Tekstskrift: Calibri

Projektrapport

IBA

Dkpbi23a2a

Eksamensprojekt – ”personregistrering”

Gruppe: Mohammad, Nickolaj, Fahad, Allan

Forside:

Login: JECO og SKQU

Indholdsfortegnelse

[Resumé: 2](#_Toc165528834)

[Problemanalysen 3](#_Toc165528835)

[Research 3](#_Toc165528836)

[1. Mindre research efter eksisterende 3](#_Toc165528837)

[Problemformulering 3](#_Toc165528838)

[Problemstilling 4](#_Toc165528839)

[Målgruppeanalyse 7](#_Toc165528840)

[User Story 7](#_Toc165528841)

[Kravanalyse **Fejl! Bogmærke er ikke defineret.**](#_Toc165528842)

[Projektforløb 13](#_Toc165528843)

[Afgrænsning 13](#_Toc165528844)

[Succeskriterier 13](#_Toc165528845)

[Kravspecifikation 14](#_Toc165528846)

[Design 16](#_Toc165528847)

[Design værktøjer 16](#_Toc165528848)

[Adfærds analyse **Fejl! Bogmærke er ikke defineret.**](#_Toc165528849)

[Arkitektur design **Fejl! Bogmærke er ikke defineret.**](#_Toc165528850)

[Koncept evaluering **Fejl! Bogmærke er ikke defineret.**](#_Toc165528851)

[Teknologi beskrivelse **Fejl! Bogmærke er ikke defineret.**](#_Toc165528852)

[Mock-up **Fejl! Bogmærke er ikke defineret.**](#_Toc165528853)

[Prototype 22](#_Toc165528854)

[ Værktøjer 22](#_Toc165528855)

[ Brugergrænseflade 23](#_Toc165528856)

[ Systemudvikling 23](#_Toc165528857)

[ Validering og verificering 23](#_Toc165528858)

[Diskussion 23](#_Toc165528859)

[ Problemformulering 23](#_Toc165528860)

[ Kvalitet 23](#_Toc165528861)

[ Metoder 24](#_Toc165528862)

[ Perspektivering 25](#_Toc165528863)

[Konklusion 26](#_Toc165528864)

[Litteraturliste 26](#_Toc165528865)

[Bilag: 26](#_Toc165528866)

# Resumé:

Denne rapport beskriver udviklingen af et avanceret personregistreringssystem for en uddannelsesinstitution, med det formål at forbedre administrationen og sikkerheden af personoplysninger. Gennem brugen af MS SQL Server og C# er der blevet skabt en robust og sikker platform, der møder de krav og behov, som moderne uddannelsesmiljøer stiller. Systemet understøtter forskellige brugergrupper, inklusiv studerende, undervisere og administrativt personale, hvilket muliggør en effektiv styring og adgang til kritiske oplysninger.

Projektet har implementeret Model-View-Controller (MVC) arkitekturen for at sikre en modulær og vedligeholdelsesvenlig løsning, der både er skalerbar og let at tilpasse fremtidige behov. Brug af værktøjer som Figma og Azure Data Studio har understøttet design- og udviklingsprocesserne, mens SCRUM og SDLC metodologierne har guidet projektforløbet for at sikre en struktureret og agil tilgang til softwareudvikling.

Et centralt fokus i projektet har været på brugervenlighed og sikkerhed, hvor der er lagt vægt på intuitive brugergrænseflader og strenge sikkerhedsforanstaltninger for at minimere risikoen for brugerfejl og uautoriseret adgang. Systemet har været igennem en grundig validering og verificering, hvor alle funktionelle og ikke-funktionelle krav er blevet testet for at sikre, at de lever op til de opstillede standarder.

Til trods for udfordringer såsom tidsstyring og integration af ny teknologi, viser projektets resultater et lovende potentiale for videre udvikling og optimering. Læringskurven har været stejl, og erfaringerne fra projektet vil uden tvivl bidrage til fremtidige initiativer og forbedre gruppens evne til at håndtere komplekse IT-projekter.

Samlet set demonstrerer dette projekt en vellykket implementering af moderne teknologier og projektstyringsmetoder, der sammen skaber en dynamisk og tilpasselig løsning, som vil gavne både institutionen og dens brugere.

# Problemanalysen

Research:

1. Mindre research efter eksisterende personregistrerings[[1]](#footnote-1) systemer til uddannelsesinstitut og vi skulede til IBAs eget system[[2]](#footnote-2) 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[[3]](#footnote-3) og C# [[4]](#footnote-4)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[[5]](#footnote-5), 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[[6]](#footnote-6), 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 ansatte. Resten er afgrænset.
* Der er kun taget højde for et undervisningssted.
* En dynamisk kalender funktion er der afgrænset for.
* Administrator delen skrives Windows Pc og er afgrænset for Mac brugere
* 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

**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[[7]](#footnote-7): (Unified Modeling Language)**: Systemdiagrammerne fra UML er brugt til at visualisere det forskellige abstrakte niveauer, samt synliggøre flowet og interaktionen, helt oppe fra User Story ned til Class og Deploy diagrammer.

**Prototypeværktøjer**: Figma[[8]](#footnote-8) 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.

**MS SQL**: Nedenstående viser 3NF diagram fra Azure Data Studion, der tydeligt viser tabellerne.

A diagram of a company

Description automatically generated

## 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 (ligger i bilag).

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

Nedenstående Use Case er over ADM UI brugeren og viser tydeligt de ”niveauer” adm skal igennem, for at komme ned til selve handlingen. Det kan give anledning at tænke kreativt, for hvordan kommer vi ned til handlingerne hurtigt og uden fejl.

A diagram of a system

Description automatically generated

Nedenstående er Use Case over de 3 brugerscenarier, vi har påtænkt os at arbejde med fra BrugerGUI. Her ses det tydelig at der ikke er mange niveauer som i foregående. Men her skal der tænkes KISS (Keep It Simpel, Stupid)

A diagram of a person's personal data

Description automatically generated

**Arkitektur Design**

I designfasen af vores system har vi valgt at implementere Model-View-Controller[[9]](#footnote-9) (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, hvor der vises at begge grænseflader i MVC-arkitekturen ligger teoretisk i View delen, men de har hver sin indgang til systemet. Hvor Bruger GUI følger den ordinær struktur gennem webserveren i Controller laget , hvorimod ADM UI har direkte adgang til Model laget.

A diagram of a computer

Description automatically generated

Her er klassediagrammer over den applikation og grundet kompleksiteten er der flere diagrammer:

A screenshot of a computer

Description automatically generated

Ovenstående viser strukturen i component DBAdgang, hvor entres point vil være DatabaseService. På denne klasse ville det være hensigtsmæssigt at implemetere et Singleton Pattern[[10]](#footnote-10), for at sikre databasen ved at kun en af adgang af gangen. Som vil være med til at holde database konsistens og stabil.

A screenshot of a computer

Description automatically generated

Ovenstående er klasse diagrammet over ADM UI delen, hvordan ADM UI handler diverse CRUD funktioner på databasen. Det er tydeligt at de fleste funktioner er lagt på knapper, som gør det brugervenligt, selv på ADM UI delen.

A screen shot of a computer

Description automatically generated

Ovenstående klasse diagram over entreprisedelen, hvor alle bruger info forespørgsler fra BrugerGui og ADM UI går igennem. Hvor der vises at hver type request er repræsenteret ved deres egen klasse, for at skabe modularitet og vedligeholdes venlig kode.

**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#[[11]](#footnote-11) og .NET 7[[12]](#footnote-12):

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[[13]](#footnote-13): 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[[14]](#footnote-14). 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:

A screenshot of a computer

Description automatically generated

Ovenstående MockUp af ADM UI delen, viser netop at funktionerne på ADM delen er lagt i knapper, som er med til at minimere risikoen for fejl og gør delen meget bruger venlig.I højre side den run-time log, der viser aktiviteten i systemet og her fra ADMs side.

A screenshot of a computer

Description automatically generated

Ovenståender er MockUp af Bruger vindue og denne er en studerende. Der vises de informationer en student profil har adgang til. Nemt og overskueligt.

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 Design afsnit:**

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

Under udviklingsfasen af projektet blev der lagt stor vægt på evaluering og justering af designet baseret på de erfaringer, som gruppen opnåede gennem arbejdet med mock-ups. Disse tidlige modeller bidrog væsentligt til den iterative designproces, hvorved en funktionel prototype blev udviklet. Denne prototype tjener som en praktisk demonstration af applikationens kernefunktionaliteter, der er afgørende for et personregistersystem. Selvom prototypen endnu er i en initial fase og indeholder flere mangler, illustrerer den potentialet for det endelige produkt. Da produktionen af applikationen er iterativ, forventes det, at yderligere ændringer vil forekomme i løbet af den sidste del af projektet, hvilket vil forbedre funktionalitet og brugeroplevelse.

Værktøjer:

MS SQL og Azure Data Studio: Valget af disse teknologier blev dikteret af deres fulde kompatibilitet med andre Microsoft-produkter, hvilket er essentielt, da projektet anvender C# og .NET framework. Disse værktøjer har vist sig at være nemme at arbejde med og tilbyder omfattende sikkerhedsfunktioner samt skaleringsmuligheder, hvilket gør dem til et effektivt valg for vores projektbehov. Yderligere overvejelser om at erstatte HttpListener med mere robuste og vedligeholdte webservere såsom ASP.NET Core blev taget i betragtning; dog, på grund af begrænset erfaring med disse alternativer, blev de ikke implementeret i denne iteration.

Figma: Dette værktøj blev anvendt til at skabe og præsentere designkoncepter tidligt i udviklingsfasen, hvilket har været afgørende for at holde fokus og levere en klar vision for projektets retning. Selvom projektet ikke havde eksterne stakeholders, blev værktøjet betragtet som værdifuldt for intern præsentation og evaluering før udviklingen af den egentlige prototype.

Brugergrænseflade: Fra projektets begyndelse har der været en stærk fokusering på brugervenlighed samt sikkerhedsmæssige foranstaltninger for at minimere risikoen for brugerfejl. Derfor er det besluttet, at en betydelig del af funktionaliteterne skal være forudbestemte, hvorved brugerinteraktionen primært faciliteres gennem veldefinerede knappevalg. Dette designvalg bidrager til at forenkle brugergrænsefladen og øge dens intuitivitet. Selvom æstetikken og det moderne aspekt af grænsefladedesignet kunne have været mere fremtrædende, har prioriteringen været at sikre funktionalitet og brugersikkerhed frem for visuel appel. Dette valg afspejler en strategisk beslutning om at vægte operationel effektivitet og brugernes lette navigation højere end den visuelle præsentation.

Systemudvikling: Vedrørende udviklingsmetoden, er der truffet beslutning om at anvende en Software Development Life Cycle[[15]](#footnote-15) (SDLC) tilgang, der inkorporerer iterative elementer. Denne beslutning understøtter en fleksibel og responsiv udviklingscyklus, som tillader løbende evaluering og tilpasning baseret på projektets behov og stakeholder feedback. Projektledelsesstrukturen er udformet som en hybridmodel, der integrerer principperne fra Scrum[[16]](#footnote-16), hvilket tilskynder til agilitet og samarbejde indenfor teamet. Denne tilgang muliggør effektiv håndtering af komplekse projektopgaver og fremmer en adaptiv projektstyring.

Validering og verificering:

I validering og verificering blev fokus lagt på at sikre, at systemet nogenlunde opfyldte alle krav og fungerede som forventet gennem en omfattende test- og dokumentationsproces. Validering og verificering var afgørende for at sikre, at systemet var af høj kvalitet og opfyldte brugerens behov.

For at validere systemet blev der testet baseret på kravspecifikationen. Disse test blev designet til at afprøve systemets funktionaliteter og sikre, at det opfyldte både de funktionelle og ikke-funktionelle krav, der var blevet identificeret. Enhedstests og integrationstests blev udført for at sikre, at alle dele af systemet fungerede korrekt sammen.

Verificeringen omfattede en evaluering af systemets kodekvalitet og struktur for at sikre, at det var robust og let at vedligeholde. Kommunikationen i webserveren blev testet ved hjælp af Rest API[[17]](#footnote-17) og JSON-beskeder for at sikre en pålidelig og effektiv dataudveksling.

# Del konklusion

Projektets udviklingsfase har tydeligt demonstreret vigtigheden af en iterativ designproces, der begyndte med en grundlæggende analyse til de grundige mock-ups og fortsatte til udviklingen af en funktionalitetsrig prototype. Denne prototype, skønt stadig i sin begyndende fase, har fungeret som en praktisk demonstrator for de essentielle funktioner der er nødvendige i et personregistersystem. Trods dens nuværende mangler, har den illustreret potentialet for det endelige produkt og dets evne til at forbedre både funktionalitet og brugeroplevelse gennem fortsatte iterationer.

Valget af MS SQL og Azure Data Studio var dikteret af deres integrationskompatibilitet med Microsofts teknologi stack, hvilket har været afgørende i projektets anvendelse af C# og .NET framework. Disse værktøjer har vist sig at være nemme at arbejde med, tilbyder væsentlige sikkerhedsfunktioner og skalerbarhed, hvilket gør dem til en fordelagtig løsning for projektets behov. Overvejelserne om at skifte fra HttpListener til mere robuste og vedligeholdte løsninger såsom ASP.NET Core understreger et ønske om teknologisk opgradering, selvom erfaringen med disse alternativer endnu ikke har muliggjort en implementering.

Figma blev effektivt brugt til at formulere og præsentere tidlige designkoncepter, hvilket styrkede projektets fokus og retning. Værktøjets evne til at facilitere præsentation og evaluering internt i teamet har været uvurderlig, selv uden eksterne stakeholders.

I brugergrænsefladens design blev der lagt stor vægt på brugervenlighed og sikkerhedsforanstaltninger for at reducere risikoen for brugerfejl, hvilket har ført til beslutningen om at prædefinere mange af systemets funktioner. Dette har gjort brugerinteraktionen mere intuitiv og sikker, selvom det har medført en vis nedprioritering af æstetiske og moderne designaspekter.

Systemudviklingen blev drevet af en SDLC-tilgang, som integrerede iterative elementer og Scrum-principper i en hybrid projektledelsesstruktur. Denne strategi har været essentiel for at sikre en fleksibel og responsiv udviklingscyklus, der har tilpasset sig projektets skiftende behov og stakeholder feedback.

Validerings- og verificeringsprocesserne har været centrale i projektet for at sikre, at systemet opfyldte de opstillede krav og fungerede som forventet. Omfattende testning og dokumentationsprocesser blev anvendt til at validere systemets funktionaliteter og sikre at både funktionelle og ikke-funktionelle krav blev mødt. Verificeringen omfattede en grundig evaluering af kodekvaliteten og systemstrukturen, samt en sikring af robust og effektiv kommunikation via Rest API og JSON-beskeder.

Sammenfattende har dette projekt ikke blot fremmet teknologisk innovation og funktionalitet, men også understreget vigtigheden af brugercentreret design og grundig systemtestning, hvilket sikrer et robust og brugervenligt slutprodukt.

# Diskussion

## Refleksion over projektets forløb og resultater:

Refleksionen over dette projekt afspejler både forventede resultater og nye udfordringer, idet projektet har indbefattet en række nye teknologier og viden, som har adskilt det fra tidligere projekter. Dette projekt har været kendetegnet ved en stejl læringskurve, som direkte er påvirket af semesterets struktur og den valgte planlægning. Disse faktorer har medført en række aha-oplevelser, især når projektets komponenter lykkedes og begyndte at fungere som forventet.

I projektets sidste fase blev vi præsenteret for Microsofts MVC framework i .NET ASP Core, som viste sig at være en uvurderlig ressource, der kunne have løftet vores projektresultater betydeligt, hvis det var blevet integreret tidligere i processen. Dette understreger vigtigheden af kontinuerlig læring og tilpasning inden for teknologiske projekter.

Resultatet af projektet viser tydeligt, at tid er en kritisk faktor, som ligger uden for vores kontrol. Selvom prototypen demonstrerer grundlæggende funktionalitet, mangler den mange af de detaljerede funktioner og elementer, der blev beskrevet under udviklingsprocessen. Ideelt set ville det have været fordelagtigt at inkludere flere af disse funktioner i den nuværende prototype.

Hele projektforløbet har også været præget af en stor diversitet blandt gruppedeltagerne, hvilket har bidraget til en varieret læringskurve. Denne mangfoldighed har både været en udfordring, idet den har belastet projektet med forskellige perspektiver, moraler og samtidig krævet nye læringspunkter.

Samlet set har denne projektoplevelse ikke kun styrket vores tekniske kompetencer, men også vores evne til at navigere i komplekse, skiftende omgivelser, og den har understreget betydningen af fleksibilitet og proaktiv tilpasning i softwareudvikling.

Problemformulering:   
Problemformuleringen var noget vag i projektbeskrivelsen, men efter grundig research og inspiration fra IBAs eget registersystem fik vi formuleret en brugbar problemformulering. Efter at have defineret ”brugeradgangen” - det vil sige, hvem der skal have adgang, og hvad de forskellige profiler skal have rettigheder til, blev det tydeligt, at en mere præcis problemformulering var nødvendig. Ved at analysere IBAs eget registersystem og udføre yderligere research lykkedes det at formulere en brugbar problemformulering, der kunne guide udviklingen af det ønskede personregistreringssystem.

Denne præciserede problemformulering gav os et klart billede af, hvad der skulle opnås med systemet og hvilke krav det skulle opfylde. Det hjalp med at identificere de nødvendige funktionaliteter og sikre, at systemet blev udviklet med fokus på de specifikke behov hos både studerende, undervisere og øvrigt personale.

Kvalitet:

Kvaliteten af prototypen er ikke fuldstændig udviklinget, men det, der allerede er skabt, viser sig at være robust og pålideligt. Dog er den visuelle præsentation af brugergrænsefladen (GUI) bemærkelsesværdigt lav, da der ikke er blevet lagt tilstrækkelig vægt på det æstetiske aspekt. Fokus har primært været rettet mod systemudvikling og integration af forskellige komponenter i applikationen, hvilket har ført til en mere funktionel, men visuelt underudviklet prototype.

Udover at fokusere på den tekniske kvalitet vil der også blive evalueret processen og projektledelsen. Ved at reflektere over projektets forløb og resultater kan der identificeres områder til forbedring og implementeres nødvendige tiltag for at sikre, at fremtidige projekter gennemføres med høj kvalitet og succes.

Metoder:

Metodologi

I projektet har vi anvendt Software Development Life Cycle[[18]](#footnote-18) (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[[19]](#footnote-19) 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

Del konklusion:

Projektets forløb og resultater har været en refleksion over både de forventede resultater og de nye udfordringer, der præsenterede sig gennem arbejdet med nye teknologier og metoder, hvilket distinkt har adskilt dette projekt fra tidligere bestræbelser. Den stejle læringskurve, ansporet af semesterets struktur og vores planlægningsmetode, har bidraget til værdifulde aha-oplevelser, især når projektets komponenter begyndte at funktionere succesfuldt.

Opdagelsen og den efterfølgende integration af Microsofts MVC framework i .NET ASP Core i projektets afsluttende fase fremhævede vigtigheden af kontinuerlig læring og tilpasning i teknologiske projekter, idet dette værktøj havde potentiale til at forbedre projektets resultater markant.

En kritisk observation fra projektet er, at tid er en begrænsende faktor, som vi ikke havde kontrol over. Selvom prototypen demonstrerede grundlæggende funktionalitet, manglede den betydeligt i forhold til de detaljerede funktioner og elementer, som var foreskrevet under udviklingen. Ideelt set ville det have været fordelagtigt at integrere flere af disse funktioner i den nuværende prototype for at realisere et mere fuldt udviklet produkt.

Projektforløbet blev også karakteriseret ved diversiteten blandt deltagerne, som bidrog med varierede perspektiver og læringsoplevelser. Denne mangfoldighed var både en styrke og en udfordring, da den berigede projektet med forskellige synspunkter, men også krævede effektiv koordinering og integration af nye ideer og metoder.

Generelt har dette projekt ikke kun styrket vores tekniske færdigheder, men også forbedret vores evne til at navigere i komplekse og skiftende miljøer. Det har også understreget vigtigheden af fleksibilitet og proaktiv tilpasning inden for softwareudvikling. Disse erfaringer har været afgørende for vores udvikling som it-professionelle og vil informere vores tilgang til fremtidige projekter, der kræver både omhyggelig planlægning og evnen til at tilpasse sig nye teknologier og metoder undervejs.

# Konklusion

Dette projekt har vist sig at være en dybtgående og oplysende erfaring, der afspejler både de forventede resultater og de nye udfordringer forbundet med indførelsen af avancerede teknologier og metoder. En nøje afgrænsning og dybdegående analyse har belyst vitale aspekter af systemdesign og funktionalitetskrav, essentielle for udviklingen af et effektivt personregistreringssystem for en uddannelsesinstitution.

Projektet har affødt en væsentlig læringskurve, drevet af en intensiv og strategisk planlægning samt introduktionen af ny teknologi såsom Microsofts MVC framework i .NET ASP Core, som desværre ikke blev implementeret. Denne tilføjelse undervejs i projektet demonstrerede tydeligt vigtigheden af kontinuerlig tilpasning og opdatering af vores tilgang til teknologi. Det fremhævede potentialet for markant forbedrede resultater, hvis sådanne værktøjer blev inkorporeret tidligere i udviklingsprocessen.

Tidsstyring viste sig at være en kritisk udfordring. Prototypen, skønt funktionalitets manglende detaljer, funktioner og elementer, hvilket belyste en diskrepans mellem den ideelle og den faktiske udførelse af projektet. Dette understreger behovet for yderligere integration af funktionaliteter for at opnå et mere fuldendt produkt.

Diversiteten i gruppen bidrog til en rig læringsproces, hvilket både var en styrke og en stor udfordring. Det førte til værdifulde perspektiver og indsigt, men krævede også omhyggelig koordinering og integration af nye ideer. Dette interaktionsspektrum har været afgørende for at navigere i de komplekse og skiftende omgivelser, som karakteriserer moderne softwareudvikling.

Vores metodologiske tilgang, som kombinerede SDLC's strukturerede rammer med SCRUM's agilitet, har vist sig yderst effektiv. Denne hybridmodel fremmede en fleksibel og målrettet udviklingsproces, der ikke kun opfyldte de funktionelle og ikke-funktionelle krav, men også sikrede, at slutproduktet var både brugervenligt og sikret.

Samlet set har projektet ikke blot fremmet teknologisk innovation og funktionalitet, men også understreget vigtigheden af brugercentreret design og grundig systemtestning. Disse elementer har været afgørende for at sikre et robust og brugervenligt slutprodukt. Erfaringerne og læringen fra dette projekt vil utvivlsomt påvirke vores fremtidige professionelle praksis og fremme en fortsat udvikling af vores tekniske og koordinative kompetencer i kommende projekter.

# Litteraturliste

“About the Unified Modeling Language Specification Version 2.5.1.” Accessed May 15, 2024. https://www.omg.org/spec/UML/.

“ASP.NET MVC Overview | Microsoft Learn.” Accessed May 16, 2024. https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview.

“Azure REST API Reference Documentation | Microsoft Learn.” Accessed May 15, 2024. https://learn.microsoft.com/en-us/rest/api/azure/.

“Database Normalization Description - Microsoft 365 Apps | Microsoft Learn.” Accessed May 15, 2024. https://learn.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description.

“Det Centrale Personregister (CPR).” Accessed May 15, 2024. https://www.borger.dk/samfund-og-rettigheder/Folkeregister-og-CPR/Det-Centrale-Personregister-CPR.

“Figma: The Collaborative Interface Design Tool.” Accessed May 16, 2024. https://www.figma.com/.

“Google Books Link.” Accessed May 15, 2024. https://books.google.dk/books?id=Lw8LEAAAQBAJ.

“Skema Web.” Accessed May 15, 2024. https://ums.basyd.dk/WebTimeTable/default.aspx.

“SQL Server-Downloads | Microsoft.” Accessed May 15, 2024. https://www.microsoft.com/da-dk/sql-server/sql-server-downloads.

“Table of Contents - C# Language Specification | Microsoft Learn.” Accessed May 15, 2024. https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/readme.

“What Is Scrum? | Scrum.Org.” Accessed May 16, 2024. https://www.scrum.org/resources/what-scrum-module.

“What Is SDLC? - Software Development Lifecycle Explained - AWS.” Accessed May 15, 2024. https://aws.amazon.com/what-is/sdlc/.

“What’s New in .NET 7 - .NET | Microsoft Learn.” Accessed May 16, 2024. https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-7.

“Windows Presentation Foundation for .NET 8 Documentation | Microsoft Learn.” Accessed May 16, 2024. https://learn.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-6.0.

# Bilag:

Arbejdsfordeling:

ProjektOpstart: Milepæl 0

1 Git oprettelse - Nickolaj 100% ok

2 Log + diver oprettelse - Nickolaj 100% ok

3 Projektplan - Allan + Fahad 50%/50% ok

4 Gantt - Fahad 100% ok

5 Rapport opstart - Allan 100% ok

6 Foreløbig Flowchart - Mohammad 100% ok

0.1 WBS - Alle 25%/25%/25%/25% ok

Analysen: Milepæl 1

7 Problemformulering - Allan+Alle 80%/20% ok

8 Problemstilling - Allan+Alle 80%/20% ok

9 User Stories - Fahad+Alle 80%/20% ok

10 Kravsanalyse - Fahad+Alle 80%/20% ok

Forløb: Milepæl 2

Projektforløb

11 Afgrænsning - Allan+Alle 80%/20% ok

12 Succeskriterier - Allan 100% ok

13 Kravspecifikation - Fahad+Alle 80%/20% ok

Design: Milepæl 3

14 Design værktøjer Allan 100% ok

14.1 OOP Allan 100% ok

14.2 Design Pattern Allan 100% ok

14.3 UML Allan 100% ok

15 Adfærd Allan 100% ok

15.1 UML - Use Case (adfærd) Mohammad 100% ok

15.3 UML - Sequence (adfærd) Nickolaj + Fahad + Allan 33%/33%/33% ok

15.4 UML - Klasse (structural) Nickolaj, Allan, Fahad 33%/33%/33% ok

16 Arkitektur: MVC Allan 100% ok

17 Koncept Allan 100% ok

18 Teknologi Allan + Fahad 70%/30% ok

19 Mock-up Nickolaj + Mohammad 50%/50% ok

Kode:

A. Controller - Webserver Fahad 100% ok

B. Model - Databasen/enterprise apl Allan 100% ok

C. ADM UI - Windows Nickolaj 100% ok

D. HTML/CSS JS - Mohammad 100% ok

Prototypen: Milepæl 4

20 Værktøjer Allan + Fahad + Nickolaj 60%/20%/20% ok

21 Brugergrænseflade Allan 100% ok

22 Systemudvikling Allan 100% ok

23 Validering og verificering

Disk Milepæl: 5

24 Refleksion Allan 100% ok

25 Problemformulering Allan + Fahad 20%/80% ok

26 Kvalitet Allan + Fahad 50%/50% ok

27 Metoder Allan 100% ok

28 Perspektivering Allan 100% ok

Delkonklusioner Allan + Fahad 75% / 25% ok

Konklusion Allan 100% ok

Resumé Allan 100% ok

Korrektur Fahad + Allan 80% / 20% ok

Nedenstående overordnede diagram over MVC strukturen inden der blev besluttet af ADM UI tilhøre direkte View laget, men diagrammet viser fint, hvordan strukturen er.

A diagram of a data flow

Description automatically generated

Nedenstående er klasse diagram over Component Databaseadgangen på modellaget. Viser en svag adskillelse af dataadgang og forretningslogik. Et Repository pattern var planlagt, men blev ikke implementeret.

A diagram of a computer

Description automatically generated

Nedenstående er sequence diagram over ADM UI delen (del af View laget), viser tydeligt flowet ændring af data og hvordan de forskellige opdateringer gennemføres

A screenshot of a computer

Description automatically generated

Nedenstående diagram over componenten DatabaseAdgang, hvor et interface er koblet på. Dette vil være med til at hæve encapsulation og styrke design pattern Repository eller Factory.

A diagram of a server

Description automatically generated

Nedenstående Flowchart over ADM UI delen

A diagram of a software development

Description automatically generated

Nedenstående et tidligere sequence diagram over data request (View, storedProcedure, storedFunction) i Model laget.

A screenshot of a computer screen

Description automatically generated

1. “Det Centrale Personregister (CPR).” [↑](#footnote-ref-1)
2. “Skema Web.” [↑](#footnote-ref-2)
3. “SQL Server-Downloads | Microsoft.” [↑](#footnote-ref-3)
4. “Table of Contents - C# Language Specification | Microsoft Learn.” [↑](#footnote-ref-4)
5. “Azure REST API Reference Documentation | Microsoft Learn.” [↑](#footnote-ref-5)
6. “Database Normalization Description - Microsoft 365 Apps | Microsoft Learn.” [↑](#footnote-ref-6)
7. “About the Unified Modeling Language Specification Version 2.5.1.” [↑](#footnote-ref-7)
8. “Figma: The Collaborative Interface Design Tool.” [↑](#footnote-ref-8)
9. “ASP.NET MVC Overview | Microsoft Learn.” [↑](#footnote-ref-9)
10. “Google Books Link.” [↑](#footnote-ref-10)
11. “Table of Contents - C# Language Specification | Microsoft Learn.” [↑](#footnote-ref-11)
12. “What’s New in .NET 7 - .NET | Microsoft Learn.”“Windows Presentation Foundation for .NET 8 Documentation | Microsoft Learn.” [↑](#footnote-ref-12)
13. “Windows Presentation Foundation for .NET 8 Documentation | Microsoft Learn.” [↑](#footnote-ref-13)
14. “Figma: The Collaborative Interface Design Tool.” [↑](#footnote-ref-14)
15. “What Is SDLC? - Software Development Lifecycle Explained - AWS.” [↑](#footnote-ref-15)
16. “What Is Scrum? | Scrum.Org.” [↑](#footnote-ref-16)
17. “Azure REST API Reference Documentation | Microsoft Learn.” [↑](#footnote-ref-17)
18. “What Is SDLC? - Software Development Lifecycle Explained - AWS.” [↑](#footnote-ref-18)
19. “What Is Scrum? | Scrum.Org.” [↑](#footnote-ref-19)