The Movies

projektrapport

AF

* Alex Holmbo
* Anna Frøslev Quaade
* Anne Kristina Rasmussen Lindby
* Johnny Torben Bjerring
* Magnus Buch

Gruppe 1



The Movies – Projektrapport

# Resumé

I dette projekt har vi arbejdet med casen 'The Movies'. Målet var at lave en prototype på et simpelt biografsystem. Vi har vist, hvordan man kan gå fra idé og modeller til en rigtig løsning i WPF med MVVM. Systemet kan oprette biografer, sale og filmforestillinger og håndtere bookinger. En vigtig regel er, at der ikke må sælges flere billetter end der er pladser i salen. Projektet har givet os øvelse i at bruge de modeller, vi har lært, og se hvordan de hænger sammen med koden.

# Indledning

Casen 'The Movies' er grundlaget for projektet. Opgaven var at bygge et lille system, der viser, at vi kan bruge analyse- og designværktøjer i praksis. Vi har valgt at fokusere på det vigtigste: at kunne lave biografer, sale og filmforestillinger, og at man kan booke billetter. For at holde det enkelt bruger vi CSV-filer i stedet for en database.

# Problemformulering

Hvordan kan vi lave et simpelt biografsystem, hvor man kan oprette biografer, sale, film og forestillinger, og sikre at der ikke sælges flere billetter end der er pladser i salen?

# High Level Design (HLD)

High Level Design er de første modeller, vi laver, når vi skal forstå et system.  
Formålet er at give et overblik over, **hvad systemet skal kunne**, uden at vi går i detaljer om kode og teknik.

HLD er vigtigt, fordi:

* Det fungerer som et fælles sprog mellem udviklere og kunden/brugerne.
* Det beskriver systemet på en måde, som alle kan forstå, også dem uden teknisk baggrund.
* Det sikrer, at vi har en fælles forståelse af kravene, inden vi begynder at lave detaljeret design og kode.

I HLD bruger vi artefakter som use cases, domænemodeller, objektmodeller, SSD’er og wireframes.  
Disse modeller viser, hvilke funktioner systemet skal have, hvilke begreber der findes (fx biograf, sal, booking), og hvordan brugeren skal kunne interagere med systemet.

.

## Use Case (reservation)

En vigtig del af systemet er, at kunder kan reservere billetter til kommende film. Usecasen beskriver, hvordan en ansat i biografen håndterer en reservation, og hvilke betingelser der skal være opfyldt, for at processen kan gennemføres.

Formålet med usecasen er at sikre, at både kunden og systemet får registreret de nødvendige oplysninger, så bookingen kan gennemføres korrekt. Oplysningerne er blandt andet kundens e-mail og telefonnummer, hvilken film og forestilling der ønskes, samt hvor mange billetter kunden vil købe.

Usecasen beskriver også de forudsætninger (pre-conditions), som skal være til stede – eksempelvis at filmen eksisterer i programmet, at der er ledige pladser, og at kunden kan kontaktes. Til sidst viser usecasen, hvilke resultater (post-conditions) man kan forvente, herunder at kunden får en bekræftelse på sin booking samt modtager en påmindelse om filmen.

## Et billede, der indeholder tekst, skærmbillede, dokument, brev AI-genereret indhold kan være ukorrekt.

## Objektmodel

Vi har lavet en objektmodel for at vise, hvordan de konkrete data kan se ud i vores system. Modellen er et slags øjebliksbillede af virkeligheden, hvor vi har sat eksempler ind på biografer, sale, film, forestillinger, bookinger og genrer.

Formålet er at gøre det lettere at forstå, hvordan de forskellige dele hænger sammen. Man kan se, at en biograf har flere sale, en sal kan bruges til forskellige forestillinger, og at kunderne laver bookinger på bestemte tidspunkter. Hver booking hænger sammen med både en kunde, en film og en sal.

Objektmodellen hjælper os med at teste, om vores domænemodel fungerer i praksis, fordi vi kan se, om sammenhængene giver mening med rigtige eksempler.

## Et billede, der indeholder tekst, skærmbillede, diagram AI-genereret indhold kan være ukorrekt.

## Domænemodel

Domænemodellen viser de vigtigste begreber i vores system og hvordan de hænger sammen. Den er mere generel end objektmodellen, fordi den ikke viser konkrete eksempler, men de klasser og attributter vi arbejder med i systemet.

Formålet med domænemodellen er at skabe et fælles sprog mellem os udviklere og brugerne, så vi er enige om, hvad systemet skal indeholde. Man kan se, at en biograf (Theatre) har flere sale (Cinema hall), og at hver sal kan have flere bookinger. En booking hænger sammen med både tidspunkt, film og antallet af billetter. Film har en genre, som hjælper med at beskrive filmens type.

Domænemodellen hjælper os med at holde overblik over strukturen i systemet, og den danner grundlag for de mere detaljerede modeller og den kode, vi har lavet bagefter.

## Et billede, der indeholder tekst, skærmbillede, diagram, Rektangel AI-genereret indhold kan være ukorrekt.

## System Sekvens Diagram (SSD)

SSD’et viser, hvordan en ansat i biografen taler sammen med systemet, når der laves en booking. Diagrammet beskriver trin for trin, hvilke beskeder der sendes mellem brugeren og systemet, og hvilken rækkefølge de kommer i.

Formålet med SSD’et er at få en klar og enkel oversigt over interaktionen, uden at vi går ned i detaljerne om klasser og kode. På den måde kan man hurtigt se, hvordan en booking starter, hvilke valg den ansatte skal tage (biograf, sal, forestilling, dato og tid), og hvordan systemet til sidst bekræfter bookingen og sender detaljer til kunden.

SSD’et hjælper os til at sikre, at vi har tænkt hele forløbet igennem, og at systemet understøtter de skridt, der er beskrevet i vores usecase.

## Et billede, der indeholder tekst, kvittering, skærmbillede, Font/skrifttype AI-genereret indhold kan være ukorrekt.

## Wireframes

Wireframes viser, hvordan brugerfladen i vores system kan se ud. De er simple skitser, som hjælper os med at få et overblik over de vigtigste funktioner og hvordan brugeren bevæger sig rundt i programmet.

Formålet med wireframes er ikke at vise et færdigt design, men at give et klart billede af, hvilke skærmbilleder vi skal have, og hvilke felter og knapper de indeholder. I vores projekt kan man se, at der er en startside med menupunkter, en booking-side hvor man kan indtaste oplysninger som biograf, film, dato, tid, antal billetter og kundens kontaktinfo, og til sidst en bekræftelse, når bookingen er gennemført.

Wireframes hjælper os med at sikre, at vi har alle nødvendige funktioner med, inden vi bygger selve brugerfladen i WPF.

Et billede, der indeholder skærmbillede, tekst

AI-genereret indhold kan være ukorrekt.

# Low Level Design (LLD)

Low Level Design er de mere tekniske modeller, som viser, hvordan systemet helt konkret skal bygges. Hvor High Level Design giver et overblik, går LLD et skridt dybere og beskriver systemets indre struktur og detaljer.

LLD er vigtigt, fordi:

* Det oversætter krav og HLD-modeller til teknisk design, som kan bruges direkte til at skrive kode.
* Det viser, hvilke klasser vi skal have, hvilke metoder de indeholder, og hvordan de hænger sammen.
* Det fungerer som en bro mellem modeller og kode, så vi sikrer, at vores implementering matcher vores design.

I vores projekt består LLD af designklassediagrammer (DCD), som viser klasser, attributter og relationer.

## Designklassediagram (DCD)

DCD’et viser, hvordan vores system er bygget op i klasser, og hvilke attributter og relationer de har. Hvor domænemodellen giver et overblik over begreberne, går DCD’et mere i dybden og viser, hvordan det hele kan kodes.

Formålet med DCD’et er at skabe en teknisk plan, som vi kan følge direkte i implementeringen. Man kan se, hvilke data der skal gemmes i de forskellige klasser (fx Seats i CinemaHall og NumberOfTickets i Booking), og hvordan klasserne hænger sammen (fx at et MovieShow kan have flere Bookings).

DCD’et hjælper os med at sikre, at vores kode bliver struktureret på en måde, der matcher både kravene fra use casene og reglerne fra domænemodellen. Det fungerer dermed som bindeled mellem design og kode.

# Et billede, der indeholder tekst, skærmbillede, diagram, Rektangel AI-genereret indhold kan være ukorrekt.

# Sporbarhed

Vi kan følge vores arbejde fra modeller til kode. Et eksempel er Seats i CinemaHall. Det findes i domænemodellen, i DCD og som en Property i klassen i koden. Det samme gælder NumberOfTickets i Booking. På den måde kan vi se en rød tråd fra krav til implementering.

# Implementering

Systemet er lavet som en WPF-app med MVVM. Models: Cinema, CinemaHall, Movie, MovieShow, Booking. Repositories gemmer data i CSV. ViewModels styrer logik og binder til Views. Views er XAML, som viser brugerfladen. Hovedfunktionen er booking, hvor systemet tjekker at der ikke sælges for mange billetter.

# Diskussion og Konklusion

Vi har lært at gå fra modeller til kode i et rigtigt projekt. Projektet viser, hvordan artefakterne hænger sammen med koden. Prototypen opfylder de vigtigste krav og kan udvides senere. Vi har set, at MVVM giver en god struktur, og at vi kan bruge de samme begreber gennem hele processen.