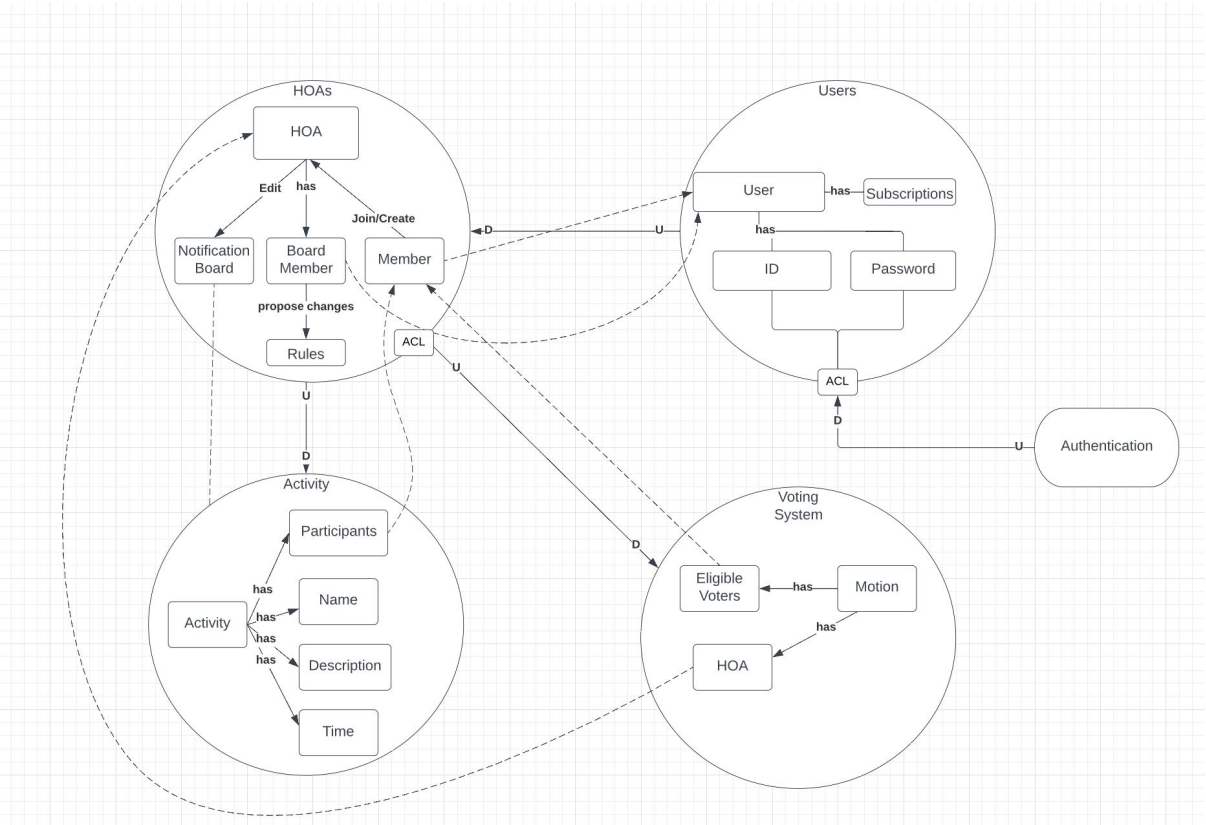
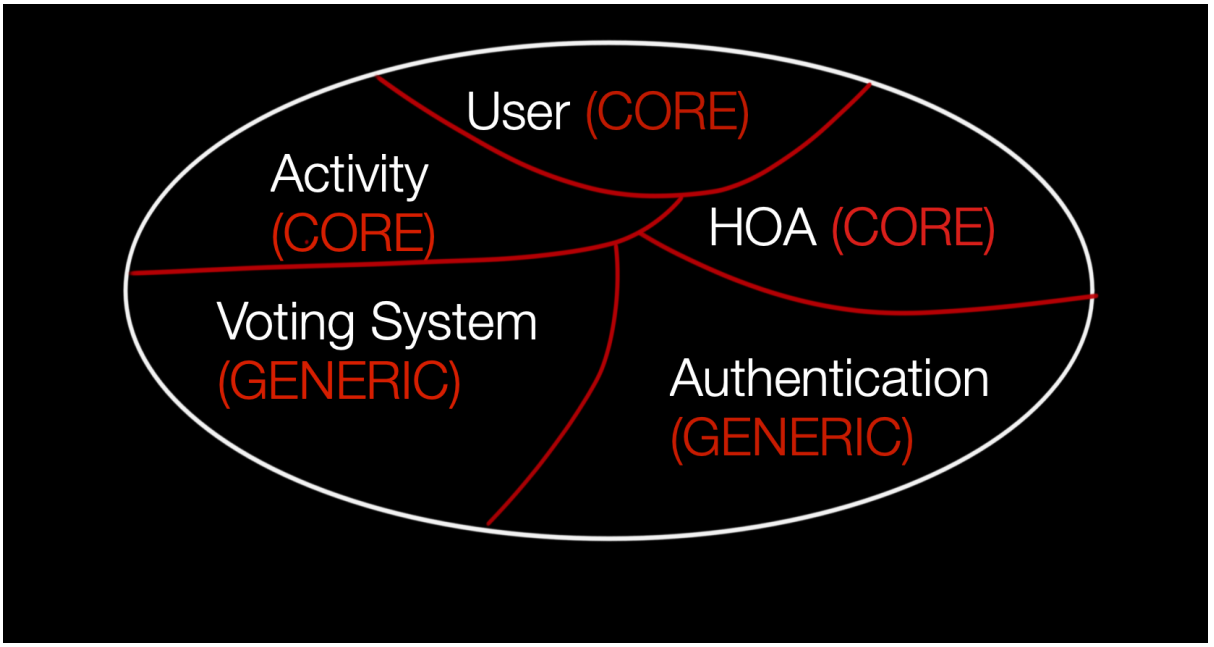
Group 22B - Assignment 1





Task 1

1. Bounded Context

We have identified 5 different bounded contexts from the given scenario using DDD (Domain-Driven Design). These domains are described as following

a) **Users -** The user is the central part of the whole project (and hence a *core domain*) and is the one who can Create or Join an HOA, Update its rules, Join its board and Create Activities. A user can have one of the following two subscriptions to a HOA:

- Member - a member of an HOA is someone who follows the joined HOAs' rules and has the option to create and participate in Activities on the Notification Board posted by the HOAs joined by the given user. When it comes to a HOA board a member of an HOA can either vote for board members or if the users meets some requirements can apply for that board, go through a Voting process and if successful will become a
- ✨Board Member✨ - board members are the core part of an HOA. They are responsible for creating, updating and voting on rules concerning the HOA they are a part of, which apply to all members of that HOA.

b) **Home Owners Associations (HOAs)** - Our whole application revolves around HOAs. That is why this is a *core domain*. A HOA is identified with a country, a city and the name of the HOAs. Users can join a HOA by providing their address. Every HOA has a board, which guides its requirements, and a notice board.

To become a board member, each user can apply as long as they satisfy certain requirements. Board members can propose changes to the rules. Then a voting process takes place, in order to decide whether the changes are accepted or not.

The notice board is used to publish activities that members of the HOA can attend or not, or to notify the members if there were any changes to the rules of the HOA.

c) **Activity** - The activities are an essential part of the HOAs. That is why this is a *core domain*. We decided to make them a separated entity since they require an independent set of operations. This includes that any members of a HOA can publish activities which other members can show interests. The activities are managed by the Notification Board. Each activity includes a name, a description, a time and participants (which are members of that HOA)

d) **Voting System** - We made this a separate bounded context that acts as a service to the HOA. That is why this is a *generic domain*. All that is needed to handle votes. Votes can be of various kinds: each member of an association can vote for the board candidates and each board member can vote on new rules for the association. We decided to separate the voting from the other contexts because it can be performed independently by a single component of the system, to avoid duplications and to improve security.
   Having a voting system separated from the rest also allows for further scalability: the system can be used to implement functionalities such as joining the activities or reporting members that don't follow the association's rules.
   Each vote will have a description of the proposal and a list of people that have the right to vote. The system will take care of applying the voting rules (such as not counting abstainers).

e) **Authentication** - We will need a service to authenticate users (as it is a service, it is a *generic domain*) and verify user credentials. This will be done by a distinct microservice. We will be using spring security for this purpose, it will check the validity of the username and password the user gives to it, and

2. Microservices

We decided to implement the system as a collection of microservices since this architecture has one major advantage over other architecture patterns: scalability. Each microservices can be developed and tested independently. This allows each member of the team to work on a different component of the project at the same time, independent of each other. Initially, we mapped each bounded context to one microservice. However, this creates unnecessary overheads for communications between different services. Therefore, we decided to merge Activity into HOA since it is a major component of HOA and Authentication into User since only User requires Authentication functionalities. Finally, we have in total 3 microservices, which are described as following.

a) **User Microservice**
   The User Microservice
b) **HOA Microservice**
   advvae
c) **Voting Microservice**
   fkafek