2. semester eksamensprojekt

Mathias Thomsen(27-12-1998)
math31t6@stud.kea.dk github: Mathias-Thomsen

Marcus Frørup Nielsen(18-06-2002)
marc599n@stud.kea.dk github: MarcusFN

Christian Skovbæk Munk-Nissen(18-09-2000)
chri47nj@stud.kea.dk github Chrilleweb

DAT22D
KEA Københavns Erhvervsakademi
Afleveret 01/06/2023

Github repository:
https://github.com/Atributties/ProjectPeak.git

Indholdsfortegnelse

Indledning

Project Peak er en applikation for projektstyring, som er designet til at hjælpe virksomheden Alpha Solutions til at opnå deres mål og styre deres projekter effektivt. Vores mål med denne applikation er en brugervenlig platform, der skaber overblik og øger produktivitet, samt skaber et visuelt billede af tids estimationen af projekter.

Project Peaks formål er at hjælpe brugerne ved at skabe deres projekt på en mere effektiv og visuel måde. Derudover ønsker vi at kunne formidle en mere forenklet proces ved kompleksiteten af at oprette et projekt.
En effektiv applikation er en nødvendighed indenfor projektstyring, for at kunne opnå succes og nå alle ens mål.

Project Peaks målgruppe henvender sig mest til virksomheder og professionelle teams, men uanset om det er projekter til professionelt brug, så kan Project Peak også sagtens hjælpe med de mere personlige opgaver, formålet med Project Peak handler i bund og grund om at skabe et mere visuelt projektstyringsprogram.

Udviklingen af Project Peak har fokus på de vigtigste funktionaliteter som oprettelse af projekter, samt opgaver og delopgaver. Project Peak lægger også vægt på, at applikationen er brugervenlig og nem at forstå, hvilket sørger for at imødekomme brugernes behov.

I denne rapport vil vi præsentere en analyse af Project Peak, som bla. vil medføre de forskellige værktøjer der er blevet brugt til at skabe Project Peak.

Software

Værktøjer

Github 3.8.0 & Github actions

I udarbejdelsen af vores projekt har github været en central del af vores udviklingsarbejde, hvori vi har anvendt en vifte af funktioner tilgængelige som f.eks. gemme, administrere og dele kode. Github fungerer som et distribueret versionskontrolsystem, hvilket vil sige at det sporer og gemmer ændringer i koden hen over tid så koden kan dokumenteres.

Issues er også en indbygget funktion vi har anvendt hyppigt, der har givet os mulighed for at planlægge og organisere udviklingen, ved at oprette opgaver med tilhørende fejlrapporter eller forbedringsforslag, dette hjælper også med kommunikationen så udviklere nemmere kan følge med.

Github indeholder også en lang række andre advancerede værktøjer vi har gjort brug af heriblandt til dokumentation, implementering, testning osv. Et hyppigt anvendt værktøj er her Github actions som vi har anvendt til at automatisere processer for testning og bygning af projektet, dette har betydet en mere stabil og pålidelig udviklingsproces undervejs.

Discord 0.0.309

Vi har brugt Discord til at kommunikere via taleopkald og chat. Discord er en gratis VOIP-applikation[1], vi valgte at bruge Discord, da vi alle kender det i forvejen og derfor havde nemt ved at navigere derinde. Derfor har Discord ligget helt naturligt at bruge i dette projekt.
Vi har brugt Discords værktøjer til at dele skærm med hinanden, det har været en nyttig funktion, så vi alle har kunne kigge med, på den person som er i gang med at vise noget, eller sidder og koder.

[1] https://discord.com/safety/360044149331-what-is-discord

Google drive

Til udarbejdelsen af vores rapport, har vi benyttet os af google drev, der har gjort det muligt for vores projektgruppe at tilgå det samme tekstdokument i google docs. Dette har været oplagt at gøre brug af, da vi via google drive cloud baseret lagring har kunne skrive og redigere på samme tid hen over flere computere. Google docs er også oplagt at benytte, da det giver mulighed for at opstille rapporten efter det rette format.

IntelliJ IDEA 2023.1.1

Som udviklingsmiljø for vores software har vi anvendt os af Intellij IDEA, der har fungeret som et integreret udviklingsmiljø “IDE” for vores udvikling af kode. Intellij har vi primært benyttet til udvikling af Java-baserede software, men har også implementeret en række andre programmeringssprog heriblandt HTML, CSS, JavaScript.

MySQL Workbench 8.0 CE

MySQL Workbench er et visuelt værktøj for databasedesign, som vi har brugt i denne opgave til at lave vores schema og test data til vores program. Vi kender alle MySQL Workbench, så derfor var det oplagt at bruge dette værktøj i vores applikation.
MySQL Workbench har været brugbart til at teste om vi har hentet vores data korrekt og om vi og om vi kan tilføje og opdatere vores data fra vores HTML filer.

Render 14.17.5

Render har vi brugt gratis for at få vores web-applikation hostet, Vi har valgt at bruge Render, da dette er en meget nem måde at få sin web-applikation hostet. Fordelen ved Redner er, at det er gratis, men ulempen ved Render er desværre, at det er lidt langsomt.

Diagrams.net

Diagrams.net har vi anvendt til udarbejdelsen af vores diagrammer, hvilket blandt andet omfatter vores klassediagrammer, Domæne model og ER diagram, Diagrams.net er et free

to use værktøj tilgængeligt på nettet, hvor så der er mulighed for at gemmen sine diagrammer i vores github repository og derfor oplagt at benytte sig af.

Frameworks og teknologier

JAVA/JDK 17

I udarbejdelsen af vores projekt har vi benyttet os af Java teknologien, som det primære programmeringssprog. Java er baseret på objektorienteret programmering, dette betyder at der bruges objekter som abstrakte eller virkelige enheder. Dette har været nyttigt, da det er et sprog vi alle er bekendte med.

Spring Boot 3.0.6

Vi bruger Spring Boot i vores projekt, da vi har brugt det igennem dette semester. Spring Boot er et Framework til Spring, som vi også bruger i projektet. Spring Boot gør det nemmere for os at se hvad vi arbejder med og kan køre programmet på Localhost, Dette har været meget nyttigt da vi har skulle designe vores applikation for at gøre den så brugervenlig som mulig.

Spring Web

Det har været nødvendigt for vores applikation at have Spring Web Dependency, da vi har lavet vores applikation efter MVC-pattern, som giver os adgang til HTML siderne i vores applikation, som kan køre på en Tomcat server.
Det har vi gjort ved hjælp GetMapping og PostMapping i vores Controller,
GetMapping laver vi en GET request, som vi bruger til at hente vores data, og vores PostMapping laver vi en POST request, som vi bruger til at opdatere vores nuværende data. Spring web er et modul under spring boot frameworket og hører derfor under framework versionen som en applikation til webudvikling.

Maven 4.0.0

Maven er et bygge automatiserings værktøj som vi har brugt vores projekt, Maven har sparet os en masse tid på automatisk bygning af vores projekt.
Maven henter automatisk de dependencies som ligger i vores pom.xml fil, dette er med til at sikre os, at vi arbejder på de samme versioner og er med til at hjælpe konsistensen i vores projekt.
Vi har i vores projekt gjort brug af forskellige plugins f.eks. Spring Boot og MySQL. De plugins vi har brugt kan findes i pom.xml filen.

JDBC

Vi har brugt JDBC i vores projekt. Igennem vores DbManager har vi etableret forbindelsen til databasen med den funktion der hedder getConnection(), her får vi adgang til databasen, hvis username og password stemmer overens med hvad der står i application.properties.
Derudover bruger vi også JDBC i vores kode, for at lave en SQL forespørgsel, f.eks. i vores login metode i DbRepository hvor vi bruger “PreparedStatement” til at lave forespørgslen.
Udover at vi opretter SQL forespørgsler, så udfører vi dem også i vores applikation, med den der hedder executeQuery(). som returnerer et ResultSet objekt.
Til sidst håndterer vi resultaterne ved hjælp af rs.next() (rs er ResultSet). Dette bruger vi til at se om email og password matcher en bruger i vores database.

SQL 8.0.33

SQL er et sprog, som vi bruger i vores database MySQL Workbench, dette sprog har vi brugt til at oprette data i vores database, derudover har vi også brug af SQL til at oprette tabeller, som vi har kunne hente fra vores database ind i vores applikation. Vi har også med SQL fået vores tabeller til at have en relation til hinanden, ved hjælp af det som hedder foreign key og references. Dette har hjulpet os med at få de forskellige data til at passe korrekt sammen på tværs af tabeller.

HTML 5.0

I vores projekt har vi brugt HTML, som er et markup language. Vi har brugt HTML til at strukturere vores hjemmeside ved at hente data ind, ændre data og få vores applikation ud på en hjemmeside. Vores HTML filer bestemmer også, hvad der bliver vist ude på hjemmesiden.

Thymeleaf

I vores projekt har vi brugt Thymeleaf, som er en server side Java-template. Thymeleaf har hjulpet os med at hente vores data og få skrevet det direkte ud i HTML.

Tailwind 2.2.16

Tailwind er et CSS framework som vi har brugt til at style vores HTML, for at gøre vores applikation så brugervenlig som muligt. Dette er essentielt for brugeren.
Et brugervenligt system og godt design er ikke bare et plus, men en nødvendighed for at holde på applikationens brugere.
Tailwind er blot et af mange CSS frameworks vi kunne have benyttet os af, vi overvejede også Bootstrap, men hældede mere over til Tailwind.
Tailwind har hjulpet os bygge vores HTML sider op på en hurtig og flot måde, derudover har Tailwind også været en overskuelig måde at ændre i designet, da man blot tilføjer de forskellige klasser direkte ind i HTML. Tailwind hjælper os også med at holde applikationen responsiv, så brugeren kan tilgå hjemmesiden fra sin mobil eller tablet.
Vi har også gjort brug af komponenter til vores Header og Footer, som vi bruger på alle vores HTML sider, dette har gjort vores kode mere clean og overskuelig at kigge på.

CSS

Udover vi har brugt Tailwind til at style vores HTML sider, har vi også brugt CSS. Dette har vi udelukkende gjort, for at få mere luft i vores HTML sider. Dette kunne vi godt have undgået, hvis vi havde installeret Tailwind i vores projekt frem for at hente Tailwind ned på hvert HTML side fra et link. Dette vil vi 100% gøre til næste gang.

Derudover har vi valgt at lave en bedre brugeroplevelse, ved at tilføje flere temaer til vores applikation, dette har vi gjort ved hjælp af CSS og Javascript.
Vi har som vores lyse version af applikationen lavet baggrunden af projektet lys med sort skrift og ved vores mørke version af programmet har vi lavet en mørk baggrund med hvid skrift, Det giver brugeren mulighed for at skifte mellem dag/nat og giver en afslappende effekt på øjet ift. hvad man bedst kan lide.

Problemformulering

Vi har fået til opgave af Alpha Solutions at udvikle et Projektkalkulationsværktøj, der har til formål, at nedbryde projekter og kalkulere informationer til brugerne, som bliver gemt i en database og kan ændres af brugerne. Denne applikation skal indeholde diverse funktionaliteter for at optimere projekt arbejdet så meget som muligt. Derudover er en del af opgaven at etablere en datamodel for projekt og projektets opgaver, samt tidsforbrug og deadlines. Ud over det skal det være muligt for brugeren at oprette og vedligeholde projekter og deres opgaver, samt delopgaver. Der skal dertil laves en opsummering af tidsforbruget, så man kan få overblik over hvor lang tid et projekt samt opgaver og delopgaver skal tage og hvor lang tid projekter, opgaver og delopgaver rent faktisk tager.
Projektet skal kunne vise et Gantt diagram, for at give det mest optimale overblik over projektet.[2]

[2] Bilag 1: Projekt case

Projektafgrænsning

Vi har valgt at afgrænse vores projekt, for at sikre, at vi når det vi skal nå inden for tidsrammen af projektet.
Vi har valgt at afgrænse vores projekt til at fokusere på de vigtigste funktionaliteter, som omfatter at oprette et projekt, tilføje opgaver og delopgaver, fremkalde et gantt diagram og udvikle et brugervenligt design. Det er vigtigt, at de basale funktionaliteter på applikationen virker med 100% optimalitet.

Vores afgrænsning af projektet har givet os mulighed for at optimere kode og fejlfinde hurtigere end normalt. Dette giver brugeren i sidste ende en bedre brugeroplevelse, da tingene virker perfekte og håndtere fejl fra brugeren.
Vores afgrænsning af projektet er med til at sikre et bedre slutprodukt og derfor medvirke til en bedre brugeroplevelse. Dette er essentielt for brugeren, da man i fremtiden kan overveje at implementere flere funktionaliteter og derfor ikke er interesseret i teknisk gæld der kan komplicere udvikling. Vores afgrænsning af projektet er ikke kun til fordel for brugeren, men også til fordel for udviklingsteamet, da det sikre vores funktionaliteter er af en højere kvalitet, samt bedre tid til løsning af potentielle problemer.

Fremgangsmåde

I vores projekt har vi som fremgangsmåde anvendt den agile udviklingsmetode Scrum, der er en velkendt og udbredt indenfor software udvikling. Udviklingsmetoden Scrum er baseret på en udviklingsproces, hvor der i løbet af korte udviklings sprinter leveres værdi med kundens indblanding og interaktion. Vores Scrum baseret fremgangsmåde i projektet består af flere kerneelementer, der omfatter product backlog, sprint planlægning, daglige stand-up-møder, sprint reviews og Sprint retrospectives.

Som det første element startede vi ud med vores produkt backlog, hvor vi her opretter en prioritetsliste over krav og funktioner som det færdige system skulle indeholde. Vi laver her vores opstilling af user stories inden for de krav og rammer som blev fastlagt i samarbejde med systemets product owner, som i dette tilfælde er Alpha Solutions. Denne del af processen blev gennemgået med henblik på projektet som helhed inden for de planlagte tidsrammer, hvor også de nødvendige interessenter har været involverede.

Det andet vigtige element i vores projekt er sprint planlægning, hvor vi ved starten af hvert sprint har afholdt et sprint planlægningsmøde. Dette er foregået ved at vi her udvælger de elementer fra product backloggen, som skal implementeres i den kommende sprint. Elementerne som sprinten skal omfatte, vælger vi her ud fra de prioriteringer der er foretaget tidligere i product backloggen og sammen med product owner. Det blev i dette tilfælde valgt at vores sprinter skulle vare to uger, hvor vi planlagde at udvikle to user stories per sprint.

Efter sprint planlægningen var på plads, satte vi sprinten i gang og vi overgik her til daglige stand-up-møder. Dette foregik ved at vi hele gruppen mødtes om morgenen og gennemgik status på hvor langt vi var nået samt diskuterede eventuelle udfordringer. Vi oplevede her at de daglige stand-up-møder gjorde det nemt at koordinere de næste skridt i udviklingen, det gav også mulighed for vi fælles har

kunne holde et overblik på de opgaver der her skulle løses, for at vi kunne komme i mål med de forskellige sprinter.

Når så vi afsluttede en af vores to ugers sprinter, fremlagde vi det som var blevet opnået i løbet af sprinten for vores product owner, der i dette tilfælde er Alpha Solutions. Dette gav os her mulighed for at afholde et sprint review, hvor vi evaluerede den nuværende situation i forhold til resten af projektet. Vi oplevede her at sprint reviewet gav os mulighed for at tilpasse vores fremtidige sprinter og se på evt. ændringer, som vi oplevede blev brug for undervejs.

Efter vores sprint review blev færdigt og evt. tiltag var taget, afholder vi i gruppen et sprint retrospektivt møde, hvor vi satte fokus på gruppen og de arbejdsprocesser vi havde været igennem. Vi erfarede her bedre måder at arbejde på, ved at dele opgaverne anderledes op undervejs og skrive flere noter.

Ved brugen af Scrum i vores arbejdsproces har vi oplevet flere fordele i vores projekt. Dette har været i omfang af vi har oplevet det har været nemt og hurtigt at tilføje funktioner eller redigere i planen undervejs, der viste sig at være nødvendig. Vi oplevede at droppe at have en fungerende SCRUM master men nærmere forskellige ansvarsområder i de forskellige sprinter, hvilket fungerede ret godt, da vi oplevede der var mange forskellige slags opgaver. Væsentlige emner vi oplevede at komme ind på i vores retrospektive møder, var vores user stories vi fandt for avancerede og omfattende, dette førte derfor til at vi efter første sprint omlagde vores user stories og delte dem op i flere. Vi oplevede heraf det var nemmere at nedbryde vores user stories i opgaver og vi opnåede effektiviseret arbejdsproces.

Tiltag der fremover kunne foretages for forbedringer var at vi syntes, at sprinterne kunne have været effektive og ved ugentlige PO-møder i stedet for hver anden uge, der også kunne give en mere brugerbaseret udvikling.

Virksomhed

Interessentanalyse

En interessentanalyse handler om at finde frem til interessenter med indflydelse på projektet, for at kunne opnå et overblik over, hvilke forhold der er vigtige at tage hensyn til.

Analysen er god til at give et indblik i hvilke interessenter projektgruppen skal holde fokus på undervejs i projekt fremgangen. Den giver modsat også mulighed for at give overblik over interessenter med mindre indflydelse, der er af mindre betydning. Til udarbejdelsen af elementerne i analysen har vi anvendt følgende[3].

interessenter og kategorisering

- Ejer (Alpha Solutions)
- Ansatte
- Samarbejdspartnere
- Konkurrenter
- Kunder
- Projektgruppen

Til kategorisering af interessenterne har vi udarbejdet et diagram til kategoriseringer der fremgår af bilag[4].

Nøgleinteressenter

Ejer (Alpha Solutions)

- Optimere og forbedre projekthåndtering
- Tilfredshed blandt ansatte
- Brugervenligt og nemt

[3] 2. 2022-Zwisler, Projekt og analyseredskaber.pdf
[4] Bilag 3: kategorisering

- Fungere uden fejl eller tilfredsstillende
- Krav er blevet opfyldt
- Tidsplanen og budget overholdt

Ansatte

- Systemet er nemt at lærer og intuitivt
- Optimere og forbedre projekt håndteringen
- Er overskueligt og nemt at håndtere
- nyt og bedre

Projektgruppe

- Projektet opnår succes
- samarbejde i projektgruppen går godt
- Godt grundlag for systemudviklingen fra kunde
- Klar og tydelig kommunikation mellem alle parter
- Konstruktiv feedback undervejs

Interessentanalyse						
Interessenter	Deres mål	Tidligere reaktion	Hvad der kan forventes	Indvirkning pos/neg	Mulig fremtidig reaktion	Ideer
Ejer	Succes og vækst	Optimistisk over ny løsning med bedre behov tilpasning	Går op i at projektet opnår succes	Meget positiv hvis tingene går godt og meget negativ hvis	vil se frem til at skifte over til det nye system, hvis det lever op	involvere dem i udvikling en og fremskridt undervejs

				tingene går dårligt	til krav	
Ansatte	Bedre og hurtigere projektplanlægning	Ser frem til et mere tilpasset og nyere system	har ikke den store reaktion på udkom pga. indflydelse	har en positiv indvirkning med henblik på succes	at følge med det nye system, hvis det godkendes	indvoldvere deres holdninger og ideer undervejs da de er slutbrugerene
Projektgruppe	At levere et tilfredstillende produkt der lever op til kundes forventinger inden for ressourcerne	Positive over at komme i gang med det nye projekt	er meget positivt indstillet på at projektet opnår succes	har en meget positiv indvirkning på projektet	implementering af systemet, hvis kunden bliver tilfreds	At holde andre interessenter involveret undervejs

I interessentanalysen har vi fået identificeret interessenter og heraf udvalgt nøgleinteressenterne, som vi har foretaget en analyse af. Disse udvalgte nøgleinteressenter omfatter ejeren, ansatte og projektgruppen.

Der er her tale om ejeren har et ønske om at opnå forbedret projekthåndtering, så der blandt medarbejderne kan opnås et nyt og forbedret system. Ejeren som i dette tilfælde er Alpha Solutions har stor indflydelse under projektet, da det er ejeren der

fastsætter krav og forventninger, det er derfor vigtigt at Ejeren forbliver involveret under udviklingen og opdateret mest muligt og fremskridt undervejs.

Den anden nøgleinteressenter er de ansatte hos Alpha Solutions, der ligesom ejeren har et ønske om et nyt og forbedret system, der kan optimere fremtidig projekthåndtering. De ansatte er af mindre indflydelse end ejeren, men stadig en vigtig interessent, da de ender som slutbrugere og derfor også skal involveres i udviklingsprocessen.

En nøgleinteressent er også projektgruppen os selv, som har et mål om at levere et tilfredsstillende produkt, inden for de havende ressourcer der gør kunden Alpha Solution tilfreds. Som projektgruppen er vi positivt indstillet på opgaven og afgørende for det færdige system, da vi varetager udviklingen. Da udviklingen foregår i projektgruppen, er det vigtigt at projektgruppen forbliver involveret og velinformeret under hele udviklingen.

Samlet set kan det konkluderes at det er vigtigt at sørge for at alle interessenter forbliver informeret, via kommunikation og samarbejde, nogle interessenter forbliver dog vigtigere end andre. Men ved at ejerens og de ansattes mål og forventninger bliver mødt, om et nyt system samt projektgruppens mål om at levere et godt produkt, vil alle interessenter se sig tilfredsstillet.

Informationsanalyse

En informationsanalyse bliver brugt til at finde ud af om projektet bliver en fiasko eller succes. Derfor er det vigtigt at samle informationer om markedet for projektværktøjer og eventuelt få feedback fra brugere, for at forstå deres krav og behov.
Til udarbejdelsen af elementerne i analysen har vi anvendt følgende[5].

[5] 2. 2022-Zwisler, Projekt og analyseredskaber.pdf

En stor konkurrent til Project Peak er Projectmanager og Trello, det er 2 store virksomheder, som sidder på størstedelen af markedet.
Med Trello, som har over 50 millioner brugere, er det derfor en stor konkurrent.[6]
Ved projektets gang, er det vigtigt, hvilke informationer som er vigtige at give videre, derudover skal det også gøres klart, hvem der formidler. Hvordan informationerne bliver formidlet og hvem der skal modtage informationerne.
Overordnet set, skal vi finde ud af, hvilke informationer de forskellige interessenter har behov for.

Derfor har vi udarbejdet en informationskanal, som viser præcis hvad og hvordan. Informations tabellen er lavet ud fra projektets interessenter.

Informations tabel

Informationstabel					
Hvem	Hvad	Type	Hvornår	Hvordan	Ansvarlig
Ejer	Løbende status	Orientering	Ved større ændring eller milepæl	Møde eller mail	Projektgruppen
Ansatte	Løbende status	Orientering	Når projektet er færdigt/ eller ved milepæl	Nyhedsbrev eller mail	Ejer
Projektgruppen	Løbende status	Orientering	Hver gang en opgave afsluttes	Møde, mail eller telefon	Projektgruppen

[6] 1. DMR, 2022

Det kan konkluderes på informations tabellen, at det spiller en rolle for projektstatus og kommunikation for de relevante interessenter. Ved brug af tabellen fastlægger vi, hvem der skal have de forskellige informationer og hvordan det skal formidles.
Der bliver illustreret hvor vigtigt ejeren og projektgruppens formidling af informationer er med til at øge fremskridt og status i projektet, begge interessenter har til ansvar at formidle og holde de ansatte opdateret om projektets fremgang. Selvom Project Peak træder ind på et marked med store konkurrenter, viser tabellen, at informationer til de ansatte vil spille en stor rolle, især hvis de specifikke krav og behov bliver opfyldt. Hvis Project Peak skal blive en succes, skal informationerne formidles korrekt, så vi kan opnå og fremme succesen.

Risikoanalyse

En risikoanalyse bruges til at identificere og vurdere potentielle risici eller udfordringer, der kan forekomme i forbindelse med et projekt eller en virksomhed. formålet er her at forstå og forebygge de risici som potentielt kunne have en negativ indvirkning på projektet eller virksomhedens mål, der kan deraf laves en plan for håndtere disse risici. Til udarbejdelsen af elementerne i analysen har vi anvendt følgende[7].

De potentielle risici opstilles i en risiko-tabel,[8] hvor man kan udregne et produkt ud fra sandsynligheden og konsekvens. Produktet kan bruges til at vurdere, om risikomomentet skal forebygges eller om det ikke har betydelig indflydelse. Er produktet af betydelig størrelse og har indflydelse, kan en udvidet risikoanalyse udarbejdes hvor præventive tiltag og løsningsforslag kan blive opsat.

[7] 2. 2022-Zwisler, Projekt og analyseredskaber.pdf
[8] 3. 2022-Lassen, Projektledelse 2. udgave (risikoanalyse, kommunikation).pdf

Risikoanalyse			
Risikomoment	Sandsynlighed	Konsekvens	Produkt
Fatale fejlestimater (deadline bliver overskrevet)	1	10	10
Gruppemedlem forlader projekt	1	3	3
Utilstrækkelige tests	2	7	14
Problemer med deployment tools (Render)	3	7	21
Tab af data eller information	1	7	7
Store ændringer i systemkrav	1	10	10
Kollektiv sygdom (alle bliver syge på samme tid)	1	10	10
Mangelfuld afgrænsnings af projekt	3	3	9

Udvidet risikoanalyse							
Risikomomen	Sandsynlighe	Konsekven	Produkt	Præventive	Ansvarli	Løsningsforsl	Ansvarlig

t	d	s		tiltag	g	ag	
Fatale fejlestimater (deadline bliver overskrevet)	1	10	10	Projektgruppen holder overblik over fremgang og der undervejs bliver holdt daily stand up meetings.	Projektgruppen	Fremgangsmåder genovervejes og tidsplan ændres.	Projektgruppen
Utilstrækkelig tests	2	7	14	Der udpeges en testansvarlig med overblik.	Testansvarlig	Der vil blive lagt mere tid til at lave flere tests.	Projektgruppen
Problemer med deployment tools (Render)	3	7	21	Tidlig opstilling og testning af værktøjer	Projektgruppen	Ekstern vejledning	Projektgruppen
Store ændringer i systemkrav	1	10	10	udvikling ud fra systemkrav (Agilt)	Projektgruppen	Tilpasning af ændringer	Projektgruppen
Kollektiv sygdom (alle bliver syge på samme tid)	1	10	10	Holde hinanden opdateret, arbejde hjemmefra. f.eks. discord og github project (Scrumboard)	Projektgruppen	kommunikere online	Projektgruppen

Feasibility Study

Feasibility study er et godt redskab til at udarbejde en analyse for om dit projekt bliver et “go” eller “no-go”. Ved at udarbejde denne analyse forholdsvis tidligt i projektprocessen kan man hurtigt skabe sig et overblik over sandsynligheden for at projektet skulle gå hen og fejle eller blive en succes. Da man godt argumentere for at tid er lig med økonomi, kan det godt konkluderes, at det ville være en fordel for projektgruppen/virksomheden, at få lavet en feasibility study analyse tidligt i projektprocessen for at undgå tab. For at konkludere om projektet bliver et “go” eller “no-go”, kigger man på styrker og svagheder omkring projektet. Vi har derfor valgt at kigge på markedet omkring andre projektkalkulationssystemer, det økonomiske aspekt, samt de tekniske forhold for projektet.

Indledende analyse

Projektet som har fået navnet “Projekt Peak” og er et projektkalkulationssystem der bliver udviklet til Alpha Solutions. Alpha Solutions er en forholdsvis ung forrentning der havde sin opstart i 2004. Det er et digitalt rådgivningsfirma, der bygger digitale løsninger på et teknisk stærkt fundament. Med 80+ mennesker i 3 forskellige lande kan planlægningen af et projekt godt gå hen og blive relativt kompliceret. Som nævnt tidligere er tid en del af det økonomiske aspekt og hvis et projekt ikke bliver planlagt struktureret og overskueligt, kan det ende med ændringer i tidsplanen og det er som regel ikke er lig med økonomisk vækst for en forrentning. Alpha Solutions ønsker derfor et system (Projekt Peak) der kan nedbryde og dokumentere hele projektprocessen så alle kan få et klart overblik og bedre overskue projektet og dets tidsmæssige omfang. Projekt Peak ville derfor på sigt kunne medføre øget effektivitet og mindre fejl i projektstyringen, dette vil betyde at der vil være mulighed for at mindske tidsspild i projektarbejde og dermed reducere mulig negativ vækst for Alpha Solutions.

Markedsanalyse

At en virksomhed er konkurrencedygtig på markedet i forhold til sine konkurrenter er blevet et meget essentielt aspekt og det er noget alle forretninger fokuserer meget på. Derfor er det vigtigt at forbedre sig som virksomhed ned til de mindste detaljer. Strukturering, dokumentation og estimering af tid er meget vigtigt for at en virksomhed kan konkurrerer på markedet. Derfor vil et projektkalkulationssystem være et godt løsningsforslag for en virksomhed, der agerer inden for projekthåndtering. Der findes allerede forskellige løsninger på et projektkalkulationssystem ude på nettet som man kan benytte sig af. Bla finder der: https://www.projectmanager.com/ og https://www.smartsheet.com/ som er nogle løsninger der allerede løser det problem som Alpha Solutions står overfor. Ved at Alpha Solutions vælger at få deres eget system, kan det tilrettelægges præcis efter deres behov og de skal ikke være afhængige af et udefrakommende system.
De andre systemer på markedet kan hurtigt gå hen og blive store og virkelig avanceret, hvilket kan blive en distraktion i forhold til et system som “bare” skal give medarbejderne et overblik over tidsplanen for et projekt.

Derfor kan det konkluderes at Projekt Peak systemet vil have en positiv indflydelse på Alpha Solutions, da de vil kunne modificere systemet helt efter deres behov og ønsker.

Det økonomiske aspekt

For at vurdere Projekt Peaks økonomiske aspekt, skal vi kigge på omkostningerne af udviklingen af systemet, samt indtjeningsmuligheder og besparelser som projektet kan medføre.

Omkostningerne ved udviklingen af systemet vil primært være lønninger til udviklerne og nødvendige softwareværktøjer, derudover ville det være nødvendigt at ansætte en designer til projektet for at få den optimale brugeroplevelse.

Besparelser for Alpha Solutions gennem dette projekt, vil primært være bundet til at forbedre effektiviteten i projektstyringen, det kunne f.eks. være at reducere fejl og tidsspild. Dette vil gøre projektet af Projekt Peak gå væsentlig hurtigere og resultere i store besparelser i lønningerne.

De tekniske forhold

For at vurdere Projekt Peak tekniske forhold skal vi tage hensyn til forskellige faktorer, f.eks. sikkerhed, teknologier, vedligeholdelse og systemkrav.
Da Projekt Peak kommer til at indeholde følsomme oplysninger for brugerne, vil sikkerhed derfor være en vigtig faktor og vil kræve stabile sikkerhedsprotokoller for at beskytte brugerens data.

Vedligeholdelse og opdateringer af systemet skal der også tages højde for, så det opfylder virksomhedens krav over tid og fungerer optimalt. Dette kræver fejlrettelser og regelmæssige opdateringer.

De juridiske forhold

Ved udviklingen og implementeringen af projektet “projekt peak” er det vigtigt at der bliver taget højde for de juridiske forhold, for at sikre gældende regler, love samt reguleringer bliver overholdt. Der er her tale om aspekter som databeskyttelse, personlige oplysninger, ophavsret, intellektuelle ejendomsret, kontraktlige forhold, juridiske krav osv.

Da Projekt Peak vil have at gøre med data vedr. oplysninger om burgere og projekter, er det afgørende at sørger for databeskyttelsesregler bliver overholdt. Der er her tale om databeskyttelsesregler som GDPR, der omfatter retningslinjer for datahåndteringen, det er derfor en nødvendighed at der implementeres de passende sikkerhedsforanstaltninger i overensstemmelse med lovgivningen.
Et andet vigtigt aspekt under udviklingen af Projekt peak er at sikre, der ikke undervejs bliver begået krænkelser af ophavsretten eller den intellektuelle ejendomsret, der kan forekomme i form af licensaftaler, open source-komponenter

eller immaterielle rettigheder. Det er derfor vigtigt, at der foretages en grundig undersøgelse af disse, da det i værste fald kan føre til potentielle retssager og konflikter.

Ved udviklingen og implementeringen er det også vigtigt, at de kontraktlige forhold er i orden, så alle interessenters interesser bliver opfyldt. Det er i dette tilfælde særligt vigtigt for Alpha Solutions, så de får defineret parters ansvar og forpligtelser. Dette gør sig ikke kun gældende mellem parterne, men også mellem ansatte i virksomheden.

Konklusion på feasibility Study

Ud fra den foretagede feasibility study analyse af Project Peak projektet der skal leveres til virksomheden Alpha Solutions, kan følgende konklusion drages.

Projektet Project Peak er realistisk samt levedygtigt og ville kunne bidrage med betydelige forbedringer til Alpha Solutions nuværende og fremtidige situation. Dette betyder at det kan ses som et gavnligt initiativ at få udviklet et nyt præferencebaseret projekt kalkulationssystem, hvor Alpha Solutions kan opnå en forbedret situation. Ved et nyt skræddersyet projekt kalkulationssystem vil Alpha Solutions kunne opnå forbedringer i form af en øget effektivitet, for mindsket tidsspild, en reduktion i fejl og et lavere antal fejl estimater. En fordel ville også være en mindre afhængighed af eksterne leverandører og systemer, der ville også her være mulige besparinger i form af mindre licensbetalinger til eksterne leverandører. Selvom der også vil opstå omkostninger i omfang af udviklingen af systemet ville det kunne forventes at det på sigt ville resultere i positiv vækst, da besparelserne og projektstyringen ville kunne opveje omkostningerne. Det er vigtigt at der inden og undervejs i udviklingen bliver lagt fokus på sikkerhed, opdateringer, vedligeholdelse og evt. juridiske krav, da dette er en nødvendighed for at sikre en succesfuld implementering og realisering af Project Peak. Det kan derfor konkluderes at Project Peak ville være et godt tiltag for Alpha Solutions og dermed et go, det ville dog være nødvendigt at foretage undersøgelser og forberedelser for at opnå succes.

Requirements

For at identificere og danne en forståelse for de krav Alpha Solutions har til det ønskede projekt kalkulationsværktøj, er det nødvendigt at opstille alle krav. Dette er nødvendigt for at skabe et ordentligt grundlag for hele udviklingen af vores Project peak system.

Krav 001	Der skal etableres en brugerflade, hvori en bruger kan integrere med brugervenlige forudsætninger.
Krav 002	Der skal kunne oprettes en bruger og mulighed for at ændre i dataen om brugeren.
Krav 003	Kunne oprette et nyt projekt med tilhørende informationer om projektet (projektnavn, projektbeskrivelse, startdato, slutdato)
Krav 004	Kunne oprette opgaver der er tilknyttet et projekt samt ændre i de informationer der gives ved oprettelse (task navn, task beskrivelse, startdato, slutdato og status.
Krav 005	Der skal kunnes oprettes subtasks som er tilknyttet tasks med task navn, task beskrivelse, startdato, slutdato og status. Der skal være mulighed for ændringer og sletning
Krav 006	Der skal implementeres tid kalkulationer for oprettede projekter og dets indhold så man kan opnå summeringer af tidsforbrug.
Krav 007	Der skal være mulighed for at se sammenligninger af det estimerede tidsforbrug og det reelle tidsforbrug ved projektafslutningen.
Krav 008	Projekter skal kunne visualiseres via et gantt diagram der kan fremvise tid horisonten og planlagte tasks.
Krav 009	Projekter skal kunne deles med andre oprettede brugere, så flere brugere får mulighed for at tilgå det samme projekt.

User stories

User story 1

- Som en system bruger.
- vil jeg kunne oprette og tilgå projekter og opgaver på en brugervenlig måde via en webside med mit eget login.
- så jeg nemt kan administrere og holde styr på mine projekter.

Acceptkriterier

1. **Tilgå web application med login**
    - Bruger skal kunne tilgå web application
    - Skal ske ved hjælp af login
2. **Man skal kunne oprette en bruger:**
    - Design og implementer en brugerflade til at oprette et projekt.
    - Implementer funktionalitet til at gemme det oprettede projekt i systemet.
3. **Man skal kunne oprette en projekt:**
    - Design og implementer en brugerflade til at oprette et projekt.
    - Implementer funktionalitet til at gemme det oprettede projekt i systemet.
4. **Visning af projektet:**
    - Design og implementer en brugerflade til at vise projektets detaljer.
    - Vis projektets navn, beskrivelse og andre relevante oplysninger.

User story 2

- Som en system bruger.
- vil jeg kunne tilføje opgaver og delopgaver, hvis behovet opstår med navn og beskrivelse.
- så projektet kan brydes ned i flere dele.

Acceptkriterier

1. **Tilføjelse af opgaver til projektet:**
    - Design og implementer en brugerflade til at tilføje opgaver til projektet.

    - Implementer funktionalitet til at gemme opgaverne og associere dem med det relevante projekt.
2. **Tilføjelse af delopgaver til opgaverne:**
    - Design og implementer en brugerflade til at tilføje delopgaver til opgaverne.
    - Implementer funktionalitet til at gemme delopgaverne og associere dem med den tilsvarende opgave.

User story 3

- Som en system bruger.
- vil jeg kunne tilpasse og redigere i mine projekter, opgaver og delopgaver hvis behovet opstår også på en brugervenlig måde.
- så jeg nemt kan administrere og holde styr på mine projekter, hvis ændringer skal foretages.

Acceptkriterier

1. **Opdatere/redigere i projektet samt i projektets opgaver/delopgaver:**
    - Design og implementer redigeringsfunktioner i brugerfladen for projekter, opgaver og delopgaver.
    - Implementer funktionalitet til at opdatere og gemme ændringer i systemet.
2. **Mulighed for at slette opgaver, delopgaver samt hele projektet:**
    - Design og implementer en funktion i brugerfladen til at slette opgaver, delopgaver og projekter som helhed.
    - Implementer funktionalitet til at fjerne de valgte elementer fra systemet.

User story 4

- Som systembruger
- vil jeg kunne se tidsestimater og en oversigt over tidsforbruget på projekter og opgaver med opsatte tidsrammer inde for projekt rammerne.
- så jeg kan vurdere om projektet er på rette spor og eventuelt justere planen, hvis det er nødvendigt.

Acceptkriterier

1. **Visning af projektets varighed:**
    - Design og implementer en funktion til at beregne og vise, hvor mange dage projektet forventes at vare fra startdato til slutdato.
    - vise den beregnede varighed på brugerfladen.
2. **Visning af dage til projektstart:**
    - Design og implementer en funktion til at beregne og vise, hvor mange dage der er tilbage indtil projektet starter fra den aktuelle dato.
    - Vis antallet af dage til projektstart på brugerfladen.
3. **Visning af dage tilbage i projektet siden startdato:**
    - Design og implementer en funktion til at beregne og vise, hvor mange dage der er gået siden projektets startdato.
    - Vis antallet af dage tilbage i projektet siden startdatoen på brugerfladen.
4. **Oversigt over tidsforbrug ved afslutning af projektet:**
    - Design og implementer en funktion til at vise en oversigt over det faktiske tidsforbrug for projektet, herunder opgaver og delopgaver, når projektet er fuldført.
    - Vis det faktiske tidsforbrug på brugerfladen.
5. **Kontrol af subtask-slutdatoer:**
    - Design og implementer en funktionalitet, der sikrer, at slutdatoen for en delopgave ikke overskrider slutdatoen for den overordnede opgave (task).
    - Vis en fejlmeddelelse eller forhindrer brugeren i at indstille en slutdato for subtask, der overstiger task-slutdatoen.
6. **Kontrol af opgaveslutdatoer i forhold til projektet:**
    - Design og implementer en funktionalitet, der sikrer, at slutdatoen for en opgave ikke overstiger slutdatoen for projektet.
    - Vis en fejlmeddelelse eller forhindrer brugeren i at indstille en slutdato for opgaven, der overstiger projektets slutdato.
7. **Automatisk tilpasning af opgaver og delopgaver ved ændringer i projektets**
    - Implementer en funktion, der automatisk justerer slutdatoen for opgaver og delopgaver, når der foretages ændringer i projektets slutdato.
    - opdater brugerfladen for at vise de ændrede slutdatoer for opgaver og delopgaver.

User Story 5

- Som bruger systembruger
- vil jeg kunne visualisere projektets fremgang på et Gantt-diagram.
- Så jeg vil have mulighed for visuelt at se, hvordan tidsplanen skrider frem og sikre at projektet overholder tidsplaner.

Acceptkriterier

1. **Tilføjelse af et Gantt-diagram for projektet:**
    - Design og implementer en komponent eller brugerflade, der viser et Gantt-diagram.
    - Integrer diagrammet i projektvisningen.
2. **Visning af tasks på Gantt-diagrammet:**
    - Design og implementer en funktion til at vise opgaver på Gantt-diagrammet.
    - Vis navn, startdato og slutdato for hver opgave.
3. **Dynamisk opdatering af Gantt-diagrammet:**
    - Implementer funktionalitet, der automatisk opdaterer Gantt-diagrammet, når der foretages ændringer i projektets tidsplan eller opgave.
    - Sørg for, at ændringerne i diagrammet afspejler de opdaterede oplysninger.

User Story 6

- Som systembruger
- vil jeg kunne dele mine projekter med andre oprettede brugere, jeg skal arbejde sammen med.
- så andre kan tilgå mine projekter og foretage ændringer.

Acceptkriterier

1. **Tilføjelse af medarbejder-tilordning på projekter:**
    - Design og implementer en brugerflade eller funktionalitet til at tildele medarbejdere til oprettede projekter.
    - Gør det muligt for brugeren at vælge medarbejdere fra en liste eller søge efter medarbejdere.
2. **Redigering af medarbejder-tilordninger:**

    - Implementer funktionalitet til at redigere eksisterende medarbejder-tilordninger på projekter.
    - Gør det muligt for brugeren at ændre medarbejdernes tilordninger eller fjerne dem.
3. **begrænsning:**
    - medarbejdere skal være oprettede før de kan tilføjes og man kan ikke tilføjes flere gange på engang.
4. **Redigerings muligheder**
    - Tilføjede medarbejdere skal kunne redigere og tilføje tasks på lige fod med owner brugeren

FURPS+

Baseret på de modtagede materialer med kravspecifikationer og product owner meetings, har vi udarbejdet følgende kravspecifikationer. Kravspecifikationerne dækker over, hvad det endelig program skal kunne leve op til ud fra kundens praksis, som et fungerende projekt kalkulationsværktøj, med henblik på aspekterne i vores projektafgrænsning.

Functional

- Det skal kunne lade sig gøre at oprette, tilgå, ændre og slette en bruger med tilhørende login funktioner, hvor en bruger har sin egen unikke login.
- Det skal være muligt for brugeren at kunne oprette, tilgå, ændre og slette projekter med tilhørende opgaver/delopgaver og subtasks/underopgaver, der ligeledes skal kunne oprettes, tilgås, ændres og slettes.
- Der skal være implementeret tidsdimensioner for projekter, opgaver og delopgaver, hvilket omfatter estimeret tidsforbrug og deadlines, der skal kunne give et overblik over fordelingen af tidsforbrug.
- Det skal være muligt at visualisere projektet i et GANTT diagram, projektet og dets indhold kan fremvises for et overblik.
- Medarbejder skal kunne tilføjes til projekt så også de kan tilgå projekter oprettet eksternt

- Ved afsluttningen af et projekt skal der være mulighed for at frembringe en projektrapport, der skal indeholde detaljer for det afsluttede projekt i form af tidsforbrug og tidsestimering.

Usability

- Der skal udarbejdes en brugervenlig og intuitiv GUI, som skal kunne understøtte de funktionelle behov, så alle funktioner er tilgængelige.
- Systemet skal kunne køres og åbnes i en browser ud fra en https adresse, der skal kunne tilgås på tværs af computere så brugere kan få adgang til projekt kalkulationsværktøjet.

Reliability

- Projekter samt dets indhold skal kunne gemmes og tilgås i en database så de er persistent for brugeren.
- Der skal være Error handling, som kan håndtere eventuelle fejl der kan opstå f.eks i forbindelse med ukorrekt brugerinput.

Performance

- Systemet skal kunne understøtte håndteringen af projekter, så der er mulighed for at gøre brug af web application samt tilknyttet database inden for rimelig svartid for at sikre en positiv responsiv oplevelse.
- For større databaseopstillinger og datasæt, kan responstiden blive reduceret og komplekse metoder der opererer i databasen bør derfor holdes på rimeligt niveau, for unødig belastning tid bliver begrænset.
- Systemet skal være skalerbart, så der er mulighed for opstart af nye projekter og oprettelse af nye brugere, uden en betydelig nedgang i ydeevne.

Supportability

- Til understøttelse af systemet er det nødvendigt med omfattende dokumentation, der dækker brugervejledning, fejlfinding og softwaren.
- Disse dokumenter skal være tilgængelige og opdateret ud fra systemets nuværende funktionalitet og egenskaber.

- Til softwaren i systemet er det vigtigt at det er designet med vedligeholdelsesvenlig kode, som overholder standarder og basale praksisser for lettere fremtidige opdateringer eller ændringer.
- Brugerfeedback er også noget der kan være med til at forbedre systemet og der kan aktivt samles data, der kan hjælpe med overvejelser til forbedringer.

Design

Database Design

Domænemodel

Domænemodellen er med til at give et struktureret overblik over vores softwareapplikationer. Formålet med domænemodel er at få en klar forståelse af domænet, elementerne og de indbyrdes relationer. Det handler både mellem os som gruppe medlemmer men også mellem projektgruppen og PO (product owner). Vi lavede diagrammet som noget af det første for netop at danne et hurtigt overblik over hvordan vores kalkulationsværktøj skulle bygges op. Når modellen blev lavet som det første, havde vi som gruppe en bedre forståelse af domænet samt et solidt grundlag for udviklingen af vores system Projekt Peak. Domæne modellen fremgår af bilag[9].

ER diagram

I vores ER diagram visualisere vi de logiske forhold mellem tabellerne i databasen, her får vi et bedre overblik over hvordan tabellerne er relateret til hinanden.
ER diagram fremgår af dette bilag[10].

SQL Script

Vi har i vores SQL script lavet 8 tables, user, project, projectmember, task, subtask, doneproject, donetask og donesubtask, disse tabeller har en reference til hinanden og derfor har været muligt at tilføje en delopgave til en opgave som ligger i et projekt.
SQL Script fremgår af dette bilag[11].

[9] Bilag 8: Domænemodel
[10] Bilag 9: ER diagram
[11] Bilag 10: SQL script

User Interface Design

Golden Rules

I udarbejdelsen af vores projektdesign har vi taget hensyn og anvendt til “Golden Rules”, disse overordnede retningslinjer har været en forudsætning for at vi har kunne skabe en bedre brugeroplevelse. Design eksempler fremgår af bilag[12].

Den første regel vi har fokuseret på er at forenkle vores brugergrænseflade mest muligt, ved at begrænse unødige funktioner og undgå for meget kompleksitet med simple og korte beskrivelser af de forskellige elementer. Den anden regel vi har fulgt er at skabe en konsistent oplevelse af systemet, hvilket vi har gjort ved at være konsistente i vores anvendelse af de elementer der indgår i vores design. Der er i designet anvendt de samme typografier, farver og symboler på tværs af systemet, hvor knapper som edit, delete og show deler hver deres unikke symbol. En tredje regel der er blevet anvendt er sørg for let navigation, hvor der er sat fokus på at vores navigationselementer som “gå tilbage” og “vis” er placeret nemme og tilgængelige steder, hvor de er nemme og lette at bruge. Et fokusområde har også været at give brugeren feedback, dette er f.eks. hvis brugeren begår fejlindtastninger under login eller de ikke kan få forbindelse til serveren. Fejl begået i systemet bliver håndteret af fejlmeddelelser, så brugeren let og hurtigt kan komme videre, dette betyder et mere brugervenligt system, hvor brugerens behov for tilbagemeldinger bliver opfyldt.

Heuristikker

I frembringelsen af designet til Project Peak, har det været vigtigt undervejs at implementere de 10 Heuristikker for at skabe det bedst mulige systemdesign. I dette omfang har det vigtige været at overveje systemets brugervenlighed, anvendelse og at det fungere intuitivt. Design eksempler fremgår af bilag[13].

[12] Bilag 2: design eksempler
[13] Bilag 2: design eksempler

1. **Konsistens:** Som det første har vi lagt vægt på konsistens og ensartethed, hvor vi gennem systemet gør brug af ensartet symbol og farve elementer på tværs af brugerfladen for at skabe en sammenhængende brugeroplevelse.
2. **Brugerfeedback:** I systemet bliver der gjort brug af brugerfeedback, i form af fejlmeddelelser opstår dette kan f.eks. være når forbindelsen mistes eller når mail eller password indtastes forkert.
3. **Informationsarkitektur:** informationer er opstillet ensformigt og organiseret logisk, så kun brugbare informationer er synlige, i det kortest læselige format.
4. **Let læselig:** Længere tekst er undgået og nogle ord er erstattet af symboler som f.eks. show, edit osv. så læsbarheden er nem og hurtig.
5. **Navigering:** For at navigationen er overskuelig og nem, er navigationens ens på alle sider og der er oftest flere genveje, heriblandt har vi ensartet "back"-knapper på alle sider, der returnerer brugen til den forrige side.
6. **Tilpasning og fleksibilitet:** Brugeren har ved anvendelse af systemet mulighed for undervejs at tilpasse samt oprette flere projekter, opgaver og delopgaver, derudover har brugerne også mulighed for at tilpasse sine login oplysninger, hvis mail, navn eller password skal ændres.
7. **Fejlhåndtering:** Hvis fejl undervejs opstår for brugeren, vil der fremkomme en fejlmeddelelse, hvor muligheden for at returnere vil forekomme hvis muligt.
8. **hierarki:** Vi har her gjort brug af visuelle elementer til at fremhæve indhold for at danne en bedre forståelse for brugeren. Dette er gjort med elementer som farve, størrelse og placering, et eksempel på dette er vores logo, der er placeret fremtrædende oppe i venstre hjørne.

Programdesign

Systemarkitektur

System arkitekturen er en vigtig del af softwareapplikationen og fungerer som fundamentet for opstillingen af den overordnede struktur, dette omfatter komponenterne i systemet samt de opbyggede relationer gennem systemet. Der vil af dette afsnit fremgå system arkitekturen i Project Peak der er bygget op med

formålet at kunne imødekomme projekt-kalkulation og planlægning. System arkitekturen fremgår af bilag[14].

I vores system er pakke strukturen organiseret så komponenter og funktioner er delt op i forskellige grupper med hver deres ansvarsområder. I systemet indgår alle pakker i den centrale pakke Projectpeak1, som indeholder primære funktioner. Under den centrale pakke hører så yderligere opdelte under pakker, der hver har deres mere specifikke ansvar og rolle i systemet. Der er her tale om controller, dto, entities, repository, service og utility pakkerne.

Controlleren i systemet indeholder klasser der har med at håndtere brugerinteraktion og data der skal præsenteres. Dto pakken i systemet indeholder objekter, der har med at overføre data mellem komponenter i systemet og de forskellige grænseflader. Entities pakken indeholder klasser der definere centrale objekter, hvor attributter også bliver tilkendegjort. Repository pakken har i systemet at gøre med at styre persistering og data hentning fra databasen. Service pakken omfatter serviceklasser, som står for at håndtere kommunikation mellem controller klasser og repository klasser. Til sidst er der utility pakken som indeholder hjælpefunktioner til at understøtte komponenter der er her tale om login exceptions.

Klassediagram

Som en oversigt over vores system opsætning har vi udarbejdet et klassediagram, der portrætterer alle klasserne i vores system, fremgår af bilag[15]. Dette omfatter alle klasser og attributter i henholdsvis DTO klasser og Entity klasser, hvor andre elementer er udeblevet for ikke at overkomplicere diagrammet og skabe et overblik.

Vi har på den baggrund også udarbejdet to yderligere klasse diagrammer som underbygger systemet for at skabe en bedre forståelse. Den første af dem er en simplificeret version af det første klasse diagram som viser klassernes generelle forhold mellem sig i et mere overskueligt aspekt og det andet er lavet på samme

[14] Bilag 4: Systemarkitektur
[15] Bilag 5: fuld klassediagram til overblik

simple niveau, men med user klasser indsat som eksempel, dette fremgår af følgende bilag[16].

Flow

Aktivitetsdiagram

Vi har udarbejdet et aktivitets diagram over flowet fra useren, logger ind eller opretter sig som bruger i applikationen, samt opretter et nyt projekt. Vi har netop valgt dette flow fordi det er ret essentielt da det dækker over hovedformålet med applikationen til Alpha Solutions. At kunne oprette sig som bruger → logge ind med login oplysninger → og til sidst kunne oprette et projekt det vil blive gemt i databasen. Applikationen indeholder selvfølgelig en lang række andre funktioner og metoder der kan benyttes, men for at aktivitetsdiagrammet ikke bliver for stort, har vi kun valgt at vise dette flow. Flowet viser, hvad der sker efter hver interaktion, useren har med systemet. Dette giver også et indblik i hvordan MVC "Model-View-Controller" foregår fra brugerens side. Useren interagere med view-delen, herefter sender controller-delen requestet videre til model-delen der håndtere requestet og sender evt data tilbage til controller-delen der returnere et nyt view til useren. Alt dette vil blive fordybet senere i rapporten. Aktivitetsdiagram fremgår af bilag[17].

Sekvensdiagram

Vi har valgt at opstille nogle sekvens diagrammer, der viser flowet fra at requested kommer til controlleren og videre ind i systemet. Disse fremgår af følgende tre bilag[18] [19] [20]. Vi har netop valgt at lave sekvens diagrammer ud fra disse metoder i controlleren fordi de er ret essentielle i applikationen. Vi har valgt at stoppe ved IProjectRepository da den enten går ind i ProjectRepository_DB eller

[16] Bilag 6: simpelt klassediagram og simpelt klassediagram med User klasser som eksempel
[17] Bilag 11: Aktivitetsdiagram
[18] Bilag 15: Sekvensdiagram updateProject
[19] Bilag 15: Sekvensdiagram showProject
[20] Bilag 16: Sekvensdiagram doneProject

ProjectRepository_STUB alt efter om der er valgt en stub eller database implementation. Nogle af metoderne nogle kalkulationer og beregninger i service-laget, som vi godt kunne tænke os at uddybe. Vi har valgt at forklare om service-laget her, fordi vi synes det var ret nærliggende, da sekvens diagrammerne viser flowet og køre igennem service-laget. Vi har kun valgt at forklare og uddybe en af de tre sekvensdiagrammer, da det er det flow vi mener er mest spændende.

Sekvensdiagram der omhandler updateProject, som er en PostMapping, fra ProjectControlleren kan ses i bilag[21].
Det kan ses i sekvensdiagrammet at strukturen følger MVC struktur pattern hvor ProjectControlleren kun snakker med ProjectService og den snakker kun med IProjectRepository og omvendt. Det som sker i koden mellem ProjectService og IProjectRepository kan ses i denne kode snippet der fremgår af bilag[22].

Det der sker er, at updateProject metoden henter det gamle projekt fra IProjectRepostitory og instantiere et nyt Projekt objekt. Derefter går den ind og tjekker at hvis det originale projekts start dato ikke er lig med det redigerede projekts start dato. Eller hvis det originale projekts slutdato ikke er lig med det redigerede projekts slutdato. Så er der sket en ændring i datoerne. Det betyder at den først skal opdatere selve projektet. Dernæst skal den opdatere task og subtask, der hører til projektet. Det er en metode der også ligger i ProjectService som den kalder og sender det nye redigerede projekt og det originale projekt med som parametre. Denne metode kan ses i bilag[23].
Metoden opdaterer de task- og subtasks datoer, der hører til projektet. Den går først ind og laver en beregning på hvor mange dage der er mellem det gamle projekt startdato og det nye projekt startdato. Det samme gælder for slutdatoen. Så går den ind og tjekker at hvis slutdatoen er lig med nul, så skal slutdatoen sættes til det den samme difference som startdatoen har. Det betyder at det kun er startdatoen der er blevet ændret. Så vi ville sikre os at hvis startdatoen på projektet blev ændret to dage frem, ville task og subtasks startdatoer OG slutdatoer også rykke to dage frem.

[21] Bilag 15: Sekvensdiagram updateProject
[22] Bilag 12: kode snippet af updateProject metode
[23] Bilag 13: kode snippet af updateTaskAndSubtaskDates

Hvis det kun er slutdatoen for projektet der er ændret, sætter vi begge difference dage til nul, fordi vi kom frem til at det ikke vil ændre i task og subtask start tidsplan.
Vi har med denne service metode sikret os at når projektets datoer ændrer sig ville de tilhørende task og subtask reguleres efter hvor mange dage projektet er rykket så man ikke skal ind manuelt, hvilket vil tage lang tid hvis man har mange task og subtask.

Implementering

I dette afsnit, der omhandler implementering, vil vi komme omkring de design patterns og arkitekturprincipper vi har benyttet i vores system. For at systemet bliver fleksibelt, vedligeholdbart og skalerbart, er det vigtigt at koden er velfungerende og velstruktureret, og det gøres ved brug af en masse principper og patterns som vi vil komme mere ind på i dette afsnit.

ProgramStruktur

Vores programstruktur er overordnet bygget op omkring den arkitektoniske tilgang som kaldes MVC(Model-View-Controller). Vi vil fordybe hvad MVC er senere i rapporten. Vores hierarki består af Controller klasser som kun snakker med de tilhørende service klasser. Service Klasserne snakker kun med de tilhørende repositorier der har til formål at modellere og håndtere de forskellige model klasser op mod enten en database eller en stub implementering. Overblik over hvordan hierarkiet i vores system ser ud fremgår af bilag[24].

Controller

Overordnet set er controlleren det lag som tager sig af de requests der kommer fra viewet, som altså er de HTML sider som brugeren bliver præsenteret for på vores website. Efter modtagelsen af requestet så sender controlleren det videre til servicelaget.

Service

Servicelaget er det lag som håndterer forretningslogikken i vores applikation. Da controlleren ikke skal have ansvaret for at beregne på data for simplificering, så sender den kun requestet afsted til service som sender requestet videre til repository. Servicelaget modtager dataen fra repository, laver evt beregninger på dataen og sender det tilbage til controller som viser det i viewet på websitet. Man kan sige at servicelaget er en slags mellemmand mellem controlleren og repository.

[24] Bilag 14: System hierarki

Repository

Repository er det lag som tager sig af database relateret operationer. Altså modtager et request fra serviceklassen, interagerer med databasen og sender dataen tilbage til service. I forhold til de repositories vi har i vores system har vi valgt at implementere interfaces. Som det kan ses i vores diagram fra tidligere bilag[25], så snakker servicelaget med et IRepository som er et interface. Dette interface bliver implementeret af to klasser under sig. Det ene er Repository_DB der i alle metoder implementerer metoden DbManager.getConnection fra klassen DbManager, som udelukkende sørger for forbindelse til databasen. Den anden klasse der implemeterer et interface er Repostitory_STUB, der kan bruges til et kontrolleret og isoleret testmiljø uden man er afhængig hele database miljøet.

Model (entities)

Modellen definerer strukturen og repræsentationen af dataene i en applikation. I vores tilfælde er det klasser der indeholder attributter, constructor, gettere og settere der primært bliver håndteret og instantieret i repository-laget.

Patterns

Model-View-Controller

MVC som står for Model-View-Controller, er et software arkitekturpattern benyttes til at organisere og strukturere softwaresystemer. Hovedformålet med dette er at adskille de forskellige komponenter, der er i et system. MVC deler systemet op i tre hoveddele: Model, View og Controller. De tre dele får hver deres ansvar, der gør at koden bliver let at vedligeholde. Hele vores applikation er netop bygget op omkring MVC strukturen.

Vi har i vores applikation brugt Spring Web der fungerer som en dependency i systemet. Spring Web er en del af String Framework der giver nogle omfattende og

[25] Bilag 14: System hierarki

robuste værktøjer til at udvikle webapplikationer ved hjælp af MVC arkitekturpattern. Spring kommer med en række funktioner og metoder der kan håndtere webanmodninger, samt give et svar tilbage igen. Som sagt er MVC delt op i tre hoveddele, vi vil beskrive med eksempler fra vores kode i de næste afsnit.

Model

Model-delen er den, der har ansvaret for data og logikken i applikationen. I vores system har vi flere dataobjekter/klasser såsom Project, Task, Subtask, User, i model-delen der kan instantieres og modelleres ved brugerinput der kommer fra controlleren. dette vil altså sige at der kommer et request fra controller-delen som sender det videre til model-delen der har ansvaret for at hente, manipulere eller gemme den data der bliver sendt med. I vores tilfælde består model-delen af et servicelag, der modtager en request og sender den videre til et repository også tilbage igen. Overordnet set kan man sige at model-delen omfatter den data der skal modelleres eller håndteres og sendes videre til view-delen der beskrives i næste afsnit.

View

View-delen er den brugergrænseflade, som brugeren bliver præsenteret for med den data, der kommer fra model-delen. Vores brugergrænseflade er en masse HTML sider, brugeren bliver præsenteret for. Når vi modtager data fra controlleren, der indeholder modeldata, bruger vi Thymeleaf til at håndtere dataen, så det kan vises for brugeren på brugergrænsefladen. Thymeleaf er en teknologi vi bruger til visning af data som beskrevet tidligere i rapporten.

Controller

Controller-delen i MVC fungerer som bindeleddet mellem model-delen og view-delen. En slags leder, der styrer informationer/data mellem de to ansvarsområder. Controlleren modtager typisk et request fra brugeren i view-delen. Herefter opdatere den dataen i model-delen baseret på det request brugeren har sendt afsted fra brugergrænsefladen. Efter controlleren har opdateret modellen, kan

controlleren også modificeres, så den opdaterer det view brugeren bliver præsenteret for.[26]

GRASP[27]

Vi har i opsætningen af vores system benyttet os af GRASP principperne til at skrive vores kode efter. GRASP står for “General Responsibility Assignment Software Principles”. Det er en samling af designprincipper og retningslinjer der hjælper med at tildele ansvar for klasserne og objekter inden for objektorienteret programmering. Som softwareudviklere vil vi gerne stræbe efter GRASP principperne da det vil skabe en mere vedligeholdelsesvenlig og forståelig kode til de næste softwareudviklere der skal håndtere den kode der er blevet skrevet.
Vi har kun valgt at uddybe de vigtigste af GRASP principperne vi gør mest brug af, da der er mange omfattende principper og vi gerne vil kunne gå i dybden med de vigtigste for vores projekt.

Controller (GRASP)

Controller princippet i GRASP handler om at tildele det ansvar der kommer mellem inputtet fra brugerinteraktionerne (view-delen) og selve applikationen (model-delen). Forespørgslerne der kommer fra view-delen har controlleren ansvaret for hvad der skal ske med denne forespørgsel. Den kan evt. kalde relevante metoder eller funktioner for at opdatere data i applikationen. Herefter kan controlleren give en opdateret visning for brugerne i view-delen.

Dette controller princip bruger vi i vores system. For at gøre kode mere overskuelig, har vi valgt at dele controlleren op i fem forskellige dele. Vi har LoginController, ProjectController, TaskController, SubtaskController og UserController der hver især har ansvaret for nogle primære ting som kan antydes ud fra navnet på controlleren.
Vi har valgt at tage vores index metode med fra ProjectControlleren for at illustrere dette controller princip. Udfor endpoint navngivningen kan det siges at denne

[26]Rafael D. Hernandez, 2021

[27]Muhammad Umair, 2018

metode har til formål at vise forsiden der indeholder brugerens projekter. index Metode fremgår af bilag[28].

Når en bruger logger ind på vores system, bliver brugeres userid tildelt en HTTP Session. Den tildelte HTTP session vil være aktiv 15 min hvorefter brugeren bliver logget ud af systemet. For at være sikker på at der er en bruger logget ind i systemet, starter vi i næsten alle metoder med at hente det userid der er tildelt httpSession. Derefter bruger vi et if statement til at tjekke om userid == 0. Hvis det er 0, er HTTP Session udløbet og brugeren vil blive henvist til login webside i view-delen. Hvis brugeren stadig er logget ind, springer den videre i metoden.
I view-delen har vi en header der kræver informationer om useren, der er logget ind. Derfor beder metoden projectService klassen om at hente dataen om useren med det userId vi hentede igennem HTTP Session og instantierer en ny user. Vi opretter derefter en model.addAttribte med den user der er hentet og sender den ud til view-delen.
Det samme princip sker med projekterne. Metoden beder projectService om at hente de projekter der er tilknyttet useren og sender dem ud til view-delen.
Alt i alt er det vigtigt at controlleren ikke laver noget arbejde selv men uddelegerer arbejdet. En rigtig ledertype.

High Cohesion (GRASP)

High Cohesion er endnu et princip, der fremgår i GRASP. Som en softwareudvikler der går op i sin kodekvalitet er High Cohesion et princip man stræber meget efter. Jo højere, jo bedre. Jo højere High Cohesion man har, jo mere har klasserne i systemet et klart og afgrænset ansvarsområde. Man sigter mod at klasserne kun er ansvarlig for at udføre en specifik opgave, hvilket gør dem mere fokuserede og gør at ansvaret ikke bliver blandet på kryds og tværs, hvilket vil kunne fremstå uoverskueligt hvis koden skal rettes. Jo mere afgrænset ansvaret bliver, jo stærkere bliver den interne sammenhæng i klasserne.

For at opnå High Cohesion i et system kan man bruge SRP princippet: (Single

[28] Bilag 18: kode snippet af getMapping index metode

Responsibility Principle) som er en del af SOLID-principperne. SRP er et grundlæggende princip inden for objektorienteret design og softwarearkitektur. Princippet handler om at opdele koden i mindre og mere specialiseret dele, hvor de enkelte dele har et klart afgrænset ansvar. Dette vil resultere i et system der opnår en High Cohesion og Low Coupling.

Det er vigtigt at skabe High Cohesion i et system, da det vil gøre koden mere struktureret, forståelig og vedligeholdelsesvenlig, hvilket er det overordnede hovedformål med GRASP. For at skabe High Cohesion i vores system har vi netop valgt at dele ansvaret op i flere klasser efter hvilket ansvar de skal have.

Det kan ses på de to billedeksempler der fremgår af dette bilag[29], at vi har valgt at dele controlleren op i fem dele. Hvis man samlede alle metoder og funktioner i én klasse, vil det hurtigt blive en lang og meget uoverskuelig kode samt det vil gøre at koden får en low cohesion, hvilket vil gå imod princippet med high cohesion. Vi har derfor valgt at dele controlleren op så de fem controllere hver har et ansvarsområde der fokusere på bestemte dele af systemet. Når en request kommer fra view-delen ved den nøjagtig, hvilken controller den skal til da endpointet er specificeret. Hvis det f.eks. er ProjectController der får et request skal den kun koncentrere sig om at sende requestet videre til ProjectService klassen da det er den eneste klasse den kender til gennem en Dependency injection i ProjectController klassen.
Hvis vi eller en anden softwareudvikler i fremtiden skal ind og rette i koden og ved at fejlen han skal rette, omhandler projektdelen i systemet. Så kan han ved denne opdeling hurtigt danne sig et overblik og gå ind i konkrete klasser der har ansvaret for netop projektdelen. Det kan derfor nemt afgøres at jo mere High Cohesion man har i sit system, jo gladere softwareudviklere kan man få i fremtiden da personerne ikke skal ind i en uoverskuelig lang kode og lede efter en bestemt metode.

Low Coupling (GRASP)

Det næste princip i GRASP som vi har haft meget i tankerne er Low Coupling. Det læner sig meget op af High Cohesion. Det er dog nogle forskellige principper, de

[29] Bilag 19: Billedeksempler controller og service

prøver at opnå. I praksis arbejder High Cohesion og Low Coupling næsten som hånd i hånd. Man plejer at sige at ved at skabe “High Cohesion” opnår man Low Coupling, da klasser der har klare ansvarsområder naturligt reducerer deres afhængighed mellem de andre klasser. Omvendt kan man sige at ved en Low Coupling opnår man High Cohesion da klasser der har en lav kobling og ikke er afhængige af hinanden, nemt kan udskiftes eller ændres uden det vil påvirke resten af systemet.
Som sagt så handler Low Coupling om at minimere afhængigheden mellem klasserne i applikationen. Dvs. at klasserne har minimal eller ingen viden om hinandens interne metoder eller lign. For at opnå en lav kobling i systemet kan man bruge det princip der hedder “Dependency Injection”. Dependency injection er et design pattern indenfor softwareudvikling der hjælper med at reducere afhængigheden og netop skaber denne lave kobling mellem klasserne. Det handler om at flytte ansvaret, så klassen ikke selv skal lave operationerne, men laver en injektion af en anden klasse og flytter ansvaret dertil. Dette gøres ved at injicere et objekt af klassen i den pågældende konstruktør.
Det har vi også valgt at implementere i vores system. Som det kan ses i kode der fremgår af dette bilag[30].

Her har vi i ProjectService klasse brug for at få fat i projectRepository da vi skal videregive det request der kommer fra ProjectControlleren og muligvis sende dataen tilbage igen til controlleren. Derfor har vi injiceret en afhængighed af klassen IProjectRepository i ProjectService klassen igennem konstruktøren. Dette gør vi i stedet for at oprette en instans af IProjectRepository da det vil give højere kobling mellem de to klasser.

Polymorphism (GRASP)

I forhold til GRASP er polymorphism et af de vigtigste principper der bruges til at opnå Low Coupling og High Cohesion. Polymorphism er et koncept indenfor objektorienteret programmering, hvor det er muligt for en klasse at implementere en abstraktion af en anden klasse, da man gerne vil genbruge de samme metoder. Med at implementere et evt interface øger det genbrugeligheden i koden og det mindsker

[30] Bilag 20: kode snippet af ProjectService

mængden af redundant kode i vores system. Klasserne kan dog stadig opføre sig på forskellige måder, men kan stadig håndteres ensartet igennem abstraktionen.

For at bruge princippet polymorphism og opnå denne Lov Coupling og High Cohesion kan man f.eks. bruge interfaces. Interfaces er med til at spille en central og vigtig rolle når man skal opnå lav kobling i systemet. Interfaces definerer en form for kontrakt, der omfatter nogle metoder som klassen kan benytte, hvis den vælger at "underskrive" og implementere et bestemt interface. Dette medfører en lav kobling da klassen kun skal kende til interfacet men ikke de klasser som implementere interfacet. Det muliggøre genbrugelighed og gør det nemmere at udskifte/redigere i koden uden man skal ind i fem forskellige steder og rette i koden. Et interface kan også hjælpe med at gøre koden let at teste, da man nemt kan foretage en såkaldt stub implementering, hvilket vi har valgt at gøre i vores system.

Som det kan ses i vores repository package, der fremgår af følgende bilag[31] har vi fem interfaces som matcher det antal controllere og services. Som tidligere sagt har vi valgt at implementere et stub repository for hver af vores repositories. Repository_DB og repostitory_STUB skal have præcis de samme metoder og funktioner. Derfor er det meget nærliggende at lave en abstraktion her. Vi valgte at lave et interface der bliver implementeret i henholdsvis repository_DB og repository_STUB, så når requestet kommer fra service laget, går det direkte til IRepository som har ansvaret for enten at gå ind i database-delen eller stub-delen. I stedet for at service-laget skal tage ansvar for hvilket repository det skal bruge, er ansvaret givet videre til IRepository som her tager ansvaret. Ved at implementere interfacet kan vi også sikre os at de to repositories har de samme metoder, hvilket er vigtigt for vores system. Som sagt er repository_stub et testmiljø der skal håndtere og reagere på samme måde som hvis man brugte database-delen.

[31] Bilag 21: Repository package

Kørselsvejledning

Use case diagram

Til at fremskaffe et overblik over vores use cases har vi udarbejdet et usecase diagram, denne frem viser brugeren og de forskellige use cases der bliver fremlagt længere nede i kørselsvejledningen. Use case diagrammet fremgår af bilag[32].

Use cases

use case 1: oprettelse af bruger

Aktør: systembruger

1. web application åbnes og bruger lander på login in front site
2. Brugeren trykker på "Sign up" -knappen.
3. Brugeren indtaster de nødvendige oplysninger, der bliver bedt om som full name, e-mail og password som skal gentages to ens gange.
4. Brugeren bekræfter oprettelsen af brugeren.
5. Systemet opretter den nye bruger og returnerer nu til login front site igen.
6. Brugeren kan nu logge ind med den oprettede bruger.

use case 2: oprettelse af projekt

Aktør: systembruger

1. Brugeren af systemet er logget ind i systemet og landet på hovedsiden.
2. Brugeren trykker på "Create project" knappen.

[32] Bilag 6: Use case diagram

3. Brugeren indtaster de nødvendige oplysninger, der bliver bedt om til oprettelsen af projektet som project name, description, start date og end date.
4. Brugeren bekræfter oprettelsen af projektet.
5. Systemet opretter det nye projekt og returnerer nu til projekt front site.
6. Brugeren kan nu se det nye projekt på projekt front site.

use case 3: oprettelse af task

Aktør: systembruger

1. Brugeren af systemet er logget ind i systemet og navigere ind på et allerede oprettet projekt.
2. Brugeren trykker på "add task" knappen.
3. Brugeren indtaster de nødvendige oplysninger, der bliver bedt om til oprettelsen af task som task name, description, start date og end date.
4. Brugeren bekræfter oprettelsen af en task.
5. Systemet opretter den nye task og returnerer nu til siden for projektet.
6. Brugeren kan nu se den nye task på projektets forside.

use case 4: oprettelse af subtask

Aktør: systembruger

1. Brugeren af systemet er logget ind i systemet og navigere ind på en task i et allerede oprettet projekt.
2. Brugeren trykker på "add subtask" knappen.
3. Brugeren indtaster de nødvendige oplysninger, der bliver bedt om til oprettelsen af subtask som subtaskname, description, start date og end date.
4. Brugeren bekræfter oprettelsen af en subtask.
5. Systemet opretter den nye subtask og returnerer nu til siden for projektet.
6. Brugeren kan nu se den nye subtask ved at klikke ind på subtasks for den valgte task.

use case 5: Se gantt-diagram for projekt

Aktør: systembruger

1. Brugeren af systemet er logget ind i systemet og navigere ind på et allerede oprettet projekt.
2. Brugeren trykker på "show gantt diagram" knappen.
3. Systemet fremkalder et Gantt diagram, der indeholder projektets tasks med en tidslinje.
4. Brugeren har nu mulighed for at få et overblik over projektets fremgang visuelt.

use case 6: deling af projekt med andre brugere

Aktør: systembruger

1. Brugeren af systemet er logget ind i systemet og navigere ind på et allerede oprettet projekt.
2. Brugeren kan her indtaste en e-mail adresse på en anden systembruger, som også er oprettet i systemet.
3. Brugeren trykker heraf på "share project" knappen.
4. Anden systembruger hvis email er indtastet får nu adgang til projektet ved login.
5. Projektet kan nu tilgås af flere brugere, som sammen kan foretage ændringer.

Software forudsætninger

Som forudsætning for at applikation skal kunne tilgås gør følgende kriterier sig gældende, dette er med udgangspunkt i at webapplication og SQL databaser er hostet via render.

1. **Operativsystem:** Som den første forudsætning for at kunne køre applikationen er at ens enhed anvender et operativsystem, der er en understøttet version fra en godkendt udbyder. Dette omfatter blandt andet macOS, Linux eller Windows.

2. **internetforbindelser:** For at systemet kan tilgås er det også vigtigt med en stabil internetforbindelse, så brugeren kan interagere med systemet hurtigst muligt og opnå best mulig bruger oplevelse.

3. **webbrowser installeret:** En nødvendighed er også at brugeren skal have en moderne og opdateret webbrowser installeret på sin enhed. Dette omfatter blandt andet browsere som Firefox, Microsoft Edge eller Safari.

Github link: https://github.com/Atributties/ProjectPeak.git
Running version link i cloud: https://projectpeak.onrender.com

Test Cases

Vi har i dette afsnit udviklet en lang række test cases, med formål at verificere og evaluere Project peak systemets funktioner og funktionaliteter. De udarbejdede test cases er lavet ud fra specifikke scenarier, der vil kunne opstå i forbindelse med anvendelse af systemet, baseret på pålagte krav. vores test cases inkluderer vigtige elementer som precondition, test steps, test data, forventet resultat, postcondition og et resultat baseret på en succes factor.

ved hjælp af test cases kan vi undervejs identificere fejl så der hele tiden kan ændres eller rettes i systemet undervejs, så man sikre overensstemmelse med de indgåede krav. Der er ved hjælp af test cases mulighed for en mere struktureret tilgang til at kunne udføre black box testing, hvor systemet kan evalueres ud fra en brugers perspektiv.
På grund af vores stub implementering, har vi haft mulighed for at indsætte test repositorier, der har gjort det muligt at injicere test data uden om databasen så database connection kan undgås ved testning. Vi har fravalgt Unit testing og valgt

kun at fokusere på system- og integrationstest, da vi ikke har set det nødvendigt pga. kriterierne for logikken i systemet ikke er så høje.

Nedenfor præsenteres de udarbejdede testcases, der skal sikre en bred testning af systemet i bredest muligt omfang, hvor krav mest muligt er efterkommet.

TEST CASE ID	TEST SCENARIE	TEST CASE	PRE CONDITION	TEST STEPS	TEST DATA	FORVENTET RESULTAT	POST CONDITION	RESULTAT
TC001	godkender bruger login	indsættelse af valid mail og password	1. Oprettet bruger 2. forbindelse til database	1. mail indsættes	“John.doe@example.com”	bruger sendes videre til user Frontend	bruger sendes videre til user Frontend	Succes
				2. password indsættes	“password1”			
				3. der klikkes login				
TC001	godkender bruger login	indsættelse af ikke valid mail og valid password	1. Oprettet bruger 2. forbindelse til database	1. mail indsættes	“forkertmail@example.com”	Returnere fejl besked om forkert mail eller password	Returnere fejl besked om forkert mail eller password	Succes
				2. password indsættes	“password1”			
				3. der klikkes login				
TC001	godkender bruger login	Indsættelse af valid mail og ikke valid password	1. Oprettet bruger 2. forbindelse til database	1. mail indsættes	“John.doe@example.com”	Returnere fejl besked om forkert mail eller password	Returnere fejl besked om forkert mail eller password	Succes
				2. password indsættes	"Forkert Password"			
				3. der klikkes login				
TC002	Oprettelse af et nyt projekt	Projekt oprettes med Name, Description, start date og end date.	1. indlogget bruger 2. forbindelse til database	1. Tryk create project		opretter nyt projekt hvor projekt detaljer fremkommer	opretter nyt projekt hvor projekt detaljer fremkommer	Succes
				2. valgte projekt detaljer indsættes	"project Name" “Description to project” “23/05/2023” “23/06/2023”			
				3. der trykkes				

				opret projekt				
TC002	Oprettelse af et nyt projekt	Projekt oprettes uden name men med description, start date og end date.	1. indlogget bruger 2. forbindelse til database	1. Tryk create project		Returnere fejl besked om felt skal udfyldes	Returnere fejl besked om felt skal udfyldes	Succes
				2. valgte projekt detaljer indsættes	"Description to project" "23/05/2023" "23/06/2023"			
				3. der trykkes opret projekt				
TC002	Oprettelse af et nyt projekt	Projekt oprettes uden description men med name, start date og end date.	1. indlogget bruger 2. forbindelse til database	1. Tryk create project		Returnere fejl besked om felt skal udfyldes	Returnere fejl besked om felt skal udfyldes	Succes
				2. valgte projekt detaljer indsættes	"project Name" "23/05/2023" "23/06/2023"			
				3. der trykkes opret projekt				
TC003	Redigering af projekt	Projekt redigeres med Projekt name, description, start date og end date ændres.	1. indlogget bruger 2. oprettet projekt 3. forbindelse til database	1. tryk edit project		projekt detaljer ændres til tilsvarende ændringer	projekt detaljer ændres til tilsvarende ændringer	Succes
				2. der indsættes nyt navn, description	"edited project name" "edited description" "25/05/2023" "25/06/2023"			
				3. trykkes update project				
TC004	Oprettelse af ny task	Task oprettes med name, description, start date og end date	1. indlogget bruger 2. oprettet projekt 3. forbindelse til database	1. tryk create task		der bliver oprette en task som vises på projektsiden	der bliver oprette en task som vises i projektet	Succes
				2. der indsættes task name, task description, start date og end date.	"task name" "task description" "25/05/2023" "30/05/2023"			
				3. det trykkes create task				
TC005	redigering af task	task name, task description, start date og end date	1. indlogget bruger 2. oprettet projekt og task	1. tryk edit task		task detaljer ændres til tilsvarende	task detaljer ændres til tilsvarende	Succes

		ændres.	3. forbindelse til database	2. der nyt task name, task description, start date og end date.	"edited task name" "edited task description" "26/05/2023" "31/06/2023"	ændringer	ændringer	
				3. der trykkes edit task				
TC006	oprettelse af ny subtask	Subtask oprettes med name, description, start date og end date	1. indlogget bruger 2. oprettet projekt og task 3. forbindelse til database	1. tryk create subtask		der bliver oprette en ny subtask som vises på projektsiden under task	der bliver oprette en ny subtask som vises på projektsiden under task	Succes
				2. der indsættes subtask name, subtask description, start date og end date.	"Subtask name" "subtask description" "26/05/2023" "28/05/2023"			
				3. det trykkes create task				
TC007	redigering af subtask	subtask name, subtask description, start date og end date ændres.	1. indlogget bruger 2. oprettet projekt, task og subtask 3. forbindelse til database	1. tryk edit subtask		subtask detaljer ændres til tilsvarende ændringer	subtask detaljer ændres til tilsvarende ændringer	Succes
				2. der indsættes nyt subtask name, subtask description, start date og end date.	"Subtask name" "subtask description" "27/05/2023" "29/05/2023"			
				3. der trykkes edit subtask				
TC008	Sletning af bruger, hvor også subtasks, tasks og subtasks slettes.	Sletning af bruger, hvor også subtasks, tasks og subtasks slettes.	1. indlogget bruger 2. oprettet projekt, task og subtask 3. forbindelse til database	1. tryk edit user		oprettede bruger slettes fra database med oprettede projekter, tasks og subtasks	oprettede bruger slettes fra database med oprettede projekter, tasks og subtasks	Succes
				2. det trykkes delete user				
				3. der bekræftes at bruger skal slettes				

Konklusion

Som en opsummering kan det konkluderes, at vi har udviklet og implementeret et succesfuldt system, ved navn “Project Peak”, der har som formål at optimere kalkulationen af et projekt så projekt håndteringen for Alpha Solutions kan blive forbedret. Vi har i projektet med succes opnået at leve op til pålagte forventninger, trods tilpasninger undervejs i fremgangsprocessen, der blandt andet gjorde vi måtte omlægge vores userstories i starten men i sidste ende betød vi fandt frem til en løsning.

Vi kan derudover også konkludere at vi gennem vores Feasibility Study, fandt frem til at det ville kunne gavne Alpha Solutions at gøre brug Project Peak frem for andre projektværktøjer, da systemet er tilpasset efter deres behov og forventninger. Det kunne derfor konkluderes ud fra vores Feasibility Study, at Project Peak vil være en succes for Alpha Solutions og dermed et “go” for os ved projektstart.

Til at starte med kom vi omkring interessenterne og risikoanalysen, først for at finde frem til hvem der har indflydelse på projektet og for at forstå hvilke risici og udfordringer projektet kunne møde i den kommende tid projektet er i gang, det kan her konkluderes, at vi stødte ind i en a risiciene fra risikoanalysen, dette var Mangelfuld afgrænsnings af projekt, da vi oplevede ikke at afgrænse vores userstories korrekt.

Gennem samtale med PO og udvikling af diagrammer kom vi frem til hvilke krav i form af funktionaliteter, Projekt Peak som minimum skulle have. Dette blev også hovedfaktoren for vores projektafgrænsning.

Samlet set har vi i vores projekt opnået et tilfredsstillende resultat, der opfylder de krav og forventninger, som var til et projekt kalkulationsværktøj tilpasset af Alpha Solutions. Leveret med fokus på en pålidelig og brugervenlig applikation, hvor kodekvaliteten har været i centrum.

Bilag

Bilag 1: Projekt case

CASE FOR 2. SEMESTER STUDERENDE

Projektkalkulationsværktøj

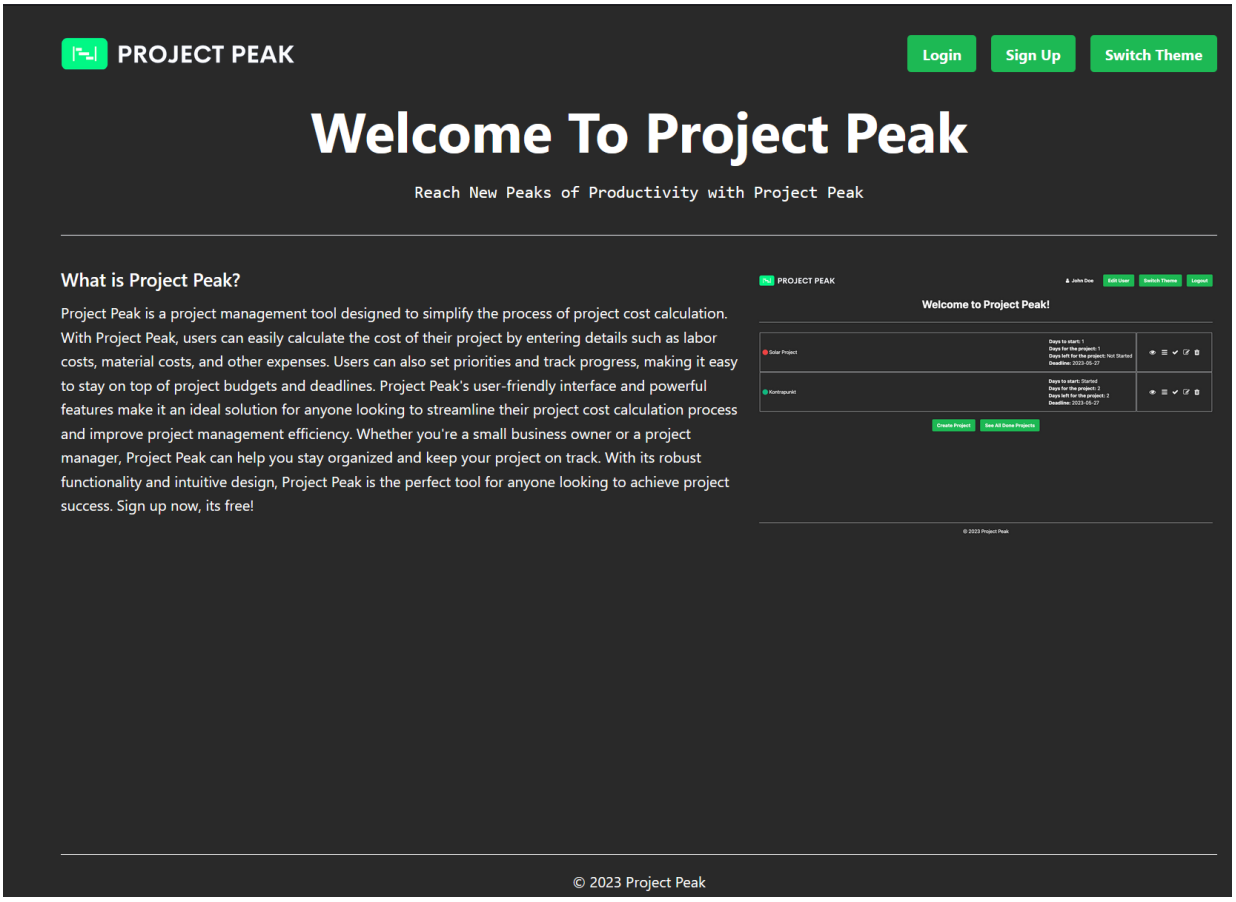
Mulighed for at nedbryde et projekt og få hjælp til kalkulation af nogle informationer

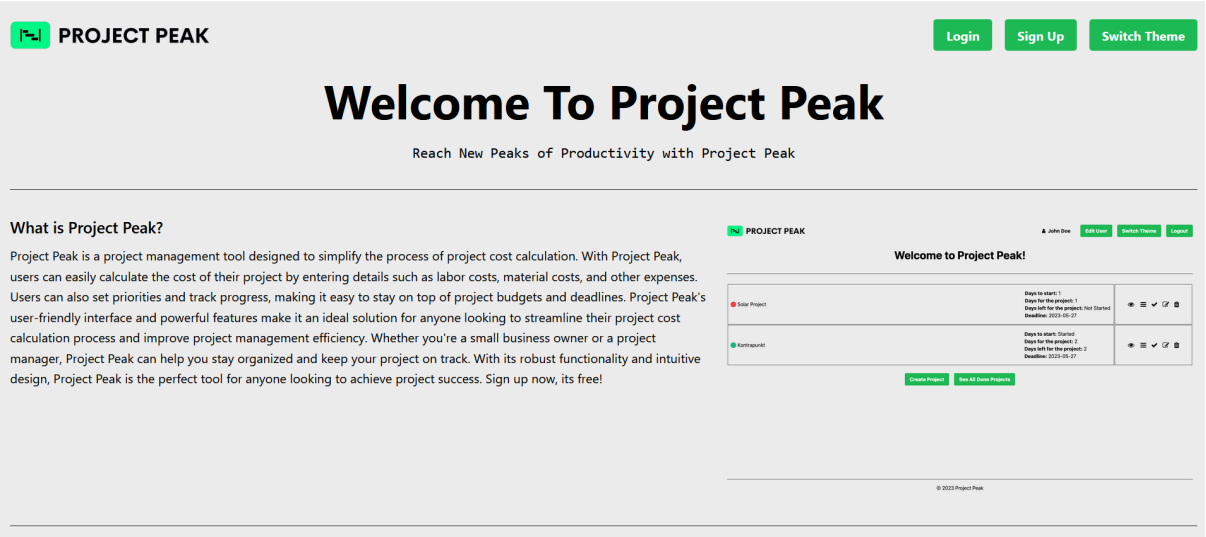
- Etablere datamodel for projekt- og tidsdimensioner (projekter, delprojekter, opgaver, tidsforbrug, deadlines mv)
- Lave en brugergrænseflade til oprettelse og vedligeholdelse af et projekt
- Summering af tidsforbrug, så man kan få overblik over tidsforbrug på projekter og delprojekter mv
- Fordeling af tidsforbrug på arbejdsdage, så man ved hvor mange timer der skal arbejdes hver dag for at projektet bliver færdigt til tiden?

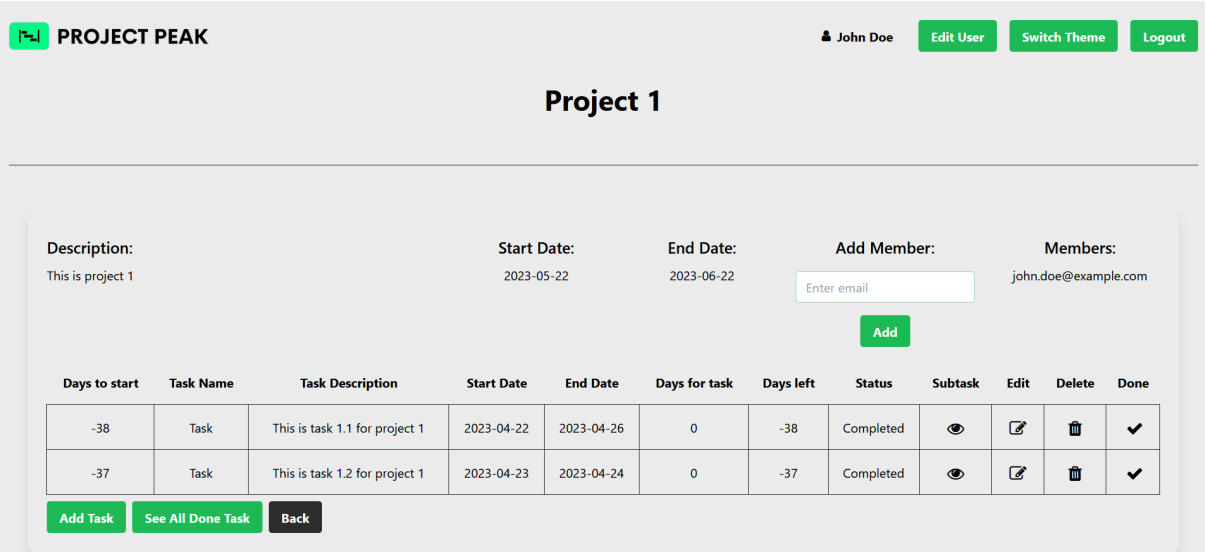
For de ambitiøse og erfarne

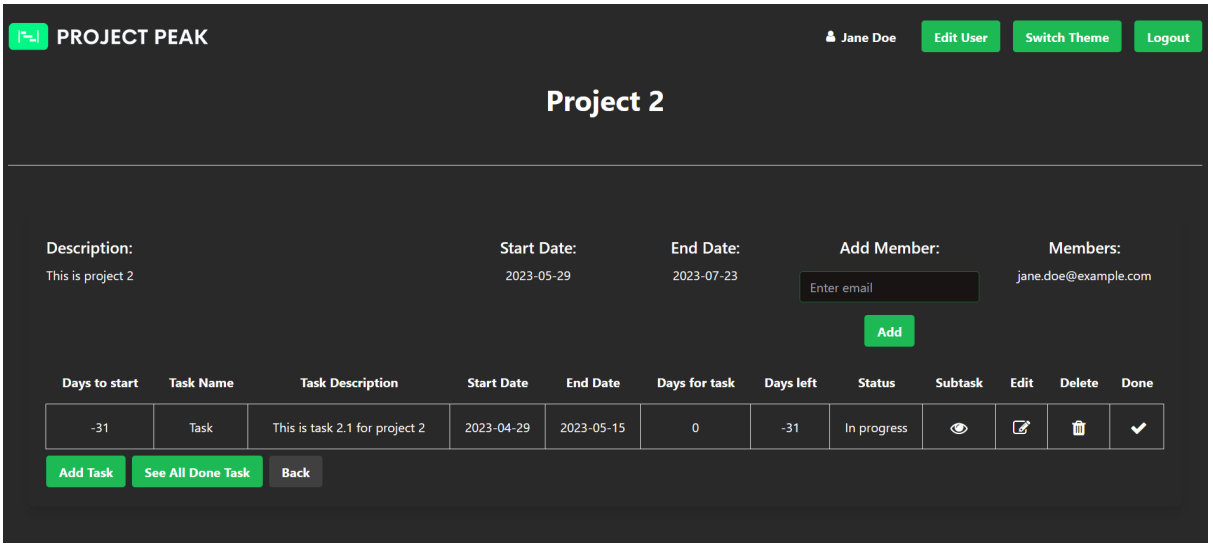
- Visualisering af projekt på et GANNT diagram
- Introducere kompetencer/ressourcetyper og give overblik over hvor mange timer hver kompetencer/ressourcetype bruger hver dag
- Introducere konkrete ressourcer (med nogle bestemte kompetencer) og finde ud af hvor hårdt belastet hver ressource er på hver arbejdsdag

Bilag 2: design eksempler

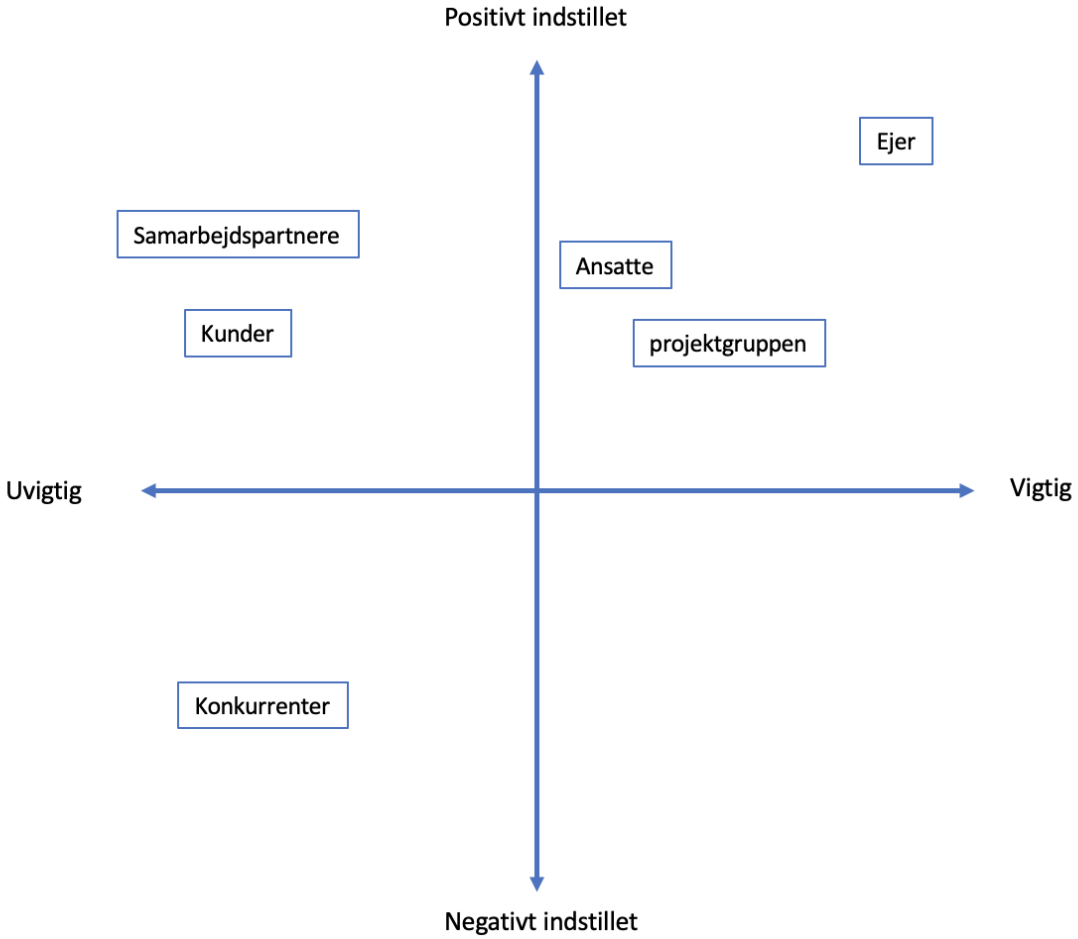




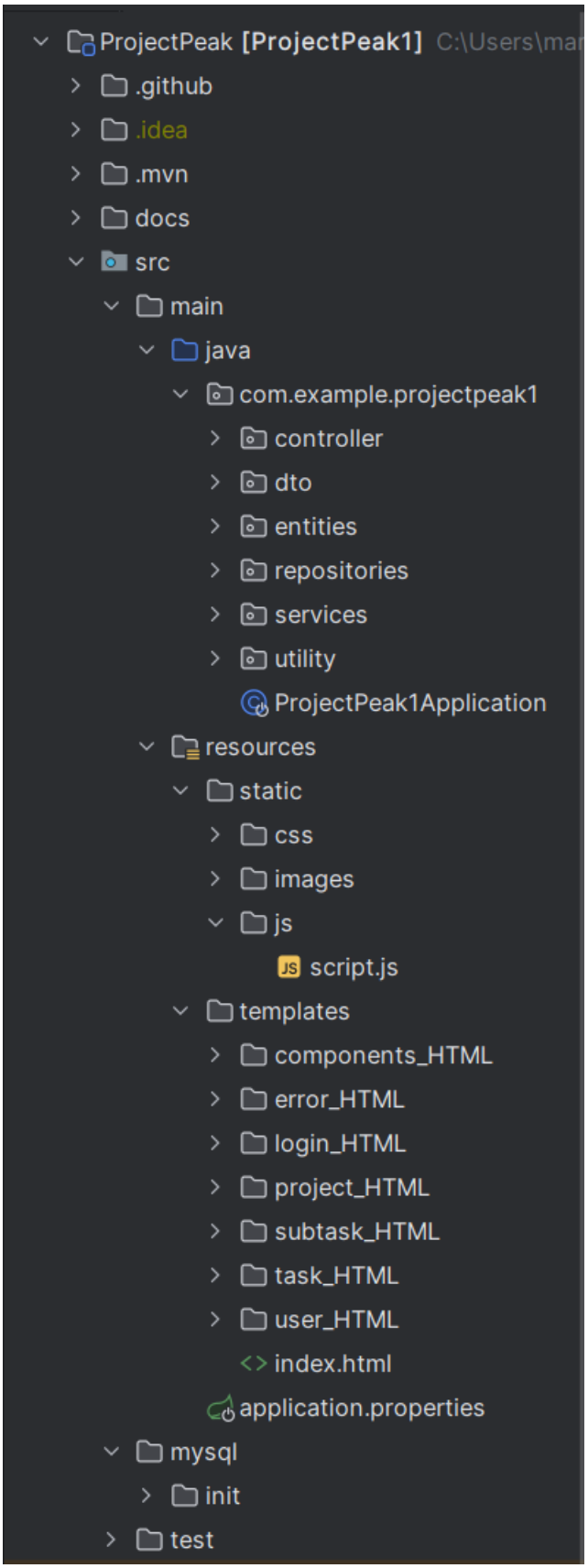

PROJECT PEAK
John Doe
Edit User
Switch Theme
Logout
Project 1
Description:
This is project 1
Start Date:
2023-05-22
End Date:
2023-06-22
Add Member:
Enter email
Add
Members:
john.doe@example.com
Days to start
Task Name
Task Description
Start Date
End Date
Days for task
Days left
Status
Subtask
Edit
Delete
Done
-38
Task
This is task 1.1 for project 1
2023-04-22
2023-04-26
0
-38
Completed
-37
Task
This is task 1.2 for project 1
2023-04-23
2023-04-24
0
-37
Completed
Add Task
See All Done Task
Back


PROJECT PEAK
Jane Doe
Edit User
Switch Theme
Logout
Project 2
Description:
This is project 2
Start Date:
2023-05-29
End Date:
2023-07-23
Add Member:
Enter email
Add
Members:
jane.doe@example.com
Days to start
Task Name
Task Description
Start Date
End Date
Days for task
Days left
Status
Subtask
Edit
Delete
Done
-31
Task
This is task 2.1 for project 2
2023-04-29
2023-05-15
0
-31
In progress
Add Task
See All Done Task
Back

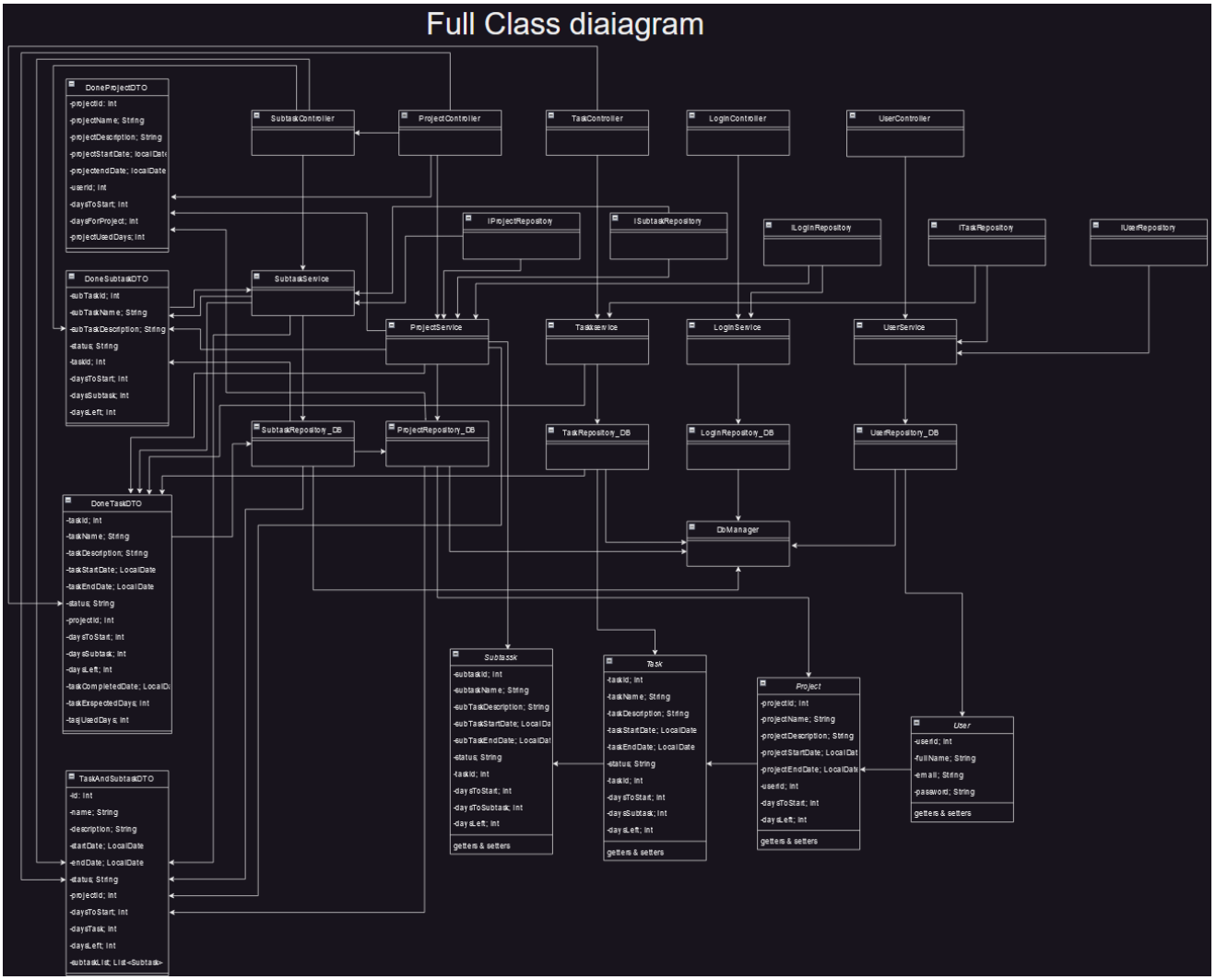
Bilag 3: kategorisering



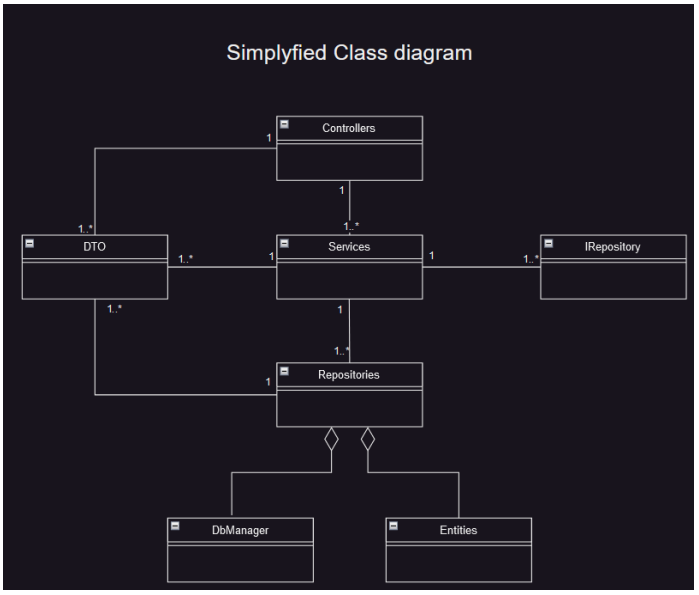
Bilag 4: systemarkitektur

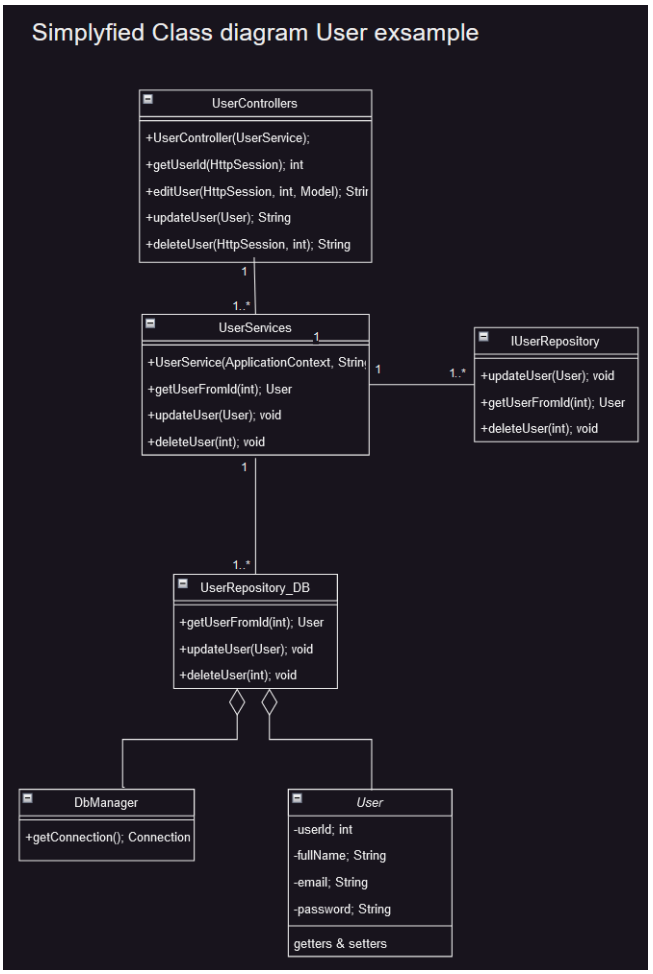


Bilag 5: fuld klassediagram til overblik

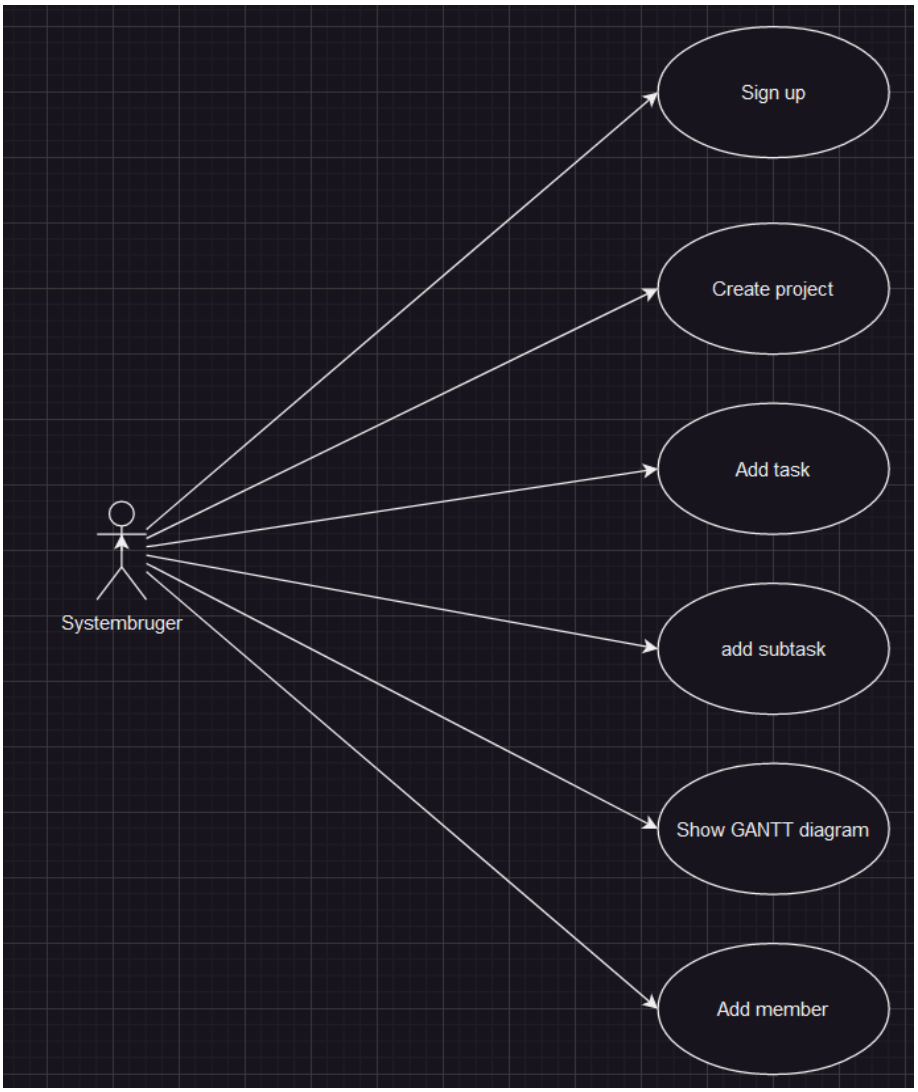


Bilag 6: simpelt klassediagram og simpelt klassediagram med User klasser som eksempel

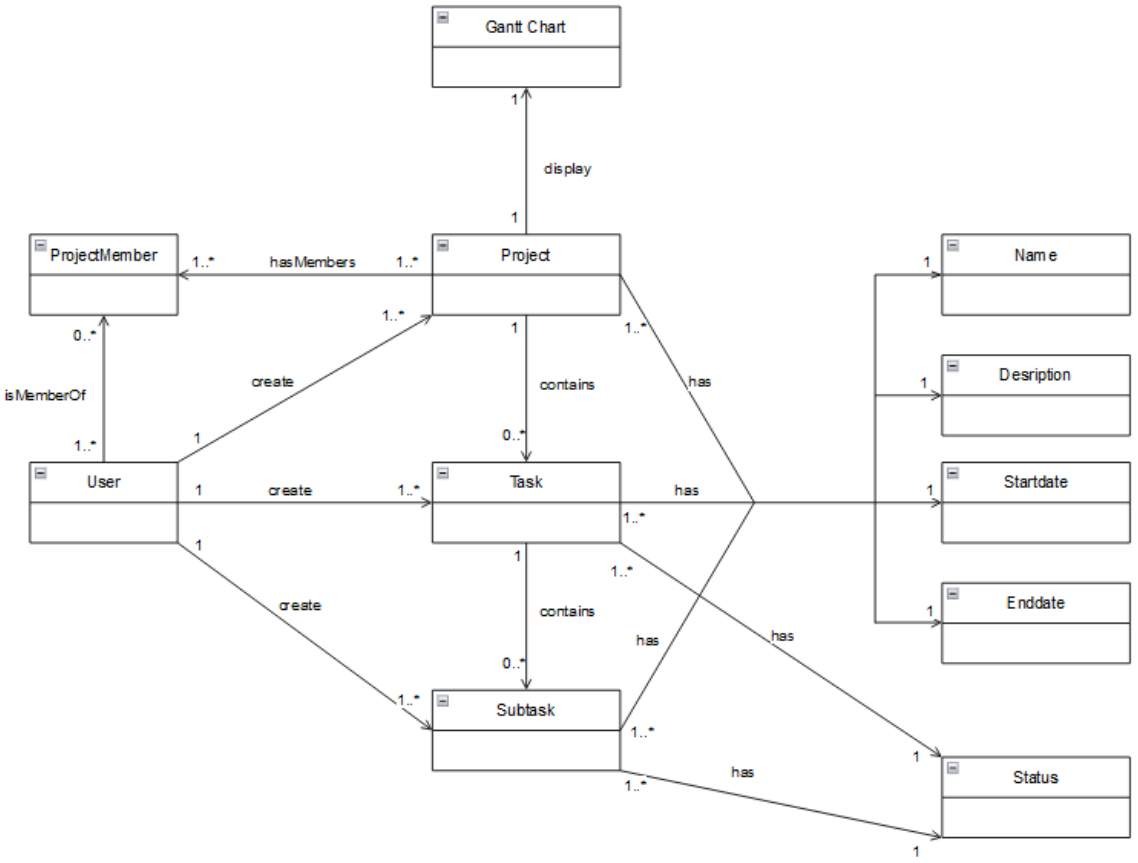




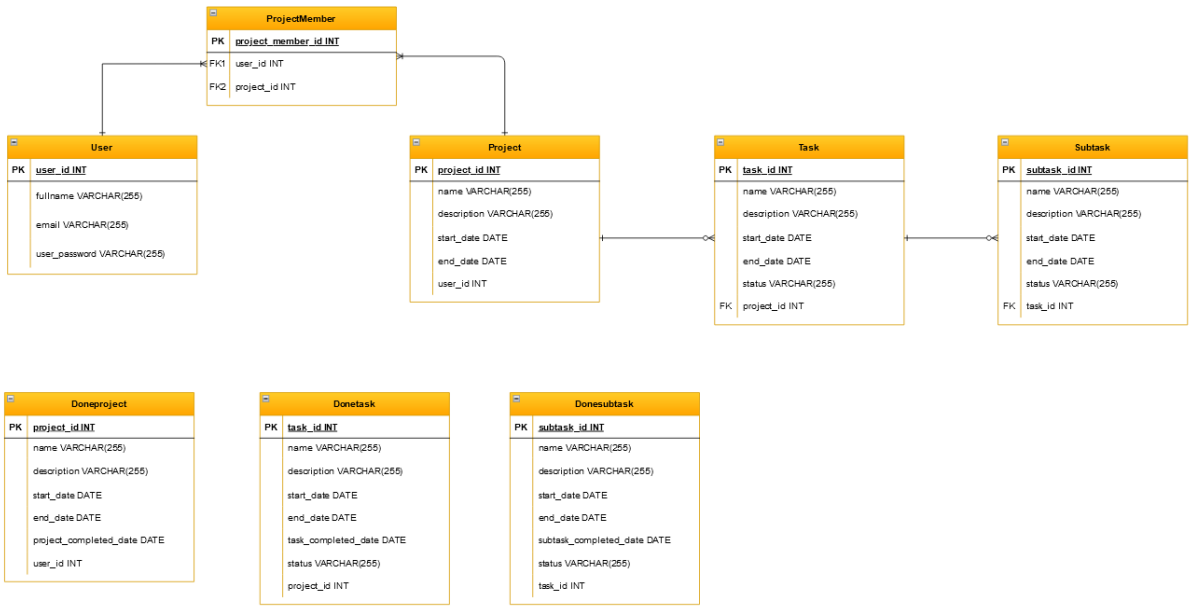
Bilag 7: Use case diagram



Bilag 8: Domænemodel



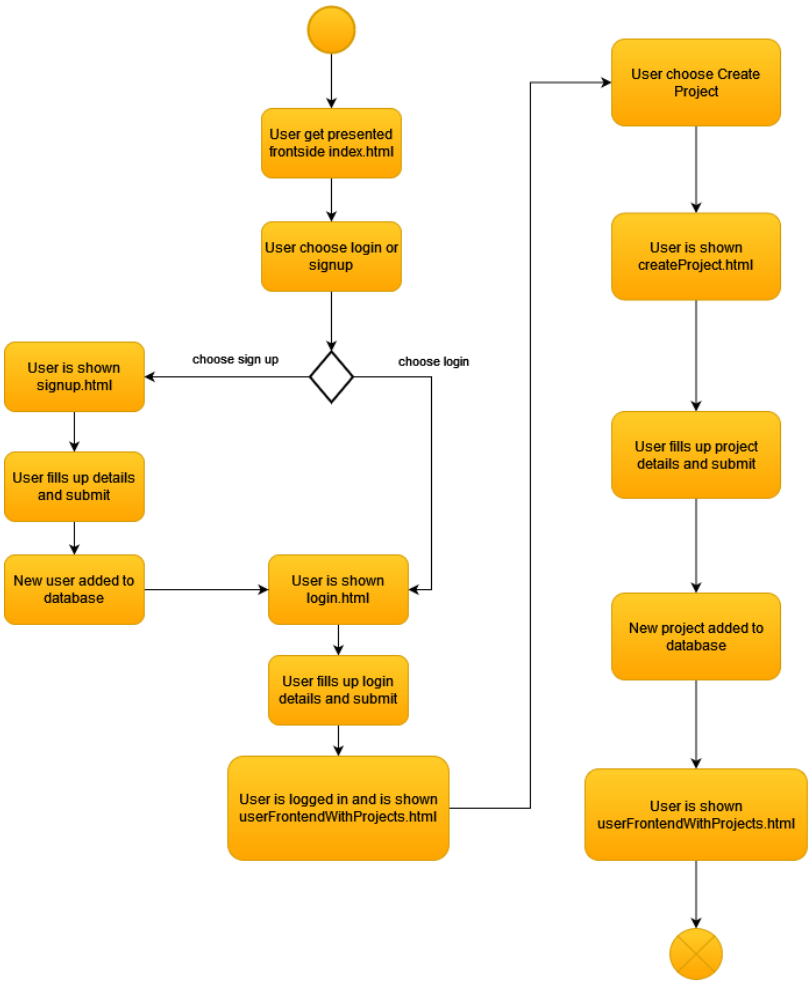
Bilag 9: ER diagram



Bilag 10: SQL script

```
DROP DATABASE IF EXISTS projectPeak;
CREATE DATABASE projectPeak;
USE projectPeak;

CREATE TABLE User (
 user_id INT AUTO_INCREMENT,
 fullname VARCHAR(255),
 email VARCHAR(255),
 user_password VARCHAR(255),
 PRIMARY KEY(user_id)
);

CREATE TABLE Project (
 project_id INT AUTO_INCREMENT,
 name VARCHAR(255),
 description VARCHAR(255),
 start_date DATE,
 end_date DATE,
 user_id INT,
 PRIMARY KEY(project_id),
 FOREIGN KEY(user_id) REFERENCES
User(user_id)
);

CREATE TABLE ProjectMember (
 project_member_id INT AUTO_INCREMENT,
 project_id INT,
 user_id INT,
 PRIMARY KEY(project_member_id),
 FOREIGN KEY(project_id) REFERENCES
Project(project_id) ON DELETE CASCADE,
 FOREIGN KEY(user_id) REFERENCES
User(user_id)
);

CREATE TABLE Task (
 task_id INT AUTO_INCREMENT,
 name VARCHAR(255),
 description VARCHAR(255),
 start_date DATE,
 end_date DATE,
 status VARCHAR(255),
 project_id INT,
 PRIMARY KEY(task_id),
 FOREIGN KEY(project_id) REFERENCES
Project(project_id)
);

 CREATE TABLE Subtask (
 subtask_id INT AUTO_INCREMENT,
 name VARCHAR(255),
 description VARCHAR(255),
 start_date DATE,
 end_date DATE,
 status VARCHAR(255),
 task_id INT,
 PRIMARY KEY(subtask_id),
 FOREIGN KEY(task_id) REFERENCES
Task(task_id)
);

CREATE TABLE DoneProject (
 project_id INT,
 name VARCHAR(255),
 description VARCHAR(255),
 start_date DATE,
 end_date DATE,
 project_completed_date DATE DEFAULT
(CURRENT_DATE),
 user_id INT,
 PRIMARY KEY(project_id)
);


CREATE TABLE DoneTask (
 task_id INT,
 name VARCHAR(255),
 description VARCHAR(255),
 start_date DATE,
 end_date DATE,
 task_completed_date DATE DEFAULT
(CURRENT_DATE),
 status VARCHAR(255),
 project_id INT,
 PRIMARY KEY(task_id)
);

CREATE TABLE DoneSubtask (
 subtask_id INT,
 name VARCHAR(255),
 description VARCHAR(255),
 start_date DATE,
 end_date DATE,
 subtask_completed_date DATE DEFAULT
(CURRENT_DATE),
 status VARCHAR(255),
 task_id INT,
 PRIMARY KEY(subtask_id)
);
```

Bilag 11: Aktivitetsdiagram



Bilag 12: kode snippet af udpdateProject metode

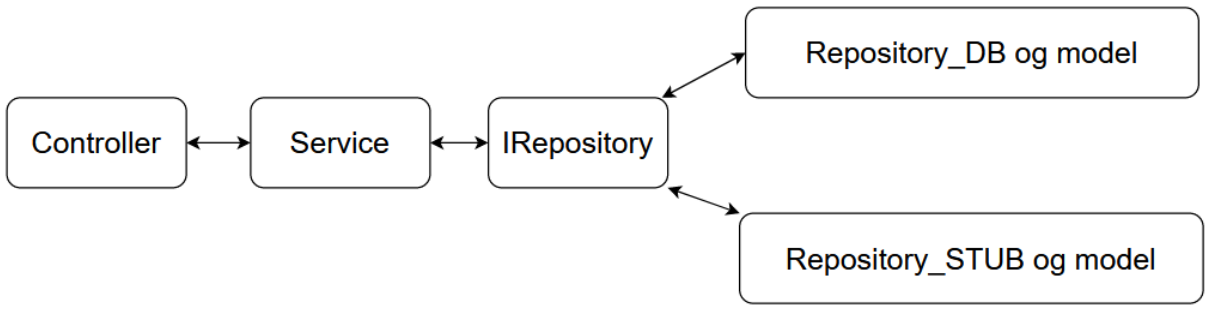
Mathias-Thomsen

```
public void updateProject(Project project){
    Project originalProject = projectRepository.getProjectById(project.getProjectId());
    // Check if the start date or end date has changed
    if (!originalProject.getProjectStartDate().equals(project.getProjectStartDate())
            || !originalProject.getProjectEndDate().equals(project.getProjectEndDate())) {

        // Update the project
        projectRepository.updateProject(project);

        // Update task and subtask dates
        updateTaskAndSubtaskDates(project, originalProject);
    } else {
        // Only update the project without updating task and subtask dates
        projectRepository.updateProject(project);
    }
}
```

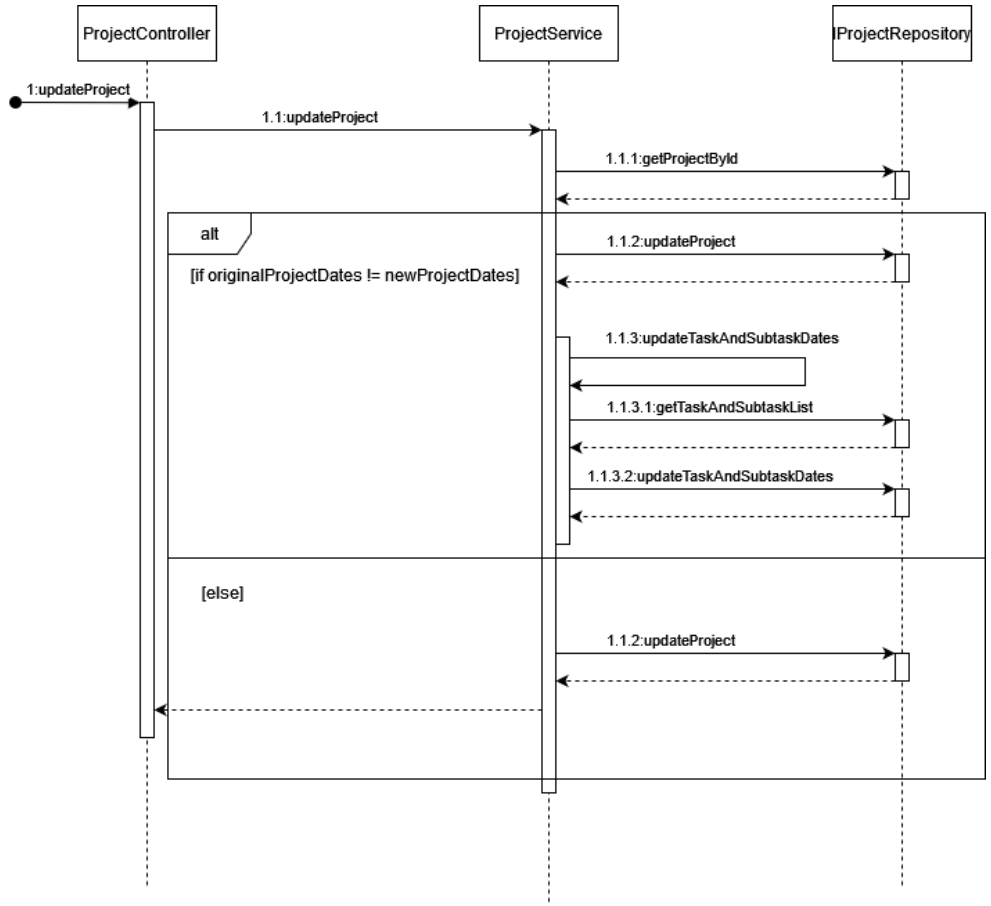
Bilag 13: kode snippet af updateTaskAndSubtaskDates

```java
public void updateTaskAndSubtaskDates(Project project, Project originalProject) {
    LocalDate originalStartDate = originalProject.getProjectStartDate();
    LocalDate originalEndDate = originalProject.getProjectEndDate();
    LocalDate newStartDate = project.getProjectStartDate();
    LocalDate newEndDate = project.getProjectEndDate();

    // Calculate the difference between the original and new project start dates
    Period startDateDifference = Period.between(originalStartDate, newStartDate);

    // Calculate the difference between the original and new project end dates
    Period endDateDifference = Period.between(originalEndDate, newEndDate);

    //If end date is not change, we set the end date to start to end date to the task and subtask move with the project dates.
    if(startDateDifference.isZero() && !endDateDifference.isZero()){
        endDateDifference = Period.ZERO;
    } else if (endDateDifference.isZero()) {
        endDateDifference = startDateDifference;
    }


    List<TaskAndSubtaskDTO> list = projectRepository.getTaskAndSubTaskList(project.getProjectId());

    for (TaskAndSubtaskDTO task : list) {
        // Update task start and end dates based on the project's start and end date differences
        LocalDate taskStartDate = task.getStartDate().plus(startDateDifference);
        LocalDate taskEndDate = task.getEndDate().plus(endDateDifference);
        task.setStartDate(taskStartDate);
        task.setEndDate(taskEndDate);

        // Update subtask start and end dates based on the task's start and end date differences
        for (Subtask subtask : task.getSubTaskList()) {
            LocalDate subtaskStartDate = subtask.getSubTaskStartDate().plus(startDateDifference);
            LocalDate subtaskEndDate = subtask.getSubTaskEndDate().plus(endDateDifference);
            subtask.setSubTaskStartDate(subtaskStartDate);
            subtask.setSubTaskEndDate(subtaskEndDate);
        }
        projectRepository.updateTaskAndSubtaskDates(task);
    }
}
```

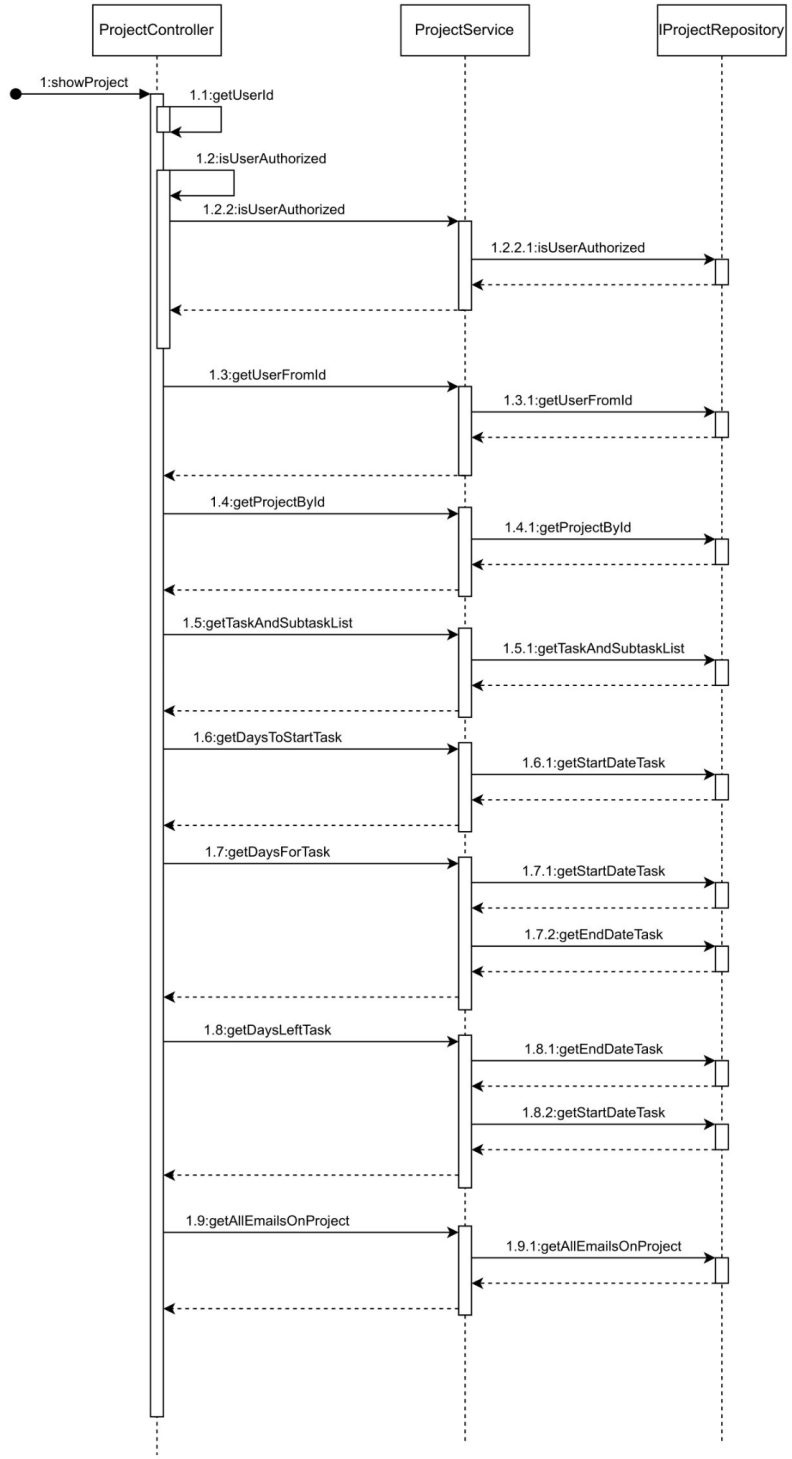
Bilag 14: System hierarki



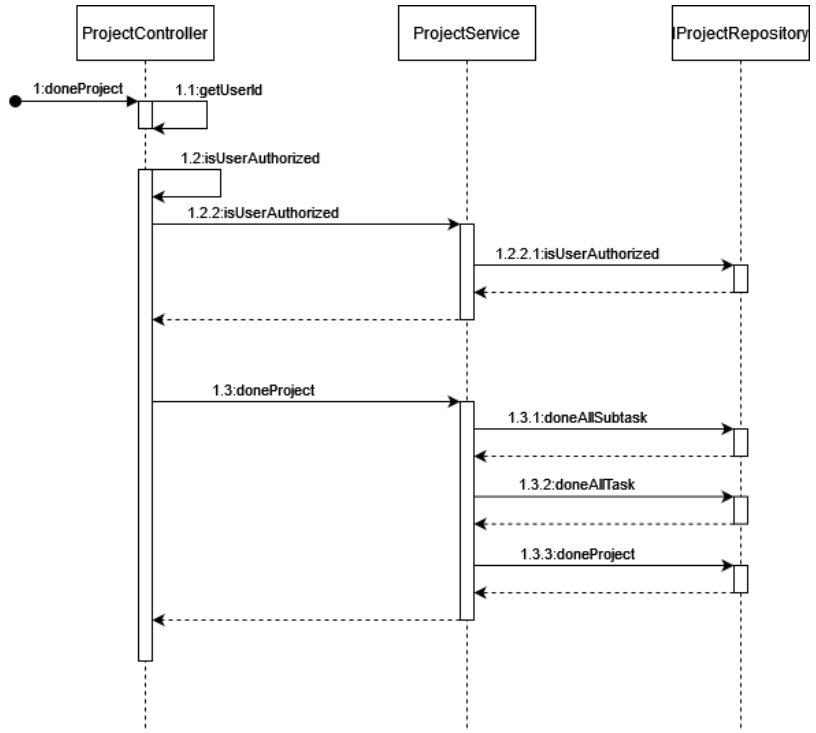
Bilag 15: Sekvensdiagram UpdateProject



Bilag 16: Sekvensdiagram showProject



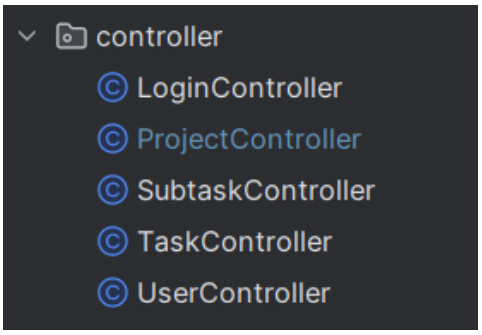
Bilag 17: Sekvensdiagram doneProject

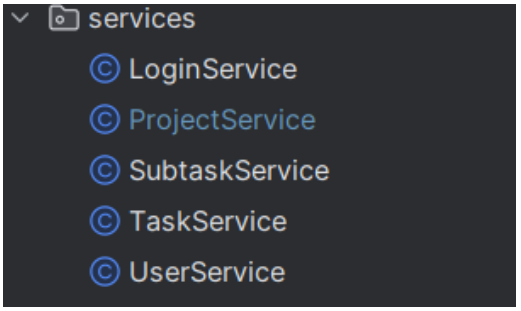


Bilag 18: kode snippet af getMapping index metode

```
Mathias-Thomsen *
@GetMapping(value = {"/frontendWithProjects"})
public String index(HttpSession session, Model model) {
    int userId = getUserId(session);
    if (userId == 0) {
        return "login_HTML/login";
    }
    User user = projectService.getUserFromId(userId);
    model.addAttribute("user", user);

    List<Project> list = projectService.getAllProjectById(userId);

    model.addAttribute("projects", list);
    return "project_HTML/frontendWithProjects";
}
```

Bilag 19: Billedeksempler controller og service

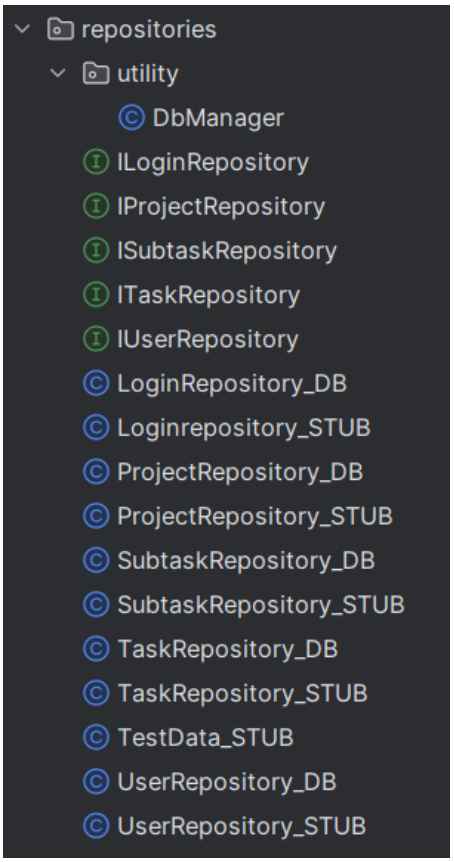




Bilag 20: kode snippet projectService metode

```
@Service
public class ProjectService {


    IProjectRepository projectRepository;

    Mathias-Thomsen
    public ProjectService(ApplicationContext context, @Value("${projectrepository.impl}") String impl) {
        this.projectRepository = (IProjectRepository) context.getBean(impl);
    }
```

Bilag 21: Repository package



Litteraturliste

1. DMR, 2022, Trello Statistics and User Count(2023)
   Trello statistics, fundet på:
   https://expandedramblings.com/index.php/trello-facts-statistics/
   (læst 19-05-2023)

2. 2022-Zwisler, Projekt og analyseredskaber.pdf

3. 2022-Lassen, Projektledelse 2. udgave (risikoanalyse, kommunikation).pdf

4. https://dzone.com/articles/solid-grasp-and-other-basic-principles-of-object-o
   (læst 25-05-2023)

5. Muhammad Umair, 2018, SOLID, GRASP, and Other Basic Principles of Object-Oriented, Design, fundet på:
   https://dzone.com/articles/solid-grasp-and-other-basic-principles-of-object-o
   (læst 25-05-2023)

6. https://discord.com/safety/360044149331-what-is-discord (læst 15-05-2023)

7. Rafael D. Hernandez, 2021, The Model View Controller Pattern - MVC Architecture And Frameworks Explanied, funder på:
   https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/ (læst 20-05-2023)