Marwa Hammani - 40289362

Individual Log Sprint 1 - Implementing Login and Registration API with MongoDB

Setting Up the Database Connection

I started by creating a MongoDB connection file inside lib/mongodb.js. This file ensures that the database connection is established and reused efficiently. Since Next.js can restart frequently in development mode, I implemented global caching to prevent multiple connections from being created.

To begin, I created the file lib/mongodb.js and imported Mongoose. I stored the MongoDB URI in an environment variable to keep credentials secure. To prevent multiple connections, I implemented a caching mechanism. The function connectToDatabase checks if a connection already exists and returns it if available. If no connection is found, it creates a new one and stores it in the cache. This ensures that only one connection is used throughout the app.

Creating the User Model

Next, I defined a User schema for MongoDB, which specifies how user data is structured. I created the file models/User.js and imported Mongoose. The schema includes fields for email and password, where the email must be unique. To prevent issues with hot reloading in development, I used a conditional export to avoid redefining the model.

Implementing the Registration API

The next step was to create the registration API in pages/api/auth/register.js. This API handles user sign-up while ensuring that the user does not already exist and that the password is securely hashed before storage.

I imported the necessary modules, including the database connection, User model, and bcrypt for password hashing. The API is restricted to POST requests to prevent unintended usage. Upon receiving a request, the server first connects to the database. It then checks if the user already exists. If the user is found, it returns an error message. Otherwise, it hashes the password using bcrypt and creates a new user in the database. If successful, it returns a success response to indicate that the registration was completed.

Implementing the Login API

After implementing registration, I created the login API in pages/api/auth/login.js. This API is responsible for verifying user credentials and generating a JWT token upon successful login.

I imported the necessary modules, including the database connection, User model, bcrypt for password verification, and jsonwebtoken for token generation. The API is restricted to POST requests. When a request is received, the server connects to the database and checks if the provided email exists. If no user is found, it returns an error message. If the user exists, the password is compared using bcrypt. If the password does not match, an error message is returned.

Once the credentials are validated, the API generates a JWT token that includes the user's ID and email. The token is signed with a secret key and set to expire in one hour. This token is then sent in the response, allowing the frontend to store it for authentication in subsequent requests.