Lecture 7

Register and login, JWT - JSON Web Tokens, authentication vs authorization

Register and login

- Most web applications give you the ability to register an account with them.
- Logging in to the account usually gives you access to more features:
  - Usually, things are read-only without login
  - An account lets you post / place orders / edit things
- We need a way to handle register + login on both the backend and the frontend, in a secure manner.

JWT - JSON Web Tokens

- Registration is easy: the user sends an e-mail and password and gets an account created.
- But how do we know which user a request is coming from?
  - We will use JWT: these are special tokens that store various user information that the server can verify.
  - We will store them in **localstorage** on the frontend and send them with each request.
  - Read more about JWT here: https://jwt.io/
- JWT authentication is implemented in various libraries across many languages and frameworks. Look up your own tech stack to see what your options are.

Authentication vs authorization

- In a nutshell:
  - Authentication: verifying if you are who you say you are.
  - Authorization: verifying that you are allowed to do what you're trying to do.
- Authentication is implemented with register and login, authorization is implemented with **user roles**.
- User roles can also be stored in JWT.
- The frontend can display different things based on whether we're logged in and on the role we have.
  - However, it's very important for the server to double check all this and **not** trust the frontend!

Examples

Some common scenarios:

- We might only want users in the **moderator** role to be able to delete other users' posts.
- On the frontend we only show a delete button for other users' posts to moderators. This button requests the endpoint **DELETE /api/posts/post_id**.
    - But what happens if a non-moderator finds out that this is the request for deleting a post and sends a request to it from their browser?
    - If the server doesn't also check that the request came from someone who has permission to do this, then we have a security issue in which anyone can delete any post.
    - If the server does check, it will simply reply with a **401** or **403** status code and nothing bad will happen.

Next steps in your projects

- You will need to implement login, register and some user roles in your next assignments.
- Make sure that you read up on how it should be done in your tech stack and do it properly. Reading up on it for a while before writing code can save you a lot of time in the long run.
- Try to also make it as realistic as possible with regards to the entities you chose.