DS3103 Webutvikling

Exam

Autumn 2023

Practical information.

- This home exam can be solved and delivered alone, in a group of 2, or in a group of 3. If you work in a group of 2 or 3, you will be expected to code more and have more complexity in the code, than if you work alone.
- When you deliver the solution folder in WISEFlow it must first be zipped – you are to deliver 1 main folder containing the React folder, and the Web API folder. **Note: remove the node_modules** folder from the React folder before uploading to WISEFlow!
- In this exam you work fullstack with a React project, and a .NET/C# Web API project. The .NET/C# Web API is used by the React project to do CRUD to the database.
- Assessment guide for internal and external assessors, and for students is at the end of this exam text.
- Read through the entire exam text before starting.
- Do not add any functionality which requires any kind of login!
- Since this is a closed exam project you may use images freely in your solution.
- The teacher may create an FAQ/AQ in the course page to give extra information and clarifications about the exam.
- The exam is seen as part of the learning process in the course, and you can still consult with the teacher and the student assistants*****. If you are uncertain about any point, take contact with the teacher.

*****Does not apply if it is for a retake exam.

Focus areas of this exam.

- React and JavaScript (ES6+)
- .NET/C# Web API with a Database
- HTTP requests from frontend to backend
- HTML5 and CSS3
- CSS Grid / CSS Framework (Bootstrap or TailWind) including responsive Design
- Universal Design

Tip before starting coding.

The entire solution will be dependent on the information in the database (tables and their attributes) you will make. Use time to plan, among other the following before you start coding:

- Which properties the information you are having in the solution should have.
- What datatype the properties should be
- Which functionality you need.

When one knows how the information looks (through creating interfaces for example or simply writing it down somewhere) it is possible to work with frontend and backend at the same time.

Case: F1 Special Event

(E-mail from the boss): An urgent task to our fullstack developers!

We have gotten a big customer: the organization behind Formula 1! They need a solution for a special racing event where both professional and amateur drivers may join as drivers, and even drive different kinds of racing cars!

They need a Web API with a database which has information about Drivers, Teams, and Races. They also need a frontend solution which makes use of the Web API, and which is both informative and fun (possibly some quiz/game for example).

A **Driver** should have the following information:

- Id, Name, Age, Nationality, Image (of person)

A **Team** should have the following:

- Id, Manufacturer (for example Ferrari), Image (of the car), Driver1, Driver2

A **Race** should include the following:

- Id, WinnerName, WinnerTime, GrandPrix (country where the race took place), NumberOfLaps

See the following links just to understand the context of the above-mentioned information:

- https://www.formula1.com/en/drivers.html
- https://www.formula1.com/en/teams.html
- https://www.formula1.com/en/results.html/2023/races.html

In addition, they also want a HTML page which describes how to use the Web API: which endpoints are available, how to use the endpoints, which information is available etc.

I'm counting on you to solve this task in a great manner! 😊

Best regards

Daglig leder

FullyStacked AS

06.11.2023

Functionality

The main functionality is based on that the Web API has methods that do CRUD (Create, Read Update, and Delete) to the Database. The methods in the Web API are used from the frontend through HTTP requests (GET, POST, PUT, DELETE).

The main categories of functionality in the Web API are these:

- Get all of something
- Get something by id
- Get something by other property than id, for example GetByName
- Delete something
- Create something (including image upload)
- Update something

Other details

- The functionality in the frontend makes use of the above methods.
- The amount and quality of implemented functionality affects the grade.
- When you deliver the solution there must already be a database with some information in it!
- Note that you are not expected to create relations between the tables/model classes in the solution.

Regarding group size and size of solution

You should have 1 table, at least, per group member. The bigger the group, the more is expected in terms of size and complexity in the solution.

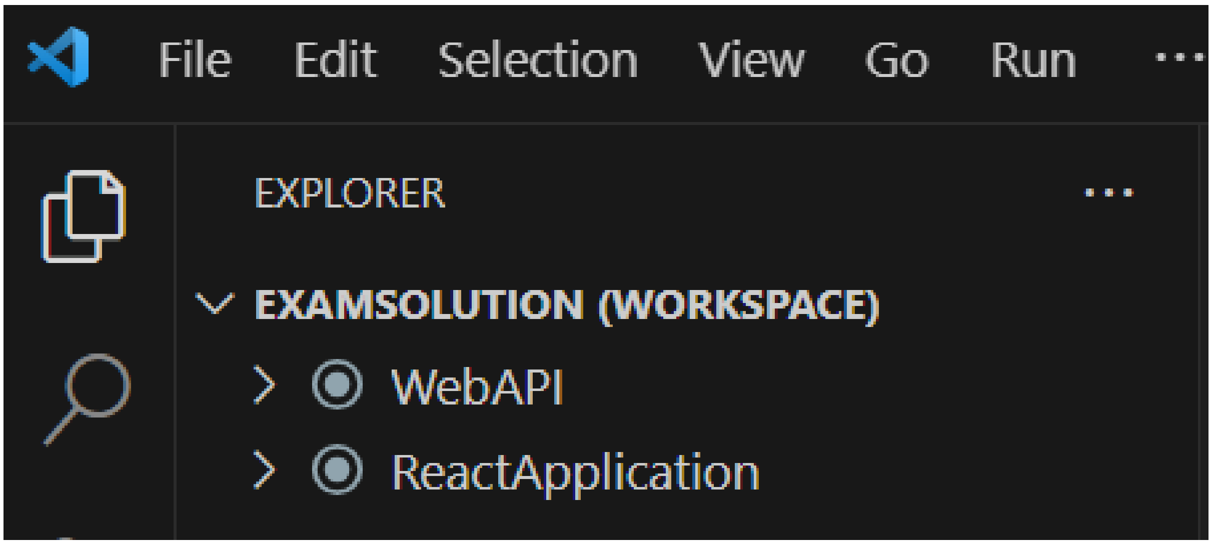
Size and complexity is an integrated part of the assessment (see assessment guide on last page). In the frontend part there are a lot of ways to implement the functionality, and it is encouraged to show that you have knowledge and skills through coding extra functionality.

Report Universal Design (UD, Norwegian: Universell Utforming)

Answer the questions below in your own words (no copy paste). Include print screen where you find it useful. Save the file in pdf format and include it inside the folder containing the rest of the solution. Size: around 200-300 words.

1. What is Universal Design about in context of a web page?
2. What is contrast about in UD? Test the contrast of a button with a contrast test tool such as this, and include a print screen of the result to include in the report: https://www.tpgi.com/color-contrast-checker/
3. How should a <img> be made according to UD?
4. Which role does semantic coding play in UD?
5. What is Usability?

The print screen below shows how it will look in Visual Studio Code when you have both the Web API project and the React project opened. These are two separate project folders. Both folders are put into a common folder, zipped, and uploaded to WISEFlow.



EXAMSOLUTION (WORKSPACE)
WebAPI
ReactApplication

Assessment guide exam DS3103 Webutvikling

For students, internal assessors, and external assessors.

The table below indicates how the exam will be assessed. Since the solution can be solved in many different manners and focus on different things the percentages can't/shouldn't be exact.

There is much more variation in what can be done in the frontend part than what can be done in the backend part.

Frontend (around 60%)	**Comment**
Component-based development in React JS, programming with JavaScript as taught in the course.	
React Routing.	
HTTP requests to Web API for CRUD.	
CSS and CSS framework (Bootstrap or Tailwind), responsive design for different sizes (mobile, tablet, laptop).	For the grid one can choose CSS3 Grid or the grid in the CSS framework.
React hooks including useState, createContext/useContext, useEffect etc.	
Backend (around 40%)	
Use of interfaces and classes	
Controllers and CRUD (Create, Read, Update, Delete) methods to Database.	
Picture upload to the images folder in wwwroot	
wwwroot API page which includes documentation about endpoints in the Web API and the information that can be retrieved.	This is the part referred to in the case which describes the Web API.
General things (included in the 100% above)	
Amount of code and complexity. Also, variation of code counts as something positive.	If in group more is expected.
Structured, tidy, good code (for example, not having unnecessary code repetition), modularization (for example breaking up code into functions/modules/services etc. especially in the JavaScript)	
Report Universal Design (weight of around 5%)	