SEM Assignment 1 - Architecture Report

December 2022

1 Bounded Contexts

The task is to create a Home Owners Association (HOAs for short) application, scalable for future expansion and optimal for current use. Speaking to clients, the group has identified six bounded contexts: security, board elections, HOA management, board management, public notice board, and requirements. Each of these are independent enough to be treated differently from others and will contain their own logic when translated into code. As far as an architecture model, the team has decided on microservices due to their scalability, ease of delegation of implementation to other services, and simplicity of future expansion with new features.

2 Mapping of Microservices

The six bounded contexts of the application translate into five microservices in the aim to achieve high cohesion and low coupling.

- The Gateway/Security microservice was chosen to do the routing and authentication of each user, handling the security bounding context. It deserves to be separate from the others for the technical purpose of direct easy access to different parts of the application, as well as handling account creation easily. The team chose not to separate the security from the gateway since when drawing out the architecture in UML, it ended up being the only service that interacted with it. On top of being the only microservice to use security, it also always uses it.

- The Voting microservice was chosen to handle the board elections and a part of the requirements bounding context (the modification/creation of requirements). The team decided for it to be separate from the others for the difference it has from other features and the possibility of future reuse. Elections are not the primary purpose of a Home Owners Association, but a means to an end of electing a board and adjusting requirements. They will be rare enough so that no network overhead will be felt while not losing any functionality. Additionally, since every type of vote is different,

the microservice can easily be extended to host other types of elections, possibly for other apps.

- The HOA microservice has the purpose of handling all of the internal maintenance of homeowner associations in general, addressing the HOA management bounded context. This will include association creation, ensuring that a (suitable) board exists, calling elections, rule enforcement, receiving results, and communicating with the public notice board. It is a separate microservice due to the complicated conditions for an HOA's board, member requirements, and conditions for different types of elections. Their handling in a separate microservice will make the implementation of all others far easier since they would not be concerned with eligibility, when to be called, or any other homeowner association abstraction.

- The Public Notice Board (PNB) microservice fully overlaps the bounded context of the same name. It handles the posting, editing, and deletion of activities, indicating interest in activities, and viewing (upcoming) activities. Each activity has a name, a time and a description. The service also keep track of past activities and persists them in the central database.

- The Requirements microservice is responsible for handling all the logic that the requirements bounded context needs. This includes keeping track of rules that are in place for each HOA, as their main purpose is to have a structured and well kept neighbourhood through these restrictions and requirements. These rules also need to be put in place somehow. The way that is handled is through the board of the HOA, who propose and then vote for any additions and changes to them. Members are then notified that they may not be fully aware of the things they need to do. The enforcement of the requirements is done through the actions of members: if someone is not complying, they will be reported by the others to the board, who then can decide on an appropriate action to take, whether that is kicking them out of the neighbourhood or just warning them. These reports are anonymous so that no biases or conflicts between neighbours for reporting each other remain. The team decided to separate it from the other contexts because it's not essential for their work, and can be required to operate independently from them.

3 Microservices Structure and Connections

The project makes use of an interconnected web of microservices.

- All initial connections are routed to the Security/Gateway microservice, which allows for user creation and authentication. From this microservice, the user can be routed to several other microservices directly. This is done to distribute logic and reduce load as much as possible. Additionally, it will simplify tasks for the user, since they will not be forced to go through

the HOA for every single task (which would also turn the HOA in just a secondary gateway). The Gateway will be sharing its user data with the HOA microservice, so that it ends up stored in the central database.

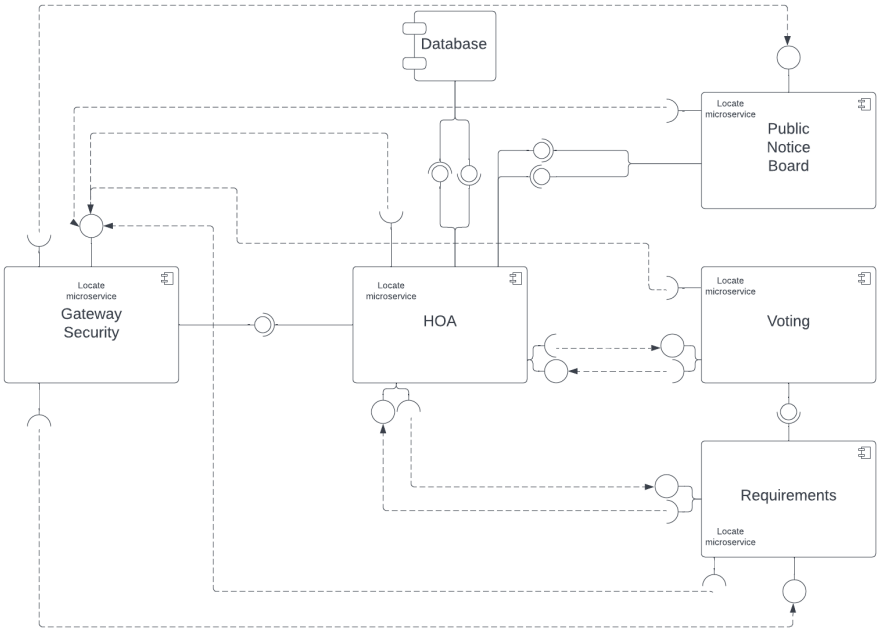
- A central HOA microservice communicates with all other microservices to keep track of members, activities, boards, and votes. The 'HOA Management' bounded context is contained within this microservice. It handles all of the internal logic to a Home Owners Association, such as how it is created, maintaining that its board exists, calling elections, enforcing the homeowner association's requirements, and communicating with the central database for the proper functionality of all other microservices. It communicates bi-directionally with the Gateway to connect the user to the proper HOA. There is also a connection to the Voting microservice, used for sending votes and receiving the results. Communication with the requirements is bidirectional so that the consequences of reports are felt, and the requirements microservice can route reports and be aware of which association members can do proposals. The HOA will also be the service that holds the central database of the application and providing the necessary tables necessary for the operation of each other microservice, including information like HOA membership, board membership, report history, public notice board history, etc. All of the microservice will be polling the HOA whenever they need that information or the HOA will be directly sending it to them.

- The access to the Public Notice Board (PNB) is routed via the gateway. For a user to access a notice board, their HOA membership needs to be verified. This functionality is achieved by linking the PNB and HOA microservices. For editing and deleting an activity, the user has to have created it, or be a board member in the relevant HOA. This information is obtained from the HOA microservice, which is why the two have a bidirectional communication established.

- A similar approach is taken for the Requirements microservice. A board member can propose deletion rules and creation requirements, so the microservice needs to communicate with the HOA microservice to verify whether a user is a board member. This is also a necessity for them to be able to place their vote, which is routed through the Voting microservice. Member reporting is also done through here, and this microservice is the last point of that request. The board members see the reports placed through the HOA microservice.

- The Voting microservice is connected to the HOA and Requirements microservices, these are the ones that will use its logic. For elections, (after the initial handling by the Gateway) all of the communications happens between the HOA and the Voting microservices. For requirement proposals, after getting a request form the Requirements microservice and conducting the voting, the Voting microservice sends the results to the HOA

microservice for persistence. The implementation inside of it is purely algorithmic, since there is no need for the Voting Microservice to store any data. All of its data is provided from either the Requirements or HOA microservices.



UML Diagram - Architecture