Dungeons and Gnoblins

Semesterprojekt

Aarhus Institut for Elektro- og Computerteknologi

Authors:

Magnus Blaabjerg Møller, 202006492
Sune Andreas Dyrbye, 201205948
Morten Høgsberg, 201704542
Rasmus Engelund, 202007668
Anders Hundahl, 202007859
Oscar Dennis, 202009975
Jacob Hoberg, 201807602
Luyen Vu, 202007393

Vejleder: Jung Min Kim (Jenny)

Anslag:

Afleveringsdato: 03-06-2022 Eksamineringsdato: 22-06-2022

Contents

1 Introduction

2 Systembeskrivelse

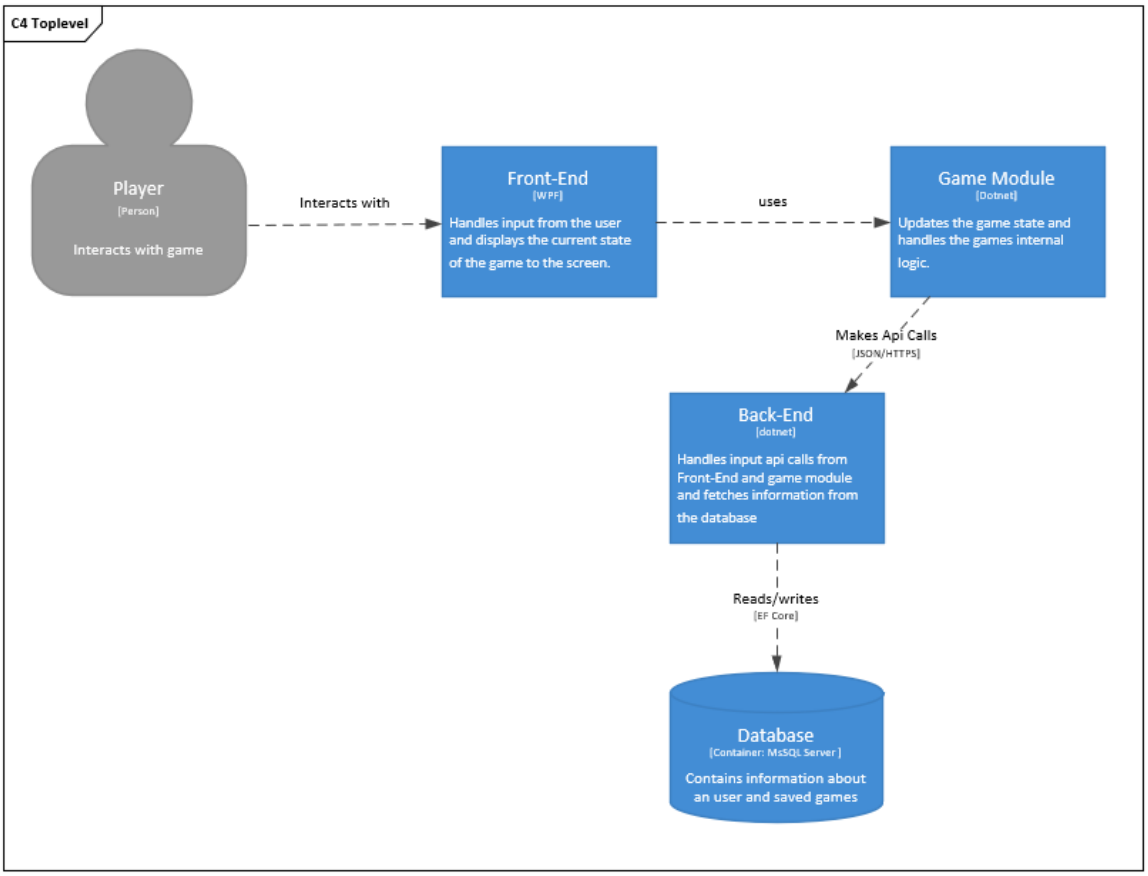

Figure 1: C4 top level diagram, som viser kommunikation mellem systemets segmenter

På figuren herover ses systemets toplevel arkitektur. Denne består af en bruger som interagerer med systemet gennem frontendapplikationen som skrives i WPF. States i denne applikation styres af Game modul som holder styr på hvor spilleren befinder sig, hvilke items der er samlet op og andre nyttige information som skal bruges gennem spillet. Spillets backend benyttes primært til bruger authentication og som bindeled til databasen. Databasen indeholder oplysninger om blandt andet brugere, de gemte spil og oplysninger om historien for de forskellige rum.

3 Kravspecifikation

I det følgende afsnit vil vi uddybe de forudbestemt krav til projektets specifikation. Dette indeholder en systembeskrivelse, samt en accepttestspecifikation til de opsatte krav. Kravene til systemet vil være opdelt i funktionelle og ikke funktionelle, hvor de opsatte ikke-funktionelle krav vil være prioriteret efter MoSCoW princippet. Der er ydermere opsat accepttest som tester alle scenarier i de opsatte user stories, samt systemets ikke-funktionelle krav.

3.1 Funktionelle krav - User Stories

I dette afsnit forklares systemets funktionelle krav i form af user stories som tager os igennem systemet og de ønskede funktioner.

User Story 1: Log in
Som Bruger
Kan jeg logge ind
For at kunne se en Main Menu så jeg kan spille spillet.

User Story 2: Opret profil
Som Bruger
Kan jeg oprette en ny profil
For at jeg kan logge ind på spillet.

User Story 3: Main Menu - Settings
Som bruger
Kan jeg tilgå Settings
For at jeg kan customize interfacet

User Story 4: Main Menu – Start Spil
Som bruger
Kan jeg starte et nyt spil
Ved at give spillet et navn som bruges til save games

User Story 5: Exit game
Som Bruger
Kan jeg trykke Esc
For at se en menu med mulighederne "Return to game", "Save and Exit", "Exit without saving" and "Settings".

User Story 6: Exit Menu - Resume Game
Som Bruger
Kan jeg trykke på "Resume game"
For at vende tilbage til spillet

User Story 7: Exit Menu - Save and Exit
Som Bruger
Kan jeg trykke på "Save and Exit"
For at spillet gemmes og bruger bliver vist Main Menu

User Story 8: Exit Menu - Exit Without Saving
Som Bruger
Kan jeg trykke på "Exit Without Saving"
For at få en "Are you Sure" alert

User Story 9: Exit Menu - Exit Without Saving - No
Som Bruger
Kan jeg trykke på "No"
For at vende tilbage til exit menu

User Story 10: Exit Menu - Exit Without Saving - yes
Som Bruger
Kan jeg trykke på "Yes"
For at vende tilbage til Hovedmenuen uden at gemme

UserStory 11 : Exit Menu - Settings
Som Bruger
Kan jeg trykke på "Settings"
For at se Settings Menuen.

UserStory 12 : Settings Menu - Efter ændringer
Som bruger
Kan jeg trykke "Apply", "back", "Cancel" og "Default"
For at tilpasse spillets indstillinger

UserStory 13 : Settings Menu - Efter ændringer - Apply
Som Bruger
Kan jeg trykke "Apply"
For at ændre settings

UserStory 14 : Settings Menu - Efter ændringer - back
Som Bruger
Kan jeg trykke "back"
For at brugeren sendes tilbage til Esc Menu

UserStory 15 : Settings Menu - Efter ændringer - Cancel
Som Bruger
Kan jeg trykke på "Cancel"
For at ændringerne bliver reverted

UserStory 16 : Settings Menu - Efter ændringer - Default
Som bruger
Kan jeg trykke på "Default"
For at sætte settings til defaults

UserStory 17 : Main Menu - Load Game
Som bruger
Kan jeg trykke på "load game"
For at få en liste a save games.

UserStory 18 : Main Menu - Load Game - Load
Som bruger
Kan jeg trykke på "load game"
For at komme ind i spillet og fortsætte med at spille det valgte spil

UserStory 19 : Main Menu - Load Game - Delete Game
Som bruger
Kan jeg trykke "delete game"
For at game statet bliver slettet fra cloud.

UserStory 20 : Main Menu - Load Game - Back

Som bruger
Kan jeg trykke "back"
For at vende tilbage til Main Menu

UserStory 21 : Spil Spillet - Bevægelse i spil
Som bruger
Kan jeg vælge mellem actioner ved at bruge tallene 0-9 eller piltasterne til at bevæge mig
For at bevæge karakteren i spillet til et andet rum.

UserStory 22 : Spil spillet - Enter nyt room
Som bruger
Får jeg en beskrivelse af et rum, når jeg kommer ind i rummet
Så jeg ved hvad jeg kan inteagere med.

User Story 23: Spil spillet - Inventory
Som bruger
Kan jeg trykke på "Inventory" knappen
For at se genstande som jeg besidder

UserStory 24: Spil spillet - Combat
Som bruger
Kan jeg trykke på "attack" knappen
For at skade fjenden

User Story 25: Spil spillet - Combat - flygt
Som bruger
Kan jeg trykke på "flee" knappen
For at blive rykket tilbage til forrige rum

UserStory 26: Spil spillet - Inventory
Som bruger
Kan jeg trykke på "Inventory" knappen
For at tilgå karakterens udrustning

UserStory 27: Spil spillet - Interact
Som bruger
Kan jeg trykke på "interact" knappen
For at interagere med objekter i det nuværende rum

UserStory 28: Spil Spillet - Level klaret
Som bruger
Kan jeg færdiggøre det sidste rum
For at få en besked om at spillet er fuldført og mulighed for at tilgå Main Menu

3.2 Ikke-funktionelle krav

I dette afsnit opsættes systemtets ikke-funktionelle krav. Disse er opdelt efter kategori, hvorved der er krav til FURPS modellens forskellige dele.
Ikke-funktionelle:
GUI:

- GUI'en skal tilbyde valget mellem 3 resolutions
- GUI'en skal kunne være fullscreen
- GUI'en skal kunne være windowed

SOUND:
- Lyden skal kunne justeres mellem 0-100% relativt til PC'ens lyd niveau.

DATABASE:
- Skal kunne gemme maksimalt 5 save games
- Skal kunne loade et spil indenfor maksimalt 5s
- Skal gemme hvilke genstande man bruger lige nu
- Skal gemme hvor meget liv man har tilbage.
- Skal gemme hvilke fjender man har slået ihjel.
- Skal gemme hvilke puzzles man har løst
- Skal gemme hvilke rum man har været i.

GAMEPLAY:
- Spillets kort skal holde styr på hvilke rum man kan komme til for et givet rum.
- Spillets kort skal kun vise de rum som spilleren har været i.
- Spillets kort skal, hvis spilleren har været i alle rum vise alle rum.

- Et rum kan have maksimalt 4 forbindelser til andre rum.
- Et rum skal have mindst 1 forbindelse til andre rum.
- Alle Rum skal kunne nås fra ethvert andet rum, måske ikke direkte, men man skal kunne komme dertil.

- Spillerens rygsæk skal kunne indeholde alle spillets genstande.
- Spilleren skal have mulighed for at bruge ét våben og én rustning af gangen.
- Spilleren skal have mulighed for at skifte hvilket våben og hvilken rustning der bruges.

COMBAT:
- Når spilleren/fjenden prøver at slå, rammer man kun hvis man på sit angreb slår højere end modstanderens rustningsværdi. Dette afgøres af et simuleret 20-sidet terninge kast, hvortil der lægges en værdi til, korresponderende til spilleren/fjendens våben bonusser.
- Hvis spilleren/fjenden rammer, bliver skaden bestemt af et/flere simulerede terningekast, afhængigt af hvilket våben der bruges
- Hvis spilleren, når nul liv inden fjenden, så dør spilleren og spillet er tabt.
- Hvis fjenden, når nul liv inden spilleren, så dør fjenden og spilleren kan nu frit udforske rummet, som fjenden var i.
- Hvis spilleren drikker en livseleksir bliver spillerens nuværende liv sat til fuldt.
- Hvis spilleren/fjenden rammer, bliver skaden bestemt af et/flere simulerede terningekast, afhængigt af hvilket våben der bruges.

PERFORMANCE:
- Spillet skal respondere indenfor maksimalt 5s
- Spillet må ikke have mere end én kommando i aktionskøen af gangen

STABILITY:
- MEANTIME BETWEEN FAILURE - 1Time +- 10Min

DOCUMENTATION:
- Spillet skal have en manual/help page til hvordan alting virker.

SECURITY:

- Username skal være mindst 6 characters
- Username skal være unikt
- Password skal være mindst 8 characters
- Password skal have store og små bogstaver
- Password må ikke indeholde username

3.3 MOSCOW krav

I dette afsnit er de ikke-funktionelle krav fra ovenstående afsnit prioriteret ved hjælp af MoSCow metoden.
MUST
- have en GUI
- have en Database
- Have Netværkskommunikation
- Have en serie af sammenhængende rum.
- Have et start og slut rum som ikke er det samme rum.
- Spillet skal kunne gemmes på en Database.
- Skal kunne hente save game fra databasen
- Skal have en authentication system (Accounts)
- Hvert Rum skal bestå af et beskrivende element og en serie af actioner.
- Spillet skal have instillinger.
- Spillet skal have en Character.
- Spillet skal kunne tage imod user input.
- Spillet skal være udviklet til Windows.
- Spillet skal være testbart i.e. skal være designet med testing in mind.

SHOULD
- Have Enemies
- Have Items
- Have Combat mechanics
- Have End Screen
- Have Character Stats

COULD
- Have Level development
- Have Procedual world
- Have Text Parser
- Have Difficulty
- Have Security

WON'T
- Have Graphics

3.4 Accepttest

I dette afsnit opsættes systemets Accepttest. Disse er opdelt efter kategori, først er der accepttest for User stories, og herefter de ikke funktionelle krav.

3.4.1 Funktionelle

Test af User Story 1 - Login Scenarie - Succesfuld login
Givet at brugerens profil er oprettet på databasen og at serveren er oppe.

- Når bruger indtaster sit login(Brugernavn og password)
- og trykker "Log in" på UI
- Så skifter skærmen til hovedmenu

Resultat:
Kommentar:

Scenarie - Fejlet login
Givet at brugerens profil er oprettet på databasen og at serveren er oppe.

- Når bruger indtaster et forkert login(Brugernavn og password)
- og trykker "Log in" på UI
- Så signaleres der om forkert login-oplysninger til bruger

Resultat:
Kommentar:

Test af User Story 2 - Opret profil Scenarie - Bruger opretter profil
Givet at brugerens profil er oprettet på databasen og at serveren er oppe.

- Når bruger vælger at oprette profil
- Så gemmes datanene for brugerens profil på databasen
- Og skærmen skiftes til log ind skærmen

Resultat:
Kommentar:

Scenarie - Bruger opretter samme profil
Givet at brugerens profil er oprettet på databasen og at serveren er oppe.

- Når bruger vælger at oprette en allerede-eksisterende profil
- Så signaleres der om at profil allerede eksisterer

Resultat:
Kommentar:

Test af User Story 3 og 4 - Main Menu Scenarie - Tilgå settings
Givet at brugeren er logget ind og har adgang til spillet.

- Når bruger trykker settings i UI
- Så går spillet til settings menuen

Resultat:
Kommentar:

Scenarie - Start spillet
Givet at brugeren er logget ind og har adgang til spillet.

- Når bruger trykker "New Game"
- Så vises game-interface for brugeren

Resultat:
Kommentar:

Scenarie - Exit game
Givet at brugeren er logget ind og har adgang til spillet.

- Når bruger trykker "Exit game"
- Så lukker spillet

Resultat:
Kommentar:

Test af User Story 5-16 - Exit Menu Scenarie - Exit spil
Givet at brugeren har trykket "New Game"

- Når bruger trykker "Escape" på keyboardet
- Så popper et menuvindue op med mulighederne "Resume Game", "Save Game", "Main Menu" og "Settings"

Resultat:
Kommentar:

Scenarie - In game menu - Resume game
Givet at brugeren har trykket "New Game" og derefter har trykket "Escape" på keyboard

- Når bruger trykker "Resume Game" i In game menu
- Så forsvinder menuvinduet og spillet fortsætter

Resultat:
Kommentar:

Scenarie - In game menu - Save – No Combat
Givet at brugeren har trykket "New Game" og derefter har trykket "Escape" på keyboard

- Når bruger trykker på "Save Game" i In game menuen
- Så vises save menuen
- Og en liste af gemte spil

Resultat:
Kommentar:

Scenarie - In game menu - Save – Combat
Givet at brugeren er i combat

- Når bruger trykker "Escape" på keyboarded
- Så kan man ikke se en "Save Game" knap i game menuen

Resultat:
Kommentar:

Scenarie - In game menu - Main Menu
Givet at brugeren har trykket "New Game" og derefter har trykket "Escape" på keyboard

- Når bruger trykker "Main Menu" i In game menu
- Så vises hovedmenuen

Resultat:
Kommentar:

Scenarie - In game menu - Settings
Givet at brugeren har trykket "New Game" og derefter har trykket "Escape" på keyboard

- Når bruger trykker "Settings"
- Så åbnes et vindue med mulighed for konfiguration af spil for bruger

Resultat:
Kommentar:

Scenarie - Exit menu - Settings - Apply
Givet at brugeren har trykket "Settings" og ændret på resolution indstilling

- Når bruger trykker "Apply"
- Så ændres indstillinger som brugeren har ændret

Resultat:
Kommentar:

Scenarie - Exit menu - Settings - Back
Givet at brugeren har trykket "Settings"

- Når bruger trykker "Back"
- Så vises In game menuen

Resultat:
Kommentar:

Scenarie - Exit menu - Settings - Default
Givet at brugeren har trykket "Settings" og har ændret mindst 1 indstlling

- Når bruger trykker "Default"
- Så ændres alle indstiliinger tilbage til default settings

Resultat:
Kommentar:

Test af User Story 17 og 18 - Save Menu Scenarie - Save Menu - Save Game
Givet at brugeren har trykket "Save game" fra Settings menu og ikke er i combat.

- Når bruger vælger et gemt spil
- Og sætter navnet til det ønskede
- Og trykker på "Save Game" knappen
- Så genoptages spillet

Resultat:
Kommentar:

Scenarie - Save Menu - Back
Givet at brugeren har trykket "Save game" fra Settings menu

- Når bruger trykker "Back" i save game menuen
- Så vender spillet ilbage til In game menuen

Resultat:
Kommentar:

Test af User Story 19-22 - Main Menu Scenarie - Main menu - Load Game
Givet at brugeren er logget ind og har adgang til spillet.

- Når bruger trykker "Load game"
- Så vises en liste af gemte spil på profilen

Resultat:
Kommentar:

Scenarie - Main menu - Load Game - Load
Givet at brugeren er logget ind og har adgang til spillet og trykket "Load Game"

- Når bruger vælger et gemt spil
- Og trykker "Load Game"
- Så loader spillet det gemte spil med det valgte game state

Resultat:
Kommentar:

Scenarie - Main menu - Load Game - Back
Givet at brugeren er logget ind og har adgang til spillet og trykket "Load Game"

- Når bruger trykker "Back" i load game-menuen
- Så vender spillet tilbage til hovedmenuen

Resultat:
Kommentar:

Test af User Story 23-28 - Spil spillet Scenarie - Spil spillet - Start spillet
Givet at brugeren har trykket "New Game"

- Når bruger anvender piltasterne på keyboard eller trykker på knappen på interfacet til at bevæge sig sydpå fra startrummet
- Så bevæger brugeren sig ind i rummet "Syd" for det rum de står i

Resultat:
Kommentar:

Scenarie - Spil spillet - Enter nyt room
Givet at brugeren har trykket "New Game"

- Når bruger går ind i et nyt rum ved brug af keyboard
- Så giver UI en beskrivelse af rummet spilleren befinder sig i

Resultat:
Kommentar:

Scenarie - Spil spillet - Combat
Givet at brugeren møder en fjende i et rum

- Når bruger trykker "Fight!"
- Så ruller brugeren et tal mod fjenden om at skade fjenden
- Og derefter ruller fjenden et tal om at skade brugeren

Resultat:
Kommentar:

Scenarie - Spil spillet - Combat - Flygt
Givet at brugeren møder en fjende i et rum

- Når bruger trykker "Flee"
- Så flyttes brugeren tilbage til det rum han kom fra
- Og får ikke sit mistede liv tilbage igen

Resultat:
Kommentar:

Scenarie - Spil spillet - Inventory
Givet at brugeren har trykket "New Game"

- Når bruger trykker "Inventory"
- Så vises genstande som bruger besidder

Resultat:
Kommentar:

Scenarie - Spil spillet - Interact
Givet at brugeren har trykket "Start spil" og at der ike er fjender i rummet

- Når bruger trykker "Interact"
- Så kan bruger tage potentielle genstande i rummet til brugerens "Inventory"

Resultat:
Kommentar:

Scenarie - Spil spillet - Level klaret
Givet at brugeren har fuldført det næstsidste rum

- Når bruger bevæger sig ind i sidste rum
- Så får bruger en besked om at spillet er klaret og får mulighed for at gå til "Main Menu"

Resultat:
Kommentar:

3.4.2 Ikke funktionelle

GUI

Table 1: Ikke funktionelle tests for GUI

Beskrivelse	Verificering	Resultat	Kommentar
GUI'en skal tilbyde valget mellem 3 resolutions	Visuel		
GUI'en skal kunne være fullscreen	Visuel		
GUI'en skal kunne være windowed	Visuel		

SOUND

Table 2: Ikke funktionelle tests for SOUND

Beskrivelse	Verificering	Resultat	Kommentar
Lyden skal kunne justeres mellem 0-100% relativt til PC'ens lyd niveau.	Auditorisk/Visuel		

DATABASE

Table 3: Ikke funktionelle tests for DATABASE

Beskrivelse	Verificering	Resultat	Kommentar
Skal kunne gemme maksimalt 5 save games	Visuel		
Skal kunne loade et spil indenfor maksimalt 5s	Visuel		
Skal gemme hvilke genstande man bruger lige nu	Visuel		
Skal gemme hvor meget liv man har tilbage.	Visuel		
Skal gemme hvilke fjender man har slået ihjel.	Visuel		
Skal gemme hvilke puzzles man har løst	Visuel		
Skal gemme hvilke rum man har været i.	Visuel		

GAMEPLAY

Table 4: Ikke funktionelle tests for GAMEPLAY

Beskrivelse	Verificering	Resultat	Kommentar
Spillets kort skal holde styr på hvilke rum man kan komme til for et givet rum.	Visuel		
Spillets kort skal kun vise de rum som spilleren har været i.	Visuel		
Spillets kort skal, hvis spilleren har været i alle rum vise alle rum.	Visuel		
Et rum kan have maksimalt 4 forbindelser til andre rum.	Visuel		
Et rum skal have mindst 1 forbindelse til andre rum.	Visuel		
Alle Rum skal kunne nås fra ethvert andet rum, måske ikke direkte, men man skal kunne komme dertil.	Visuel		
Spillerens rygsæk skal kunne indeholde alle spillets genstande.	Visuel		
Spilleren skal have mulighed for at bruge ét våben og et skjold af gangen.	Visuel		
Spilleren skal have mulighed for at skifte hvilket våben og hvilket skjold der bruges.	Visuel		

COMBAT

Table 5: Ikke funktionelle tests for COMBAT

Beskrivelse	Verificering	Resultat	Kommentar
Når spilleren/fjenden prøver at slå, rammer man kun hvis man på sit angreb slår højere end modstanderens rustningsværdi. Dette afgøres af et simuleret 20 sidet terninge kast, hvortil der lægges en værdi til, korresponderende til spilleren/fjendens våben bonusser.	Visuel		
Hvis spilleren/fjenden rammer, bliver skaden bestemt af et/flere simulerede terningekast, afhængigt af hvilket våben der bruges.	Visuel		
Hvis spilleren/fjenden rammer, bliver skaden bestemt af et flere simulerede terningekast, afhængigt af hvilket våben der bruges.	Visuel		
Hvis spilleren, når nul liv inden fjenden, så dør spilleren og spillet er tabt.	Visuel		
Hvis fjenden, når nul liv inden spilleren, så dør fjenden og spilleren kan nu frit udforske rummet, som fjenden var i.	Visuel		
Hvis spilleren drikker en livseleksir bliver spillerens nuværende liv sat til fuldt	Visuel		

PERFORMANCE

Table 6: Ikke funktionelle tests for PERFORMANCE

Beskrivelse	Verificering	Resultat	Kommentar
Spillet skal respondere indenfor maksimalt 5s	Visuel		
Spillet må ikke have mere end én kommando i aktionskøen af gangen	Visuel		

STABILITY

Table 7: Ikke funktionelle tests for STABILITY

Beskrivelse	Verificering	Resultat	Kommentar
MEANTIME BETWEEN FAILURE 1Time + 10Min?	Visuel		

DOCUMENTATION

Table 8: Ikke funktionelle tests for DOKUMENTATION

Beskrivelse	Verificering	Resultat	Kommentar
Spillet skal have en manual/help page til hvordan alting virker.	Visuel		

SECURITY

Table 9: Ikke funktionelle tests for SECURITY

Beskrivelse	Verificering	Resultat	Kommentar
Username skal være mindst 6 characters	Visuel		
Username skal være unikt	Visuel		
Password skal være mindst 8 characters	Visuel		
Password skal have store og små bogstavers	Visuel		
Password må ikke indeholde username	Visuel		

4 Metode og Proces

I nedstående afsnit fremlægges metoder og processer brugt til udviklingen af Dungeons and Gnoblins spillet.

4.0.1 Gruppedannelse

Gruppen blev dannet på baggrund af at vi som studerende selv skulle finde medlemmer og danne projektgrupper. 4 af gruppens medlemmer var dannet inden semesterets start og de resterende 4 blev indmeldt efter offentliggørelsen af semesterprojektet. Gruppen redegjorde først ideer til projektet og blev hurtigt forventningsafstemt om projektets resultat skulle være middelmådigt.

4.0.2 Projektforløb og møder

Projektets endelige mål har været implementeringen af et text-based adventure game. Hertil har gruppen gjort brug af agile processen til at imødekomme dette mål.

Først har gruppen specifiseret Kravspecifikationerne for spillet, med ønsker om hvilke features spillet har skulle inkludere. Continuous integration er dernæst været et vigtigt værktøj til implementeringen af de ønskede features, med et klart ønske om at integrere nye features ind i projektet få tidligt som muligt.

I løbet af projektforløbet blev der afholdt to faste ugentlige møder. Et internt gruppemøde om mandagen og vejledermøde om onsdagen. Hertil var der fra start uddelt rollerne Mødeleder og referent til gruppens medlemmer som gik på skift hver uge. Til hvert møde blev der gennemgået hvad der var blev arbejdet med, hvilke udfordringer der forekom samt hvad der fremadrettet skulle arbejdes med for hver af gruppens medlemmer. Dette gjorde vi i form af SCRUM for at give et bedre overblik for projektets fremgang og dermed vurderer om gruppen var bagud eller foran ift. Gruppens tidsplan. Vores SCRUM-proces var ikke avanceret hvor der blev koordineret SCRUM Master og Product Owner, men blev derimod anvendt som redskab til at skabe et overblik over arbejdsfordelingen og udviklingen. Tidsplanen blev udarbejdet ved projektets start og havde formålet at give gruppen deadlines for projektets udviklingsproces.

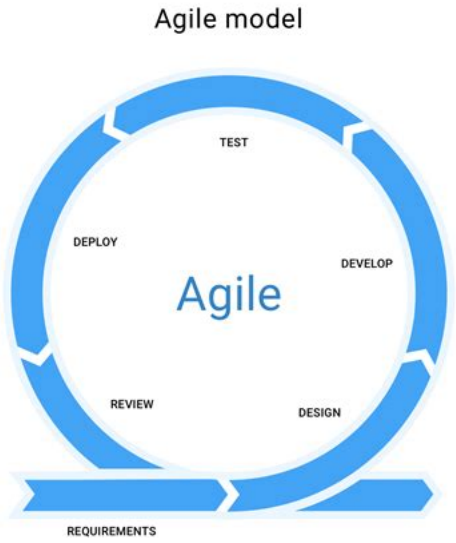

Figure 2: Viser et billed af agile modellen som består af mangle iterationer af design, develop, test, og deploy. Dette minder om continuous integration og har hindret mange problemmer for udviklingsforløbet

SCRUM og AGILE bringer klarhed til medlemmerne om deres roller og opgaver over en kommende tidperiode med en backlog over opgaver, som skal færdiggøres over et sprint (1 uge). Ugentlige opdateringer og møder omkring potentielle problemmer udviklingsforløbet betyder at gruppen har kunne tage hånd om evt. problemmer tidligt i forløbet og derved løse dem før de har udviklet sig til større problemmer.

Til styring og implementering af souce code har vi benytte git som et version control system. fordelen ved git er at det tillade flere udviklere at arbejde på det samme projekt og integere deres løsninger. Der er dog chancer for merge konflikter hvilket kan, i værkste tilfælde, før til problemmer med at integrer de forskellige løsninger korrekt med hinanden.

4.1 Modellering

Projektet benytter UML til at beskrive og modellere software– arkitektur og design, hvilket gør det nemt at simplificere og visualisere strukturen på software løsningen.

Selve arkitekturen er vist med C4 modellen som giver et lageret indblik i både akitekturen på et højt niveau, men med evnen til at give en detaljeret beskrivelse af systemmets komponenter og deres kommunikation med hinanden.

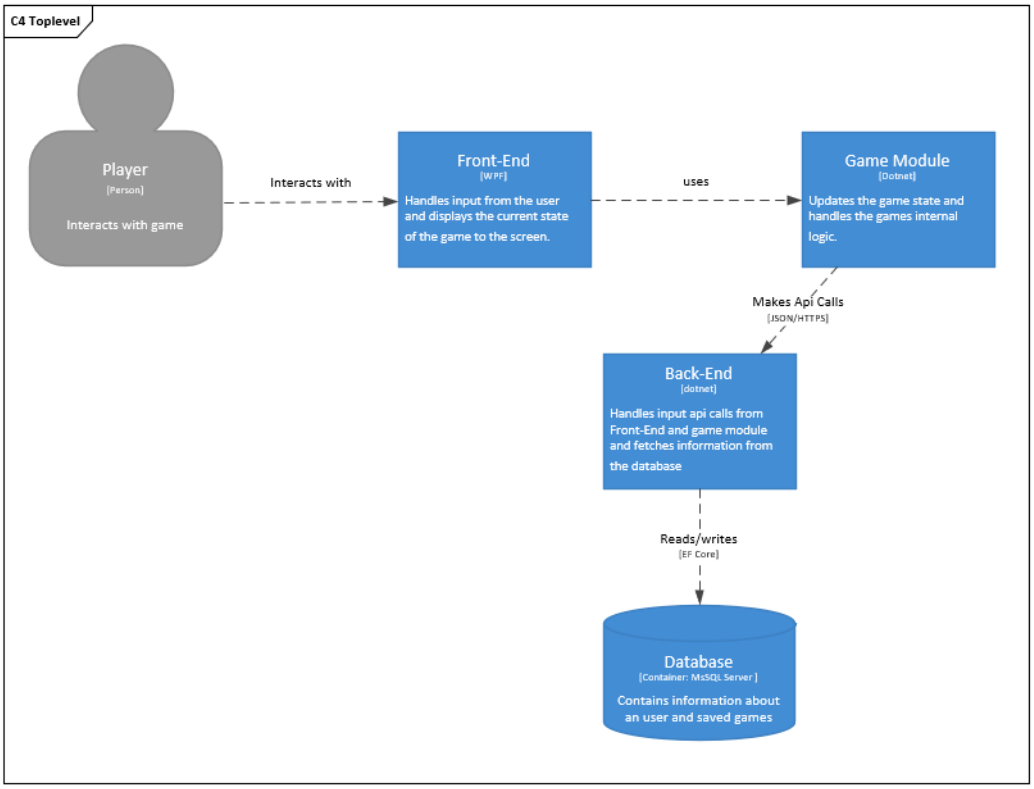

Figure 3

STM og SD diagrammer er brugt til at give programmets udførsel struktur, og fungere som en hjælp til at visualisere programmets forskellige states og flow of execution. Det giver et nemt overblik over hvordan funktionskald mellem forskellige komponenter har skal fungere således, at der ikke opstår misforsåelse grupperne imellem.

Generelt er arkitekturen opbygget som "pseudo" diagrammer. Dvs. de ligner ikke fuldstændigt det endelige design, men har til formål at vise den overordnede tankegang i systemet og samtidig virke som et udviklingsværktøj til at videreudvikle systemet. Da der i gruppen er blevet arbejdet iterativt, er der forskelle mellem diagrammer for arkitektur og design, da der er blevet tilføjet flere moduler og funktioner undervejs. Den overordnede tanke i projektet er der dog ikke blevet ændret på.

4.1.1 Iterativ Udviklingsforløb

AGILE og continuous integration fungere på en naturlig iterativ måde, der tillader ændringer til måde designet og implementeringen undervejs i udviklingsforløbet. Under hvert SCRUM møde er der taget stilling til om, der skulle laves ændringer i gruppens tilgang til projektet, altså om en implementering skulle ændres.

Et eksempel har været kommunikationen mellem Game Engine og Frontend, hvor frontend har haft svært ved at håndtere komplicered return types. Der er sålede lavet ændringer for at gøre det nemmere for frontend at udnytte den infomation som Game Engine har returneret efter et funktionskald.

Denne flexible arbejdsmetode har igen ført til at problemmer ikke har kunne vokse men at der blevet taget hånd om dem mens det stadig har været muligt at håndtere dem.

5 Arkitektur

5.1 Systemarkitektur

Dette afsnit forklarer hvordan systemets toplevel arkitektur er opbygget. Denne består af en bruger som interagerer med systemet gennem frontendapplikationen som skrives i WPF. States i denne applikation styres af Game modul som holder styr på hvor spilleren befinder sig, hvilke items der er samlet op og andre nyttige information som skal bruges gennem spillet. Spillets backend benyttes primært til bruger authentication og som bindeled til databasen. Databasen indeholder oplysninger om blandt andet brugere, de gemte spil og oplysninger om historien for de forskellige rum. For at se en visuel repræsentation af hvordan de forskellige moduler snakker sammen se Figure 7

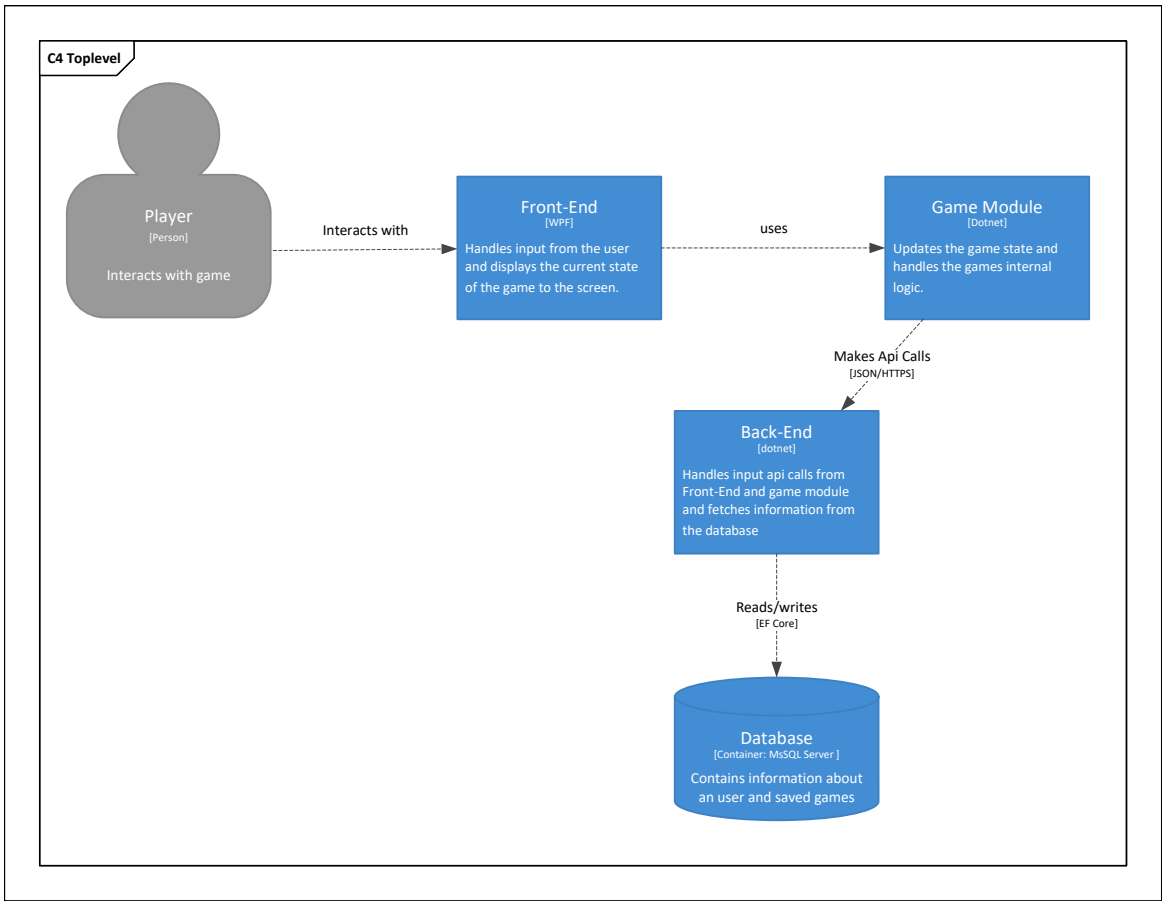

Figure 4: C4 Top-Level diagram for systemets arkitektur. Her ses et diagram for systemets Top-Level arkitektur, hvori der er skabt et overblik over hvilke moduler der er til stede i systemet og hvordan de kommunikere. Heri er der også tilføjet kommunikationsmetode for de forskellige forbindelser.

5.2 Frontend Arkitektur

Front-end applikationen er det man kan kalde, brugerens vindue til spillet, det er i dette modul at brugeren vil få alt sit information og vil få mulighed for at lave inputs til spillet. Det er yderligere her at der skal tages hånd om brugerens input således at de rigtige funktioner i Game Enginen bliver kaldt når brugeren trykker på en vilkårlig knap.
Front-end'en er opbygget af et hav af forskellige skærme og menuer, og for at give et overblik over

disse er der lavet følgende C3 model for Front-end'en (Figure 5), som giver en idé om hvilke skærme og menuer der kan gå til hvilke andre menuer/skærme. Udover dette fortæller modellen også om hvordan og hvilke skærme og menuer der snakker med noget uden for Front-end'en selv f.eks. skal der ved Login/Register kontaktes databasen for at få verificeret logind-oplysninger, og samtidig skal der ved save og load-game hentes en liste af gemte spil i databasen hvorefter der skal henholdsvis skrives og hentes fra databasen alt efter om man gemmer eller henter et spil. Der skal hertil nævnes at Login og Register står til at snakke med backenden direkte og ikke igennem gamelogic blokken, dette er valgt da funktionen ikke kaldes i gamelogic blokken, men den kaldes direkte i backend controlleren. Derudover er modellen mere overskuelig på denne måde og fungerer bedre til at give overblik over navigationen igennem menuerne.

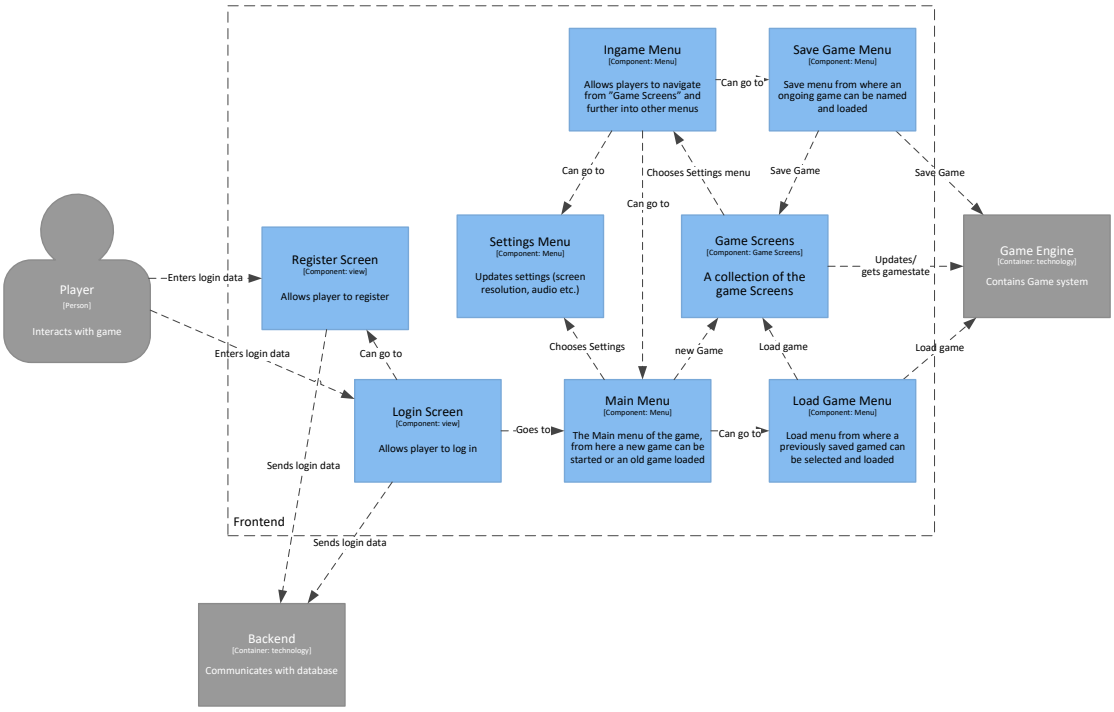

Figure 5: C3-Model for Frontend. Modellen fortæller hvordan man kan navigere igennem forskellige menuer og hvilke menuer der kan føre til hvad. Derudover kan man se hvilke blokke der snakker ud af frontenden og sammen med resten af systemet.

5.2.1 Pseudo Frontend Arkitektur

For at give overblik over, hvordan kommunikationen mellem frontend, backend og gamecontroller kommer til at foregå, er der lavet et pseudo sekvensdiagram for følgende UserStories:
- Login
- Register
- Save Game
- Load Game

Der vil i dette afsnit kun blive vist "Save Game". "Load Game","Login" og "Register" kan findes i Tekniskbilag **(INDSÆT REFERENCE HER)**.
På Figure 6 ses "Save Game", som viser forløbet når en bruger gerne vil gemme sit igangværende

spil, set fra Frontends perspektiv. Her kan man bide mærke i, at når der skiftes skærm, vil den nye skærm få initialiseret sine variabler i sin constructor og derfor er der kun et selvkald, hver gang skærmen skiftes. Dette selvkald tager hånd om at opdatere mappen således at spilleren er i det rigtige rum, og at man kun kan se det af mappen, man har været i. Hertil skal der nævnes at hvis brugeren er i "Combat State" er det ikke muligt at gemme spillet og knappen "Save Game" på "In Game Menu" vil ikke kunne ses eller bruges.

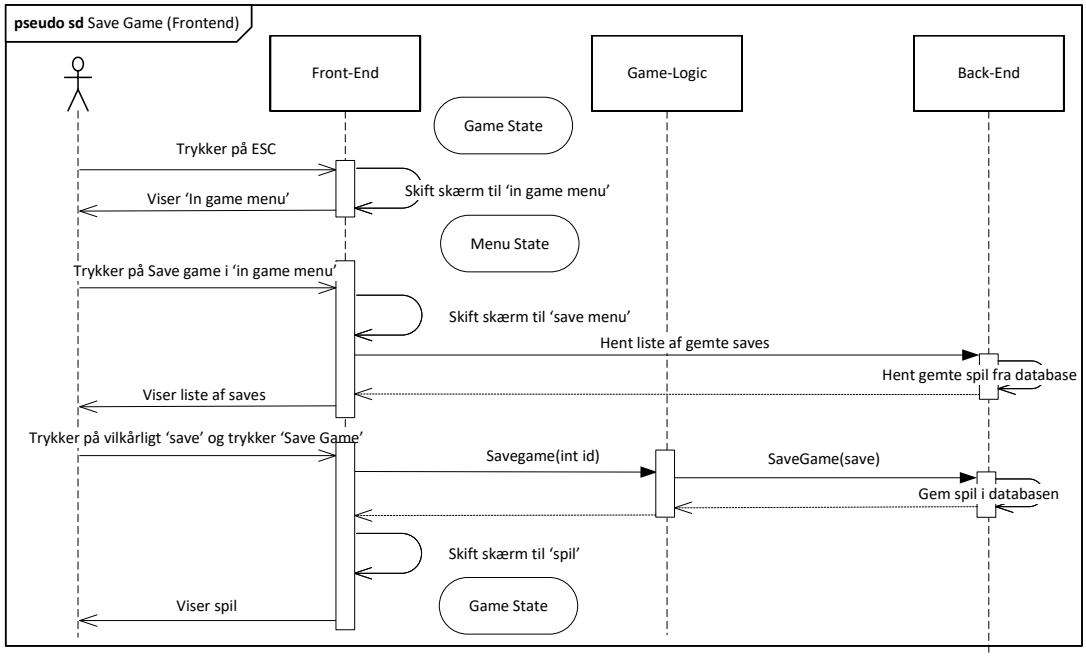

Figure 6: Pseudo sekvensdiagram af forløbet af userstory "Save Game", set fra Frontends perspektiv. Der laves 2 kald til databasen igennem Backenden, hvori der i det første kald, "Hent liste af gemte saves" hentes en liste af brugerens gemte spil og i andet kald gemmes brugerens nuværende spil henover det valgte spil.

6 Design

6.1 Overordnet System Design

Dette afsnit repræsenterer hvordan modulerne ønskes at kommunikere med hinanden samt give et overblik over hvor meget der kommunikeres mellem modulerne i de forskellige stadier. Der vil i dette afsnit indgå 4 User stories til at repræsentere systemets design. Disse User Stories er relevante for design, da de involverer alle moduler samtidigt og giver et indblik i kommunikationen på tværs af systemet.
Herunder ses et sekvensdiagram for User Story 7-10 - Load. Her ses hvordan det ønskes at de forskellige moduler kommunikerer på tværs af systemet når en bruger loader et save fra databasen.

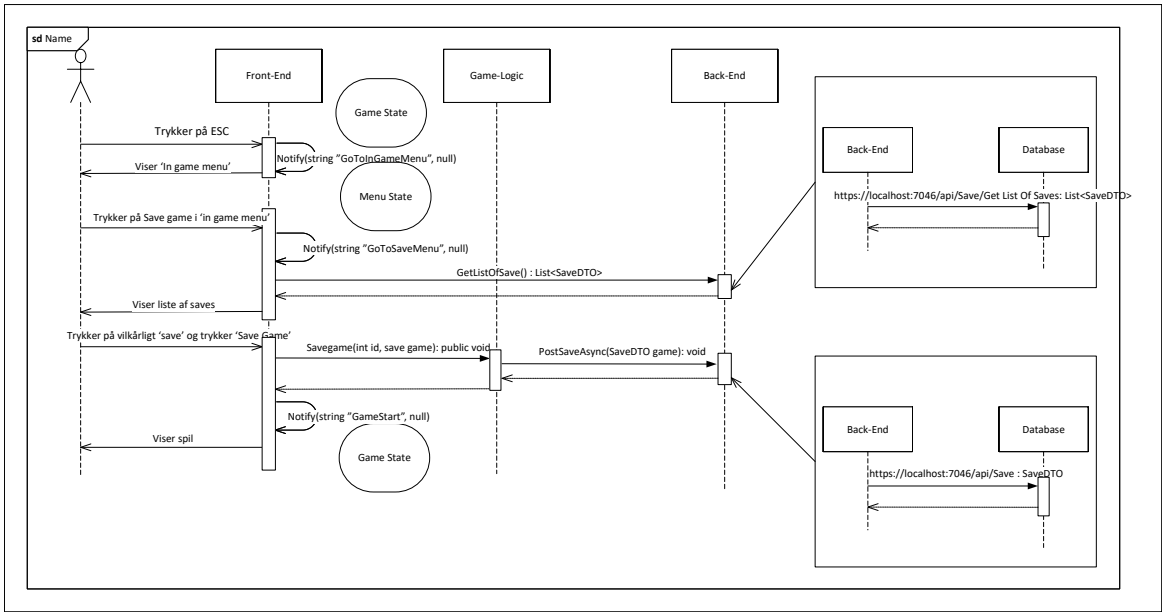

Figure 7: SD diagram for User Story 7-10. Diagrammet viser hvordan det ønskes at systemet overordnet skal kommunikere på tværs af hinanden når en bruger skal gemme sit aktuelle save

Herunder ses et sekvensdiagram for User Story 17-18 - Load. Her ses hvordan det ønskes at de forskellige moduler kommunikerer på tværs af systemet når bruger gemme et save i databasen.

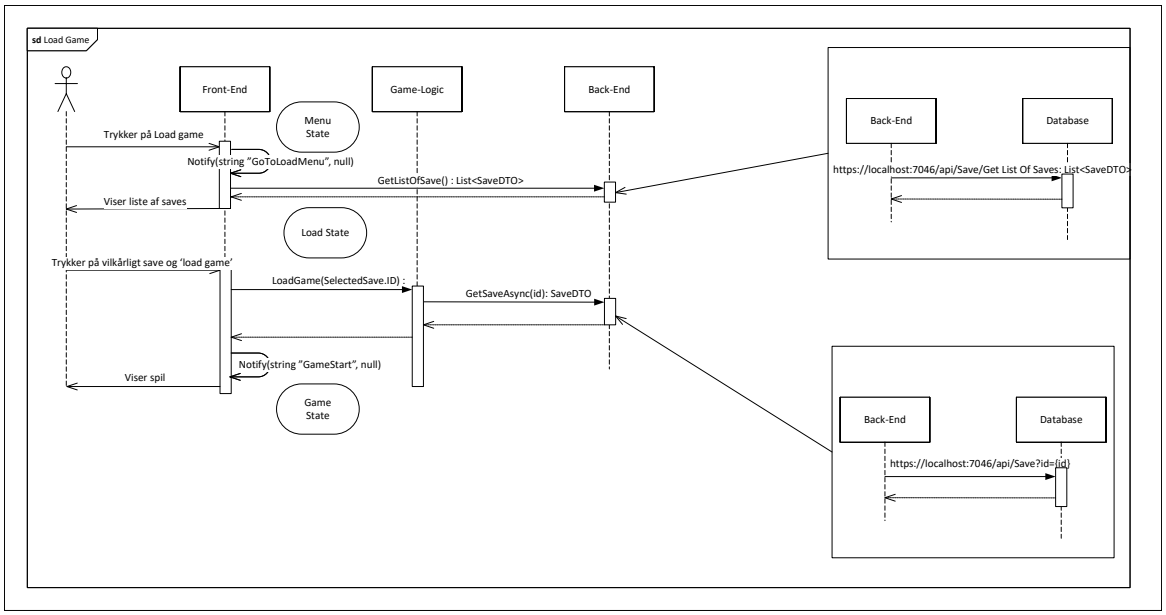

Figure 8: SD diagram for User Story 17-18. Diagrammet viser hvordan systemet overordnet skal kommunikere på tværs når bruger skal loade sit aktuelle save.

Herunder ses et sekvensdiagram for User Story 1 - Log in. Her ses hvordan det ønskes at de forskellige moduler kommunikerer på tværs af systemet når bruger skal logge ind.

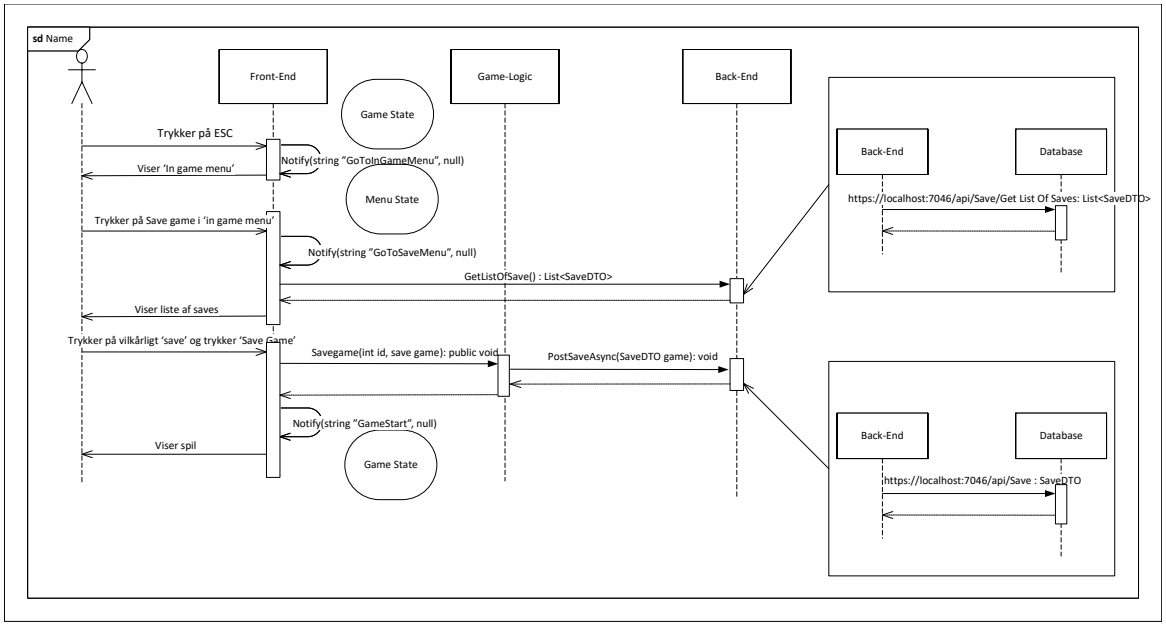

Figure 9: SD diagram for User Story 1. Diagrammet viser hvordan systemet overordnet skal kommunikere på tværs når bruger skal logge ind

Herunder ses et sekvensdiagram for User Story 2 - Opret Bruger. Her ses hvordan det ønskes at de forskellige moduler kommunikerer på tværs af systemet når bruger skal oprette en bruger i systemet.

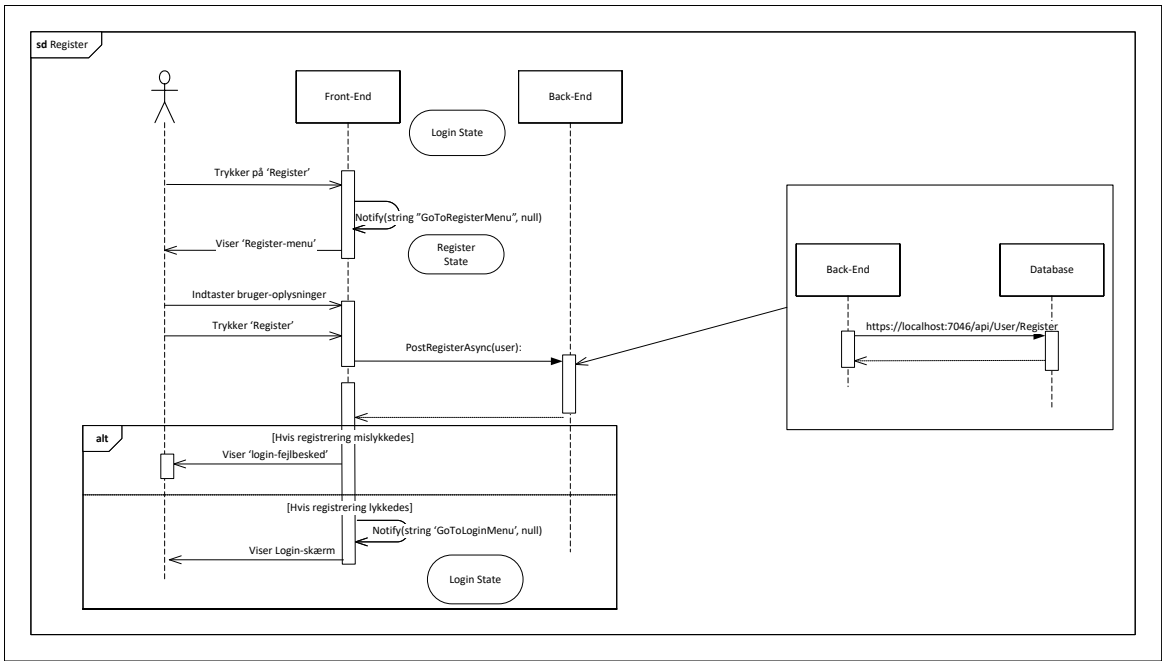

Figure 10: SD diagram for User Story 2. Diagrammet viser hvordan systemet overordnet skal kommunikere på tværs når bruger skal oprette en bruger i systemet

6.2 Game Engine

Spillets logik styrres fra game engine, der består af de komponenter, som modellere spillets verden og dens logik. Af disse komponenter er Game Controlleren, Combat Controlleren, Room, og Player de mest essentielle.

6.2.1 Game Controller

Game Controlleren styre alle UI funktionaliteter, når brugeren trykker på en knap, der påvirker spillets state, gøres dette gennem game controlleren. Game controlleren instantieres når et spil startes; hvilket danner et map [Section 10.3.1 1], spilleren kan navigere rundt på.

Game controlleren tillader at spilleren kan bevæge sig mellem spillets forskellige rum [Section 9.3.1 1] i overenstemmelse med spillets map layout. Skulle spilleren forsøge at bevæge sig i en retning, der ikke er tilladt ifølge map layoutet bliver spilleren stående i det nuværende rum.

Skulle spilleren bevæge sig ind i et rum med et "item" styre game controlleren spillerens evne til at inteagere med "itemet". Spilleren har evenen til at samle "items" op, og derved øge deres stats.

I det tilfælde, at spilleren bevæger sig ind i et rum med en modstander, giver game controlleren spilleren evnen til at bekæmpe fjenden eller flygte fra denne. Game controlleren gør dette ved at kalde combat controlleren, som håndtere kamp i spillet.

6.2.2 Combat Controller

Combat controlleren håndtere spillet logik, i det omfang det relatere sig til kamp mellem spilleren og en fjende [Section 9.3.3 1, Figur 17].

Combat i spillet afvikles ved hjælp af flere simulerede terningekast (RNG), der afgører både om spilleren/fjenden rammer fjenden/spilleren og hvor meget skade fjende/spilleren modtager. Spillets "items" har indflydelse på vægtningen af disse simulerede terningekast, og kan hjælpe spilleren med at ramme og skade modstanderen. "items" kan også gøre det svære for fjenden at ramme spilleren.

Spilleren kan efter hvert sæt af terningekast vælge at forsætte kampen eller flygte. Skulle spillerens liv nå nul, dør spilleren og spillet er færdigt.

6.2.3 Room

Et "Room" eller rum er en diskret delmængde af spillets verden, som kan indeholde spilleren, fjender og forskellige genstande. "Room" er kun ansvarlig for at tilføje og fjerne spilleren og fjender for rummet. Ikke desto mindre kunne spilleren ikke navigere verden uden existensen af disse diskrete områder.

6.2.4 Player

Spilleren karakter (PC) repræsentere spilleren i spillet. Dette komponent for at holde styr på spillerens tilstand i spille og giver spilleren evnen til at forsvare sig selv i kamp. Denne simulere en spillers forsøg på at ramme en fjende, skade en fjende og opdatere spilleren liv, forsvar og "items" undervejs i spillet [Section 13.3.2 1].

6.3 Database Design

Projektets database har gruppen valgt at hoste i lokal storage. Dette er valgt da der under semesterets forløbet opstod problemer med skolens licens af Microsoft produkter. For at undgå at komme ud for udfordringer med hosting senere i forløbet, blev der valgt at gå med den sikre løsning, at hoste databasen lokalt på enheden. Til dette benyttede gruppen et docker image[**SQL server with docker**], specifikt det samme image som blev benyttet i DAB undervisning, til vores SQL server. Hvis ikke der var problemer microsoft, havde gruppen i stedet valgt at lagre data på en cloud-based storage fremfor lokal storage.

For at kunne udarbejde et ER diagram til modellering af vores sql database skal vi start med at finde ud af hvilke krav vi har til og hvilke attributter vi ønsker at gemme i vores database. Først og fremmest ønskede gruppen at vi kunne gemme beskrivelserne af de forskellige rum, i spillets layout, for at formindske antallet af filer i klienten, og samtidigt gøre eventuelle senere tilføjelser nemmere. Her benyttes rummets id som key, da vi ikke ønsker at man skal kunne oprette flere beskrivelser til samme rum. Diagrammet kan ses på Figure 11.

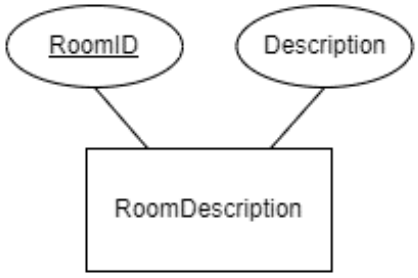

Figure 11: ER diagram for Roomdescription. En beskrivelse består blot af en beskrivende string samt det tilhørende unikke rumid.

Her efter kommer kravene til at kunne gemme et spil for en bruger. Her ønskede vi at man kunne stå et vilkårligt sted i spillet, med untagelse af en combat, og gemme spillet. Det skulle derefter være muligt for spilleren at loade spillet igen, hvorefter spillet er i samme stadie som man gemte det i. ER diagrammet for at gemme et spil til en bruger på Figure 12.

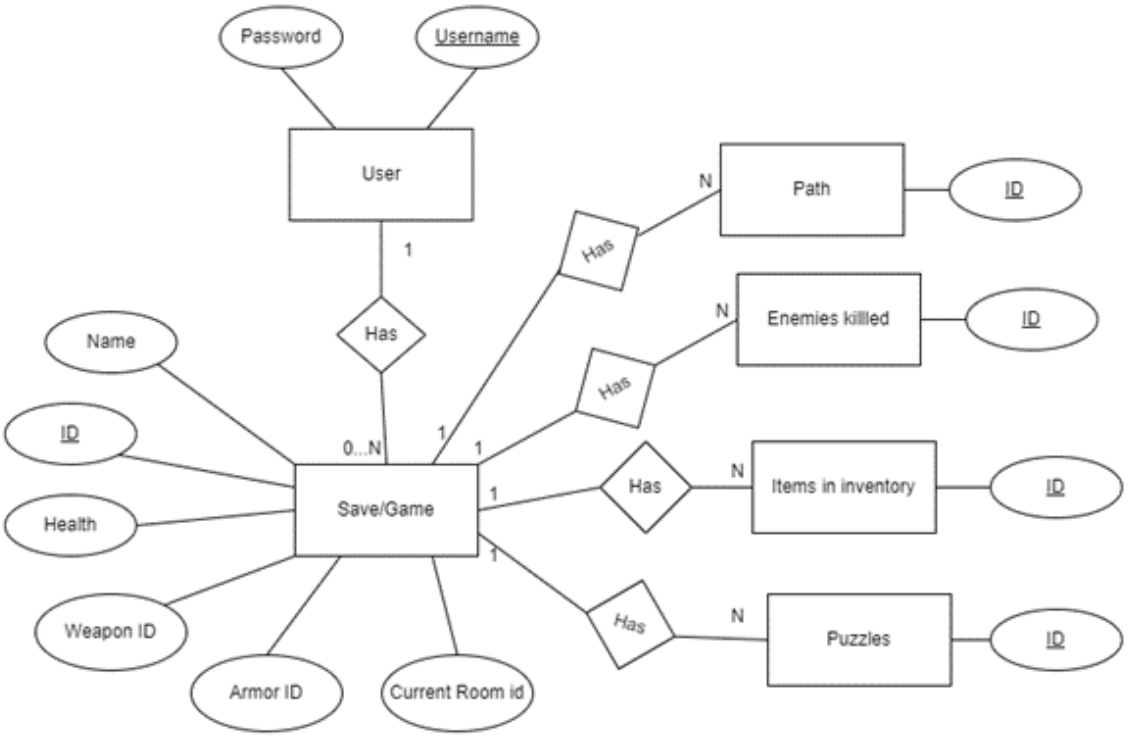

Figure 12: ER Diagram til at gemme et spil til en specifik bruger. Her ses de forskellige informationer som skal til for at kunne gemme et helt spil

Først og fremmest ønskede gruppen et bruger system, så eventuelle gemte spil kun tilhørte en bruger. Der gemmes derfor en bruger entitet med et unikt brugernavn og et tilhørende password. Sikkerhed på password og versalfølsomhed på brugernavnet håndteres af spillets backend.
En spiller skal derefter kunne gemme 5 unikke spil med forskellige oplysninger. Restriktionen med max 5 forskellige spil pr. Bruger, håndteres ved at oprette 5 "tomme" gemte spil ved oprettelsen af en bruger. Et af disse gemte spil overskriver derfor når vi gemmer. Der kan på denne måde ikke oprettes mere en 5 gemte spil pr. bruger.
I et gemt spil ønkede vi at gemme en række forskellige attributter for spilleren. Første og fremmest får hvert spil et unikt id som vi benytter til identifikation og at lave forhold mellem de forskellige tabeller. Et gemt spil får et navn, valgt af brugeren, som gør det nemmere for brugeren at differentiere mellem de forskellige spil.
Dette navn skal være forskelligt fra de 4 andre gemte spil som tilhører samme bruger.
En spillers Health gemmes også, da man kan have taget skade efter en kamp.
Det gemmes også hvilket rum, spilleren står i når spillet gemmes, så vi loader korrekt tilbage. En spiller kan derudover også holde genstande, som armor og våben, i hånden eller i sit inventory. Dette gemmes også henholdsvist som en del af et spil og i en inventory liste tilhørende spillet.

Tabellen med inventory har 2 atributter, et ID, som svarer til en bestemt genstand, og en reference til et SaveID. Denne parring er unik, da man ikke kan holde 2 af den samme genstand.
Tabellerne med Enemies og puzzles fungerer på samme måde. Her har hver enemy og puzzle i spillet et unikt id. Id'et gemmes i kombination med et saveId, som et unikt par, da man ikke kan vinde over samme enemy og løse samme puzzle flere gange.
Til slut ønskede vi at kunne vise spilleren de rum som allerede er blevet besøgt. Derfor gemmes der i path tabellen, for hvert spil, en unik kombination af saveID og besøgte rum id. Denne parring er unik da man blot behøver at besøge et rum en gang, før det er synligt på kortet.

Der er i projektet oprettet klasser tilsvarende ER diagrammerne. Forholdene mellem disse, samt keys, er opsat i DaG-db klassen og er skrevet ved hjælp af fluentAPI. Dette kan ses i Implementationsafsnittet **??**.

7 Implementering

7.1 System Implementering

Under implementeringen af nogle funktioner i samtlige moduler, er funktioner blevet lavet til funktioner der følger følgende opstilling:
public async Task <T> Foobar();
Funktionerne returnerer en Task af typen T og den kan gøres asynkront. Dette er specielt vigtigt når man kontakter databasen, da programmet ikke må køre videre før den asynkrone funktion er færdig med sit database kald. Når en funktion i f.eks. Game Controlleren kalder: public async Task <SaveDTO> GetSaveAsync(int id) i backend controlleren, i sin egen funktion SaveGame() skal SaveGame() også erklæres async og returnerer en Task af typen T og derfor er nærmest alle funktioner der kontakter databasen erklæret for async kald, som returnerer typen Task.

7.2 Game Engine

8 Test

Testing er en grundsten i ethvert succesfuldt softwaresystem. Uden testing kan der ikke stilles garanti for et systems opførsel. Nedenstående afsnit dækker alle test foretaget af "Dungeons and Gnoblins" spillet, for at sikre den korrekte opførsel i henhold til kravspecifikationerne.

Her dækkes test af alle de største komponenter, Frontend, Game Engine, Backend og database. Testene inkluderer automatiseret unittests, integrationstest og accepttest.

8.1 Modultest Frontend

Modultesten af Frontenden er lavet i meget tæt samarbejde med Game Enginen. Dette er valgt sådan, at når Game Engine lavede en ny funktion eller funktionalitet, gik Frontend holdet igang med at implementere en visuel repræsentation af denne nye funktion/funktionalitet. Således var det ikke kun en visuel test af at views så godt ud eller at man kunne trykke på en knap, men derimod kunne både frontenden og Game Enginen testes i samarbejde, hvor den reelle funktionalitet for systemet blev testet.

8.1.1 Test metoder

Hver gang der er blevet lavet små eller større ændringer i frontenden, er der blevet lavet både en funktionel og visuel test af de nye ændringer. Dette blev gjort for æstetiske ændringer ved at kigge i preview vinduet i WPF for det view der blev ændret. Var det derimod en ændring der inkluderede databinding til Game Enginen, blev det nødvendigt at teste hele programmet ved brug af compileren og derved lave en runtest, som inkluderede at det rigtige data blev hentet fra Game Enginen eller at den rigtige funktionalitet blev kaldt ved et tryk på en knap.

For eksempler på hvordan Frontenden specifikt er blevet testet i forbindelse med Game Enginen se Teknisk Bilag **(INDSÆT REFERENCE HER)**.

9 Fremtidigt Arbejde

Sammenlignes det færdige produkt med det produkt der blev diskuteret under idefasen, er der nogle funktionaliteter som ikke er blevet færdiggjort, men kunne have været implementeret. Spillet kunne have haft flere udvidelser såsom Puzzles, flere niveauer, character udvikling og quest items som kunne have givet bruger en mere underholdende oplevelse og mere avanceret gameplay. Dette høre ikke under et specifikt modul i systemet, men tilføjes over alle moduler, hvis nogen af de overstående features blev tilføjet.

F.eks. hvis der blev tilføjet flere niveauer, skulle der implementeres et nyt map i Game Controller og Front-end siden. Derudover skal der tilføjes en udvidelse af load og save game, så Back-end og database kunne gemme specifikt for det nye niveau, hvilket niveau er spilleren på, hvor langt brugeren er på det nuværende niveau og til sidst hvor meget spilleren har udforsket af tidligere niveauer.

Specifikt for Database og Back-End, kunne der tilføjes en lagring af data på en cloud-based storage fremfor lokal storage. Der blev valgt under forløbet at lagre dataen i en local storage, da der i midten af semestret var problemer med skolens licens af Microsoft. For ikke at komme ud for udfordringer senere i forløbet, blev der valgt at gå med den sikre løsning, at hoste databasen lokalt på enheden. Resultatet af et cloud-based lagring ville resultere i at spillet ville være mere tilgængelig for brugere på forskellige enheder eller platforme.

Derudover kunne spillet laves til at kunne køre på andre styresystemer som f.eks. IOS eller Unix. Dette ville lade en bredere målgruppe spille spillet og derved give en større kundegruppe. Denne ændring ville dog resulterer i at det valgte framework til Frontenden skulle skiftes til noget andet, såsom Unity. Dette ville betyde at frontenden og backenden skulle omskrives helt, men det kunne også give en generel bedre spil oplevelse.

10 Konklusion

Gruppen har endt ud med et funktionelt projekt, hvor de vigtigste features blev implementeret. Der er nogle features som ikke er blevet lavet og er derfor sat til fremtidigt arbejde. Ud fra projektet blev der valgt at disse features ikke var nødvendige for at opnå et funktionelt spil, som kunne spilles igennem og stadig fungere som et spil med fjender og genstande, som kunne samles op og anvendes. Dette blev gjort da der var fokus på kommunikation mellem systemets segmenter og integrationen af disse segmenter i sidste ende. Modulerne er dog lavet med fremtidigt arbejde i tanken, så f.eks. i Game Module er det lavet relativt nemt at implementere yderligere features, så fremtidige features kunne implementeres uden at komprimere kommunikation mellem systemets segmenter.

10.1 Personlige konklusioner

10.1.1 Luyen Vu

Dette semesters projekt har været spændende og anderledes. Anderledes grundet at vi i dette semester ikke skulle arbejde med embedded software. Jeg vil argumentere for at vi i dette semester anvendte semestrets fag mere sammenlignet med de andre semestre, da vi som gruppe delte vores moduler op i de forskelige fags faggrupper. Derved har spørgsmål og troubleshooting været nemmere, idet at vi alle havde en ide om hvordan de forskellige moduler fungerede ift. en opdelling af Software- og elektronikstuderende. Vores produkt er endt fint ud i min optik og det har været et interessant projekt hele vejen igennem. Det faktum at det kun var software har også haft en indflydelse på mit syn af interessen af projektet.

10.1.2 Oscar Dennis

Projektet dette semester fungerede på en anden måde en projekter, som man har været vant til. Dette var på baggrund af at dette projekt ikke kravede samarbejde med elektronik ingeninørerne, og var derfor et rent software baseret projekt. Dette resulterede i at da vi alle var på samme studie, så var det nemmere at forstå og reviewe andres arbejde. Yderligere opstod der problemer med integration af de forskellige segmenter, da grupperne, såsom Front-end, Back-end og database, arbejde meget individuelt og opdelt. Derfor var der adskellige problemer som skulle løses før systemet kunne sættes sammen og fungere.
Dog har gruppen endt ud med et funktionelt projekt som jeg personligt er meget tilfreds med.

10.1.3 Rasmus Engelund

Dette projekt har været positivt anderledes end de tidligere, da vi denne gang skulle lave et helt softwarebaseret projekt, og ikke skulle forholde os til elektronik og embedded programmering.
Vi har i projektet formået at komme godt rundt i de forskellige fag hvor vi har inddraget nyttig viden, som vi har tilegnet os i de forskellige kurser. Her har projektets opbygning gjort opdeling af arbejdsopgaver nemmere, dog med undtagelse af nogle småting hvor kommunikationen og klarheden omkring arbejdsfordelingen, kunne have været bedre fastlagt. Vi fik implementeret næsten alle de forudbestemte features, til en tilfredsstillende grad. Generelt er vi, trods min umiddelbare tvivl om emnet, endt ud med et fornuftigt projekt, som jeg personligt er fint tilfreds med.

10.1.4 Magnus Blaabjerg Møller

Igennem dette projektforløb har jeg fået et indblik i hvad det vil sige at arbejde med et rent software projekt. Der er blevet anvendt en række nye dokumentationsmetoder end på de tidligere semestre, til at dokumentere de forskellige aspekter i projektet som user stories og C4-modellen. Det har været lærigt med en ny tilgang til tingene. Jeg har gjort mig nogle gode erfaringer omkring integrationstest og test generelt af rene software projekter, blandt andet hvor vigtigt det, er at have kommunikationen igennem systemet på plads i et tidligt stadie, således at alle i gruppen er enige om hvordan kommunikationen fungerer, så der ikke opstår komplikationer under integrationstest. Derudover har jeg erfaret hvor vigtigt det er at teste tidligt og ofte hver gang der ændres i implementeringen af et modul. Hvad angår samarbejdet i gruppen har det generelt set fungeret godt, og det har været muligt at spørge hinanden om hjælp, da alle har haft en hvis viden inden for de forskellige tekniske områder igennem semestrets fag.

10.1.5 Morten Høgsberg

Dette er det første projekt uden restriktioner på projektet emne. Projektet har således for første gang være et rent softwareprojekt. Denne gang har ideèn været at implementere et spil, hvilket er gået over alt forventning. Arbejdsmoralen har været høj igennem projektet, med input fra alle medlemmer, og en villighed til at lytte og give konstruktivt feedback til hinanden. Det har været nemmere at følge up på hinandens arbejde da vi alle har haft en basis forsåelse af hvordan de andre har udført deres opgaver. I sidste ende har vi leveret i projekt, som opnådede at implementere de fleste features og som fungere bedre end forventet.

10.1.6 Anders Hundahl

Personligt har dette års semesterprojekt vist sig at være mere opslugende, men også mere belønnende end de tidligere projekter. Til dels tænker jeg at det skyldes at vi i år har kunnet fokusere hundrede procent på softwaren, og at vi har kunnet teste vores produkt uden at skulle sætte en masse ledninger og hardware op først, som har kunnet skabe støj eller problemer i systemet. Derudover har vi i dette

semester haft mulighed og evner til at skabe et produkt som reelt set kan noget og som man kunne se virke ude i den virkelige verden. I forhold til samarbejde i gruppen føler jeg at vi har fundet ud af at arbejde sammen på en ordentlig og professionel måde, men hvor der samtidig har været mulighed og plads til at have det sjovt indimellem.

10.1.7 Sune Dyrbye

Dette semesterprojekt har været det bedste projekt jeg har været en del af. Samarbejdet i gruppen har været problemfrit og vi har været i stand til at færdigørevores produkt. Det har været rart at arbejde på et projekt som var rent software, da det har tilladt os at kunne teste vores implementeringer tidligt, uden at skulle vente på at hardware blev lavet. Det har været godt at kunne bruge det vi har lært i de forskellige fag i projektet. Gruppen har været godt og der har været plads til at lave sjov, mens der stadig blev arbejdet koncentretet.

10.1.8 Jacob Hoberg

Dette har været det første projekt, hvor det virkelig har følt som om at man havde frie tøjler. Det er resulteret i at vi har fået lavet et projekt, der har været meget sjovt at arbejde med. Samarbejdet har været godt i gruppen, da der har været stor enighed i hvad man kunne tænke sig at arbejde med, samt ambitionsniveauet har været meget ens. Desuden har der været god kemi blandt gruppemedlemmerne, hvilket har resulteret i at det ofte har været en fornøjelse at sidde og arbejde på projektet. Arbejdet har dog været meget opdelt internt i gruppen, og dette viste sig tydeligt da vi skulle integrere vores system. Så til fremtiden, skal vi være lidt bedre til at vise de ting, vi laver, frem for hinanden.

References

[1] Magnus Blaabjerg Møller et al. *Teknisk Bilag - Dungeons and Gnoblins*. 2022.