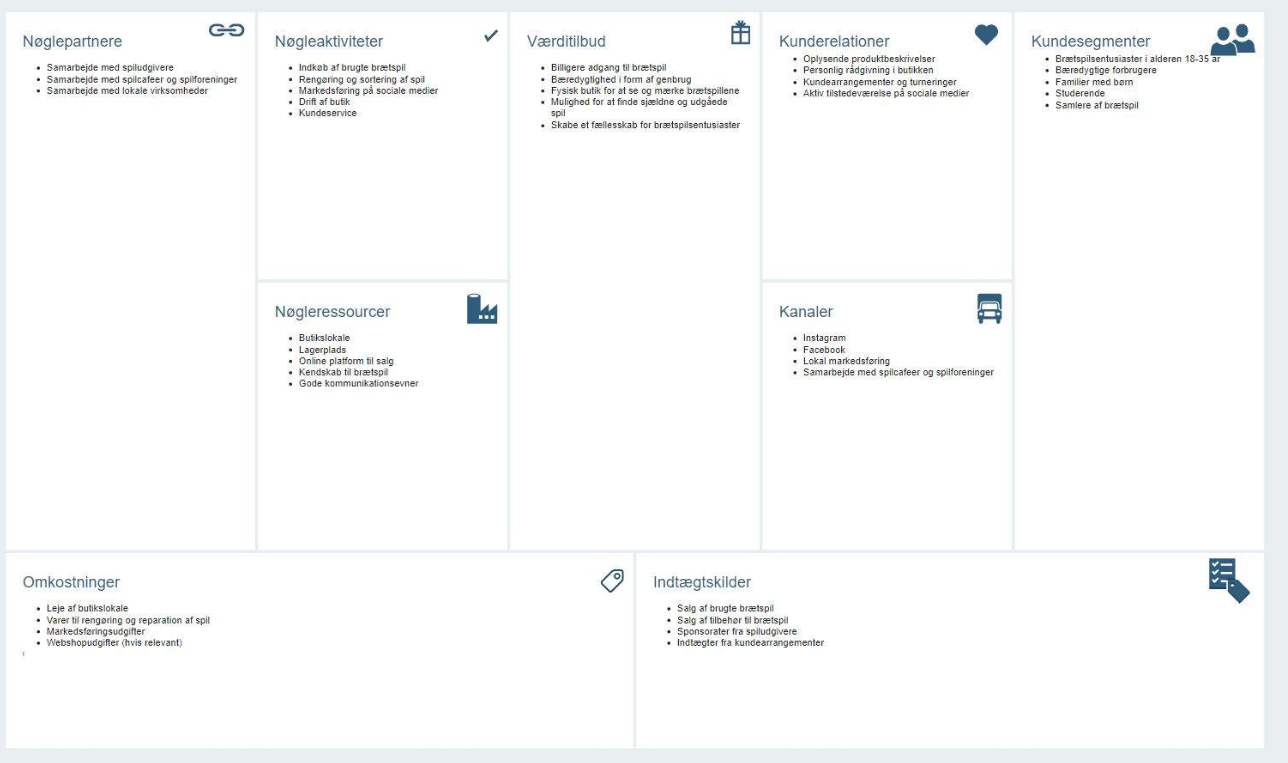
# Opstart og første plan

* Projektstyring
  + Scrum
    - Statusmøder 1 gang om ugen
    - Uddellegering
    - Værktøj: Trello
      * [Trello board Genspil](https://trello.com/invite/b/VMGVBhtC/ATTIb7d7b4836d68adaf5e221fd742ad975fEEC08B3A/genspil)
* Kravindsamling
  + Se nederst i dette dokument:
    - ESSENSIEL INFO FRA CASEN
    - Projektrammer for Case Genspil
  + FURPS+
    - Functionality
    - Usability
    - Reliability
    - Performance
    - Supportability
* High level design
  + Usecase og scenarier
  + Object-model
  + Domænemodel
  + Systemsekvensdiagram (SSD)
  + Operationskontrakter
* Low level design
  + Design Class Diagram (DCD)
* Programmering
* Test

# BMC



# Use cases.

1. *Søgning efter et spil og om den er på lager.*

Brugeren åbner systemet og vælger de kriterier som der efterspørges, han får en liste tilbage over spil på lager der har indeholder de kriterier.

Scenarie:

Jamal skal finde spil som har 2-8 spillere og er holdspil, han vælger kriterierne og får en liste tilbage med Cluedo, Plingo, Codenames.

1. *Indtastning af efterspørgsel fordi det efterspurgte spil ikke er i systemet.*

Brugeren søger efter navnet på spillet, systemet vender tilbage med at det ikke er der. Brugeren finder kriterier for spillet på nettet og indtaster i systemet.

Scenarie:

Emilie skal tjekke om de har Twister på lager, Listen kan ikke genereres og spørger kunden om der skal laves en forespørgsel på det, det vil kunden gerne og Emilie tilføjer spillet og reservation.

1. *Udskrivning af lagerliste.*

Brugeren klikker på udskriv lagerliste, systemet trækker ud de spil som er sat til at være på lager samt deres tilstand.

Scenarie:

Jonas skal lave status og har derfor brug for en liste over hvad der burde være på lager, han vælger kriterier for at lager er over 0 og får en liste ud.

1. *Kunder efterspørger spil baseret på alder*

Brugeren åbner systemet og vælger alders-kriterier som der efterspørges, han får en liste tilbage over spil på lager der har indeholder de kriterier.

# Pseudokode

**Søgning:**

Der er 3 typer søgning:

1. Søgning kriterier/filtrering
   1. Afkrydsning/valg
2. Søgning på fritekst
   1. Efter navn:
      1. “Ticke” giver resultater på fx “Ticket to ride”
   2. MÅSKE: Kan eventuelt også give resultat fra kriterier:
      1. “Strate” giver resultater for “Strategi spil”
      2. “hold” giver resultater for “holdspil”
3. En blanding af A og B, hvor der vælges kriterier og søges på fritekst

*“Vi vil gerne kunne søge på spil ud fra forskellige kriterier. Det kan for eksempel være genre, antal spillere, stand, pris, navn (eller del af navn) og hvis det kan lade sig gøre at kombinere kriterier, vil det være super.”*

**Overbliks funktion:**

Sortering: Primary og secondary (Fx: Genre først, derefter navn)

*“Vi vil gerne kunne udskrive en lagerliste til lageroptælling. Den skal både kunne sorteres efter navn og efter genre.”*

Farver? Kan sættes i 3 modus:

1. Reservationer
   1. Lilla
2. Forespørgsler
   1. Blå
3. Lager status
   1. Grøn - Over 3
   2. Gul - mellem 1-3
   3. Rød - ingen tilbage

*"Vi ikke har overblik over, hvilke brætspil vi har på lager, ligesom vi heller ikke har styr på, hvis vi har nogle forespørgsler eller reservationer liggende. "*

**Lægge nye spil ind i systemet**  
Instanciere objekter ud fra Spil-klasse og måske også specifikke spil ud fra classes. Fx: “Ticket to ride” så de er formateret og ensartede

Lagerliste

Når vi instantiere en aktuel lagerliste, så skal datoen for instantieringen skrives på i navnet. Den skal loop hvor der bliver tilføjet alle de brætspil som er på lager og deres stand.

Kig på at lave reservation, Løst

Læse kladelisten i stedet for inde i debug. (eller beskrive at listen skal lægges ind under debug mappen.) Løst

Tjek at søgefunktion virker med andet end kun navn. Løst

Få Visio og word fil fra Team 15 filer med i git uploaden. Løst

Klassediagrammer

|  |  |  |
| --- | --- | --- |
| **Game** | **SortAndSearch** | **PrintLagerliste** |
| **+ Name : String**  **+ Edition : String**  **+ genre : String**  **+ numberOfPlayers min : int**  **+ numberOfPlayers max : int**  **+ price : double**  **+ Condition : int**  **+ Quantity : int (version?)**  **+ waitList : array (person,date)**  **+ reserved : array (person,date)**  **+ BeingRepaired : bool** |  |  |
| **NewGame() :**  **DeleteGame () :**  **EditGame () :**  **SellGame () :**  **ReserveGame () :**  **SignUpWaitlist () :** |  | **rssssssssssssssssssssssssssrk** |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

# ESSENSIEL INFO FRA CASEN:

Udfordring:

"Vi ikke har overblik over, hvilke brætspil vi har på lager, ligesom vi heller ikke har styr på, hvis vi har nogle forespørgsler eller reservationer liggende. "

"Lige nu har vi bare en liste i word, som vi forsøger at holde opdateret - men den bliver meget hurtigt rodet og den er svær at finde rundt i, da den faktiske lidte er på over 50 sider.

I listen forsøger vi at holde styr på hvilke spil vi har på lager, hvilken stand de er i og hvilke spil vi har bestillinger på, hvis de skulle komme ind."

**Det vi gerne vil kunne**

Vi vil gerne have et bedre overblik over, hvilke spil vi har på lager. Vi vil også gerne kunne se hvilken stand det enkelte spil er i.

Vi vil gerne kunne søge på spil ud fra forskellige kriterier. Det kan for eksempel være genre, antal spillere, stand, pris, navn (eller del af navn) og hvis det kan lade sig gøre at kombinere kriterier, vil det være super.

Vi vil selvfølgeligt kunne lægge nye spil ind i systemet - også spil vi ikke har på lager, men som der for eksempel er forespurgt på (Altså i den forstand, at der er nogen, der har bedt os om at finde et bestemt spil til dem) Vi vil gerne kunne se, hvilke spil vi har forespørgsler på og hvem det er, der har forespurgt.

Spillene skal kunne prissættes individuelt, da de jo er i forskellig stand.

Vi vil gerne kunne udskrive en lagerliste til lageroptælling. Den skal både kunne sorteres efter navn og efter genre.

Systemet skal være enkelt at bruge og funktionalitet vægter for os højere end udseendet af systemet.

Kriterier:

* Genre, antal, spillere, stand, pris, navn(eller del af navn), evt. kombinationer af kriterier.
* Indtaste spil, selv hvis det kun er forespørgsel.
* Pris ud fra tilstand.
* Udskrive aktuel lagerliste. (sort efter navn eller genre)
* Funktion over udseende.

Tilstande:

* Ok stand (2)
* Helt ny (1)
* Til rep (4)
* God stand (2)
* Lettere ridset (3)
* Skadet på hjørnet ellers fin stand (3)
* Ikke åbnet (1)
* Lidt slidt (3)
* Lidt skrammet (3)
* Kan måske reddes (5)

# Projektrammer for case: Genspil

Dette projekt er endnu en øvebane for de læringsmål, der arbejdes med i undervisningen. Formålet er at få praktisk erfaring, så I opnår faglige færdigheder og kompetencer.

## Rammer for samarbejde

Projektet gennemføres i grupper. Alle gruppemedlemmer bør øve i alle læringsmål.

## Tidsrammer for projekt:

Der forventes at I arbejder ca. 10 timer per person om ugen.

Projektet løber i ugerne: 11,12,13 ,14 og 15

## Rammer for programmering:

### Brugergrænsefladen

Grænsefladen skal implementeres som en konsol applikation. For at navigere i applikationen. kan i evt. arbejde videre med brugen af det menu system der skulle implementeres i uge 11 opgaven.

I er tidligere stødt på nedstående C# metoder til styring af konsollen:

**Console.WriteLine():** Skriver en streng til konsollen med et linjeskift.

**Console.Write():** Skriver en streng til konsollen uden et linjeskift.

Og metoder til indlæsning fra konsollen.

**Console.ReadLine():** Læser en streng fra konsollen.

Anvend gerne andre Console metoder for at forbedre brugeroplevelsen. Eksempler på metoder kunne være:

**Console.ForegroundColor:** Ændrer konsollens forgrundsfarve.

**Console.BackgroundColor:** Ændrer konsollens baggrundsfarve.

Andre metoder til læsning fra konsollen:

**Console.ReadKey():** Læser en enkelt tast fra konsollen.

**Console.KeyAvailable:** Undersøger om tast er tilgængelig i konsollen.

Yderligere funktioner er:

**Console.Clear():** Rydder konsollen.

**Console.CursorVisible:** Gør konsollens cursor synlig eller usynlig.

**Console.Title:** Sætter konsollens titel.

Læs her mere om de forskellige console metoder og hvordan de anvendes:

<https://learn.microsoft.com/en-us/dotnet/api/system.console?view=net-8.0>

## 

## Persistens

Persistens (i C#) refererer til evnen for et program at gemme data permanent. Det vil sige at programmet husker efter at programmet er afsluttet.

Der er to hovedtyper af persistens i C#:

### Objektpersistens

Objektpersistens er programmets evne til at gemme og gendanne objekter i hukommelsen. Dette kan være nyttigt så data ikke går tabt når et program ikke kører aktivt. Typisk gemmes information om objekterne i filer.

Nedenstående programstump gemmer strengen “Hello\nWorld” i filen fido.txt.

<https://learn.microsoft.com/en-us/dotnet/api/system.io.file.writealltext?view=net-8.0>

Applikationen skal altså kunne gemme indholdet af programmets datastrukturer i en fil. Ligeledes skal applikationen kunne gendanne indholdet ved at indlæse data fra en fil.

Det er denne type for persistens i skal anvende i projektet.

### Datapersistens

Datapersistens er evnen til at gemme data i en database. Det kan være nyttigt eftersom en database har forskellige mekanismer der er skræddersyet til arbejde med data. Vi er endnu ikke nået til databaser, derfor venter vi med at bruge denne type persistens.

## 

## Datastrukturer

Programmets interne datastrukturer indeholder informationer om lister med spil etc.

Datastrukturerne skal implementeres ved brug af klasser og anvendelsen af arrays eller collections.

## Rammer for systemudvikling/virksomheden:

I skal benytte de artefakter I har lært i undervisningen samt kunne vise hvordan jeres systemdesign kan ses i jeres kode (Sporbarhed mellem artefakter og kode fx. i form af klassenavne og attributnavne, at systemet kan benyttes på den måde, som det er beskrevet i use casene etc.)

## Aflevering

Jeres applikation (og dokumentation) skal være uploadet på GitHub. Husk at invitere underviserne ind i jeres GitHub (medmindre jeres repository er offentligt). Upload linket til jeres GitHub i afleveringsmappen på ItsLearning.

Upload et dokument med jeres artefakter samt visning af sporbarhed i afleveringsmappen i itslearning.

Feedback

I virksomheden/systemudvikling gives der feedback (= status) på det afleverede i form af en graduering A, B eller C

A = over det forventede niveau

B = På det forventede niveau

C = ikke helt på det forventede niveau

samt feedforward, der er en kort vejledning til, hvad næste skridt er for at blive bedre.

Mundtlig feedback gives grupperne imellem.

**Præsentation i programmering**:   
Hver fire grupper møder sammen og bruger gruppe i midten struktur, hvor en gruppe præsenter i 10 minutter og så de andre 3 grupper får 5 minutter hver for at stille spørgsmål til gruppe i midten. Så der er brugt 25 minutter og det sidste 5 minutter bruges til at aftale hvilken løsning skal præsenteres i den næste synkron undervisning i programmering. Bagefter præsenter en anden gruppe og bruger det samme struktur indtil alle fire grupper præsenter for hinanden.

Eksempel for præsentation struktur kan være at grupperne 1 til 4 møder i Team 1. Et medlem fra Team 2 holder styr på tiden.

· Team 1 præsenter fra kl. 10:00 til 10:10.

· Team 2 stiller spørgsmål til 10:15 (et medlem fra Team 2 dokumenter).

· Team 3 stiller spørgsmål til 10:20 (et medlem fra Team 3 dokumenter).

· Team 4 stiller spørgsmål til 10:25 (et medlem fra Team 4 dokumenter).

· Et medlem fra Team 1 er ansvarlig for tiden for den næste runde.

· Udskiftning dvs. Team 2 præsenter og så den samme struktur bruges for det næste 30 minutter.

· Udskiftning dvs. Team 3 præsenter og så den samme struktur bruges for det næste 30 minutter.

Brug denne struktur for dette projekt for at møde så mange medstuderende som muligt og lære hinanden mere at kende:

Team 5, 9, og 13 møder i Team 1.

Team 6, 10, og 14 møder i Team 2.

Team 7, 11, og 15 møder i Team 3.

Team 8, 12, og 16 møder i Team 4.

Præsentation for hinanden finde sted mandag d. 15 april kl. 10.

I præsenterer de fire løsninger I bliver enige om tirsdag d. 16 april når vi mødes til programmering undervisning. Fokus her er det samme som præsentation for hinanden samt hvad der gik godt og mindre godt.