1. **Stakeholders:**
    a. Regular user - normal user of the system
    b. Author - person who has an officially published book that is included in the database of the application
    c. Admin - overall control over the system's users and database

2. **Functional Requirements:**
    a. Must Haves
        i. Users can add reviews to a book.
        ii. Users can edit previously written reviews.
        iii. Users can delete previously written reviews.
        iv. Users can associate a rating for the book with each review.
        v. The average rating of each book is visible to every user, which should always be up-to-date.
        vi. Users can see all reviews connected to a book.
        vii. Users can interact with reviews.
            1. Write comments.
            2. Upvote and downvote reviews and comments.
        viii. Users have to be able to see when a review was written.
    b. Should Haves
        i. Authors can request a report about the interactions and feedback of their books.
            1. They have to be able to see the average rating, most relevant reviews, etc.
        ii. Users can filter reviews based on different criteria: most recent, highest rated, most helpful and so on.
        iii. Comments should be rejected if they contain inappropriate words.
        iv. Admins can review comments and remove them.
        v. If needed, admins can ban users (emails) after a report.
    c. Could Haves
        i. Users can report reviews for inappropriate/offensive comments.
        ii. The system should filter out spam and foul language.
    d. Would/Won't Haves
        i. Authors can see how many people interacted with their books.

3. **Non-functional Requirements:**
    a. Spring Boot and Gradle.
    b. Web-based communication via some format.

c. The system should have no GUI, interactions only through API.
d. Storing everything in a database for access across multiple platforms and time.
e. Java 15.
f. Mutation testing for the microservices.
g. Usage of GitLab during the development process for organizing tasks and keeping a code base with a passing pipeline.