Assignment 1, Task 1

SEM Group 26A

Tommy Hu
Andrei Drăgoi
George Iftode
Konstantinos Stergiou
Lennart Verstegen
Ansh Kumar

Bounded Contexts

The first step of this project was to identify the bounded contexts. To do this we first went through the scenario description. Once we had a grasp of the scenario we ran through multiple storytelling scenarios both from the perspective of HR and an employee. In order to derive the bounded contexts we applied some concepts from DDD. We created a list or requirements through the MoSCoW method. In the end we were able to identify 3 bounded contexts; users, contracts, and authentication

Users

Firstly, the Users bounded context. We identified this bonded context to be a core domain, as it plays a crucial role in the system representing both the HR and the employees functions. Within Users there are three major types or users; HR, existing employees, and candidates. Depending on the type of users they would be given access to different parts of the application. For instance, only HR would have access to a list of all existing employees and potential candidates, while candidates would only have access to their proposed contract.

Contract

Secondly, the Contract bounded context. Similarly to the users bounded context we identified the contract bounded context as a core domain, as it too plays a crucial role in the system, as it represents the contracts that are assigned to the employees. This bounded context's primary function is to allow certain users to edit their contracts. Furthermore, this bounded context should allow users to propose certain changes to their contracts, and once approved by HR should change the contract.

Authentication

Finally, the Authentication bounded context. In contrast to the previous bounded contexts, we identified the authentication bounded context to be a generic domain, as, even though it is not one of the most important parts of the system, the system still depends on the authentication process. This is due to the purpose of the authentication process. Not only does the authentication provide a level of security to the system it also redirects different users to their respective sites, where they can only access information that they have permission to. For instance existing employees won’t have access to each other's contracts, while HR does.

Relationships

Each of these bounded contexts also exhibit relationships between each other. The user bounded context has an upstream relationship with the contract, because, depending on the specific user and the user type, they would be able to access certain contracts and have specific permissions to edit these contracts. For instance an HR user would have access to multiple contracts and be able to propose edits to each of them, while an existing employee would only have access to their own. Therefore the contract context has a downstream relation to the user context. Furthermore the user context has a downstream relation to the authentication context, this is because the authentication context determines what type of user it is and therefore their permissions. Therefore the authentication context has an upstream relation to the user context.

Microservices

Once we had the bounded contexts identified the next step was to create the microservices that would be able to complete the operations that the application requires. For this we created five microservices; Authentication, User, Contract, Request, Notification.

Authentication

The authentication microservice is used to register new users and authenticate existing users. Each of the different microservice requires authentication in order to function properly. Upon using the authentication endpoint, a user is provided with an authentication token. This token provides access to the rest of the microservices' functionalities. Depending on the user type, the user will have access to different parts of the application.

User

The user microservices stores the users data, and allows the users to make requests, such as to view their contract. There are three types of users; HR, Employee, Candidate. Each different user has their own permissions and are able to perform different tasks. HR users can create draft contracts for candidate users and can agree on the changes proposed by candidate users. Furthermore HR users have access to multiple contracts at once. In contrast, the Candidate and Employee only have access to their own contracts. Candidate users can request modifications to the contract or agree with changes made by HR. Employee users can also

request to view their documents, such as their contract. All users also have access to notifications that they can all view to see updates on requests made.

Contract

The contract microservice stores contracts and additional information in an internal database. The contract microservice enables users to create new draft contracts for new candidates, modify contracts of existing candidates, update contracts of employees, and terminate contracts of denied candidates or discharged employees. The contract object consists of many fields that help identify the content of the contract, who it is for, additional benefits, and more.

Requests

The requests microservice handles communicating different requests between the microservices, The requests microservice acts as the middleman between the contract, users, and notification microservices. For instance if an employee needs to view their contract the request microservice sends a request to the contract microservice to retrieve the contract. The primary reason for this microservice is for modularity and scalability reasons. In case in the future the clients want to add additional features to the application they only need to edit the request microservice to ensure that they can all communicate with each other rather than having to essentially edit every microservice.

Notification

The final microservice is the notification microservice. This microservice allows the different users to be notified of any new changes to their existing or draft contracts. It also alerts users on whether their changes have been approved or denied. This microservice also primarily exists for modularity and scalability reasons, as right now the only notifications that users would receive would be regarding the contracts and information, in the future this microservice can also be used to notify users of any other features that would be implemented.

On the next page is an UML component diagram depicting each microservice and how they work together.

UML Component Diagram