SEM - Assignment 1

Requirements Homeowners Association

Part 1 - Requirements

Functional Requirements

MUST HAVES

- The user must be able to create an account.
- The user must be able to create a new HOA or join an existing one.
- **Every HOA must have an associated country, city and a custom name.**
- The user must be able to register with their address **that contains a country, city, street, house number, and postal code.**
- Every HOA must have a board which contains current requirements.
- A **board election** must be held once every year.
- Any member can apply for a board if given requirements are met:
    - An applicant will need to have been part of the HOA for at least 3 years.
    - An applicant is not allowed to be in 2 boards at once.
    - A member is not allowed to have been on the board for more than 10 years in total for a specific HOA.
    - Any member of the HOA can vote for a potential applicant, except themselves.
- A member must be capable of changing their vote.
- Members of the HOA must be able to vote for the next board.
- Any board member must be able to request a rule change, whether it's a new rule or change of an existing one.
- A proposal requires a majority vote to pass.
- Any request must be approved in a vote that takes 2 weeks after its proposal.
- Only current board members can vote on the new proposal.
- A proposal voting must not count abstain votes.

SHOULD HAVES

- The user should be able to leave a HOA.
- Every HOA should have a public notice board.
- Any member of the HOA should be able to publish activities.
- Every activity on the notice board should require **a name, a time, and a short description.**

- All members should be able to show interest in given activities that they wish to participate in by selecting Interested/ Going.
- Activities should be visible only to the members of the pertaining HOA.
- Old activities should no longer be visible on the notice board.

COULD HAVES

- In the case a proposal gets accepted all members of the HOA could be notified that a change has taken place and what that change was.
- All elections could be public in their entirety, both results and vote count.
- All users could be able to view the full history of the HOA and its elections.
- The user could be able to report other users of violating a requirement including a reference to the specific requirement that was broken.
- A board member could be able to change the voting dates.

WON'T HAVES

- The app shall not have a GUI.

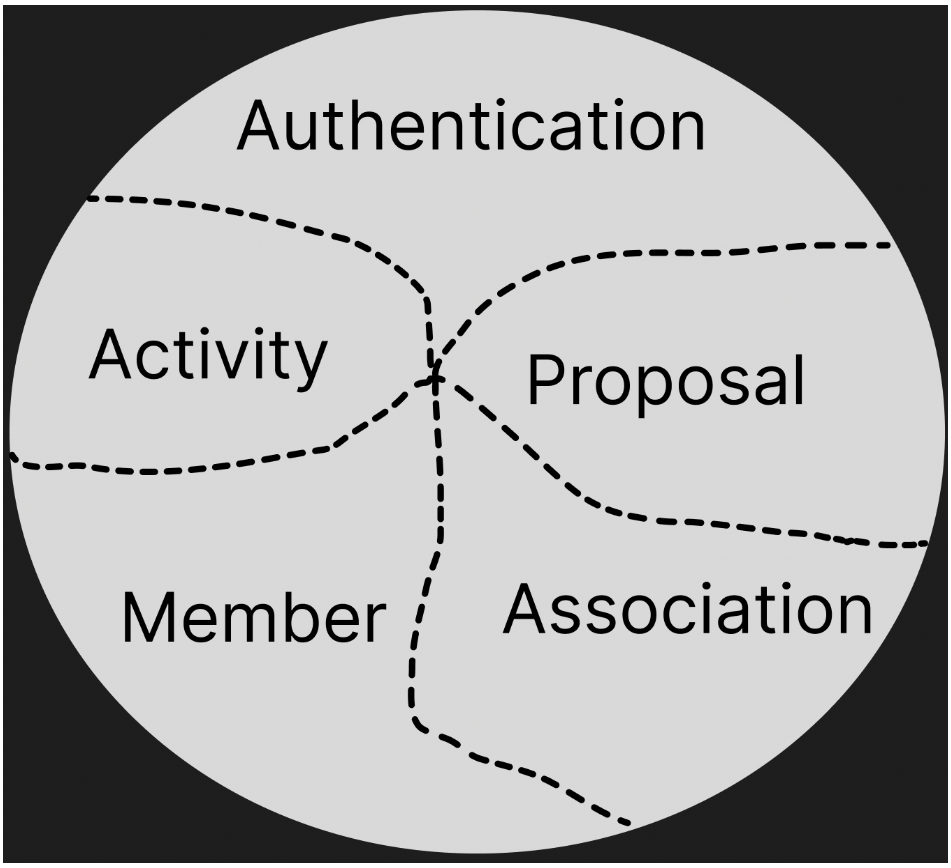
Non-Functional Requirements

- The app shall be implemented in Java 11 with Spring Boot/Security and Gradle.
- The app should be maintainable and expandable.
- The system should make use of at least 2 microservices.
- The interactions with the system are handled through APIs.
- For the iterations after the delivery of the first fully working version, the Scrum methodology shall be applied.
- The app should have secure authentication utilizing member IDs and a hashed password.
- The member IDs should be unique.

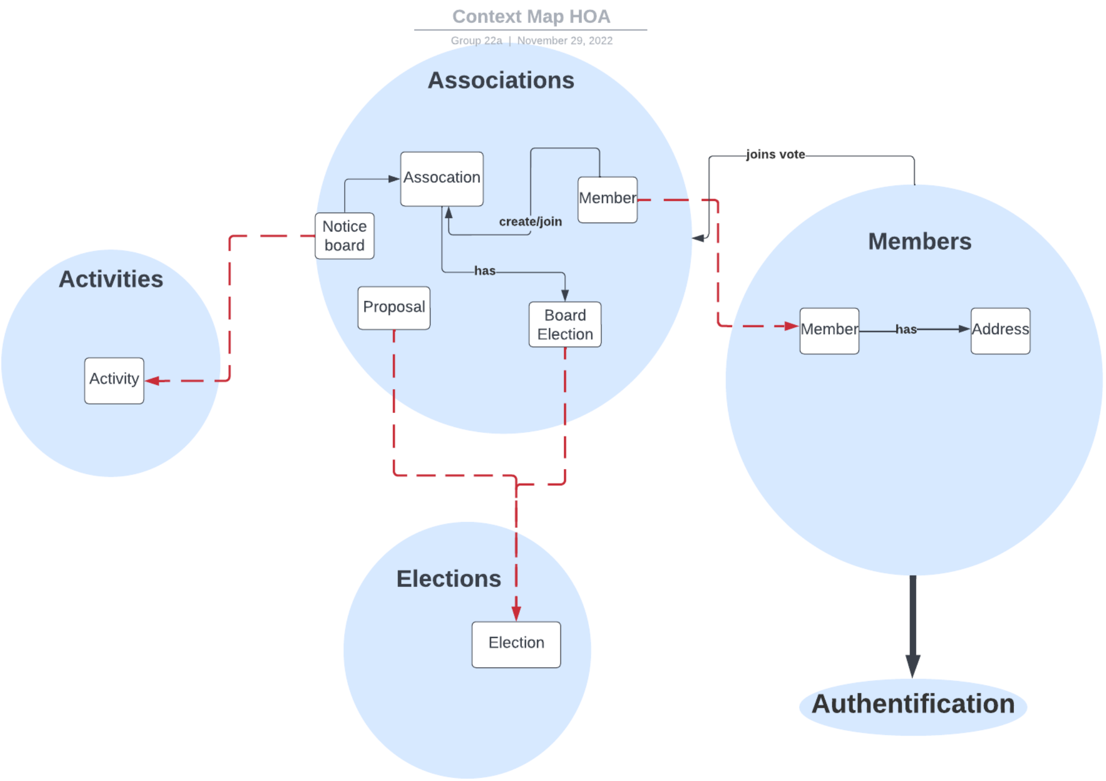
Part 2 - Domain Driven Development

Step 1: Bounded contexts

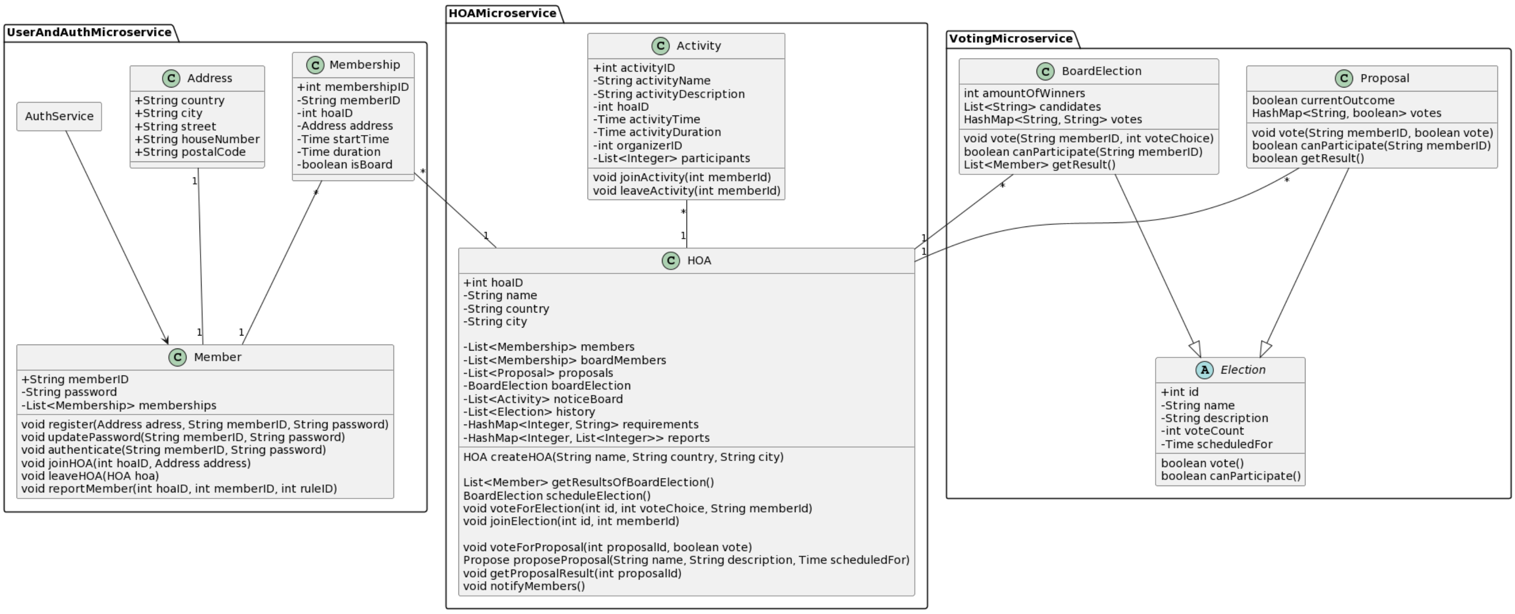
- **Authentication - Generic**
- **Member - Core**
- **Association - Core**
- **Activity - Support**
- **Proposal - Support**



Step 2: Context map



Step3: UML Diagram



```
Address
+String country
+String city
+String street
+String houseNumber
+String postalCode
```

```
Membership
+int membershipID
-String memberID
-int hoaID
-Address address
-Time startTime
-Time duration
-boolean isBoard
```

```
Member
+String memberID
-String password
-List<Membership> memberships
void register(Address adress, String memberID, String password)
void updatePassword(String memberID, String password)
void authenticate(String memberID, String password)
void joinHOA(int hoaID, Address address)
void leaveHOA(HOA hoa)
void reportMember(int hoaID, int memberID, int ruleID)
```

```
Activity
+int activityID
-String activityName
-String activityDescription
-int hoaID
-Time activityTime
-Time activityDuration
-int organizerID
-List<Integer> participants
void joinActivity(int memberId)
void leaveActivity(int memberId)
```

```
HOA
+int hoaID
-String name
-String country
-String city

-List<Membership> members
-List<Membership> boardMembers
-List<Proposal> proposals
-BoardElection boardElection
-List<Activity> noticeBoard
-List<Election> history
-HashMap<Integer, String> requirements
-HashMap<Integer, List<Integer>> reports
HOA createHOA(String name, String country, String city)

List<Member> getResultsOfBoardElection()
BoardElection scheduleElection()
void voteForElection(int id, int voteChoice, String memberId)
void joinElection(int id, int memberId)

void voteForProposal(int proposalId, boolean vote)
Propose proposeProposal(String name, String description, Time scheduledFor)
void getProposalResult(int proposalId)
void notifyMembers()
```

```
BoardElection
int amountOfWinners
List<String> candidates
HashMap<String, String> votes
void vote(String memberID, int voteChoice)
boolean canParticipate(String memberID)
List<Member> getResult()
```

```
Proposal
boolean currentOutcome
HashMap<String, boolean> votes
void vote(String memberID, boolean vote)
boolean canParticipate(String memberID)
boolean getResult()
```

```
Election
+int id
-String name
-String description
-int voteCount
-Time scheduledFor
boolean vote()
boolean canParticipate()
```