DS3103 Webutvikling

Exam

Autumn 2025

Practical information.

- This home exam can be solved in a group of 2, or in a group of 3.
- 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, and the report and video.
- **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.
- Regarding AI use see next page which is from Kristiania's rules/guidelines.

KI og fusk (from Kristiania's web pages)

Ved eksamen og obligatorisk aktivitet skal studentens kunnskap og ferdighet testes. Handlinger som tar sikte på å gi studenten et uberettiget fortinn, regnes som fusk. Oppgaver skal derfor være utarbeidet av studenten selv. Dersom studenten leverer inn noe som ikke er utarbeidet av studenten selv kan dette ansees som fusk.

Det vil også ansees som fusk dersom studenten har oppgitt kildehenvisninger som ikke eksisterer eller som et KI verktøyet har funnet på. Det er studentens ansvar å kvalitetssikre innhold i teksten, og litteratur og referanser foreslått av KI.

Studenten skal følge akademiske normer for redelighet, etterrettelighet og åpenhet, og tydelig angi hvordan og hvor KI-baserte programmer er brukt. Ved bruk av tekst, tabeller og annet fra andre kilder må dette refereres til på korrekt måte. Selv om "alle hjelpemidler" er tillatt må kilder oppgis og refereres til på korrekt måte.

https://egms.service-now.com/csp?sys_kb_id=1783efce9723a5108050f386f053afc5&id=kb_article_view&sysparm_rank=1&sysparm_tsqueryId=67fd093797bae6107d94be1de053af40

Focus areas of this exam.

1. HTML5 and CSS3
2. CSS Framework: TailWind CSS
3. Universal Design
4. React and JavaScript (ES6+)
5. .NET/C# Web API with SQLite Database
6. Connecting frontend and backend

Note: you will use TailWind for the ReactJS frontend project, and vanilla CSS for the API web page in backend.

Suggestion before starting coding.

Step 1 is getting an overview and plan! Create a task list to get an overview of what is and isn’t done.

The entire solution will be dependent on the information in the database. 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.

Have a clear overview of:

1. The tables, properties, and datatypes.
2. Which functionality you want to include and how it should work.

Case: SportsWorld

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

We have received a client that is a big sports company, SportsWorld, that needs a web application for handling purchases and sales of athletes for international special events which they arrange.

They handle different sports, but for this first round they want us to focus on only one of the following 4 sport types which you can choose from:

1. Football (soccer)
2. Tennis
3. Basketball
4. Mixed Martial Arts

Pages

The following page descriptions are for the core pages and core functionality to include in them.

Page 1. Administering athletes

Shows all registered athletes (purchased and not purchased). Search, edit and delete athlete.

Page 2. Register potential athlete

Registering a new potential athlete. When you first register the athlete, they have status of "not purchased" as default. See information about tables for more information.

Page 3. Dashboard for finances and purchase

Dashboard consisting of 3 different and separate components / sections:

- **Section 1. Financial situation.**
  - How much money the company has left
  - How many athletes have been purchased
  - Total spending so far
- **Section 2. Get more money (loan) from bank.**
  - The user should be able to write the amount of loan needed into an input field and click a button to increase the amount of money the company has.
- **Section 3. Purchase component:**
  - Show athletes that have yet not been purchased
  - Purchase athlete. Purchasing an athlete updates its purchased status and the financial situation.

For groups of 3 you are expected to add 2 more pages in addition to the 3 pages mentioned above for Venue. See *"Information needed…"* on next page. You can for example have one page for showing and searching for venues, and the other one for editing, adding, and deleting.

Information needed - the tables in the database

Table 1. Finance*:

- Id
- MoneyLeft
- NumberOfPurchases
- MoneySpent

Table 2. Athlete:

- Id
- Name
- Gender
- Price
- Image
- PurchaseStatus**

Table 3. Venue:

- Id
- Name
- Capacity***
- Image

**Finance will only have 1 row in the database as it is the info data for the one company.*

***PurchaseStatus is if SportsWorld has purchased the athlete or not. Logically a boolean value.*

****Capacity is the number of people that fit into the venue.*

Best regards

Daglig leder

FullyStacked AS

25.11.2025

Regarding group size and size of solution

Regarding obtaining a high grade: You should have 1 table per group member. I.e. 1 Interface, 1 Model Class, 1 Controller, frontend code for that, and so on, and full CRUD per table.

Note:

- Table 1 and 2 should be implemented by all groups
- A group of 2 doesn’t need to also add table 3.
- Table 3 should also be added for the groups of 3 that have as a goal to get a higher grade.

The bigger the group, the more is expected in terms of size and complexity in the solution. 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.

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.
- You are expected to give feedback to the user, for example, after successful or not successful save and so on.
- 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.

Report and video

Report (pdf)

This is a report where you will write about the technical aspects of your solutions as well as Universal Design.

1. Overview of functionality and techniques you have made and used in frontend (for example TypeScript, React hooks, axios and so on). Length: 50-100 words per person.
2. Explain how your frontend solution and backend solution are connected. Length: 100-150 words per person.
3. Write about what you have done in your frontend solution, code and design, to follow Universal Design (Universell Utforming). Length: 100-200 words per person.

Video

You will deliver a video of up to 1 minute that showcases as much of the functionality you have made. Audio isn't expected here.

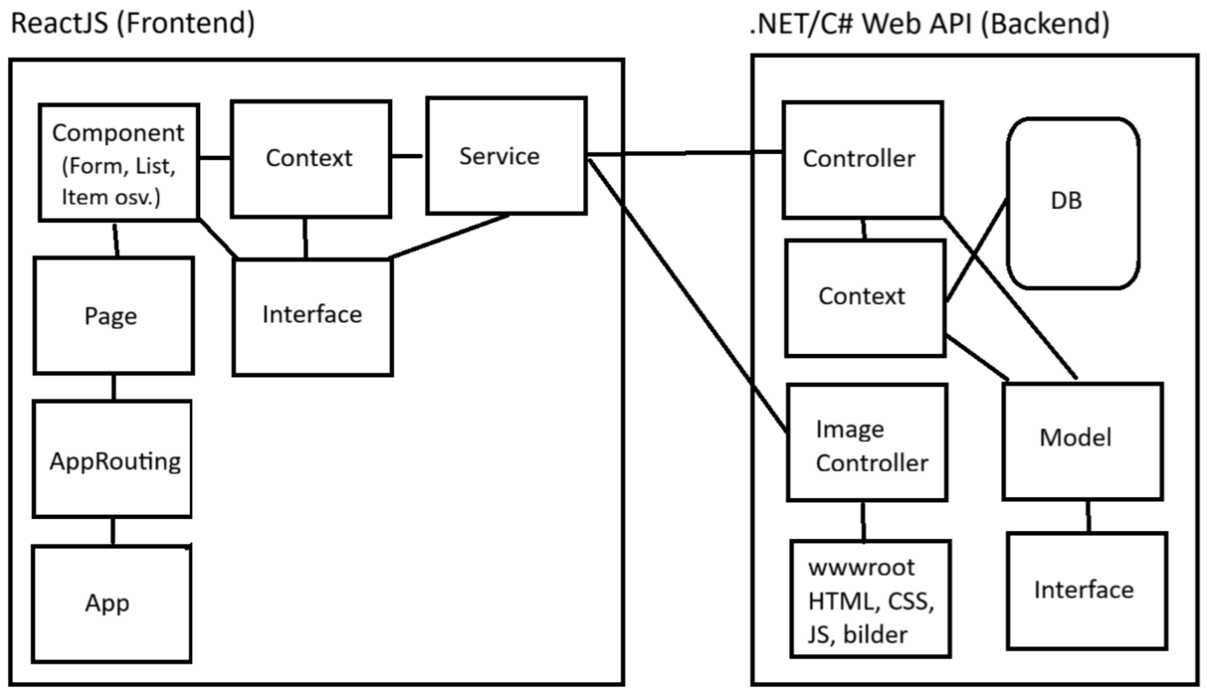
Assessment guide exam DS3103 Webutvikling

For students, internal assessors, and external assessors.

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.

The diagram below shows an overview of main techniques taught in React and .NET/C# Web API and how they ideally are connected.



The table below indicates how the exam will be assessed.

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.	
CSS3 and Tailwind, responsive design for different sizes (mobile, tablet, laptop). Styling will be assessed based on your understanding and application of styling techniques.	Use Tailwind CSS in the ReactJS project. Use vanilla CSS (i.e. no CSS framework) in the index.html in wwwroot (the Web API page).
React hooks including useState, createContext/useContext, useEffect etc.	
Backend (around 40%)	
Use of interfaces and model classes	
Controllers and CRUD (Create, Read, Update, Delete) methods to the SQLite Database.	
Picture upload to the images folder in wwwroot, using a dedicated image upload controller.	
wwwroot API page which includes documentation about endpoints in the Web API and the information that can be retrieved.	
Report and video (around 10%)	
General things (included in the 100% above)	
Amount of code and complexity. Also, variation of code counts as something positive.	
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)	
An important thing is that frontend and backend are connected and functioning together. If only either frontend or backend is delivered you can't get full score on the separate part.	