idg2100-2025-oblig3

This document contains the description and starter code for `oblig3: IDG2100 Spring 2025`.

Goal

Prove and demonstrate:

- you understand the `React` library and how to use it to create `SPAs` using React Router v7
- you can build react components and understand the `component lifecycle`
- you know how to build `stateless` and `stateful` components using props and hooks
- you understand the concepts of state, context and side effects, and their role within `React` components
- you can connect the front-end to the back-end of your application
- you understand client-side routing and how to secure routes against unauthorized access
- you know and rely on the `React` good coding practices

Context

In this "oblig" you will be presented with a brief description of the problem you have to solve using what you have learnt so far in IDG2100.

This is an **individual assignment** and a continuation of the task presented in Oblig 2. The `backend` folder should contain the API you developed earlier for Oblig 2. If you modify the backend code, the modifications should be clearly acknowledged in the backend `readme.txt` and be visible as separate Git commits (on top of the 'baseline' version you delivered in Oblig 2).

This assignment has two parts: **a coding component** and **an on campus oral presentation**. You must submit your code via both GitHub and Blackboard (please refer to the Delivery section for specific instructions). The details regarding the dates and requirements for the physical oral presentation will be posted on Blackboard. Not attending the physical oral presentation will result in a non-passing grade.

Although, you may utilise snippets of code from tutorials or official documentation, you must clearly acknowledge the sources in the comments of your code. Plagiarism or cheating will be deemed to have taken place if the submitted code shows substantial similarities to other students' assignments or projects found online. In such cases, the matter will be reported to the NTNU appeals committee for further examination. If you have any doubts regarding the use of materials for your project, please reach out to the instructor for clarification.

If the assignment is graded as "not approved" you will have an additional opportunity based on the following conditions:

1. The first version of the project must have been delivered within the set deadline (never after);
2. The project must consist on a significant piece of work (i.e.: do not deliver an empty assignment);

For the second attempt you will receive a very short deadline to fix your project.

Brief: AI Literacy and Awareness quiz Interface

Scenario

Following the successful development of the AI Literacy and Awareness quiz API, the next step is to create a simple, intuitive user interface for the **Participant** part of it. This interface should allow users to interact with the quiz, answer its questions, and view their performance. The **Researcher/Educator** part of UI (responsible for CRUD on quizes, questions, and uploads) is to be scaffolded, but not implemented: the associated project files and components should be created, but may be mostly empty (e.g., the components could simply return "*This components is for X*" strings instead of JSX). The interface will serve as a crucial tool for enhancing AI literacy among the general public.

Key Features

The UI should allow for the following interaction flow:

- the user encounters a front page for a quiz enticing them to take it, which could include an image if there is one, a large-font title (call to action), a brief description of the quiz, a small-font section with terms and conditions (e.g., *By clicking 'begin' you agree to your data being recorded, used in academic research, and potentially presented in an aggregated form to an audience. Personal data - anything that allow us to uniquely identify you as a person - will **not** be recorded.*), and a button to initiate the quiz.
- the user initiates a quiz
    - quiz questions are shown one by one (e.g., '*which of these two images was generated by AI?*'), which the user answers
    - the user can navigate back and forth among the questions, but their answers cannot be changed after they've been given; yet-unseen questions cannot be navigated to
    - the user sees how many questions have been answered and how many are left
    - depending on how the quiz was configured:
        - an answer to the questions - and corresponding free-form text explanation - is revealed right after it is answered and remains visible if the user navigates back
        - answers to all questions - and corresponding free-form text explanations - are revealed at the end, together with the user performance information
    - the design of the quiz pages should be minimalistic, with only a question, answer button, and (possibly) answer itself visible
    - a minimalistic, unobtrusive footer with brief info about the quiz and its authors is always visible at the bottom
    - if the user abandons the quiz (2 min of no activity), the system automatically navigates back to the quiz front page
- once all questions are answered, the user fills in a demographic questionnaire (age group, gender, and familiarity with AI as multiple-choice items)
- the user sees the results of their participation: the number of correct answers, and their performance relative to all other users and to users in their category (either an age group, or age group within their gender)
- the system navigates back to the front page either automatically - after 30 seconds with a countdown visible - or when the user presses a "finish quiz" button

The above interaction flow presumes having:

- a quiz front page `/quizes/:quizId`
- a question page `/quizes/:quizId/sessions/:sessionId/questions/:questionId` or `/sessions/:sessionId/questions/:questionId` (if you can easily extract quizId from a

session and don't want to validate it every time)
- a demographics questionnaire page `/sessions/:sessionId/demographics`
- a results page `/sessions/:sessionId/results`

Note that `sessions` and `sessionId` refer to experimental sessions - a person taking a quiz - not to user sessions, which are a related but separate concept, referring to a web app 'remembering' the user during continuous interaction.
Note that the URLs above are only a suggestion, and you are free to change them as you see fit.

The UI should also include:

- a generic homepage for this entire project, showing some "About Us/Contact Us" info
- a page listing currently running quizes with links to them
- a page listing all Educators that contributed quizes to this project
- a profile page for each Researcher/Educator, with a picture, title, paragraph of description, and list of their quizes
- all pages should have a header with the project logo that redirects to the homepage
- the header should contain a theme toggle, allowing for switching between a light and dark visual theme
- the non-quiz pages should have a navigation menu
- the menu should have a navigation link to a "For Educators" page
- the "For Educators" page should be behind a ProtectedRoute component (i.e., accessing it should require a login); the login functionality itself does **not** have to be implemented.

The decisions on how to slice the UI in components are up to each individual student, but the list of re-usable components could include:

- Header. Includes a clickable logo, theme toggle and navigation menu (for the non-quiz pages only)
- Footer. Contains info about the site and quiz authors, and site purposes.

The navigation menu should, at least, link to the homepage, quiz listing page, educator listing page, and the protected-route "For Educators" page.

Task

- Understand the requirements: Review and understand the project requirements to ensure you have a clear understanding of what needs to be built.
- Research how to implement protected routes using `React router v7`.
- Define the User Interface: Sketch out the UI layout (consider creating wireframes in this step), and split it in components. Consider the design and content, the types of components to be used, and the flow of the user interactions.
- Decide if the UI requires you to add or modify the endpoints you've developed for Oblig 2, backend REST API; modify your backend API accordingly.
- Build a functional prototype of the quiz app using React components. Start with the landing page and continue to build out the other pages. Ensure that the design is responsive, user-friendly, and meets the requirements of the project. Ensure that all components work as expected, and the app is fully functional.
- Write a good readme file explaining how to install and deploy the application on a local machine. Include a script in the backend to seed the database with some dummy data for testing purposes.

Note: this assignment must include the code of both the front-end and the back-end.

Delivery

This assignment must be delivered in two different places: GitHub classroom and Blackboard.

- To deliver the assignment in GitHub Classroom, you only need to make sure all your changes and commits are pushed to your Git repository.
  - A Pull request is created automatically when the repository is cloned. Feedback will be included there if needed. Do not remove or close that Pull Request.
  - Only the changes in the "main" branch will be considered for giving feedback or grading the assignment.
- It is imperative that you work exclusively with this Git repository to ensure that all modifications are trackable and your code is backed up on a regular basis. Hence, you should commit your progress directly to this repository each time you make advancements.

- Before delivering the assignment in Blackboard, make sure your project has all the files it needs. Delete all files and folders that are not needed (this is `.git/` folder, `node_modules`, etc.). Zip the project and upload the file to Blackboard.
- Don't forget to add/update all the `API` specs in `documentation` and your query collection in the `REST scripts` folder, if needed.
- Remember you will have to present your project orally and on campus (the date to be posted on BB).

Useful resources

You can find additional resources here:

- React Router 7: Private Routes (alias Protected Routes)
- (video) React Protected Routes | Role-Based Authorization | React Router v6
- (Video) The New Way To Create Protected Routes With React Router V6