Projektrapport

IBA

Dkpbi23a2a

Eksamensprojekt – ”personregistrering”

Gruppe: Mohammad Fahad, Nickolaj, Allan

Forside:

Login: JECO og SKQU

Indholdsfortegnelse

[Resumé: 2](#_Toc165487999)

[Problemanalysen 2](#_Toc165488000)

[Research 2](#_Toc165488001)

[1. Mindre research efter eksisterende 2](#_Toc165488002)

[Problemformulering 3](#_Toc165488003)

[Problemstilling 3](#_Toc165488004)

[Målgruppeanalyse 4](#_Toc165488005)

[User Story 4](#_Toc165488006)

[Kravanalyse 6](#_Toc165488007)

[Projektforløb 8](#_Toc165488008)

[Afgrænsning 8](#_Toc165488009)

[Succeskriterier 8](#_Toc165488010)

[Kravspecifikation 9](#_Toc165488011)

[Design 10](#_Toc165488012)

[ Design værktøjer 10](#_Toc165488013)

[ Adfærds analyse 11](#_Toc165488014)

[ Arkitektur design 11](#_Toc165488015)

[ Koncept evaluering 11](#_Toc165488016)

[ Teknologi beskrivelse 11](#_Toc165488017)

[ Mock-up 11](#_Toc165488018)

[Prototype 12](#_Toc165488019)

[ Værktøjer 12](#_Toc165488020)

[ Brugergrænseflade 12](#_Toc165488021)

[ Systemudvikling 12](#_Toc165488022)

[ Validering og verificering 12](#_Toc165488023)

[Diskussion 12](#_Toc165488024)

[ Problemformulering 12](#_Toc165488025)

[ Kvalitet 12](#_Toc165488026)

[ Metoder 12](#_Toc165488027)

[ Perspektivering 12](#_Toc165488028)

[Konklusion 12](#_Toc165488029)

[Litteraturliste 12](#_Toc165488030)

[Bilag: 12](#_Toc165488031)

# 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. ”En simpel 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 reseach 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

Udviklingenen 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. XX
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.

# 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 studieplan aktiviteter.

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.

Kravanalyse:

**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.

# Projektforløb

Afgrænsning:

* Dynamiske kalender funktion vedrørende studerende og ansatte.
* ADM login på mac.
* Login sikkerhed – Salt og Peber

Succeskriterier:

* Opgave (backlog):
  + Gruppen godkender opgaveløsningen, når opgavekrav er fuldført efter pensum og aftale.
* Milepæl (backlog):
  + Når alle opgaver i milepælen er godkendt af gruppen.
* Rapport:
  + Når der er en fyldestgørende rapport jf til kravene i projektbeskrivelsen.
  + En klar sammenhæng mellem problem, løsning og applikation.
  + Rapporten opfylder de beskrevne krav.
  + alle i gruppen er tilfredse, så kan den godkendes.
* Applikationen:
  + Kravspecifikationen er opfyldt og godkendt af gruppen.
  + 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.

# Design

Planlægning og skitsering af systemets arkitektur og brugerinteraktion:

Design værktøjer:

1. En holistisk tilgang integrerer forskellige værktøjer og metoder på tværs af systemudviklingsprocessen, hvilket resulterer i en mere sammenhængende og effektiv tilgang til design og implementering af softwareløsninger. Ved at kombinere OOP, UML og designmønstre opnår vi flere fordele:
2. **Helhedsorienteret design**: Ved at anvende OOP som fundament for systemarkitekturen og UML til at visualisere og kommunikere designet opnår vi en helhedsorienteret tilgang, der tager højde for både strukturelle og adfærdsmæssige aspekter af systemet. Dette bidrager til en mere sammenhængende og gennemsigtig forståelse af systemet blandt udviklere, designere og interessenter.
3. **Klar kommunikation og dokumentation**: UML-diagrammer fungerer som en universel visuel sprog, der muliggør klar kommunikation og dokumentation af systemarkitekturen på tværs af forskellige interessenter. Ved at bruge UML til at beskrive systemets struktur, adfærd og interaktioner kan vi sikre en fælles forståelse af systemet på tværs af udviklingsteamet og andre interessenter.
4. **Genbrugelighed og vedligeholdelse**: Designmønstre fremmer genbrugelighed af kodestykker og etablerer best practices for systemdesign. Ved at anvende velkendte mønstre som Singleton, Repository og Adapter kan vi reducere kodens kompleksitet, øge dens genbrugelighed og gøre systemet mere modulært. Dette gør det lettere at vedligeholde og udvide systemet over tid.
5. **Fleksibilitet og skalerbarhed**: En holistisk tilgang til systemdesign giver os mulighed for at udvikle fleksible og skalerbare systemer, der kan tilpasses ændrede krav og vokse med virksomhedens behov. Ved at kombinere OOP's modulære designprincipper med UML's abstraktionsmuligheder og designmønstres løs kobling kan vi skabe systemer, der er robuste, fleksible og lette at tilpasse.
6. Samlet set muliggør en holistisk tilgang til systemdesign, der kombinerer OOP, UML og designmønstre, udviklingen af softwareløsninger, der er mere robuste, skalerbare og vedligeholdbare, samtidig med at de opfylder virksomhedens behov og mål. Denne tilgang fremmer også en dybere forståelse af systemets kompleksitet og en mere effektiv samarbejde på tværs af udviklingsteamet og andre interessenter.

Adfærds analyse:

Adfærdsanalyse er en vigtig del af systemdesignprocessen, hvor vi fokuserer på at forstå og definere systemets ønskede adfærd i forskellige situationer. I vores tilgang benytter vi os primært af adfærdsdiagrammer fra Unified Modeling Language (UML) for at visualisere og analysere systemets adfærd på en struktureret og forståelig måde.

User Story-diagrammer bruges til at beskrive systemets funktionalitet set fra brugerens perspektiv. Disse diagrammer fokuserer på brugerens mål og interaktioner med systemet og hjælper med at definere, hvordan systemet skal opføre sig i forskellige scenarier.

Use Case-diagrammer identificerer og beskriver de forskellige brugsscenarier eller situationer, hvor systemet skal interagere med brugeren eller andre systemer. Ved at modellere disse brugsscenarier får vi en klar forståelse af systemets krav og funktionalitet.

Sequence-diagrammer viser, hvordan forskellige objekter i systemet kommunikerer med hinanden i en bestemt rækkefølge for at opfylde en given opgave eller brugssag. Disse diagrammer hjælper med at visualisere systemets dynamiske adfærd og sekvensen af interaktioner mellem objekterne.

Activity-diagrammer eller State Machine-diagrammer anvendes til at beskrive systemets arbejdsgange, processer eller tilstande samt de overgange mellem dem. Disse diagrammer giver et overblik over systemets adfærd i form af trin-for-trin processer eller tilstandsmaskiner.

Ved at anvende disse adfærdsdiagrammer fra UML kan vi identificere, analysere og specificere systemets ønskede adfærd på en systematisk måde. Dette gør det muligt for os at designe og implementere softwareløsninger, der opfylder brugernes behov og forventninger på en effektiv og struktureret måde.

Arkitektur design:

Efter en nøje overvejelse og analyse af kravene til vores applikation, som omfatter en webserver med HTML-brugergrænseflade, en administrations-GUI og et MS SQL-datalag, har vi fastslået, at anvendelsen af MVC-arkitekturen er både passende og velbegrundet for vores projekt. MVC-arkitekturen er valgt for dens fremragende evne til at adskille applikationens komponenter—model, visning og controller—hvilket faciliterer en klar opdeling af ansvarsområder. Denne opdeling er essentiel for effektiv udvikling og vedligeholdelse i vores komplekse systemstruktur, der kræver nøje kontrolleret adgang til data.

Selvom Model-View-ViewModel (MVVM) også blev overvejet på grund af fordelene ved databinding og løs kobling, viser vores analyse, at MVC tilbyder større direkte kontrol med dataadgangen, en kritisk faktor i forhold til vores arbejde med MS SQL. MVC understøtter dermed en mere struktureret og objektorienteret tilgang til programmering, som er afgørende for vores projekts succes. Denne arkitektur tilbyder en optimal balance mellem struktur, fleksibilitet og vedligeholdelsesvenlighed, hvilket gør det til det mest hensigtsmæssige valg for at imødekomme vores specifikke behov og krav til applikationens udvikling og struktur.

Koncept evaluering: Vurdering af det initielle designforslag - Det involverer at sammenholde konceptet med projektets mål og kravspecifikationer for at identificere styrker, svagheder, og potentielle forbedringer. Dette trin inkluderer:

* + Gennemgang af systemets layout og navigationsflow.
  + Evaluering af brugergrænsefladens intuitivitet og tilgængelighed.
  + Vurdering af datamodellens skalérbarhed og fremtidssikring.
  + Sikkerhedsevaluering, herunder beskyttelse af følsomme data.
  + Teknisk gennemgang af systemarkitekturen og dens evne til at understøtte vedligeholdelse og udvidelse.
  + Feedback indsamling fra potentielle brugere og tekniske eksperter.

Teknologi beskrivelse: Beskrivelse af anvendte teknologier og frameworks.

Mock-up: Visuel prototype af brugergrænsefladen.

# Prototype

Evaluering og justering af design baseret på mock-up erfaringer:

* Værktøjer: Revision af anvendte design- og udviklingsværktøjer.
* 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: Evaluering af Problemformuleringen.
* Kvalitet: Evaluering af systemets funktionalitet og brugervenlighed.
* Metoder: Vurdering af anvendte forsknings- og udviklingsmetoder.
* Perspektivering: Fremtidige forbedringer og potentielle nye features.

# Konklusion

(Opsummering af projektets resultater og læring.)

# Litteraturliste

# Bilag:

Arbejdsfordeling