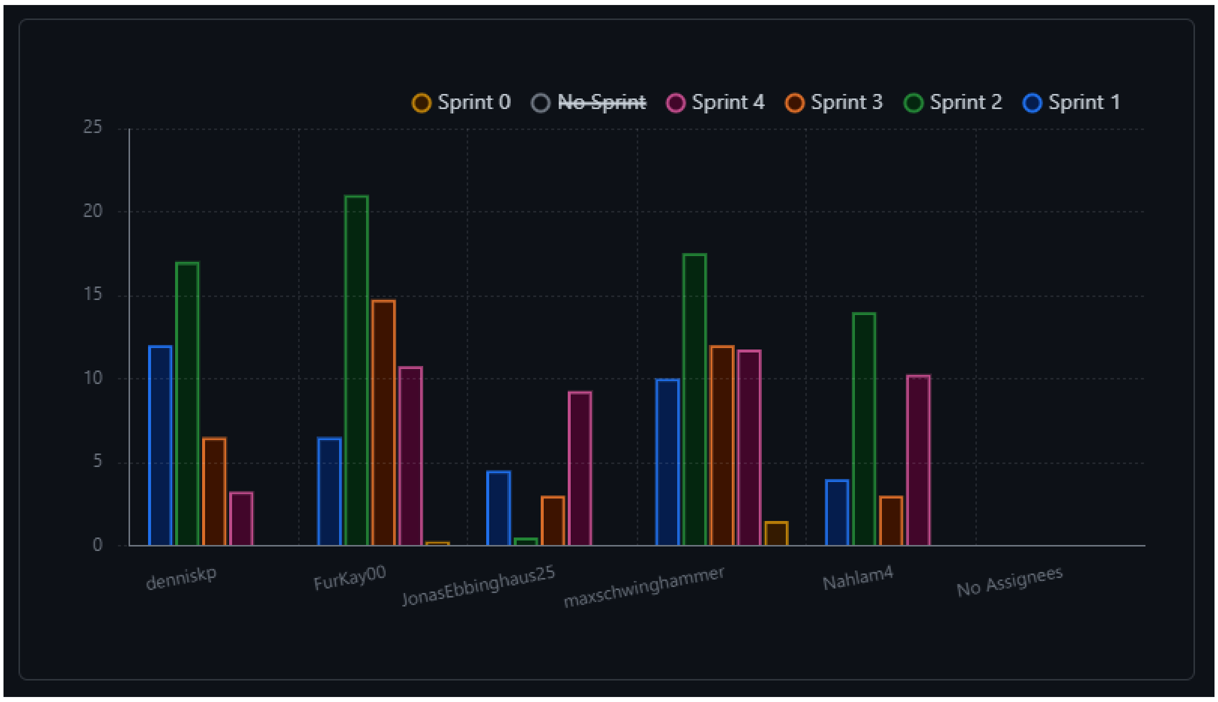
DHBWhub

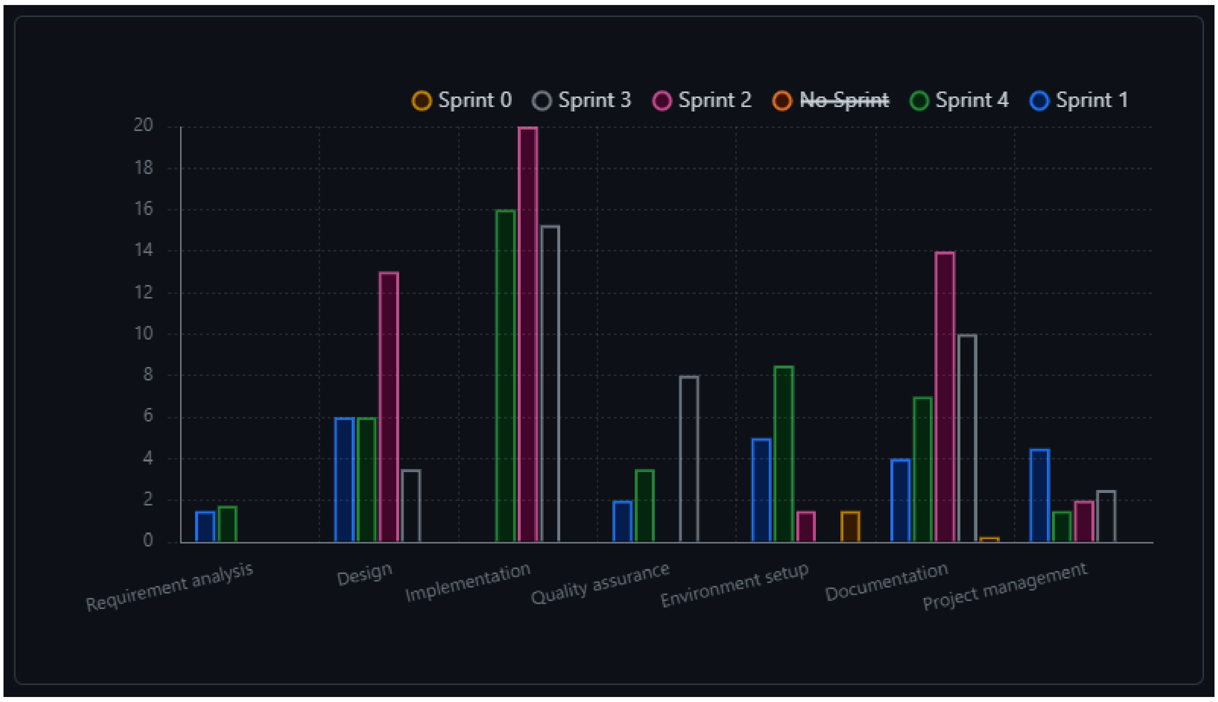
On this paper, you'll get a short overview of the main aspects of our project such as the general project statistics, main use cases, class diagrams and design patterns as well as the technologies, software and platforms we have used so far.

Statistics of efforts

Hours per person and major contributions



Hours per workflow



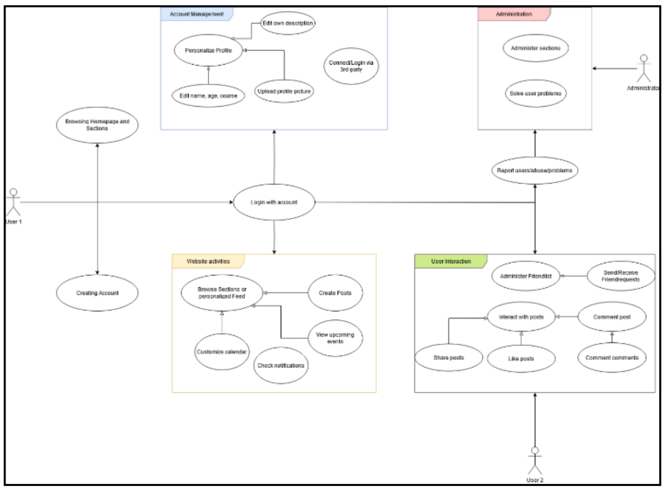
Major contributions:

- Dennis: Java code, Github Actions setup, Database setup
- Furkan: Documentation, Database setup, Java code
- Jonas: Mockup and first implementation of login/sign-up
- Josha: Documentation, Mockup of the calendar view
- Max: Mockup of homepage and post view, implementation of the header and posts, Github Actions setup, Dependabot setup

Use Cases

General use cases for the end user (red marked points are the cases we have mainly worked on this semester):

- Browsing Homepage and sections
- Creating an account/Signing up
- Signing in with your account
- Report problems to the administrator
- Account management:
  - Profile personalization
  - Managing 3rd-party sign-in
- Website activities:
  - Browse personalized feed or sections
  - Create posts
  - Customize/View Calendar
  - Check notification
- User interaction
  - Interact with post:
    - Like/Share/Comment post
    - Comment other comments
  - Administer friends list:
    - Create new friend request
    - Accept/Receive new friend requests
    - Manage your friends (Tags, Delete, etc.)



Class diagrams and design patterns

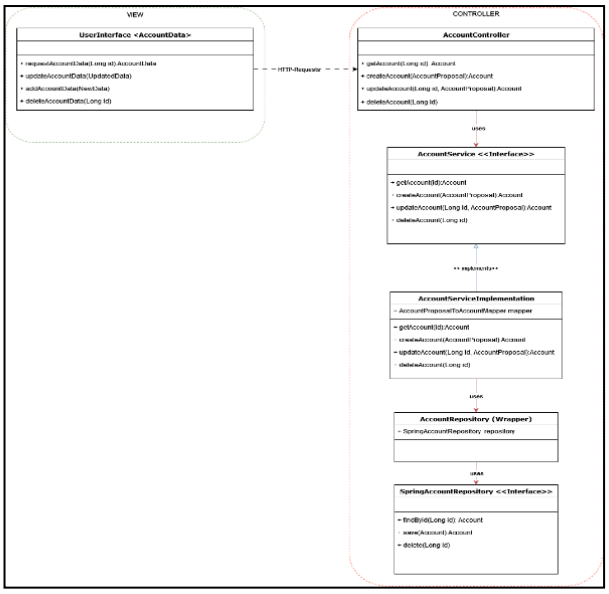
Identified classes/entities:

- Accounts (contain general, identifying information regarding both user types)
  - Client User/End user (Consumers of our product, must belong to a course)
  - Administrators (Administrating site, solving end user problems)
- Pictures (optional, may be used by posts or accounts)
- Faculty (rough segmentation of the university like economics or technology)
- Course (granular segmentation of the faculty e.g., 22B6)
- Post (can be saved or created by client users, may also include pictures)
- Comments (must be written under a post, can also be liked)

Design patterns

MVC:

- Models:
  - Entities previously mentioned/captured as classes
- View:
  - User interface/the website itself displaying all the content
  - Gets all the necessary filtered information through the Spring application
- Controller:
  - Spring Web Service, is called by the corresponding view component
  - Composed of:
    - Entity-controller (end point for the view component, uses service)
    - Entity-service (Implements business logic, uses entity repository, responds to controller)
    - Entity-repository (Retrieves data from backend, responds to service)


VIEW
UserInterface <AccountData>
+ requestAccountData(Long id):AccountData
+ updateAccountData(UpdatedData)
+ addAccountData(NewData)
+ deleteAccountData(Long Id)
HTTP-Requests
CONTROLLER
AccountController
+ getAccount(Long id): Account
+ createAccount(AccountProposal):Account
+ updateAccount(Long id, AccountProposal):Account
+ deleteAccount(Long Id)
uses
AccountService <<Interface>>
+ getAccount(Id):Account
+ createAccount(AccountProposal):Account
+ updateAccount(Long Id, AccountProposal):Account
+ deleteAccount(Long id)
<<implements>>
AccountServiceImplementation
- AccountProposalToAccountMapper mapper
+ getAccount(Id):Account
+ createAccount(AccountProposal):Account
+ updateAccount(Long Id, AccountProposal):Account
+ deleteAccount(Long id)
uses
AccountRepository (Wrapper)
- SpringAccountRepository repository
uses
SpringAccountRepository <<Interface>>
+ findById(Long Id): Account
+ save(Account):Account
+ delete(Long Id)

Interface-Segregation:

- Each interface and controller are only used for one entity, thus guaranteeing segregated responsibilities

Tech-Stack

The technology used:

- Backend: Java, Maven, Spring Boot
- Database: PostgreSQL
- Frontend: React, TypeScript, npm
- IDE: IntelliJ IDEA
- Project Management: GitHub, Discord, WhatsApp
- Documentation: GitHub Discussions, draw.io, Figma
- CI/CD: GitHub Actions (CodeQL, Maven CI, React CI, Dependabot)
- Libraries: JUnit