INB374
Slightly Overpriced Appliances

Connor Livsey – n8510873
Joseph Maher – n8571520

Description

We have implemented a basic POS system for the Slightly Overpriced Appliances store.

The system allows for the addition of customers to the POS system, and for the ordering of various products.

Orders are assigned to customers via a unique customer number, assigned at the time of creation in the POS system.

The user of the system takes the roll of a sales person, interacting with the customer to ascertain their needs, in order to complete a sale. Small items are kept at the store which allow for the customer to take the item at the time of payment.
Larger items are kept at the warehouse, and as such are required to be delivered. Should an item be out of stock at the warehouse, the warehouse will make contact with the supplier in order to request new stock.

Should the customer accept the delivery time, the sale is processed and the delivery will be made at the pre-arranged date.

URL & Client Name

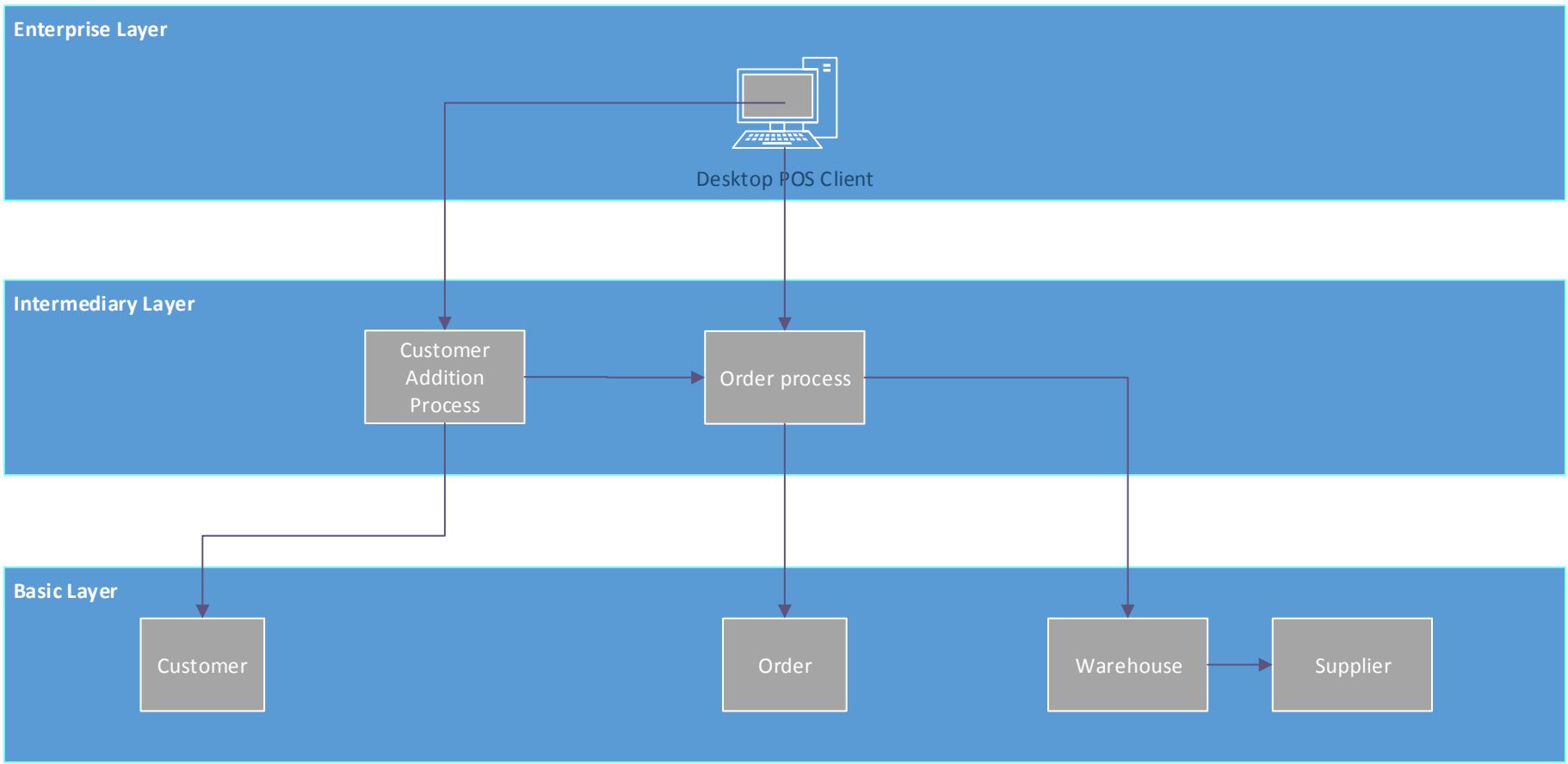
Service Repository:

http://fastapps04.qut.edu.au/n8571520/Repo/index.html

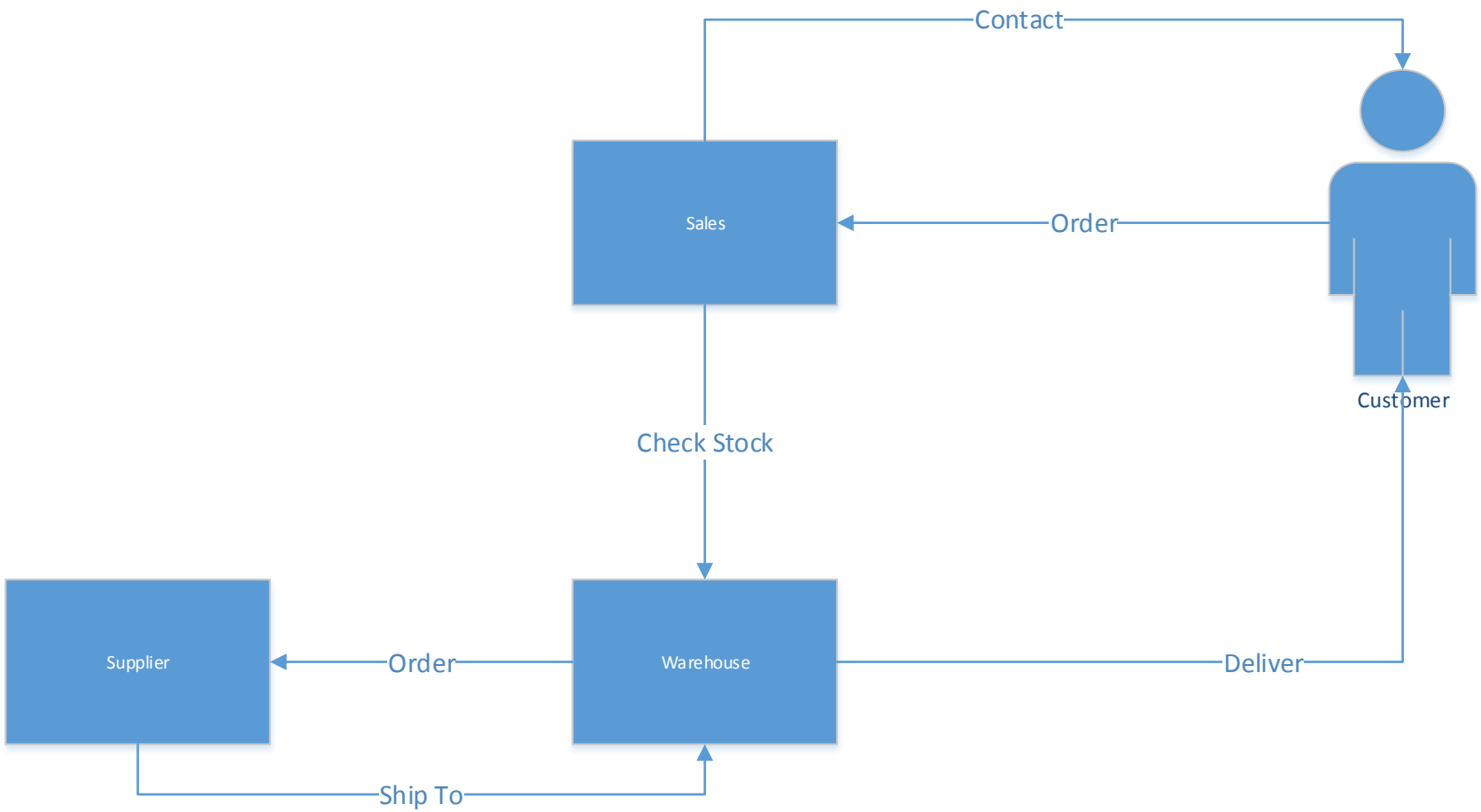
Client:

SOA.exe

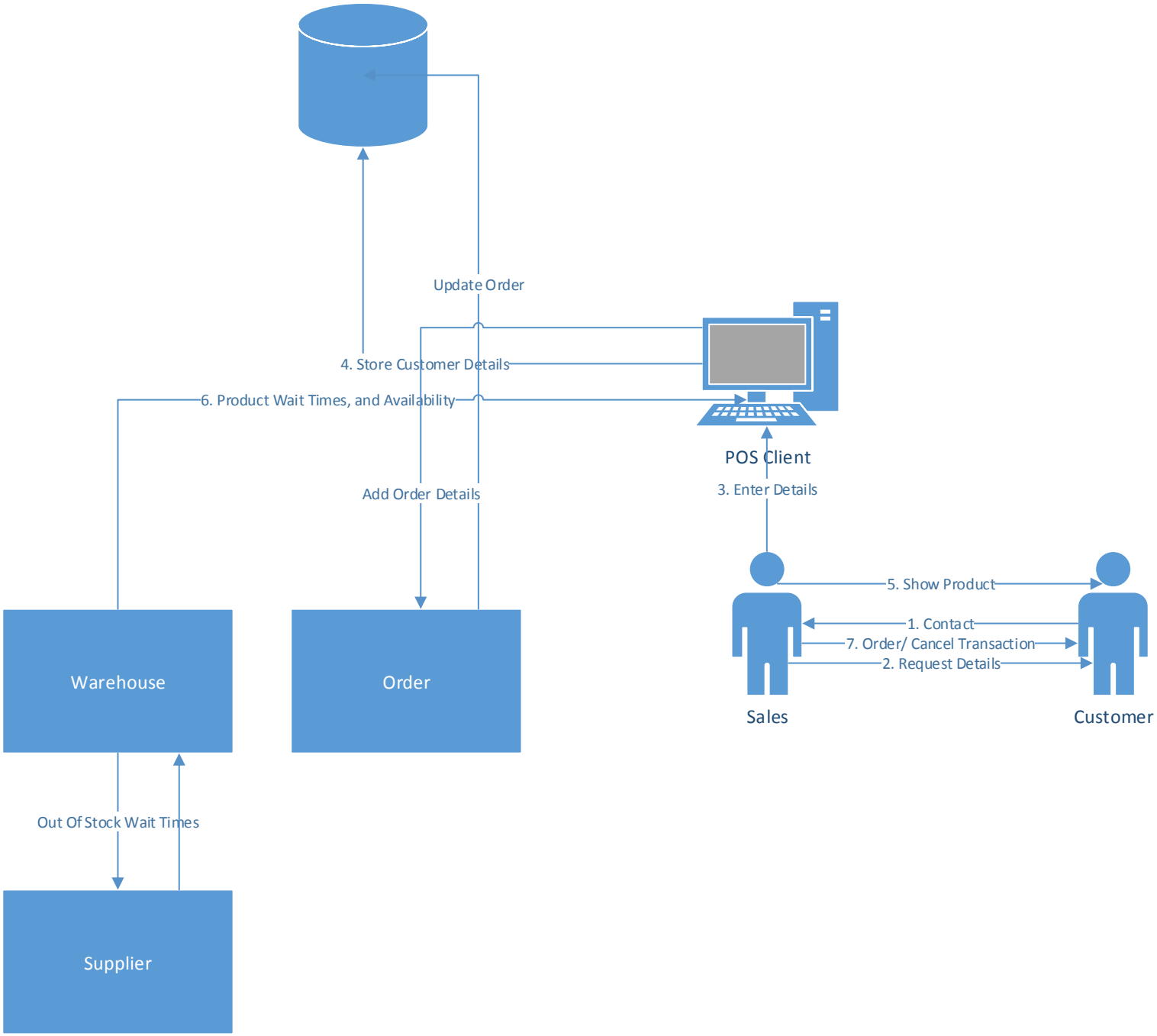
Slightly Overpriced Appliances
Logically Layered Architecture Diagram



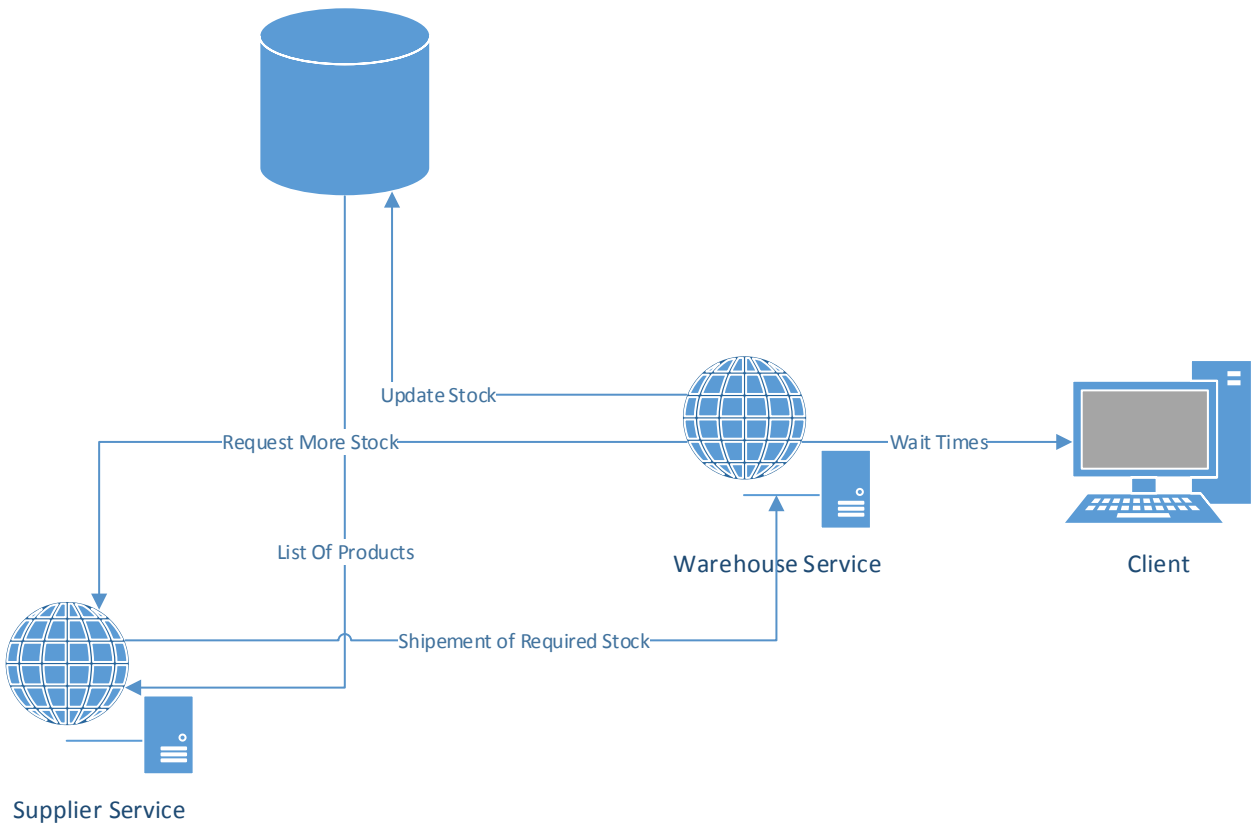
Slightly Overpriced Appliances
Level 0 Interaction Diagram



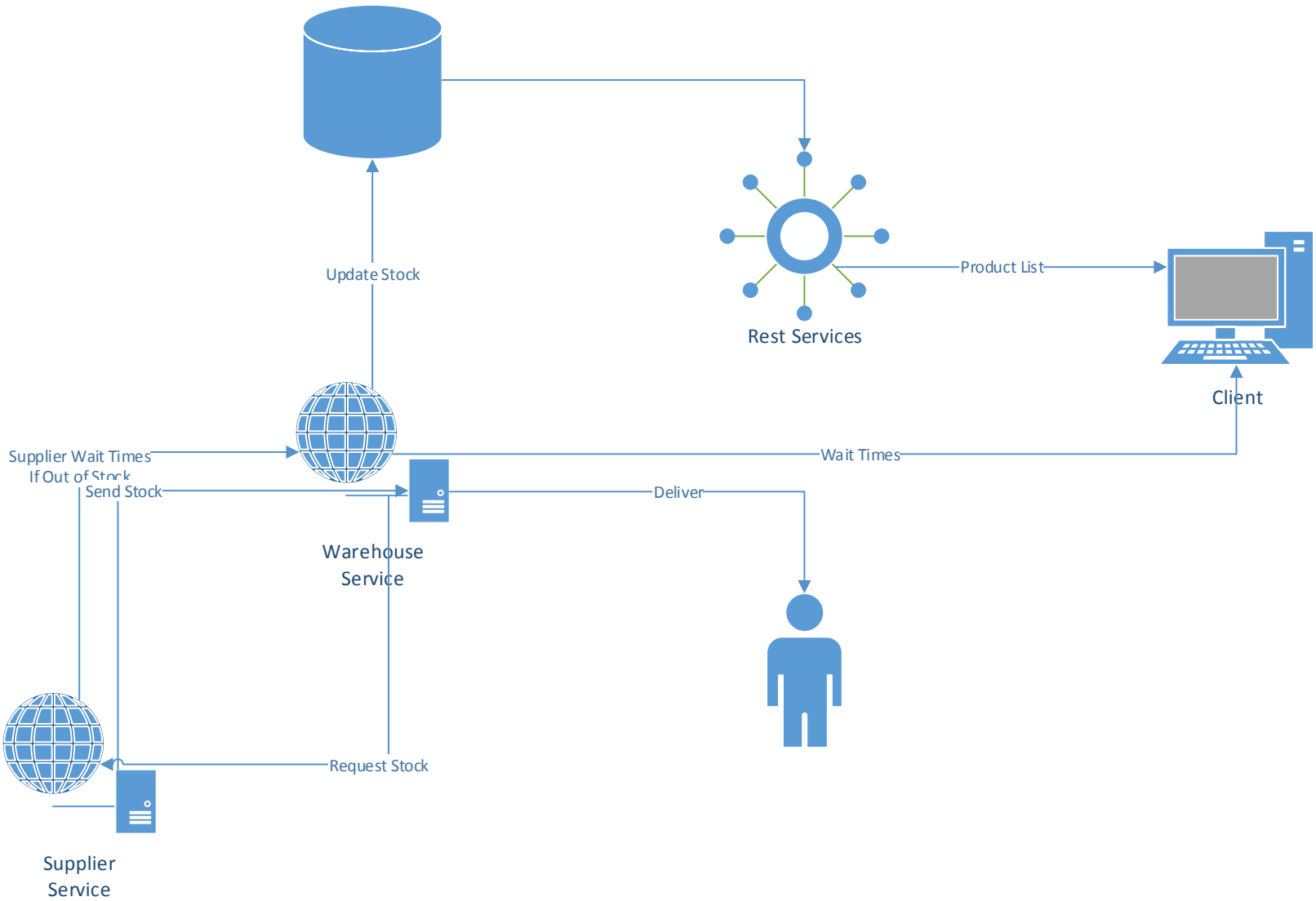
Slightly Overpriced Appliances
Sales Level 1 Interaction Diagram



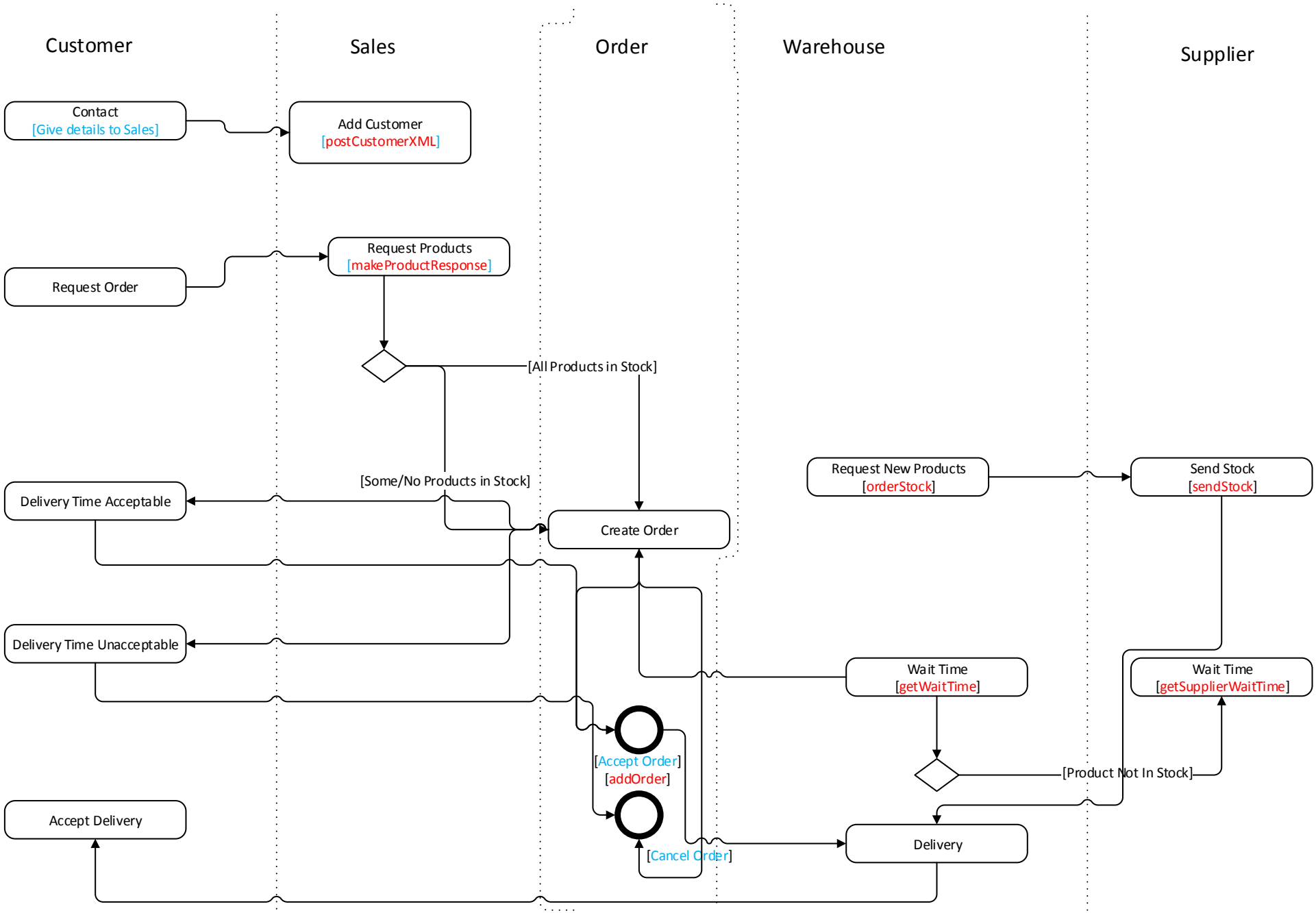
Slightly Overpriced Appliances
Supplier Level 1 Interaction Diagram



Slightly Overpriced Appliances
Warehouse Level 1 Interaction Diagram



Slightly Overpriced Appliances Choreography Model



SOA Analysis

A top-down approach was used throughout the assessment as we were designing an application from scratch and had no existing application logic to work with. As the application was being assessed from a black box perspective we were more concerned about focusing on what the business problem is about, not the how of the implementation. A bottom-down approach was considered due to the business already having a tried and proven business process as well as already being broken down into individual entities; however it was decided that the top-down approach was more suited to our needs.

The first step of the approach was to define the services with the application acting as the sole virtual service and the core services being; Order, Supplier, Warehouse, Product and Customer. This was followed by defining the external drivers who would be interacting with the services. In this case there was only one external driver, the sales team member taking input from the customer.

The final step was to define the interactions of the various services and the sole actor:

- Customer :
    - Interacts with a sales team member who is driving the application.
- Application:
    - Gets list of products.
    - Gets list of all registered customers.
    - Adds new customers.
    - Gets delivery time of products from warehouse
    - Creates customer orders
- Order:
    - Updates product stock levels
    - Modifies order status of orders that have been shipped
- Supplier:
    - Restocks the warehouse
- Warehouse:
    - Gets delivery time from supplier to warehouse
    - Automatically orders stock from supplier when needed.

When undertaking the task the following assumptions were observed:

- Delivery is automated and always on schedule, the shipping date agreed upon is the date the products get shipped.
- Ordering stock is an automated process that is triggered every time an order is placed and requires no manual input.
- The customer pays in full outside of the application when an order is confirmed

Service Orientation

Customer

- Data-centric
- Implemented as a data-centric service as it is only required to manipulate a database table.
- Loose coupling, does not have any dependencies.
- Application is aware of the service but the service doesn't know about the application.
- Business orientated, service is aligned with the business process and entities presented in the initial specification.

Product

- Data-centric
- Implemented as a data-centric service as it is only required to read a database table.
- Loose coupling, does not have any dependencies.
- Application is aware of the service but the service doesn't know about the application.
- Business orientated, service is aligned with the business process and entities presented in the initial specification.

Order

- Data-centric
- Logic-centric
- Implemented as a data-centric service as it is required to manipulate a database table.
- Some logic-centric elements as it is required to modify stock levels based on quantities of stock ordered and maintain the shipping of all existing orders.
- Relatively loose coupling as it relies on an product database to be reachable.
- Application is aware of the service but the service doesn't know about the application.
- Business orientated, service is aligned with the business process and entities presented in the initial specification.

Supplier

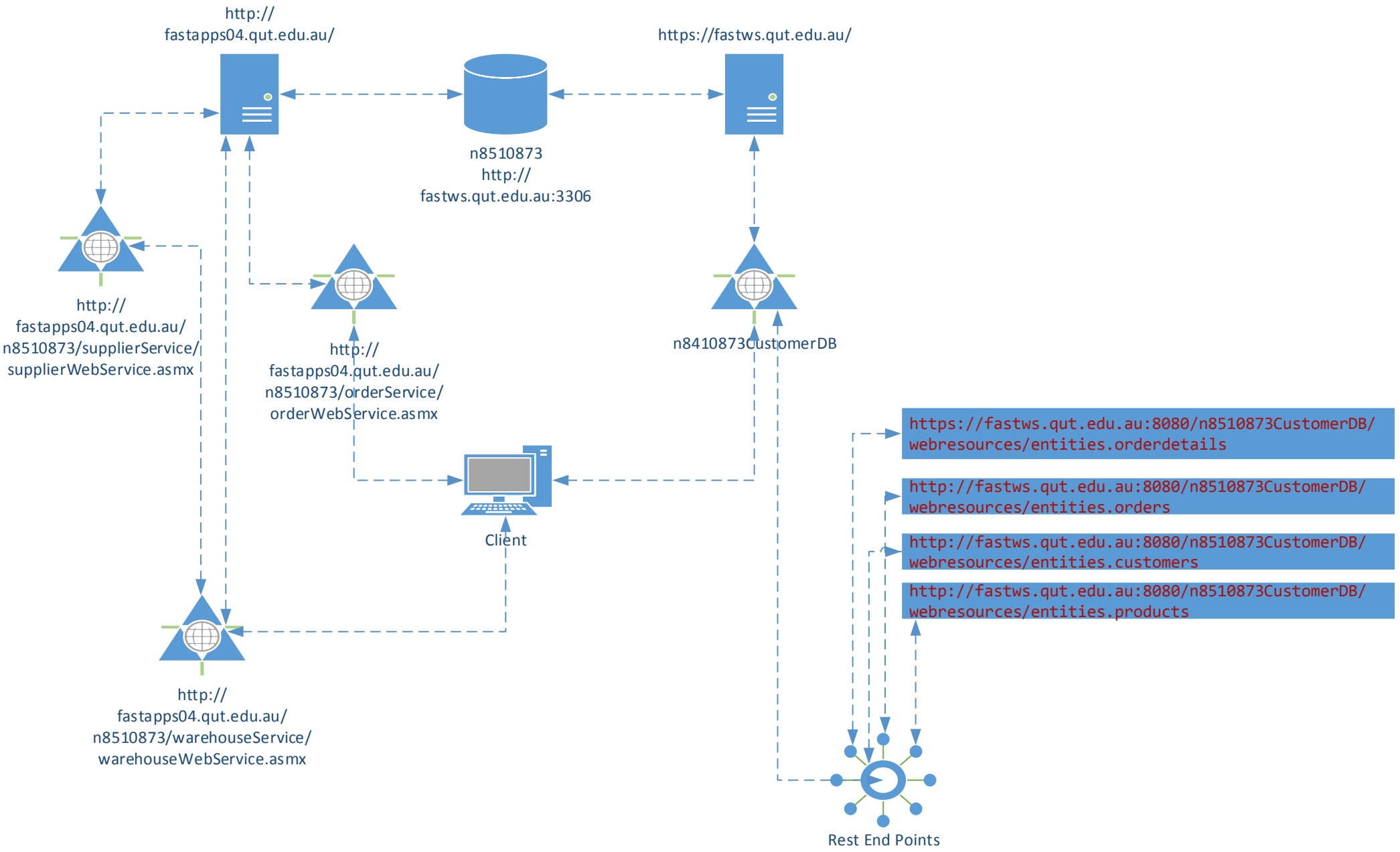
- Data-centric
- Logic-centric
- Implemented as a data-centric service as it is required to read a database table.
- Some logic-centric elements as it is required to restock the warehouse.

- Relatively loose coupling as it communicates with the warehouse.
- Application is aware of the service but the service doesn't know about the application.
- Business orientated, service is aligned with the business process and entities presented in the initial specification.

Warehouse

- Data-centric
- Logic-centric
- Implemented as a data-centric service as it is required to read a database table.
- A few logic-centric elements as it is required to order stock from the supplier and send stock to the warehouse
- Not so loose coupling as it is required to order stock from the supplier and send stock to the store
- Application is aware of the service but the service doesn't know about the application.
- Business orientated, service is aligned with the business process and entities presented in the initial specification.

Slightly Overpriced Appliances
Deployment Diagram



Name	Sub	Language	API	Complexity	Author
SOA		C#	Windows Form	595	Connor | Joe
	Constants.cs	C#		31	
	Customer.cs	C#		41	
	CustomerRestController	C#		126	
	OrderController.cs	C#		57	
	Product.cs	C#		28	
	ProductRestController.cs	C#		73	
	SupplierControler	C#		16	
	WarehouseController	C#		24	
			Total Lines:	991	

n8510873Customer DB		Java EE	JAX-RS		Connor | Joe
	Customers.java	Java EE		216	
	CustomerFacadeREST.java	Java EE		95	
	Orderdetails.java	Java EE		136	
	OrderdetailsFacadeREST.java	Java EE		115	
	OrderdetailsPK.java	Java EE		83	
	Orders.java	Java EE		169	
	OrdersFacadeREST.java	Java EE		90	
	Products.java	Java EE		209	
	ProductsFacadeREST.java	Java EE		90	
			Total Lines	1203	

orderService	orderWebService.asmx.cs	C#	ASP.NET	135	Connor | Joe
	Order.cs	C#		22	
	OrderDetails.cs	C#		20	
			Total lines:	177	

supplierService	supplierWebService.asmx.cs	C#	ASP.NET	107	Connor | Joe
	OrderQueue.cs	C#		16	
	Product.cs	C#		30	
			Total Lines:	153	

warehouseService	warehouseWebService.asmx.cs	C#	ASP.NET	164	Connor | Joe
			Total Lines:	164	

Project Total Lines	2688