

# EFTERÅR 2020

## FORPROJEKT TIL DIPLOMINGENIØRPROJEKT

<b>Projekttitel:</b>	Plug-n-Play spil platform		
<b>Projektnr.:</b>	<i>Udbyder</i>	<i>Vejleder</i>	<i>Bivejleder</i>
2021F12		TG	
Deltager 1	Studienr.	Navn	Underskrift
	201610974	Tri Nguyen	-----
Deltager 2	Studienr.	Navn	Underskrift
	20106235	Niclas Spas	-----

# Indhold

<b>1</b>	<b>Opgavebeskrivelse</b>	<b>3</b>
<b>2</b>	<b>Kravspecifikation</b>	<b>6</b>
2.1	Aktørkontekst . . . . .	6
2.1.1	Aktørbeskrivelse . . . . .	7
2.2	User stories . . . . .	7
2.2.1	Gæst spiller spil . . . . .	7
2.2.2	Gæst bruger chatten . . . . .	8
2.2.3	Gæst registrerer sig . . . . .	8
2.2.4	Gæst logger ind . . . . .	8
2.2.5	Bruger spiller spil . . . . .	8
2.2.6	Bruger bruger chatten . . . . .	8
2.2.7	Bruger bruger vennelisten . . . . .	8
2.2.8	Bruger ser på egen brugerprofil . . . . .	8
2.2.9	Bruger ser på andres brugerprofil . . . . .	9
2.3	MoSCoW analyse . . . . .	9
2.4	FURPS analyse . . . . .	10
2.5	Skitser . . . . .	11
<b>3</b>	<b>Analyse</b>	<b>15</b>
3.1	Teknologier . . . . .	15
3.1.1	Frontend teknologier . . . . .	15
3.1.2	Backend teknologier . . . . .	15
3.1.3	Database teknologier . . . . .	16
3.1.4	Udviklingsværktøjer . . . . .	16
3.1.5	Udviklingsstruktur . . . . .	16
3.2	Lignende Projekter . . . . .	16

<b>4</b>	<b>Arkitektur</b>	<b>18</b>
4.1	Logical View . . . . .	19
4.2	Process View . . . . .	21
4.3	Deployment View . . . . .	22
4.4	Data View . . . . .	23
4.5	Security View . . . . .	24
4.6	Implementation View . . . . .	25
4.7	Development View . . . . .	26
4.8	Scenarios/Use Case . . . . .	27
<b>5</b>	<b>Tidsplan</b>	<b>28</b>
<b>6</b>	<b>Samarbejdskontrakt</b>	<b>29</b>
6.1	Arbejdsforhold . . . . .	29
6.1.1	Arbejdstid . . . . .	29
6.1.2	Arbejdssted . . . . .	29
6.1.3	Arbejdsform . . . . .	29

# Kapitel 1

## Opgavebeskrivelse

## Specifications of Bachelor project for students at [Aarhus University](#) / [Aarhus School of Engineering](#)

<b>Date</b>	22-10-2020		
<b>Project title</b>	Plug-n-Play spil platform		
<b>The project applies to at least two students with a specialisation in (mark with X)</b>			
<a href="#">Electronic Engineering</a>		<a href="#">Software Technology</a>	X
<a href="#">Electrical Power Engineering</a>		<a href="#">Health Care Engineering</a>	
<b>Has project been pre-qualified by specific ASE staff?</b> -Indicate name of staff	PER		
<b>Special demands to:</b> - equipment - place - confidentiality			
<b>Project provider</b> Company, ASE staff or Students	Company	Name	Telephone
	Title	Email	Mobile phone
<b>ASE supervisor</b> To be approved later by ASE	Company	Name	Telephone
	Title	Email	Mobile phone

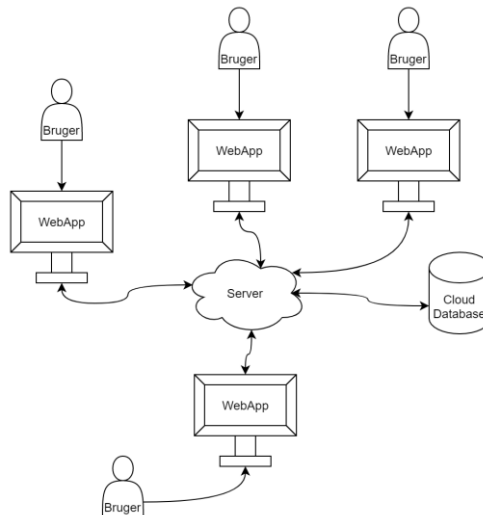
Question can be directed to coordinator of bachelor project:

[Electronic Engineering](#)  
[Electrical Power Engineering](#)  
[Software Technology](#)  
[Health Care Engineering](#)

Lene Häuser Petersen  
 ASSOCIATE PROFESSOR  
 E-mail: [lh@ase.au.dk](mailto:lh@ase.au.dk)  
 Office: [byg. 5125](#), [Edison-322](#)  
 Phone.: 4189 3266

## Project description

Dette projekt ønsker en udvikling af en web-platform hvor brugere til platformen ville kunne spil og interagerer med hinanden. Spillet i sig selv er ikke fokus for projektet, da hvilket som helst spil ville kunne implementeres til platformen. Der vil være fokus på den overordnede platform, dets funktionalitet og brugernes mulighed for interaktion med hinanden gennem funktionaliteter såsom chat.



Som udgangspunkt antages der disse teknologier til udviklingen af platformen

- .Net Core backend
- Angular eller React frontend
- Cloud database
- Websockets til diverse kommunikation
- JWT authentication

De ønskede grund funktionaliteter

- Login
- Brugerprofiler
- Chat
  - General chatrum
  - Privat chatrum
  - In-game chatrum
- Venneliste
- Spilstatestik
- Sende spilanmodninger gennem chat/venneliste
- Matchmaking
  - Tilfældig
  - Struktureret
- Spil

Udover grund funktionaliteterne af platformen kunne der tilføjes potentielle ekstra funktionaliteter

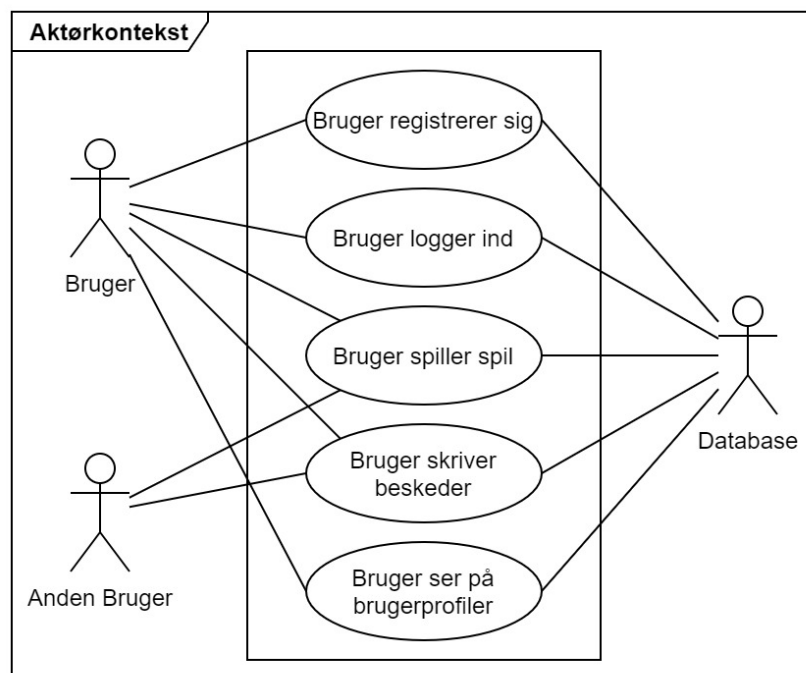
- Understøtte spil med mere end 2 spillere (mere end kun 1 mod 1 spil)
- Understøtte variabel antal spillere (eksempel spille ludo med 3 spillere frem for 4 som det er beregnet til)

# Kapitel 2

## Kravspekifikation

### 2.1 Aktørkontekst

I aktør-kontekstdiagrammet nedenfor ses der hvordan brugeren interagerer med systemets hovedsagelige user stories, samt hvilke andre aktører der er medvirkende.



Figur 2.1: Aktørkontekstdiagram af Plug-n-Play spil platform

### 2.1.1 Aktørbeskrivelse

<b>Aktørnavn:</b>	<b>Bruger</b>
<b>Alternativ reference:</b>	Spiller
<b>Type:</b>	Primær
<b>Beskrivelse:</b>	Brugeren af systemet, som interagerer med systemet gennem spil eller anden socialmediefunktionalitet.

Tabel 2.1: Beskrivelse af aktøren "Bruger".

<b>Aktørnavn:</b>	<b>Anden Bruger</b>
<b>Alternativ reference:</b>	Modstander
<b>Type:</b>	Sekundær
<b>Beskrivelse:</b>	En anden bruger af systemet, som interagerer med Bruger gennem spil eller anden socialmediefunktionalitet.

Tabel 2.2: Beskrivelse af aktøren "Anden Bruger".

<b>Aktørnavn:</b>	<b>Database</b>
<b>Alternativ reference:</b>	
<b>Type:</b>	Sekundær
<b>Beskrivelse:</b>	Databasen indeholder alle brugere, som er oprettet i systemet; oplysninger om afsluttede spil for hver bruger; og beskeder fra chatrum.

Tabel 2.3: Beskrivelse af aktøren "Database".

## 2.2 User stories

Der anvendes user stories til at beskrive kerne funktionalitet i projektet. Disse user stories omhandler to typer brugere. Gæst beskriver en bruger der enten ikke har en registreret bruger eller en bruger som endnu ikke er logget ind. Bruger beskriver en bruger som er registreret og logget ind i systemet.

### 2.2.1 Gæst spiller spil

Som gæst kan jeg starte et spil mod en tilfældig modstander.



### **2.2.2 Gæst bruger chatten**

Som gæst kan jeg læse den globale chat men ikke selv sende beskeder. Dog kan jeg både læse og skrive i chatten med min modstander under et spil.

### **2.2.3 Gæst registrerer sig**

Som gæst kan jeg oprette en bruger på siden, og få adgang til funktioner som påkræver man er logget på.

### **2.2.4 Gæst logger ind**

Som Gæst kan jeg efter at have registreret mig logge ind på siden, og få adgang til ekstra funktionalitet.

### **2.2.5 Bruger spiller spil**

Som bruger kan jeg, udover at starte et spil mod en tilfældig modstander, udfordre andre brugere til et spil gennem chatten ved at klikke på deres brugernavn eller gennem min venneliste.

### **2.2.6 Bruger bruger chatten**

Som bruger kan jeg deltage i den globale chat samt starte private samtaler med andre brugere.

### **2.2.7 Bruger bruger vennelisten**

Sam bruger har jeg adgang til min egen vennelist hvor jeg kan tilføje andre brugere og se deres online status.

### **2.2.8 Bruger ser på egen brugerprofil**

Som bruger har jeg adgang til min egen profilside, hvor jeg kan tilføje et profilbillede samt se relevante spil statistikker.

### 2.2.9 Bruger ser på andres brugerprofil

Som bruger kan jeg tilgå andre brugeres profiler gennem chatten ved at klikke på deres brugernavn eller gennem min venneliste.

## 2.3 MoSCoW analyse

MoSCoW analysen [4] er en metode til at hjælpe med rangering og prioritering af funktionelle krav. Dette sker ved at inddele kravene i fire kategorier Must, Should, Could og Wont. Must beskriver de krav som er absolut essentielle for produktet. Should er de krav som er vigtige for produktet men ikke knap så kritiske som de krav man finder i Must kategorien. Could indeholder de krav som ville være rare at kunne nå de er ikke essentielle for produktets funktionalitet. Sidst er der Wont som er en slags afgrænsning ting man måske kunne forvente af produktet men som ikke vil være med i den pågældende iteration.

#### 1. Must

- Det skal være muligt at tilgå Plug-n-Play via en URL.
- Det skal være muligt at registrere en konto.
- Det skal være muligt at logge ind på en konto.
- Det skal være muligt at iagttage en brugerprofil.
- Det skal være muligt at blive matched tilfældigt med en anden bruger.

#### 2. Should

- Der bør være kommunikation mellem alle brugere gennem et fælles chatrum.
- Der bør være en venneliste tilknyttet en konto.
- Der bør være mulighed for at blive matched struktureret med en anden bruger.
- Der bør være kommunikation mellem brugere der er i samme spil gennem et in-game chatrum.

#### 3. Could

- Det kunne være muligt at se spilstatestik fra en bruger og/eller et spil.
- Det kunne være muligt at bede om at få en e-mail om at nulstille ens password.

- Det kunne være muligt at kommunikerer mellem brugere gennem chatrum privat.
- Det kunne være muligt at invitere/udfordre brugere på ens venneliste til at spille.
- Det kunne være muligt at understøtte spil med mere end 2 spillere.

#### 4. Won't

- Der vil ikke været understøttet variable mængde spillere til spil.

## 2.4 FURPS analyse

FURPS er en analyse metode som bruges til at inddele krav i kategorier og give en prioritering af de disse kategorier. Prioriteringen er et tal mellem 1 og 5 hvor 1 er lav prioritet og 5 er høj prioritet. De fem kategorier i FURPS er Functionality, Usability, Reliability, Performance, Supportability for uddybning se [3]. FURPS benyttes her hovedsaglig til at kategorisere de ikke funktionelle krav. De funktionelle er allerede gennemgået i MoSCoW analysen, så kun de væsentlige overpunkter nævnes her.

#### 1. Functionality

- 5 - Login-system
- 4 - Brugerprofiler
- 3 - Chatrum
- 3 - Venneliste
- 2 - Matchmaking
- 2 - Spil-integration
- 1 - Spilstatestik

#### 2. Usability

- 4 - Platformen skal kunne bruges af skærme med forskellige størrelser, dvs. responsivt design.
- 2 - Under normale serverbetingelser skal platformens startside kunne åbnes og hentes på maks 10s med en 5/5 Mbit hastighed eller højere.

### 3. Reliability

- **5** - Følsom data bliver lagret sikkert i databasen. (Herved koder mm.)
- **3** - Under normale serverbetingelser vil det pågældende chatrum opdateres inden for 10 sekunder.

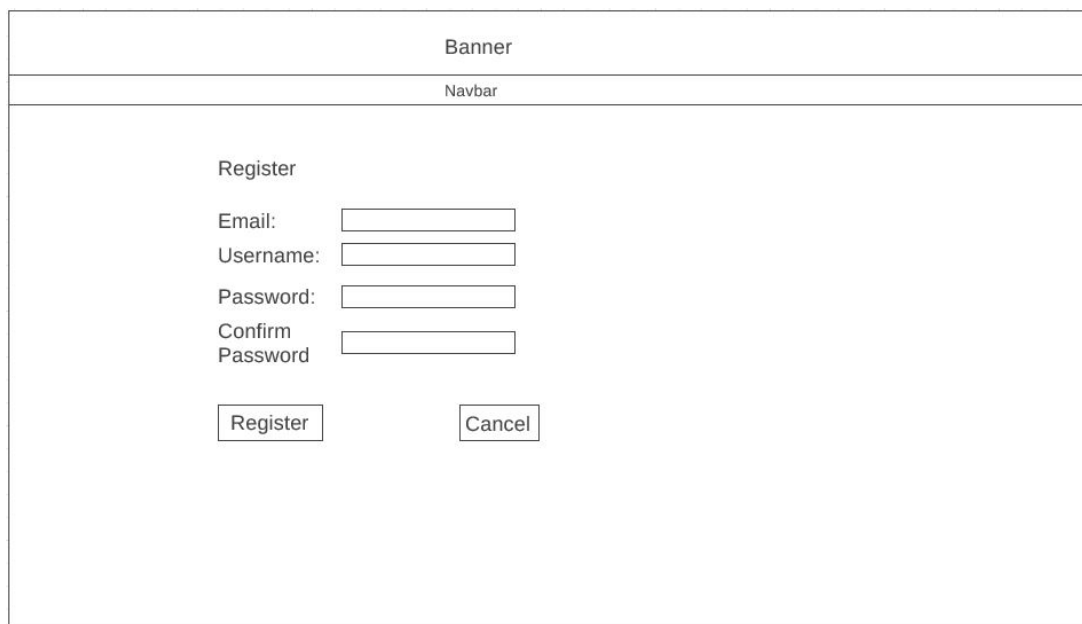
### 4. Performance

- **4** - Databasestrukturen bliver designet til at være skalerbar med mange brugere.

### 5. Supportability

- **4** - Backend-koden bliver designet efter SOLID design-principperne for at gøre den fleksibel men også robust.
- **4** - Vilkarlige spil inden for et framework fungerer på platformen.

## 2.5 Skitser



The wireframe shows a web page layout for user registration. At the top is a 'Banner' section, followed by a 'Navbar'. The main content area is titled 'Register' and contains four input fields: 'Email:', 'Username:', 'Password:', and 'Confirm Password'. Below these fields are two buttons: 'Register' and 'Cancel'.

Figur 2.2: Skitse af bruger registrering

Banner	
Navbar	
<p>Login</p> <p>Email: <input type="text"/></p> <p>Password: <input type="password"/></p> <p><input type="button" value="Login"/> <input type="button" value="Register"/></p>	

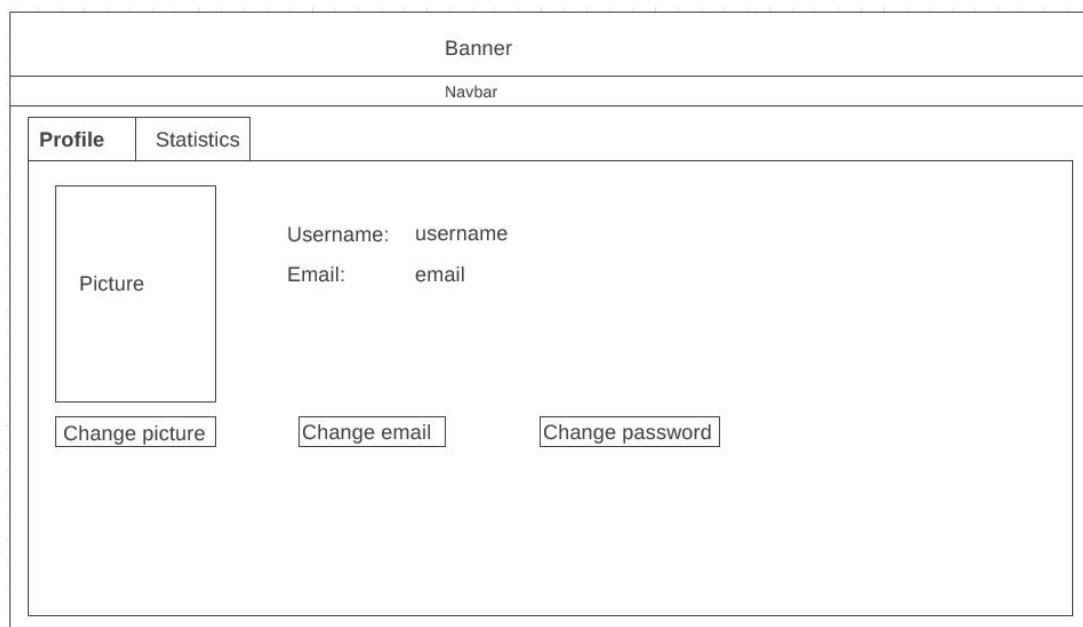
Figur 2.3: Skitse af bruger login

Banner	
Navbar	
<p><b>Menu</b></p> <p><input type="button" value="New Game"/></p> <p><input type="button" value="Profile"/></p>	<p>Chat</p> <p><input type="text"/> <input type="button" value="Send"/></p>

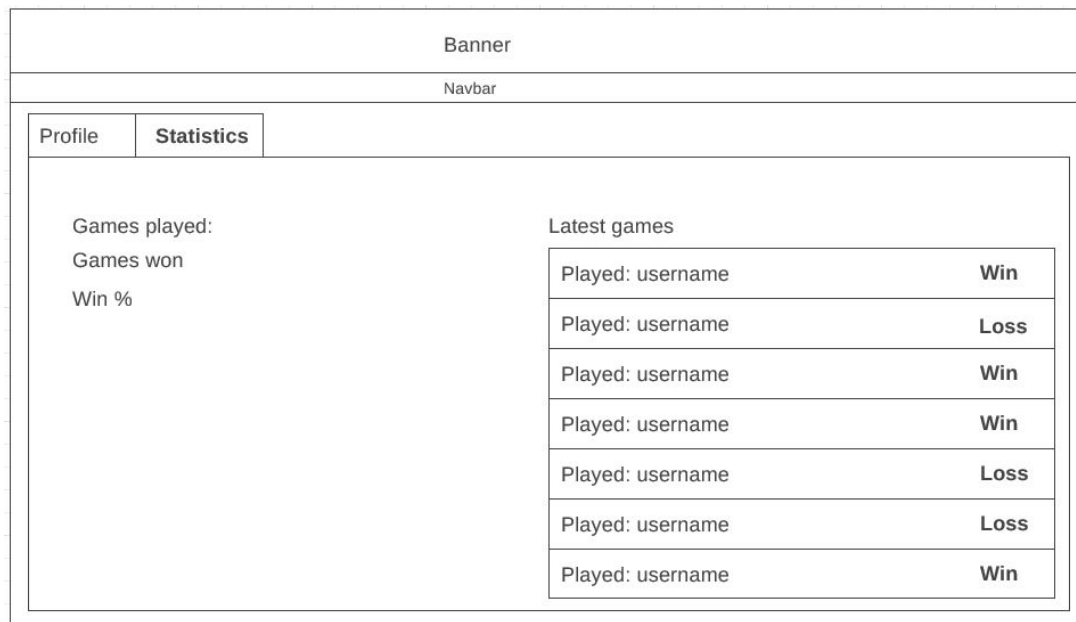
Figur 2.4: Skitse af menu efter brugeren er logget ind



Figur 2.5: Skitse af layout under spil



Figur 2.6: Skitse af Profile tabben på profilsiden



Figur 2.7: Skitse af Statistics tabben på profilsiden

# Kapitel 3

## Analyse

Der forventes ikke udvikling af nye teknologier eller udførsel af nogle eksperimenter til dette projekt. Da der ikke bruges tid til at udvikle ny teknologi gøres der tanker til teknologi der kan bruges for at kunne implementere Plug-n-Play spil platformen.

### 3.1 Teknologier

#### 3.1.1 Frontend teknologier

#### 3.1.2 Backend teknologier

Når der skal tænkes på forskellige mulige backend-teknologier er der forskellige at vælge mellem, hver med deres fordele og ulemper, som kan være mere eller mindre relevante for det overordnede projekt. Når der skal vælges teknologier for backend skal der vurderes på kode-sproget der bliver brugt, og hvilken frihed/restriktion det sprog har til brugen i backend.

#### **C# - ASP.Net Core**

.Net Core er et fleksibelt og stærkt backend framework. Styrkerne for brug af dette framework ligger i dets optimering af kode gennem kompilering, hvor der ikke selv behøver at tænke så meget til hvordan ens kodestruktur er sat for at være optimeret. Dertil er det også et af de frameworks vi er blevet undervist i at bruge på studiet, så dertil vil der være grunderfaring med brug af dette framework. Gruppen har også tidligere haft erfaring i at udvikle backend i .Net Core, så denne erfaring er også relevant til det fremtidige valg af teknologi.



## JavaScript - Node.js/Express

Node.js er et runtime environment, der gør brug af styrkerne fra JavaScript i dets backend-udvikling. Fordelen ved brug af Node.js er at frontend og backend vil have muligheden for at blive skrevet i samme sprog, så der er færre variabler at overveje når der skal ses mellem de to blokke. Med et JavaScript baseret framework er det derfor også meget let at finde packages/moduler i open-source biblioteker, hvilket gør det let at finde rundt i hvilke packages man måske gerne vil gøre brug af. Node.js agerer som en server som eksekverer JavaScript, og det kan være svært at skalere hen ad vejen på en cloud server. Der er gjort brug af Node.js og Express på studiet til samarbejde med frontend udvikling på studiet, dog på en mindre grad i forhold til .Net Core.

## Firestore

Firestore er en mobil- og webudviklingsplatform, platformen har mange forskellige tjenester som backend i en applikation med færdigudviklet materiale til at opbevare data, autentikation, og gemme filer som brugere af applikationen så kan få adgang til igen senere. Firestore har en dokumentbaseret cloud-storage database, som var en mulighed til database teknologi.

### 3.1.3 Database teknologier

### 3.1.4 Udviklingsværktøjer

Frontend udvikling

Backend udvikling

INDSÆT NOGET BASERET PÅ DE VALG DER ER TAGET MED TEKNOLOGIER

### 3.1.5 Udviklingsstruktur

INDSÆT NOGET BASERET PÅ DE VALG DER ER TAGET MED TEKNOLOGIER

## 3.2 Lignende Projekter

Der er i 2020 udviklet en online skakplatform ved navn ChessBot[2], hvor mange basale web-udviklingsovervejelser også kan ses i dette projekt. Dette projekt abstraherer dog selve

spillet væk, og fokusere mere på de bruger-interaktive funktionaliteter, som ChessBot ikke gør brug af.

# Kapitel 4

## Arkitektur

Arkitekturen for projektet vil være delt op i N+1 View Model [1]. Arkitekturn vil beskrive projektets planlægning, struktur, og intern kommunikation i forskellige abstraktionsniveauer med fokus på forskellige områder i projektet.

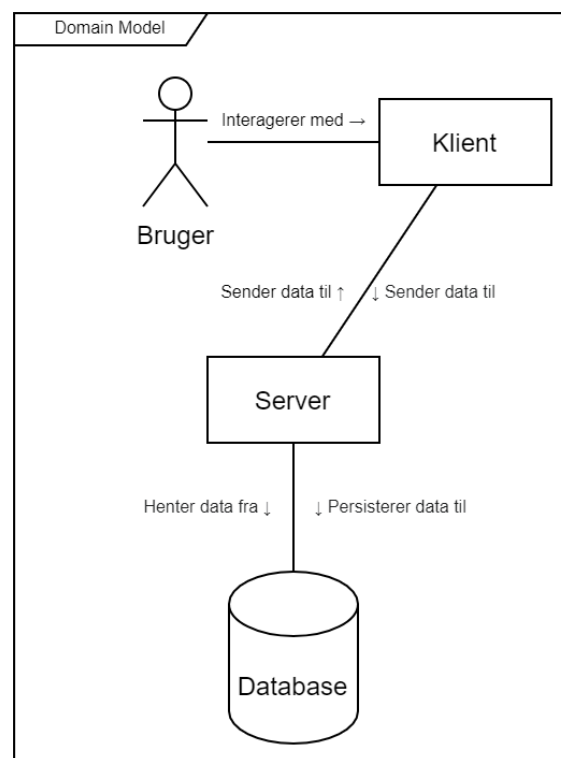
Overblik over de (mulige) lag af abstraktioner der vil gøres brug af er:

- Logical View, som omfatter sekvensdiagrammer og domænemodeller. Der vil i hvert logival view laves en applikationsmodel hvor hver Use Case bliver beskrevet. diagrammerne vil være delt op i client-side og server-side kode, for at adskille de to moduler og kun at fokusere på det enkelte modul.
- Process View, som omfatter processer der kører samtidigt. Her kan sekvensdiagrammer også blive brugt til at vise flow af processerne., samt vejledende tekst der beskriver hvilke processer der kører asynkront samt deres overvejelser.
- Deployment View, som omfatter datakommunikationen internt for systemet. Der vil gennemgås en beskrivelse af low level (kommunikation mellem database, sever, og klient) og high level (systemets meddelelser som sendes rundt).
- Data View, som omfatter strukturen for den persisterede data i systemets database. Hvordan det er gemt, hvilke overvejelser der er til databasestrukturen, og hvilke restriktioner der arbejdes under.
- Security View, som omfatter de sikkerhedsforanstaltninger der er gjort for datakommunikation og/eller datapersistering.
- Implementation View, som omfatter implementeringsteknologierne som er gjort brug af til projektets udvikling. Der vil dannes et overblik over systemet med klasser,

metoder, og andre beskrivelser der ikke tidligere er beskrevet. Her kan det være klasser, metoder, eller andre komponenter som ikke var repræsenterede gennem Use Cases.

- Development View, som omfatter udviklingsstrukturen og opsætningen af projektet. Her vil der ligge ordforklaringer bag kodestrukturen, dets inddelinger, brug af biblioteker, og andre opsætninger som kan skille sig ud fra normal brug af udviklingsværktøjer.
- Scenarios/Use Case, som omfatter kort gennemgang af de relevante og mest signifikante use cases, som beskrives overordnet for hele systemet og dets interaktion/kommunikation.

## 4.1 Logical View



Figur 4.1: Domænemodel over systemets helhed.

Fig. 4.1 beskriver kort systemets overordnede flow. En bruger interagerer med en klient, som så har dataoverføring ind til en server, so så behandler data og gemmer det i en

database. Der kan endvidere tænkes til, at der kan være flere klienter, hver med deres bruger, forbundet til den samme server.

## 4.2 Process View

Mulige processer vi kommer til at inddrage er når folk kan stille sig i kø til et spil, og når folk vil sende beskeder til hinanden. Da de spil der måske implementeres er tur-baserede behøves der ikke at laves asynkront behandling af spil-data, da det alligevel sker sekventielt.

## 4.3 Deployment View

Klienten sender beskeder til serveren, serveren agere her som et REST API, der har et interface der modtager og sender det klienten har brug for. Den cloud-baserede database håndteres separat fra serveren, hvor den kaldes til med serverens framework implementationer.

## 4.4 Data View

Der er ikke sat noget bestemt op endnu, men som det ser ud kan det godt være en SQL eller NoSQL server der gøres brug af.



## 4.5 Security View

Der tænkes at se på HTTPS løsninger til kommunikation mellem klient og server, samt brug af bearer tokens til autentificering og adgang til specielle data.

## 4.6 Implementation View

Ud fra de konklusioner der vil komme fra analyseafsnittet, vil der her komme nogle beskrivelser af de valgte udviklingsværktøjer der er gjort brug af.

## 4.7 Development View

Ud fra de konklusioner der vil komme fra analyseafsnittet, vil der her komme nogle beskrivelser af de valgte udviklingsværktøjers kodestruktur.

## 4.8 Scenarios/Use Case

Ud fra de use cases der bliver beskrevet i applikationsmodeller i Logical View vil der her blive beskrevet de sidste "ubrugte" use cases

# Kapitel 5

## Tidsplan

Projektet er opdelt i fire dele. Analyse og arkitektur udvikling denne fase fokusere på at afklare hvilke teknologier der anvendes i projektet samt udvikling af arkitekturen for projektet. Udviklingsfasen hvor projektet implementeres. Testing og debugging her udføres accepttest og potentielle bugs fundet under accepttest udbedres. Sidste fase er rapport-skrivning.

Projektet løber over 16 uger med følgende fordeling.

- Uge 1-3 Analyse og arkitektur udvikling.
- uge 4-10 Udvikling - Med feature stop efter uge 9.
- Uge 11-12 Test og debugging.
- Uge 13-16 Rapportskrivning.

# Kapitel 6

## Samarbejdskontrakt

### 6.1 Arbejdsforhold

#### 6.1.1 Arbejdstid

Den ugentlige arbejds tid forventes at ligge omkring 32 til 35 timer.

#### 6.1.2 Arbejdssted

Gruppen har søgt om arbejdsplads på skolen såfremt dette bliver tildelt er det forventet at medlemmerne møder op her 3 dage om ugen. Disse dage lægges fast inden semesterstart når skemaerne for de andre fag er fastlagt.

#### 6.1.3 Arbejdsform

Til afvikling af projektet anvendes forskellige redskaber fra agil software udvikling. Der anvendes ikke fuldt ud SCRUM pga. gruppens størrelse. Arbejdet struktureres i sprint af 1-2 ugers varighed. Derudover anvendes et kanban bræt til at holde styr på de relevante arbejdsopgaver som tilhører det givende sprint.

# Litteratur

- [1] P. Kruchten. Architectural blueprints—the “4+1” view model of software architecture, 1995. URL <https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf>.
- [2] V. P. W. Tri Nguyen, Niclas Spas. Chessbot - projektrapport, 2020.
- [3] Wikipedia. Furps, 2019. URL <https://en.wikipedia.org/wiki/FURPS>.
- [4] Wikipedia. Moscow method, 2020. URL [https://en.wikipedia.org/wiki/MoSCoW\\_method](https://en.wikipedia.org/wiki/MoSCoW_method).