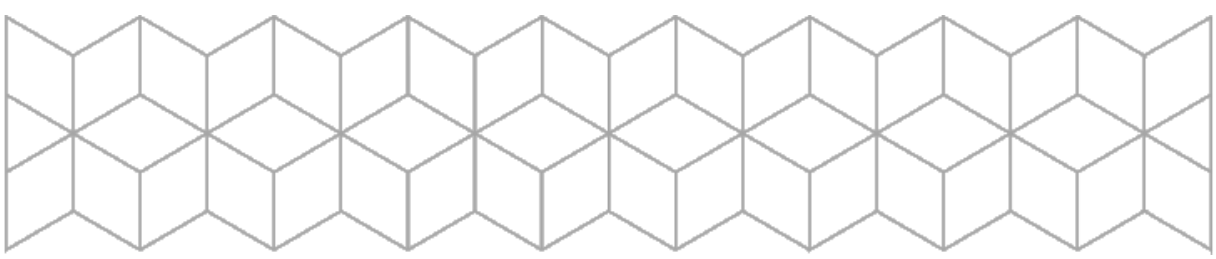
Høgskolen i Østfold

SENSORVEILEDNING

Emnekode:	ITF11012
Emnenavn:	**.NET (Industriell programmering i .NET)**
Eksamensform:	**Individuelt prosjekt**
Dato:	2019.05.31
Faglærer(e):	Øyvind Øhra
Eventuelt:	



Eksamen består av en prosjektinnlevering.

Det forventes en forkunnskap til kurset tilsvarende gjennomføringen av kurset objektorientert programmering i Java.

Fokuset til kurset går ut på å lære seg relevante ferdigheter, for å utføre programmering i en profesjonell sammenheng. Det betyr at studentene skal vise at de behersker:

- Inndeling av kode i flere lag, i.e. brukergrensesnitt-, tjeneste- og databaselag.
- Strukturering av koden i ulike prosjekter
- Benytte ulike typer kode-prosjekter
- Korrekt navngiving av kode-elementer; alt fra prosjekter, namespaces og ned på metode og parametere
- Beherske OOP-prinsipper i programmeringsspråket C#
- Utvikling av tjenester etter REST-arkitekturen
- Utvikle kode som er robust og vedlikeholdbar, derunder validering av input og beskyttelse mot SQL-injection, samt god lesbarhet, struktur og korrekt bruk av .NET rammeverket
- Korrekt feilhåndtering
- Korrekt bruk av ressurser, e.g. filsystem og database
- Dokumentasjon av koden vha. XML-kommentarer i koden
- Korrekt bruk av .NET og C#-språket innenfor bl.a. områdene;
    - Grunnleggende egenskaper ved C#
        - datatyper, og forståelse av Value vs. Reference Types
        - kontrollflyt; conditional, loop, (jump)
        - Enumerations
        - Namespaces
        - Class vs. Struct
        - Ulike typer data members og functional members
        - Base Classes
        - System.Object
        - System.String
        - Extension metoder
    - Arv i C#
        - Klasser og arv
        - Constructors and Inheritance
        - Aksess og andre modifiers, e.g. internal, virtual, static, partial, …
        - Interfaces
    - Defensiv programmering
        - Try*X* vs. try-catch, e.g. benytte TryParse i tillegg til å håndtere uventede avvik
        - Stenge ressurser så fort som mulig, e.g. using og/eller finally
        - Robust feilhåndtering, e.g. når skal feil kastes/fanges, bruk av egne feilklasser
        - Korrekte datatyper og kontroller ved input
        - Validering av parametere, både fra brukergrensesnitt og i metodekall, herunder kaste korrekte feil-klasser
    - Generics
        - Benytte Generics implementasjoner fremfor spesifikke der det er relevant
        - Lage egen Generics kode
        - Begrense Generics
        - Generics Interface
    - Grupper av objekter

        - Arrays og Tuples
        - Collections
        - Korrekt bruk, samt utvikle egne klasser, benytte og lage Generics funksjonalitet
        - Bruk av LINQ på grupper av objekter, fremfor primitive for-each løkker
    - Typer
        - Korrekt bruk av operatorer
        - Implisitt vs eksplisitt cast
        - Utvikle egne typer (klasser og structs) med operatorer
        - Strukturell vs referanse sammenligning
    - Asynkron programmering
        - Korrekt navngiving av egne metoder som er asynkrone (Async-suffix)
        - Benyttet TAP-pattern korrekt
        - Tillate kansellering
        - Korrekt feilhåndtering
        - Både benytte asynkrone metoder og utvikle egen asynkron kode
        - Vise asynkron parallell prosessering, e.g. hente flere filer samtidig
        - Benytte asynkron programmering til å oppnå en responsiv App
    - Data-aksess
        - Både vise bruk av EF og ADO.NET
        - Benytte både attributter og fluent-api for å styre opprettelsen av modellen
        - Benytte 1:M og M:N relasjoner
        - Benytte både lazy og eager loading
        - Alle CRUD operasjoner
        - Det teller positivt å benytte en rik datamodell, attach/detatch, concurrency og transaksjoner
    - RESTful-e tjenester
        - Korrekt navngiving av ressurser (controller)
        - Korrekt bruk av Verb og (http) Status-koder
        - Benytte attributter i koden, e.g. Route, HttpGet, …
        - Vise ulike tjenester, e.g. både database aksess og filaksess
        - Bygge tjenestene korrekt mhp validering, feilhåndtering, returverdier,…
        - Benytte 3dj-parts tjenester, e.g. karttjenester, kognitive tjenester, …
        - Benytte JSON, og korrekt konvertering til fra JSON
    - Delegates, Lambdas, and Events
        - Benytte egne delegater mot rammeverkskode, for å løse kodeoppgaver mer effektivt
        - Benytte delegater i ulike sammenhenger, e.g. event, parametere til generiske metoder, callbacks
        - Programmere egne delegater, i ulike sammenhenger
        - Utvikle egen Generics funksjonalitet som benytter delegater
        - Benytte Func/Action fremfor å kode egne delagater
        - Korrekt bruk av lamda-uttrykk, e.g. expression bodies, anonyme metoder
        - Lage egne event-er for signalering mellom ulike komponenter i løsningen
        - Benytte korrekt event args, lage egne event args
    - UWP App
        - Dele App opp i korrekte lag, e.g. M-V-VM
        - Benytte korrekte UWP-kontrollere

- Responsiv
- Intuitiv (evt. godt forklart i beskrivelsen)
- Korrekt bruk av binding, e.g. ikke kode for å fylle GUI
- 2-veis binding, converters, i.e. Boolean-2-Visibilty-Converter
- Bruk av INotifyPropertyChanged og Observable collection, utvikle egne klasser med INotifyPropertyChanged
- Korrekt bruk av control-templates, i.e. ItemTemplate
- Notifications
- Toast
- Splashscreen
- Settings
- Holde på tilstand mellom kjøringer
- Det teller positivt at brukergrensesnittet skalerer fint mellom ulike formfaktorer, men vil ikke vektlegges. Kandidaten kan velg å «låse» størrelsen til Full-HD-Landscape

Krav til innleveringen

Det skal levere én zip-fil. Denne skal inneholde all kildekode* og et dokument som minimum inneholder:

- Kandidatens navn, kandidatnummer, prosjektets tittel
- Krav for å kunne kjøre løsningen
- Beskrivelse av deler som ikke er ferdige
- Kjente feil
- Beskrivelse av løsningen og dens funksjonalitet

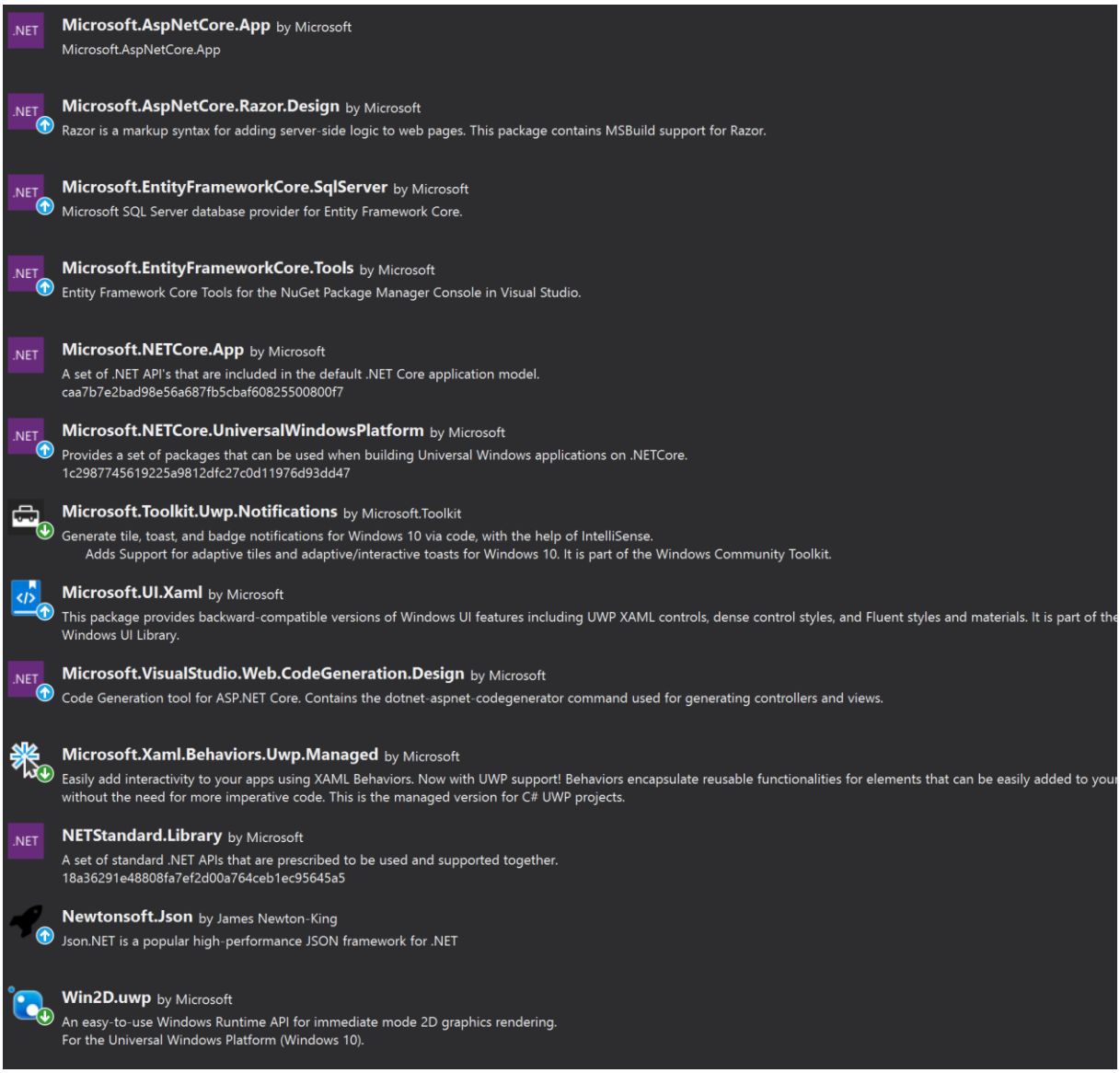
*Det teller positivt hvis kandidaten har fjernet alle overflødige filer fra løsningen som leveres inn, e.g. temporære filer, NuGet-nedlastninger, binærfiler, .vs-folder, …

Følgende er satt som absolutte krav. Avvik fra disse kravene vil normalt føre til karakteren F, men dette kan avvikes hvis det er særskilte grunner til det.

- C# / XAML
- Visual Studio 2017
- Windows 10 UWP App (Windows 10, versjon 1809 (10.0; Build 17763)
- .NET Core 2.1 (ikke benytte .NET Framework)
- Database: vise alle CRUD operasjoner, mot Donau databasen
- Det skal være mulig å åpne all kildekode fra én «solution» fil
- Man skal kunne starte (Run) løsningen i «Debug mode» ved å trykke F5**
- Det skal ikke benyttes annen 3je parts komponenter, enn de som er angitt som gyldige (NuGets) på Canvas***

**Det trekkes ikke hvis sensor må sette start-up prosjektene selv, før løsningen startes.

***Tillate NuGets er gitt i skjermdump-en nedenfor. Hvis NuGet er "**by Microsoft**" er det greit å benytte den, selv om den ikke står i listen under. Listen under er den samme som er hentet fra eksempel "09-Sample-Uwp".



Evaluering

Evalueringen vil bli gjennomført i forhold til kravene som er stilt i oppgavebeskrivelsen.

En typisk gjennomsnittlig innlevering (karakter C), vil inneholde en applikasjon som:

- Inneholder en vesentlig del av kriteriene utover det som er satt som minimumskrav
- Har fornuftig oppbygning av kode og struktur
- Har tydelig skille på modell, logikk og grensesnitt (MVVM)
- Har en datamodell med flere tabeller og relasjoner mellom disse
- Benytter .NET rammeverket fornuftig, og har en fornuftig bruk og oppdeling av:
    - Namespaces, prosjekter, klasser, metoder etc.
- Har fornuftig bruk av OOP-prinsipper
- Er responsiv
- Er av god størrelse implementasjonsmessig, som tilsvarer jevn jobbing med prosjektet i en til to måneder i et 10 studiepoengsfag

En typisk innlevering med ståkarakter (karakter E), vil inneholde en applikasjon som:

- Inneholder minimumskravene fra oppgavebeskrivelsen

- Minimum re-implementerer eksempel-oppgaver, som er lagt ut av faglærer på Canvas, for et annet behov, i.e. ikke er en ren kopi
- Benytter .NET rammeverket fornuftig, og har en fornuftig bruk og oppdeling av:
    - Namespaces, prosjekter, klasser, metoder etc.
- Er av tilstrekkelig størrelse implementasjonsmessig, som tilsvarer jevn jobbing med prosjektet i en til to måneder i et 10 studiepoengsfag

---

Karakterbeskrivelse

A	Fremragende	Fremragende prestasjon som klart utmerker seg. Kandidaten viser svært god vurderingsevne og stor grad av selvstendighet.
B	Meget god	Meget god prestasjon. Kandidaten viser meget god vurderingsevne og selvstendighet.
C	God	Jevnt god prestasjon som er tilfredsstillende på de fleste områder. Kandidaten viser god vurderingsevne og selvstendighet på de viktigste områdene.
D	Nokså god	En akseptabel prestasjon med noen vesentlige mangler. Kandidaten viser en viss grad av vurderingsevne og selvstendighet.
E	Tilstrekkelig	Prestasjonen tilfredsstiller minimumskravene, men heller ikke mer. Kandidaten viser liten vurderingsevne og selvstendighet.
F	Ikke bestått	Prestasjon som ikke tilfredsstiller de faglige minimumskravene. Kandidaten viser både manglende vurderingsevne og selvstendighet.