Oprettelse af Lagersystem til butikken Genspil

Oprette og læse til og fra fil

Deltager(e):

Nickolai S. C. Nygaard
Alex Holmbo

Vejleder(e):

Anja Birkelund
Lene Vestergaard Andersen
Diaa Zobair Shollar
Thomas Tjellesen

Afleveringsdato:

08-04-2025

Uddannelse / Skolens navn:

Online Datamatikker, UCL Erhvervsakademi og Professionshøjskole

Abstract

Dette projekt har til formål at bruge det, vi har lært fra systemudvikling og programmering. Vi skal udvikle en konsol applikation for butikken *Genspil*, som skal håndtere deres lager (*Stock*). Vi samarbejder via GitHub og anvender SCRUM til fordeling af opgaver.

Vi bruger centrale emner fra vores undervisning, såsom systemudviklingsmodellerne DM, OM, DCD, BPMN og SSD, for at kunne analysere og forstå virksomhed flowet, samt tage deres eksisterende fagtermer med i udviklingen af vores konsol applikationer. Derudfra sørger vi for sporbarhed gennem alle elementerne af vores produkt.

Applikationen skal kunne håndtere og vedligeholde informationer om brætspil og varianter, henholdsvis *BoardGameVariant* og *BoardGameCopy*. Den skal kunne oprette et nyt brætspil og udskrive informationerne derfra, samt tilføje en *BoardGameCopy* til et eksisterende brætspil. Derudover skal brugeren kunne søge på diverse spil, der er blevet oprettet i systemet. Systemet er bygget op af klasser, collections og metoder. Data skal gemmes og kunne genindlæses fra en fil for at sikre persistens.

Indholdsfortegnelse

1. Introduktion

Casen for butikken Genspil har handlet om at uddybe os i et projekt med specifikke rammer for, hvad der forventes af vores konsolapplikation, og levere et relevant produkt til de specifikationer, der er angivet i casen. Heriblandt at kunne tilføje nye spil til lageret, udskrive lagerlisten, søge på brætspil, der er i lageret, ud fra specifikke kriterier (navn, genre, antal spillere) og at kunne tilføje ønsker om nye produkter sammen med den relevante person, der har lavet henvendelsen. Derudover vil de også gerne kunne prissætte de forskellige versioner af spil, da de har forskellige stande – nogle er slidte, og andre er nye.

Vores system er en C#-baseret konsolapplikation, der har til formål at opfylde de specifikke krav fra casen Genspil. Ud fra use cases for de forskellige funktioner, vores system skal indeholde, er der blevet brugt relevante modeller som:

1. Domænemodellen (DM),
2. Objektmodellen (OM),
3. Design Class Diagram(DCD),
4. Business Process Model (BPMN),
5. System Sequence Diagrams (SSD),

som stammer fra systemudvikling og forretningsudvikling. Disse modeller er blevet brugt til at analysere flowet og skabe en ensartet navngivning og forståelse for, hvordan man navigerer i vores konsol applikation. Vi bruger en dynamisk menu, der er nem og hurtig at navigere i, for at hjælpe brugeren med at opnå sit mål – hvad end det er at oprette nye spil, tilføje kopier med forskellige tilstande eller søge i systemet efter oprettede spil ud fra specifikke kriterier.

Teknologier, der er blevet brugt, er C# til programmering af konsol applikationen. Vi har implementeret persistens for at sikre, at man kan skrive til filer, og at data bliver gemt i en tekstfil. Derudover samles data fra BoardGameCopy til de forskellige BoardGameVariant-objekter i systemet, hvor data opbevares i tekstfilen games.txt. Det gør det muligt altid at opdatere og tilføje flere spil, og sikre, at data bliver gemt, så det kan læses fra filen næste gang konsol applikationen åbnes.

2. Projektoversigt og proces

2.1 Udviklingsmetode – SCRUM

Vi havde fokus på at opsætte SCRUM og uddelegere opgaver derigennem. Vi startede med at udvælge en Product Owner og en Scrum Master.

- **P.O.**: Ayesha
- **Scrum Master**: Alex Holmbo

Derefter startede vi et sprint 0, hvor vi afklarede relevante arbejdsmidler som Lucidchart med henblik på at lave konkrete modeller inden for de forskellige use cases.

Vi anvendte GitHub både til at styre SCRUM-processen og til at dele koden med hinanden. Det gav os et bedre samarbejde og mulighed for at håndtere konflikter i koden og samle dem igen på en hensigtsmæssig måde.

Vi opsatte brugsscenarier (use cases) klart og tydeligt og lavede de nødvendige modeller, som vi skulle arbejde ud fra. Det sikrede, at vi havde en ensartet terminologi og dermed undgik konflikter eller duplikering af kode ved forskellige navngivninger.

Derudover aftalte vi en fælles kode konvention: Vi benyttede både snake_case og camelCase afhængigt af konteksten.

Sprint 1 og frem

Efterfølgende gik vi videre til Sprint 1, hvor vi nedbrød de forskellige opgaver i mindre dele. Med hjælp fra GitHubs SCRUM-værktøjer kunne vi byde ind på opgaver, fordele arbejdet og committe koden direkte, efterhånden som vi færdiggjorde de enkelte dele.

2.2 Domæneanalyse

Forklaring af problemområdet

Projektet handler om et **lagersystem**, hvor medarbejderne skal kunne:

- Tilføje spil og kopier af spil
- Søge efter spil og varianter
- Registrere køb

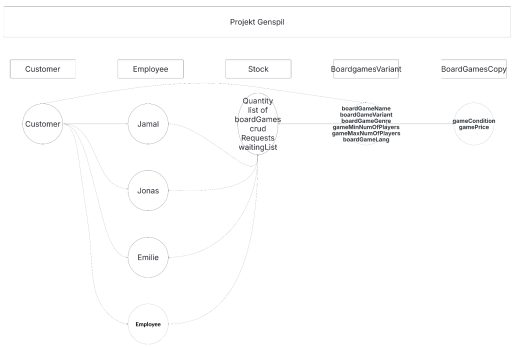
- Tilføje spil til en ønskeliste, hvis det ikke er tilgængeligt

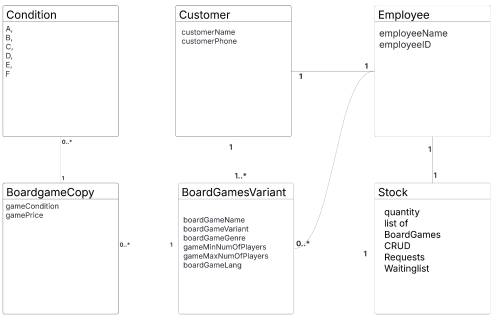
projektet handler om et lagersystem hvor medarbejderne tilføje spil og kopier af spil og kunne søge efter spil samt diverse varianter og Data på spil registrere køb eller at tilføje spil til en ønskeliste hvis det ikke er tilgængeligt

Domænemodel – første udkast

Vores første domænemodel blev baseret på en objektmodel, hvor vi analyserede forretnings flowet: En kunde kommer ind, kontakter en medarbejder og spørger til et bestemt spil. Medarbejderen slår op i lageret og tjekker, om spillet er tilgængeligt.

Ud fra dette vurderede vi, hvilke informationer der var relevante at registrere for henholdsvis Employee, Customer, BoardGameVariant, BoardGameCopy, Condition og Stock.





[Appendix 1 (OM) Objekt Model]

[Appendix 2 (DM) Domæne Model]

Eksempler på begreber i domænemodellen:

Customer: customerName, customerPhone

Emlpoyee: employeeName, employeeID

boardgameVariant: boardGameName, boardgameVariant, boardGameGenre, gameMaxNumOfPlayers, gameMinNumOfPlayers, gameLang

BoardGameCopy: gameCondition, gamePrice

Condition: A, B, C, D, E, F

Stock: quantity, requests, waitingList

3. Analyse

3.1 Use Case Model

Use Case 1: Tilføj nyt brætspil (BoardGameVariant)

[Appendix 4 (SSD) System Sequence Diagram for tilføjning af BoardGameVariant]

Formål:
At tilføje en ny brætspils variant (BoardGameVariant) til lageret, så det bliver tilgængeligt i systemets database og kan gemmes til fil.

Primær aktør:
Medarbejder (Employee)

Trigger:
Medarbejderen vælger "Add new game to the stock" i game management-menuen.

Preconditions:

Systemet er startet og kører.

Medarbejderen er i hovedmenuen og vælger "Game management".

Spillet findes ikke i systemet i forvejen.

Postconditions:

En ny BoardGameVariant er oprettet og tilføjet til listen (List<BoardGameVariant>).

Varianten er gemt i tekstfilen `games.txt` via `Stock`.

Medarbejderen føres tilbage til game management-menuen.

Hovedforløb:

1. Medarbejderen vælger "1. Game Management" fra hovedmenuen.
2. Systemet viser Game Management-menuen.
3. Medarbejderen vælger "1. Add a new game to the stock".
4. Systemet beder om følgende oplysninger:

Navn på spil

Variant

Genre

Minimum antal spillere

Maksimum antal spillere

Sprog

5. Medarbejderen indtaster oplysningerne.
6. Systemet opretter et nyt BoardGameVariant-objekt.
7. Varianten tilføjes til listen.
8. Data gemmes til fil via `Stock`.
9. Systemet viser en bekræftelse og returnerer til game management-menuen.

Use Case 2: Tilføj ny Copy af spil (BoardGameCopy)

[Appendix 5 (SSD) System Sequence Diagram for tilføjning af af Copy til BoardGameCopy]

Formål:
At tilføje en ny kopi (BoardGameCopy) til en eksisterende variant i systemets lager, inkl. stand (condition) og pris.

Primær aktør:
Medarbejder (Employee)

Trigger:
Medarbejderen vælger "Add copy to the stock" i game management-menuen.

Preconditions:

Systemet er startet og kører.

Medarbejderen er i "Game Management"-menuen.

Der findes mindst én BoardGameVariant i systemet.

Postconditions:

En ny BoardGameCopy er tilføjet til en eksisterende variant.

Kopien gemmes i kopi-listen og i filen `copies.txt`.

Medarbejderen føres tilbage til game management-menuen.

Hovedforløb:

1. Medarbejderen vælger "1. Game Management" fra hovedmenuen.
2. Systemet viser Game Management-menuen.
3. Medarbejderen vælger "2. Add copy to the stock".
4. Systemet tilbyder søgning på spilnavn.
5. Medarbejderen indtaster spillets navn.
6. Systemet viser matchende varianter.
7. Medarbejderen vælger en variant.
8. Systemet spørger om spillets stand (enum).
9. Medarbejderen vælger stand.
10. Systemet beder om pris.
11. Medarbejderen indtaster prisen.
12. Systemet opretter en kopi og knytter den til varianten.
13. Kopien gemmes til filen.
14. Systemet viser bekræftelse og sender medarbejderen tilbage til game management-menuen.

Alternative Flows:

AF1: Spillet findes ikke i systemet
Systemet viser fejlbesked og sender brugeren tilbage.

AF2: Forkert variant- eller condition-valg
Systemet viser fejl og spørger igen.

AF3: Ugyldig pris
Systemet beder medarbejderen indtaste korrekt pris igen.

Use Case 3: Søg efter brætspil og køb/ønskeliste

[Appendix 6 (SSD) System Sequence Diagram søg efter brætspil ønske om køb/ønskeliste]

Formål:
At hjælpe en kunde med at finde og eventuelt købe et spil eller tilføje det til ønskelisten, hvis det ikke er på lager.

Primær aktør:
Medarbejder (Employee)

Trigger:
Kunden henvender sig og spørger efter et bestemt spil.

Preconditions:

Medarbejderen er logget ind og har adgang til systemet.

Systemet indeholder varianter og kopier.

Kunden leder efter et bestemt spil.

Postconditions:

Spillet bliver fundet og enten solgt eller sat på ønskelisten.

Eller kunden vælger ikke at købe eller ønsker ikke at oprette ønskelisten.

Hovedforløb:

1. Kunden henvender sig og spørger efter spillet.
2. Medarbejderen åbner søgemenuen.
3. Medarbejderen vælger søgning via navn.
4. Systemet beder om spillets navn.
5. Medarbejderen indtaster navn.
6. Systemet viser varianter og kopier (stand og pris).

7. Kunden vælger en kopi.

8. Medarbejderen registrerer salget.

Alternative Flows:

AF1: Kunden vil ikke købe spillet
Medarbejderen afslutter interaktionen.

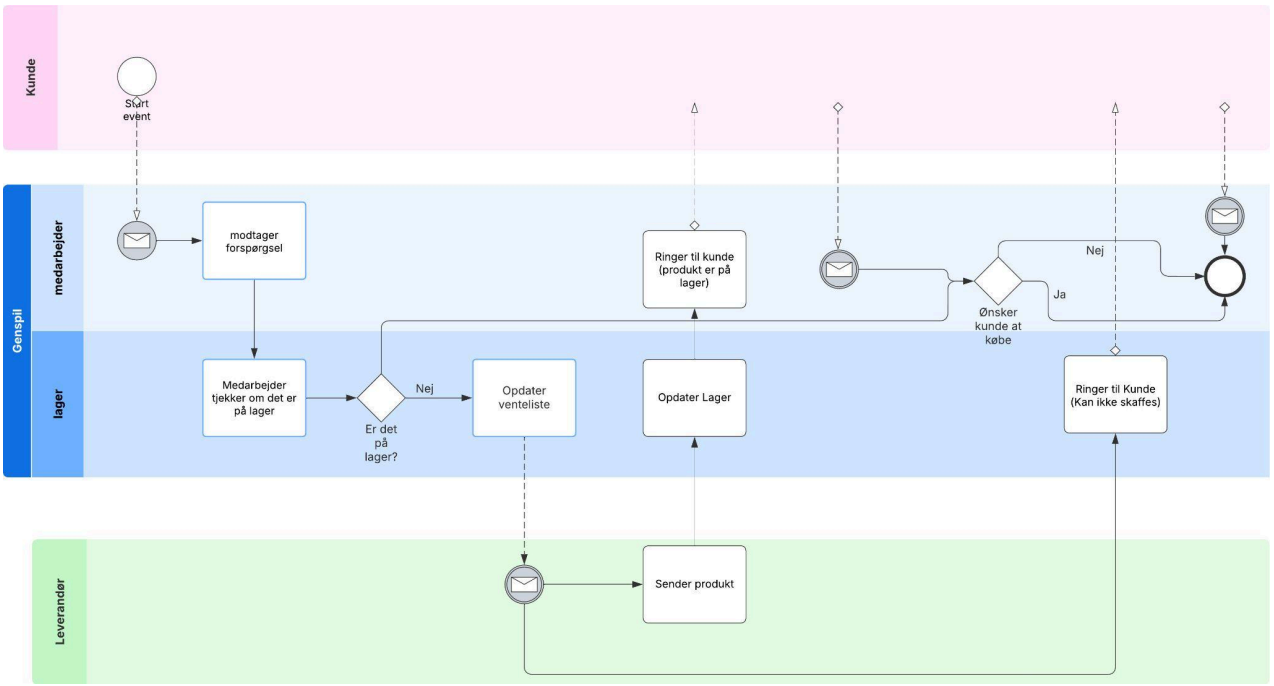
AF2: Spillet er ikke på lager
a. Systemet spørger om kunden vil på ønskeliste
b. Hvis ja:
Medarbejderen indtaster navn, mail og telefonnummer

Systemet gemmer det og bekræfter
c. Hvis nej:

Medarbejderen afslutter dialogen

AF3: Spillet findes ikke i systemet
Ingen handling muligt, medarbejderen afslutter.

3.2 BPMN-diagram



[Appendix 7 (BPMN) Business Process Model and Notation]

Employee modtager et request fra en Customer.

Employee tjekker, om det er i Stock.

Flow 1 – Boardgame er i Stock

Boardgame er i Stock.

Employee spørger, om Customer vil købe spillet.

Hvis ja: Makes Sale.

Flow 2 – Boardgame er ikke i Stock (kan skaffes)

Boardgame er ikke i Stock.

Employee spørger, om Customer vil på waitList.

Customer siger ja.

Employee opdaterer waitList.

Request bliver sendt til Leverandør.

Leverandør sender Boardgame til Genspil.

Stock bliver opdateret.

Customer får besked om, at Boardgame er i Stock.

Customer kommer tilbage til butikken.

Employee makes Sale.

Flow 3 – Boardgame er ikke i Stock (kan ikke skaffes)

Boardgame er ikke i Stock.

Employee spørger, om Customer vil på waitList.

Customer siger ja.

Employee opdaterer waitList.

Request bliver sendt til Leverandør.

Leverandør kan ikke skaffe Boardgame til Genspil.

Genspil får besked.

Employee giver Customer besked.

Flow 4 – Customer ønsker ikke at blive skrevet på waitList

Employee tjekker, om Boardgame er i Stock.

Hvis nej:

Employee spørger, om Customer vil oprettes på waitList.

Hvis nej:

Employee stopper salget.

4. Design

4.1 DCD – Design Class Diagram

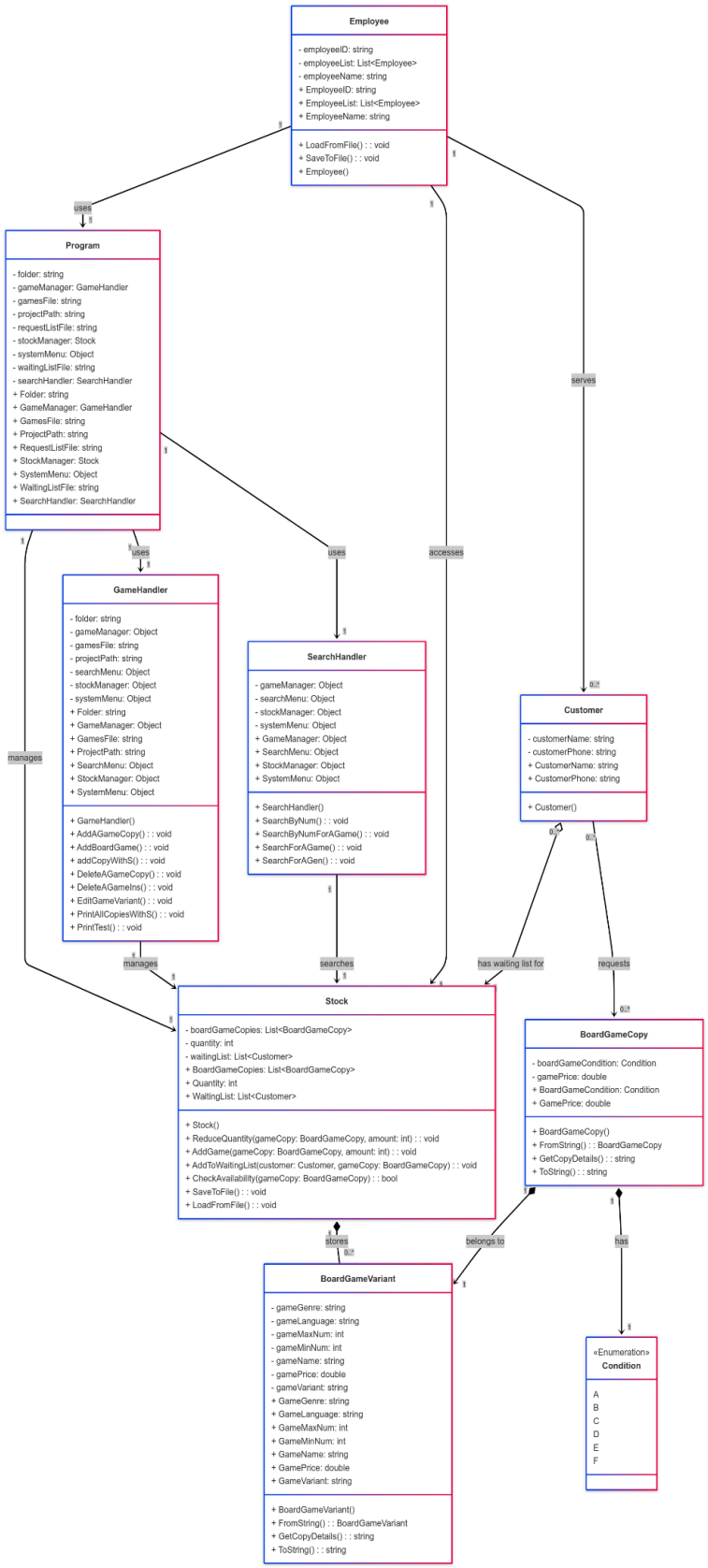
- Hvordan klasserne hænger sammen

  Employee bruger Program, og håndterer Customer og tilgår Stock.
  Program bruger GameHandler og SearchHandler
  Stock indeholder BoardGameVariants og RequestList fra Customer

- Arv, association, aggregering
  Fra stock til Customer er der en aggregering, Customer kan godt eksistere uden Stock.

  Enum er en komposition på BoardGameCopy, som er en komposition på BoardGameVariant, som er en komposition på Stock.
  De kan ikke eksistere uafhængigt af hinanden

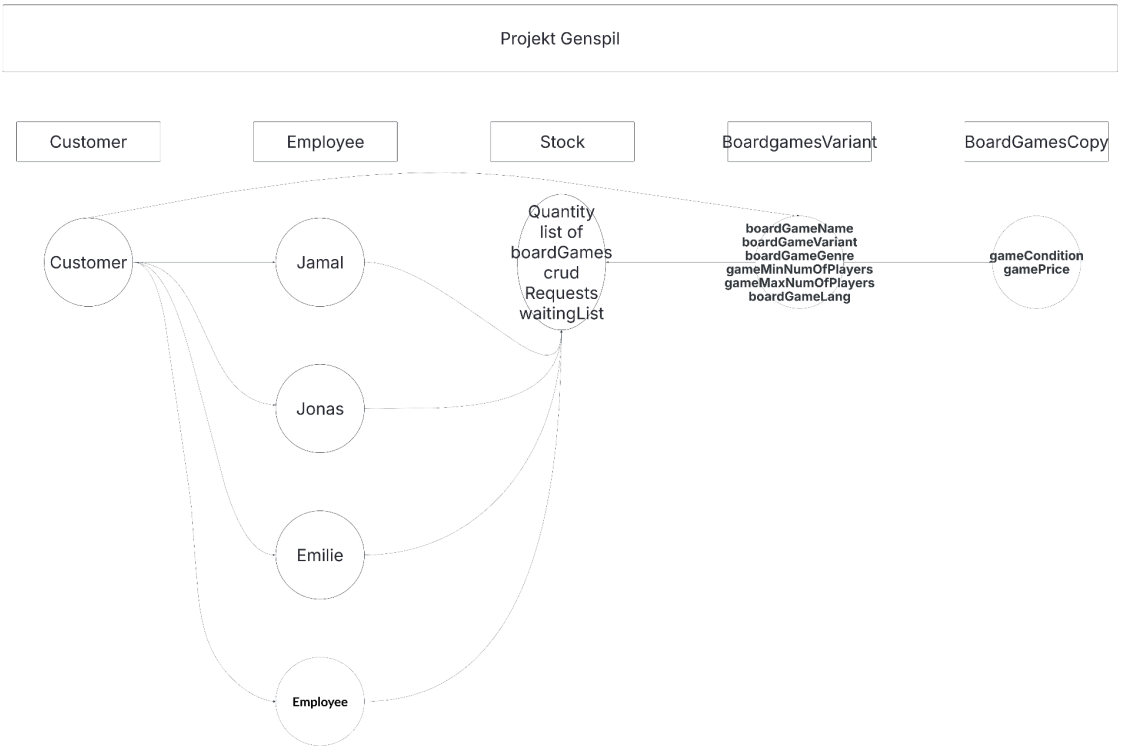
  [DCD på næste side]



[Appendix 3 (DCD) Design Class Diagram]

4.2 Objektmodel

- Eksempel på hvordan en spilvariant og dens kopier eksisterer som objekter i systemet.



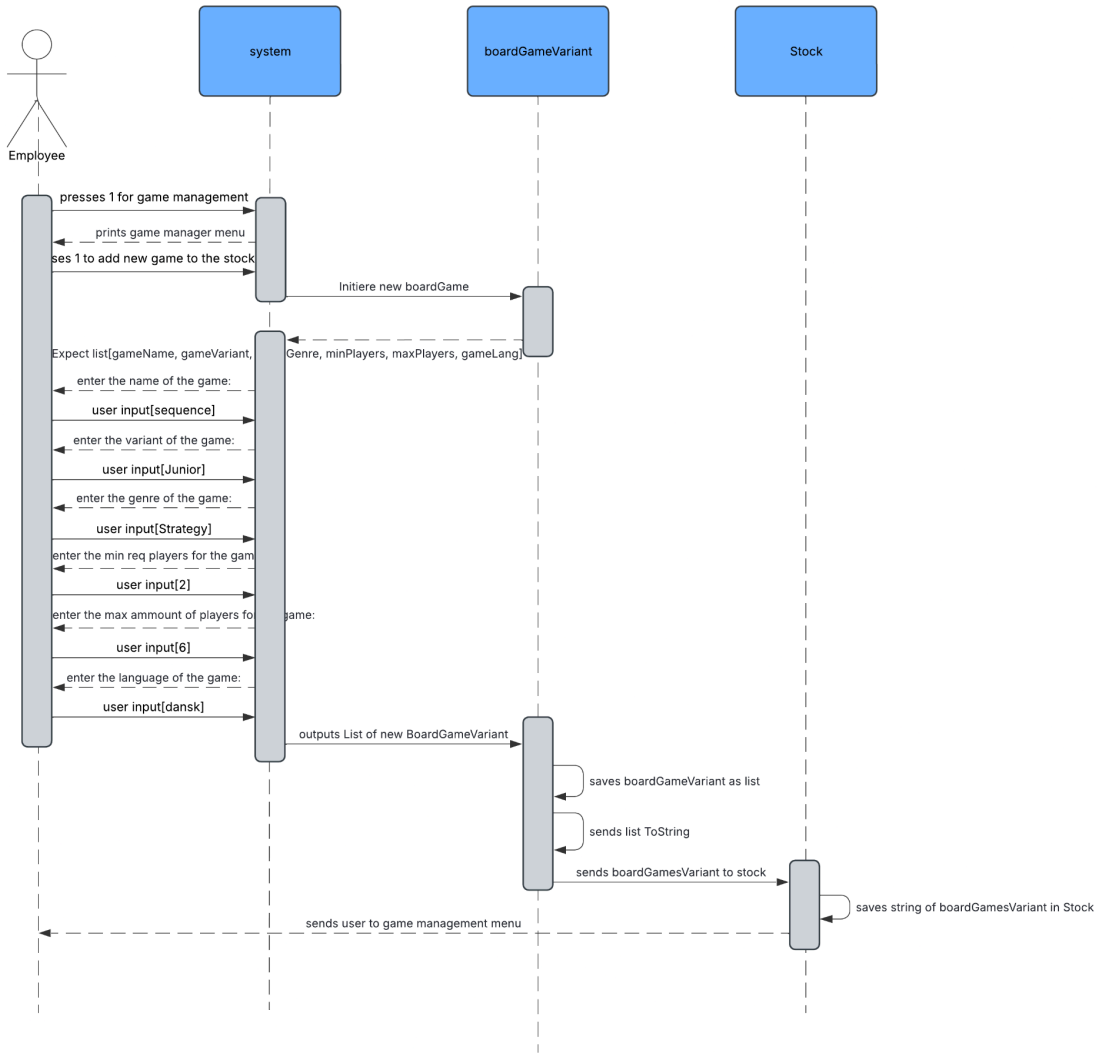
[Appendix 1 (OM) Objekt Modellen]

Objektmodellen starter med, at en Customer har mulighed for at henvende sig til en Employee og lave en forespørgsel på et BoardGame. Herefter tjekker Employee systemet for at se, om boardgamet er i Stock. Hvis det er tilfældet, vises hvilke Condition- og Price-varianter der er tilgængelige for det pågældende spil.

En Customer har desuden mulighed for at browse i butikken og foretage forespørgsler på andre spil, som vedkommende har set.

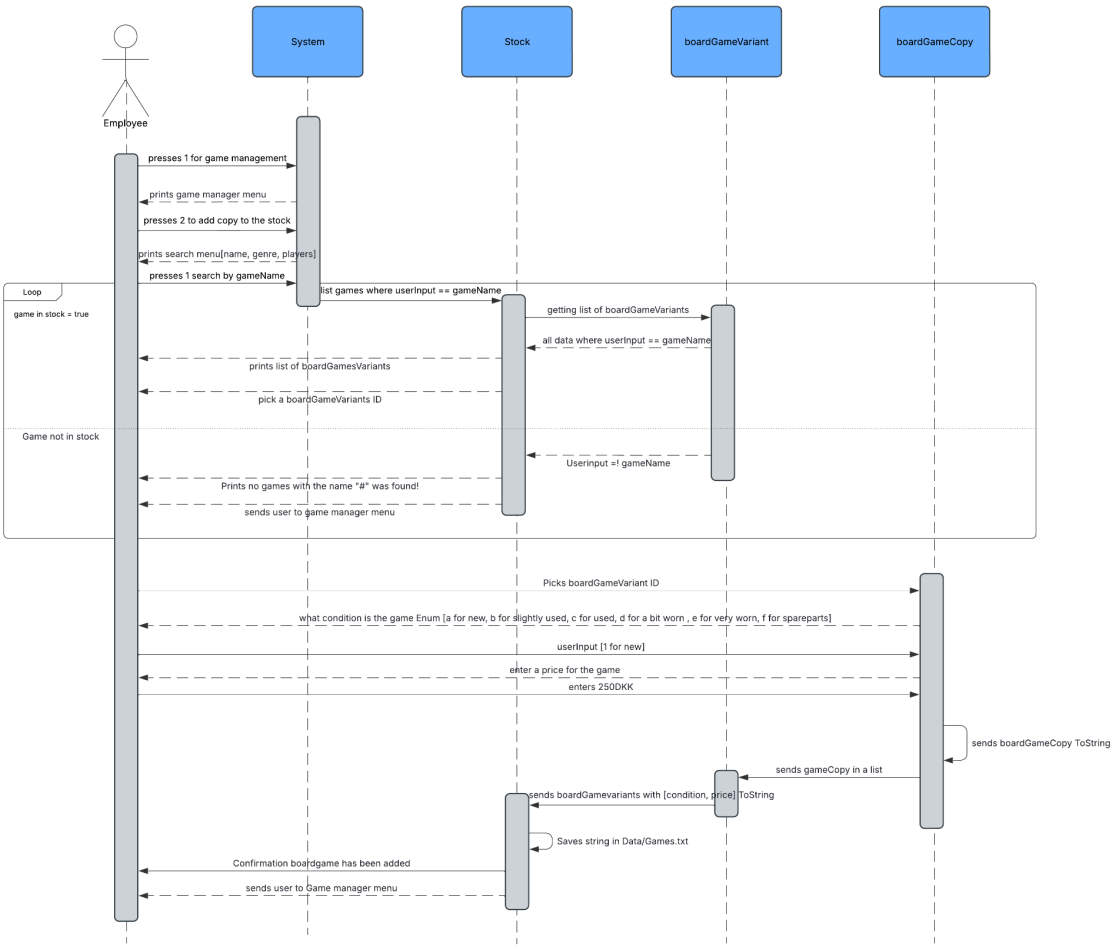
4.3 System Sequence Diagrams (SSD)

som beskrevet i [3.1 Use Case: Tilføj nyt BoardGameVariant]



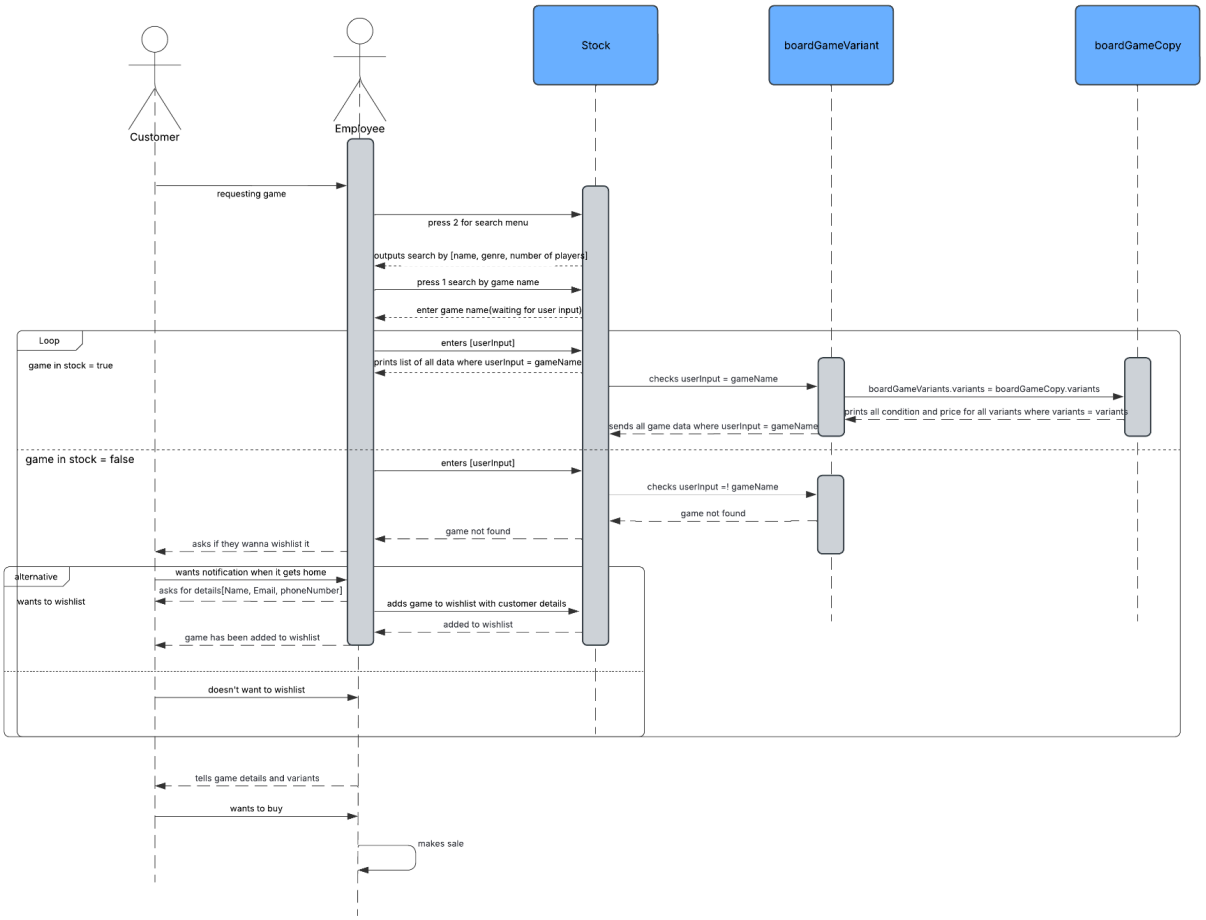
[Appendix 4 (SSD) System Sequence Diagram for tilføjning af BoardGameVariant]

som beskrevet i [3.1 Use Case: Tilføj Copy til BoardGameCopy]



[Appendix 5 (SSD) System Sequence Diagram for tilføjning af af Copy til BoardGameCopy]

som beskrevet i [3.1 Use Case: Tilføj nyt BoardGameVariant]



[Appendix 6 (SSD) System Sequence Diagram søg efter brætspil ønske om køb/ønskeliste]

5. Implementering

- Hvordan systemet er bygget op:

  Klassen `BoardGameVariant` indeholder `Constructor`, `ToString`, `FromString`, `Add/Remove` og `GetGameDetails` metoder.

  Klassen `BoardGameCopy` indeholder `Constructor`, `ToString`, `FromString` og `GetCopyDetails` metoder.

  Klassen `Customer` indeholder `Constructor`, `ToString`, `FromString` og `GetRequestDetails` metoder.

  `Condition` enum
  [Appendix 8 Enum for Condition]

  `Stock` til persistens

  `Program.cs` håndtere menuerne og deres switch-cases.

  `GameHandler` til håndtering af metoder til `BoardGameVariant` og `BoardGameCopy.`
  [Appendix 9 Kodeudsnit fra `GameHandler` DeleteAGameCopy metode `BoardGameCopy`]

  `SearchHandler` til håndtering af søgefunktionerne.

  `Customer` indeholder `Constructor`, `FromString`, `ToString` og `GetRequestDetails` metoder

- Persistens: brug af `StreamWriter`, `StreamReader`, filstruktur

6. Resultater

- Hvad fungerer i systemet?

Vores system fungerer ved, at man åbner op til menuer, der er baseret på switch cases [Appendix 10 BrugerScenarier og menuer], som gør det muligt at navigere til oprettelse af spil og oprettelse af kopier på spillet samt give mulighed for at fjerne kopier fra spil. Derudover kan vi også lave forespørgsler på spil og printe alle kopierne af et specifikt spil. Det er også blevet gjort muligt at fjerne et spil helt fra systemet, inklusiv de pågældende kopier. Vi har en søgefunktion, som kan håndtere søgning efter spillets navn, genre og antallet af spillere, og man kan udskrive alle spil, der er på lager. Vi kan se en liste over forespørgsler, der er på spil, og til sidst har vi implementeret persistence, så vi kan gemme til en .txt-fil og hente data fra en .txt-fil [Appendix 11 Eksempel på games.txt med tilhørende copies tilknyttet det enkelte objekt.]

- Konsol Output eller testresultater
  Vi har lavet unit tests på de forskellige metoder tilknyttet BoardGameVariant – om alle parametre sender de rigtige værdier, om Copies sender den rigtige værdi, og om ToString sender den rigtige værdi. Som det ses i vores
  [Appendix 12 Unit tests.], blev alle 9 tests gennemført succesfuldt og tog i alt 45 ms.

- Brugervenlighed?

  Systemet er nemt, hurtigt og brugervenligt. De dynamiske menuer gør at systemet føles veludviklet. Det er simpelt bygget op, og navigere rundt i ved hjælp af tastetryk [Appendix 10 BrugerScenarier og menuer]

- Filer gemt korrekt?

  Filer bliver gemt og hentet korrekt, der er ingen korruption eller fejl i strings når de er hentet ind igen.
  [Appendix 11 Eksempel på games.txt med tilhørende copies tilknyttet det enkelte objekt.]

7. Konklusion

- Har I opfyldt målet?

  Vi har overordnet med vores gruppes udfordringer opfyldt målet om at lave et MVP (Minimum Viable Prototype), som vi ville kunne præsentere for en kunde som Genspil.

  VI har opfyldt vores egne krav om funktionalitet og brugervenlighed

- Hvad lærte I om design / udvikling / struktur?

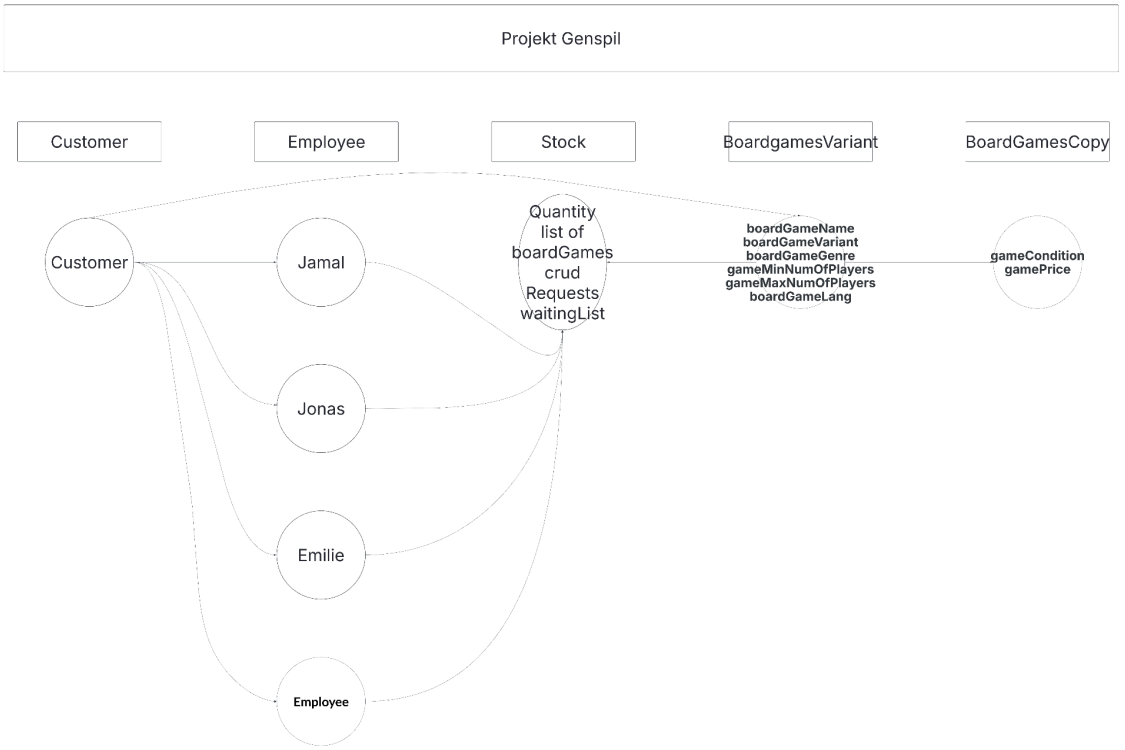
  Vi har lært, hvor vigtigt det er at bruge HLD, LLD og kode struktur, da det sikrer sporbarheden. Da vi på baggrund af at vi kun er 2, måtte ofre nogle artefakter og SCRUM undervejs.
  Vi har godt kunne mærke, at den overordnede struktur og den agile udvikling manglede, for at holde orden i projektet

8. Fremtidigt arbejde

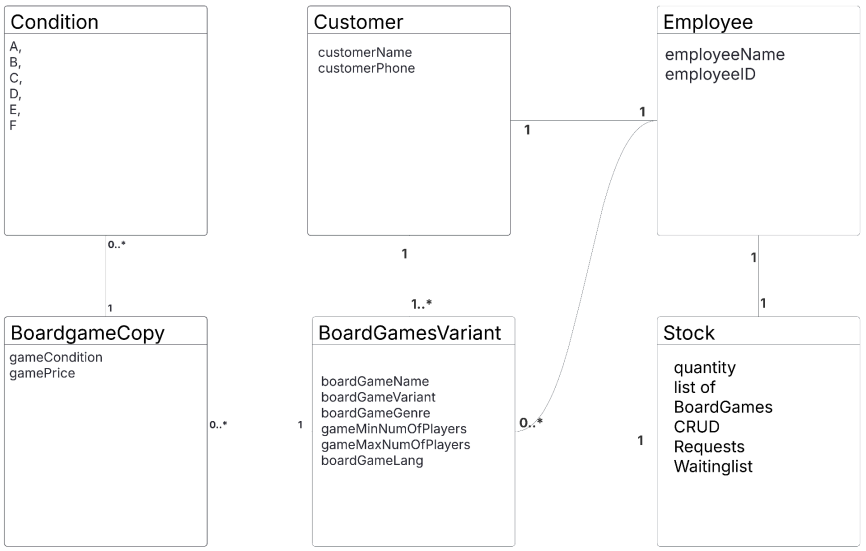
- GUI i stedet for konsol
- Database i stedet for tekstfiler
- Mere avancerede søgefunktioner
- Statistik over brug/kopier

9. Appendix

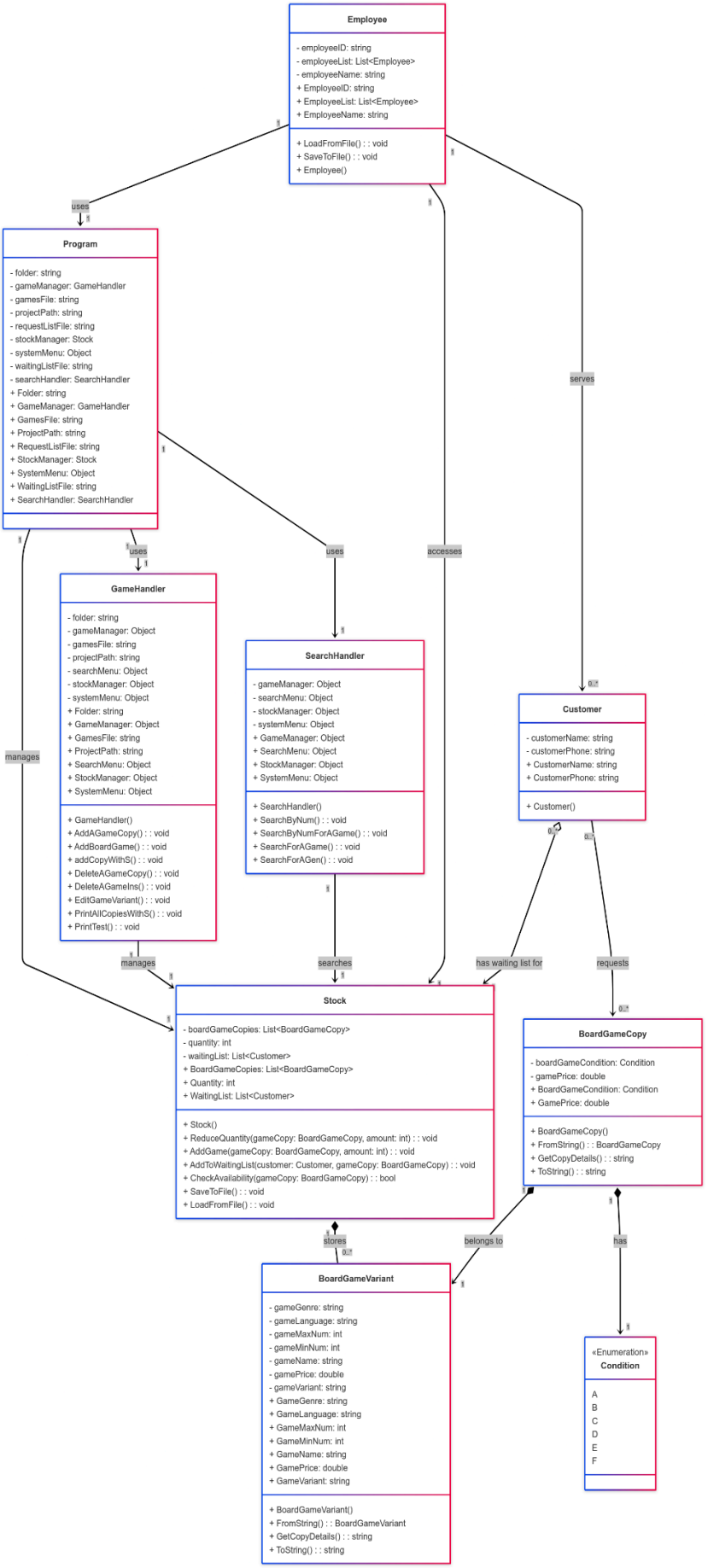
[Appendix 1 (OM) Objekt Modellen]



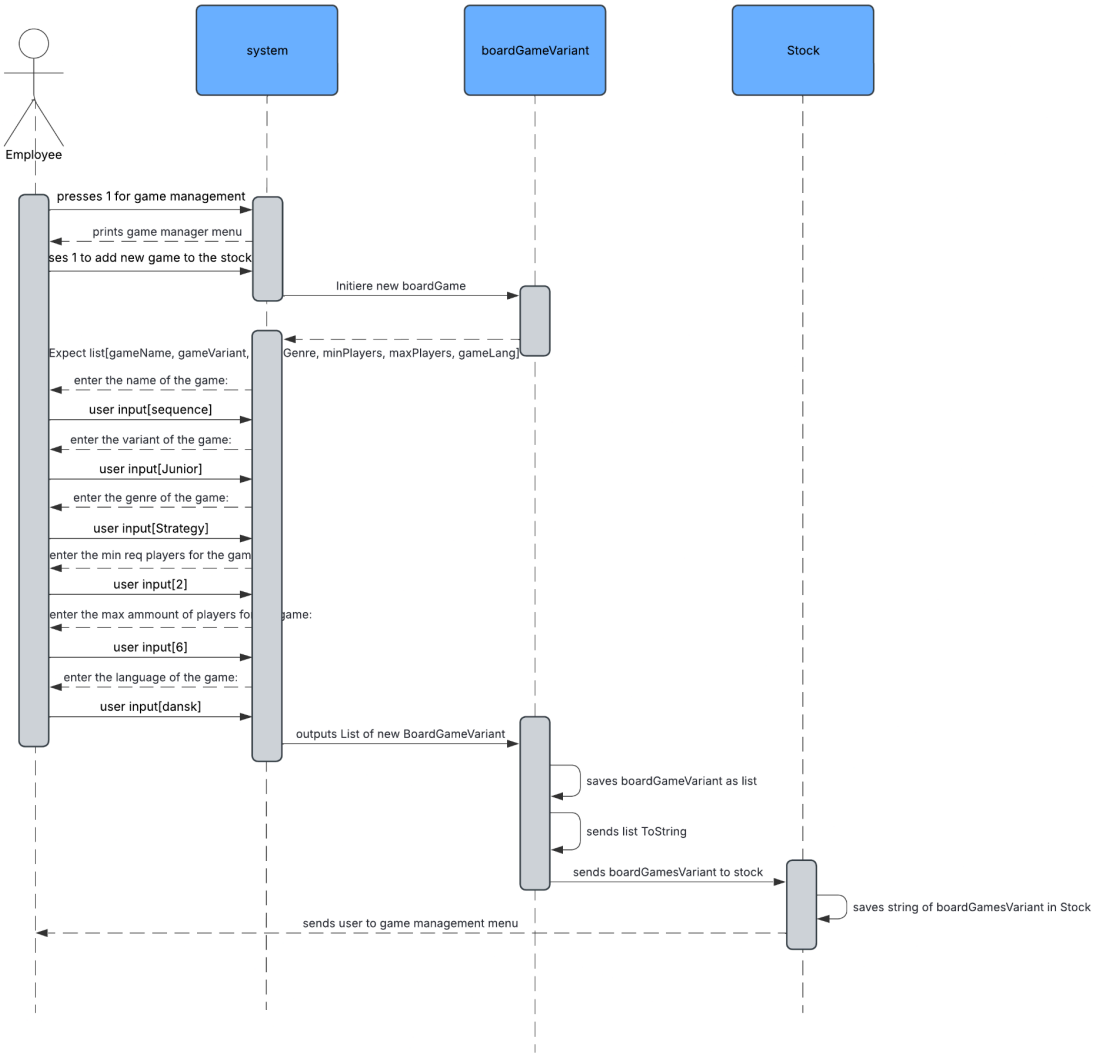
[Appendix 2 (DM) Domænemodellen]



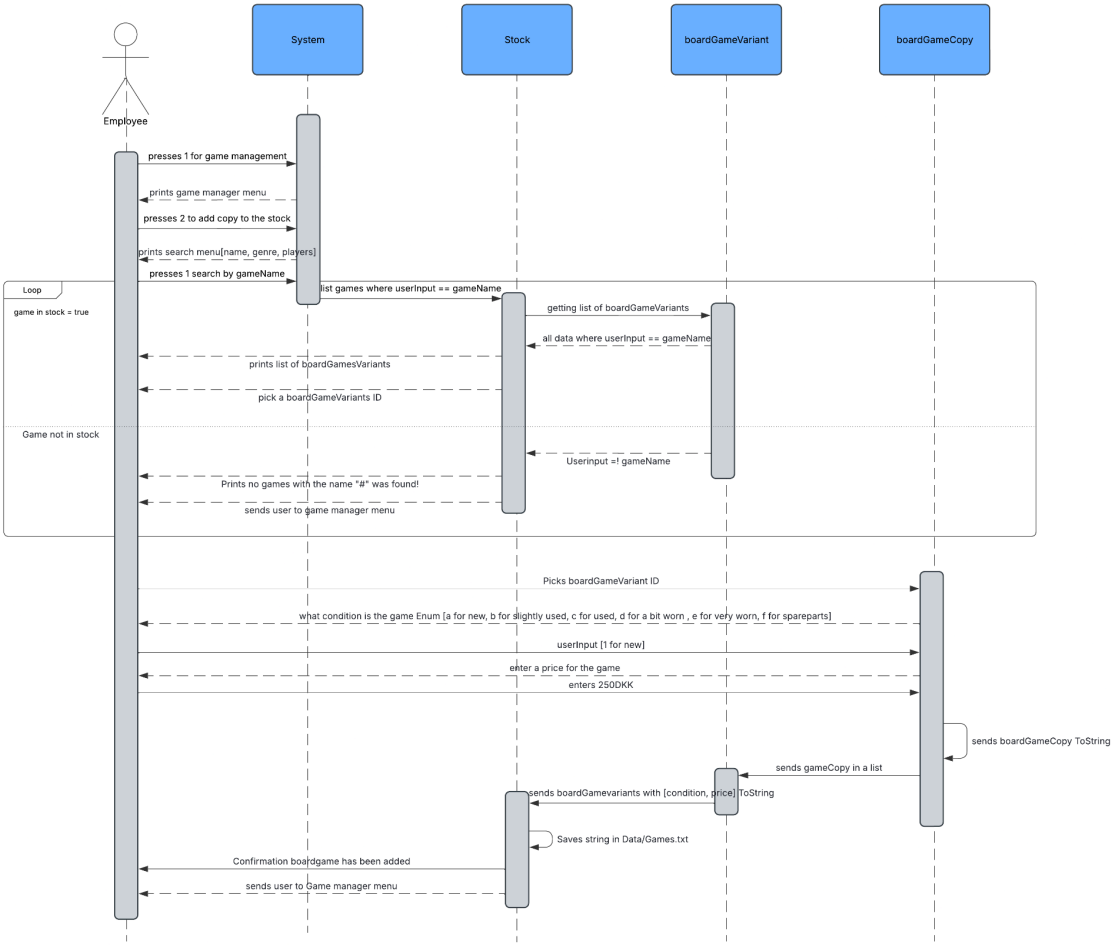
[Appendix 3 (DCD) Design Class Diagram]



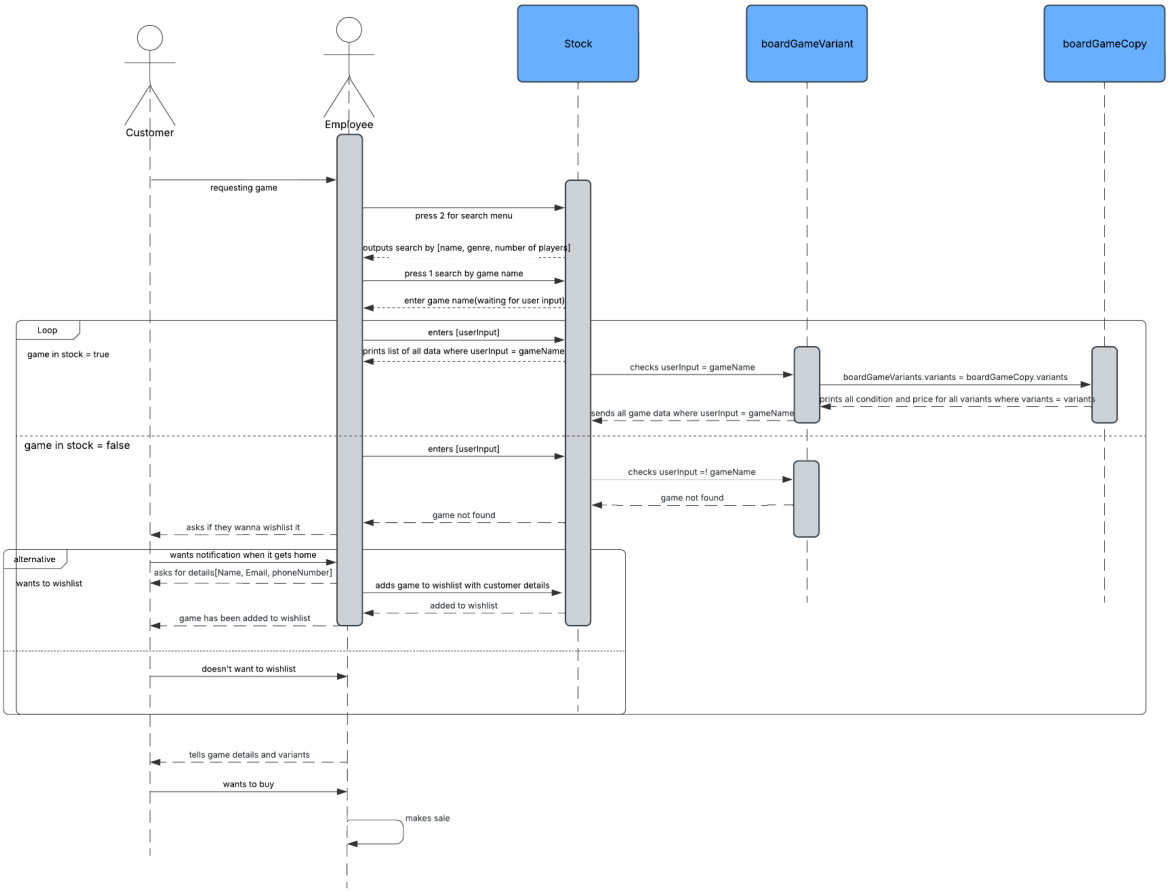
[Appendix 4 (SSD) System Sequence Diagram for tilføjning af BoardGameVariant]



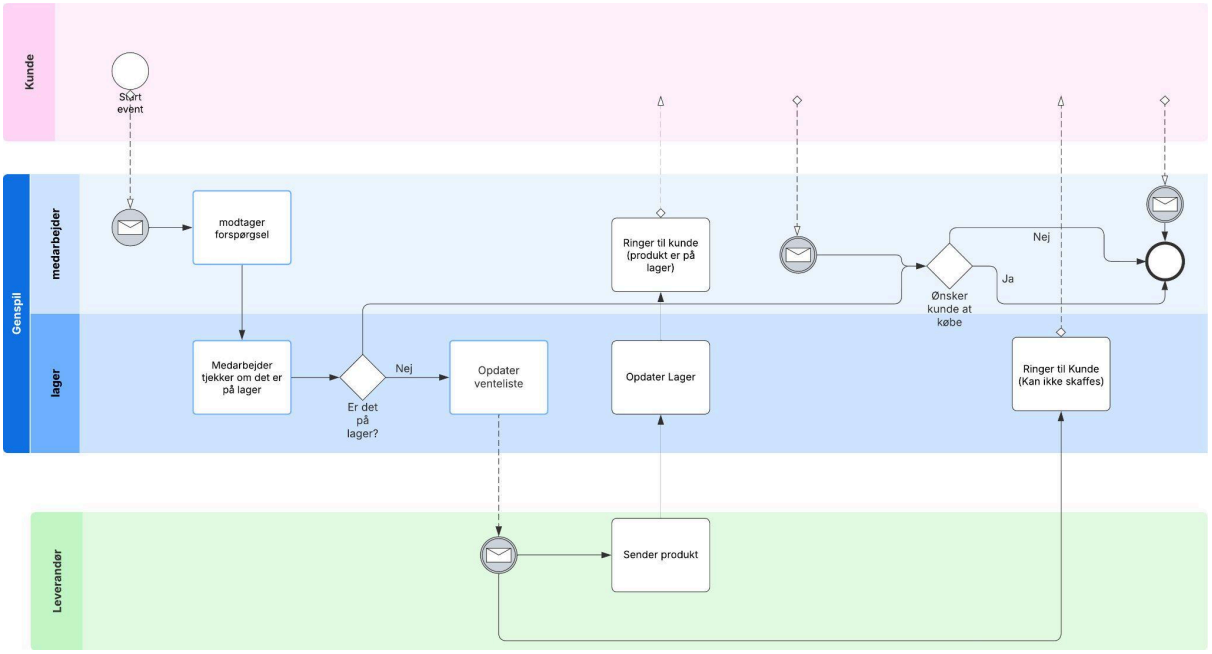
[Appendix 5 (SSD) System Sequence Diagram for tilføjning af af Copy til BoardGameCopy]



[Appendix 6 (SSD) System Sequence Diagram søg efter brætspil ønske om køb/ønskeliste]



[Appendix 7 (BPMN) Business Process Model and Notation]



[Appendix 8 Enum for `Condition`]

```
namespace ProjGenspil
{
    7 references | WildclawsDK, 3 hours ago | 1 author, 2 changes
    public enum Condition
    {
        A,
        B,
        C,
        D,
        E,
        F


    }
}
```

[Appendix 9 Kodeudsnit fra `GameHandler` DeleteAGameCopy metode `BoardGameCopy`]

Kode Udsnit

```
public static void DeleteAGameCopy(Stock stock)
{
    Console.WriteLine(gameManager);
    Console.SetCursorPosition(0, 16); // Move cursor below the menu
(line 16)
    Console.Write("Enter the games name:
".PadRight(Console.WindowWidth));
    Console.SetCursorPosition("Enter the games name: ".Length, 16);



    string search = Console.ReadLine();
    bool foundGame = false;

    for (int i = 0; i < stock.Games.Count; i++)
    {
        if (stock.Games[i].GameName.ToLower() == search.ToLower())
        {
            Console.Write($"\nGame index number: {i + 1}
{stock.Games[i].GetGameDetails()}\n");
            foundGame = true;
        }
    }
    if (!foundGame)
    {
        Console.Clear();
        Console.WriteLine(gameManager);
        Console.SetCursorPosition(0, 16);
        Console.Write($"No games with the name \"{search}\" was
found!");
        //return -1;
    }

    Console.Write("\nEnter the games index number: ");


    int gameIndex = Convert.ToInt32(Console.ReadLine()) - 1;
```

```
    if (gameIndex < 0 || gameIndex > stock.Games.Count - 1)
    {
        Console.Clear();
        Console.WriteLine(gameManager);
        Console.SetCursorPosition(0, 16); // Move cursor below the menu
(line 15)
        Console.WriteLine("Invalid input, please try
again!".PadRight(Console.WindowWidth)); // Clear line
        Console.SetCursorPosition(0, 16); // Move cursor below the menu
(line 15)
        //return -1;
    }


    for (int i = 0; i < stock.Games[gameIndex].BoardGameCopies.Count;
i++)
    {
        Console.Write($"\nCopy index number: {i + 1}
\n{stock.Games[gameIndex].BoardGameCopies[i].GetCopyDetails()}\n");

    }

    Console.Write("\nEnter the copy index number: ");
    int copyIndex = Convert.ToInt32(Console.ReadLine()) - 1;

    if (copyIndex < 0 || copyIndex >
stock.Games[gameIndex].BoardGameCopies.Count - 1)
    {
        Console.Clear();
        Console.WriteLine(gameManager);
        Console.SetCursorPosition(0, 16); // Move cursor below the menu
(line 15)
        Console.WriteLine("Invalid input, please try
again!".PadRight(Console.WindowWidth)); // Clear line
        Console.SetCursorPosition(0, 16); // Move cursor below the menu
(line 15)
        //return -1;
    }

    stock.Games[gameIndex].BoardGameCopies.RemoveAt(copyIndex);
    stock.SaveToFile(gamesFile);

    Console.Clear();
    Console.WriteLine("\x1b[3J");
```

```
    Console.Clear();

    Console.WriteLine(gameManager);
    Console.SetCursorPosition(0, 16); // Move cursor below the menu
(line 15)
    Console.WriteLine("The copy has been removed from the
game".PadRight(Console.WindowWidth));
    Console.SetCursorPosition(0, 16); // Move cursor below the menu
(line 15)

}
```

[Appendix 10 BrugerScenarier og menuer]

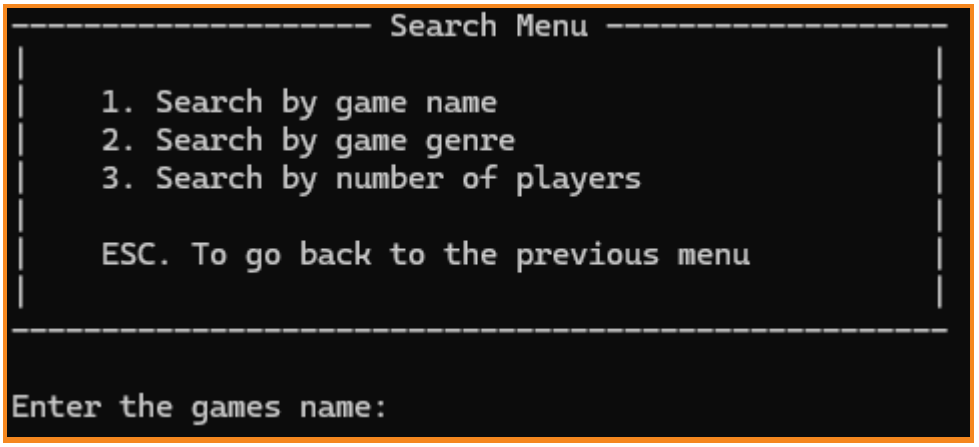
Brugerscenarier / testeksempler

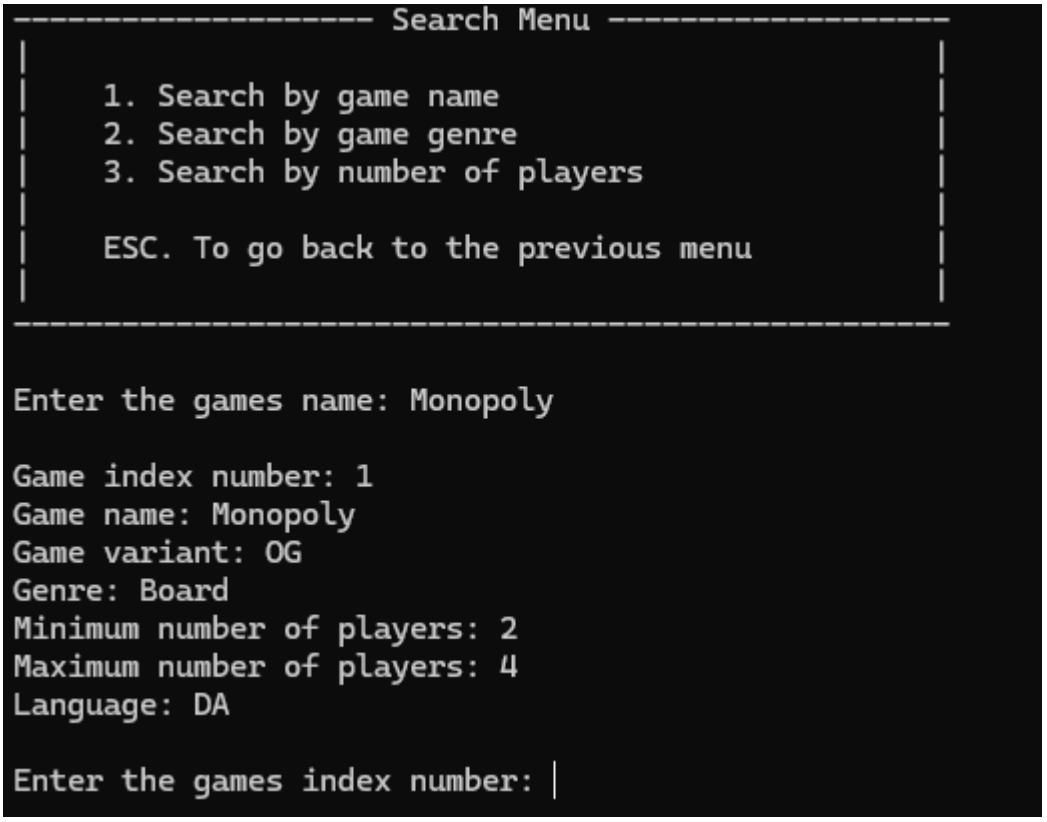
```
-------------------- Game Manager --------------------
|                                                     |
|    1. Add a new game to the stock                   |
|    2. Add a new copy to the stock                   |
|    3. Edit a games details                          |
|    4. Remove a copy from the stock                  |
|    5. Add a game to waiting list                    |
|    6. Make a request on a game                      |
|    7. Prints all copies of a game                   |
|    8. Remove all instances of a game                |
|                                                     |
|    ESC. To go back to the previous menu             |
|                                                     |
-------------------------------------------------------
```

Trykker på 2

```
-------------------- Search Menu --------------------
|                                                     |
|    1. Search by game name                           |
|    2. Search by game genre                          |
|    3. Search by number of players                   |
|                                                     |
|    ESC. To go back to the previous menu             |
|                                                     |
-------------------------------------------------------
```

Trykker på 1


Search Menu
1. Search by game name
2. Search by game genre
3. Search by number of players
ESC. To go back to the previous menu
Enter the games name:


Search Menu
1. Search by game name
2. Search by game genre
3. Search by number of players
ESC. To go back to the previous menu
Enter the games name: Monopoly
Game index number: 1
Game name: Monopoly
Game variant: OG
Genre: Board
Minimum number of players: 2
Maximum number of players: 4
Language: DA
Enter the games index number:

```
Enter the games name: Monopoly

Game index number: 1
Game name: Monopoly
Game variant: OG
Genre: Board
Minimum number of players: 2
Maximum number of players: 4
Language: DA

Enter the games index number: 1

What's the condition of the copy?
1. "A" for new
2. "B" for slightly used
3. "C" for used
4. "D" for a bit worn
5. "E" for very worn
6. "F" for spareparts
6

Enter a price for the game: 0
```

```
-------------------- Game Manager --------------------
|                                                     |
|    1. Add a new game to the stock                   |
|    2. Add a new copy to the stock                   |
|    3. Edit a games details                          |
|    4. Remove a copy from the stock                  |
|    5. Add a game to waiting list                    |
|    6. Make a request on a game                      |
|    7. Prints all copies of a game                   |
|    8. Remove all instances of a game                |
|                                                     |
|    ESC. To go back to the previous menu             |
|                                                     |
-------------------------------------------------------

The copy has been added to the game
```

[Appendix 11 Eksempel på games.txt med tilhørende copies tilknyttet det enkelte objekt.]

```
Monopoly,OG,Board,2,4,DA
2
A,50
A,250
Matador,JR,Board,1,4,DA
0

```

[Appendix 12 Unit tests.]