**Lexicon LMS**

The project you will be working on during the final module is a learning platform, a so-called LMS (Learning Management System), adapted for Lexicon's postgraduate courses. 1 An LMS simplifies and centralizes the communication between teacher, institution and student by gathering schedules, course material, other information, exercise tasks and submissions in one place.

You will from the beginning produce the system with database, back-end functionality and a well-thought-out front-end. This is called a "full-stack project" and aims to showcase your understanding of all parts of a web application and current systems. The project aims to test the breadth of your understanding and that you have a foundation to stand on regardless of future orientation within .NET.

**Product Description**  
The main task and goal of the system we are to build is to easily make course material and schedule for students. It also serves as a gathering place for assignments. In order for this to be possible, we also need smooth functionality for teachers to be able to easily administer these classes, pupils, schedules and documents. Because if it is not easy for the teacher to use the tool, then the students will never have the opportunity to use it.

The finished system is intended to cover basic functionality, but in a well-thought-out and elaborate manner. Less is more or less true of this type of application to be used daily. Unfortunately, an order to reach as broad a market as possible, most LMSs that are available today are heavily heavy and overcrowded by all possible functionality that one rarely has for - you should change this! Less is no need for pure functionality, but rather for perceived complexity. It may well be deep functionality, but the user should not need 14 options in each choice.

**Framework and techniques**  
The application must have a back-end built with C # .NET and MVC Core. The database will be built using the Entity Framework according to the code first method. Front-end should use Bootstrap 4. Dynamic front-end functionality should be written with javascript or javascrip framework.

**Entities, relationships and attributes (basic form)**  
Entities and attributes described below are a minimum, not an absolute description. Above all, the attributes will need to be expanded when you plan the system in more detail.

**User**  
The application must handle users in the roles of students and teachers, all of them having logins and accounts in the application. Users should be saved with at least one name and email address.

**Course**  
All students belong to a course which in turn has a course name, a description and a start date. Example of course name: ".NET Autumn 2019".  
  
**Module**  
Each course reads one or more modules, these have module names, a description, start date and end date. Examples of modules: "Database design", "Angular"

**activities**  
the modules in turn have activities, these activities can be e-learning passports, lectures, exercise sessions or other. The activities have a type, a name, a start / end time and a description.  
  
**Document**  
All entities above can hold documents; assignments from the students, module documents, general information documents for the course, module documents, lecture material or exercises related to the activities. These document entities should have a name, a description, a time stamp for the uploading event, and information about which user uploaded the file. If the document is a submission assignment, it must also keep time information about the deadline.

**Use cases (requirements)**  
These use cases are not comprehensive; Depending on the implementation, more detailed cases need to be developed to cover all practical functionality.  
  
An unregistered visitor should be able to:  
● Log in

**A student should be able to:**  
● See which course it is reading and what the other course participants are  
● See which modules it is reading  
● See the activities for a specific module (module schedule).  
● See if a specific module or activity has any documents attached to it and in this case download them.  
● See what exercise tasks it has received; if it has already been submitted, when it is to be submitted at the latest and if it is late.  
● Be able to upload files as practice submissions.

**A teacher should be able to:**  
● View all courses  
● See all modules included in a course  
● See all activities and documents a module contains  
● Create and edit users (teachers and students)  
● Create and edit courses  
● Create and edit modules  
● Create and edit activities  
● Upload documents for courses / modules / activities  
● Receive submission assignment

**Use cases (desirable)**  
An unregistered visitor should be able to:  
● Request a new password  
  
**A student should be able to:**  
● Receive feedback on assignments  
● Share documents with their course or module  
  
**A teacher should be able to:**  
● Provide feedback on assignments

**Front-end**  
The desire is that the front-end visually relate to Lexicon's graphic profile, this with log and colors that can be seen at http://www.lexicon.se. This should not be done in the smallest detail, but can be seen as a guideline.  
  
In addition to these purely aesthetic desires, the remaining front-end focus should be directed towards the user experience and to reduce the user's cognitive friction. The application should be responsive. Bonus points for a well-functioning mobile version.

**Working**  
**Scrum**  
The project must be carried out in groups with a scrum-based approach. We will work in 7-day sprints from Wednesday to Wednesday. A new sprint starts every Wednesday afternoon with a sprint planning where you set up a sprint-backlog, distribute the work and update your task-board.

Every day begins with a standup (daily scrum) where you briefly, one by one, discusses 1. what you have done since last standup, 2. what you plan to do next and 3. if it is something that blocks planned work. You hold the meeting in front of your task-board, and update accordingly.

During the following Wednesday morning, you finish the sprint with a sprint demo and a retrospective together with teachers and clients.

**Version Management**  
The project is to be version managed with git and GitHub.

**Tuning points and supply cables**  
During the course of the project, you are expected to report some elements before continuing. This is to avoid dead ends and maximize your effective development time.  
● Product backlog must be approved before you start implementation.

● ER Diagram must be approved before you start an implementation.  
● Wireframes for some key views must be approved before you start an implementation.  
● Sprint backlog must be approved before you start a new sprint.  
● Upon completion of the sprint, all ready-made changes must be demonstrated at sprint demo.

**Accounting**  
Detailed information on the accounting element will come later, but will include use cases, questions about code solutions, file management, detail solutions, error management, working methods etc.  
  
The system must be demonstrated at the projector through a round-trip based on some use cases. Then selected parts of the code base are reported, some questions about implementation and working methods are answered. This is probably followed by warm applause and happy exclamation comments and constructive criticism.  
  
Good luck!