1. Exam task:  
   **A  web-based homecare appointment management tool**
   * where you let users (healthcare personnel, elderly) create/read/update/delete available days for healthcare appointment, and appointments with concrete tasks (e.g., assistance with daily living, medication reminders, shopping, or household chores)
   * for delivering the exam project no extra information is strictly required. Use your creativity to create the webapp which you think is reasonable. Design is part of the task.
   * [extra info](https://oslomet.instructure.com/courses/32014/pages/info-extra-info-about-the-exam-task-1-only-interesting-for-those-who-choose-task-1) is provided that serves as inspirations for development.

Extra info:  
**Homecare Appointment Management Tool: Exam Project**

1. **Background**

Homecare services are vital for helping older adults maintain their independence. However, managing appointments and communication between healthcare personnel and clients can often be inefficient. This project challenges you to build a web application that simplifies this process. Your solution should improve the communication and organization of homecare services, making them more accessible and user-friendly for everyone involved.

1. **Goal**

The primary goal is to develop a user-friendly, web-based tool for managing homecare appointments. The application could support different user roles and their associated functions.

* **Healthcare Personnel:** A user who schedules and manages available days for their clients.
* **Older Adult (Client):** A user who books, views, and manages their own homecare appointments within the available days given by the healthcare personnel, and write concrete tasks performed within the appointment, e.g., "medication reminder," "assistance with shopping"
* **Admin:**A user who can create/edit/everything.

1. **Core Functionality (MVP)**

Your web application must implement full CRUD (Create, Read, Update, Delete) functionality for two main entities:

**Entity 1: Available days given by the healthcare personnels.**

**Entity 2: Appointment with tasks booked by the clients.**

1. **Design and Creativity**

While the core functionality is a must, you are encouraged to use your creativity to design the user interface and overall user experience. The design is an important part of this assignment. Focus on making the application intuitive and easy to use for both older adults and healthcare personnel. You are not required to follow any specific external requirements; you get to decide on the details that make your web app a success.

**Important Note:** This project focuses on the programming aspect. No extra information is strictly required for you to deliver the exam project. The descriptions above should give you a clear enough picture.

1. **Opportunity**

One student group that excels in this project will have a unique opportunity to continue its development in a bachelor’s thesis. The extended project will involve the following:

* Expanding user roles and permissions.
* Integrating features for better communications between the above-mentioned users.
* Adding a calendar-based view for easier scheduling.

**Requirements**

**Basic requirements (to pass the exam):**

* The application should not crash.
* The application must have a front-end user interface with interactive form in the form of a web page.
* CRUD (create, read, update, delete) **must**be supported in the task for at least one entity in the task.
* A documentation that explains the project.

**Additional requirements (that serve as guidelines for improving quality):**

**Web application:**

* User experience, including:
  + Basic design, not plain text and bare buttons
  + Dynamic content
  + Friendly navigation
  + If the user experience is confusing or does not reflect the normal expectation of the given task, it will negatively affect the grade.
* Coding style, including:
  + Neat code, structured, and modularised
  + Understandable code, either self-explanatory or with suitable comments (in English)
  + If the program is buggy or has warnings (use dotnet build, npm run build to verify), it will negatively affect the grade.
* Functionality, including:
  + Database operation with at least 2 types of entities (including component and CRUD). The entity used for users authentication does not count here.
  + Conditional rendering and content filtering (client-side)
  + Forms (client-side)
  + Input validation (server and client-side)
  + Error handling and logging (server-side and client-side)
  + Repository pattern and DAL (server-side)
  + Asynchronous database access (server-side)
  + API service layer (separate http request handling from the business logic on client-side)
  + Unit testing (**server-side**), at least 8 tests, including the complete positive tests and negative tests for each CRUD method (where testing is meaningful) of 1 entity/table
  + Authentication and authorisation (server and client-side). Note: for some tasks you may see that role-based authentication is very suitable. It is not a mandatory functionality but counts as an extra functionality. You could assume all users as admins if you do not wish to program the role-based authentication.
  + If the requirements are finished in a low quality, or incomplete (e.g., for the requirement error handling and logging, if only part of the code is covered by error handling and logging), it will negatively affect the grades.
* In addition, it will be taken into account in the evaluation if you have beautiful design, extra rich user experience, and extra functionality (be creative).

**Documentation:**

* Range between 2000-5000 words (in English). Word counters are not strict principles but serve as guidance for being sufficiently detailed and not too verbose.
* Should clarify the requirements analysis, why they are needed, what parts are implemented and what are not.
* Should include some diagrams for the software architecture and database. The diagram should not be too general, e.g. only depicts backend, frontend, database, without detailed information on the architecture or database.
* Should clarify functionality and how they are coded.
* Should clarify necessary version information, such as node version, so that the project can be correctly set up by the examiner to run the code.
* Should clarify how to run your code and how to login, etc.
* Proper citations on references, detailed way of how chatbots are used, etc.

**Grading percentage distribution:**

* Web application: 80%.
* Documentation: 20%

**Important Notice:**

* The project should be implemented in .NET Core 8.0 for the backend and single page application functionality for the frontend.
* **Which version of Node.js is built with must be stated in the documentation. If this is not given, it may cause that the grader cannot run the project and affect the grades.**
* It is allowed to take code snippets and get inspiration from other solutions, but in order for this not to count as plagiarism, this must be documented.
* Grading guidelines follow the University Council's [grading guidelines.Links to an external site.](https://student.oslomet.no/en/examination-grading-system)
* In addition, it will be taken into account in the evaluation if you have beautiful design, extra rich user experience, and extra functionality (be creative!).
* **The solution must allow the examiner(s) to access it.**
  + **Either the solution allows the user to register, or the login** **information for users must be provided in the documentation. If the examiner cannot access the app because it is not possible to login, this will negatively impact the grade.**
  + If the solution runs on a cloud solution, the URL must be stated in the Documentation.
  + If the solution uses an API with a key, the key must accompany the submission, possibly an instruction on how to obtain one.