Functional requirements:

1) *Must*:
    - Users need to have an ID and a password as the credentials for the account
    - All users of the system need to authenticate themselves to determine who they are and what they can do on the platform
    - A user must be able to publish training sessions or competitions with positions that still need to be filled.
    - A user must be able to create an account so that he/she can specify his availabilities and what positions he/she is capable of filling
    - The user must be able to create an activity on the platform
    - The activity owner can accept or decline a request from a user to join the training
    - A person is capable of reviewing all the training sessions and competitions that they are able to pick.
    - Users can select to be part of a training session
    - The owner can decline or accept the request for a trainee to join the training
2) *Should*:
    - Availabilities should be flexible
    - A user should not be able to match with a competition within a day of the start of the training
    - 5 different positions : cox, coach, port side rower, starboard side rower, sculling rower
    - In order to be allowed to be a cox you need to have earned a certificate per boat type. Such as a C4, a 4+, or an 8+.
    - The system should function for competitions, partaking in a competition will have extra strings attached such as:
        - Rowers in the boat need to be of the same gender and same organization.
        - Certain competition matches should not be available to amateur rowers
    - A user should only be able to edit or delete an activity they are the owner of.
3) *Could*:
    - The boat certificates supersede each other in that order: C4 < 4+< 8+
    - Trainee should be notified when accepted for training
    - Support new/different boat types (certificates)
    - A user should not be able to match with a training within half an hour of the start of the training
4) *Won't*:
    - Not have a GUI, but all interactions are handled through APIs

Non-Functional requirements:

1) *Must*:
    - Java programming language (version 11)
    - The system must be built with Spring Boot (Spring framework) and Gradle.
    - Individual components of the system need to be scalable so that when the demand for that service increases, the service does not get overloaded (microservices)
    - The password needs to be stored safely.
    - The working application should be done by the 23rd of december

2) *Should*:
    - Use Spring Security
3) *Could*:
    - The platform shall have at least 50% of meaningful line test coverage.