Objective:

Add a database to store user account data. Implement sign-up form logic with validation.

Theory:

Deadline: 5 days

1. Read about ORM.
2. Read about relational and NoSQL databases, where they are used, the differences between them, and when one suits better than the other.
3. Read about SQL queries.

Task:

Requirements:

1. Start this task based on Stage 5.1.
2. Projects and users should now be stored in a database.
3. Use PostgreSQL as the required database.
4. Implement authentication logic using JWT strategy (POST /login endpoint).
    a. Use the jsonwebtoken library for JWT operations.
    b. JWT should be implemented with access and refresh tokens.
    c. Tokens should not be stored in the database.
5. Unauthorized users should be redirected to login if they are unauthorized or if tokens are expired.
    a. If the access token is expired (server sends a 401 status error) but the refresh token is valid, then the frontend should automatically refresh the access token and repeat failed requests.
6. The endpoint for getting projects should be protected by JWT.

7. Frontend: Create a `*/signup*` URL with a sign-up page where users need to fill in the following input fields to submit the form: username, password, repeat password, first name, last name, and age. The design is at your discretion.
8. Backend: Implement a REST API to provide the sign-up logic from the bullet above.
9. Create a database schema (tables, relationships) for all the functionality above.
10. Validation (happens on the backend, and the frontend only shows the error message received from the backend):
    a. If validation fails, then the validation error message should be shown under the field it refers to.
    b. Username must contain 3 symbols or more.
    c. Password must contain at least 1 number and 1 letter.
    d. Password must contain 4 symbols or more.
    e. Repeat password section validation (passwords should be the same).
    f. First name and last name must contain 3 symbols or more.
    g. Age must be a number and can't be zero.

Questions:

1. What is ORM, and how does it work?
2. What is the difference between relational and NoSQL databases, and when does each one suit better than the other?
3. What is Bearer authentication?
4. Explain the JWT flow:
    a. What are access and refresh tokens responsible for?
    b. Can a server have only access tokens without refresh ones? If yes, what are the pros and cons of such a solution?

c. What are the pros and cons of storing access/refresh tokens in a database compared to not storing them?
d. How does a modern web application understand that a client is unauthorized?