DS3103 Webutvikling

Exam

Autumn 2024

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.
- The exam consists of a fullstack technical solution, a report, and a video of up to 1 minute showing the functionality on the web page.
- 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.
- An important part of this exam is to be able to put together all that you have learned in the course. Check the material given in the course (slideseries, tasks, codes from lectures and so on). If you begin including techniques outside of the course, you may not get points. It isn't allowed to copy paste finished code from the internet, co-students, ChatGPT etc. I.e. you are expected to show what you are able to do based on the material given in the course!

Focus areas of this exam.

1. HTML5 and CSS3
2. CSS Grid and CSS Framework (Bootstrap or TailWind) including responsive design
3. Universal Design
4. React and JavaScript (ES6+)
5. .NET/C# Web API with SQLite Database
6. HTTP requests from frontend to backend

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:

1. Which properties the information you are having in the solution should have.
2. What datatypes the properties should be
3. Which functionality you need.

When one knows how the information looks, for example through creating interfaces, it is possible to work with frontend and backend at the same time.

Case: TrumpVerse

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

First of all, this is not a joke! We have just received a call from the U.S. by an employee of the Trump administration.

They need a Web API with a database where they can store information given by Trump and the Trump administration. They also need a frontend solution which makes use of the Web API.

This is the information they need to store in the database:

- **TrumpThought:** Wise words and news from the president about politics, the state of the country, thoughts, philosophy, facts about the world etc.
- **TrumpMerch:** Diverse set of merchandise (products) that can be bought from the store.
- **TrumpStaff:** The administration needs to have an overview over the staff helping the president.

Don't forget to make documentation pages in wwwroot to help frontend developers understand how to use the Web API!

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

Best regards

Daglig leder

FullyStacked AS

22.11.2024

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:

1. Get all of something
2. Get something by id
3. Get something by other property than id, for example GetByName
4. Create something (including image upload)
5. Update something
6. Delete something

Other details

- The functionality in the frontend makes use of the above mentioned methods.
- The amount and quality of implemented functionality affects the grade.
- When you deliver the solution there must already be a database with at least 8 rows of information per table 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

Regarding obtaining a high grade: You should have 1 table (i.e. 1 Interface, 1 Model Class, 1 Controller, frontend code for that, and so on per group member) per group member, and full CRUD per table. The bigger the group, the more is expected in terms of size and complexity in the solution. If you want to expand further with more functionality and complexity you may add more tables, but they should also be connected to the case.

Size and complexity are 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 and video

This is a report where you will write about the technical aspects of your solutions as well as Universal Design. You will also deliver a video that showcases how you have created a functional solution.

Length: 300-500 words per person in the group.

Format: pdf and video

Report and video tasks

1. A video of up to 1 minute showing the functionality on the web page.
2. Overview of functionality you have made in frontend, and short about the techniques used in context of the functionality.
3. Write about what you have done in your frontend solution, code and design, to follow Universal Design (Universell Utforming).

Assessment guide exam DS3103 Webutvikling

For students, internal assessors, and external assessors.

The table below indicates how the exam will be assessed.

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

In this course knowledge into fullstack development is key. To receive credit for this solution, it is essential that both the frontend and backend are functional and able to interact with each other. A solution will for example not be considered complete if only the backend is functional, and therefore not receive full points by itself.

Frontend (around 50%)	Comments
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.	
Report (around 10%)	
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 (IIFE) etc. especially in the JavaScript)	