Project Documentation üK-223

Title:	Project Documentation Modul 223
Subject:	Realizing object-oriented multi-user applications
Team:	03 (Timofey Makhanov, Tsering Lhamo Anodunkhartsang, Sheung Tsung Lam)

Contents

Introduction

This is the Documentation for the ueK-223. For this course we must realize a [1]full-stack object-oriented [2]multi-user Application. We were divided into groups of 2-3 members. Each group is given a [3]model, our got the "**Group**" model.

** 1. Full-stack development: end-to-end development of applications. That includes a Frontend, Backend, and a Database.*

** 2. Multi-user Application: A Multi-User-Application entails that the software should be able to handle multiple users at once.*

** 3. Model: Models are the entities that are mapped to a table in a database.*

Tools

Below is the list of Tools we intend to use to develop our application.

- IntelliJ IDEA, various versions
- Visual Studio Code, various versions
- Docker Desktop
- Postman
- ClickUp
- draw.io
- D2
- Microsoft Teams (for communication)
- Microsoft Word (for documentation)

Project Location

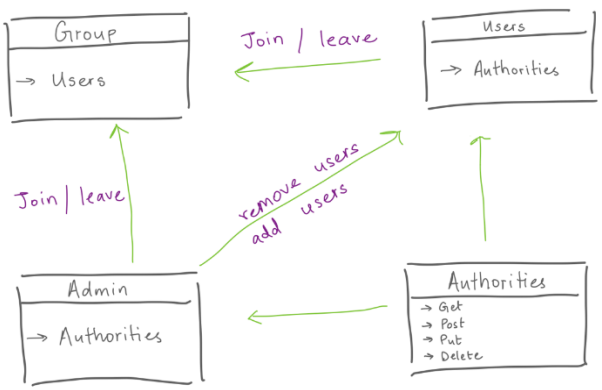
Backend:
We have uploaded our backend on GitHub publicly. You can go on this link to visit our code. There in the GitHub Repo, a starter Guide “README.md” file would be available for you. We also documented our Endpoints with the help of Swagger. You can visit the documentation here.

Frontend:
We have uploaded our frontend on GitHub publicly. You should be able to find our Frontend code here. There, you will find the starter guide of our Frontend with installation steps for required tools.

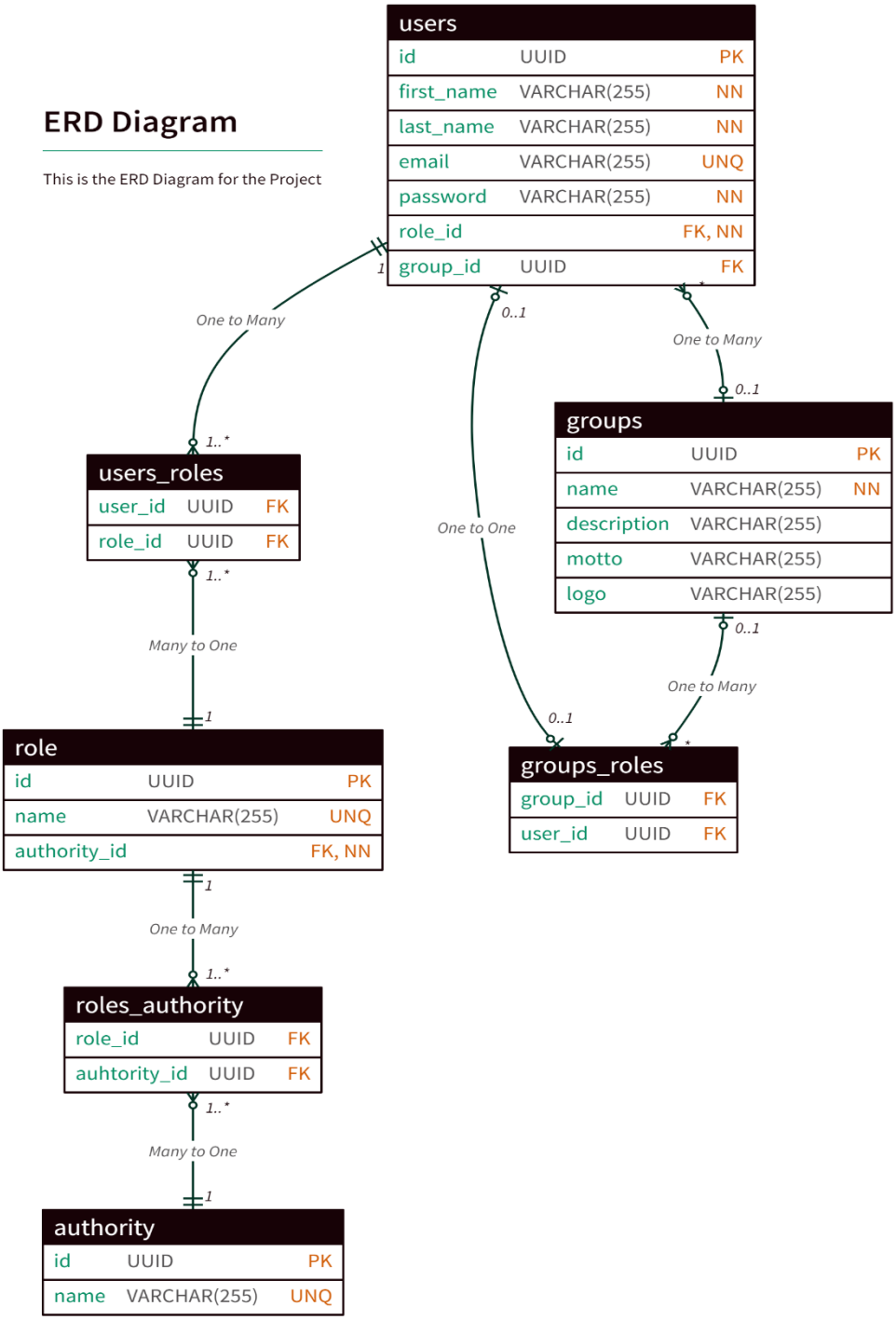
UML (Unified Modelling Language)

We created some UML (Domain Model, ERD, Sequence Diagram) to help us visualize our project.

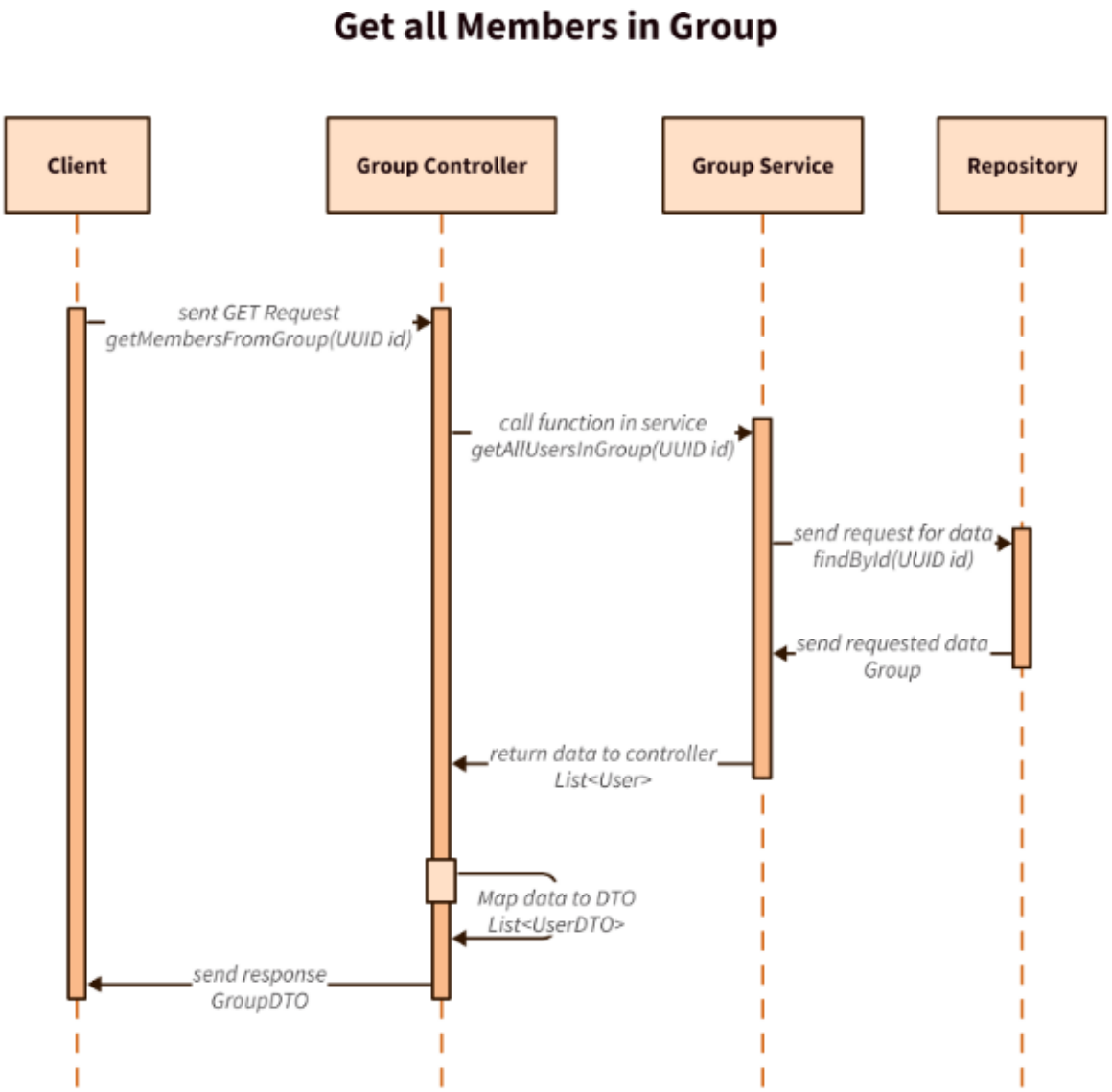
- Domain Model
  Below is a quick domain Model sketch of our program. Here we loosely define our entities and their relations with other entities.



- ERD (Entity Relational Diagram)
  We created our ERD with D2. In the Diagram we can see the notations between the tables and the Junction tables.



- Sequence Diagram
  Below is our Sequence Diagram. It tells us in a chronological manner what happens when the client sends a “Get” request, which gets all the members of the chosen group.



- Use Case Definition and Use Case Diagram

  The Use Case Definition and Diagram helps us define the process, including its pre- and postconditions, as well as the different possible resolutions.

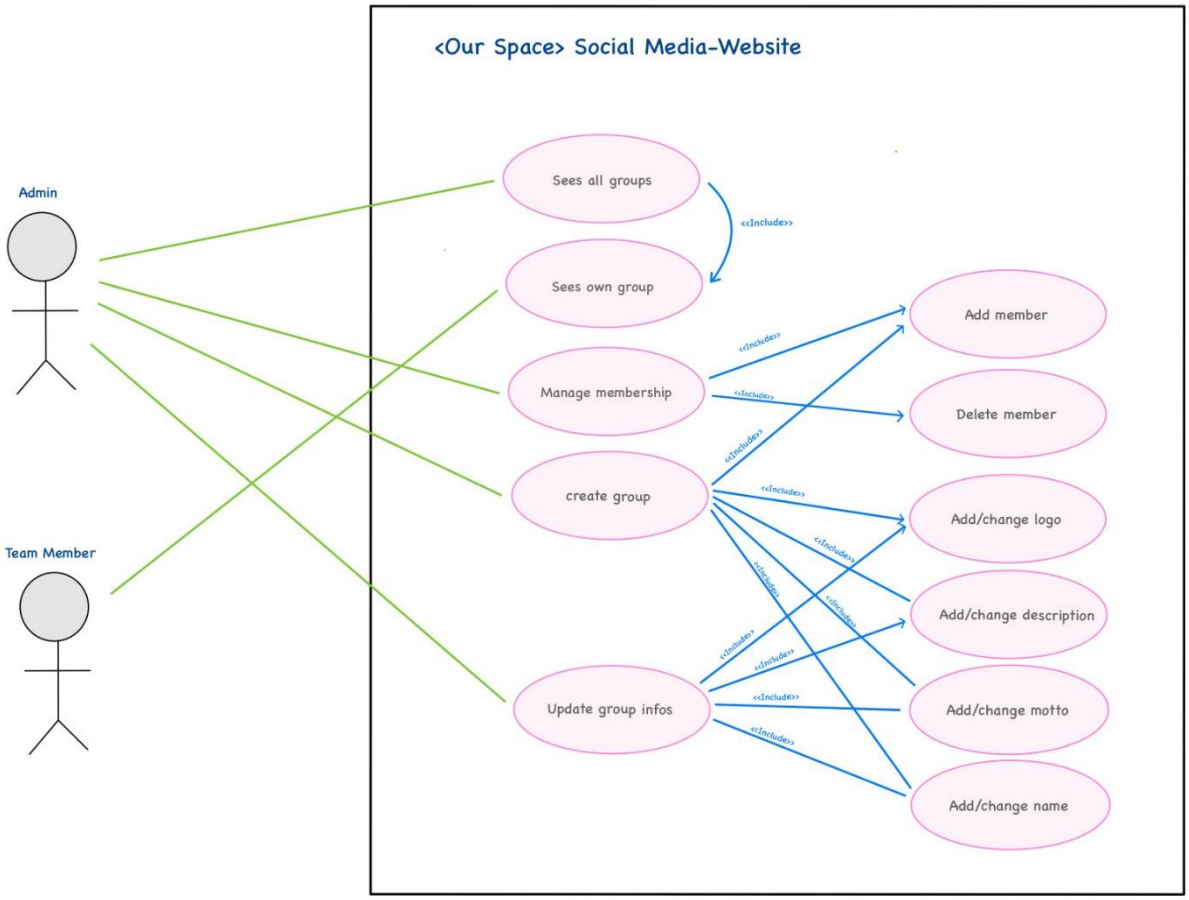
1.

Actor:	Admin
Description:	The Admin should be able to update a user's group id.
Preconditions:	✓ Admin credentials (email, password) have been validated. ✓ Admin has the authority to see the user's credentials. ✓ Admin has the update rights. ✓ Group must have users
Postconditions:	✓ Admin saves the changes. ✓ Admin changes are updated. ✓ Admin can see the updated data. ✓ User is in the new group.
Normal Course:	1. Admin logs in. 2. Admin then navigates to the user details list of a specific group (http://localhost:3000/:groupId/usersDetails). 3. Admin changed a user's membership. 4. Admin gets redirected to the homepage.
Alternative Courses:	-
Exception:	-

2.

Actor:	Team Member (normal user)
Description:	The team Member should be able to see its own group.
Preconditions:	✓ The team member credentials (email, password) have been validated. ✓ User has the authority to see only its own credentials. ✓ User can see its own Group Logo, motto, description, and name. ✓ Group must have users. ✓ User doesn't have the update authorities.
Postconditions:	✓ Users see the groups they are members of in the group list. ✓ Users see the members of his/her group.
Normal Course:	1. User logs in. 2. User navigates to group list (http://localhost:3001/) 3. User navigates to the user list (http://localhost:3000/users/:groupId)
Alternative Courses:	-
Exception:	-

Use-Case Diagram:



Testing

Frontend

Testing strategy:

Testing Level:	End-to-end
Tool:	Cypress
Reason:	- Our goal with this form of testing is to check the program from start to end. - We chose cypress because it automates user interactions programmatically and records tests and provides snapshots.

Backend

Testing strategy:

Testing Level:	Component
Tool:	Postman
Reason:	- We want to be sure that every endpoint gives the client the requested data. - We chose Postman as our testing tool because it is user friendly, able to automate and easily test multiple requests.

Sources

- Image-1: https://miro.medium.com/v2/resize:fit:605/1*vTLOUrJWoo4Kzj7ly8o0jA.png