Tekstskrift: Calibri

Projektrapport

IBA

Dkpbi23a2a

Eksamensprojekt – ”personregistrering”

Gruppe: Mohammad, Nickolaj, Fahad, Allan

Forside:

Login: JECO og SKQU

Indholdsfortegnelse

[Resumé: 2](#_Toc165528834)

[Problemanalysen 2](#_Toc165528835)

[Research 2](#_Toc165528836)

[1. Mindre research efter eksisterende 2](#_Toc165528837)

[Problemformulering 3](#_Toc165528838)

[Problemstilling 3](#_Toc165528839)

[Målgruppeanalyse 4](#_Toc165528840)

[User Story 4](#_Toc165528841)

[Kravanalyse 6](#_Toc165528842)

[Projektforløb 8](#_Toc165528843)

[Afgrænsning 8](#_Toc165528844)

[Succeskriterier 8](#_Toc165528845)

[Kravspecifikation 9](#_Toc165528846)

[Design 10](#_Toc165528847)

[Design værktøjer 10](#_Toc165528848)

[Adfærds analyse 11](#_Toc165528849)

[Arkitektur design 11](#_Toc165528850)

[Koncept evaluering 12](#_Toc165528851)

[Teknologi beskrivelse 12](#_Toc165528852)

[Mock-up 12](#_Toc165528853)

[Prototype 12](#_Toc165528854)

[ Værktøjer 12](#_Toc165528855)

[ Brugergrænseflade 12](#_Toc165528856)

[ Systemudvikling 13](#_Toc165528857)

[ Validering og verificering 13](#_Toc165528858)

[Diskussion 13](#_Toc165528859)

[ Problemformulering 13](#_Toc165528860)

[ Kvalitet 13](#_Toc165528861)

[ Metoder 13](#_Toc165528862)

[ Perspektivering 13](#_Toc165528863)

[Konklusion 13](#_Toc165528864)

[Litteraturliste 13](#_Toc165528865)

[Bilag: 13](#_Toc165528866)

# Resumé:

# Problemanalysen

Research:

1. Mindre research efter eksisterende personregistrerings systemer til uddannelsesinstitut og vi skulede til IBAs eget system og har taget det som udgangspunkt. Der valgt at have typer ansatte Studerende, Pedel/kantine og Underviser, de skal have adgang til lidt forskellige informationer.
   1. Studerende: Egen information (Fornavn, Efternavn, Mail), hold/kursus, lokale.
   2. Underviser: Egen information, som studerende, hold/kursus, lokale og tilmeldte studerende til kurser.
   3. Pedel/Kantine: Egen information (fornavn, Efternavn, Mai).
2. ”Et simpelt login”: Dette refererer typisk til design og funktionalitet af GUI. Og indebærer ikke nødvendigvis fraværet af sikkerhedsforanstaltninger som kryptering. Vi fortolker det, at selve brugergrænsefladen er ukompliceret og let at bruge. Der udføres dybere research senere i analysen vedrørende sikkerhed og kryptering.
3. Gemme/Læse IP-adresser: Ligger op til at kunne justere konfigurationsindstillinger uden at skulle re-deploy eller direkte ændre i selve applikationen. Et nyttigt område, hvor der er brug for fleksibilitet angående brug af forskellige databaser. Altså en dynamisk ændring uden dybere indgreb i funktionaliteten. Her vil der også fortages dybere research senere i analysen.
4. Diverse udleverede teknologier, har vi kørt research på for at styrke disse. Og for at kunne implementere sikkerhed og OOP ind.

Problemformulering:

Hvordan kan vi designe og udvikle et brugervenligt, sikret og skalerbart personregistreringssystem baseret på MS SQL Server og C# til en uddannelsesinstitution i Danmark, som opfylder følgende krav:

1. Muliggør adgang for studerende til deres personlige oplysninger, tilmeldte kurser og oplysninger om lokaler.
2. Giver undervisere adgang til personlige oplysninger, kurser, lokaledetaljer og lister over tilmeldte studerende.
3. Giver øvrige personale adgang til personlige oplysninger.
4. Tillader administration af applikationen gennem en admin adgang, hvor brugere (ansatte og studerende), kurser, lokaler og kursustilmeldinger kan håndteres.
5. Inkluderer at logge alle aktiviteter for at lette fejlsøgning og opklaring af misforståelser, som kun adm. har adgang til

Udviklingen skal omfatte:

* En C#-baseret Enterprise-applikation til databaseadministration med en simpel UI, som er kompatibel på tværs af Mac og Windows.
* En C# Web Server-applikation til håndtering af read-only dataanmodninger fra en klient browser-applikation via et REST API, med logning af forespørgsler i XML/JSON-format.
* Dynamisk konfigurerbar databaseforbindelsesindstillinger gemt i JSON-format for nem vedligeholdelse og skalerbarhed.

Problemstilling:

Udviklingen af et personregistreringssystem for en dansk uddannelsesinstitution kræver en omhyggelig tilgang til design og funktionalitet for at sikre sikker og effektiv håndtering af ansatte og studerendes data. Overvejelser skal omfatte:

1. **Database Design**: Skabe en databasestruktur i 3NF, der understøtter nøjagtig og effektiv håndtering af datamængden.
2. **Enterprise Applikation**:
3. Udvikling af en brugervenlig brugergrænseflade.
4. Implementering af en enkel login-dialog til autentificering.
5. Evnen til at håndtere og udnytte Stored procedures, Funktionel procedures og Views i databasen.
6. Fleksibilitet i konfiguration af databaseforbindelser, med mulighed for at ændre forbindelsesdetaljer som IP og login oplysninger.
7. Funktionalitet til dynamisk at læse og gemme konfigurationsdata, såsom IP-adresser, fra JSON filer.
8. **C# Web Server Applikation**:
9. Konstruktion af en C# webserver, der sikrer pålidelig kommunikation med en klient-baseret browser-applikation.
10. Brug af HTML, CSS og JavaScript til at udforme en interaktiv og responsiv brugergrænseflade.
11. Implementering af et Rest API for datatransmissioner, som både modtager og sender data i JSON format.
12. Garanti for read-only adgang til databaseinformation gennem brug af SQL-views, fra Bruger-GUI.
13. Detaljeret logning af alle henvendelser til webserveren i JSON format til understøttelse af sporbarhed og fejlfinding.
14. **Sikkerhed**:
15. Anvendelse af sikkerhedstiltag for databeskyttelse inklusive datakryptering, brugeradgangsstyring og overholdelse af databeskyttelsesregler.
16. **Test og Dokumentation**:
17. Gennemførelse af omfattende systemtests for at bekræfte dataflows og komponentintegreringens integritet.
18. Grundig dokumentation af alle systemaspekter, som tjener som referencemateriale for nuværende og fremtidige brugere og udviklere.
19. Grundig dokumentation af alle systemaspekter, som tjener som referencemateriale for nuværende og fremtidige brugere og udviklere.

**Delkonklusion: Problemanalysen**

Gennemgangen af vores indledende research har afdækket vitale aspekter af systemdesign og funktionalitetskrav for udviklingen af et personregistreringssystem til en uddannelsesinstitution. Med inspiration fra IBAs eget system og under inddragelse af specifikke brugerroller, som studerende, undervisere og pedel/kantine personale, har vi identificeret de grundlæggende adgangsbehov til information. Dette omfatter ikke kun personlige oplysninger og kursusdetaljer, men også lokationsdata, der er kritiske for hverdagens logistiske udfordringer på en uddannelsesinstitution.

Vores tilgang til 'Et simpelt login' understreger vigtigheden af en brugervenlig, men sikker adgangsstyring. Det skal være let at navigere.

Desuden har behovet for en fleksibel systemkonfiguration ført os til at overveje dynamiske konfigurationsindstillinger gemt i JSON-format. Dette vil tillade os at tilpasse systemforbindelser uden behov for genudrulning, hvilket optimerer vedligeholdelsesprocessen og understøtter en skalerbar anvendelse af forskellige databaser.

Endelig har vores research på diverse teknologier, herunder sikkerhed og objektorienteret programmering (OOP), forberedt os på at tage højde for både nuværende og fremtidige sikkerhedsudfordringer, samtidig med at vi sikrer, at systemet kan udvikle sig sammen med teknologiske fremskridt og ændrede brugerbehov.

# 

# Målgruppeanalyse

User Story:

**Bruger**: Administrator

**Handling**: Oprette, Ændre, Slette undervisere, studerende og pedel/kantine til systemet, samt lokaler og kurser og have adgang til log.

**Mål**: Registrere lokaler, kurser, undervisere, studerende og pedel/kantine i systemet for at sikre korrekt håndtering og opbevaring af deres data.

**User Story US-A1:**

Som administrator ønsker jeg at kunne oprette undervisere, pedel/kantine, studerende, til systemet for at sikre alle relevante oplysninger (lokaler og kurser) er korrekt registreret og opbevaret i databasen.

**User Story US-A2:**

Som administrator ønsker jeg at kunne oprette lokaler/kurser/ tilføjet underviser og studerende, at sikre alle relevante oplysninger er korrekt registreret og opbevaret i databasen.

**User Story US-A3:**

Som administrator ønsker jeg at kunne slette brugere, lokaler, kurser, for at sikre alle relevante oplysninger er korrekt registreret og opbevaret i databasen.

**User Story US-A4:**

Som administrator ønsker jeg at kunne ændre brugere, lokaler, kurser, for at sikre alle relevante oplysninger er korrekt registreret og opbevaret i databasen.

**User Story US-A5:**

Som administrator ønsker jeg adgang til Loggen over systemet, for at kunne fejlfinde/overvåge systemet, for at sikre alle relevante oplysninger er korrekt registreret og opbevaret i databasen.

**Bruger**: Studerende

**Handling**: At Studerende kan se deres egne data

**Mål**: At give den studerende adgang til at se deres egne data for at kunne holde styr på deres tilmeldinger og kurser.

**User Story 2 US-S:**

Som Studerende, ansatte og Pedel/Kantine ønsker jeg at kunne se mine tilmeldte lokaler og kurser for at have en oversigt over mine aktuelle studieplanaktiviteter.

## Kravanalyse:

**Bruger**: Undervisere

**Handling**: At undervisere kan se deres egne data og de tilmeldte studerende i deres kurser **Mål**: At give de undervisere mulighed for at se deres egne personlige data samt en oversigt over de studerende, der er tilmeldt deres kurser, for at kunne administrere undervisningen effektivt.

**User Story US-U:**

Som undervisere ønsker jeg at kunne få adgang til mine personlige data samt en liste over de studerende, der er tilmeldt mine kurser for at kunne administrere undervisningen og følge op på de studerendes fremgang.

**Bruger**: Pedel/kantine

**Handling**: At pedellen/kantinen får adgang til en oversigt over de aktuelle aktiviteter og arrangementer på skolen samt oplysninger om antallet af studerende og personale til stede på et givent tidspunkt.

**Mål**: At give pedellen/kantinen et nemt og hurtigt overblik over skolens aktiviteter og mængden af mennesker til stede for at kunne planlægge og forberede ressourcer effektivt.

**User Story US-PK**

Som pedel/kantine ønsker jeg at kunne få adgang til en oversigt over skolens aktiviteter og antallet af tilstedeværende studerende og personale på et givent tidspunkt, så jeg kan planlægge og forberede ressourcer effektivt.

**Funktionelle krav FR-:**

* **FR-US-A**: Brugeren skal kunne logge ind på systemet ved hjælp af et simpelt login-dialogvindue.
* **FR-US-A**: Brugeren skal have adgang til en formular til at indtaste oplysninger om den nye medarbejder eller studerende.
* **FR-US-A**: Systemet skal validere de indtastede oplysninger for at sikre korrekthed.
* **FR-US-A**: Efter godkendelse skal de indtastede oplysninger gemmes i databasen.
* **FR-US-A**: Systemet skal kunne behandle og lagre ikke-fast information som f.eks. IP-adresse i JSON-filer.
* **FR-US-A**: Brugeren skal have mulighed for at ændre IP-adresse, loginoplysninger osv. Til MS SQL Serveren.
* **FR-US-S**: Brugeren skal kunne logge ind på systemet med deres personlige loginoplysninger.
* **FR-US-S**: Efter vellykket login skal brugeren have adgang til en oversigtsside, der viser deres tilmeldte lokaler og kurser.
* **FR-US-S:** Brugeren skal kunne filtrere eller søge efter specifikke oplysninger, f.eks. kurser eller datoer.
* **FR-US-S**: Systemet skal vise oplysningerne på en brugervenlig måde, f.eks. i form af en liste eller et kalenderformat.
* **FR-US-U** Underviseren skal kunne logge ind på systemet med deres personlige loginoplysninger.
* **FR-US-U** Efter vellykket login skal underviseren have adgang til en oversigt over deres personlige data samt en liste over de studerende, der er tilmeldt deres kurser.
* **FR-US-U**: Underviseren skal kunne filtrere eller søge efter specifikke oplysninger, f.eks. studerende eller kurser.
* **FR-US-U**: Systemet skal vise oplysningerne på en brugervenlig måde, f.eks. i form af en liste eller en tabel.
* **FR-US-PK**: Pedellen/kantinen skal kunne logge ind på systemet med deres personlige loginoplysninger.
* **FR-US-PK**: Efter vellykket login skal pedellen/kantinen have adgang til en oversigt over skolens aktiviteter samt antallet af tilstedeværende studerende og personale på et givent tidspunkt.
* **FR-US-PK**: Pedellen/kantinen skal kunne filtrere eller søge efter specifikke oplysninger, f.eks. aktiviteter eller datoer.
* **FR-US-PK**: Systemet skal vise oplysningerne på en brugervenlig måde, f.eks. i form af en liste eller et kalenderformat.

**Ikke-funktionelle krav NFR-:**

* **NFR-US-A**: Systemet skal være baseret på en MS SQL Server opbygget med database tabeller i 3NF normalisering.
* **NFR-US-A**: Enterprise-applikationen skal udvikles i C#.
* **NFR-US-A**: UI-teknologien kan være valgt efter udviklerens præference, og det skal fungere på både Mac og Windows.
* **NFR-US-A**: Systemet skal kunne håndtere stored procedures, funktioner, views osv.
* **NFR-US-A**: Kommunikationen mellem Web Server og klienten skal være baseret på Rest API og JSON-beskeder.
* **NFR-US-A**: Web serveren skal logge alle request i JSON-baseret tekstfil.
* **NFR-US-A**: C# Web Server-applikationen skal udvikles og implementeres til at håndtere kommunikationen mellem en readonly klient browser applikation og MS SQL Serveren ved hjælp af HTML, CSS og JavaScript.
* **NFR-US-S:** Brugergrensesnittet skal være let at navigere og intuitivt at bruge for studerende med forskellige tekniske færdigheder.
* **NFR-US-S:** Systemet skal have hurtig respons og være tilgængeligt med minimal ventetid for at give en god brugeroplevelse.
* **NFR-US-S:** Brugerdata skal være sikret og beskyttet mod uautoriseret adgang eller misbrug, f.eks. gennem kryptering af data under transmission og opbevaring.
* **NFR-US-S:** Systemet skal være kompatibelt med forskellige webbrowsere og enheder, så studerende kan få adgang til deres data fra forskellige platforme, herunder computere, tablets og smartphones.
* **NFR-US-U:** Brugergrensesnittet skal være let at navigere og intuitivt at bruge for undervisere med forskellige tekniske færdigheder.
* **NFR-US-U:** Systemet skal have hurtig respons og være tilgængeligt med minimal ventetid for at give en god brugeroplevelse.
* **NFR-US-U:** Brugerdata skal være sikret og beskyttet mod uautoriseret adgang eller misbrug, f.eks. gennem kryptering af data under transmission og opbevaring.
* **NFR-US-U:** Systemet skal være kompatibelt med forskellige webbrowsere og enheder, så undervisere kan få adgang til deres data fra forskellige platforme, herunder computere, tablets og smartphones.
* **NFR-US-PK:** Brugergrensesnittet skal være let at navigere og intuitivt at bruge for pedeller/kantiner med forskellige tekniske færdigheder.
* **NFR-US-PK:** Systemet skal have hurtig respons og være tilgængeligt med minimal ventetid for at give en god brugeroplevelse.
* **NFR-US-PK:** Brugerdata skal være sikret og beskyttet mod uautoriseret adgang eller misbrug, f.eks. gennem kryptering af data under transmission og opbevaring.
* **NFR-US-PK:** Systemet skal være kompatibelt med forskellige webbrowsere og enheder, så pedeller/kantiner kan få adgang til deres data fra forskellige platforme, herunder computere, tablets og smartphones.

**Delkonklusion: Målgruppeanalyse**

Gennemførelsen af en dybtgående målgruppeanalyse har givet os en detaljeret indsigt i de forskellige brugergruppers specifikke behov og krav til systemet. Ved at definere nøje udformede User Stories for hver brugergruppe, har vi formået at tilpasse systemdesign og funktionaliteter for at møde hver enkelt gruppes forventninger og krav.

Administratorer, som udgør kernen i systemhåndteringen, har brug for omfattende funktionaliteter til at administrere undervisere, studerende, pedel/kantinepersonale, samt lokaler og kurser. Deres adgang til systemlogs er afgørende for effektiv fejlsøgning og systemovervågning, hvilket understreger behovet for en robust logningsmekanisme.

Studerende skal kunne tilgå personlige oplysninger og kursusrelaterede data, hvilket kræver et intuitivt og sikret brugerinterface, der understøtter deres behov for at navigere igennem deres akademiske informationer uden tekniske besværligheder.

Undervisere har behov for adgang til både personlige data og omfattende informationer om de studerende, de underviser, hvilket gør det nødvendigt at have en pålidelig og sikker datatilgang. Dette muliggør en mere effektiv håndtering af undervisningsforløb og understøtter en bedre opfølgning på studerendes præstationer.

Pedel/kantinepersonale kræver adgang til oversigter over skolens daglige aktiviteter og arrangementer for effektiv ressourceplanlægning og -forvaltning. Systemet skal derfor være i stand til at tilbyde realtidsdata, der kan hjælpe dem med at forberede og tilpasse deres arbejde efter skolens behov.

Disse indsigter fra målgruppeanalysen har været fundamentale for udformningen af funktionelle og ikke-funktionelle krav. Ved at sikre, at disse krav er opfyldte, stiler vi efter at skabe et brugervenligt, sikret og skalerbart personregistreringssystem, der er skræddersyet til at håndtere de unikke behov hos en uddannelsesinstitution.

# Projektforløb

Afgrænsning:

Der er på forhånd nogle områder der er afgrænset for.

* Der arbejdes med følgende brugere: Administratorer, Undervisere, studerende og øvrige ansatte. Og generelt Administrator, Underviser og studerende.
* Der er kun taget højde for et undervisningssted.
* En dynamisk kalender funktion er der afgrænset for.
* Administrator delen skrives Windows Pc.
* Login sikkerhed – Der valideres på password med længde og tegn ellers afgrænses sikkerheden i Login funktionen

Succeskriterier:

* Fuldstændighed: Alle definerede funktion er implementeret og fungerer som specificeret.
* Brugertilfredshed: Et niveau hvor brugere kan bruge applikationen og få glæde at funktioner.
* Sikkerhed: Systemet opfylder definerede sikkerhedskrav, som ikke er afgrænset.
* Skalerbarhed og vedligeholdelse: At applikationen kan ændre database adgang uden kodning, at der nemt at tilføjes GUI. Altså modularitet
* Opgave (backlog): Gruppen godkender opgaveløsningen, når opgavekrav er fuldført efter aftale.
* Milepæl (backlog): Når alle opgaver i milepælen er godkendt af gruppen.
* Rapport: Når der er en fyldestgørende rapport jf. Til kravene i projektbeskrivelsen. Og en klar sammenhæng mellem problem, løsning og applikation.
* Applikationen: Kravspecifikationen er opfyldt og godkendt af gruppen. Og direkte sammenhæng mellem Problemstilling og resultat.
* Projekt: Når præsentationen er vel forsvaret og overstået.

Kravspecifikation:

1. Formål:

* Softwaren skal udvikles til at administrere ansatte, studerende, lokaler og kurser på en uddannelsesinstitution i Danmark. Formålet er at effektivisere håndteringen af persondata og sikre en struktureret database, hvor Ansatte og studerende har view rettigheder.

2. Funktionelle krav:

Systemet skal:

* Oprettelse, redigering og sletning af ansatte, studerende, lokale og kurser.
* Understøtte opdeling i forskellige brugerroller med adgangsrettigheder.
* Gemme og håndtere persondata såsom navn, adresse, kontaktoplysninger osv.

3. Ikke-funktionelle krav:

Systemet skal:

* Systemet skal være baseret på en MS SQL Server med database-tabeller i 3NF normalisering.
* Brugergrænsefladen skal fungere på enten Mac eller Windows.
* Systemet skal være sikkert, skalerbart og brugervenligt.

4. Brugergrænseflade:

Brugergrænsefladen skal inkludere:

* Login-dialog for adgang til systemet.
* Nem navigation mellem forskellige funktioner (ADM).
* Mulighed for at søge efter specifikke personer (ADM).

5. Systemgrænser og -begrænsninger:

Systemet skal:

* Bestræbelse på at opfylde lovgivningsmæssige krav for opbevaring og behandling af persondata.

6. Testkrav:

Softwaren skal testes med:

* Enhedstests for hver funktion i systemet.
* Integrationstests for at sikre korrekt samspil mellem forskellige moduler.

**Delkonklusion over Projektforløb**

Projektets nøje afgrænsede omfang sikrer, at udviklingsressourcerne fokuseres på de mest essentielle funktioner og brugergrupper, hvilket er afgørende for en effektiv implementering. Succeskriterierne er veldefinerede og dækker alle aspekter af projektets livscyklus, fra funktionalitet og brugertilfredshed til sikkerhed og skalerbarhed, hvilket understøtter en målrettet og kvalitetsorienteret udviklingsproces. Kravspecifikationen omfatter både funktionelle og ikke-funktionelle krav, der klart definerer systemets nødvendige kapaciteter og sikrer, at det endelige produkt er sikkert, effektivt og brugervenligt. Dette fokuserede og grundigt planlagte projektforløb lægger fundamentet for et vellykket system, der kan håndtere komplekse krav fra forskellige brugergrupper på en uddannelsesinstitution.

# 

# Design

Planlægning og skitsering af systemets arkitektur og brugerinteraktion: XXXXXXXXXXXX er under renskrivning

**Designfasen**

Er afgørende for at sikre, at systemet ikke blot fungerer teknisk, men også opfylder brugernes behov og forventninger på en intuitiv og effektiv måde. Nedenfor er en detaljeret gennemgang af, hvad der forlanges i de forskellige dele af designfasen:

Design værktøjer:

**UML-værktøjer: (Unified Modeling Language)**: Systemdiagrammerne fra UML er brugt til at visualiser det forskellige abstrakte niveauer, samt synliggøre flowet og interaktionen, helt oppe fra User Story ned til Class og Deploy diagrammer.

**Prototypeværktøjer**: Figma er anvendt som prototypeværktøj, da det er et avanceret designværktøj. Et webbaseret værktøj uden tung installation. Robuste prototyping-funktioner, der tillader interaktive demoer. Brugervenligt og tilgængelig for designere på alle niveauer.

## Adfærdsanalyse

Der er brugt egne erfaringer som: **Brugerscenarier**: Udvikling af detaljerede scenarier, der beskriver, hvordan forskellige brugertyper vil interagere med systemet. Der er lagt op til IBAs egen Personregistersystem og gruppens egne erfaringer. Ved at dele **Brugerrejsen** op i forskellige scenarier. Adfærden er beskrevet i User Story og ”rejsen” i tilhørende Sequence diagrammer.

I projektet er der valgt User Story, som beskrivelse brugerscenarierne og efterfølgende Sequence diagrammer, for at beskrivelse af interaktionen i applikationen.

**Arkitektur Design**

I designfasen af vores system har vi valgt at implementere Model-View-Controller (MVC) arkitekturen. Da denne struktur er bygget til bla. Webbaseret applikation med tilhørende database. Der er en klar separation i lagene og derved en skarp ansvarsfordeling. Modularitet er høj, som er med til at forbedre Skalerbarheden, forenkle vedligeholdelsen og gør kode genbrugelig. Samt gør adskillelsen af grænseflade og forretningslogik det lettere at teste. Hvis det lykkedes at implementere arkitekturen efter hensigten, at det være med til at sikre et robust system. I store træk ser MVC ud på følgende måde: **Model** er ansvarlig for data og forretningslogikken, altså håndtering, opbevaring, hentning og validering af data, samt sikrer, at alle forretningsregler bliver opfyldt. **View** genererer brugergrænsefladen, præsenterer data for brugeren, altså brugerinteraktion.

**Controller** fungerer som bindeleddet mellem model og view. Den styrer brugerinput og omdanner det til kommandoer for modellen eller visningen.

Yderlige arkitektoniske beslutninger, er beskrevet I strukturelle UML diagrammer:

Her er vises et Component diagram, der viser overordnede MVC-strukturen

XXXXXXX

Her er klasse diagrammet over den ønskede applikation.

XXXXXXX

**Koncept Evaluering**

Gennem initial research og problemanalyse er det identificeret, at nøglekomponenterne i et effektivt personregistreringssystem for en uddannelsesinstitution omfatter skræddersyede brugerroller, en sikker login-mekanisme og fleksible systemkonfigurationer. Vores valg af MS SQL Server og C# som teknologisk fundament styrker disse elementer ved at tilbyde robuste sikkerhedsfeatures og skalerbarhed, sammen med brede udviklingsmuligheder.

Ved at implementere MVC-arkitekturen opnår vi en høj grad af modularitet, og de beskrevne fordele i foregående afsnit. Definerede brugerroller som studerende, undervisere, og administrativt personale, hver med specifikke adgangsrettigheder, reflekterer deres daglige interaktionsmønstre og sikrer en effektiv håndtering af persondata og logistiske oplysninger. Dette fremmer ikke kun systemets effektivitet men også dets sikkerhed ved at indskrænke adgang til følsomme data.

En fleksibel styring af konfigurationsdata, forenkler systemet uden gentagne udrulninger eller kompileringer, hvilke kan belaste oppetiden for systemet

Ved en release, efter gennemgående eller gennemgribende test, for at sikre både de tekniske og funktionelle krav og de reelle bruger behov, er det vitalt at finjustere systemet, for at sikre dets relevans og effektivitet.

Systemet eller konceptet, som omfatter de strategiske valg bygget på den foregående analyse, er med til at sikre ”Personregistersystem” konceptet, som et effektivt og yderst brugbart system for brugerne.

**Teknologi Beskrivelse**

For Personregistreringssystem

Systemet til personregistrering i en uddannelsesinstitution er udviklet med en række moderne og robuste teknologier, der tilsammen danner et effektivt og sikret miljø for både brugere og administratorens interaktioner. Her er en detaljeret beskrivelse af de teknologiske valg fordelt på forskellige aspekter af systemet:

1. Programmeringssprog og Frameworks

C# og .NET 7:

Controller/Webserver & Model Niveau: Systemets backend, herunder forretningslogik og webserver funktionalitet, er implementeret i C#, benytter .NET 7 framework. Dette valg giver robust performance, sikkerhed og nem integration med andre Microsoft-produkter og -tjenester.

C# blev valgt på grund af sin robusthed og objektorienterede natur, hvilket gør det til et fremragende valg til udvikling af komplekse og skalerbare applikationer. Dets syntaks er let at lære og forstå, hvilket bidrager til produktiviteten under udviklingsprocessen. Derudover er C# bredt anvendt i industrien, hvilket betyder, at der er et rigt økosystem af udviklingsressourcer og support til rådighed.

.NET-frameworket blev valgt på grund af dets omfattende sæt biblioteker og værktøjer, der gør det muligt at udvikle alt fra desktopapplikationer til web- og serverapplikationer. Med .NET har udviklere adgang til en række kraftfulde funktioner, såsom indbygget sikkerhed, ydeevneoptimering og integreret fejlhåndtering, hvilket forenkler udviklingsprocessen og øger produktiviteten.

Anvendelsen af HttpListener i C# muliggør oprettelsen af en simpel HTTP-server, der passer perfekt til denne applikation. Denne tilgang eliminerer behovet for tunge webservere som IIS eller Apache, hvilket resulterer i en lettere og mere effektiv løsning. Desuden udnytter systemet SQL Server til at gemme og validere brugeroplysninger. Mens SQL Server er en pålidelig og udbredt databaseplatform, kan det være overkill for mindre applikationer eller applikationer, der ikke kræver en fuld SQL Server-database. Derfor kunne valget af en mere letvægtsdatabase som SQLite have været en alternativ tilgang, især hvis der er behov for en mere agil og skalerbar løsning.

Der udnyttet asynkron programmering i C#, hvilket gør det muligt at håndtere flere anmodninger effektivt uden at blokere tråden. Dette øger systemets ydeevne og responsivitet, hvilket resulterer i en bedre brugeroplevelse.

UI: Administrativt Interface: Adm-grænsefladen er også udviklet i C#, hvilket sikrer konsistens og effektivitet i udviklingsprocessen og understøtter komplekse interaktioner og datavisualisering direkte inden for applikationen. Men er valgt at udvikles på Windows computer.

HTML/CSS og JavaScript: View Del: Brugergrænsefladen for slutbrugere er udviklet med HTML, CSS, og JavaScript, hvilket giver en responsiv og interaktiv oplevelse. Dette sikrer, at applikationen er tilgængelig på tværs af forskellige enheder og browsere og systemer.

2. Database: Microsoft SQL Server: En robust og skalerbar database, MS SQL Server, er valgt til at håndtere alle datalagringsbehov. Dette system er ideelt til komplekse forespørgsler og stor datamængde, hvilket er essentielt for en uddannelsesinstitution med mange brugere og transaktioner. Samt et velkendt system, der bruges af stort set over 80% af alle brugere

3. Logging og Fejlhåndtering:

NLog: Til logging af systemaktiviteter og fejlhåndtering anvender vi NLog, et fleksibelt og kraftfuldt logging framework. Dette værktøj hjælper med at spore applikationens tilstand og diagnostisere problemer, hvilket er vitalt for vedligeholdelse og support.

Try/catch: til at fange de fejl der vil komme under kørslen af applikationen, og for at håndteringen af disse ikke stopper systemet.

4. API og Serverteknologi

REST API med .NET 7: Der er implementeret RESTful API'er ved brug af .NET 7, hvilket tillader effektiv og standardiseret kommunikation mellem klientapplikationer og vores server. Dette understøtter en moderne, service-orienteret arkitektur.

5. Udviklingsværktøjer og Versionsstyring

Visual Studio og GIT:

-Udviklingsmiljø: Hele systemet er udviklet i Visual Studio, et avanceret udviklingsmiljø der tilbyder omfattende værktøjer og integrationer for at øge produktiviteten og forbedre kodekvaliteten.

Versionsstyring: Git bruges til versionsstyring, hvilket sikrer robust håndtering af kodeændringer og samarbejde mellem udviklerne. Dette er essentielt for at kunne skalere udviklingsteamet og håndtere kodebasen effektivt.

Ved at vælge disse teknologier har vi sikret, at vores personregistreringssystem ikke kun er robust og sikret, men også fleksibelt tilpasset fremtidens krav. Disse teknologiske valg reflekterer vores engagement i at levere en højkvalitets løsning, der kan understøtte både de daglige operationer og langsigtede mål for uddannelsesinstitutionen.

**Mock-up**

Som afledt af de foregående analyser, krav, afgrænsninger, valg, er følgende Mock-up udfærdigede i Figma. De er designet til at vise det visuelle layout og design af Personregistersystem, uden at indeholde funktionel kode eller interaktive elementer. Der er tale om visuelle MockUp, der viser grundlæggende funktioner på brugersiden. Mockup’erne er brugt som en ledetråd i prototypeudviklingen:

Login:

XXX

XXXX

XX

Designfasen afsluttes typisk med udviklingen af en eller flere **mock-ups**, som er visuelle repræsentationer eller prototyper af det endelige produkt. Dette giver alle stakeholders en konkret fornemmelse af, hvordan det færdige system vil se ud og fungere.

**Afsluttende bemærkning**

Designfasen er fundamentet, der sikrer, at systemet ikke blot bliver funktionelt, men også optimeret til brugernes interaktioner og behov. Ved at anvende passende værktøjer og teknikker sikrer man, at både systemets adfærd og struktur er godt forberedt før selve udviklingsfasen begynder. Mock-ups fungerer som det sidste trin til at bekræfte, at designopgaverne er fuldført, og at projektet er klar til overgangen til implementeringsfasen.

**Delkonklusion:**

Valget af MVC-arkitekturen til vores systemprojekt er afgørende for at sikre en modulær, vedligeholdelsesvenlig og skalerbar løsning, som aktivt støtter effektiv udvikling og en positiv brugeroplevelse. Denne arkitektur sikrer, at vores system er robust nok til at håndtere både nuværende og fremtidige krav.

Vores strategiske valg af systemarkitektur og teknologier er dybt forankret i en omfattende forståelse af målgruppens specifikke behov. Implementeringen af en struktureret MVC-arkitektur, sammen med nøje udvalgte teknologier og sikkerhedsforanstaltninger, placerer os ideelt til at udvikle et sikret, brugervenligt og skalerbart personregistreringssystem. Projektets ultimative succes vil afhænge af, hvordan disse planer realiseres i praksis og bliver modtaget og anvendt af de endelige brugere i deres daglige funktioner.

Ved at vælge disse teknologier har vi sikret, at vores personregistreringssystem ikke kun er robust og sikret, men også fleksibelt og tilpasset fremtidens krav. Disse teknologiske valg reflekterer vores engagement i at levere en højkvalitets løsning, der kan understøtte både de daglige operationer og langsigtede mål for uddannelsesinstitutionen.

# Prototype

Evaluering og justering af design baseret på mock-up og de erfaringer gruppe har fået i projektet.

Ude fra den grundlæggende analyse og udvikling, samt Mock-up, der er udviklet en prototype, der kan bruges til præsentation, for at vise nogle af de funktionaliteter applikationen skal have for at opfylde de krav, der ligger i et personregistersystem.

Prototypen har mange mangler og er ikke færdig produceret og da produktionen af applikationen er en iterativ proces, kan der stadig forekomme ændringer under den sidste del af produktionen.

Værktøjer:

MS SQL og Azure Data Studio: Som er 100% kompatibel med alle andre Microsoft produkter og da der i projektet bruges C# og .Net framework har det været oplagt at bruge dette, det har så vist sig at det også er et værktøj der er nemmere at arbejde med og i. Der er mange sikkerhedsfunktioner, og det er forholdsvist nemt at skalere. Effektivt valg.

Der er flere værktøjer som muligheder og mulige forbedringer for systemet. For eksempel kunne man overveje at erstatte HttpListener med mere robuste og vedligeholdte webservere som ASP.NET Core. Derudover kunne man undersøge alternativer til SQL Server, såsom en letvægtsdatabase som SQLite, der er lettere at administrere og ikke kræver en separat database-server. Men grunden manglende erfaring i dette er det ikke blevet taget i brug.

Figma: Mock-ups er brugt til at præsentere designkoncepter. Og har hjulpet os i at holde fokus. Nu er der ingen stakeholdere eller lignende, ellers havde de været brugbar for første præsentation inden udvikling af prototypen. Er yderst brugbart værktøj.

Brugergrænseflade: Optimering af interaktionsdesign.

Systemudvikling: Implementering af ændringer i systemet.

Validering og verificering: Sikring af at systemet møder alle kravspecifikationer.

# Diskussion

## Refleksion over projektets forløb og resultater:

Problemformulering: Problemformuleringen var noget vag fra projektbeskrivelsen, men efter noget research og skelen til IBAs egen registersystem, kom der er brugbar problemformulering på plads. Efter en defineret ”bruge- adgang”, altså hvem skal have adgang og hvad skal de forskellige profiler have rettigheder til.

Kvalitet: Kvaliteten er ikke voldsom høj, da prototypen ikke er færdig udviklet, men det der er udviklet, virker robust. Det synlige kvaliteten af GUI er meget lav, da der ikke er lagt energi i det visuelle, men mere fokuseret på systemudvikling og integration ad de forskellige dele i apllikationen.3

Metoder:

Metodologi

I projektet har vi anvendt Software Development Life Cycle (SDLC), en struktureret, faseopdelt tilgang til softwareudvikling. Denne metodologi sikrer en grundig behandling af projektets krav og udviklingsfaser, som beskrives nedenfor:

1. Analyse og Kravspecifikation: Denne indledende fase fokuserer på at indsamle og præcisere systemkravene. Her defineres de funktionelle og ikke-funktionelle krav, der styrer de efterfølgende faser af projektet.

2. Systemudvikling: Byggende på de indsamlede krav fra fase 1, omhandler denne fase udformning af systemarkitekturen og detaljeret design. Resultaterne fra denne fase er grundlæggende for resten af udviklingsprocessen og er nøje forbundet med designafsnittet i projektet, hvilket sikrer kohærens og kontinuitet i projektarbejdet.

3. Implementering og Kodning: I denne fase oversættes designspecifikationerne til faktisk kode. Vores tilgang har involveret udviklingen af en fungerende prototype, der demonstrerer kernefunktionaliteter og systemlogik.

4. Test: Selvom en dedikeret testfase traditionelt følger efter implementeringen, har vores tilgang været at integrere test som en løbende aktivitet gennem hele udviklingsprocessen, herunder enhedstest og integrationstest, hvilket har forbedret kodekvaliteten og reduceret antallet af fejl i det færdige system.

5. Udrulning (Release): Denne fase indebærer implementering af det færdige system i en produktionssætning, hvor slutbrugerne får adgang til applikationen. Det er en kritisk fase, hvor systemets ydeevne og stabilitet bliver evalueret i et realtidsmiljø.

6.Der er reelt også en 6. fase som er omkring vedligehold og opdatering, denne fase ligger oftest i forlængelse af en service kontrakt.

Ydermere har vi integreret elementer fra \*\*SCRUM\*\* metodologien for at fremme fleksibilitet og responsivitet i udviklingsprocessen. Dette omfatter daglige stand-up møder, brug af en produkt-backlog og en done-log samt en iterativ tilgang, hvor teamet kontinuerligt evaluerer og tilpasser arbejdet baseret på projektets fremdrift og tilbagemeldinger. Denne hybride tilgang har vist sig effektiv ved at kombinere SDLC's strukturerede ramme med SCRUM's agilitet, hvilket understøtter en dynamisk og målrettet projektforvaltning.

Denne metodologi understøtter ikke blot en disciplineret fremgangsmåde til softwareudvikling men fremmer også en adaptiv og brugerorienteret proces, hvor der er plads til løbende evaluering og justering af både design og funktionalitet, hvilket sikrer et endeligt produkt, der lever op til både brugernes og projektets forventninger.

Perspektivering:

Fremtidige forbedringer og potentielle nye features. En naturlig perspektivering i udviklingen af applikation, vil være at tilføje en dynamisk kalender, som kan tilgås fra uddannelsesstedet, men som samtidig kan bruges aktivt af de studerende og ansatte, med deres egne og mere lokale aftaler. Måske med en mobil online version, som en mini timemanagement app, set fra brugernes point of view. Et godt værktøj, der kan give beskeder fra uddannelsessteder, fra underviseren, og fra studerende og mellem dem.

En anden features kunne være at lokale reservationer kan foretages gennem systemet / app., så der vises hvilket lokaler er ledige. Med en rettighed hierarki at uddannelsesstedet kan overtrumfe en reservation af nogle studerende. Det kan gradueres, så det lykkedes.

Visuelt er en opgradering i etiske og visuelle en selvfølgelighed og anses ikke for at

# Konklusion

(Opsummering af projektets resultater og læring.)

En erfaring: Det er nemt at vedligeholde, det blev synligt ved at logge funktionen i Model laget blev implementeret efter koden, var klar. Og det var nemt og overskueligt, samt Login funktionen kom på som det sidste. Og når applikationen er struktureret på en overskuelig og signende måde, med sigende og nye kommentar, er det lige til at udvide koden med nye funktioner.

En erfaring: Nu er der blevet arbejder med både MySQL og nu her i projektet MS SQL og erfaringen er, at det er betydelig nemmere at arbejde med MS SQL AZURE-database og clientværktøjerne.

# Litteraturliste

# Bilag:

Arbejdsfordeling