**Mini Case: From Zoko to Amazing Bookstore**

## Background

Zoko, the retail company that sells books and other items, has been in operation for many years. Its IT applications have become large and complex, difficult to change and upgrade to face the competitive challenges from other online stores.

Its management and IT team have decided to experiment with a new solution based on **microservices architecture**, named as *Amazing Bookstore*, to bring agility to its business. As Zoko's IT team learned that microservices architecture encourages *loose coupling* among microservices, and different microservices can be developed and deployed independently, they decided to design a simple version of the solution based on a few basic microservices at first, with a plan to gradually refine those microservices, and gradually add more.

## Basic Business Requirements

Amazing Bookstore will be a web-based solution that provides a graphical user interface (UI) for users to browse books and place an order for a book.

Each and every order from users is stored permanently within a persistent data store when it is confirmed successfully. Whether an order is confirmed or not, the user is notified of its status, and the status is recorded as activity logs for future statistical analysis. When an order fails to confirm, an error handling functionality should be activated to handle and fix the error (e.g., check network connection, check book inventory, etc.). For simplicity in the initial implementation, the statistical analysis of activities logs and error handling functionalities can be replaced with simple printing.

When an order is confirmed successfully, the order is to be processed further (e.g., collecting the ordered books from the warehouse, packaging, etc.) for shipping to the user. For this purpose, an order should trigger the generation of a shipping record corresponding to the order, which is used for the shipping process. If the generation of the shipping record or the shipping process fails to complete for any reason, an error handling function should be activated to handle and fix the error (e.g., check book inventory, check delivery address, etc.). For simplicity in the initial implementation, the actual shipping record generation and the shipping process, and error handling functions can be replaced with simple printing.

An administrator of Amazing Bookstore shall be able to: view all the books in the bookstore's inventory, view all the orders, and add a new book into the inventory. A warehouse operator shall be able to: process a shipping record and update the status of the relevant order (e.g., "NEW", "PROCESSING", "SHIPPED"). A customer shall be able to: view all the books in the inventory too, place an order for a book, and view the status of an order. For simplicity in the initial implementation, we ignore reliability and security issues, such as user authentication and authorization, input validation, etc.

**Design Exercise**

Consider the following questions and use them to guide the design of a microservices architecture to fulfil the basic business requirements.

* What are the basic business processes (user scenarios) which implement the above requirements?
* What activities are there in the business processes that can be implemented by some mini-applications or microservices?
* What microservices (or functions) are needed to implement the activities involved in the business processes?

More specifically,

1. What kinds of basic data entities are necessary for the bookstore?
   1. How to implement each kind of data entities? What is its storage, data format, etc.?
2. What microservices are needed to provide the functions for the data entities?
   1. What are the CRUD operations needed for each kind of data entities that should be supported by which microservice?
   2. How to implement each microservice? What is its data store, programming language, platform, etc.?
   3. What are the interfaces for each microservice, including its transport, input/output data formats, etc.?
3. What may be the interactions needed among the microservices?
   1. What are the interaction needs? What is the data to be sent across microservices? How to send/receive data? How to handle errors (if any)?
4. What may be the interactions between users, UIs, and the microservices?

**Draw design diagrams** that can show the users, UIs, microservices, data stores, and the interactions needed. The diagrams should reflect the business processes (user scenarios); it'd be even better if they reflect the microservices architecture, and additional implementation details if *not* cluttered.

Note that:

* The above questions serve as a template to remind you of the factors that may be considered when designing a microservices architecture for a scenario.
* Specific answers are NOT expected for the questions at this initial design stage, especially for the implementation-level questions. Although more detailed, specific answers will be useful for actual implementation of the microservices in a later stage, they can be distracting for the initial design stage.
* During the initial design stage, please focus on identifying the needed microservice(s) and figuring out relevant interactions among them in the needed business processes (user scenarios).