Procesrapport

Svendeprøve

Elev
Mathias Frederik Græsholt

Projektnavn
FunRun

Uddannelsessted
TECHCOLLEGE
Struervej 70,
9220 Aalborg

Elevplads
EMD International
Niels Jernes Vej 10,
9220 Aalborg Ø

Projektperiode
21. marts 2022 - 19. april 2022

Afleveringsdato
19. april 2022

Fremlæggelsesdato
26. april 2022

Vejledere
Frank Rosbak
Lærke Brandhøj Kristensen

Censor
Allan Kaa Jensen

Forord

Denne rapport er en af to skrevet til svendeprøveprojektet FunRun af Mathias Frederik Græsholt.

Projektet er udarbejdet i perioden 21. marts 2022 til 19. april 2022, med vejledning fra Frank Rosbak og Lærke Brandhøj Kristensen.

I denne rapport beskriver jeg hvordan jeg kom frem til mit endelige produkt.

Jeg beskriver projektets case og problemformulering, redegør for hvordan jeg planlægger at gennemføre projektet, samt hvordan jeg endte med at afvige fra denne plan, og reflekterer over grundene til dette.

Jeg fortæller om de metoder og teknologier jeg har valgt at bruge til projektet, hvilke jeg kunne have valgt i stedet, og hvorfor jeg har valgt dem fra.

Til slut konkluderer jeg på projektforløbet, hvad jeg har lært, og hvad jeg tænker om projektet som helhed.

Jeg takker alle involverede i projektforløbet.

Indholdsfortegnelse

Læsevejledning

Denne rapport er en af to der hører til svendeprøveprojektet FunRun, skrevet af Mathias Frederik Græsholt. Dette er procesrapporten og den anden er produktrapporten.

I produktrapporten beskrives projektets produkt og teknologier. I procesrapporten beskrives forløbet, og hvordan produktet blev formet.

Det anbefales at man starter med at læse produktrapporten, og får et indblik i hvad projektet er, før man læser procesrapporten.

Samlet kildeliste findes bagerst i rapporten.

I denne rapport bruges en del tekniske forkortelser, men især to er vigtige at kende:

API står for Application Programming Interface, og er en type program der faciliterer kommunikation mellem to andre teknologier.

PWA står for Progressive Web App, og er en type hjemmeside der kan downloades som en app.

Begge teknologier bliver beskrevet dybere i produktrapportens teknologiafsnit.

Indledning

Som teknologien skrider frem, kræves mindre og mindre af vores kroppe, som flere timer end aldrig før får lov at sidde stille.

Procenten af befolkningen der lider af overvægt, er stadigt stigende, og dette er et problem for folkesundheden der nedsætter livskvaliteten.

Folk der ønsker at komme dette til livs, kan vælge et væld af motionsmuligheder for at holde sig i form, men en af de mest tilgængelige motionsformer er nok løb.

Løb kræver ikke noget udstyr, og næsten alle kan gøre det på et niveau der passer til dem.

Et brugbart udstyr til løb kunne dog være et system der kunne vise ens ture og give en indblik i relevant statistik deromkring, og derved fastholde løbere, som uden konkret bogføring og visualisering af deres fysiske motion hurtigt kunne miste interesse.

Projektet FunRun er et svar på dette, og i denne rapport uddyber jeg den proces jeg gik igennem for at nå frem til mit endelige produkt.

Case beskrivelse

Løb er en kæmpe hobby. Ud over at være en god måde at få motion på i vores i stigende grad sedentære hverdag, er der flere og flere der tager det meget seriøst på hobbyplan.

I denne hektiske hverdag kan det være en udfordring at holde styr på både job, hus, samlever, børn, og så ovenikøbet ens løberuter. Derfor er der behov for et system der er letvægtigt som kan gemme de ruter man løber, uden at tvinge en bruger til at have flere enheder på sig end højest nødvendigt.

Andre problemer for løbere kan være at de taber motivationen, og mister overblikket over hvad de får ud af deres træning, eller at de igen og igen udskyder at komme i gang, fordi de ikke har nogen måde at overvåge deres fremskridt på.

Problemformulering

Hvordan kan jeg opbevare en brugers løbedata, og præsentere dem med brugbar statistik og logning?

Estimeret tidsplan

Min estimerede tidsplan er vedlagt denne rapport som bilag 1.

Et valg jeg har taget, er at inkludere weekender og helligdage, selvom jeg som udgangspunkt ikke planlægger at arbejde disse dage. Dette vil gøre sammenligning af estimeret og realiseret tidsplan mere overskuelig, hvis jeg ender med at tage de dage i brug.

Min plan for projektet er at starte med de formalia i forhold til rapporterne der skal gøres færdige og fastlægges inden for første uge først, det værende case beskrivelse, problemformulering, tidsplan, og kravspecifikation.

Efterfølgende vil jeg fordybe mig i projektet og opbygge et minimum viable product, som i mit projekts tilfælde vil betyde et API med tilhørende database, som kan lagre data fra en PWA. Når der er hul igennem for den kommunikation, er den mest komplicerede del af projektet i min optik overstået.

Efter minimum viable product er opnået, vil jeg vende tilbage til rapporterne med en god ide om hvordan mit projekt er opbygget. Dette gør mig i stand til at skrive de store metode- og teknologiafsnit i begge rapporter, samt afsnittet om mit projekts arkitektur.

Efter dette vil jeg atter vende tilbage til projektet for at finpudse kommunikationen mellem PWA og API, og udvide PWA'ens funktionalitet. Det er her jeg planlægger at kode visualisering af den data jeg programmerede opsamling og lagring af i første omgang.

Sidst, men ikke mindst, vil jeg vende tilbage til rapporterne for at skrive brugervejledninger, og derefter finpudse og skrive de sidste afsnit, navnlig forord, læsevejledninger, og konklusion. Det er også over påsken jeg vil få mine bekendte og allierede til at læse mine rapporter igennem og hjælpe mig med korrektur.

Jeg har valgt at sætte rigeligt tid af til hver opgave, som er noget jeg som regel gør når jeg planlægger, ligegyldigt formålet. Forventningen er, at jeg kommer hurtigere igennem nogle af opgaverne end jeg har anført, men jeg vil hellere planlægge at bruge for meget tid på en opgave end for lidt. Skulle jeg komme forud for planen, har mit projekt områder nok jeg kan tage fat i og udvide.

Metode- og teknologivalg

I dette afsnit vil jeg begrunde hvorfor jeg har valgt de forskellige teknologier som projektet benytter.

API

Jeg har valgt at bruge et API til mit projekt som bindeled mellem min database og PWA, da de ellers ville være nødsaget til at køre på samme enhed. Jeg bruger et ASP.NET webAPI fordi det er noget jeg har kendskab til, og man får meget foræret i forhold til opsætningen, så man hurtigt kan komme i gang.

ASP.NET, Node.js, Rust, og PHP

I stedet for ASP.NET kunne man have brugt et utal af forskellige andre teknologier til at opsætte et API.

Node.js er et open-source alternativ, hvor man kan eksekvere JavaScript serverside, og man kunne derfor have valgt at holde sig til færre sprog på tværs af projektet. Node.js er i særdeleshed interessant hvis man arbejder med en microservice arkitektur[1].

Man kunne have valgt at bruge Rust, som flere og flere gør. Rust er hurtigere end ASP.NET og på visse punkter nemmere at arbejde med, men vanskeligere at lære[2].

Man kunne også have brugt PHP til at arbejde med databasen. PHP er ikke helt så hurtigt som ASP.NET, og egner sig dårligere til store virksomheder og systemer[3].

Jeg har valgt ASP.NET grundet mit kendskab og højere synergi med andre Microsoft produkter jeg benytter.

bcrypt og SHA-family

Til hashing af kodeord i databasen bruger jeg bcrypt, som er en af de sikreste måder at hashe kodeord på.

Man kunne have brugt en af algoritmerne fra SHA-familien (SHA1, SHA256, eller SHA512) til at hashe kodeord, men forskellen er at de alle er 'hurtige' algoritmer, mens bcrypt er 'langsom'. Det at en algoritme er hurtig er godt på mange tidspunkter, men når det kommer til kodeord, gør det den nemmere at knække og derfor usikker[4].

Database

Mit projekt benytter en MS-SQL database som er sat op via Entity Framework ud fra Code-First princippet.

Jeg har valgt MS-SQL fordi den passer godt sammen med mit API, og det er igen noget jeg har kendskab til.

[1] https://www.geeksforgeeks.org/difference-between-node-js-and-asp-net/
[2] https://visualstudiomagazine.com/articles/2021/05/03/net-rust.aspx
[3] https://www.pixelcrayons.com/blog/php-vs-asp-net-how-to-choose-the-right-one/
[4] https://rietta.com/blog/bcrypt-not-sha-for-passwords/

Dokument- vs. relationelle databaser

Man kunne have brugt en dokumentdatabase i stedet for en relationel database. En dokumentdatabase er typisk hurtigere end en relationel, og bedre egnet til at arbejde med big data, men opbevarer ikke dataene i tabeller på samme måde. En dokumentdatabase har heller ikke indbyggede relationer.

Det er min erfaring at dokumentdatabaser er rigtig gode til det de er lavet til, men lige så snart man skal ud i at bruge relationer, er man bedre tjent med en relationel database, frem for at kode relationerne manuelt.

SQLite

Man kunne også have lavet projektet med en SQLite database, som er en form for lille database der kan køre lokalt, uden en server. SQLite er typisk langsommere end en fuld database, og bytter også en del andre funktioner for sin evne til at køre separat.

Da filosofien bag mit system er at brugeren skal kunne logge ind og tilgå deres data hvor som helst, er SQLite ilde egnet.

Entity Framework

Jeg har valgt at sætte min database op via Entity Framework, som er Microsofts bud på et Object Relational Mapping tool, eller ORM. Det vil sige at jeg kan arbejde med og lagre data i objekter i stedet for tabeller, og det gør dataene nemmere at arbejde med i koden. Ydermere, gør brugen af Code-First princippet at man undgår eventuelle fejl der kan opstå i opsætningen af en traditionel database som man kobler sammen med et API manuelt[5].

PWA

Jeg har valgt at lave en Progressive Web App til mit projekt. En PWA kan tilgås og installeres fra nettet på en bred vifte af enheder, og det er en nem måde at opnå en cross-platform løsning på.

Native App, PWA, WPF

I stedet for en PWA kunne man have lavet en native app. Med en native app begrænser man sig i forhold til platform, og appen ville ikke længere være tilgængelig i en browser. En native app ville køre hurtigere end en PWA, på bekostning af at være en del sværere at sætte op. Native apps kan også tilgå funktioner på visse enhedskomponenter som PWA endnu ikke har adgang til, skønt den tilgængelige funktionalitet er voksende hos PWA[6].

Hvis jeg ikke ville lave en app, kunne jeg i stedet have lavet en WPF applikation til desktop. WPF tager over for Windows Forms, og har en række ting der er blevet strømlinet og gjort nemmere. Jeg mente dog at

[5] https://www.entityframeworktutorial.net/what-is-entityframework.aspx
[6] https://web.dev/what-are-pwas/

det ville blive svært at forsvare ikke at bruge en lokal database, hvis det kørte som WPF, i hvilket tilfælde det ikke ville opfylde alle de obligatoriske projektkrav. Derfor valgte jeg at lave en App. Med en WPF applikation skulle jeg også have fundet en anden måde at få data på, da en typisk computer ikke har en GPS indbygget. Svaret på dette kunne måske have været en embedded enhed.

Angular, Blazor, React, og Vue.js

Min PWA er programmeret i et framework der hedder Vue.js. Jeg har arbejdet med Vue.js før, og jeg finder det intuitivt.

Angular, som Vue.js er inspireret af, ligger lige til højrebenet at nævne som et alternativ. Angular har dog et par ting der gør det bedre egnet til større systemer end Vue.js. Derudover er Angular sværere at arbejde i og gå til, og dets performance er langsommere end Vue.js'. Desuden er der mange ting der skal gentages rundt omkring i koden, som også bidrager til at jeg vælger det fra[7].

Blazor er også et alternativ. Hvor Node.js kan køre JavaScript på serversiden og på den måde gøre at meget af koden ville være et og samme sprog, gør Blazor noget af det samme. Blazor gør at du kan kode din PWA i C#, og opnå kodeensartethed på tværs af et projekt. Hvis jeg havde arbejdet i Blazor, ville mit projekt være mere ensartet at se på, og jeg vil hellere demonstrere et bredere teknologikendskab[8].

Man kunne også have skrevet i React. React og Vue.js ligner meget hinanden, og mange rigtig store hjemmesider og services bruger React. React bruger dog en sværere syntax, og i React programmerer du hele din hjemmeside eller applikation i JavaScript, hvor du i Vue.js kan dele det op i HTML, CSS, og JavaScript[9].

Versionsstyring

Versionsstyring er vigtigt. Lige så snart du arbejder med et moderat komplekst system er det vigtigt at kunne gå tilbage i kodens historik hvis problemer skulle opstå.

Jeg har valgt at bruge Git til versionsstyring af mit system, selvom det ikke er det jeg har arbejdet mest med, da jeg mener at Git er fremtiden indenfor versionsstyring.

Dropbox, Git, Subversion

I stedet for Git, kunne man have valgt at bruge Subversion, Git's forgænger og det system jeg er mest fortrolig med. Git har nogen fordele over Subversion, for eksempel at det er distribueret, som vil sige at Git kører lokalt og man kan comitte til sit eget lokale repository før man pusher til main, hvor Subversion er centralistisk, som vil sige at man comitter til main hver gang. Denne forskel betyder også at Git kan comitte

[7] https://www.simform.com/blog/angular-vs-vue/
[8] https://www.telerik.com/blogs/blazor-vs-vue-web-developers
[9] https://www.codica.com/blog/react-vs-vue/

offline, hvor Subversion skal have forbindelse til repository. Ud over dette, er Git også bedre egnet til at arbejde med branches end Subversion. Essentielt er forskellen på de to ikke så væsentlig i et enmandsprojekt på denne størrelse. Jeg er dog interesseret i at arbejde mere med Git, for at lære at bruge teknologien[10].

Ydermere har GitHub en funktionalitet, GitHub Actions, som gør det muligt at sætte services op med Continouos Integration og Deployment. Med GitHub Actions kan du i en fil i dit repository definere jobs, som at builde, teste, og publishe, som bliver udført ved events, som for eksempel når du pusher. Når jeg pusher til main er mit API sat op til at builde, unit teste, og publishe, og min PWA er sat op til at builde og publishe, så ved et enkelt click har jeg ti minutter senere ændringer ude som jeg kan teste på min enhed.

Jeg vil også nævne at Dropbox har 30 dages historik på alle filer, så man kunne til nød bruge det som et kortsigtet alternativ. Dropbox har dog ikke en brugervenlig måde at gendanne flere filer på en gang til en tidligere version, og ej heller har de en struktureret måde at sikre at alt kan bygge når det bliver comittet. Det vil derfor være en mangelfuld afløser for enhver der er vant til at arbejde med Git.

IDE'er og hjælpeprogrammer

Gennem projektet har jeg brugt en serie af forskellige værktøjer til at udvikle mit produkt. Disse vil jeg begrunde i dette afsnit.

Visual Studio 2022

Jeg har valgt at bruge Visual Studio 2022 til at udvikle mit API, da jeg arbejder i ASP.NET og begge er udviklet og vedligeholdt af Microsoft. Man kunne have bevæget sig op og ned i kompleksitet for ens IDE ved at vælge andre, men til udvikling af et funktionelt API har Visual Studio hverken mere eller mindre end jeg behøver. Ville man bevæge sig ned i kompleksitet kunne man vælge Visual Studio Code, som er et mere letvægtigt alternativ.

Swagger

Jeg har Swagger indbygget i mit API, som gør mig i stand til at teste alle mine endpoints uden behov for et tredjepartsprogram. Swagger er smart fordi det også giver dig meget information om dine endpoints, som for eksempel forslag til syntaxen til requests, hvor man bare kan fylde ind. Med Swagger har jeg også mulighed for at enable authentication på siden, så jeg kan kalde mit login endpoint for at få en JWT token, tilføje den til siden, og så uden problemer kalde og teste de endpoints der forventer sådan en.

Ellers kunne man have testet sine endpoints med Postman, som jeg har gjort før, men efter at arbejde med Swagger tror jeg aldrig jeg vil bruge Postman igen, i hvert fald ikke hvis jeg har valget om at få Swagger indbygget fra starten. I Postman kan du skrive requests og sende til dine endpoints. Resultatet er det

[10] https://blog.hackbrightacademy.com/blog/svn-vs-git/

samme som Swagger, men du har flere muligheder for at begå fejl, for eksempel ved at sende det forkerte format eller forkert syntax.

Conveyor

Til at gøre mit API tilgængeligt over nettet, for testning af min app på en anden enhed, har jeg brugt en Visual Studio extension der hedder Conveyor. Conveyor gør det meget nemt at tilgå ens services på denne måde, primært med fokus på at teste dem.

Visual Studio Code

Til at udvikle min PWA har jeg brugt Visual Studio Code. Man kunne også have brugt Visual Studio, men jeg kan godt lide at bruge det mest letvægtige jeg kan til en given opgave, og Visual Studio har ikke rigtig nogen værktøjer jeg skal bruge til at udvikle en PWA.

Visual Studio Code er et letvægtigt alternativ til Visual Studio, og det er noget nemmere at finde rundt i. Det er også nemt at finde og konfigurere extensions i, og jeg bruger det meget når jeg koder i min fritid.

Firebase

Fordi min PWA ikke kan køres gennem Visual Studio, er Conveyor ikke en løsning når jeg vil teste den på en smartphone. For at gøre min PWA tilgængelig online til test har jeg brugt Firebase. Firebase er en Google service hvor man gratis kan oprette en bruger og deploye services, som derefter kan tilgås udefra.

Microsoft SQL Server Management Studio

Til at inspicere min database efter oprettelse gennem Visual Studio, har jeg brugt Microsoft SQL Server Management Studio. Da det er en MS-SQL database, og Microsoft SQL Server Management Studio er den tilknyttede IDE, gav det mening. Herigennem kunne jeg inspicere tabeller og kolonner i et grafisk interface, og kontrollere at det hele så ud som jeg havde planlagt.

GitKraken

Til at interagere med Git og mit repository på Github.com har jeg brugt GitKraken. Der er mange forskellige programmer til at bruge Git, for eksempel GitHub Desktop og Tortoise Git. Ud fra min erfaring kan de stort set det samme, så jeg valgte en jeg vidste var populær og som jeg har set andre bruge.

Jeg har ikke gjort meget andet i selve GitKraken end at pushe til GitHub, og til det har det virket fortrinligt.

Hosting

Til endelig hosting af min løsning har jeg valgt at bruge Microsoft Azure.

Microsoft Azure er den næststørste Cloud Computing platform, næst efter Amazon Web Services, men den hurtigst voksende[11].

[11] https://www.simplilearn.com/tutorials/azure-tutorial/what-is-azure

Jeg valgte at bruge Microsoft Azure fordi den er nyere end Amazon Web Services, jeg har hørt mere positivt om den, og som endnu et Microsoft produkt kan jeg publishe til den direkte fra Visual Studio uden at skulle døje med extensions eller lignende. Jeg kunne også bruge Microsoft SQL Server Management Studio til at konfigurere firewall på min SQL database på Microsoft Azure, som er endnu en ting jeg ikke nødvendigvis ved var gået gnidningsfrit på Amazon Web Services.

Microsoft Azure tillod mig også at sætte Continouos Integration/Deployment op på begge dele af projektet med GitHub actions. Jeg måtte lære at bruge Microsoft Azure da jeg nåede til hosting, men havde jeg vidst hvor nemt det ville være når det var oppe at køre, havde jeg gjort det som det allerførste. Det havde sparet mig at bruge Conveyor og Firebase til testing. Den eneste ulempe ved Microsoft Azure i den sammenhæng er at den er noget længere om at deploye, hvor Conveyor og Firebase tilsammen er næsten øjeblikkelige. Derfor er det svært at sige om jeg i virkeligheden havde sparet tid og besvær ved at gå direkte til Microsoft Azure. Når jeg siger det var nemt, skal det også siges at det tog et tocifret antal timer over flere dage, men de timer kunne lige så godt have været givet ud i starten, som lidt over halvvejs.

Alternativet til at bruge en Cloud Computing platform, kunne også være en traditionel server. Hvis jeg havde en fysisk server og et domæne til rådighed kunne jeg lave mit API og min PWA til Docker images, som jeg så kunne køre på serveren, og derefter route trafikken mod et subdomæne til exposede porte på containerne. Derved ville man kunne tilgå begge services udefra via subdomæner.

En traditionel server blev ikke, trods forespørgsel, stillet til rådighed for dette projekt.

Væsentlige elementer fra produktrapporten

Jeg vil vælge at fremhæve afsnittet Overordnet Arkitektur fra Produktrapporten som repræsentativt af det der er kodemæssigt interessant ved projektet.

Ydermere, hvis der er forvirring eller spørgsmål til hvordan systemet fungerer, er brugervejledningerne sidst i produktrapporten skrevet til nye brugere af systemet, og de beskriver i detaljer hvordan man gør alt fra at installere appen til at oprette, vise, og slette løbeture.

Realiseret tidsplan

Min realiserede tidsplan er lagt ved denne rapport som bilag 2.

Mit projekt har mere eller mindre fulgt den fremgangsmåde, hvis ikke helt den tidsplan, som jeg lagde ud i starten.

Ligesom jeg forudsagde, tog mange opgaver mindre tid end jeg havde planlagt, hvilket gav mig tid til et længere, mere koncentreret programmeringsforløb i midten. Dette gjorde at mit produkt blev bedre og pænere end jeg havde turdet håbe at det ville.

Hvis jeg skulle planlægge projektet forfra, ville jeg granulere mine overskrifter i tidsplanen yderligere, selvom jeg synes at jeg gjorde mig umage med det. Der er nogle opgaver der kom til at gå ind under overskrifter hvor de ikke helt passer på min realiserede tidsplan, for eksempel at deploye services til Microsoft Azure, som gik ind under opsætning, siden jeg ikke havde et punkt til det.

Helt konkret var det ugennemtænkt at planlægge at skrive afsnittet Overordnet Arkitektur før halvdelen af arkitekturen var planlagt at være færdig.

Som sagt da jeg estimerede min tidsplan, så jeg det mest som en beskrivelse af min fremgangsmåde, og en guide for hvornår ting burde være færdige, og efter den intention har jeg fulgt den ret godt.

Min logbog med detaljerede beskrivelser af hvad jeg har lavet dag for dag, er vedlagt som bilag 3.

Konklusion

Gennem projektet har jeg udviklet en proof-of-concept løsning på det problem jeg lagde ud i min case beskrivelse og problemformulering, og jeg mener at systemet besvarer dem mere end tilfredsstillende.

Systemet jeg kom frem til, benytter et API med tilhørende database, som en PWA kan udveksle information med, deriblandt positionsdata fra en enheds GPS. Denne information bliver lagret i databasen, og kan senere ses af brugeren.

Projekter som disse ser jeg som en kæmpe læringsmulighed. Jeg har arbejdet med ting som jeg har kendskab til gennem min undervisning, fordi jeg til hverdag arbejder med kode på en måde der ikke egner sig godt til besvarelse af de krav der er stillet til dette projekt. Ud over disse, har jeg også stiftet bekendtskab med teknologier og udfordringer jeg ikke ser i min hverdag på min læreplads, som for eksempel Git og Microsoft Azure, som jeg i den grad kan tage med mig videre.

Jeg går til enhver mulighed for at arbejde på tværs af flere teknologier med iver, for det er noget jeg synes er enormt spændende, og desværre ikke noget jeg arbejder med så ofte.

Det har været et udfordrende og hårdt forløb, men jeg er stolt af resultatet.

Hvis jeg skulle arbejde videre med projektet har jeg nogen ideer til funktionalitetsudvidelser jeg ville starte med:

- Hvis systemet skulle bruges i den virkelige verden, skulle der være en måde at give ture navne på, så man kunne genkende dem senere. Det ville dog kræve at hele siden der viser løbeture gendesignes, da der på nogen enheder i forvejen ikke er plads til mere på skærmen.
- Det kunne også være fedt at kunne ændre kodeord, men for det skulle jeg have en måde at verificere at en bruger er hvem de er, så der skulle tilknyttes enten e-mail eller telefonnummer til kontoerne.
- Ellers ville det være fedt at kunne fremvise noget længeresigtet statistik over hvordan en løber har forbedret sig, for eksempel et diagram der viser topfart på tværs af ture eller lignende.

Disse er ting der kunne inkluderes i projektet fremadrettet, men systemet i dets nuværende form indeholder og demonstrerer allerede meget forskellig funktionalitet. Det kan tages i brug og give brugere værdi.

Kildeliste

Difference Between Node.js and Asp.net
https://www.geeksforgeeks.org/difference-between-node-js-and-asp-net/
(Sidst besøgt 12/04/2022)

Rust Language Gains Traction in .NET Community
https://visualstudiomagazine.com/articles/2021/05/03/net-rust.aspx
(Sidst besøgt 12/04/2022)

PHP Vs ASP.NET: How to Choose the Right One?
https://www.pixelcrayons.com/blog/php-vs-asp-net-how-to-choose-the-right-one/
(Sidst besøgt 12/04/2022)

What is the difference between bcrypt and SHA256?
https://rietta.com/blog/bcrypt-not-sha-for-passwords/
(Sidst besøgt 12/04/2022)

What is Entity Framework?
https://www.entityframeworktutorial.net/what-is-entityframework.aspx
(Sidst besøgt 12/04/2022)

What are Progressive Web Apps?
https://web.dev/what-are-pwas/
(Sidst besøgt 12/04/2022)

Angular vs Vue: Which Framework to Choose in 2022?
https://www.simform.com/blog/angular-vs-vue/
(Sidst besøgt 12/04/2022)

Blazor vs Vue
https://www.telerik.com/blogs/blazor-vs-vue-web-developers
(Sidst besøgt 12/04/2022)

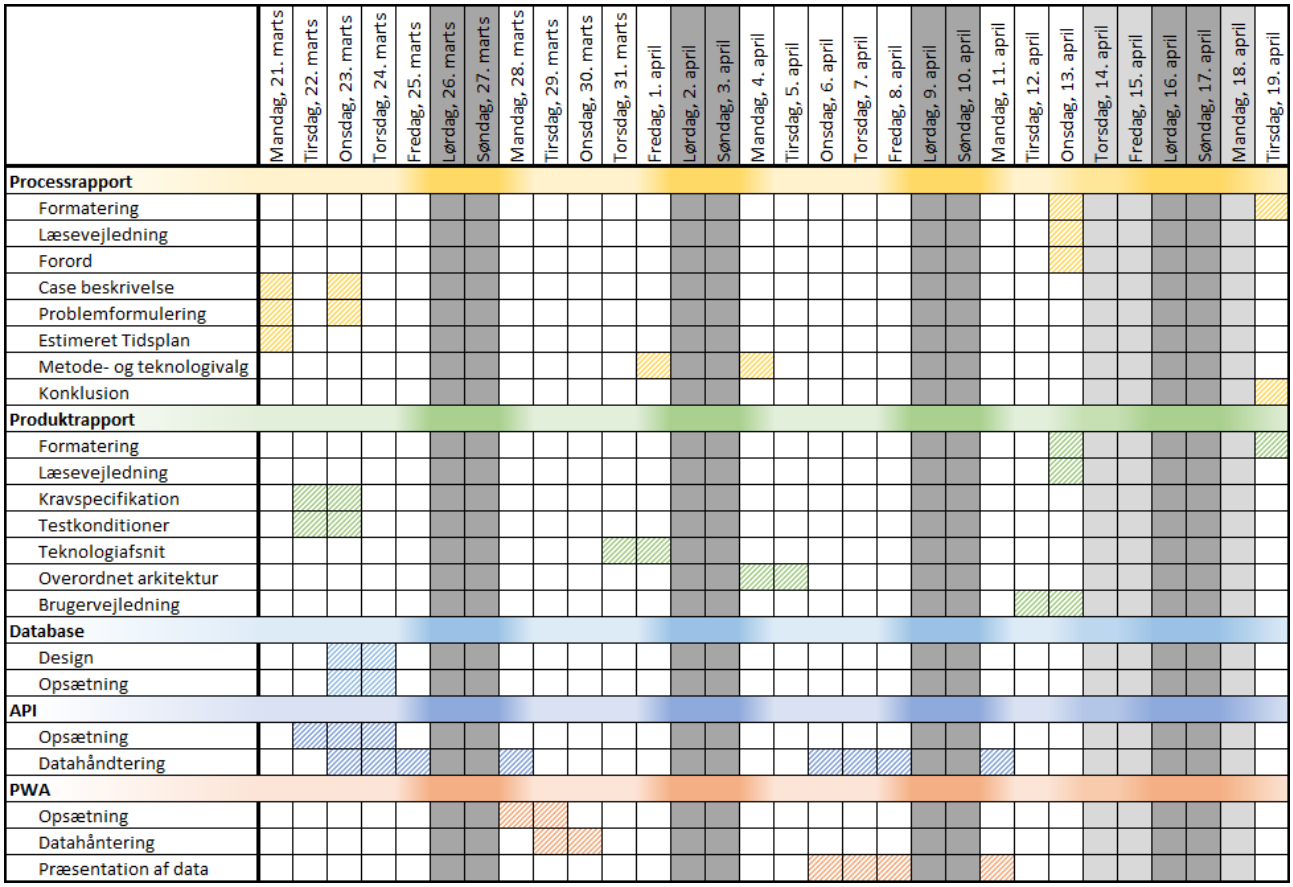
Vue.js vs React: Comparison of Two Most Popular JS Frameworks
https://www.codica.com/blog/react-vs-vue/
(Sidst besøgt 12/04/2022)

SVN vs Git: Which One Is Best for Your Needs?
https://blog.hackbrightacademy.com/blog/svn-vs-git/
(Sidst besøgt 12/04/2022)

What is Microsoft Azure: How Does It Work and Services
https://www.simplilearn.com/tutorials/azure-tutorial/what-is-azure
(Sidst besøgt 12/04/2022)

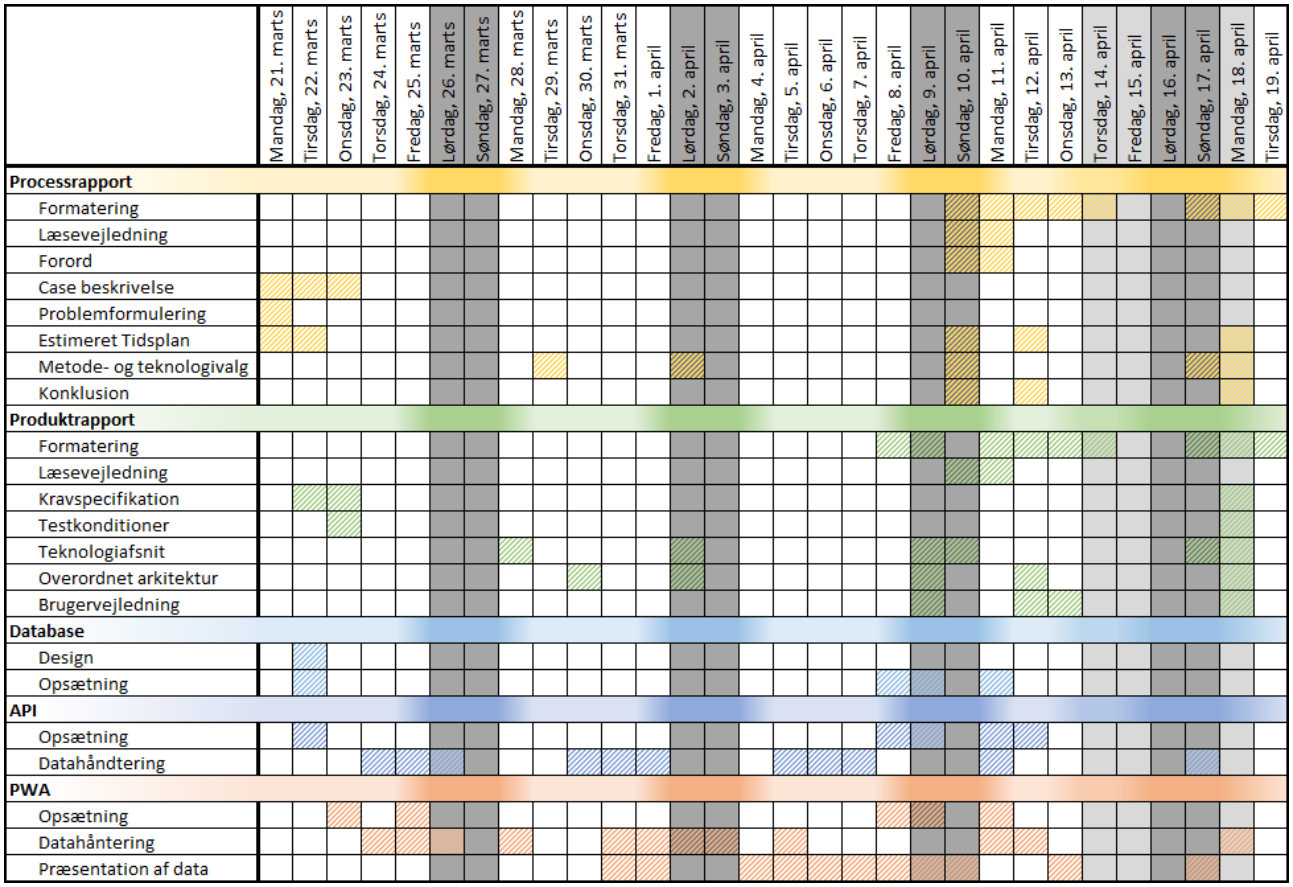
Bilag 1

Estimeret Tidsplan



Bilag 2

Realiseret Tidsplan



Bilag 3
Logbog

Uge 1

Mandag, 21. marts

Opstart til svendeprøve. Ud over praktisk information har vi arbejdet med Case beskrivelse, problemformulering, og tidsplan.

Over middag gik mest med information fra Dansk Metal.

Arbejdssted: Skolen, lokale C105.

Arbejdstimer: 6 (2 timers Dansk Metal besøg medregnet, da deltagelse var obligatorisk).

Tirsdag, 22. marts

I dag startede jeg med lige at kigge min tidsplan igennem, og gik ellers i gang med at skrive kravspecifikation.

Jeg kom også i gang med mit API, og fik sat en database op derigennem med Entity Framework, samt lavet en overordnet plan for hvilke endpoints jeg skal bruge.

Sidst på dagen gik jeg tilbage til min case beskrivelse og finpudsede den lidt, før den skal godkendes i morgen.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 6:25 - 14. Minus frokost bliver det omkring 7.

Onsdag, 23. marts

I dag arbejdede jeg på skolen, da vi skulle have godkendt vores case beskrivelser og problemformuleringer.

Ud over at arbejde med dem, fik jeg kigget mere på min kravspecifikation, og ellers startet med at sætte min PWA op.

Arbejdssted: Skolen, lokale C105.

Arbejdstimer: 6.

Torsdag, 24. marts

I dag har jeg arbejdet på at få gjort de API endpoints jeg skal bruge her i starten af mit projekt så klar som muligt. Det drejer sig om endpoints i UserController'en som blandt andet omfatter at hashe og verificere hashede passwords, og de endpoints i RunController og PointController der specifikt har med at oprette nye ture og punkter at gøre. Det er vigtigt at få dette klar så hurtigt som muligt, så jeg kan begynde at samle data som jeg senere i projektet kan vise frem.

Efter dette skiftede jeg gear og kiggede på at få gjort de dele af min PWA der bruger de endpoints klar, uden noget styling eller noget fancy lige her til at starte med. Lige nu er funktionaliteten prioriteten!

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: Omkring 7, igen.

Fredag, 25. marts

I dag har jeg slåsset i timevis med at finde en måde at få min PWA til at kunne arbejde i baggrunden når en telefon er låst. Det lod sig til allersidst gøre med en web worker, men at finde en der kunne med Vue 3 var en større udfordring.

Ud over det har jeg fået Vuex til at virke, så jeg kan finde login token fra andre sider og sikre at folk ikke kan kalde API'en uden.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: lidt over 8.

Lørdag, 26. marts

Prøver at holde weekend, men går lidt til og fra projektet.

Hoppede på i morges for at rename en masse ting i mit API jeg ikke havde camelcaset til at starte med, fordi det er lang tid siden jeg har programmeret C#.

Ellers har jeg slåsset med den web-worker og prøvet at få den til at ville dø når jeg vil have den til det.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: Nok omkring 3.

Søndag, 27. marts

Prøvede at holde weekend og berolige mig selv i forhold til at det hele nok skal gå selvom jeg sidder med noget kode der ikke virker.

Uge 2

Mandag, 28. marts

Fik i løbet af de første par timer endelig min web-worker til at makke ret. Det var ikke den, den var gal med, men måden jeg fik koordinater på, der hele tiden startede nye jobs. Der burde være styr på det nu.

Nu hvor jeg kan indsamle data er jeg gået videre til at skrive mine store metodeafsnit, og planen er at få dem mere eller mindre færdige i den her uge, så jeg kan vende tilbage til koden og få skrevet det som rent faktisk fremviser dataene.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 7.

Tirsdag, 29. marts

I dag har jeg brugt dagen på at få langt det meste af mit afsnit om metodevalg i procesrapporten skrevet.

Arbejdssted: Skolen, lokale C105.

Arbejdstimer: 6.

Onsdag, 30. marts

I dag ville jeg gribe fat i afsnittet om mit projekts overordnede arkitektur i produktrapporten, men det viser sig dog at det er vanskeligt at skrive før det er kodet. Jeg fik skrevet den halvdel jeg har kodet, og derefter begyndte jeg at kode de endpoints færdige som anden halvdel benytter.

Nu er mine endpoints mere eller mindre færdige, bortset fra noget logik på serversiden, som er mindre vigtigt end at få gjort PWA'en i stand til snart at fremvise noget data.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 7.

Torsdag, 31. marts

I dag startede jeg med at arbejde på at få gjort det endpoint der skal generere data mere færdigt, og gik derefter over til at arbejde på PWA'en med visning af en brugers ture.

Man kan nu slette ture fra listen, som så bliver flagget som slettede, men ikke rent faktisk slettede i databasen, og man kan få nogle af de statistikker jeg vil kunne vise frem om en given tur. Dagen kulminerede med at jeg fik et kort til at virke på siden for en given tur, så nu skal jeg have vist turen på selve kortet.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 7.

Fredag, 1. april

I dag har jeg fået kortet helt til at virke, og fremvise ruter, samt start- og slutpunkter. Ydermere har jeg fået deployet appen til Firebase og kontrolleret at den kører tilfredsstillende på telefonen.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 7.

Lørdag, 2. april

Valgte at bruge dagen på at gennemlæse og rette de tre store teoriafsnit jeg skrev i den foregående uge.

Ud over dette havde jeg appen med ude at løbe på den første store test, og det gik mindre end ideelt. Tror at topprioritet mandag, eller måske allerede søndag, må blive at få forsendelsen af punkter strømlinet så de ikke kommer frem i en stor pærevælling.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 4.

Søndag, 3. april

Jeg kunne ikke dy mig, og jeg måtte have mit system til at virke, så fra morgenstunden af skrev jeg koden der sender koordinater om fra at bruge `navigator.geolocation.getCurrentPosition()` med et interval, til i stedet at bruge `navigator.geolocation.watchPosition()`.

Man skulle næsten tro at dem der fandt på de metoder, vidste hvad de havde gang i, for da jeg senere testede systemet på turen til og fra min lokale købmand, målte og sendte den koordinater fuldstændig upåklageligt.

Min kæreste testede også systemet på en gåtur senere på hendes telefon, som havde haft mærkværdige problemer med systemet i da det brugte `getCurrentPosition()`, og det virkede som det skulle med den nye metode.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 1,5.

Uge 3

Mandag, 4. april

I dag har jeg arbejdet på styling af login-, registrerings-, mine løbeture, og informations-siderne. Der er stadig en del at komme efter, men nu begynder systemet også at ligne noget.

Jeg har også tilføjet funktionalitet til at systemet lagrer brugerens token på tværs af sessioner, så man ikke længere bliver logget ud af at genindlæse siden.

Det var så sjovt at se stylingen komme på, at jeg ikke kunne lægge det fra mig da jeg kom hjem.

Arbejdssted: Skolen, lokale C116 (grundet gulvbehandling i lokale C105), og senere hjemme, Bjerringbro.

Arbejdstimer: 10.

Tirsdag, 5. april

I dag har jeg arbejdet på rigtig mange forskellige ting. Det første jeg gjorde, var at få implementeret wakeLock, og få diagrammer til at virke på appen, som jeg sad og boksede lidt med i går aftes.

Ydermere har jeg fået stylet siden til at oprette nye løbeture, med kort hvor man kan følge turen mens man løber, samt tid på ens tur. Ud over det har jeg pillet ved stylingen på mange af de sider jeg bearbejdede i går, og fået gjort alting mere lækkert.

Sidst, men ikke mindst, har jeg kodet udregningen af gennemsnitshastighed per minut i mit API.

Jeg kunne igen ikke lægge arbejdet fra mig i dag, og det blev endnu en sen en.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 13.

Onsdag, 6. april

Jeg startede dagen med at få implementeret det sidste diagram for gennemsnitshastighed per minut på min app, så nu er den ved at være i hus med funktionalitet, og alt andet jeg kan nå udover, er bare sjovt at have med.

Derefter spildte jeg nok over en halv dag med at forsøge at få unit testing implementeret på mit API. Det endte med at jeg måtte opgive og gå over til black-box testing i stedet. Det er mit håb at dette stadig kan sættes op med continuos integration, men nu må vi se.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 12.

Torsdag, 7. april

I dag har jeg arbejdet færdig med test, og jeg har nu en test for hvert endpoint, der kalder det og kontrollerer at jeg får den rigtige information tilbage.

Ud over det, har jeg stylet lidt mere, og ellers brugt tid på at tegne et pænt logo som jeg kan bruge både til appen og min rapport. Dette har jeg yderligere implementeret på siden som en komponent, så jeg også viser noget Vue.js funktionalitet der.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 9.

Fredag, 8. april

I dag startede jeg med at få favicon, app ikon, og lidt andet lækkert lagt ind i buildprocessen af appen. Derefter fik jeg sat op så der automatisk deployes til Firebase når jeg pusher til Git.

Jeg fik også skrevet på afsnittet om krav til projektet og hvordan jeg opfylder dem i min produktrapport.

Da jeg kom hjem, fik jeg lavet logoet helt færdigt, nu hvor det var implementeret og jeg var lidt klogere på hvordan det så ud i appen.

Ud på aftenen gik jeg i gang med at lære hvordan man deployer til Azure og det viste sig at blive et større eventyr, og jeg kommer mere end en time for sent i seng, med kode der ikke virker.

Arbejdssted: Skolen, tilbage i lokale C105, og senere hjemme, Bjerringbro.

Arbejdstimer: 13.

Lørdag, 9. april

Startede dagen med at løse de problemer jeg sad med til sent i går aftes, og fik deployet til Azure.

Nåede desværre ikke at få det hele til at køre på Azure før dagens løbetur, så det blev på test appen, men nu skulle vi være klar til at samle data på den rigtige.

Efter løbeturen investerede jeg lidt tid i at få GitHub actions til at virke på Azure, så jeg nu har Continouos Deployment. Hver gang jeg pusher til GitHub, bliver både API og PWA deployet i deres nye versioner. Jeg har testet begge ved at ændre noget åbenlyst, pushe, se ændringen uden at deploye noget manuelt, ændre tilbage, pushe, og se ændringen forsvinde igen.

Continous Deployment er sygt lækkert, det er ærgerligt at mine tests ikke egner sig til Continous Integration.

Jeg har ordnet et par småting i appen, og eller skrevet produktrapport. Jeg besluttede mig for at den bedste måde at komme den til livs på, var at starte fra en ende af, så det gjorde jeg. Alle sektioner er blevet læst igennem, rettet og tilføjet til, og jeg har fået skrevet brugervejledninger. Har dog mange spørgsmål om forskellige rapportting jeg skal have svar på mandag.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 14.

Søndag, 10. april

I dag har jeg skrevet procesrapport, same procedure as yesterday. Jeg startede fra starten og fyldte på hvor der manglede hele vejen igennem.

Derudover har jeg fikset småting i appen og testet løsningerne i live versionen på Azure.

Fik også lige, i et øjebliks genialitet, gjort måden kortet finder hvad der skal vises, når du ser en færdig tur, meget bedre og mere dynamisk. Så det var fedt.

Sidst på dagen gik jeg i kast med at dokumentere min kode, som er noget jeg kun har gjort løbende for mig selv, men gerne vil gøre mere detaljeret før aflevering.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: Jeg tog den lidt med ro i dag, så det blev kun til 12.

Uge 4

Mandag, 11. april

Fik stillet alle mine dumme spørgsmål og fik vejledning til rapportskrivning.

Fik dokumenteret mine testcases, lavet fem nye så jeg er oppe på 12 black-box tests, og dokumenteret min PWA kode.

Da jeg kom hjem rehostede jeg min løsning på Microsoft Azure med en betalt konto, så den nu er samlet i Norge, i stedet for spredt rundt i Europa. Dette er forhåbentlig den endelige løsning, og jeg håber på ikke at skulle starte med tom database igen.

Efter det fik jeg skrevet og testet username og password validation, på både front-end og back-end. Front-enden er hurtigere til at give besked til brugeren, og back enden er for ekstra sikkerhed.

Til allersidst fik jeg lige kigget på modtagelsen og håndteringen af tider på PWA siden, og sikret at de bliver konverteret korrekt fra UTC til lokal tid.

Arbejdssted: Skolen, lokale C105, og senere hjemme, Bjerringbro.

Arbejdstimer: 12.

Tirsdag, 12. april

I dag har den stået på rapportskrivning, og at få dem klar til i morgen eftermiddag/aften hvor jeg gerne vil kunne sende dem ud i god tid til dem der vil hjælpe med at læse dem igennem.

Ydermere, nu hvor jeg har unit tests til serverside password valideringen, kunne jeg få sat det op i GitHub Actions med Continouos Integration, så det gik der også lidt tid til at kæmpe med, men det skulle køre nu.

Senere kom jeg i gang med at få skrevet de brugertestcases, og jeg tror at de blev gode. Jeg fik også fikset et par ting i API og skrevet et par nye black-box tests.

Jeg arbejder mange steder rundt i projektet hele tiden, så det er svært at få det hele med, men det er de store ting.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 12.

Onsdag, 13. april

Startede dagen med at fikse så diagrammer på infosiden viser nogenlunde brugbar information, selv hvis en løbetur er meget kort, og ikke har data der egner sig til lange diagrammer.

Så fik jeg stillet de sidste dumme rapportspørgsmål, og gik i gang med endelig formatering af begge.

Jeg fik taget nogle screenshots af appen til brugervejledninger og sat dem ind.

Jeg tog mig tid til at prøve samtlige af mine brugertestcases igennem, og sikre at de kunne gennemføres som de stod, og gav de korrekte resultater. Lidt vildt at jeg kendte systemet godt nok til at skrive dem uden at prøve samtidig, og at der ikke var noget at ændre da jeg gjorde, men det er trods alt mit system.

Sidst på dagen fik jeg sendt rapporter ud til gennemlæsning. Nu venter jeg på rettelser.

Arbejdssted: Skolen, lokale C105.

Arbejdstimer: 6.

Torsdag, 14. april

I dag har jeg modtaget første omgang rettelser til begge rapporter, som tog overraskende lang tid at gennemgå og taget stilling til.

Ellers har jeg arbejdet med formatering, men prøvet at tage den lidt med ro, siden jeg har haft nogle ret lange dage sidste uge og denne.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 8.

Fredag, 15. april

Jeg sagde til mig selv at jeg burde holde en fridag i dag. Ikke at jeg af den grund var succesfuld i at tænke på andet end mit projekt, men jeg gjorde mit bedste.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 0.

Lørdag, 16. april

Mens jeg venter på at andet sæt rettelser kommer tilbage, har jeg i dag startet arbejdet på at planlægge og min fremlæggelse og lave PowerPoint.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 6.

Søndag, 17. april

I dag fik jeg andet set rettelser i hus, så dem fik jeg rettet i mine rapporter.

Ydermere fik jeg strømlinet måden appen authenticater brugere på, når siden genindlæses, og arbejdet mere med mine PowerPoint slides.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 8.

Uge 5

Mandag, 18. april

I dag har jeg læst mine rapporter igennem en sidste gang, så det eneste der skal gøres i morgen før aflevering, er at stille de sidste dumme spørgsmål og ændre hvad der skal ændres, og så den sidste formatering. Det er sådan noget som lige at dobbeltkontrollere sidetallet står som det skal, og at opdatere indholdsfortegnelser.

Derudover har jeg arbejdet videre på mit fremlæggelses PowerPoint, og lavet nogen meget små app-ændringer, der bare strømlinende en ting eller to.

Arbejdssted: Hjemme, Bjerringbro.

Arbejdstimer: 10.

Tirsdag, 19. april

I dag skal der afleveres rapporter, så dette er farvel.

Dagen går med at få hjælp til de sidste små formalia, og så ellers gå igennem og rette de sidste småting, indholdsfortegnelser og sidetal.

Arbejdssted: Skolen, lokale C105.

Arbejdstimer: 3, ved aflevering.