Text based social media

1.Tech Stack

- Java 17
- Eclipse IDE with maven as build tool.
- PostgreSQL
- Java Spring boot

2. Use Cases

➢ 1.1 LogInToApplication

Use case ID	UC1
Actors	User
Pre conditions	User must be registered in the database.
Main flow of events	1. Use case starts when user selects Log In 2. System asks for username & email. 3. User fills in his credentials. 3.1. Use case end if credentials are correct. 3.2. If the user name and password are incorrect, an exception is thrown.
Post conditions	User logs in to the app.

➢ 1.2 RegisterToApplication

Use case ID	UC2
Actors	User
Pre conditions	-
Main flow of events	1. Use case starts when user selects register. 2. System asks to fill in his username & email considering if the total number of registered users are less than 500. 2.1. If the total number of registered users are more than 500, an exception is thrown and the use case ends. 3. User chooses a username & fills in his email. 3.1. Use case end if user's username & email are both unique. 3.2. If it's not an exception is thrown.
Post conditions	User is registered in database.

➢ 1.3 PostASmallText

Use case ID	UC3
Actors	User
Pre conditions	User has to be registered in the database.
Main flow of events	1. Use case starts when user wants to write a comment on a post. 2. User selects a post to comment. 3. User types a small text. i. If text is less 1000 characters it is posted. ii. If its not system throws an exception. In both cases, the use case ends.
Post conditions	Post is saved in database.

➢ 1.4 CommentOnPost

Use case ID	UC4
Actors	User
Pre conditions	User has to be registered in the database. Post has to be in the database.
Main flow of events	1. Use case starts when user wants to comment on a post. 2. System checks how many time he has commented in that post. 3. If it's less than 10 times the use case end. 4. If it's not an exception is thrown.
Post conditions	Comment is saved in the database.

➢ 1.5 ViewPostsFromFollowingUsers

Use case ID	UC5
Actors	User
Pre conditions	User has to be registered in the database.
Main flow of events	1. Use case starts when user wants to view the posts from the people they follow.

	2. System returns all the posts ordered by reverse chronological order.
Post conditions	-

1.6 ViewListOfFollowers

Use case ID	UC6
Actors	User
Pre conditions	User has to be registered in the database.
Main flow of events	1) Use case starts when user wants to view who they follow. 2) System returns a list with all the users who follow the actor.
Post conditions	-

1.7 GetPosts&CommentsFromUser

Use case ID	UC7
Actors	User
Pre conditions	User has to be registered in the database.
Main flow of events	1) Use case starts when user wants to see all the posts from another user. 2) System returns every post posted by the given user and also returns all the comments on that post. (Up to 100 comments)
Post conditions	

1.8 Add/RemoveFollower

Use case ID	UC8
Actors	User
Pre conditions	Both users to be registered in the database.
Main flow of events	1) Use case starts when the actor want to update a follow status. 2) The actor selects to add/remove a follow status from another user.

Post conditions	The new follow status are saved in the database, the deleted ones are removed. Also the list of followers are update on each user object.

3.Database

I have created a database which consists of 4 tables.

1. Table users(username, email), username is a primary key and email is unique attribute. Each user has a unique combination of username & email.
2. Table posts(id, username, date, content). The attribute id is autogenerated by java spring, username refers to a user, date is the exact time the post was created and content is the small text written by the user.
3. Table comments(id, postid, username, date, content). The attribute id is autogenerated by java spring, postid refers to the id of the post entity, date is the exact time the post was created and content is the small text written by the user.
4. Table follows(id, user1email, user2email). The attribute id is autogenerated by java spring, user1email & user2email both refer to the email attribute of the user entity. The meaning of this table is that the user with user1email follows the user with user2email.

4.Instructions

- In order for the program to run you need to change the configuration of the java spring boot. In the application.properties file, it is needed to change username & password to connect to a database service at your local host.
- I have created an .sql file to load some data to the database. The data.sql file is automatically run at the beginning of the program.
  !!!! IMPORTANT: if you want to rerun the program you have to delete all the tables of to comment out the .sql file. This is needed due to the fact that every username/mail combination is unique so an duplicate primary key error is thrown.
- In the **DataGramApplication** class, in the **String run** method I have created a small scenario between two users.
- Some basics functionalities can be seen in the response endpoint of the localhost. More specific:

UC1	localhost:8080/users/login/username/email
UC2	localhost:8080/users/register/username/email
UC3	localhost:8080/posts/uploadPost/username/content
UC4	localhost:8080/comments/postComment/username/postid/content
UC5	localhost:8080/view/getPostsFromFollowingUsers/username

UC6	localhost:8080/view/getFollowers/username
UC7	localhost:8080/view/getPostsFromUser/username
UC8	localhost:8080/view/getFollowingUsers/username

- username should be replaced with the actual username.
- email Username should be replaced with the actual email.
- content should be be replace with the actual content you want to upload.
- postid should be replaced with the id of the post you want to comment.