SYSTEM DESIGN FOR INTEGRATION WITH MECHANIC SUPPLIES

- This document outlines the design for integrating with the fictional "Mechanic Supplies" system to send and receive orders.
- The design is broken down into components, each with a specific responsibility, along with technology choices and trade-offs.
- Mechanic Supplies provides RESTful APIs for sending and receiving orders.
- The communication between our system and Mechanic Supplies is synchronous via HTTP for sending orders and asynchronous for receiving orders, using a message queue.
- Mechanic Supplies supports JSON as the data format for requests and responses.

1. System Components:

1.1 Order Service:

- ✓ Handles all order-related operations, including receiving orders from internal systems, sending them to Mechanic Supplies, and receiving responses.
- ✓ ASP.NET Core Web API, RabbitMQ for asynchronous messaging.
- ✓ Receives orders from the internal system or clients.
- ✓ Sends orders to Mechanic Supplies via a REST API.
- ✓ Processes responses from Mechanic Supplies and updates internal systems.

1.2 Mechanic Supplies Integration Service:

- ✓ Acts as an intermediary between the Order Service and Mechanic Supplies. Handles communication, error handling, and retries.
- ✓ Sends orders to Mechanic Supplies and handles responses.
- ✓ Receives orders from Mechanic Supplies via a message queue (RabbitMQ).

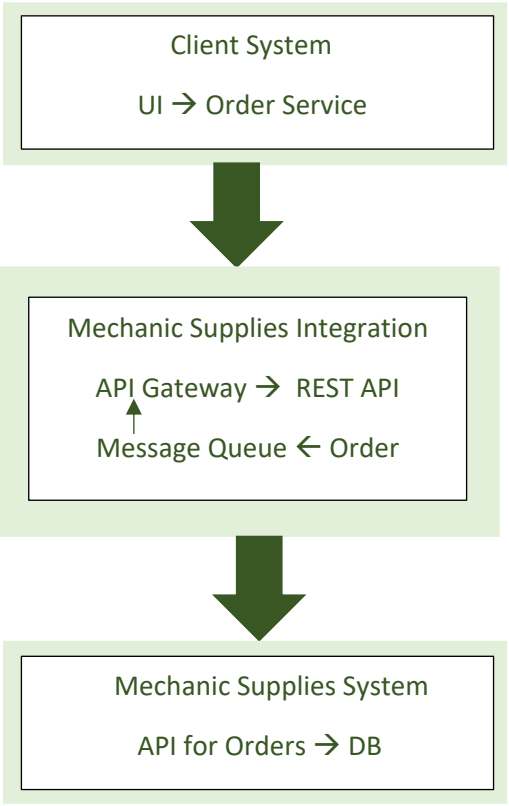
1.3 Message Queue:

- ✓ Facilitates asynchronous communication for receiving orders from Mechanic Supplies.
- ✓ Mechanic Supplies pushes orders to a queue.
- ✓ The Order Service consumes the queue to process incoming orders.

1.4 Monitoring and Logging:

- ✓ Monitors the health of the integration, logs errors, and provides alerts.
- ✓ Logs all HTTP requests and responses for auditing.

2. System Architecture Diagram:



3. Technology Choices:

- ✓ **ASP.NET Core:** Chosen for its robustness in building Web APIs and ease of integration with various services.
- ✓ **RabbitMQ:** Enables reliable asynchronous communication between systems, ensuring that orders from Mechanic Supplies are not lost.
- ✓ **SQL Server:** Relational database chosen for structured data storage, transaction support, and familiarity.
- ✓ **Serilog**: Structured logging that integrates well with ASP.NET Core, allowing for detailed logging.

4. Requirements Met:

- ✓ **Sending Orders:** Orders are sent to Mechanic Supplies using RESTful APIs.
- ✓ **Receiving Orders**: Orders from Mechanic Supplies are received through a message queue and processed asynchronously.
- ✓ **Error Handling**: Implemented using Polly, ensuring retries on transient errors and circuit breaking on repeated failures.
- ✓ **Logging and Monitoring**: Comprehensive logging and monitoring are implemented to track system health and troubleshoot issues.