Lexicon LMS

Projektet ni skall arbeta med under den avslutande modulen är en lärplattform, ett så kallat LMS[1] *(Learning Management System)*, anpassat för Lexicons påbyggnadsutbildningar. Ett LMS förenklar och centraliserar kommunikationen mellan lärare, lärosäte och elev genom att samla schema, kursmaterial, övningsuppgifter, inlämningar och övrig information på ett och samma ställe.

Ni skall från grunden producera systemet med databas, back-end funktionalitet (både ett MVC- och API-projekt) och ett genomtänkt front-end. Detta kallas ett ”full-stack projekt” och syftar till att visa upp er förståelse för samtliga delar av en webbapplikation och nutida system. Projektet ämnar att testa bredden av er förståelse och att ni har en grund att stå på oavsett framtida inriktning inom .NET.

Produktbeskrivning

Systemet vi skall bygga har som främsta uppgift och mål att enkelt tillgängliggöra kursmaterial och schema för elever. Det skall även fungera som en samlingsplats för inlämningsuppgifter. För att detta skall vara möjligt behöver vi även smidig funktionalitet för lärare att enkelt kunna administrera dessa klasser, elever, scheman och dokument. För om det inte är enkelt för läraren att använda verktyget, så kommer eleverna aldrig få chansen att använda det.

Programmet ska även via ett API kunna kalla på en databas som har information om kursmaterial. Elever ska kunna söka i denna databas. Lärare ska kunna söka och även redigera dess innehåll.

Det färdiga systemet är ämnat att framför allt täcka grundläggande funktionalitet, men på ett genomtänkt och genomarbetat sätt. *Less is more* är ofta sant när det gäller denna typ av applikationer som skall användas dagligen. Tyvärr, för att nå en så bred marknad som möjligt är de flesta LMS som finns tillgängliga idag enormt tunga och överbelamrade av all tänkbar funktionalitet som man sällan har användning för - detta skall ni ändra på! Less is more behöver inte nödvändigtvis syfta till ren funktionalitet, utan snarare om upplevd komplexitet. Det får gärna finnas djup funktionalitet, men användaren skall inte behöva 14 alternativ i varje val den gör.

Ramverk och tekniker

Applikationen skall ha en back-end byggd med C# .NET och MVC Core. Databasen skall byggas med Entity Framework enligt *code first*-metoden. Front-end skall använda Bootstrap 4. Dynamisk funktionalitet i front-end skall skrivas med javascript eller javascript ramverk.

---

[1] Mer information: https://sv.wikipedia.org/wiki/Lärplattform

Applikationen ska även ha ett publikt .NET Core API som integreras med MVC Core web gränssnittet.

Entiteter, relationer och attribut (grundform)

Nedan beskrivna entiteter och attribut är ett minimum, inte en absolut beskrivning. Framför allt attributen kommer behöva byggas ut när ni i närmare detalj planerar systemet.

MVC Applikationen:

Användare

Applikationen skall hantera användare i rollerna av elever och lärare, dessa skall alla ha inloggningar och konton i applikationen. Användarna bör sparas med minst ett namn och en e-postadress.

Kurs

Alla elever tillhör **en kurs och endast en** som i sin tur har ett kursnamn, en beskrivning och ett startdatum. Exempel på kursnamn: “NA21”.
Det är inget krav att lärare knyts till någon specifik kurs. (Men beroende på implementation kan det finnas skäl till att hantera det.)

Modul

Varje *kurs* har flera *moduler*, dessa har modulnamn, en beskrivning, startdatum och slutdatum. Exempel på moduler: “Databasdesign”, “Javascript” osv.
Moduler får inte överlappa varandra eller gå utanför kursen.

Aktiviteter

Modulerna i sin tur har *aktiviteter*, dessa aktiviteter kan vara e-learning pass, föreläsningar, övningstillfällen, inlämningsuppgifter eller annat. Aktiviteterna har en typ, ett namn, en start/sluttid och en beskrivning.
Aktiviteter får inte överlappa varandra eller gå utanför modulen.

Dokument

Alla entiteter ovan kan hålla dokument. Det finns flera olika typer av dokument.
Exempel på dokument: inlämningsuppgifter från eleverna, generella informationsdokument för kursen, moduldokument, föreläsningsunderlag eller övningsuppgifter kopplade till aktiviteterna.
Dessa *dokument entiteter* bör ha ett namn, en beskrivning, en tidsstämpel för uppladdnings tillfället samt information om vilken användare som laddat upp filen.

API Applikationen för information om kurslitteratur:

Författare

Har ett eller flera litterära verk. Ska sparas med ett förnamn, efternamn samt ett födelsedatum. Utåt exponeras födelsedatum som en ålder.

Litteratur

Kan vara böcker, blogginlägg, artiklar eller snarlikt. Har en eller flera författare och ett ämnesområde. De ska även ha en titel, utgivningsdatum, beskrivning och nivå (nybörjare, avancerad etc).

Ämne

Har inga eller flera litterära verk samt ett namn.

Use-cases (krav)

Dessa use-cases är inte heltäckande; beroende på implementation måste mer detaljerade fall tas fram för att täcka in all praktisk funktionalitet.

En icke-inloggad besökare skall kunna:

- Logga in

Både en elev och en lärare ska kunna:

- Söka efter kurslitteratur på titel, författare och ämnesområde
- Sortera författare efter ålder eller namn eller båda
- Sortera kurslitteratur

En elev skall kunna:

- Se vilken kurs denne läser och vilka de andra kursdeltagarna är
- Se vilka moduler denne läser
- Se aktiviteterna för en specifik modul (modulschema)
- Se om en specifik modul eller aktivitet har några dokument kopplade till sig och i sådant fall ladda ned dessa
- Se vilka inlämningsuppgifter denne har fått, om den redan är inlämnad, när den senast skall lämnas in och om den är försenad
- Ladda upp filer som övnings inlämningar

En lärare skall kunna:

- Se alla kurser
- Se alla moduler som ingår i en kurs
- Se alla aktiviteter och dokument en modul innehåller
- Skapa och redigera användare (lärare och elever)
- Skapa och redigera kurser

- Skapa och redigera moduler
- Skapa och redigera aktiviteter
- Ladda upp dokument för kurser/moduler/aktiviteter
- Ta emot inlämningsuppgifter
- Ladda upp information om kurslitteratur
- Redigera information om kurslitteratur

Use-cases (önskvärda men ses som extra om tid finns)

En icke-inloggad besökare skall kunna:

- Begära nytt lösenord

En elev skall kunna:

- Ta emot feedback på inlämningsuppgifter
- Dela dokument med sin kurs eller modul
- Få notifieringar när en lärare har gjort lagt upp nya dokument för kursen tex en nytt dokument
- Få notifieringar när en lärare har lämnat feedback på en inlämningsuppgift
- Registrera sig själv efter ha fått en inbjudan via mejl
- Skall kunna ta bort sig från systemet samt radera all information om sig som finns sparad enligt GDPR
- Se information om eventuell rekommenderad kurslitteratur

En lärare skall kunna:

- Ge feedback på inlämningsuppgifter
- Lägga upp nya ämnen till Apiet
- Ha tillgång till en statistiksida
- Lägga till en rekommendation till kurslitteratur för en kurs
- Klona en föregående kurs för att snabba på processen med att lägga upp en ny kurs
- Generera ett kursschema komplett med moduler och aktiviteter.
- Generera ett dokument för kurslitteratur

Auktorisera våra request till Apiet med hjälp av Bearer Tokens.

Front-end

Önskemålet är att front-end visuellt har ett enhetligt utseende. Gränssnittet ska vara byggt med Bootstrap. Titta på de komponenter som finns färdiga!

Utöver dessa rent estetiska önskemål skall resterande front-end fokus riktas mot användarupplevelsen och att minska användarens kognitiva friktion.

Systemet ska vara lättanvänt. Och det ska vara tydligt hur det fungerar.
Applikationen skall vara responsiv. Bonuspoäng för en välfungerande mobilversion.

Arbetssätt

Scrum

Projektet skall utföras i grupp med ett *scrum*-baserat arbetssätt. Vi kommer att arbeta i 7-dagars sprintar från måndag till måndag. En ny sprint startar varje måndag eftermiddag med en *sprintplanering* där ni sätter upp en sprint-backlog, fördelar arbetet och uppdaterar er *task-board*.

Varje dag inleds med en *standup (daily scrum)* där ni kort, en och en, avhandlar
1. vad ni gjort sedan förra *standup*,
2. vad ni planerar att göra fram till nästa och
3. om det är något som blockerar planerat arbete.

Ni håller mötet i Teams appen i gruppens kanal. Ni ska ha er task-board framme så ni visuellt kan se hur ni ligger till.

Under efterföljande måndag förmiddag avslutar ni sprinten med en sprintdemo inför de andra grupperna följt av ett retrospektiv.

Versionshantering

Projektet skall versionshanteras med git och GitHub.
Ni ska minst ha varsin personlig branch samt en master och en development.

Test

Testa minst tre valfria kontroller metoder. Testa gärna att skriva flera test det kan vara otroligt lärorikt.
Plus för att skriva testen först enligt TDD.

Avstämningspunkter och leverabler

Under projektets gång förväntas ni redovisa vissa moment innan ni fortsätter. Detta för att undvika återvändsgränder och maximera er effektiva utvecklingstid.

- *Produkt-backlog* skall godkännas innan ni startar implementation.
- ER-Diagram skall godkännas innan ni startar en implementation.
- Wireframes för några nyckelvyer skall godkännas innan ni startar en implementation.
- *Sprint-backlogg* skall godkännas innan ni startar en ny sprint.
- Vid avslutad sprint skall alla leveransklara ändringar demonstreras vid sprintdemo.

Planering

Under projektets uppstart ska ni börja arbeta med att planera arbetet. Ni ska ta fram dokumentation enligt punkterna i avstämningspunkter och leverabler.

Fundera på hur systemet ska fungera för en lärare. Vad är det primära en lärare är intresserad av? Hur skapas ett bra flöde för att skapa upp nya kurser med allt en sådan består av? Vad är ni som elever mest intresserade av när ni loggar in?
Hur ska ni navigera till all funktionalitet osv.

Redovisning

Detaljerad information om redovisning momentet kommer senare, men kommer att innehålla use-cases, frågor om kod lösningar, filhantering, detaljlösningar, felhantering, arbetssätt etc.

Systemet skall demonstreras genom en *round-trip* baserat på några use-cases. Sedan redovisas utvalda delar av kodbasen, några frågor om implementation och arbetssätt besvaras. Detta följs troligen av varma applåder och glada utrop kommentarer och konstruktiv kritik.

Lycka till!