SEM Assignment 1

Group 14B

November 2021

1 Task 1: Software Architecture

1.1 Bounded Contexts

The following domains will be mapped to micro-services in our system.

- **User**: A *core domain* that represents a user in the system. A user can be a teacher or a student. The type of user can be used to determine the functionalities provided and defines how they can interact with other domains in the system.
- **Course**: A *core domain* that contains all relevant information to a course. This domain defines which TA applications can be made and stores relevant course information that can be send to students in TA contracts.
- **Application**: A *core domain* manages all the information relevant to a TA application, connecting the Course and User domain. This domain groups the whole process of how a TA application is created to how a student will receive a work contract for their TA position.
- **Notification**: A *generic domain* that supports the process of a student receiving updates for their TA applications from our system.
- **Authorization**: A *generic domain* that is responsible for authenticating users when they try to access the system. Users are given different privileges depending on the role they are authenticated to.
- **Rating**: A *supporting domain* that allows lecturers to rate TAs. The rating system will allows lecturers to modify attributes for students who were TAs to their courses.

1.2 Bounded Contexts map Microservices

User

The **User** micro-service will contain a Student and Lecturer object to separate responsibilities of the two roles. For a student, the service will provide all basic user information (id, email) and grades for all taken courses, which would be used by the application service to check the qualifications of a student. Basic information about the lecturers are also stored. Moreover, the user service will track students who are TAs, so that we have an efficient way to check whether a student is a TA. Moreover, students can be updated by other services, but lecturers can not.

Motivation: It is possible to create two individual micro-services for the Student and Lecturer objects. However, it is preferable to merge these objects into one as they have many underlying attributes that are similar including NetID, email and password and have similar behaviour. Furthermore, the authentication endpoints for both Students and Lecturers are the same.

Course

The **Course** micro-service will maintain the lecturers for a course, which can be used to determine which lecturers have the rights to accept or reject TA applications.We assume that all basic information for a course to be predefined – meaning that no user in the system can modify the course code, course start date, and course lecturers. Each course will also maintain a start date, so we know when TA applications can no longer be accepted. Courses will also track the TA to student ratio, and inform lecturers on the ratio through the notification service.

Motivation: It is of value to establish a Course micro-service as there are many overlapping endpoints that can be grouped together in this object. This micro-service is responsible for sending course related data to other micro-services (number of free spots, TAs that are selected, ...).

Application

The **Application** micro-service will make sure that all relevant information is present in the TA application and that the user has the basic credentials, provided by user micro-service, to qualify as a TA. Lecturers are able to view applications and accept or reject applications. If an application is rejected or retracted, this information will be available to users upon request of their application status. Once an application is accepted, the micro-service is also responsible for generating a contract (PDF or JSON) that can be sent to a user through the notification micro-service. We chose to create an application service that will handle applications and contracts since all data provided by the user service for an application is sufficient to create a contract. Moreover, we assume that contract signing are done external to the system, thus the application service will only need to contact one other service to complete the full procedure.

Motivation: The Application is arguably the most critical micro-service in the entire architecture. It is of great necessity to store the state of the application, whether accepted, pending or denied, and all necessary information so as to not have to constantly retrieve it from the database and thereby consume loads of unnecessary data. It is responsible for recommending TAs for some specific courses.

Notification

The **Notification** is responsible for sending emails to students once an application has accepted. On request of the application service, the notification service will serialize the contract to a format (PDF or JSON) that will be emailed to the user specified by the contract. Sending emails to users of the system will be a functionality that is frequently requested. Thus for scalability of the system, this should be an independent micro-service that can potentially be replicated across nodes to balance the load.

Motivation: The notification micro-service serves as a layer between the Application and User micro-services, informing and updating the Student subclass, when its application state has changed or when its application status is being requested by the student themselves.

Authorization

The **Authorization** micro-service will be the entry point of any user into the system. The role a user authenticates to will determine the functionalities they have access to throughout the system. For our system, we assume that all valid NetIDs and passwords are predefined and users will not be able to create new accounts that can authenticate to the system. Once a user has authenticated, they will receive JWT token that can be used to authenticate a user when making requests to other micro-services in the system. When a request is made to any service in the systems, that system will need to verify the JWT token in the request with the authentication service. Since this will be frequently requested, this needs to be its own micro-service to prevent it from being the bottle neck in the system.

Motivation: This micro-service is responsible for assuring that all the operations inside our system are valid and it assures that the user cannot access information which he is not supposed to know. Every request through an endpoint will go through the Authorisation micro-service.

Rating

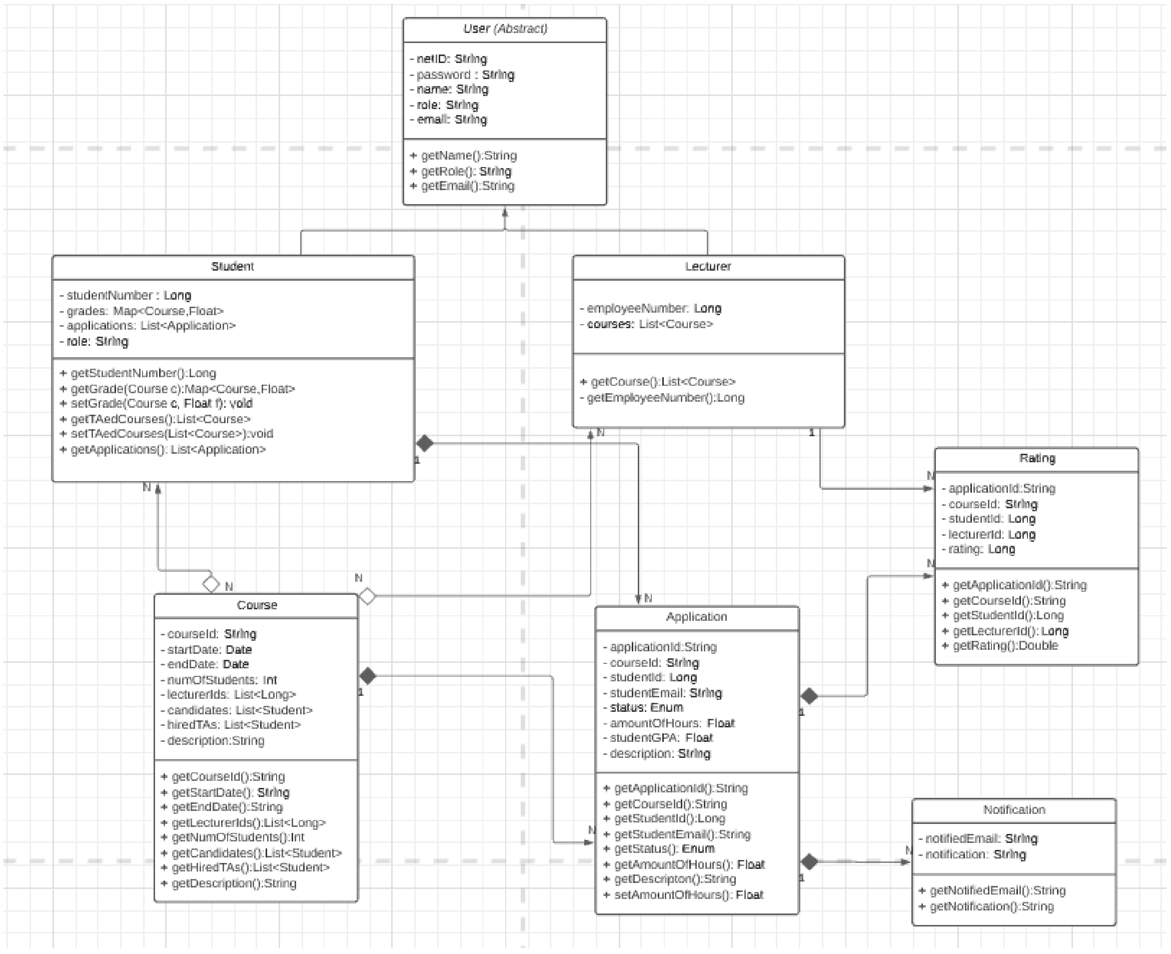
The **Rating** micro-service allows lecturers to rate TAs. The rating micro-service will need to fetch all information on TAs from the user service and find relevant course lecturers from the course service to determine which TAs they can rate. Once ratings are completed by lecturers, this will lead to update requests to the user service. This has been made into a micro-service of its own because this will allow for extendability of additional rating features, such as rating a course or lecturer, when adding features to our system.

Motivation: It is arguable to implement a rating micro-service simply to offer potential Users of the Lecturer subclass the possibility to rate the TAs that participated in their course. The value in having this micro-service is to potentially extend this functionality beyond simply TA ratings into a far larger rating system.

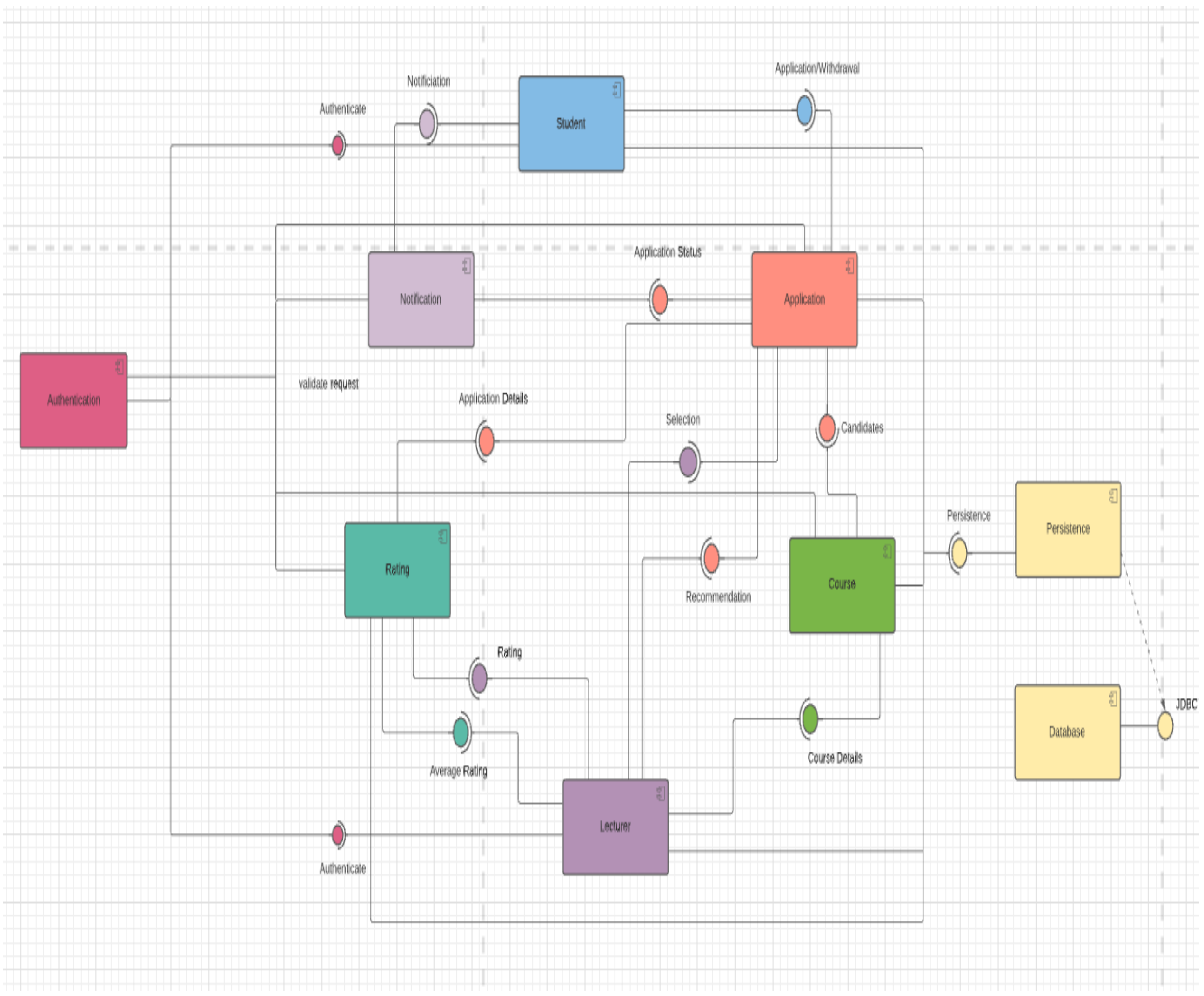
Micro-services

Provide motivations for architecture. Note how the context bounds from DD are mapped to the micro-services. How they interact with each other.

UML Class Diagram(Draft)



UML Component Diagram(Draft)



Design Patterns