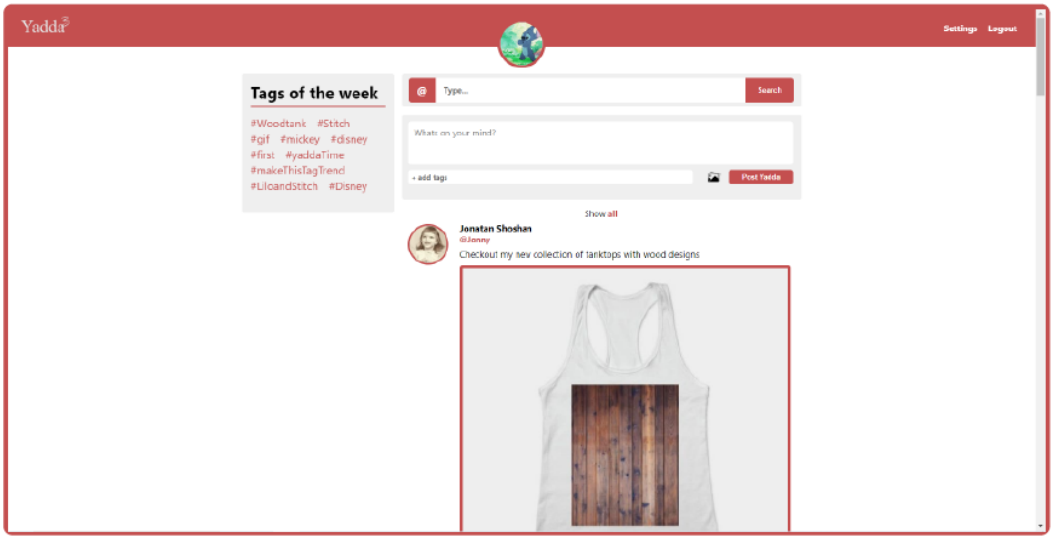
YaddaYaddaYadda

YaddaYaddaYadda
@yadda3

Projekt af Simon Kalmar Riis Mortensen, Christian Kjer Bang og Jonatan Shoshan.
Antal sider af tegn med mellemrum: 14,85 sider.



#projekt #yaddayaddayadda #yadda3

6 Likes **2 replies** **24 MAY 2020 - 23:59**

Webhosting:
https://yadda3.herokuapp.com/

Indledning

(Simon)

I denne rapport vil der blive givet en beskrivelse for processen ved udviklingen af en social-media hjemmeside kaldet YaddaYaddaYadda og hvad tankerne bag hjemmesiden var. Denne proces indeholder både eksempler fra research fasen, designfasen og selve programmeringsfasen af funktionerne. Hver fase vil blive beskrevet på en måde, så der kan ses tankegangen med de forskellige valg, som er blevet gjort i løbet af projektets forløb.

I research fasen vil der blive kigget på hvordan andre sociale medier virker, og hvilke valg de har lavet i deres konstruktion af deres hjemmesider. Der vil også være en liste over forskellige overvejelser som er blevet lavet i forbindelse med udviklingen af hjemmesiden. Disse forskellige tanker gik derefter videre til de senere faser, så projektet kunne blive det bedste, som det kunne være.

Senere i designfasen, så vil der blive kigget på hvordan design af logo og hjemmeside vil give siden en identitet, som adskiller sig fra andre sociale medier som Twitter og Facebook.

For programmeringsfasen, så vil der være beskrivelser og forklaringer for hvordan forskellige funktioner er blevet skabt, hvordan hele siden blev sat op og hvordan enkrypteringen af passwords er blevet lavet.

Ved at læse denne rapport vil der komme et fuldt billede over alle de valg, som er blevet taget under hele projektforløbet.

Problemformulering

(Christian)

I dette projekt vil gruppen udarbejde et funktionelt og brugervenligt “social media website”, med navn “YaddaYaddaYadda”. Applikationen skal skrives i Node.js med brug af web-interfacet Express. Databasen skal være en dokumentdatabase af typen MondoDB, og der skal bruges mongoose til håndtering af denne i programmellet.

Hvordan kan der udarbejdes en løsning som besidder de stillede krav til Yaddas, samt sikre at brugen af Tags kan benyttes til navigationen af Yaddas?

På hvilken måde kan brugerrejsen fra Registrering til Login samt første interaktion blive skabt på, på en måde som benytter sig af hashing af password og godkendelse via email?

Hvordan kan sitet blive designet så der både er plads til annoncer, samt den konfigurerings-mulighed, at brugeren kan vælge mellem mindst 2 temaer (et mørkt og et lyst)?

Metodeovervejelse

(Jonatan & Christian)

Tidsplan: Der er blevet valgt at lave en tidsplan, for at sætte nogle vigtige deadlines for projektet, samt sikre at projektet bliver gennemført til tiden.

Scrum: Der er valgt at lave daily scrum sektioner over skype, samt scrumboards i Trello, for at holde styr på hvor langt vi er, og hvor meget vi mangler.

Research af kode: Research skal bruges til at undersøge, hvordan eventuelle problemer kan blive løst på, samt hvilke værktøjer der ville være bedst egnet til dette projekt.

Research af Platforme: Da opgaven lyder på at designe en social medie side, har vi valgt at undersøge andre succesfulde platforme, for at søge inspiration af deres mange års erfaring og ekspertise.

Spørgeskema: Et spørgeskema har været overvejet af flere omgange, men blev til sidst droppet. Dette skete pga. at vi var bange for, at hvis vi sendte et spørgeskema ud på de sociale medier, så ville vi kun fange dem som i forvejen er tilfredse med de nuværende sociale medier.

Must Have / Nice to Have / Never: Der er blevet valgt at bruge denne metode, til at kategorisere hvilke elementer YaddaYaddaYadda skal implementere, hvilke elementer der ville være relevante at implementere og hvilke som ikke høre hjemme på platformen.

SWOT/TOWS: For at finde ud af hvem vores målgruppe er, og hvor der er et markede for YaddaYaddaYadda, blev der valgt at lave en SWOT/TOWS analyse, som kategorisere hvor YaddaYaddaYadda kan specialisere sig.

Sketching & Wireframe: Der blev valgt at gøre brug af Sketching & Wireframe, da disse metoder giver alle gruppens medlemmer muligheden for at tolke og eksperimenter, uden at være farvet af andres perspektiv. Dette giver nye og spændende ideer muligheden for at blomstre.

Prototype: Der blev valgt at lave prototyper af de forskellige logoer og af sidernes opbygning og design. Dette blev gjort i Photoshop og XD Design. Dette skulle være en mere samlet og raffineret udgave af de tidligere Sketches.

Test: Gruppen har haft diskuteret forskellige måder at udføre en omfattende test på. Men dette viste sig at være for problematisk, da vi ikke rigtigt havde muligheden for at møde og test på fremmede.

Stack: Vi har valgt at bygge applikationen op omkring nodejs med serverside rendering, hvilket skyldes fokus på læringsprocessen og muligheden for at se hvilke fordele og ulemper

der er ved denne opbygning. Der kan dertil argumenteres for at det ville være fordelagtigt at rykke flere af processerne fra serveren til klienten, samt muligheden for bedre UX og skalering ved en single-page opsætning med et dertilhørende API.
Vi har gjort brug af mongoDB, da det har indbygget muligheder for skalering, samt nem struktur til senere at tilføje nye features, efter behov.

Research

Research af populære medie platforme

(Christian)

Først og fremmest overvejede gruppen at lave et spørgeskema, som skulle indsamle information på hvilke sociale medier der er mangler på, samt hvilke elementer folk synes er essentielle for en funktionel social medie side.

Denne ide var dog hurtigt droppet, da der var for mange problemer ved at få dannet et brugbart og nuanceret resultat. Der var problemer med at få skabt spørgeskemaet til YaddaYaddaYadda, så svarene på spørgeskemaet ville være relevante i forhold til tilpasset til at klientens ønsker.

En anden udfordring var hvordan vi kunne nå en specifik målgruppe. For hvis vi f.eks delte spørgeskemaet til unge på facebook, ville vi kun få svar fra brugere som har vænnet sig til facebook osv.

Det blev derfor konkluderet at et spørgeskema nok ikke ville være særligt brugbart, på dette tidspunkt i processen. Derfor blev det i stedet besluttet at søge inspiration fra de UX Patterns som andre succesfulde virksomheder har noget frem til, ved at eksperimentere, teste og undersøge deres bruger igennem årene.

Gruppen undersøgte, Facebook, Youtube, Tumblr, Reddit, Instagram, Twitter.

Gruppen havde et fokus på at finde ud af hvordan YaddaYaddaYadda kunne være konkurencedygtige overfor de andre platforme. Hvilke ligheder, fordele og ulemper YaddaYaddaYadda ville havde overfor de andre.

Nedenfor er 3 eksempler på hvordan denne information blev noteret, disse 3 blev valgt som eksempler pga. At de er populære, forskellige fra hinanden og har en vis lighed til YaddaYaddaYadda.

Facebook:

Denne side giver brugeren muligheden for at lave et opslag på maks 63.206 tegn. Et opslag kan laves på brugerens egen tidslinje, andres profiler, ens egne og andres grupper samt sider. Dog forekommer der ofte, at man enten begrænsninger af disse muligheder.
F.eks. kan en gruppe gøre at kun medlemmer af gruppen kan lave et opslag, eller en bruger

kan en bruger vælge, at det kun er brugerens venner som kan skrive på vedkommendes profil.

Brugeren har et bredt udvalg af forskellige baggrunde, som brugeren kan tilføje til sit opslag. Det er muligt at tilføje Facebook specifikke Emojis, GIF's, billeder og videoer.

I opslaget er det muligt at tagge ens Facebook venner, skrive et # for at lave et andet tag, tilføje ens lokation, eller tilføje ens følelse eller ens aktivitet.

Opslaget kan også gøres til en livevideo, en videosammenkomst eller en forspørgelse.

Herudover har brugeren muligheden for at hvem opslaget skal vises til. Det kan bl.a være offentlig, for udvalgte venner, eller kun for en selv.

Når et opslag er blevet posted, kan dette opslag kommenteres, deles, eller synes godt op.

Herudover har facebook også en bred vifte af andre måder at interagere med andre på, f.eks Spil, begivenheder og markedsplads. Her er deres messenger service nok den mest mærkbare, da den tilbyder mange af de samme muligheder som der findes ved facebooks grupper/sider og opslag.

Facebook har også mange forskellige måder at vise annoncer på. De har annoncer i sidelinjen, i mellem facebook opslag, før, efter og midt i facebook videoerne, imellem salgsvarende på markedspladsen og før man spiller et af deres spil. m.m.

Instagram:

Denne side giver brugeren muligheden for at uploade et eller flere billeder eller videoer. Her kan der tilføjes en tekst, samt tags lokation og tags af andre Instagram profiler. 'Instagram har en karakter limit på 2200 tegn.

Hvert billede kan kommenteres, likes, deles og gemmes. Kommentarerne kan dog kun være tekst, tags og Emojis, det er ikke muligt at kommentere med et billede eller en video.

Reklamerne forekommer i mellem de forskellige instagram post.
det er muligt at se alle de billeder og videoer som en bruger har lagt op, samt alle de billeder og videoer.

Instagram har også en indbygget Chat funktion.

Twitter:

Denne side er nok den som minder mest om YaddaYaddaYadda, da den også har en meget stor begrænsning på tegn per opslag. Et Tweet kan makt havde 280 tegn, men det plagede kun at være 140 tegn.

Twitter giver brugeren muligheden for at kommentere, like, re-tweete og dele.

Når brugeren klikker på en tweet, ser man tråden af tweets som i YaddaYaddaYadda.

Twitter kommer også med forslag til hvilke brugere man kan følge og hvilke tags som trenter. Derudover har Twitter også en mulighed for at skrive til andre brugere direkte.

Must have, Nice to have, Never

(Christian)

Først lavede vi en Must Have, som består af klientens krav/ønsker. Herefter blev der gjort brug af de observationer, som gruppen havde noteret via undersøgelsen af de andre sociale medie sider, til at udfylde hvilke funktioner der ville give mening for YaddaYaddaYadda, og hvilke der ikke ville.

Dem som gav god mening blev noteret i Nice to have, og dem som ikke gav mening blev noteret i Never. Samt med en kort begrundelse, for hvorfor valget blev truffet.

YaddaYaddaYadda:

Must have

Denne side giver brugeren muligheden for at skrive en besked (Yadda) på maks 167 tegn, og evt. indeholde et billede. Enhver Yadda vises med ”handle”, miniudgave af forfatterens avatar, dato og tid for oprettelsen.

En bruger kan følge andre brugere, og vil derefter se de forskellige Yadda fra de brugere de følger. De forskellige Yadda vil blive vist i kronologisk rækkefølge, de nyeste Yadda først.

En Yadda kan også indeholde Tags. Brugeren har muligheden for vise Yadda som kun indeholder et bestemt tag. Hvis flere Yadda indeholder dette tag, vil de også blive vist i kronologisk rækkefølge, de nyeste Yadda først.

Brugeren har muligheden for at se en liste over hvem brugeren følger, samt hvem der følger brugeren.

Det er også muligt at besvare en Yadda, og derved danne en tråd. Det er muligt at se hele tråden af beskeder, ved at klikke på en Yadda.

Brugeren har også mulighed for at customisere deres profil, ved at vælge eget navn, billede og tema.

Det er muligt at sælge annoncer, som bliver fremvist på siden.

Nice to have

Da YaddaYaddaYadda i forvejen er en platform, som giver brugere mulighed for at dele billeder, vil det også give god mening også at kunne dele video, da de andre platforme tillader som tillader billeder også har haft stor succes med video, i en form eller en anden (film, Youtube, GIF's, Loops m.m.).

En Like funktion, er blevet en essentiel del af sociale medie sider. Denne funktion giver brugen nyttig og vejledende feedback, og YaddaYaddaYadda ville sansynligtvis føgles mangeltfuldt uden.

Separation af Tags og Kommentare. En af YaddaYaddaYadda's udfordringer er, at der er en streng begrænsning på de Tegn, som der må være i en Yadda. Youtube er den eneste af de store sociale medie platforme, der har en mindre tegnbegrænsning end YaddaYaddaYadda. Youtube tillader kun 100 tegn, men har tilgengæld Tagsne separat fra overskrifterne. Dette kunne også være en brugbar løsning for YaddaYaddaYadda. (Tumblr har også Tags og Tekst separat, selvom Tumblr har en karakter begrænsning på hele 500.000 tegn).

Mulighed for at Skrive Tags på andres brugers sider, eller dele Yaddas med andre (på YaddaYaddaYadda, eller andre sociale medie sider.)

En menu som hvilke Tags som trenter i øjeblikket, eller hvilke andre brugere som ville være relevant for brugeren at følge.

Notifikationer kunne også være relevant i fremtiden.

Never

Efter som Twitter, Youtube, Instagram, Reddit, Facebook m.fl. alle har tilføjet en måde at skrive private kommentare til hinanden, og derved undgå den den primære (offentlige) måde at kommunikere på. Ville dette være en måde YaddaYaddaYadda kunne skilde sig ud på, da siden i så fald vilde kunne appellere til brugere som vil undgå at blive kimet ned af privat beskeder.

Livestreams, Computerspil, Auktionshus, m.m. YaddaYaddaYadda specialisere sig i at komunikere med forskællige Yaddas. Derfor vil fællesskaber, fandome, og reklamer alt sammen Yaddas som omdrigningspunkt.

SWOT / TOWS

(Christian)

Det blev besluttet at det var relevant at lave en SWOT/TOWS analyse, som først ville vise hvilke Styrke, Svaheder, Muligheder og Trusler YaddaYaddaYadda har over for deres konkurrenter. Derefter viser modellen hvor YaddaYaddaYadda skal putte sit fokus, for at være konkurrencedygtige med de andre platforme.

SWOT

Strengths	Weaknesses	Opportunities	Threats
Er et hurtigt og let fordøjeligt medie. YaddaYaddaYadda vil være meget unik i at alt information er offentligt. Det er nemt at finde ud af hvad folk med fælles interesse synes om et emne.	YaddaYaddaYadda er meget specialiseret. Alt kommunikation er gives gennem Yaddas. (Ingen alternativer).	Mulighed for at forme tætte fællesskaber, blandt sidens brugere. Muligheder til at bygge videre på siden i fremtiden. Det er muligt for selskaber at reklamere i Yaddas.	Andre Medie sider: (Facebook, Youtube, Tumblr, Reddit, Instagram, Twitter). Andre meddelelser services som f.eks. WhatsApp, Skype. Andre desideret “comunity” sider/ Apps som FANDOM & Amino

TOWS

	Strengths	Weaknesses
Opportunities	Eftersom YaddaYaddaYadda er et hurtigt og let fordøjeligt medie, er det muligt markedsføre siden til folk som ikke har meget tid til at danne tætte fællesskaber. Det er nemt at finde ud af hvad medlemmerne af de forskellige fællesskaber synes om et specifikt emne. (f.eks. hvis et nyt album, eller netflix afsnit udkommer).	YaddaYaddaYadda er meget specialiseret, og simpel. Dette give os muligheden vente med at udvide målgruppen indtil den rette målgruppe er blevet fundet. Herefter kan siden forme sig efter brugernes behov. YaddaYaddaYadda giver selskaber muligheden for at reklamere via Yaddas. (Billeder, videoer og Tekst). Dette minder meget om Instagrams model, men er meget begrænset i forhold til Facebook.
Threats	YaddaYaddaYadda har mange konkurrenter, og mange af dem har tilføjet mange af hinandens funktioner. Dette betyder også at de alle har tilføjet en mulighed for kommunikere med andre privat. Dette kunne være et sted hvor YaddaYaddaYadda skilder sig ud fra mængden. Eftersom	Der er mange andre etablerede platforme, som har kæmpet mod hinanden, om folks opmærksomhed i årevis. Alt kommunikation på YaddaYaddaYadda vil foregå i Yaddas, inklusivt reklamer. Dette kan sandsynligvis, være for meget af en begrænsning for nogle af

	YaddaYaddaYadda vil være en platform hvor alle skal stå ved det man skriver, og ingen har muligheden for at gemme sig væk i en privat chat.	platformens potentielle brugere.

Konklusion af SWOT og TOWS

Det ville give mening for YaddaYaddaYadda's at fokuser på en målgruppe, der har travlt og som måske ikke får dækket deres sociale behov i hverdagen. Eller på anden vis har mange små pauser i løbet af dagen. Platformen bør markedsføre sig som stedet, hvor man nemt kan blive en del af et fællesskab. Samt hurtigt finde ud af, hvad et fællesskab synes om et relevant emne. (Der er dog meget konkurrence på dette område).

YaddaYaddaYadda skal prøve at vinkle deres specialisering (f.eks manglen af en privat chat funktion) som noget positivt. Derudover skal platformen laves på en måde, så der altid vil være muligheden for at bygge videre på den i fremtiden. Hvilket kan gøre siden mere relevant overfor brugerne og de virksomheder, som kunne overvejer at have annoncer på platformen.

Analyse

Design af Logo

(Christian)

Gruppens medlemmer designede hvert deres bud på et logo. Dette skete først ved at lave nogle skitseringer, hvor der derefter blev udvalgt nogle eksemplarer, som blev genskabt i en mere defineret udgave. Den store udfordring var at få det lange navn YaddaYaddaYadda, designet til at være mere branding venlig.

Den første ide blev lavet på som en skygge effekt. Ideen var at vise tråden, med de tidligere Yaddas som fadede mere og mere ud i baggrunden. Dette lykkedes ikke helt, da kom til at virke mudret.
3 forskellige farver var tilføjet, for at gøre det nemmer at skelne mellem de forskellige design. Uheldigvis, fremstod logoet nu mere barnlig.

Den anden idé, var at gøre lidt af det samme som have den første, men prøve at gøre det med en 3D effekt i stedet for skyggeeffekten. Her løb vi igen ind i det samme problem, med at det ikke virkede som om at det var designet til den rette målgruppe.

Det tredje og sidste design droppede illusionen, og fokuserede mere på at være simpelt og stilren. Navnet i logoet var dog stadigt kun Yadda, men der blev tilføjet et opløftet tretal, for at vise at noget skulle gentages 3 gange. Der blev lavet en lys og en mørk udgave eftersom siden ville have en lys og en mørk udgave. Dette er også det logo som vi endte med at gå med. Men den lyse udgave af logoet endte ikke med at blive brugt.

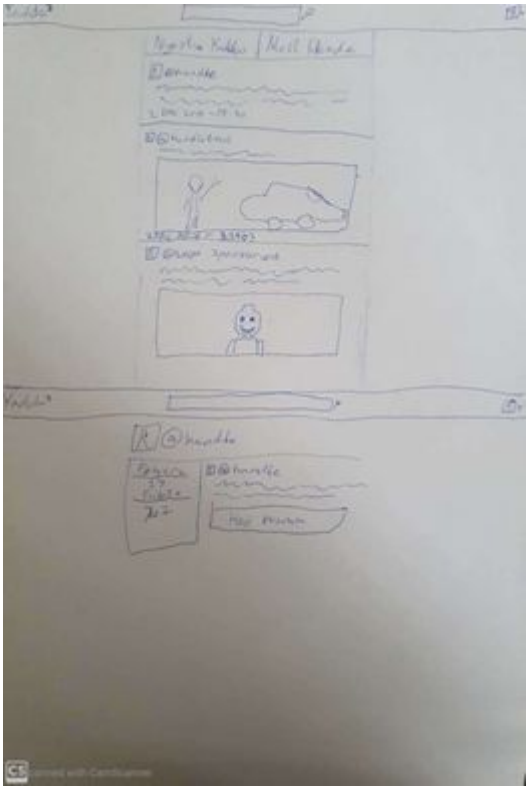
Wireframes – og det at komme med et design

(Simon)

YaddaYaddaYadda skulle blive designet, så det var nemt at bruge for brugeren og så forskellige aspekter af hjemmesiden ville kunne bruges dagligt for brugeren. Elementerne og funktionerne på hjemmesiden skulle ikke være noget super kompliceret, som ville skubbe brugerne væk fra at bruge siden.

Selve processen med wireframing begyndte med skitsering af forskellige idéer til hvordan hjemmesiden ville kunne komme til at se ud, så man havde en idé om hvordan hjemmesiden skulle komme til at se ud. Så man også havde mulighed for at tale om designet af hjemmesiden før man rent faktisk valgte et design alle var enige i.

Nogle af de design, som blev skabt, var følgende:

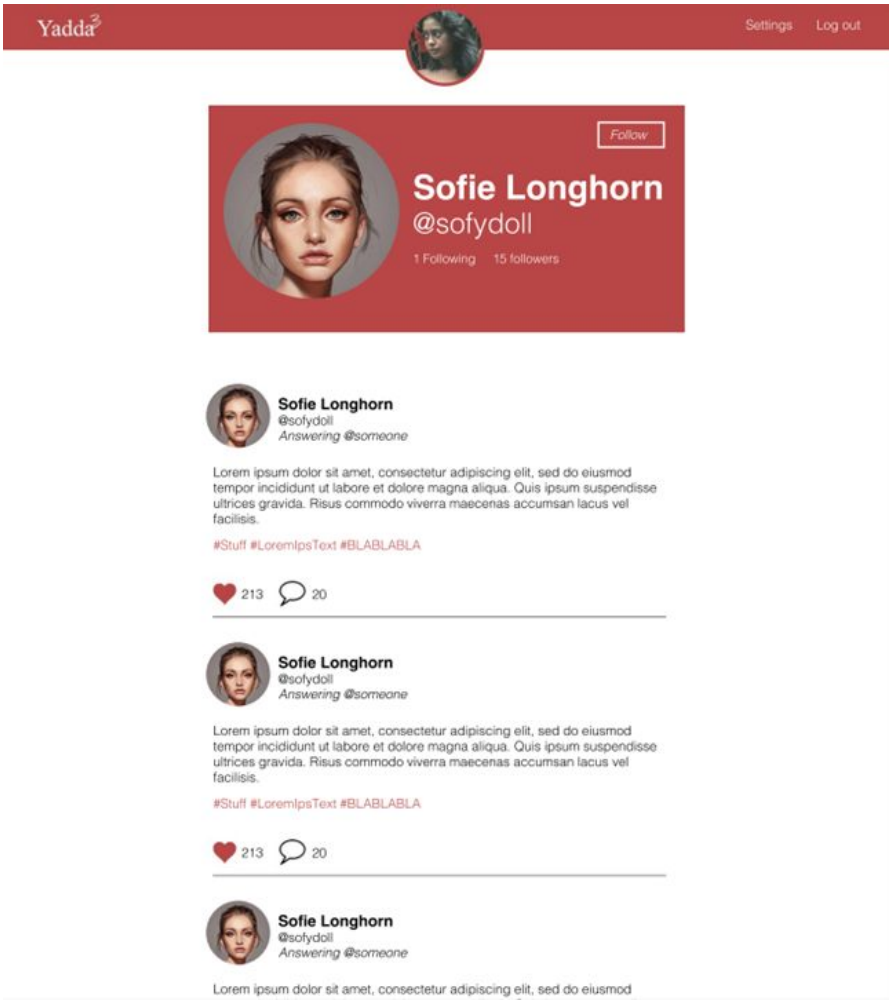


Efter arbejdet med at skitsere hjemmesiden, så blev der skabt et overblik over de forskellige ting, som var vigtige og hvad det var vi kunne lide ved de forskellige design valg, som var blevet taget under arbejdet med hjemmesiden og de forskellige skitser.

Det næste skridt handlede om at lave en mere visuel udgave af designet, hvor vi satte os ned i noget tid med Photoshop eller XD, så man kunne designe hvad man tænkte kunne være muligheder til selve siden.

Farverne var endnu ikke valgt på dette tidspunkt, så der blev også brainstormet nogle idéer til dette på dette tidspunkt, men intet var sat i sten og kunne altid ændres i løbet af projektet.

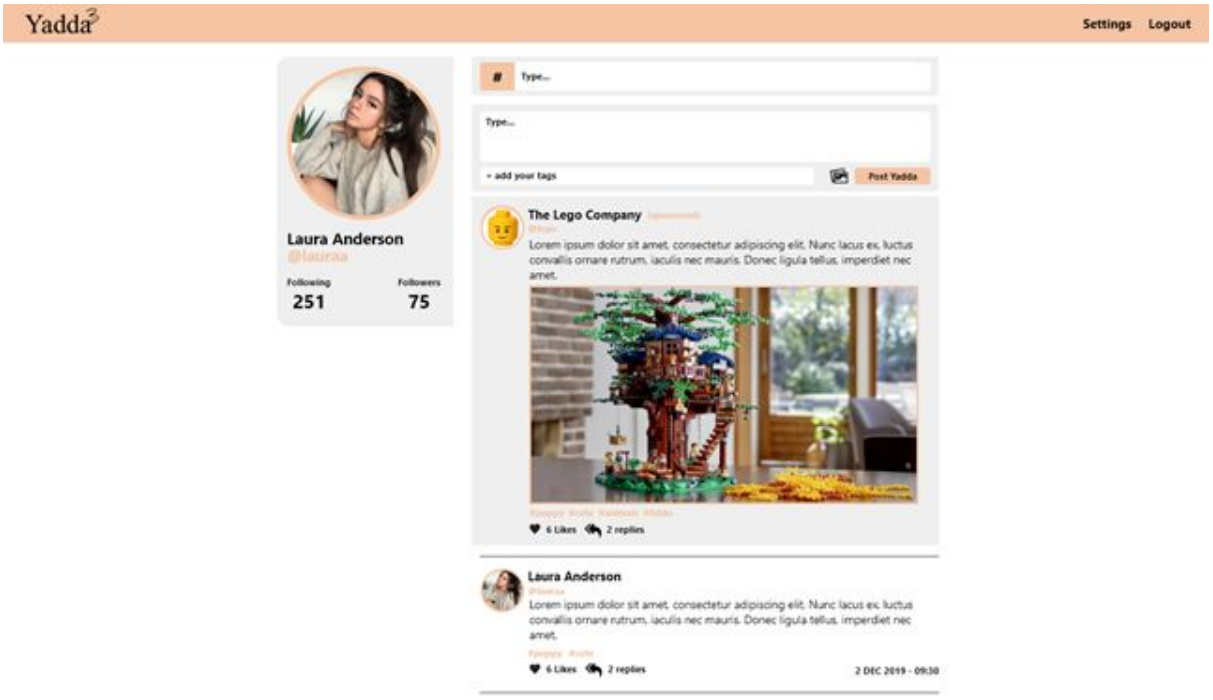
Vi udarbejdede en liste over de forskellige sider, som skulle bruges på siden for at kunne finde ud af hvor mange sider vi skulle designe efter og det var også vigtig at gøre for processen med selve programmeringen. Efter at denne liste var lavet, så var disse nogle af siderne, som blev lavet som wireframes:



Dette var en af de første bud, som blev lavet til hjemmesiden, hvor nogle af idéerne fulgte med ind i det endelige design, hvor resten var med til at inspirere det næste design. Farverne virkede godt sammen og ville passe godt med et mørkt tema, mens idéen med

profilbillede i midten af navigationsbaren var noget vi valgte at komme tilbage med senere hen i processen.

Det næste wireframe er tættere på hvad det var siden kom til at ende op som, men manglede nogle af de forskellige elementer, som var fremme i det forrige wireframe:



Der var mange forskellige idéer i løbet af selve projektet, som passede godt sammen og nemt kunne blive sat sammen for at få det bedste udtryk. Der er selvfølgelig forskellige indflydelser fra sider, som Twitter og Facebook i selve designet, men det var vigtigt at kunne sætte et præg af os selv på hjemmesiden.

I forhold til announcer, så havde vi tanken om at reklamer skulle være sponsorerede posts, så de ville passe ind på siden uden at irritere brugeren alt for meget, men stadig være synlig på siden i stedet for.

Farver

(Simon)

Som det nok kan ses i de forskellige wireframes, så var der ikke rigtigt et fast farve tema bortset fra at det lyse tema ville være det primære tema, hvilket var det som siden var designet efter og senere ville blive til et mørkt ved siden af. Det var nemmere at fokusere på det hvide tema og senere tilføje mørke farver til hjemmesiden bagefter.

En vigtig ting med siden var at de ikke skulle virke for tætte på farverne, som Facebook og Twitter havde, så det var vi havde vores egen identitet. Blå var en farve, som vi holdte os fra i løbet af hele designprocessen.

Farven, som startede ud på hjemmesiden, som en placeholder, var den orange farve fra det andet wireframe. Denne farve blev holdt for bare at have noget på siden i den tid, som siden var under konstruktion og ville blive kigget på senere i projektets forløb.

Da det mørke tema skulle laves, så var der et problem med den lyse orange farve. Selvom farven passede godt sammen med en hvid baggrund, så var farven en smule for lys til en mørkere baggrund, hvilket ikke passer godt med hvad et mørkt tema står for.

Det var på det her tidspunkt at man kiggede tilbage på den røde farve, som blev brugt i det første wireframe. Dens nuance var hverken for lys eller for mørk til selve siden, da det var en farve, som lå nogenlunde i midten af farvespektret, når man kigger på mellem sort og hvid i den røde hue.

Den lyse orange farve havde også lidt mere af et feminint præg til den, hvilket muligvis ville have skræmt mandlige brugere væk fra siden. Så, det endte med at blive rød.

Mongo Schema med Mongoose

(Christian)

For at tilføje data til Mongo DB lavede vi model, også kendt som et schema, som informere Mongoose om den data, som vi ønsker at gemme i Mongo. Mongo er en dokumentdatabase som gemmer dataen i JSON.

Denne model beskriver hvilke elementer som vi ønsker at gemme. Dette inkluderer de navne, som vi giver til den information, der skal gemmes (f.eks. Navn, E-mail, Password…). Her udover hvilken slags type informationen skal gemmes som (f.eks. String, number, date…).

Det første objekt i skamaet har navnet ”confirmed”, som bruges til at sikre at brugen først komme ind på selve YaddaYaddaYadda, efter at han/hun har bekræftet linket som modtages i e-mailen. ”Confirmed” har datatypen ”Boolean” med ”false” som ”default”, indtil linket er bekræftet.

Herefter har vi objekterne ”firstname” og ”lastname”. Disse objekter har datatypen ”string”, samt ”true” som ”required”. Så man først kan oprette sig som bruger, når man har skrevet sit navn i hver af disse felter. Disse ”strings” vil blive gemt som Lowercase i databasen, for at have en gennemgående databasestruktur.

”Username” og ”Email” er gemt på samme måde som ”firstname” og ”lastname”, dog med den tilføjelse at brugerens ”Username” og ”Email” skal være unikt.

Det er ikke ”required” at tilføje et profilbillede. Men hvis man gør, så vil det blive gemt som en ”String”. Det er til gengæld ”required” at brugerens ”password” er ”true”, som også bliver gemt som en ”String”.

”Following” er den liste af andre brugere som brugeren følger. Denne information bliver gemt som ”String”. Det er selvfølgeligt ”required” at dette er ”true”, før at en bruger bliver tilføjet til ens ”Following”.

”date” er naturligtvist gemt i datatypen ”date”, og har som ”default” ”Date.now”, som sørger for at datoen stemmer over ens med dagens dato.

Til sidst er der ”darktheme”, som har datatypen ”Boolean” med ”false” som ”default”. Dette bruges til at give brugeren muligheden for at skift imellem et light og et dark theme.

Konstruktion

Opsætningen af siden med Express

(Simon)

Til selve opsætningen af hjemmesiden og Node.js, så blev der gjort brug af en package til Node.js kaldet Express, som hjælper en med at autogenere elementer til ens arbejde med Node. Med Express kan man vælge hvilken viewport man ønsker i at bruge til at vise sin HTML for dynamisk indhold, hvor disse kan være pug, hbs eller for eksempel dust. Der er mange forskellige af disse view templates, men det vi valgte til dette projekt, var pug, selvom det er en af de viewports der er længst fra HTML, men med en converter af HTML til pug, så var det ikke det største problem. Pug virkede til at være nemmere for nogle if-sætninger i selve filerne, så det blev valget for dette.

```
default
  .Yadda
    .profilePicYadda
      a(href=`/profile/${yadda.user._id}`)
        if yadda.user.image
          img( alt='' style=`background-image:url(https://res.cloudinary.com/kuko/w_80/${yadda.user.image})`)
        else
          img(style="background-image:url('/../images/profile.jpg')" alt='')

    .yaddaBox
      .yaddaUser( class=`${yadda.parent? "parent" : ""}`)
        a(href=`/profile/${yadda.user._id}`)
          h2
            = yadda.user.fullname
        a(href=`/profile/${yadda.user._id}`)
          h3
            = "@" + yadda.user.username
        if yadda.parent
          .answer
            p Answer to
            a(href=`/profile/${yadda.parent.user._id}`)
              = `@${yadda.parent.user.username}`
            p   >
            a(href=`/thread/${yadda.parent._id}`) yadda
```

Der var selvfølgelig også andre packages udover Express, som hjalp med selve oprettelsen, som gjorde nogle ting lidt nemmere. Denne ene package er Passport, som var med til at give nogle bedre muligheder for at håndtere login og registrering på selve hjemmesiden. Hvilket kan ses i brugen af Passport her til at bekræfte et login:

```
exports.postLogin = function (req, res, next) {
  passport.authenticate("local", {
    successRedirect: "/",
    failureRedirect: "/user/login",
    failureFlash: true,
  })(req, res, next);
};
```

Efter at alt er blevet sat op til at kunne bruges, så var det bare med at prøve på at få alle funktioner ind på siden, men Express gjorde det nemt at kunne starte med programmering med det samme og ikke skulle stå for at sætte ting op, som man sikkert har gjort så mange gange før.

Routing

(Simon)

Når man er på en hjemmeside, så er det vigtigt at kunne komme til at gå på forskellige sider, når man enten indlevere en form eller trykker på en knap, så derfor er routing en vigtig del af arbejdet med en hjemmeside.

Med Node.js er det muligt at kunne route en side og gør at den holder denne information, når man arbejder med det og det kan sætte funktioner i gang, når det er man henter en bestemt route, hvilket enten kan være via post og get methods.

```
router.get("/", index.frontpage);
router.get("/all", ensureAuthenticated, index.frontpageAll);
router.get("/search/:type/:search", ensureAuthenticated, index.search);
router.get("/settings", ensureAuthenticated, index.settings);
router.get("/followers/:id", ensureAuthenticated, index.followers);
router.get("/following/:id", ensureAuthenticated, index.following);
router.get("/profile/:id", ensureAuthenticated, index.profile);
router.get("/thread/:id", ensureAuthenticated, index.thread);

router.post("/profilepic", ensureAuthenticated, upload, api.profilePic);
router.post("/name", ensureAuthenticated, api.name);
router.post("/password", ensureAuthenticated, api.password);
router.post("/follow/:id", ensureAuthenticated, api.follow);
router.post("/theme", ensureAuthenticated, api.theme);
router.post("/yadda", ensureAuthenticated, upload, api.createYadda);
router.post("/yadda/like", api.likeYadda);
```

Her kan man se de forskellige routes, som bliver lavet, når man er inde på siden, hvor det handler om at kalde funktioner, som derefter tjekker databasen og begynder at bygge siden fra det indhold, som bliver hentet.

For eksempel, så henter routen ”/” alt det indhold, som burde komme frem, hvis det er man er logget ind.

```
exports.frontpage = async (req, res) => {
  if (req.user) {
    let tagsOfTheWeek = await helper.tagsOfTheWeek();
    let yaddas;
    if (req.user.following.length) {
      yaddaArr = [...req.user.following, req.user._id];
      yaddas = await Yadda.find({ user: { $in: yaddaArr } })
        .sort([["createdAt", -1]])
        .populate({
          path: "user parent",
          populate: { path: "followers user" },
        });
    } else {
      yaddas = await Yadda.find({})
        .sort([["createdAt", -1]])
        .populate({
          path: "user parent",
          populate: { path: "followers user" },
        });
    }
```

Når dette så er gjort, så er det vigtigt at kunne komme til at fremvise den pug fil, som skal fyldes ud med information, som vil blive gemt i requesten af siden. Dog kan man også se at der bliver testest om req.user er authenticated og derefter vil den udføre selve funktionen.

```
    res.render("index", {
      title: "Frontpage",
      yaddas,
      user: req.user,
      users: [],
      tagsOfTheWeek,
    });
  } else {
    res.render("frontpage", {
      title: "frontpage",
    });
  }
};
```

Når det er funktionen er færdig med at gøre hvad den skal gøre, så vil den vise index.pug, men hvis selve brugeren ikke er authenticated, så vil brugeren blive skubbet tilbage til forsiden af hjemmesiden og ikke ens feed, fordi man ikke er logget ind på siden.

Email Confirmation

(Jonatan)

For at verificere brugeren, har vi implementeret “email confirmation”, som først aktiverer en registreret bruger når de har klikket på linket de modtager på deres mailadresse.

For at generere en URL med info om hvilken bruger der skal verificeres, har vi gjort brug af modulet **jsonwebtoken**, som generer en JWT med brugerens id.

```
try {
  let emailToken = await jwt.sign(
    {
      userId: user._id,
    },
    process.env.EMAIL_SECRET,
    {
      expiresIn: "10d",
    }
  );

  const url = `http://localhost:3000/user/confirmation/${emailToken}`;
```

Derefter benytter vi os af modulet **nodemailer**, som gør det muligt at sende en email til brugeren fra serveren.
Da opsætningen af nodemailer (se nedenfor) kræver oplysninger til personlig mail login, har vi brugt modulet **dotenv** til at gemme disse properties i en .env fil, som derefter kan hentes i process.env objektet.
Denne .env fil undlader vi så at dele gennem github af sikkerhedsårsager.

```
  let transporter = nodemailer.createTransport({
    service: process.env.EMAIL_SERVICE,
    auth: {
      user: process.env.EMAIL_USERNAME,
      pass: process.env.EMAIL_PASSWORD,
    },
  });
```

Til sidst sender vi mailen til brugeren med den tilhørende JWT, som en url parameter.

```
  await transporter.sendMail({
    from: '"YaddaYaddaYadda" < ' + process.env.EMAIL_USERNAME + ">", // sender address
    to: user.email,
    subject: "Confirm Email",
    html: `Please click this link to confirm your email: <a href="${url}">${url}</a>`,
  });
```

Når brugeren sender en get request til den generede url, henter vi dens token og bruger **jsonwebtoken** igen, til at hente den tilhørende userId.
Vi går så ind og ændre brugerens status på confirmed i databasen til true, hvorefter brugeren kan logge ind.

```
const { userId } = await jwt.verify(
  req.params.token,
  process.env.EMAIL_SECRET
);

const user = await User.findById(userId);
```

Håndtering af billeder

(Jonatan)

For at gøre det muligt at uploade billeder til yaddas og profilbillede, startede vi med at bruge modulet **multer**, som er en middleware der gør det nemt at uploade billeder lokalt.
Hertil måtte vi så sætte restrictions op mht. fil størrelse, da det ellers ville sænke hastigheden på vores site. Dette var uhensigtsmæssigt, da vi gerne ville have mere kontrol og uploadede billeder, samt bedre brugervenlighed mht. Upload processen, hvorfor vi valgte at gøre brug af Cloudinary (Cloud baseret billede- og video management).
Cloudinary giver muligheden for at redigere billeder og video (Crop, resize, facereckognision m.m), samt automatisk generering af tags.
Vi har som udgangspunkt brugt muligheden for resize, så vi henter passende størrelse billede, for pagespeed optimering.
Fremtidigt er der dog mulighed for at implementere andre features efter behov, med stor mulighed for forbedret UX.
Eksempelvis lazyload af billeder er nemt at sætte op, da man kan hente en blurred udgave af billedet på 1kb, som så bliver skiftet ud med originalen på load.

Cloudinary Serverside

Vi bruger stadig multer til at hente filer pr. request, men i stedet for at gemme dem på serveren bruger vi memoryStorage, hvorfra vi kan hente filerne og sende dem videre til cloudinary.

```
const storage = multer.memoryStorage();
module.exports = multer({ storage, fileFilter }).single("image");
```

Først sætter vi konfigurationer op, med data omkring vores cloudinary bruger; igen med enviromental variables for sikkerhed.

```
// CLOUDINARY CONFIG
cloudinary.config({
  cloud_name: "kuko",
  api_key: process.env.CLOUDINARY_KEY,
  api_secret: process.env.CLOUDINARY_SECRET,
});
```

Herefter laver vi en upload funktion, som uploader billeder til cloudinary og returnerer et publicId, som senere hen kan bruges til at hente billedet frontend.

Multer giver os fil-data på req objected, hvortil vi kan bruge modulet **Datauri** til at returnere en BLOB type string.

Vi bruger derefter **cloudinary** modulet til at uploade filen.

```
module.exports.upload = async (req) => {
  const dUri = new Datauri();

  let file = dUri.format(
    path.extname(req.file.originalname).toString(),
    req.file.buffer
  );

  try {
    return await cloudinary.uploader.upload(file.content, {
      folder: `yadda/`,
    });
  } catch {
    return;
  }
};
```

Vi kan så bruge denne funktion når vi laver et request til eksempelvis at uploade profilbillede.

```
exports.profilePic = async (req, res) => {
  try {
    let img = await cld.upload(req);

    let image = img.public_id;

    await User.findByIdAndUpdate(req.user._id, { image: image });

    req.flash("success_msg", "Profile picture was changed");
    res.redirect("/settings");
```

Clientside

Når vi nu skal hente billedet, sætter vi en cloudinary url ind med navnet på vores cloudinary bruger, og til sidst billedets PublicID.

```
background-image:url(https://res.cloudinary.com/kuko/w_80/${profile.image}
```

Cloudinary giver os muligheden for at redigere billedet direkte i URL'en, her er "w_80" eksempelvis en måde at hente en udgave af billedet der kun er 80p i bredden.
For SEO er det også muligt at ændre på billedets navn på request, hvilket til fordel kunne være en af de automatisk genererede tags, så søgemaskiner ved hvad billedet indeholder.

Password kryptering

(Jonatan)

Efter længere research på forskellige hjemmesider, kunne vi komme frem til at der var 4 bredt accepterede hashing algoritmer, dog med forskellige holdninger til hvad der var bedst. Da password hashing kræver en "langsom" algoritme, kunne vi skære mange andre fra og endte ud med at kigge på (bcrypt, pbkdf2 og argon2)

Pbkdf2 er den ældste men også den der er størst enighed om er den mindst sikrer.
Denne har dog den fordel, at den er en del af nodejs, hvilket giver mindre risiko for fejl, samt nem implementering.

Bcrypt er en af de mest brugte hashing algoritmer og er nem at bruge, da den, med modulet bcryptjs, selv genererer en salt, som den sætter på starten af af det hashede password. Dette gør at vi ikke behøver at gemme salt's separat. Den kommer dertil med en verify funktion, der gør det nemt at tjekke hash mod password.

Argon2 vandt i 2015 "password hashing competition" (Preziuso, 2019) og bliver mange steder anbefalet som den bedste hashing algoritme at bruge i fremtidige projekter.
Den har også den fordel, at dens node modul er sat op, meget ligesom bcryptjs og det er derfor nemt at skifte imellem de to.
Denne giver mulighed for memory harness, hvilket hjælper på sikkerhed, men kan lægge meget pres på serveren ved hacking forsøg (ukendt, 2019).
Dette kan derfor skabe risiko med hensyn til tilgængelighed af sitet og dertil brugeroplevelsen.

OWASP.ORG anbefaler argon2 som den foretrukne algoritme mht. sikkerhed, med det forbehold, at man skal være i position til at kunne indstille den korrekt mht. de forskellige parameter; og ellers anbefaler de bcrypt. (ukendt, n.d.)

Af disse årsager har vi valgt at benytte os af bcrypt, dog med planer om sætte os mere ind i argon2 til fremtidige projekter.

Hosting

(Jonatan)

Mht. hosting har vi valgt at bruge heroku da det er gratis og har CLI implementering, som gør det nemt at håndtere versionsstyring i sammenhæng med git.
Vi vil ikke komme nærmere ind på deploy af app, da vi bare har fulgt Herokus' dokumentation (ukendt, 2020)

Database

(Jonatan)

Til hosting af database har vi valgt at bruge atlas, som bruger mongoDB.
Opsætningen er meget enkel, da vi bare skal skifte vores nuværende database url ud med den vi får genereret af atlas.

```
let MONGODB_URI = `mongodb+srv://yadda:${process.env.ATLAS_PASSWORD}@yadda-lxawk.mongodb.net/test?retryWrites=true&w=majority`;
```

Evaluering

(Christian)

Projektet skred hurtigt fremad, stort set lige med det samme, efter at gruppen havde opgavebeskrivelsen. Den første weekend blev brugt på at gennemtænke opgaven, samt undersøge mulige måder at løse de forskellige udfordringer, som opgaven krævede.

Herefter benyttede vi os af Facebook Messenger og Skype til at få lavet en tidsplan, trello, en problemformulering og få oprettet selve databasen. Facebook Messenger og Skype blev benyttet hyppigt igennem hele processen. Da tidsplanen og problemformuleringen var blevet godkendt, begyndte research af andre Sociale medie sider og skitseringen af Wireframes. Herefter blev opgaverne mere uddelegeret blandt gruppens medlemmer.

Gruppen mødtes over Skype hver dag (mandag til og med fredag), her snakkede vi op proces, hvad der var blevet udarbejdet siden sidste møde og hvad der skulle laves inden næste møde. Disse Skype møder to fra en halv time til lidt over en hel time.

Her var det meningen at de forskellige arbejdsopgaver skulle noteres på Trello's Scrumboards, sammen med de problemer og fejl, som opstod i løbet af processen. Dette blev dog ikke rigtigt gjort. Årsagen til dette var, at det ikke blev særligt aktuelt. Gruppens Skypemøder var relativt lange, og gennemført. Opgaver blev uddelegeret efter tidsplanen. Vi fulgte dog ikke helt tidsplanen, efter som vi prøvede at være foran så vidt som muligt. Når problemer og fejl opstod blev de sendt over Facebook Messenger, og blev oftest løst med det samme. Større ting blev diskuteret den følgende dag over Skype.

Der var meget få problemer, ud over de forventede. Skype drillede et par gange, og vi måtte nogle gange udskyde vores møder til om aftenen, pga lægebesøg og planer med familien. Men ikke noget som spillede den store rolle. Det problem var at Simon var nødt til at udskifte hans en computer, pga at hans batteri ikke ville lade. Men dette blev heldigvis løst over en weekend, så det forsinkede os ikke for meget.

Da vi endte med at være lidt foran, gjorde at vi havde tid til at få implementeret næsten alle vores Nice to Have. Vi fik dog ikke tid til at lave en større spørgeundersøgelse / test, eftersom da vi ikke har haft muligheden for at mødes under projektet og teste andre.

Konklusion

(Jonatan & Christian)

Der er blevet udarbejdet en social medie side, hvor brugeren kan registrere sig som brugere, logge ind og benytte sin profil til at udforske platformen. Bekræftelsen af registreringen sker ved hjælp af et link, som modtages over mail.

Når man er logget ind, er muligt at oprette sine egne Yaddas med tilhørende Tags, samt søge på andre brugere og forskellige tags. Herudover er der blevet arbejdet på at gøre siden mere brugervenlig. Dette er bl.a. blevet gjort ved at undersøge andre sociale medieplatforme, samt give brugen konfigurerings-muligheder. F.eks. kan der skiftes profilbillede og ændres imellem et lyst og et mørkt tema m.m.

Node.js og mongodb har vist sig, som en nem og hurtig måde at opbygge en SOME app, med mulighed for opgraderinger og skalering. Valget af serverside rendering med ‘pug’ giver en hurtig konstruktionsfase, men har nogle dårlige sider når det kommer til brugervenligheden.

Node.js som fundament har mange moduler, samt en indbygget metode til authentication, hvortil verification af mail også hurtigt og nemt kan opsættes med brug af jwt’s.
Vi har valgt at bruge bcrypt til hashing, hvilket er acceptabelt mht. sikkerhed, men vi vil i fremtiden kigge på Argon2 som et mere sikkert alternativt.

Litteraturliste

Preziuso, M., 2019. Password Hashing: Scrypt, Bcrypt and ARGON2 [WWW Document]. Medium. URL https://medium.com/analytics-vidhya/password-hashing-pbkdf2-scrypt-bcrypt-and-argon2-e25aaf41598e

ukendt, 2020. Deploying with Git [WWW Document]. Heroku. URL https://devcenter.heroku.com/articles/git

ukendt, 2019. Is bcrypt safe to use? [WWW Document]. Reddit. URL https://www.reddit.com/r/crypto/comments/caxu4o/is_bcrypt_safe_to_use/

ukendt, n.d. Password Storage Cheat Sheet [WWW Document]. Owasp. URL https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html#modern-algorithms