Technische Universität München
Fakultät für Informatik

Exercise Sheet 2: Modularization

Welcome to the second week of Advanced Topics of Software Engineering. This week, we will practice the concepts of modularization and component-based system engineering learned from the lecture. Additionally, you will apply the concepts in identifying and designing the architecture of your system for the course project.

Theoretical Exercise

There is only one theoretical exercise for this week, which solely focuses on sharpening our foundational skills of modularization and modelling.

Exercise 1: Component-based Software Engineering

Consider the following system description:

ASE Fingerprint Scanner (AFS) is a biometric authentication system designed to help manufacturers identify staff and grant entry to authorized users. AFS consists of multiple biometric fingerprint analysis devices. Each device contains:

1. A fingerprint sensor to read the pattern of ridges in the user's finger.
2. A mini LCD screen to display the result of the authentication and authorization process.
3. A microcontroller to process the fingerprint and enforcing the authentication or authorization process. The microcontroller has an Ethernet network connection to the intranet of the factory.

All the devices retrieve the personal information and biometric profile of all the staff from a database. The database resides in a server that is accessible internally via a LAN connection. The devices shall use a secure communication channel to interact with the database.

Currently, AFS controls the entrance by identifying whether **the user is an employed factory worker, and is authorized to access an area**. As a result, employed workers can only access the areas that they are permitted based on their identity and role. The time and user identity of each successful or denied entry request are recorded in the database.

In the future, AFS can combine other biometric authentication techniques or include time in the access control, i.e., an authorized user can only access an area within the time of their working shift.

Task 1

Visualize the sub-systems in your architecture using the **Container Diagram (Level 2)** of the C4 Model.

Note: In C4 Model, the term Container is equivalent to **Component** in the lecture. Likewise, Component in C4 Model is equivalent to **Module**.

Explain your architecture rationale in the following aspects:

- Functionality
- Reduction of Complexity
- Maintainability and Reusability
- Evolvability

Task 2

Design the architecture of the system. Applying the decomposition techniques you learn from the lectures to identify modules in the system, and group them into components.

Select two containers from your system that communicate with each other. Use the C4 Component Diagrams (Level 3) to describe the modules and communication interfaces of each component. You can create your diagrams at https://app.diagrams.net/?libs=c4.

Practical Exercise

This week's practical exercises will help you better understand and model various facets of the *ASEDelivery* software. You are going to be utilizing the C4 Component Diagrams to visualize the different levels of abstraction of your software. By using C4 diagrams, you will have the opportunity to address the audience with diagrams that are *relevant/intended* for them.

For each of the following exercises, we are going to be showing how the same diagram can be created for an imaginary online food delivery service called *My Food Basket* (MFB). MFB is an online platform where users can order food from restaurants that are partners with MFB. The application provides an interface for users where they can see menus of the restaurants, add food to their basket, and finally checkout. The application also has an interface for the restaurants where they can log in and update their information in terms of food options, prices, and opening/closing hours. Besides users and restaurants, system admins also have an interface where they can manage users and restaurants. While users can sign up for the application by themselves, restaurants can only be registered by system admins.

You can use https://app.diagrams.net/?libs=c4 for your diagrams.

Exercise 2: System Context Diagram

Let's start with the System Context Diagram. This diagram gives you the ability to see the big picture for your software. It depicts how your software system interacts with people (actors, roles, personas, etc.) and other software systems. The focus is *not* on technologies or protocols, but rather on people and software systems. The System Context Diagram is intended for both technical and non-technical audiences. (*The System Context Diagram for My Food Basket application can be seen in Figure 1.*)

1. Identify the *people* and *software system(s)* in the ASEDelivery system and define their roles/functions.
2. Create a System Context Diagram, showing the relationships between the people and software systems you have defined. (Don't forget to annotate the relationships.)

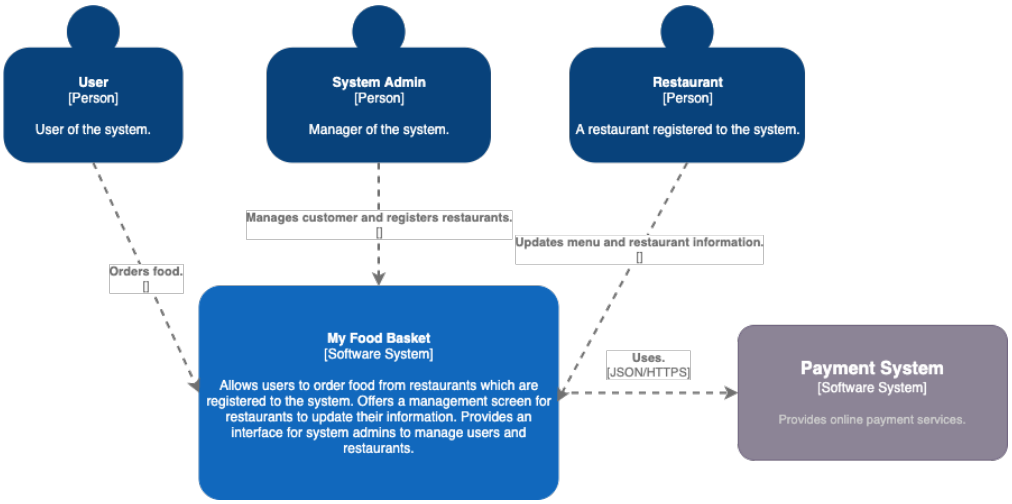

Figure 1: My Food Basket - System Context Diagram

Exercise 3: Container Diagram

The Container Diagram helps you to understand the high-level technical architecture design of your software system. In this diagram, a container can be anything that is deployable/runnable as an independent unit. A container is mainly responsible for executing code or storing data. Some example containers are: *desktop application*, *mobile application*, *web application*, and *database schema*. This diagram is mainly relevant for software developers and operations staff. (*The Container Diagram for My Food Basket application can be seen in Figure 2.*)

1. Identify the potential *containers* in the ASEDelivery system and define the role and technology of each.
2. Create a Container Diagram for the ASEDelivery system.

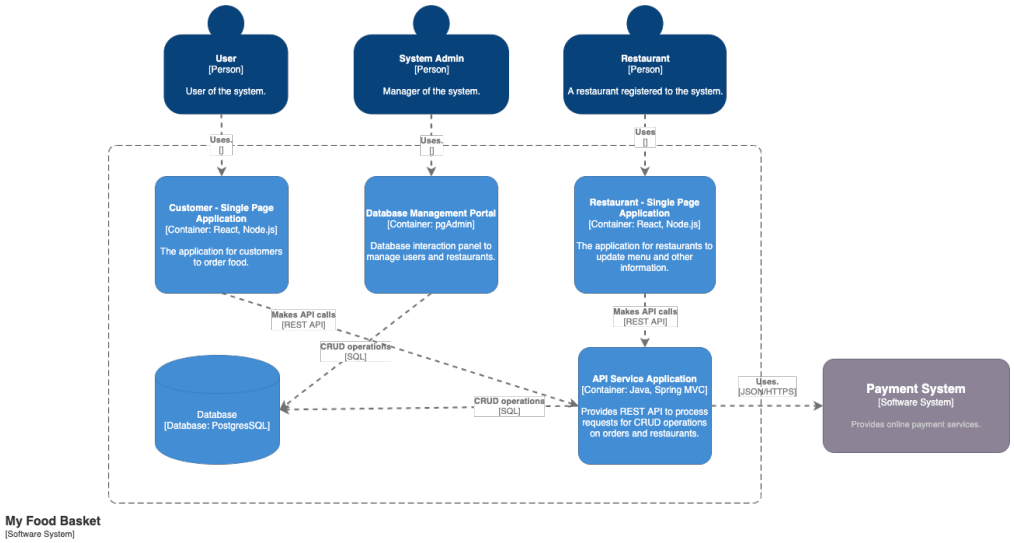

Figure 2: My Food Basket - Container Diagram

Exercise 4: Component Diagram

The Component Diagram zooms in on the containers from your Container Diagram. It helps you to understand the building blocks of a container and how they interact with each other and other systems. This diagram is intended for software developers and architects. (*The Component Diagram for My Food Basket-API Service Container can be seen in Figure 3.*)

1. Choose one of the containers from your ASEDelivery Container Diagram and decompose it into *components*. For each component, define its role and technology.
2. Create a Component Diagram for the container you have decomposed in the previous step.

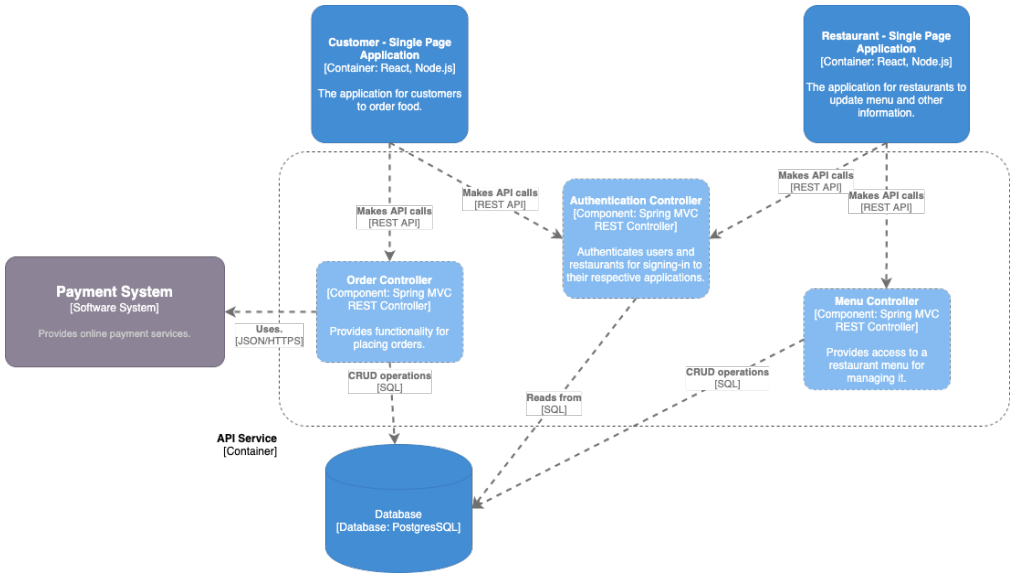

Figure 3: My Food Basket - Component Diagram for the API Service Container