Functional requirements

- ***Must***
    - The system must support user authentication. All users of the system need to authenticate themselves to determine who they are and what they can do on the platform
    - Users will have an **ID** (member ID) associated with their account and the **type of subscriptions**
    - The **credentials** for an account will be represented by the combination of **ID** and **password**
    - The **User ID** will be a unique string (primary key, unique identifier for each user)
    - The password of each user needs to be stored safely
    - Users must be able to create an account
    - Users must be able to specify availability time slots
    - A user must be allowed to review all the trainings and competitions that they are able to pick
    - The activity owner must be able to accept or decline users' requests to join the trainings
    - There must be 5 types of positions that can be filled: cox, coach, port side rower, starboard side rower, and sculling rower
    - Users must be able to fill in positions they are capable to take
    - Users must be allowed to publish trainings or competitions with positions that still need to be filled
    - Users must not be able to edit or cancel activities published by other users
- ***Should***
    - Availabilities should be flexible
    - Users that published an activity should be able to cancel that specific one
    - Users that published an activity should be able to edit that specific one
    - The system should be able to host competitions
    - Users should be allowed to specify gender, organization in which they take part in, and competitiveness
    - Some competitions should have an entry requirement of **competitiveness level, gender and affiliation**
    - The matching should be done on a first come first serve basis
    - A user should not be matched with a training if the training starts within a half hour
    - A user should not be matched when the competition starts within a day
    - Different certificates could be earned by each user per boat type
    - Each user must be able to provide the certificate they possess
    - Certificates for different boat types should supersede each other with the following priority: C4, 4+ or 8+ (where 8+ can replace any of the previous certificates)
    - Taking cox position will only be filled by a user that has earned a certificate for that boat type

- ***Could***
    - Future different boat types could be added later on
    - Competitions could have additional requirements for partaking a specific one, such as users to be of similar gender OR users to be from the same organisation
    - The trainee should be notified that they were accepted for a specific training after the activity owner has approved the request
- ***Won't***
    - The system will not have a graphical user interface (GUI)

Non-functional requirements

- The system will be modular (extendable with extra functionalities later)
- The system will expose data by using APIs to allow for easy integration with other systems
- All interactions with the system will be handled through APIs
- The system will follow the microservice design pattern in order to be scalable
- The system will be written in Java programming language (version 11)
- The system will be built with Spring Boot (Spring framework) and Gradle
- The authorization and authentication of the system will be implemented using Spring Security