# Exercise 01 for Formative Feedback

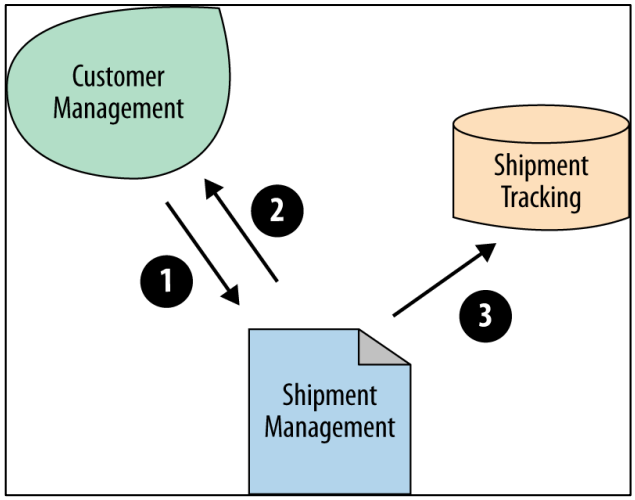
## Microservices development

This exercise is for formative feedback, not counting towards the final mark of the module. The exercise is similar to the individual component of the assessment, to get prepared for your submission of the assessment.

### Scenario

We will be using an imaginary startup. Let's assume that we are designing a microservice architecture for a fledgling shipment company, aptly named Shipping, Inc. As a parcel-delivery company, they need to accept packages, route them through various sorting warehouses (hops on the route), and eventually deliver to the destination. Shipping, Inc. is building native web applications for a variety of platforms to let customers track their packages all the way from pickup to final delivery. These applications will get the data and functionality they need from a set of microservices.

A high-level context map for Shipping, Inc.'s microservice architecture is illustrated in the figure below. This figure shows the key bounded contexts of the system.




The capabilities of the three contexts and some of the data flows between the contexts, depicted by the arrows and numbers on the graph are as follows:

1. **Customer Management** creates, edits, enables/disables customer accounts, and can provide a representation of a customer to any interested context.
2. **Shipment Management** is responsible for the entire lifecycle of a package from drop-off to final delivery. It emits events as the package moves through sorting and forwarding facilities, along the delivery route.
3. **Shipment Tracking** is a reporting application that allows end users to track their shipments.

## Development task

In this exercise, you will develop microservices based on current best practices in software development. The microservices should work independently.

You should create microservices of the three bounded contexts (customer management, shipment management, shipment tracking) fulfilling the functionalities mentioned above.

The developed solution should be structured as three separate microservices, main program, testing project and data folder. The Main program should simulate the Shipping Inc. system, running one instance of each microservice. The Testing project should include Unit Testing for all the microservices. The Data folder should include the database files used to permanently store the collected data with sample data stored.

The developed solution should fulfil the following technical requirements:

- Development in Java using any code editor or IDE (e.g. Eclipse).
- Implementing API Gateway for the communication between the microservices.
- Using a suitable database (e.g., MongoDB, SQLite) to store the collected data. Each microservice should have its own database.
- Implementing error and exception handling.
- Testing APIs to ensure functionality and reliability.
- Writing code documentation.

The developed solution should follow the following best practices:

- Solution and projects should be well-structured.
- Object-oriented principles should be used in designing the solution.
- Functions should be used.
- Code conventions should be followed.
- Professional code styling and formatting should be followed.

## Technical Report

The report should document the developed microservices-based solution, and should include:

- Introduction – (~ 100 words) provide a brief overview of the work that has been done, and the document contents that will be presented next to prepare the reader for the following sections.
- Architecture diagram – provide a diagram of the architecture of the developed solution, clarifying the communication between different microservices.
- Database Schema – provide the schema for the database used by the developed solution, in the form of clear diagrams, including tables, fields, and relationships.
- Class diagram – provide the design of the classes implemented in the developed solution, in the form of clear diagrams, including classes, interfaces, properties and functions.
- APIs documentation – list all APIs endpoints and their associated HTTP methods, e.g., GET, POST, PUT, DELETE, their Request and Response, and returned error codes. You can use screenshots of Postman.
- Critical evaluation – (~ 500 words) critically evaluate your product's inception to completion, i.e. what have been done well, what could have been done better and how.
- References – include references to relevant external documentation or resources.

The report should be 600 words. The approximate length defined above in each section is for guidance.

## Submission for feedback

The following files should be submitted:

- one zip file (this MUST be a zip file, NOT any alternative, such as rar file) containing a zipped copy of all the files of the development project (i.e. the whole running project including the database folder).
- one PDF file (this MUST be a PDF file, NOT any alternative, such as word file) of the technical report.

Files should be submitted using the link "**Formative feedback Ex.1**" under Assignment Submission in the Assessment and Submission Points on Blackboard of this module by 06th November 2024 4pm UK time.

## Notes

- You can attempt as much as you can, and submit it to receive feedback.
- The submission deadline is a soft deadline, so that you can receive feedback early enough before the submission of the assessment. Late submission is accepted, but means receiving late feedback.