.NET Core: For building the API.

HttpClient: For making HTTP requests to Mechanic Supplies.

Entity Framework Core: For data storage if persistence is required.

Swagger: For API documentation.

FluentValidation: For input validation.

Serilog: For logging.

## . Requirements, Trade-offs, and Assumptions

### Requirements Met:

Sending Orders: The system can send orders to Mechanic Supplies using the MechanicSuppliesClient.

Receiving Orders: The system can receive orders from Mechanic Supplies and process them using the OrderService.

### Trade-offs Considered:

In-Memory Storage vs. Database: Using in-memory storage for simplicity and speed during development. For production, a database like SQL Server or PostgreSQL would be more appropriate.

Synchronous vs. Asynchronous Communication: Using synchronous HTTP requests for simplicity. Asynchronous communication (e.g., using message queues) could be considered for better scalability and reliability.

### Assumptions Made:

Mechanic Supplies API: Assumed to have endpoints for sending and receiving orders.

Authentication: Assumed that the Mechanic Supplies API requires authentication (e.g., using API keys or OAuth).

Order Format: Assumed that the order format is compatible between the systems.

## Implementation Time Estimate

### Estimated Time:

OrderService and MechanicSuppliesClient: 2-3 days

Implementing the business logic for sending and receiving orders.

Creating the client to communicate with the Mechanic Supplies API.

Integration and Testing: 1-2 days

Integrating the components and ensuring they work together.

Writing unit and integration tests to verify functionality.

Documentation and Final Touches: 1 day

Adding Swagger documentation.

Final code review and polishing.

Total Estimated Time: 4-6 days

This design and breakdown should provide a clear understanding of how the system would be implemented and the time required

=============================== END ===========================