National College of Ireland

SMART WAREHOUSE APPLICATION

Distributed Systems
March 2023

Mohammad Asem Soufi
Student nr: x18191665

Contents

1 Introduction

The purpose of the application is to create a smart warehouse that can track stock and automate orders.
The system will consist of smart services that inter-communicate with each other via gRPC.
The warehouse will be equipped with services that monitor the stock levels and that can automate the ordering process.
The overall goal is to reduce the workload of the warehouse staff and to minimize errors in stock management.

2 Service 1: Stock Management Service (service StockManagement)

This service will be responsible for managing the stock levels of all products in the warehouse.

2.1 Methods

2.1.1 RPC Method 1 – Update Product Stock (Unary)

This method will allow the client to retrieve the current stock level of a particular product. The request will include the product code, and the response will include the current stock level as a number of available products.

```
rpc updateProductStock(UpdateProductStockRequest) returns (UpdateProductStockResponse) {}

message UpdateProductStockRequest {
    string productCode = 1;
    sint32 qty = 2;   // could be a negative number if decreasing quantity
}
message UpdateProductStockResponse {
    bool success = 1;
}
```

2.1.2 RPC Method 2 - Get Stock Level (Unary)

This method will allow the client to retrieve the current stock level of a particular product. The request will include the product code, and the response will include the current stock level as a number of available products.

```
rpc getStockLevel(StockLevelRequest) returns (StockLevelResponse) {}

message StockLevelRequest {
```

```
    string productCode = 1;
}
message StockLevelResponse {
    int32 quantity = 1;
}
```

2.1.3 RPC Method 3 - Update Stock Level (Bidirectional Stream)

This method will allow the client to update the stock level of multiple items at the same time. The client will send a stream of UpdateStockRequest messages, each containing the item code and the new stock level. The server will respond with a single UpdateStockResponse message indicating whether the updates were successful.

```
rpc updateStockLevel(stream UpdateProductStockRequest) returns (stream UpdateProductStockResponse) {}

// uses pre-existing messages
```

3 Service 2: Order Management Service (service OrderManagement)

This service will be responsible for managing the ordering process in the warehouse.

3.1 Methods

3.1.1 RPC Method 1 – Make Order (Unary)

This method will allow the client to create a new order. The request will include the item code and the quantity, and the response will include the order ID.

```
rpc makeOrder(MakeOrderRequest) returns (MakeOrderResponse) {}

message MakeOrderRequest {
    string productCode = 1;
    int32 quantity = 2;
}
message MakeOrderResponse {
    int32 orderNumber = 1;
}
```

3.1.2 RPC Method 2 – Cancel Order (Unary)

This method will allow the client to create a new order. The request will include the item code and the quantity, and the response will include the

order ID.

```
rpc cancelOrder(CancelOrderRequest) returns (CancelOrderResponse) {}

message CancelOrderRequest {
    int32 orderNumber = 1;
}

message CancelOrderResponse {
    bool success = 1;
}
```

3.1.3 RPC Method 3 – Get Order (Unary)

This method will allow the client to create a new order. The request will include the item code and the quantity, and the response will include the order ID.

```
rpc getOrder(GetOrderRequest) returns (GetOrderResponse) {}

message GetOrderRequest {
    int32 oredrNumber = 1;
}

message GetOrderResponse {
    string orderDetails = 1;
}
```

3.1.4 RPC Method 4 – Get All Orders (Server Stream)

This method will allow the client to request a list of all created orders, no variables needed. The server will respond with a stream of Order messages, each containing information about a single order.

```
rpc getAllOrders(GetAllOrdersRequest) returns (stream GetOrderResponse) {}

message GetAllOrdersRequest {}
```

4 Service 3: Employee Management Service (service EmployeeManagement)

This service will be responsible for managing employees in the warehouse.

4.1 Methods

4.1.1 RPC Method 1 – Add Employee (Unary)

This method will allow the client to assign an employee to the warehouse.

```
rpc addEmployee (AddEmployeeRequest) returns (AddEmployeeResponse) {}

message AddEmployeeRequest {
    int32 employeeNumber = 1;
}
message AddEmployeeResponse {
    bool success = 1;
}
```

4.1.2 RPC Method 2 – Get Employee Details (Unary)

This method will allow the client to get the details of a specific employee in the warehouse.

```
rpc getEmployee (GetEmployeeRequest) returns (GetEmployeeResponse) {}

message GetEmployeeRequest {
    int32 employeeNumber = 1;
}
message GetEmployeeResponse {
    string employeeDetails = 1;
}
```

4.1.3 RPC Method 3 – Add Employees (Client Stream)

This method will allow the client to assign multiple employees to the warehouse.

```
rpc addEmployee (stream AddEmployeeRequest) returns
(AddEmployeeResponse) {}

// uses pre-existing messages
```