Kapitel 1 –Baggrund og problemstilling

SmartLearning er et typisk CRUD program, hvor det er muligt at oprette, se, (rette) og slette studerende, lærer, kurser mv.

1. Bagrund

Programmet laves som en obligatorisk afleveringsopgave i faget avanceret programmering.

2. Problemstilling

> Smartlearning forespørger et nyt kursusadministrations (vi forsimpler det her) system til administrering af deres kurser, underviser, studerende og eksaminer.
>
> Kravene til systemet:
>
> 1. Der skal kunne oprettes og slettes kurser, underviser, studerende og eksaminer
>
> 2. Der skulle kunne tilmelde og afmelde studerende fra både et fag eller en eksamen
>
> 3. Der skal kunne tilknyttes underviser til forskellige fag. De fag underviserne er tilknyttet skal de automatisk være eksaminator ved.
>
> 4. Eksamen kan være online eller fysisk fremmøde
>
> 5. Et kursus har op til flere opgaver
>
> 6. Et kursus kan være tilknyttet flere underviser

Kapitel 2 – Afgrænsning mv.

I de følgende afsnit gennemgås emner der ligger uden for selve applikationen.

1. Afgrænsning

Det er en rimelig afgrænset problemstilling, som burde kunne løses uden at der skal afgrænses noget. Det er dog valgt at fravælge GUI og database, da det blev givet som en mulighed. Desuden fokuseres der ikke på unit-test eller exception handling.

2. Dataopsætning

Systemet fødes med nogle data, der gør testen lettere, men ellers afprøves alt via consolen.

3. DataStorage

Pt. Bliver data gemt i collections, hvilket kunne ændres til en database, hvis behovet viste sig at opstå.

4. Teknologi valg

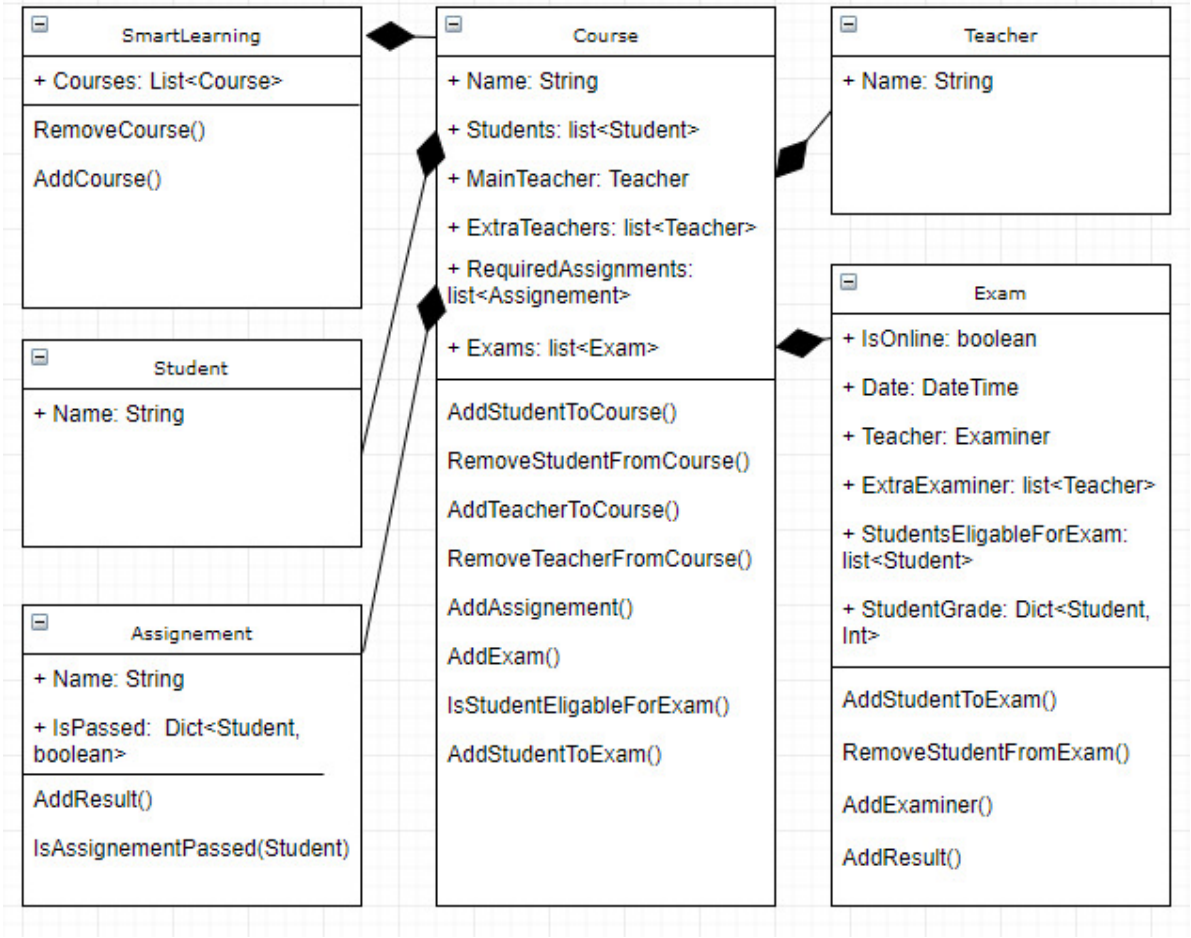
C#, og VS.

5. Teststrategi

Der er lavet manuel test for at sandsynliggøre at programmet fungere efter hensigten i alle henseender. Der er fundet flere bugs i processen, som er løst med det samme.

Kapitel 3 - Arkitektur

Der er lavet følgende UML diagram, for at vise strukturen af programmet.



Smartlearning oprettes som klasse for at holde en liste af kurser, som der er ønsket og her kan der tilføjes nye kurser eller fjernes eksisterende kurser. Der er hovedsagligt brugt lister fremfor arrays, da lister er hurtigere til at fjerne og tilføje elementer, hvor array er langsom. Lister er dog langsomme når der skal slås et element op, da der ikke er random acces som i en array – så her er der valgt at bruge et hash table (dictonary).

Når Smartlearning opretter et kursus sker det med AddCourse som instanciere et course objekt med de relevante felter. Generelt bliver lister initialiseret til en tom liste så der ikke risikeres at listen bliver overskrevet af en ny liste senere.

Al funktionalitet vedr. kursus – tilføjelse af lærer/studerende/opgave og/eller eksamener er alle placeret under kurser, hvor der instancieres nye objekter, hvor der er behov.

Når exam er tilføjet til et kursus sker det med tid, sted (online/fremmøde), main teacher fra course (evt. null) og instanciere tomme lister og dictonaries. Når en elev så skal tilføjes til en eksamen kaldes IsStudenEligableForExam som løber RequiredAssignments igennem og ser om IsAssignmentPassed er ”True” for alle assignments – Hvis alle opgaver er godkendte kan den studerende tilføjes til listen ”StudentsEligableForExam”

Der er tilføjet metode i assginment som kan kaldes når en lærer bedømmer en opgave og for en god ordensskyld er der lavet en collection under exam til at holde resultatet.

Man kunne evt. ændre så Students list<Student> modtog et student interface i stedet og så have Student implementere det interface – Samme er gældende for Teacher. Det vil hjælpe til at lave en mere loosly coupled application, men virker lidt overkill for denne applikation.

SmartLearning instanceres som en del af programmet og kan ikke slettes.

I opgave teksten står der specifikt at man skal kunne slette kurser, studerende mv. – hvor jeg i min løsning kun fjerner objekter fra lister. Da de objekter der bliver fjernet fra listerne ikke længere har en reference svare det til en implicit sletning, da garbage collectoren før eller siden vil slette objekter der ikke bliver refereret til.

Kapitel 4 - Koden

Er vedlagt i zip filen.