Forside

IS:304 2018

Tittel: Ecoinomy

Emnekode	IS-304
Emnenavn	Bacheloroppgave i informasjonssystemer
Emneansvarlig:	Hallgeir Nilsen
Veileder	Janis Gailis
Oppdragsgiver	Cav

Studenter

Etternavn	Fornavn
Haaland	Erik
Skarpmoen	Ola
Nomme	Sebastian
Lindberg	Oscar

Jeg/vi bekrefter at vi ikke siterer eller på annen måte bruker andres arbeider uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	JA X	NEI___
Kan besvarelsen brukes til undervisningsformål?	JA X	NEI___
Vi bekrefter at alle i gruppa har bidratt til besvarelsen	JA X	NEI___

Forord

Denne rapporten er skrevet av prosjektgruppen Cav som en del av bacheloroppgaven i IT og informasjonssystemer (IS-304) ved universitetet i agder, våren 2018. Prosjektet rapporten baserer seg på er utviklingen av det sosiale mediet “Ecoinomy”. Ideen til Ecoinomy er utviklet av gruppemedlemmene i Cav, og vi er derfor også eier av prosjektet.

Vi vil gjerne takke vår veileder Janis Gailis for god veiledning gjennom dette prosjektet.

Sammendrag

Vi i gruppen Cav utvikler et sosialt medie kalt “Ecoinomy” som er ment å samle “kryptoentusiaster” over hele verden på en og samme plattform, der man kan dele sine meninger og sin kryptovaluta-portefølje med andre. Gruppen består av Oscar Lindberg, Ola Eriksen, Sebastian Nomme, og Erik Haaland. I denne rapporten har vi dokumentert utviklingsprosessen, våre erfaringer, hva vi synes har vært bra og hva vi ville gjort annerledes dersom vi skulle gjort det på nytt.

Ettersom vi i gruppen har tatt rollen som en oppstartsbedrift har vi kommet frem til at vi trenger en agil utviklings-metodologi som også gir oss mulighet til å tidlig bekrefte om tjenesten vi lager er etterspurt av målgruppen. Da kan vi tidlig finne ut hva vi burde satse på av funksjonalitet og hva vi ikke trenger å bruke tid på. Vi har, med dette i bakhodet, kommet frem til at en smidig utviklings-metodologi vil kunne passe denne gruppen godt. Vi kombinerer Lean startup med elementer fra Scrum, da gruppen er godt kjent med Scrum metodologien og lean passer godt til nystartede prosjekter.

Selve utviklingsprosessen har vært svært lærerik og utfordrende. Vi har måtte gjøre en del endringer i prosjektet underveis, men det hele har vært en god læringsprosess og vi mener vi har kommet frem til en god måte å gjennomføre prosjektet.

Alt i alt synes vi sluttresultatet ble veldig bra, og at denne rapporten godt reflekterer arbeidet vi har lagt inn i utviklingen av vår tjeneste.

Innholdsfortegnelse

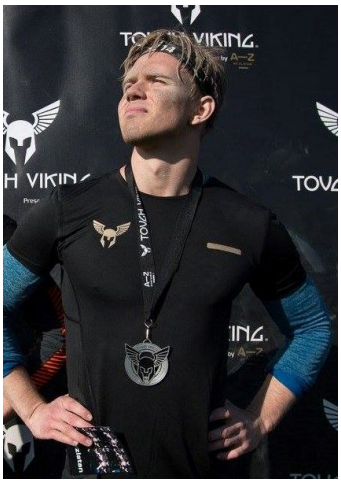
Tabelliste

Figurliste

1.0 Introduksjon

1.1 Gruppens medlemmer:

1.1.1 Sebastian Nomme



1.1.2 Erik Haaland

1.1.3 Oscar Lindberg

1.1.4 Ola Eriksen

1.2 Klienten og hva vi skal lage

Klienten i dette prosjektet er en fiktiv startup bedrift som heter Coin Adept Ventures (CAV). Den ble laget og startet i 2017. Starten på Ecoinomy kom etter at en av idè-innehaverne kontinuerlig spurte de andre om informasjon om kryptovaluta og hvilken valuta de investerte i. For å finne informasjon om kryptovaluta og blockchain må man lete igjennom diverse facebook grupper, youtubekanaler, twitter posts og instagramkontoer mm. Dette kan være en smertefull prosess, og det kan ta lang tid å finne informasjonen man er ute etter. Dette er problemet vi ønsker å adressere med Ecoinomy. Det vi ønsker å gjøre med Ecoinomy er å samle informasjon om kryptovaluta på en plattform hvor man også skal kunne sette opp en personlig portfolio over valutaen man holder, slik at man enkelt kan dele med andre.

Klienten ønsker en fungerende versjon av Ecoinomy, med fokus på de sosiale elementene. Dette innebærer å lage en nettside der man kan opprette brukere, lage innlegg, og dele innlegg. Hver bruker skal også ha en egen profilside der man kan redigere hva som skal vises offentlig.

1.3 Hvorfor vi valgte dette prosjektet

Valget av prosjekt er basert på at medlemmene i gruppen har en genuin interesse for både kryptovaluta og teknologien bak. Kryptovaluta-markedet er fortsatt veldig ungt, men det har i den siste tiden opplevd en eksplosiv økning i antall interesserte og investorer. Dette har gjort at det også har blitt en stor mengde mennesker som søker informasjon om forskjellige kryptovalutaer og hva som er lurt å investere i. Som nevnt tidligere kan det være en utfordring å finne god informasjon, spesielt når den er så spredt som den er. Det har altså oppstått et behov for informasjon som ut i fra våre erfaringer ikke blir tilfredsstilt. Derfor har vi valgt å lage plattformen Ecoinomy slik at vi kan skape en samlingsplass for informasjon om kryptovaluta og blockchain teknologi, og tilfredsstille dette behovet.

2.0 Analyse

2.1 Brukerhistorier:

Brukerhistoriene er med på å illustrere behovene vi har for forskjellig funksjonalitet i systemet. Disse historiene inneholder hvem, hva, og hvorfor for hvert behov/funksjon.

Brukerhistoriene ble utarbeidet i pre-sprinten, men vi har hele tiden jobbet med å forbedre og legge til nye brukerhistorier som vi føler vi mangler, eller endret eksisterende dersom vi har sett nye behov eller forbedringer. Dette gjør listen med brukerhistorier til en dynamisk liste. Formatet vi har brukt for brukerhistoriene er: Som **X** ønsker jeg at **Y** slik at **Z**.

Vi la alle brukerhistoriene inn i et dokument på google disk. Ettersom vi har brukt en agil utviklingsmetode vil brukerhistoriene alltid kunne endres underveis, men den foreløpig siste versjonen av brukerhistoriene ser slik ut:

Brukerhistorier:

- **Som en bruker vil jeg ha muligheten til å vise mine investeringer av kryptovaluta grafisk, for å ha bedre oversikt over mine midler.**
- **Som en bruker vil jeg ha mulighet til å dele min portefølje, for å få tilbakemelding fra andre brukere.**
- **Som en bruker vil jeg kunne bli medlem av en gruppe, for å knytte nettverk og samarbeide med andre brukere.**
- **Som en bruker ønsker jeg å kunne se andre sine investeringer slik at jeg kan få inspirasjon og råd til fremtidige kjøp.**
- **Som en bruker vil jeg kunne skrive og dele innlegg, slik at jeg kan dele informasjon med andre.**
- **Som en bruker vil jeg kunne 'upvote' andre sine innlegg, for å gi noe tilbake for nyttig informasjon og kvalitetssikre innlegg.**
- **Som en bruker vil jeg kunne rapportere upassende og støtende innlegg, for å sikre egen trivsel på nettsiden.**
- **Som en bruker vil jeg kunne rapportere andre brukere som ikke agerer i tråd med retningslinjene.**
- **Som en bruker vil jeg at innhold jeg har bidratt med kunnskap eller økonomisk fortsatt lagres og er tilgjengelig selv om original poster har slettet innlegget. for?**
- **Som en bruker vil jeg at innlegg skal ha en eier, slik at eieren må stå for det den har skrevet.**
- **Som bruker vil jeg ha en chattefunksjon for at jeg skal kunne konversere med andre brukere i et lukket miljø.**

Administrator:

- **Som administrator ønsker jeg at hver bruker skal ha en unik ID slik at det er lett å skille de fra hverandre og å behandle riktig bruker.**
- **Som administrator vil jeg kunne slette innlegg, for å sikre kvalitet på siden.**
- **Som administrator vil jeg kunne utestenge brukere ved brudd på regler, for å opprettholde saklighet.**
- **Som en administrator vil jeg kunne slette grupper, for å sikre kvalitet på siden.**
- **Som en administrator vil jeg at navn som kan være støtende, skal ikke være lov å bruke. slik at nettsiden kan sile ut støtende navn automatisk.**

2.2 MoSCoW

Basert på tidligere erfaringer har vi valgt å bruke Moscow som metode for å prioritere brukerhistoriene. Dette innebærer at vi rangerer brukerhistoriene etter Must have, Should have, Could have, og Won't have. Måten vi har tolket denne rangeringen på er at brukerhistorier som er "Must have" må bli implementert i systemet for at det skal fungere og gi noe verdi for brukere. De som er rangert som "Should have" er historier som bør implementeres, men som ikke er absolutt nødvendig. "could have" er historier som kan bli implementert dersom vi har tid, men som ikke trengs. "Won't

have" er historier som vi ikke vil prioritere å ha med i dette prosjektet enten fordi det ikke gir nok verdi til brukeren, eller fordi det er for omfattende til å kunne bli gjort i løpet av bacheloren.

Vår Moscow rangering av brukerhistoriene ser slik ut:

Nummer	Brukerhistorie	MOSCOW rangering
1	Som en bruker vil jeg kunne skrive og dele innlegg, slik at jeg kan dele informasjon med andre.	Must have
2	Som en bruker ønsker jeg en egen profilside, for å ha all informasjon om meg på et sted.	Must have
3	Som en bruker vil jeg kunne 'upvote' andre sine innlegg, for å gi noe tilbake for nyttig informasjon og kvalitetssikre innlegg.	Must have
4	Som en bruker vil jeg ha muligheten til å grafisk fremstille mine investeringer av kryptovaluta, for å ha bedre oversikt over mine midler.	Must have
5	Som en bruker vil jeg ha mulighet til å dele min portefølje, for å få tilbakemelding fra andre brukere.	Should have
6	Som en bruker ønsker jeg å kunne se andre sine investeringer slik at jeg kan få inspirasjon og råd til fremtidige kjøp.	Should have
7	Som en bruker vil jeg kunne lage en gruppe, for å knytte nettverk og samarbeide med andre brukere.	Could have
8	Som en bruker vil jeg kunne bli medlem av grupper, for å knytte nettverk og samarbeide med andre brukere.	Could have
9	Som en bruker vil jeg kunne rapportere upassende og støtende innlegg, for å sikre egen trivsel på nettsiden.	Could have
10	Som en bruker vil jeg kunne rapportere andre brukere, fordi de kan dele støtende innhold som ikke er i tråd med retningslinjene.	Could have
11	Som en bruker vil jeg at innhold som har blitt besvart ikke slettes fordi jeg har bidratt økonomisk eller med støtte med mindre det bryter med retningslinjene.	Could have
12	Som bruker vil jeg ha en chattefunksjon for at jeg skal kunne konversere med andre brukere i et lukket miljø.	Won't have
	administrator brukerhistorier	
13	Som administrator ønsker jeg at hver bruker skal ha en unik ID slik at det er lett å skille de fra hverandre og å behandle riktig bruker.	Must have
14	Som administrator vil jeg kunne slette innlegg, for å sikre kvalitet på siden.	Must have

15	Som administrator vil jeg kunne utestenge brukere ved brudd på regler, for å opprettholde saklighet.	Must have
16	Som en administrator vil jeg kunne slette grupper, for å sikre kvalitet på siden.	Must have
17	Som en administrator vil jeg at navn som kan være støtende, skal ikke være lov å bruke. slik at nettsiden kan sile ut støtende navn automatisk.	Should have

Tabell 1, MoSCow

2.3 Akseptanse

For at vi skal kunne si oss ferdig med å implementere en spesifikk brukerhistorie, trenger vi noen kriterier som må innfris før vi kan si oss ferdige med brukerhistorien. Dette er akseptansekriterier, de beskriver hvilke krav den spesifikke brukerhistorien må oppfylle før den kan anses som ferdig implementert.

I listen under har vi tatt med alle våre "must have" brukerhistorier, og deres akseptansekriterier. Den fullstendige listen ligger i vårt MoSCoW dokument under vedlegg.

Brukerhistorier.	**Akseptansekriterier.**
Bruker	
Som en bruker vil jeg kunne skrive og dele innlegg, slik at jeg kan dele informasjon med andre.	En funksjon for å opprette innlegg. Med tittel, eier og datostempel. Andre brukere skal kunne se innlegget i sin feed.
Som en bruker ønsker jeg en egen profilside, for å ha all informasjon om meg på et sted.	Hver unike bruker skal ha sin egen profilside med profilbilde, personinfo og alle sine innlegg i form av "personlig feed".
Som en bruker vil jeg kunne 'upvote' andre sine innlegg, for å gi noe tilbake for nyttig informasjon og kvalitetssikre innlegg.	Brukere skal ha mulighet til å upvote innlegg. Man skal bare kunne upvote en gang. Upvote er bindende.
Som en bruker vil jeg ha muligheten til å grafisk fremstille mine investeringer av kryptovaluta, for å ha bedre oversikt over mine midler.	Portfolio Funksjon på profilsiden. Fremstilt grafisk. Oversikt over alle nåværende midler.
Administrator	
Som administrator ønsker jeg at hver bruker skal ha en unik ID slik at det er lett å skille de fra hverandre og å behandle riktig bruker.	Når hver bruker som registrerer seg på plattformen får en unik ID som identifiserer brukeren.
Som administrator vil jeg kunne slette innlegg, for å sikre kvalitet på siden.	Administrator skal kunne gå inn på et spesifikt innlegg og slette både innhold og hvem som har skrevet det.

Som administrator vil jeg kunne utestenge brukere ved brudd på regler, for å opprettholde saklighet.	Administrator skal kunne gå inn på en spesifikk bruker og utestenge brukeren fra plattformen.
Som en administrator vil jeg kunne slette grupper, for å sikre kvalitet på siden.	Administrator skal kunne gå inn på en spesifikk gruppe og slette denne .

Tabell 2, akseptansekriterier

2.4 Testcase

I vårt MoSCoW dokument har vi også tatt med andre elementer som knytter seg til hver brukerhistorie, dette er elementer som: Usecase, teknisk testcase, brukergrensesnitt testcase, avhengighet, scene (view) og akseptanse.

Usecase er en detaljert beskrivelse av brukerens handlinger eller steg for å kunne gjennomføre en brukerhistorie.Vi har to typer testcaser per brukerhistorie. Vi har her lagt til et eksempel på teknisk testcase og et eksempel på brukergrensesnitt testcase, resten vil ligge under vedlegg. De tekniske testcasene går ut på å teste om vår kode gjør det den skal etter det vi har beskrevet i vår brukerhistorie. Brukergrensesnitt testcase er tester vi har laget for å kunne teste brukergrensesnittet, og hvordan brukeren interagerer med vårt design. Vi ser etter feil eller mangler som kan påvirke brukbarheten av grensesnittet.

Oppsettet for testcasene ser slik ut:

Forhåndsbetingelse:
Test steg:
1.
2.
Forventet resultat:

Forhåndsbetingelsen er det som må være på plass på forhånd for at vi skal kunne utføre den spesifikke testen. Deretter kommer test-stegene, og til slutt det forventede resultatet. Dersom det er en teknisk testcase og vi ikke får det forventede resultatet, vil vi gå tilbake i koden og se etter feil og gjøre nødvendige endringer. Dersom det er en brukergrensesnitt test vil vi analysere hva som gjorde at brukeren ikke fikk det forventede resultatet, og gjøre nødvendige tiltak for å forbedre grensesnittet og gjøre det mer brukbart.

Eksempel på utfylt test:

Teknisk testcase brukerhistorie nr.4: “Som en bruker vil jeg ha muligheten til å grafisk fremstille mine investeringer av kryptovaluta, for å ha bedre oversikt over mine midler.”

Forhåndsbetingelse: Registrert bruker.
Test steg:
1. Gå inn på sin profil.
2. Redigere portfolio.
3. Skrive inn hvilken type krypto.

4. Skrive inn bokstaver i antall.
Forventet resultat: Få feilmelding. Riktig input er tall.

Brukergrensesnitt testcase brukerhistorie nr.1: "Som en bruker vil jeg kunne skrive og dele innlegg, slik at jeg kan dele informasjon med andre."

Forhåndsbetingelse: Registrert bruker
Test steg:
1. Opprett innlegg.
2. Publiser innlegg.
Forventet resultat: Bruker skal raskt kunne poste innlegg.

Vi har laget både positive og negative tester. De positive testene er tester som skal fullføres med et positivt resultat, det skal ikke komme opp noen feilmeldinger og programmet skal kjøres som det er tenkt. De negative testene er tester som skal gi et "negativt" resultat, altså en feilmelding eller lignende. Dette er for å sjekke at vi håndterer handlinger som ikke skal kunne utføres på riktig måte.

2.5 Risikoanalyse

Risikoanalysen tar for seg forskjellige risikoer og konsekvensene de kan ha. Dette er alle typer risikoer som kan påvirke prosjektet, både internt og eksternt. Figuren under viser hvordan vi har satt opp risikomatrisen vår. Konsekvensene har fem forskjellige alvorlighetsgrader som går fra ubetydelig til katastrofalt, mens sannsynlighetsgraden går fra lite sannsynlig til svært sannsynlig. Ut i fra denne matrisen henter vi en score på hvor stor risiko hendelsen utgjør.

Konsekvens→ Sannsynlighet	1. Ubetydelig	2. Mindre alvorlig/ liten fare	3. Betydelig / Kritisk	4. Alvorlig / Farlig	5. Svært alvorlig/ katastrofalt!
5.Svært sannsynlig	5	10	15	20	25
4.Meget sannsynlig	4	8	12	16	20
3.Sannsynlig	3	6	9	12	15
2.Mindre sannsynlig	2	4	6	5	10
1.Lite sannsynlig	1	2	3	4	5

Tabell 3, mal for risikoanalyse

I analysen har vi med hva som kan gå galt, dette kan være alt fra sykdom til ulykker. Etter at analysen er ferdig må vi forberede tiltak ut i fra sannsynligheten og konsekvensen av trusselen. Den totale scoren på truslene bestemmer hvor høyt det blir prioritert å lage tiltak for å hindre den spesifikke risikoen og hvor omfattende tiltaket skal være.

Risikoanalyse:

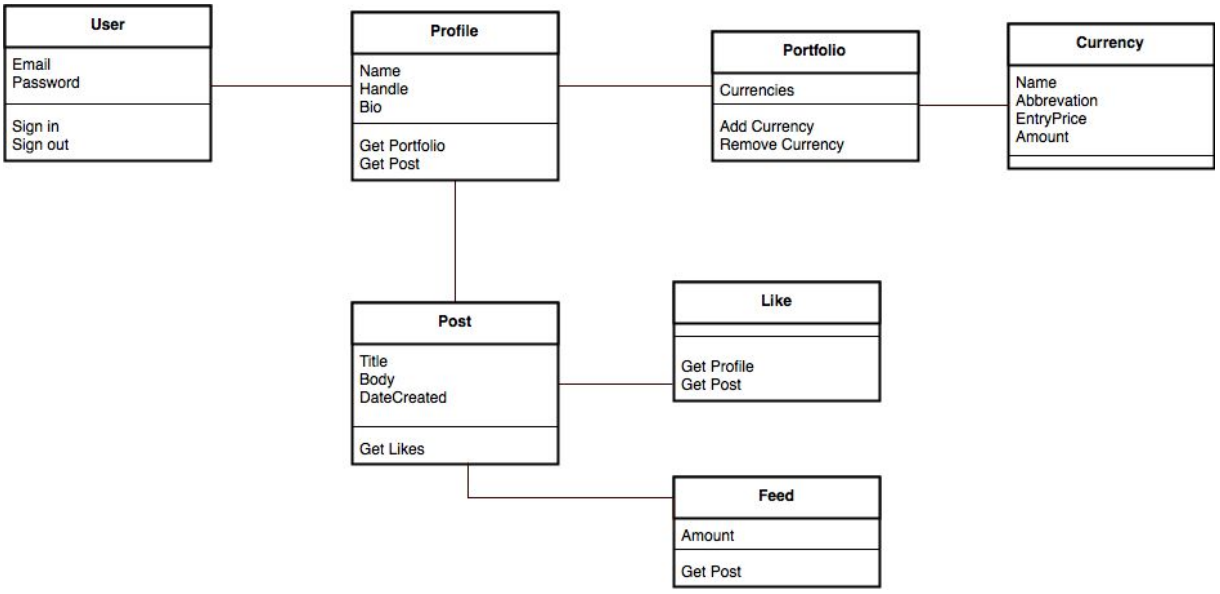
Hva kan gå galt?	Sannsynligheten	Konsekvens	Tiltak	Score
Sykdom i gruppen	Meget sannsynlig	Mindre alvorlige. Tap av arbeids timer.	Holde seg hjemme for å minimere smitte.	8
Ulykke	Mindre sannsynlig	Alvorlig/farlig.	Være oppmerksom.	5
Treningskader	Svært sannsynlig	Mindre alvorlig / Liten fare	Passe på teknikk.	10
Maskin kræsj	Sannsynlig.	Alvorlig. Tap av arbeid, som må gjøres på nytt.	Ha god backup av filer og arbeid.	12
Hardware feil.	Sannsynlig	Betydelig. Tap av arbeids timer. Kan bli mange timer, hvis man ikke får fikset det raskt.	Være obs på potensielle ulykker. Ta vare på maskinvare.	9
Feil med google docs.	Lite sannsynlig	Katastrofalt/svært alvorlig. Ingen tilgang til våre dokumenter. Miste dokumenter.	Ha backup flere steder enn bare hos google.	5
Forsentkomming	Meget sannsynlig.	Mindre/liten fare. Vil ta lengre tid før arbeidet starter.	Bli hardere på punktlighet.	10
Ikke ledig plass på Nyskapning.	Sannsynlig.	Mindre/liten fare. Vi må lete etter plass å sitte på universitetet.	Dette er ikke noe vi kan gjøre med.	6

Tabell 4, risikoanalyse

2.6 Klassediagram

Klassediagram er et diagram over systemets klasser, attributter, metoder og hvordan de er koblet sammen. Klassediagrammet vårt er laget med utgangspunkt i brukerhistoriene som ble laget i pre-sprint, og viser hvordan systemet vårt skal bygges opp.

"User" klassen vår er der brukerne logger inn og ut, de attributtene som trengs her er Epost og passord. Informasjon om brukeren blir lagret i en "profile" klasse, her lager brukeren navn og biografi om seg selv. Metodene "Get portfolio" og "Get post" brukes sånn at brukeren skal kunne se sin portefølje og poster på profilen sin. Brukerens portefølje bruker metoden "Add currency" fra "currency" klassen, og i denne klassen lages objektet som porteføljen bruker til sine beregninger. Det skal være mulig å både se og poste sine innlegg i fra profil siden og feed, dette løser vi ved at både profil og feed henter ut poster fra "Post" klassen. Post henter ut likes fra "Like" klassen.



Figur 1, klassediagram

3.0 Metodologi

3.1 Scrum og Lean startup

Etter mye research har vi valgt en metodologi som kombinerer Scrum og Lean startup. Vi valgte å bruke lean fordi det passer godt til startup-bedrifter som enda ikke har noen kunder. Målet med lean er å finne ut tidlig hva potensielle brukere ønsker fra tjenesten ved å teste produktet og få tilbakemeldinger på hva som er bra og hva som bør justeres. Alle funksjoner vi utvikler lages etter hva vi tror potensielle brukere ønsker, deretter justeres kursen etter tilbakemeldingene. Planen er derfor å begynne å teste produktet på sluttbruker etter lean startup's best practises, så snart vi har en fungerende utgave av fase 1.

Lean har ikke en fast fremgangsmåte eller struktur, derfor har vi valgt å bruke scrum sammen med lean for å strukturere arbeidet, og opprettholde tett oppfølging fra veileder og produkteier. Scrum elementene vi har brukt er: sprints, daily Scrum meetings, sprint planning, sprint review, sprint

retrospective, og Scrum master. Vår arbeidsuke går fra mandag til torsdag, og vi har i største delen av prosjektet hatt iterasjoner på tre uker. Sprintene startes med sprint planning, hvor vi lager en sprint backlog, og en overordnet oversikt over hvordan de neste tre ukene skal bli. Sprintene avsluttes siste torsdag i iterasjonen med sprint review og retrospective. På disse møtene går vi gjennom hvordan sprinten har gått, hva som var bra, hva som ikke gikk så bra, og hvordan vi kan gjøre forbedringer til neste sprint. Dette samkjører vi, så godt det lar seg gjøre, med styrings gruppemøtene. Da har vi alt ferskt i minne, og styringsgruppemøtet vil fungere i stor grad som sprint review. Hver arbeidsdag startes med 10-15 min daily scrum hvor vi planlegger resten av dagen, og får en oversikt over hva de andre på gruppen jobber med. Vi har også fredag som en tilgjengelig arbeidsdag dersom det skulle bli behov for mer gruppearbeid den uken. Vi utnevnte tidlig Ola til å være Scrum master i prosjektet. Oppgaven hans har vært få gruppen til å holde fokus og hjelpe til med å løse problemer dersom det er noe som er i veien for arbeidet.

Ved å følge denne hybriden av Scrum og Lean, mener vi at vi kan utnytte fordelene ved begge. Vi får da lean's best practises for nystartede prosjekter, samtidig som vi får strukturen og den tette kontakten med produkteier og veileder fra scrum. Scrum elementene ble tatt i bruk fra første arbeidsdag. Testfasen fra Lean har blitt tatt i bruk så godt det har latt seg gjøre tidlig i prosjektet. Vi har testet papirprototypen på sluttkunder i de første sprintene. Og vi kommer til å ta det i bruk enda mer når vi har kodet opp første versjon av systemet, da kan vi bruke sider som www.userreport.com til å få direkte feedback fra sluttbruker, og justere der etter. (Club, B. V. 2016, Februar 9), (Blank, S. 2018, Februar 9), (Conferences, G. 2013, April 3), (Entrepreneurs, G. f. 2014, Juni 17), (Lean Enterprice Institute. u.d.)

3.2 Fossefall

Fossefall er en lineær systemutviklingsmetode som har stort fokus på faste faser som er godt planlagt før de startes, ofte er disse fasene: kravspesifikasjon, design, testing, implementasjon og drift. I denne metoden lager produkteier en meget omfattende kravspesifikasjon som utviklerne følger, dette gir et godt estimat både på kostnad og leveranse.

Ved bruk av fossefall går man fremover til man er ferdig, og det er meget kostbart å forandre kravspesifikasjonen etter at prosjektet har startet. Dette er en av hovedgrunnene til at vi ikke valgte en fossefallsmetode. Det passet oss bedre med Scrum, som gir oss muligheten til og gå tilbake å forandre. kontakten med produkteier under en slik lineær metode er bare i starten når man diskuterer fasene, ved å bruke smidig metode har man jevnlig kontakt med produkteier. Ved eventuelle forandringer vil ikke kostnadene være like stor som ved fossefall, endringer i kravspesifikasjonen, eller ny teknologi som kommer ut på markedet. (Høiseth, Y. u.d.) (Muller, K. 2015, November 20)(*Wikipedia*. 2018, Mai 12)

4.0 Kvalitetssikring

Hva innebærer ordet kvalitet i dette prosjektet?
Det vi ønsker å oppnå med dette prosjektet er en plattform der investorer og interesserte kan få tilgang til god informasjon, samt at de kan ha oversikt over sin egen porteføljе. Det er viktig at informasjonen er lett tilgjengelig og enkel å finne. Dette innebærer også at plattformen er enkel å bruke, vi vil at

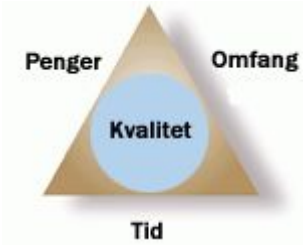
brukere med en gang de besøker plattformen skal kunne finne det de vil og utforske fritt. Dermed blir det viktigste for å sikre kvalitet i dette prosjektet; brukbarhet, tilgjengelighet, og trygg utforsking. Disse punktene har vi da hatt stort fokus på under utviklingen av tjenesten vår.

For å kvalitetssikre dette prosjektet har vi på et tidlig stadie gjennomført testing av papirprototype for å få input på design og brukbarhet fra potensielle brukere før vi har begynt å programmere. Dette har gjort at vi har sluppet unna potensielle tidkrevende endringer av feil som oppdages etter at vi har begynt å lage selve tjenesten, og at vi har kunnet tilpasse tjenesten og designet etter brukerens behov. Vi vil også gjennomføre brukertester av tjenesten etterhvert som de forskjellige funksjonene har begynt å komme på plass.

Vi har i tillegg satt opp en kommunikasjonsplan som inneholder retningslinjer for hvordan vi skal opptre under gruppearbeid og gruppemøter. Dette har hjulpet med å holde arbeidet med prosjektet profesjonelt, og gjort at kvaliteten ikke har blitt påvirket av hendelser eller omstendigheter som ikke er knyttet til prosjektet.

4.1 The iron triangle

"The Iron Triangle", eller "prosjekttrekanten" på norsk, er sammenhengen mellom tid, kostnader, og omfang. Disse tre elementene er det som til sammen utgjør kvaliteten av det som utvikles. Prosjekttrekanten illustreres ofte slik:



Figur 2, Prosjekttrekanten

Tanken bak denne trekanten er at dersom man gjør endringer på hvilken som helst av hjørnene vil dette påvirke kvaliteten (som er i midten). Hva man velger å definere som kvalitet i forskjellige prosjekter vil variere, og dette vil også påvirke hvordan man prioriterer elementene i prosjekttrekanten. En populær antakelse om denne modellen er at man kan kun velge to av de tre hjørnene, men man kan ikke prioritere alle tre. Dermed blir det en slags trade-off der man gjerne må kutte på omfang for å ikke få for dårlig tid, eller man må gå over budsjett for å få nok tid osv. For oss vil "omfang" være antall funksjoner vi tar med i systemet, "Tid" vil være tiden vi har til rådighet frem til eksamen, og ettersom dette er en bacheloroppgave der vi ikke har noen kostnader i form av utgifter for arbeid osv. vil "penger" være antall timer vi har til rådighet. (*https://www.rga.com*. 2014, Mai 1), (Microsoft. u.d.), (*Wikipedia*. 2018, Mai 8)

Ettersom vi har to mål på tid i denne trekanten, vil dette være det største hinderet for kvalitet i prosjektet. "Penger" som for oss er tiden vi har til rådighet før eksamen er ikke noe vi kan gjøre store endringer på, vi kan så klart velge å jobbe overtid for å få ferdig forskjellige elementer, men vi vil på et tidspunkt gå tom for timer vi kan bruke. Det samme gjelder tidsfristen vi har, den er umulig å endre. Derfor blir den store trade-offen for oss omfanget av prosjektet. For å sikre at kvaliteten er god må vi hele tiden gjøre vurderinger på at vi ikke gaper over for mye slik at kvaliteten blir for dårlig, men vi

må samtidig sørge for at omfanget ikke bli så lite at det ikke har noen verdi. På grunn av dette har vi gjort grundige analyser om hva som er det absolutt viktigste for å skape verdi i dette prosjektet, og vi har gjort vårt beste for å få dette til passe inn under tiden og "budsjettet" vi har til rådighet.

4.2 Programvare testing

Det finnes flere aktiviteter i en utviklingsprosess. En utviklingsprosess må inneholde en spesifikasjon som definerer programvaren som skal produseres og operasjonelle begrensninger. Programvaren må designes og programmeres etter kravene som er satt. Videre må programvaren som er produsert valideres for å se om den møter kundenes krav. Dette er et område vi ikke har arbeidet så mye med men som vi gradvis vil implementere i prosjektarbeidet. Som nevnt tidligere har vi allerede gjennomført brukertesting, men det er flere typer testing vi vil integrere i arbeidsflyten vår fremover. Dette er regresjonstesting, enhetstesting og integrasjonstesting.

4.2.1 Regresjonstesting

Regresjonstesting handler om å finne feil etter at en større kodeendring har skjedd. Det gjennomføres en kontroll av funksjonalitet som tidligere fungerte for å se om den har sluttet å virke som ønsket. Feil er ofte en utilsiktet konsekvens av endringer som oppstår når den nye koden kolliderer med den eksisterende. For å teste at koden fortsatt opprettholder riktig funksjonalitet, er det vanlig å bruke testtilfeller som er brukt tidligere og kontrollere om feil kan ha gjenoppstått.

Vi ser for oss at i første omgang bør regresjonstesten inneholde alt fra funksjonelle feil til skrivefeil og dårlig navngivning. Fremover tenker vi at den beste fremgangsmåten er å felles gå gjennom hver stabile versjon av prosjektet og teste at vi ikke har gjort noen feil. Ved hver gjennomgang lager vi en rapport som inneholder:

- Hva er feilen?
- Hvordan oppstår feilen?
- Alvorlighetsgrad av feilen
- Om feilen oppstår hver gang
- Eventuelt et forslag til hvordan feilen kan løses

Alle rapportene bør samles på et felles sted hvor alle har oversikt og kan gå inn å verifisere om feilen har blitt rettet eller ikke.

4.2.2 Kontinuerlig Integrasjon

Videre tenker vi å implementere *Kontinuerlig Integrasjon* som en utviklingspraksis for rutinemessig integrering av nye versjoner for å teste endringer, så tidlig og ofte som mulig. Kontinuerlig Integrasjon består av automatiserte tester som kan kjøres uten behov for manuelt arbeid på en repeterbar måte. Det vanligste er å skrive ned noen påstander som kan validere at koden opptrer som tiltenkt. Etterhvert som prosjektet vokser seg større med flere som legger til nye endringer uten å teste, kan feilretting for bli en ekstremt stor jobb med å granske kode for å komme til roten av problemet. Dette kan løses ved å bruke automatiserte tester. Et alternativ for dette er Jenkins, som er en ledende *open-source*

automatisering server som tilbyr plugins for bygging, distribusjon og automatisering. Vi vil implementere automatisert testing i Jenkins. Dette har to av gruppens medlemmer arbeidet med tidligere.

4.2.3 A/B testing

Vi har gjennomført A/B testing for å teste designet underveis. Vi har laget flere ulike versjoner for å se hva forskjellen er. Eksempel på dette har vært om vi vil at informasjon om brukeren skal ligge horisontalt med avataren eller under avataren vertikalt. Vi har så sammenlignet versjonene og begrunnet valget basert på design prinsipper og hva vi synes ser estetisk best ut.

4.2.4 Enhets og integrasjonstesting

Det finnes flere rammeverk for å teste React. Vi har sett på Jest, som er et rammeverk opprettet og vedlikeholdt av Facebook. Jest bruker blant annet Jasmine for *assertions,* det er anbefalt å bruke disse sammen med Enzyme. Det finnes andre valg som for eksempel Mocha med Enzyme og mjackson sitt *assert library*, men grunnen til at vi valgte Jest er fordi det er en stor aktør som opprettholder biblioteket. I en enhetstest testes moduler av applikasjonen isolert. Når vi gjør *assertions* mot React-komponenter kan vi enten sjekke input (*state* og *props*) og så se på output eller så kan vi gjøre en *assertion* for å se hvordan en gitt brukers handling påvirker komponenten. Den kan for eksempel iverksette en *state*-oppdatering eller kalle en *prop*-funksjon som er hentet fra en foreldre-komponent.

Integrasjonstester er tester der flere moduler eller deler av en programvare testes sammen. For en React-app er hver enkelt komponent en individuell modul. Derfor vil en integrasjonstest innebære å teste applikasjonen som en helhet. Videre er integrasjonstester ut mot server, eller ende-til-ende tester.

4.3 Oppsett av kode

4.3.1 Kodestandard

Kodestandard definerer en programmeringsstil. Det finnes ikke noe rett eller galt i en abstrakt forstand. Det er bare et sett med regler og retningslinjer for formatering av kildekoden. Noen aspekter av kodestandard er:

- Navnekonvensjoner
- Filnavn
- Formatering og innrykning
- Kommentar og dokumentasjon
- Klasser, funksjoner og grensesnitt
- Referanse bruk
- Testing

Det er flere fordeler ved å opprettholde en uniform kodestandard blant annet lesbarhet, vedlikeholde og kompatibilitet. Alle medlemmer av utviklingsprosessen bør kunne lese koden til et annet medlem. Videre er dette med på å minimere kommunikasjon og fører til en mer ensartet problemløsning.

På dette området har vi vært ganske konvensjonelle og basert oss på standardene som for tiden er vanlig i JavaScript. Vi har brukt et verktøy som heter Prettier for å håndheve en konsekvent kodestandard. Når vi bruker Prettier spesifiserer vi hvordan vi ønsker koden formatert også oppdateres koden automatisk hver gang en fil lagres. Ettersom vi ikke er erfarne webutviklere har dette vært en enorm ressurs som vi har dratt nytte av. Det er ikke alt Prettier kan formatere for oss som for eksempel filnavn, kommentarer, osv. og da har vi referert til AirBnB sin React stilguide for frontend og Felixge sin Node stilguide for backend.

4.3.2 Versjonskontroll

For å holde oversikt over hvilken versjon av applikasjonen vi er på ser vi for oss å bruke Semantic Versioning 2.0.0. Applikasjonen sitt versjon nummer endres etter hva slags type endring som har blitt gjort. Gitt et versjonsnummer MAJOR.MINOR.PATCH økes:

- MAJOR versjon når man gjør inkompatible endringer
- MINOR versjon når man legger til funksjonalitet på en bakoverkompatibel måte
- Patch versjon når du lager bakoverkompatible feilrettinger.

Vi har for øyeblikket på MAJOR versjon null. Ved første *production build* som åpner opp for brukerregistrering vil vi øke versjonskontroll nummeret til 1.0.0 og arbeide deretter.

5.0 Kommunikasjon

5.1 Kommunikasjonsplan

Ettersom dette prosjektet er ganske omfattende vil vi ha noen retningslinjer for hvordan vi skal opptre. Kommunikasjonsplanen vår inneholder informasjon om kjernetid for gruppemøter og regler for hvordan vi skal opptre under gruppearbeidet.

Vi besluttet at kjernetiden vår skal være mandag-torsdag fra kl 09.00 til kl 15.30, totalt 6,5 timer per dag. Vi møtes i arbeidslokalene på UIA nyskapning, med mindre det er avtalt noe annet senest dagen før. Generell kommunikasjon mellom gruppemedlemmene utenom gruppearbeid skal foregå på Messenger. Viktig informasjon som også angår veileder tas på Slack eller mail. Avtale om styringsgruppemøter gjøres via mail i god tid i forveien.

Generelle regler for gruppen:

- Alle skal møte punktlig på avtalte møter.
- Alle skal si ifra hvis det er noe de trenger hjelp med.
- Alle skal si ifra i god tid hvis man ikke kan komme, eller kommer for sent.
- Alle I gruppen har like mye rett til å si sine meninger, og alle meninger skal bli hørt på av alle, for å unngå konflikt og for at alle skal bli hørt.
- Alle må godta at ekstra møter kan forekomme hvis det blir nødvendig.

Grove brudd på gruppens regler kan føre til utkastelse.

5.2 Produkteier

Dette prosjektet skiller seg fra de fleste andre bachelor prosjektene da vi opererer som vår egen arbeidsgiver. For å sørge for at arbeidsoppgaver blir utført, og at gruppens medlemmer jobbet mot det overordnede målet, valgte vi å rullere på rollen som produkteier. Alle gruppens medlemmer har hatt hver sin periode som produkteier.

5.3 Konfliktløsning

I en gruppe der alle i utgangspunktet er likestilte kan det fort oppstå konflikter eller uenigheter i beslutninger. Vi kjenner hverandre godt fra tidligere samarbeid, så vi valgte å gå for et adhocracy. Adhocracy fungerer motsatt av et byråkrati, alle gruppens medlemmer har like mye de skulle sagt i enhver avgjørelse. De fleste avgjørelsene vi har tatt i prosjektet er derfor et resultat av god diskusjon og argumentasjon. I de tilfellene vi derimot ikke har klart å bli enige, er det den som på det tidspunktet opererte som produkteier som hadde det siste ordet.

5.4 Kommunikasjonsverktøy

5.4.1 Facebook Messenger

For daglig kommunikasjon vil vi bruke Facebook sitt kommunikasjonsverktøy Messenger. Siden alle i denne gruppen har en Facebook konto, brukes Messenger til daglig kommunikasjon. Vi har brukt dette før og bruker det aktivt utenfor dette prosjektet, derfor var dette et naturlig valg. Messenger blir også brukt som et verktøy for videokonferanse. Dette ble brukt utenfor kjernetiden når vi hadde samtaler som ble tungvinte å ta på chat.

5.4.2 Slack

Slack er et populært kommunikasjonsverktøy som blir brukt mye i næringslivet. Vi bruker det til å dele viktig informasjon mellom oss og vår mentor. Dette kan være nyhetsartikler eller annen for form relevant informasjon.

5.4.3 Skype

Vår veileder Janis har vært bosatt i Wales dette semesteret, derfor har vi hatt de fleste av våre møter med han over videokonferanse. Valget av verktøy for videokonferanse falt naturlig på skype, da det er gratis, og alle parter har tidligere erfaringer med dette.

6.0 Prosjektstyring

I dette kapittelet vil vi beskrive strategier og verktøy som brukes for å opprettholde struktur og orden i prosjektet.

6.1 Dagbok

Under hele prosjektet har vi skrevet gruppedagbok, dette har vært en av metodene vi har brukt for å dokumentere arbeidet som har blitt gjort. Dette er en enkel og kosteffektiv måte å dokumentere på.

Dessverre har ikke dagboken blitt brukt på en god måte. Vi ser i ettertid at vi burde dokumentert arbeidet vi har gjort mer spesifikt. Refleksjon rundt viktige valg, og tekniske utfordringer burde blitt tatt med i dagboken. Vår bruk av dagboken har vært mer overordnet, og litt vag. Derfor fikk vi ikke utnyttet dagbokens optimale potensial, men vi har allikevel fått bruk for oversikten den har gitt oss under skriving av rapporten.

6.2 Product Backlog

Product backlog er en oversikt over oppgaver som skal gjøres i prosjektet. Vi har brukt excel til å lage product backlogen, her har vi ført inn alle oppgavene for hele prosjektet. Dette er et dynamisk dokument vi kontinuerlig har oppdatert med nye oppgaver. Vi har hentet ut oppgaver fra denne backlogen og ført dem inn i de forskjellige sprint-backlogene etterhvert.

6.3 Sprint backlog

Sprint backlog er en oversikt over oppgaver som skal gjennomføres i den utvalgte sprinten. Sprint backlog blir laget under sprint planleggingsmøte, og blir fulgt under sprinten. Oppgavene i sprint backlogen som ikke blir fullført, blir videreført til neste sprint.

6.4 Estimering av tid

For å estimere oppgavene i backlogen har vi brukt *Planning Poker*. Planning Poker er en teknikk for planleggingen og estimering av tid. Vi delte ut Planning Poker-kort med verdier på 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 og “?”(Fibonacci tallrekken). Verdiene representerer antall timer. Produkteier leste opp oppgavene fra sprintens backlog. Vi valgte så kortet med antall timer vi tror det tar å løse oppgaven. Når alle hadde valgt seg et kort ble det åpnet for diskusjon for å finne konsensus om et felles estimat. Denne prosessen ble gjentatt helt til konsensus ble oppnådd.

Underveis i prosjektet så vi bedre hensikten med å bruke Planning Poker for å estimere oppgaver. Det har vært et nyttig verktøy for å kunne planlegge arbeidet med sprintene bedre. Hvis vi hadde for lite å gjøre (antall timer lavere en prosjektert) kunne vi legge til flere oppgaver i backlog. Etterhvert som vi ble mer vant til denne prosessen opplevde vi at estimatene ble mer nøyaktige.

Vi har utført planning poker på alle våre brukerhistorier og analyseoppgaver. Vi skrev ned verdien ned på lapper, som deretter ble vist samtidig. Den av oss som var produkteier for den perioden var den som tok ansvar for prosessen. Til sammen har vi kjørt to runder med planning poker, siden noen brukerhistorier kom senere enn andre.

Utdrag runde 1:

Oppgave: Programmere feed

Navn:	**Erik**	**Ola**	**Oscar**	**Sebastian**
Tid estimert.	**89**	**89**	**144**	**144**

Tabell 5, tidsestimering utdrag 1
Estimert: 117t

Utdrag runde 2:

Oppgave: Lage funksjon som setter en unik ID på ny bruker

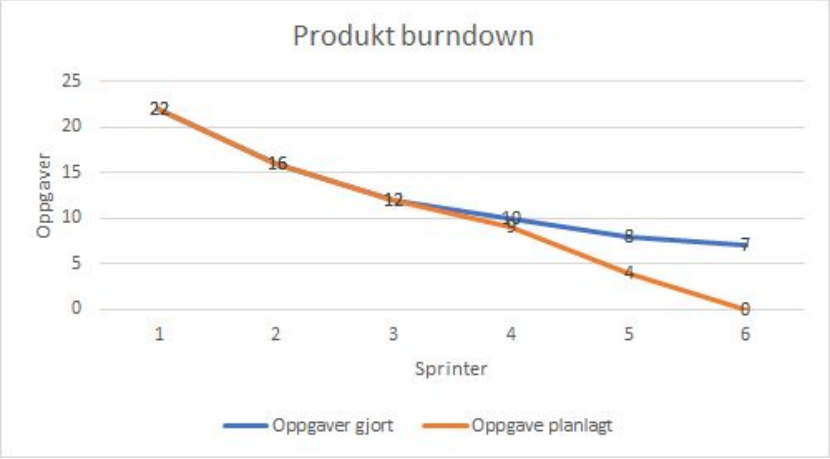
Navn:	**Erik**	**Ola**	**Oscar**	**Sebastian**
Tid estimert.	**3**	**5**	**2**	**1**

Tabell 6, tidsestimering utdrag 2
Estimert 3t.

Komplett planning poker liste ligger under vedlegg.

6.5 Burndown Chart

Dette er et visuelt målingsverktøy som viser arbeidet som gjenstår mot tiden som er igjen, man kan måle i oppgaver gjort, timer brukt eller poeng fullført. Man kan bruke burndown chart for å vise arbeidet gjort i hver sprint eller hele prosjektet. Grafen settes normalt opp med arbeidet som skal gjøres på y-aksen, og tidslinjen på x-aksen. Vi har brukt burndown chart for å vise fremgangen i prosjektet sprint for sprint. Burndown chart under viser hvordan vi har planlagt fremgangen (oransje linje), og hvordan fremgangen gikk(blå linje).



Figur 3, burndown chart

6.6 Styringsgruppemøte

Styringsgruppemøte er et statusmøte vi har med veileder og oppdragsgiver. Som nevnt tidligere i denne rapporten har vi et prosjekt som er litt utenom det vanlige. Vi er vår egen oppdragsgiver, og vi har valgt at alle gruppemedlemmene skal operere som produkteier i hver sin del av prosjektet. Siden vi er fire medlemmer i gruppen har vi hatt tilsammen fire slike møter, med ca tre ukers mellomrom.

I forkant av møtet laget vi en agenda som vi har gjennomgått punktvis under møtet. Etter dette har det vært diskusjon med veileder om videre arbeid. Gruppen har hatt stort utbytte fra disse møtene, da vår veileder kommer med konkrete innspill om hva og hvordan vi kan forbedre dette prosjektet.

Etter hvert styringsgruppemøte har vi hatt et nytt møte med bare gruppemedlemmene der vi har sett gjennom det som kom frem og diskutert dette. Her har vi laget et referat hvor vi skriver hva vi kom fram til og eventuelle endringer vi gjorde på dette møtet.

6.7 Styringsverktøy

I dette prosjektet har vi brukt flere forskjellige styringsverktøy som gjør det enklere å fordele og holde oversikt over oppgaver som skal gjøres. Vi hadde litt problemer med å finne styringsverktøy, da vi ikke hadde erfaring med noen fra før. I dette avsnittet beskriver vi de forskjellige styringsverktøyene vi har vært innom.

6.7.1 Asana

I starten av prosjektet brukte vi asana som et kalenderverktøy, der vi satt inn hvilken dager vi var borte og andre gjøremål vi hadde utenfor prosjektet. Dette var for at vi skulle få en grafisk fremstilling av hvordan dagene til gruppemedlemmene var, og når gruppemedlemmer ikke ville være tilstede hele dagen. Vi synes Asana var uoversiktlig, og det hadde ikke de funksjonalitetene vi ville ha. Vi valgte derfor å slutte å bruke det.

6.7.2 MS planner og Jira

Siden vi er medlem av UiA nyskapning, har vi tilgang til mentorer gjennom dem. Under et møte med disse mentorene ble vi anbefalt å teste Microsoft planner. MS planner er et gratis verktøy vi får gjennom Universitetet. Dette er egentlig ikke et scrum verktøy, men et generelt planleggingsverktøy med muligheten til å ‘drag and drop’ oppgaver inn i en kalender. Planner ga oss funksjonene vi ville ha, men det var ikke lagt opp for scrum, dette var noe vi måtte tilpasse planneren til.
Vi ble senere tipset av en medstudent om Atlassian med deres verktøy Jira, som blir brukt mye i næringslivet. Dette verktøyet krever månedlig betaling, men det var mulighet for en prøveperiode. Vi bestemte oss for å prøve verktøyet, men vi var ikke helt overbevist da det krever betaling.

6.7.3 Google sheet

Jira fungerte greit, men vi følte ikke vi hadde full oversikt, og vi var redd for å bruke for mange timer på å sette oss inn i enda et verktøy. Vi brukte Jira som et styringsverktøy i de første sprintene, før vi innså at alle funksjonene vi brukte i Jira også var mulig i et regneark. Vi valgte derfor tilslutt å lage sprint og produkt backlog i regneark. Google har et eget verktøyet for å lage regneark, som fungerer

sammen med google docs. Dette er gratis i motsetning til Jira. Vi brukte kunnskapen vi har lært i IS-200 og laget disse backloggene manuelt.

6.7.4 Google Drive

Dette verktøyet bruker vi til å lagre alt av dokumenter som blir produsert i dette prosjektet. Her kan vi også jobbe på samme dokument ved hjelp av Google Docs eller sheets, som gjør at vi ikke er tvunget til å jobbe på samme kontor for at flere skal kunne bidra. Vi har valgt dette verktøyet på bakgrunn av at alle på gruppen har brukt det før. Vi unngår da å måtte bruke tid på å sette oss inn i noe nytt. Vi har også delt vår google drive med vår mentor, slik at han kan følge med på hva vi gjør, og har lett tilgang til dokumenter vi har lyst til at han skal se på.

7.0 Teknologi

I dette kapittelet skal vi fortelle om teknologien vi har brukt, og hvorfor vi har valgt det.

7.1 Git & GitHub

Vi har valgt å bruke Git som verktøy for versjonskontroll, og vi har lastet opp vår kode på Github.com så den enkelt kan deles med personer utenfor gruppen. Dette verktøyet er enkelt å sette seg inn i og bruke. Hele gruppen har kjennskap til dette verktøyet, og det var et enkelt valg siden vi vet at det er et stort samfunn rundt det. Git er enkelt å implementere i vår IDE Webstorm som gjør det enkelt å oppdatere repositoryen med nye forandringer av kode.

7.2 HTML

HTML er et formateringsspråk benyttet for å lage hypertekst-dokumenter på Internett (World Wide Web). Formateringsspråket er industristandard og noe som er vanskelig å komme unna om vi skal lage en web-applikasjon.

7.3 CSS

CSS(Cascading Style Sheets) er et programmeringsspråk for å stilsette HTML-dokumenter på Internett.

7.4 JavaScript

JavaScript er primært utviklet for å legge til interaktivitet på websider og og opprette webprogrammer. Klient-side JavaScript-programmer kan legges direkte inn i HTML-kilden til websider. Avhengig av utviklerens hensikt kan koden kjøre når brukeren åpner nettsiden, klikker på noe, skriver inn noe, sender inn et skjema eller forlater siden. JavaScript er også et objektorientert språk med prototypisk arv. Språket støtter flere innebygde objekter, og programmer kan lage eller slette egne objekter.

7.5 React

React er et JavaScript-rammeverk som gjør det enkelt å konstruere og vedlikeholde et brukergrensesnitt. Det blir ofte omtalt som et rammeverk for å bygge Single Page Applications, men det er i bunn og grunn et visningsbibliotek. React er V-en i MVC (modellvisningskontrollen). En av fordelene med å bruke React er at du kan dele opp brukergrensesnittet i flere komponenter. Hver enkelt komponent kan ha oversikt over sin egen state. Dette gjør det enklere å utvikle applikasjoner som må holde oversikt over mange ulike states. Ved å bruke react kan hele siden lastes opp, og funksjoner kjøres ikke før de bli aktivert av bruker, dette gir god ytelse. Dette er spesielt viktig ettersom mange velger å forlate en nettside om den har for lang lastetid. Ytelsesforbedringen kommer av at React bygger opp applikasjonen med en falsk representasjon av DOM eller Document Object Model. Noe React kaller en virtuell DOM. En virtuell DOM er et tre av JavaScript-objekter som representerer den faktiske DOM. Idéen om å gjenskape hele DOM-resultatet på hver oppdatering resulterer i en utviklingsmodell som er enkel å forstå; istedenfor å holde oversikt over alle DOM-tilstandsendringer, trenger vi kun kun å returnere den DOM-en vi ønsker å se. React tar seg av denne transformeringen bak kulissene. Det er er kompliserte prosesser, enkelt forklart gir dette god ytelse ved at den virtuelle DOM-en vil bruke ulike diffing algoritmer for å vite hva som har endret seg.

7.6 Node & Express

Node.js er et åpent kryssplattform runtime-system for server- og nettverksapplikasjoner. Ved hjelp av Googles' V8-motor kan Node eksekvere JavaScript kode slik at JavaScript kan brukes på servere. Altså Node.js er en JavaScript runtime som kan brukes til å kjøre kode utenfor nettleseren.

Express er et bibliotek som kan bruke Node sin runtime. Express er ikke en frittstående kodebase, men en samling funksjoner eller hjelpere som gjør det lettere å jobbe med HTTP- aspektene ved Node.js.

7.7 Prisma

Prisma er en GraphQL database proxy som gjør databasen vår til en GraphQL API. Denne API-en gir tilgang til CRUD operasjoner for datamodellen. Dette kan vi gjøre med hjelp av GraphQL binding som hjelper med å jobbe med GraphQL API-en. En GraphQL binding generer dedikerte JavaScript-funksjoner for hver API-operasjon. Sammen med Prisma bruker vi GraphQL-Yoga som er en GraphQL server er bygget på Express og Apollo.

7.8 Webpack

Webpack er en module bundler for JavaScript. Prosjektet vårt består av mange komponenter som kan sees på som egne moduler. Alle modulene som er avhengige av hverandre må bevare koblingen mellom seg for at applikasjonen skal fungere. Webpack samler alle ressurser i en

HTML-fil som fungerer som prosjektets inngangsport og sørger for at alle modulene bevarer koblingen mellom seg og blir plassert i riktig rekkefølge.

Prosjektet vårt består av mange statiske filer og Webpack gir oss en klar fordel med at vi kun trenger å konfigurere én gang og så være ferdig med det. Inntil ES6 (ECMA Script 6) ble det brukt en rekke

forskjellige løsninger for å lage modulær JavaScript. Enkelte løsninger fungerer bare i nettleseren, avhengig av nettleser miljøet (med tilstedeværelse av window) mens andre fungerer kun i Node.js. Nettlesere støtter ennå ikke ES6-moduler. ES6 sin syntax er intuitiv og lar oss unngå bizarre taktikker brukt med ES5, og de fungerer både i og utenfor nettleseren. Med andre ord, Webpack håndterer alle våre statiske filer, konfigurerer prosjektet vår og lar oss skrive logisk kode som er lettere å forstå.

7.9 Yarn

Yarn er et pakkesystem (ofte kalt pakkebehandler) som er en samling verktøy for å automatisere prosessen ved å installere, oppgradere, konfigurere og fjerne programvarepakken fra et utviklingsprosjekt. Npm er standard pakkebehandler for Node.js plattformen og er blitt defacto standard for pakkebehandling i JavaScript verden. Problemene oppstår imidlertid når man jobber med prosjekter med mange avhengigheter. Installasjon av pakker med npm gjøres på en ikke-deterministisk måte, slik at rekkefølgen og versjonen kan være forskjellig fra person til person. Videre krever også installasjonsprosessen for npm en Internett-tilkobling fordi hver pakke er lastet på installasjonstid, det er ikke noen caching-mekanisme. Det er altså ingen mulighet til å bruke pakkebehandleren i et offline-scenario, og selve installasjonsprosessen er tidskrevende.

Det er to hovedgrunner til at vi bruker Yarn istedenfor andre pakkebehandlere og det er fordi: Yarn kan arbeide i frakoblet modus. Den har en cachemekanisme, så avhengigheter som er lastet inn en gang lastes inn i Yarn cache. Hvis de blir etterspurt, kan Yarn hente dem fra hurtigbufferen uten å laste dem fra Internett. Yarn kjører installasjonen i en deterministisk modus. Strukturen til node_modulene er nøyaktig den samme på hver maskin der Yarn installasjonen utføres. Vi slipper å ende opp med å håndtere problemer knyttet til å ha en annen installasjonsstruktur.

7.10 GraphQL

Vi planlegger å implementere GraphQL i prosjektet vårt. GraphQL er en ny API standard som gir et mer effektivt, kraftig og fleksibelt alternativ til REST. Det ble utviklet av Facebook men de gjorde GraphQL open-Source. Nå blir det vedlikeholdt av et stort samfunn av bedrifter og enkeltpersoner fra hele verden.

Kjernen av i GraphQL er at dataen er deklarativ og at klienten kan hente ut eksakt hvilken data den trenger for en API. I stedet for flere endepunkter som returnerer faste datastrukturer, viser en GraphQL-server bare et enkelt sluttpunkt og svarer med nettopp de dataene klienten ba om.

Vi vil følge den mest vanlige arkitekturen som enkelt og greit er GraphQL med en database. Vi konfigurer en server som implementerer GraphQL-spesifikasjonen. Når en forespørsel kommer til SQL-servereren, leser serveren “payloaden” og henter nødvendige data fra databasen. Deretter konstrueres det et svar-element som beskrevet i den offisielle spesifikasjonen og returnerer den til klienten.

7.11 Apollo

Apollo er en samling av teknologier som kan legges til i prosjekt *stacken*. *Apollo Client* brukes for å koble data til brukergrensesnittet. *Apollo Engine* brukes for infrastruktur og verktøy som innebærer

blant annet caching og feilsøking. *Apollo Server* er tredje familiemedlem i Apollo og brukes for å oversette REST-API og backends til et GraphQL-skjema.

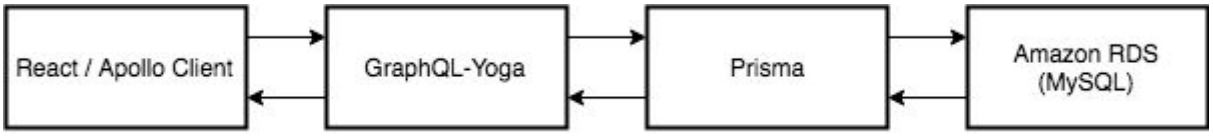
7.12 Adobe XD

Adobe XD (Experience Design) er et program for design av brukeropplevelser, utviklet og utgitt av Adobe Systems. Programmet støtter vektordesign, wireframing og gir muligheten for å lage enkle interaktive prototyper. XD tillater å sømløst iterere og dele interaktive prototyper med andre på tvers av enheter og plattformer, som Windows, Mac, iOS og Android.

Vi har valgt å arbeide med Adobe XD da dette programmet krever lite forhåndskunnskaper, det fungerer på tvers av alle plattformer, og det føles mer intuitivt enn motparter som Sketch og InVision.

8.0 Systemarkitektur

Overordnet består arkitekturen til løsningen av fire tydelig definerte lag med mange ulike oppgaver. React og Apollo Client brukes på klientsiden for blant annet UI og caching. GraphQL Yoga er en GraphQL Server bygget på rammeverk som Express og Apollo Server. Hovedoppgaven er å håndtere forespørsler og respons. Prisma er et grensesnitt som fungerer som proxy og gir tilgang til CRUD operasjoner. For øyeblikket er kun Amazon RDS og Amazon Aurora det eneste som er støttet for database hosting gjennom Prisma Cloud. Det arbeides med å få en MongoDB *database connector* ut fra beta. Enn så lenge bruker vi Prisma Cloud for å redusere kompleksitet og da er vi låst til Amazon RDS med enten PostgreSQL eller MySQL.



figur 4, systemarkitektur

Applikasjoner i dag er ofte data-intensive, i motsetning til beregningsintensive. Rå CPU-kraft er sjelden en begrensende faktor for disse programmene. Større problemer er vanligvis mengden data, datas kompleksitet og hastigheten som den endrer seg på.

For å utvikle vår applikasjon er det forskjellig funksjonalitet vi må utnytte:

- Lagre data, så vi kan finne det ingen senere (*database*)
- Huske resultatet av en dyr operasjon, for å kunne øke hastigheten (*caches*)
- Prosessere en større mengde data ved forespørsel (*batch processing*)

Det er noen faktorer (basert på kvalitetssikring) vi må ta med i beregningen når vi skal velge teknologi for å gi oss spesifisert funksjonalitet. Dette er systemets pålitelighet, skalerbarhet og vedlikehold.

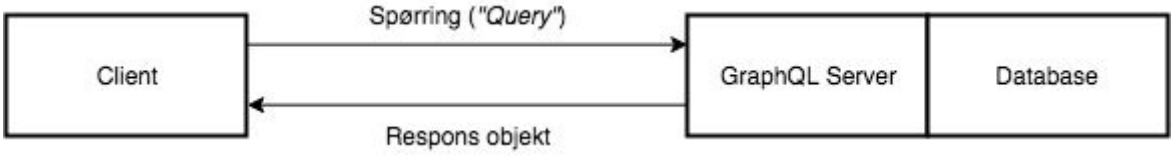
- **Pålitelighet** - Systemet bør fortsette å fungere korrekt selv om programmet blir utsatt for feil (maskinvare, programvare eller menneskelige feil)
- **Skalerbarhet** - Etterhvert som systemet vokser i datavolum, trafikkvolum eller kompleksitet, bør det finnes rimelige måter å håndtere denne veksten.
- **Vedlikehold** - Over tid vil det være flere forskjellige personer som jobber med å vedlikeholde systemet eller å tilpasse det for nye bruksområder. Alle skal kunne arbeide med systemet produktivt.

8.1 Database

Først ut er arkitekturen sin database. Det er en del forskjeller som må tas med i betraktning når man skal velge mellom en relasjonsdatabase eller en NoSQL-database. En database med relasjonsmodell gir bedre støtte for *joins*, mange til én og mange til mange relasjoner. Derimot en ikke-relasjonell NoSQL database som f.eks MongoDB som følger en dokument modell kan gi *schema* fleksibilitet og bedre ytelse. Andre typer NoSQL-databaser som ulike grafdatabaser er best egnet for bruk ved tilfeller der noe potensielt er relatert til alt.

Det er ikke uvanlig at større applikasjoner består av flere databaser for ulike formål. For vår applikasjon finnes er det mange fordeler og ulemper å ta med i betraktningen, men vi har en fordel med at GraphQL er database uavhengig. Du skriver en GraphQL server som kobles opp mot en database av eget valg. GraphQL er har også et agnostisk transportlag og kan dermed potensielt brukes med alle tilgjengelige nettverksprotokoller (f.eks TCP, WebSockets, osv.).

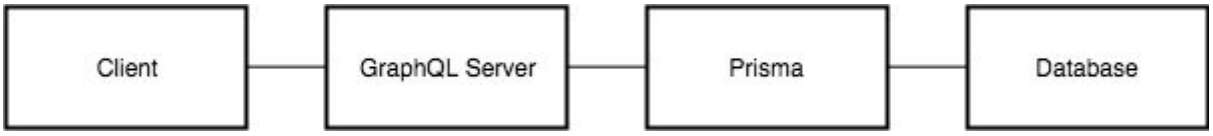
Vår arkitektur på det mest overfladiske består av en GraphQL server med en tilkoblet database. I dette oppsettet har vi en enkelt server som implementerer GraphQL-spesifikasjonen. Når GraphQL-serveren får en spørring (*"query"*), leser den spørringens nyttelast (*"payload"*) og henter den nødvendige informasjonen fra databasen. Deretter konstrueres det et respons objekt som returneres til klienten.



Figur 5, graphQL query

8.2 Prisma

Videre bruker vi Prisma som er et API-lag som sitter foran databasen. Prisma er et grensesnitt for databasen og fungerer som en proxy som gir tilgang til CRUD operasjoner. GraphQL fungerer på den måten hvis du har tilgang til et endepunkt i en GraphQL-API, kan du hente hele database-skjemaet gjennom det endepunktet.



Figur 6, Prisma

Vi må derfor ha to GraphQL API-lag. Et *applikasjonslag* og et *databaselag*. Applikasjonslaget inneholder all *business logic* som autentisering, tredjepart tjenester, etc. Databaselaget leveres av Prisma og er som nevnt tidligere et GraphQL basert grensesnitt for databasen. Dette laget hjelper til med å skrive spørringer mot databasen. Dette er fordi Prisma speiler en database-API, slik at vi kan utføre CRUD-operasjoner for bestemte datatyper.

Når datamodellen (samling av datatyper) er definert i GraphQL sitt *Schema Definition Language* (SDL), oversetter Prisma database-skjemaet og setter opp den underliggende databasen tilsvarende. Dette er en prosess som blir automatisk generert av Prisma etter datamodellen vi oppga.

For at GraphQL-serveren skal snakke med Prisma brukes *Prisma Bindings.* Det er et lag for å bygge GraphQL-servere på toppen av Prisma-tjenester. Det delegerer utførelsen av spørringer (eller mutasjoner) til API for den underliggende Prisma-databasetjenesten.

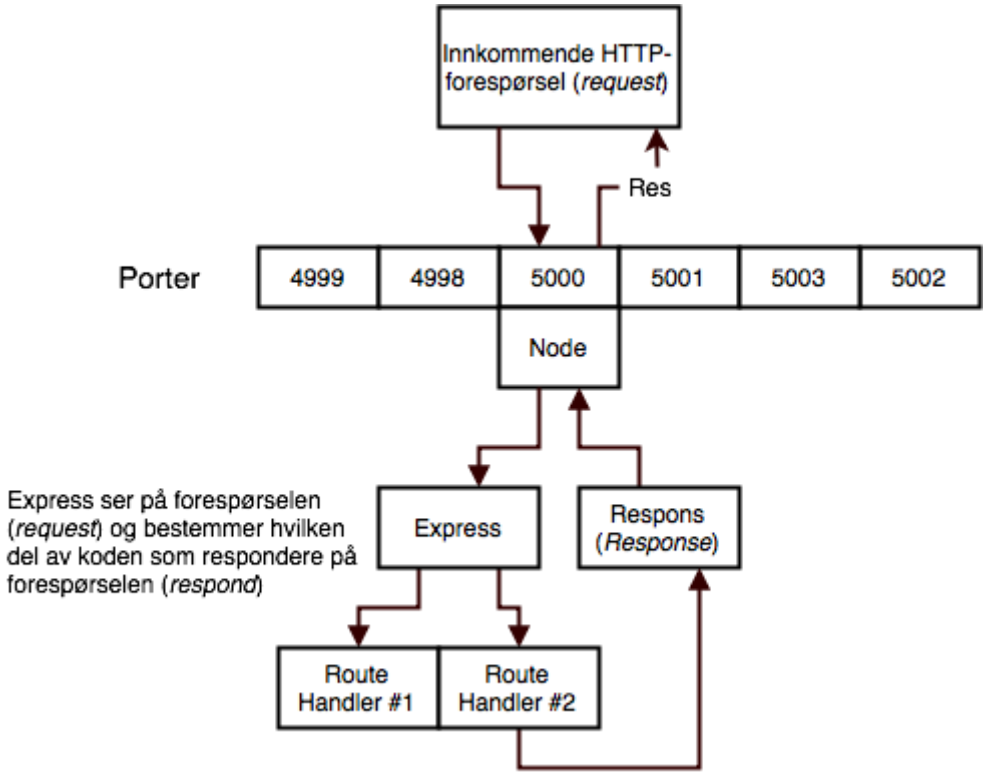
8.3 Batching

Vanligvis foregår *batching* på server-siden hvor en tjeneste venter på at en forespørsel eller en instruksjon fra klienten skal komme. Når en forespørsel er mottatt, forsøker tjenesten å håndtere den så raskt som mulig og sender et respons tilbake. Responstid er vanligvis det primære målet for ytelsen til en tjeneste sammen med tilgjengelighet (hvis vi ikke når tjenesten, får vi en feilmelding).

GraphQL Yoga bruker Prisma bindinger som bruker *DataLoader* mønsteret til batch forespørsler til Prisma API. Det foretatt batching i form av at klienten kan gjøre flere spørringer i en forespørsel. Se for deg at vi trenger data til flere komponenter innen et kort tidsintervall, istedenfor å sende alle separat, kan disse sendes sammen i en enkelt runde.

For GraphQL-serveren bruker vi GraphQL-Yoga som er bygget på toppen av blant annet Express og Apollo-server. Her er det mye abstraksjon så det viktigste er å forstå hvordan HTTP-forespørsler utføres og for å se på hvordan dette fungerer kan vi se på sammenhengen i Express og Node.

8.4 Sammenhengen mellom Express og Node



Figur 7, Express og Node

Ved å kjøre en server på en lokal maskin ser serveren etter HTTP-trafikk på en enkelt port. Vi kan ha et innkommende *request* utstedt av nettleseren som kommer inn på bestemt port på vår lokale maskin. Node og Express konfigureres til å se etter trafikk som prøver å få tilgang til en bestemte porter på vår maskin. Node ser spesifikt etter informasjon fra den porten og tar denne informasjonen som strømmer inn fra denne HTTP-forespørselen, for å så overlate det til Express siden av applikasjonen vår.

Express vil da se på forespørselen og bestemme hvilken del av logikken på innsiden av Express programmet som skal håndtere eller på respons til denne innkommende forespørselen. I Express skriver vi det som kalles *Route handlers*. En *route handler* brukes til å håndtere en forespørsel som ber om en svært spesifikk tjeneste. Disse tjenestene kan for eksempel være en *handler* som er ansvarlig for autentisere en bruker eller opprette en post. Altså lager vi disse *route handlers* for å behandle den innkommende forespørselen og generere noe utgående svar. Dette svaret blir så sendt tilbake til Node-prosessen. Node vil da svare på forespørsel med en respons som vi har skrevet. Respons vil så bli sendt tilbake til den som gjorde den innkommende HTTP-forespørselen.

8.5 Klient

Vi har sett på hvordan det gjøres HTTP-forespørsler underliggende. Det fine med en GraphQL klient at vi kan unngå arbeide med nettverksoppgaver på et så lavt nivå. Med de ekstra lagene er det eneste vi trenger å deklarere krav til data og la systemet ta seg av å sende forespørselen og håndtere responsen.

Når responsen har blitt mottatt og håndtert av GraphQL-klienten må dataen ende opp i brukergrensesnittet på et vis. Konseptet som omhandler *higher-order components* i React og GraphQL kan brukes for å hente de nødvendige dataene under panseret og gjør den tilgjengelig som `props` for React komponentene.

8.6 Caching

Vi ønsker å opprettholde en cache av dataene som tidligere har blitt hentet fra serveren. Det er viktig å informasjon cachet lokalt for å gi en bedre brukeropplevelse og redusere mengden data brukeren må laste inn. Når man cacher data er det vanlig å legge informasjon som hentes eksternt inn i *localstore*, hvor den kan hentes igjen senere. Apollo Client hjelper til med å optimalisere *cachingen* av data ved å normalisere dataene på forhånd. Localstore vil kun inneholde individuelle poster som kan refereres til med en globalt unik ID.

Altså bruker vi React som et front-end rammeverk for å bygge brukergrensesnitt og Apollo Client for oppgaver som er repeterende for applikasjon. For eksempel er dette å sende forespørsler eller mutasjoner uten å måtte arbeide direkte med lavere nettverk detaljer eller vedlikeholde en lokal cache.

9.0 Design

Vi har designet Ecoinomy etter etablerte designprinsipper som vi lærte om i emnet IS-104 *Brukergrensesnitt*. Hovedarbeidet har gått på å utarbeide et skjermbasert grensesnitt og å skape en god brukeropplevelse.

Brukeropplevelse (Brukbarhet) er hvordan vi opplever å bruke et produkt, et system eller tjeneste. Begrepet er relativt nytt og ble først brukt av Don Norman. Han skriver i boken *Emotional Design: why we love (or hate) everyday things* at vi oppfatter produkter som enklere å bruke hvis de gjør at vi føler oss bra, og at vi gjerne liker produkter bedre hvis de er pene å se på, til og med i den grad at vi kan tilgi dårlig funksjonalitet.

Det er spesielt viktig for å oss å skape et produkt med ett mest mulig appellerende design fordi vi ikke kommer til å lansere en fullverdig plattform i første omgang, men ett *minimum viable product*. Altså den enkleste versjonen av noe som har verdi. Det gir oss noe å evaluere og lære fra. Vi har valgt et minimalistisk design slik at forkunnskapene for å ta i bruk tjenesten er så liten som overhode mulig. Målet er å i så stor grad som mulig ta hensyn til brukerens allerede etablerte *mentale modell.* Det er en modell som er bygd opp av gjenkjennelse og baserer seg på ett inntrykk av hvordan vi forventer at løsninger skal fungere. Eksempelvis om vi skal finne ut hvor menyen er plassert, starter vi som oftest oppe i ett av hjørnene på brukergrensesnittet. Ecoinomy skal være så intuitiv som overhodet mulig.

9.1 Brukersentrert design

Det er viktig å forstå brukerne, deres behov, situasjon og ikke minst involvere faktiske brukere gjennom hele designprosessen. *Design Tenkning* (“*design thinking”*) er en metodikk som handler spesifikt om dette. Ved å arbeide etter en brukersentrert prosess får vi innsikt i hva brukerene trenger og vil ha. God dialog med brukere gjennom designprosessen kan hjelpe oss med å designe en tjeneste

som kan gi brukeren en følelse av *Instant Gratification*, som går ut på at brukerens handling samsvarer med et ønske de har sett for seg, og så oppfylles dette uten forsinkelse eller utsettelse. Det hjelper derimot ikke kun å fokusere på brukersentrert design for å lage gode produkter. Vi må også lage noe som dekker behov, er teknisk gjennomførbart og i tråd med forretningsstrategien. Produkter skal være attraktive, gjennomførbare og ha et eksistensgrunnlag. Å ta utgangspunkt i etablerte designprinsipper og retningslinjer er et godt utgangspunkt for oppfylle kravet om at produktet skal være attraktivt og intuitivt.

9.2 Designprinsipper

Designprinsipper kan sees på som regler for godt design. Designprinsippene er utarbeidet ut i fra hvordan mennesker oppfatter, lærer og husker ting. Når vi skal designe eller evaluere gir designprinsippene et godt utgangspunkt for forståelsen av hva som fungerer og ikke fungerer. Elementer i designet kan også være designet med tanke på ett eller flere designprinsipper.
Vi har tatt i bruk Don Normans prinsipper for design. Han har beskrevet prinsippene om synlighet, sammenheng, tilbakemelding, konsistent design, begrensninger og hint som skal hjelpe oss med å utforme grensesnittet mest mulig brukbart.
Spesielt fokus har ligget på Gestaltprinsipper for persepsjonspsykologiske forklaring på hvordan vi sanser og organiserer visuelle inntrykk i design av nettsiden. For å nevne noen Gestaltprinsipper som er brukt:

- **Likhet** - Vi har designet elementer med lik funksjonalitet slik at de har en likhet, både i form og farge. Dette gjør at brukeren lettere vil se sammenhengen mellom elementer med lignende funksjoner. Vi kan få elementer til å skille seg ut ved å endre fasong og farge. Eksempel på dette vil være “Report”-knappen på et innlegg. Denne knappen vil ha en annen fasong og farge for at brukere skal skjønne at det ikke er sammenheng mellom “Report” eller det å “Like” innlegget.
- **Gruppering** - Vi har plassert elementer som har sammenheng med hverandre i nærheten av tilhørende elementer og funksjoner. Dette gjør det enklere å forstå hva som til hører hva. Dette brukes også sammen med prinsippet om likhet; elementer som er like og gruppert sammen vil ofte oppfattes som en helhet selv om det består av separerte elementer.
- **Symmetri og Orden** - Dette er et viktig prinsipp å opprettholde om vi vil at lærekurve skal være så liten som mulig. Dersom brukeren kommer inn på siden og designet er asymmetrisk og ubalansert, vil brukeren sløse mye tid på å finne funksjoner og elementer som trengs for å bruke plattformen. Ved å bruke symmetri i designet vil brukeren mer intuivit finne frem til forskjellige funksjoner og løsninger i vår plattform.

Vi har også arbeidet etter Jeff Johnson sine prinsipper som omhandler hvordan vi går frem når vi designer. Johnson har utarbeidet prinsippene:

- Fokuser på brukerne og oppgavene, ikke teknologien
- Funksjon før presentasjon
- Design fra brukerens ståsted
- Design for de vanligste oppgavene
- Ikke distrahere brukerne fra oppgavene de skal gjøre
- Tilrettelegge for læring
- Gi informasjon, ikke bare data

- Design for responsivitet
- Test og fiks

9.3 Vår designprosess

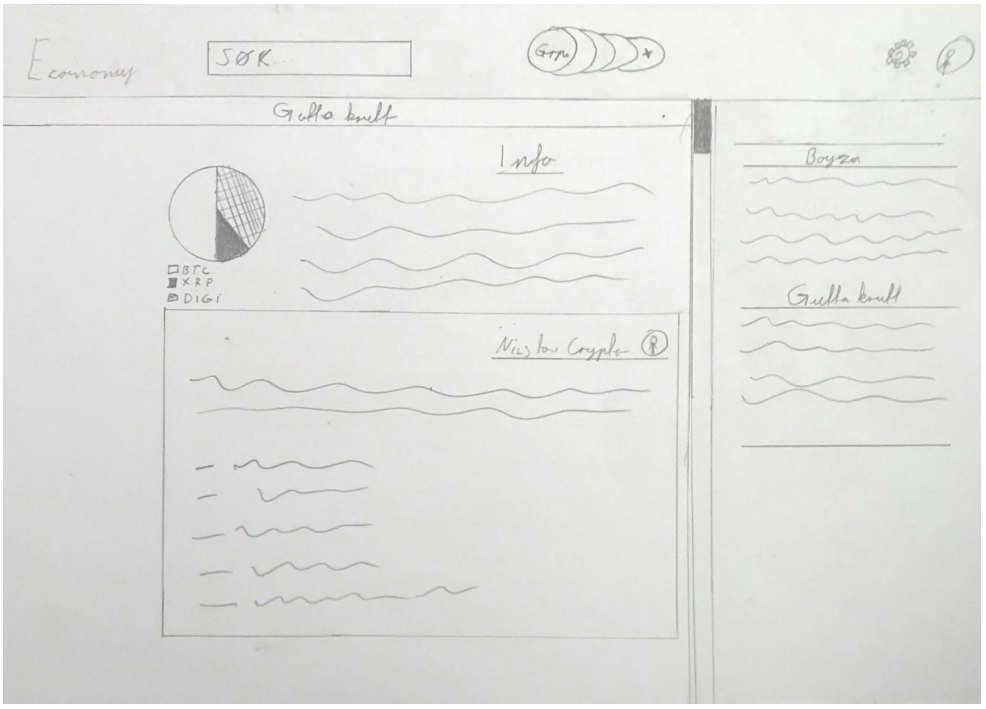
Vår designprosess har hovedsakelig bestått av trinnene utforskning, utforming, og utvikling. Deretter kom evaluering som gjøres på flere stadier i prosessen. Vi har først fått en konseptuell modell på plass, en overordnet idé om hva som skal lages. Dette er basert på hva vi har tilegnet oss av brukerinnsikt som vi tok med oss når vi skulle skissere ut løsninger. Prosessen kan defineres som brukerinnsikt til konsept, krav for løsningen, så laget vi brukerhistorier knyttet til brukerbehov og hvordan vi skal løse disse. Deretter har vi begynt å tenke på hvordan vi skal strukturere innhold og elementer.

9.4 Papirprototype

Papirprototype er et godt verktøy for å tidlig kunne utvikle design på en enkel og billig måte. Papirprototypen kan brukes til brukertester for å finne funksjonalitetsproblemer med designet. Vi har brukt papirprototypen for å kunne avklare våre antagelser av kravene brukerne stiller til designet. Ved å utføre brukertester finner vi eventuelle feil i design og funksjonalitet som kan rettes opp før det blir laget en digital prototype. Det er lønnsomt å starte med å papirprototype for å unngå mer arbeid dersom det oppstår feil i brukergrensesnittet. Noe som kan føre til store kostnader om feilen er stor. Det er derimot langt mye lettere å forkaste en idé og tegne opp på nytt enn å implementere for å så finne ut at det ikke fungerer.

Vi begynte med å lage papirprototyper tidlig i prosjektet for å kunne utnytte alle fordelene det kan gi oss. Allerede ved første papirprototype, før brukertester, fant vi flere problemer og nye løsninger vi ikke hadde tenkt på før det var designet på papir. Det er også vært en god fremgangsmåte for å få alle sine meninger med når vi har designet brukergrensesnittet.
Første utkast av designet så slik ut:

Figur 8, papirprototype - portfolio



Figur 9, papirprototype - gruppe

9.5 Designtest

Testing av papirprototype:
Vi valgte å gjøre to sesjoner med brukertesting av papirprototype – første gang med det første designet, og så en gang til etter vi har rettet opp og endret på designet som et resultat av første brukertest gjennomgang. Hensikten med brukertesten er å få en reell tilbakemelding fra potensielle brukere om hvordan designet fungerer og hvor lett det er å navigere seg på siden.

Vi har gjennomfør brukertestene ved å gi testobjektet forskjellige scenarier som skal utføres. Disse scenariene er:

1. **Lage ny bruker.**
    - **Registrere bruker.**
    - **Gå inn på profil siden.**
2. **Legge til innlegg.**
    - **Skrive innlegg**
    - **Publiser det.**
3. **Upvote / downvote et innlegg.**
    - **Finne et innleg**
    - **Enten up eller down vote.**
4. **Lage ny gruppe.**
    - **Lage gruppen.**
5. **Legge til ny krypto valuta på profilen.**
    - **Gå tilbake til feed**
    - **Naviger til profilside**
    - **Se hva du har investert i.**
    - **Legge til 100 ripple (XRP) til profilen.**

Vi bruker også teknikken «think aloud». Vi har valgt denne teknikken fordi flere i gruppen har brukt den tidligere i faget IS-104, og den har fungert veldig bra. Denne typen brukertest utføres ved at vi som observatører gir testobjektet noen scenarier som skal utføres mens testobjektet snakker høyt om hvordan h*n tenker mens h*n prøver å gjennomføre scenariet. Vi som observatører skal hjelpe så lite som mulig med oppgaven, men vi vil gi hint dersom testobjektet står fast. Testobjektene vil være studenter ved UIA.
Disse testen ble gjennomført av Erik og Ola. Erik hadde som rolle å dokumentere og gi testpersonen scenarioene de skulle gjennom. Ola hadde rollen som datamaskin, det innebærer å bytte scener når brukeren trykker på papirprototypen. Begge var observatører, og observerte hvordan testpersonen gjennomføre scenarioene.

Alle brukertestene vil være lagt til i vedlegg, men vi tar her med en test fra første runde med brukertester, og en test fra andre runde.
Punktene under oppgavebeskrivelsen er notatene som ble gjort under utførelsen av brukertesten.

Test runde 1: student X

1. **Lage ny bruker.**
    - Registrere bruker.

- Navigerer seg riktig, skjønner godt hvor han skal trykke

2. **Legge til innlegg.**
    - Skrive innlegg

- Finner ikke noe sted å legge til innlegg
    - (Bør endre slik at man kan legge til innlegg fra profilside)
- navigerer til feed for å se om man kan legge til innlegg der
- Finner ikke hvor man skal legge til innlegg
- Trenger hjelp

- Finner pluss-tegn (må gjøres bedre, ikke beskrivende nok med bare plusstegn)
    - Publiser det.
- Klarer å publisere

3. **Upvote / downvote et innlegg.**
    - Finne et innleg
- Finner innlegg
    - Enten up eller down vote.
- Skjønner godt hva pilene betyr, upvoter innlegget

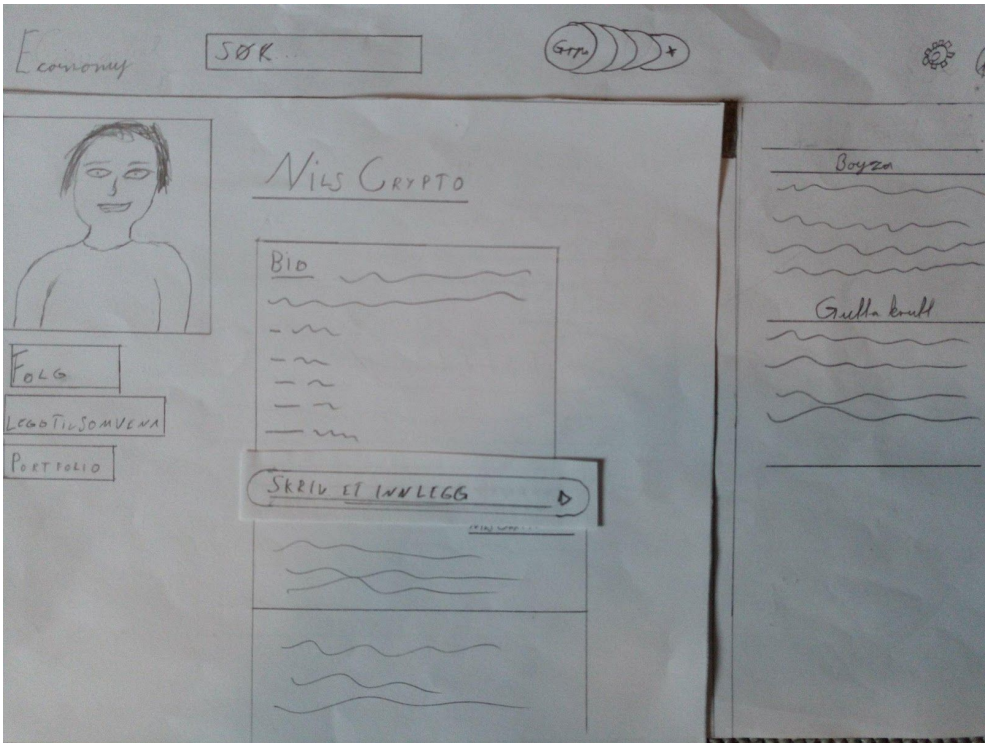
4. **Lage ny gruppe.**
    - Lage gruppen.
- + tegn ved grupper
- Veldig godt gjennomført, forstår hva plusstegnet betyr (mye fordi det står "grupper" i nærheten av det)

5. **Legge til ny krypto valuta på profilen.**
    - Gå tilbake til feed
- Finner veien tilbake fint
    - Naviger til profilside
- Navigerer seg lett til profilside
    - Se hva du har investert i.
- Trykker portfolio
    - Legge til 100 ripple (XRP) til profilen.
- Add
- Syntes dette var veldig greit
- Ønsker innkjøpspris når man legger til valuta, slik at man kan følge med på prisen fra når man faktisk kjøpte.

Første testen ga oss mye god tilbakemelding, vi så da at det er mye som måtte endres. Det er veldig lett å se seg blind på enkelte elementer når man jobber så intenst med de, og derfor er det veldig fint å kunne få input fra friske øyne på denne måten.

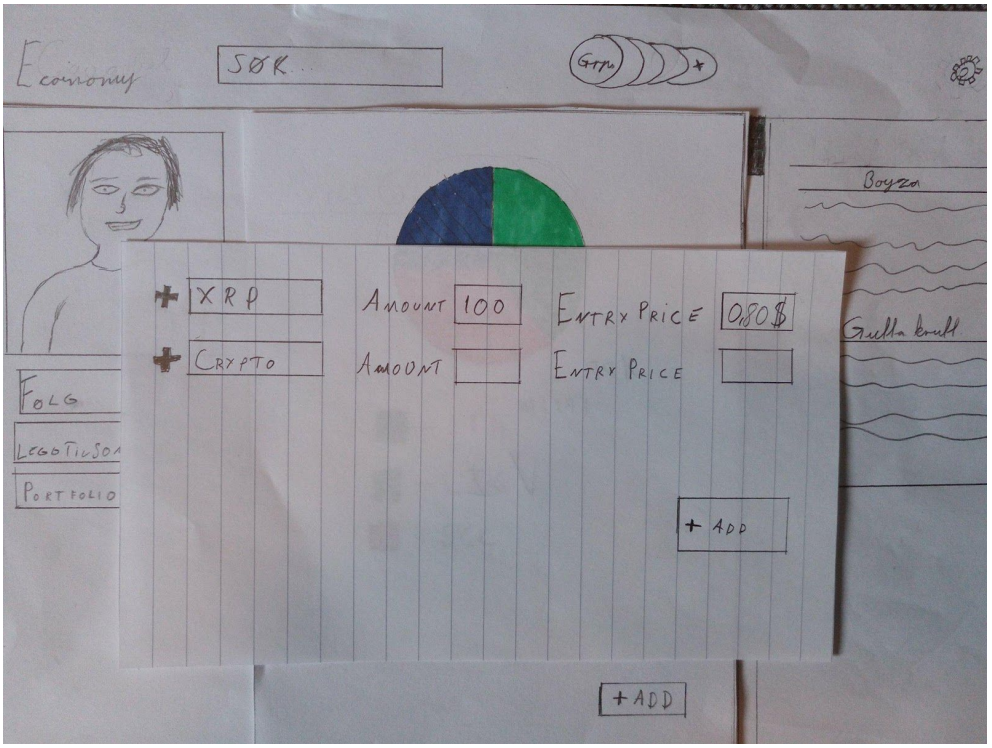
Brukertesten ovenfor oppsummerer godt de viktigste punktene vi har tatt med oss videre fra denne runden med testing av papirprototypen. For det første er det ikke like lett å legge ut innlegg som vi trodde. Plusstegnet på feed-siden er ikke beskrivende nok i seg selv til at testobjektene med en gang skjønte at det var her man måtte trykke for å legge til et innlegg. I tillegg til dette hadde det også vært naturlig å kunne legge til et innlegg fra profilsiden da flere av testobjektene ikke skjønte at man skulle navigere seg vekk derfra og til Feed for å kunne gjøre dette. Vi fikk også tilbakemelding på at det burde være mulig å sette inn prisen valutaen var når man kjøpte den, slik at man kan se fortjenesten.

Annet enn dette fikk vi god tilbakemelding på de andre designelementene, upvote pilene fungerte bra og trengte ikke videre forklaring, og generell navigering på siden var enkel og intuitiv.



Figur 10, papirprototype forbedret - profil

Forbedret prototypen ved å legge til et "skriv innlegg felt" på profil og feed.



Figur 11, papirprototype forbedret - portfolio

Endret designet ved å legge til kryptovaluta, la til en “entry price” der bruker vil legge til prisen de kjøpte coin/token for. Dette er data som må inn for at bruker skal kunne få oversikt over opp eller nedgang i investeringene sine.

Test runde 2: student Y

1. **Lage ny bruker.**
    - Registrere bruker.

- Finner lett frem, får registrert bruker

2. **Legge til innlegg.**
    - Skrive innlegg .

- Den nye knappen for å skrive innlegg fungerer bra, testobjektet finner lett ut av hvordan man skal legge til innlegget. Innlegget blir postet fra profilside.
    - Publiser det.
- Finner knapp for publisering.

3. **Upvote / downvote et innlegg.**
    - Finne et innleg

- Går inn på grupper, testobjektet trodde man måtte inn på grupper for å finne innlegg, ikke feed. Kanskje vi burde gitt mer info på forkant om hvordan nettsiden fungerer.
- Navigerer seg til slutt inn på feed.
    - Enten up eller down vote.
- Trykker pil opp

4. **Lage ny gruppe.**
    - Lage gruppen.

- Trykker plussknapp på grupper.
- Klarer lett å lage gruppen, syntes det var oversiktlig og lett å få til.

5. **Legge til ny krypto valuta på profilen.**
    - Gå tilbake til feed.
    - Naviger til profilside.

- Navigerer seg rett til profilside.
    - Se hva du har investert i.
- Navigerer til portefølje.
    - Legge til 100 ripple (XRP) til profilen.
- Trykker add.
- Legger til det som skal.
- Kommenterer at det ikke finnes en måte å lukke porteføljesiden som kommer opp, burde kanskje være en x i hjørnet slik at man ikke trenger å navigere seg til en annen side for å få vekk porteføljen.

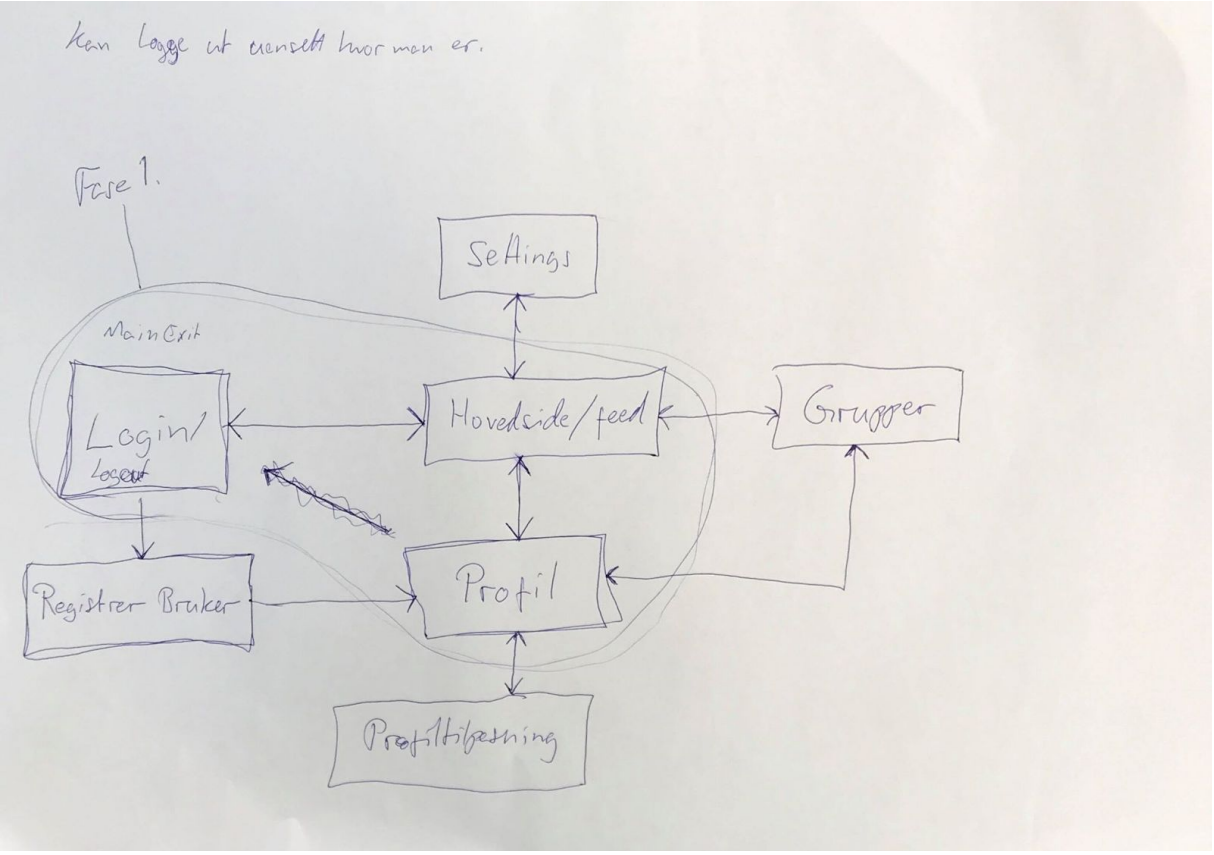
Test runde 2 ga oss god tilbakemelding og bekreftelse på at endringene vi gjorde etter runde 1 var gode. Det er ikke lenger et problem å finne ut hvordan man legger til innlegg. Det eneste vi har endret

på etter runde to med tester er at vi har lagt til en liten X oppe i høyre hjørnet på porteføljе-siden slik at man lett kan lukke den uten å måtte navigere seg vekk fra profilsiden.

Vi er nå sikre nok på designet til å gå videre til å lage en digital versjon i Adobe XD.

9.6 Sitemap

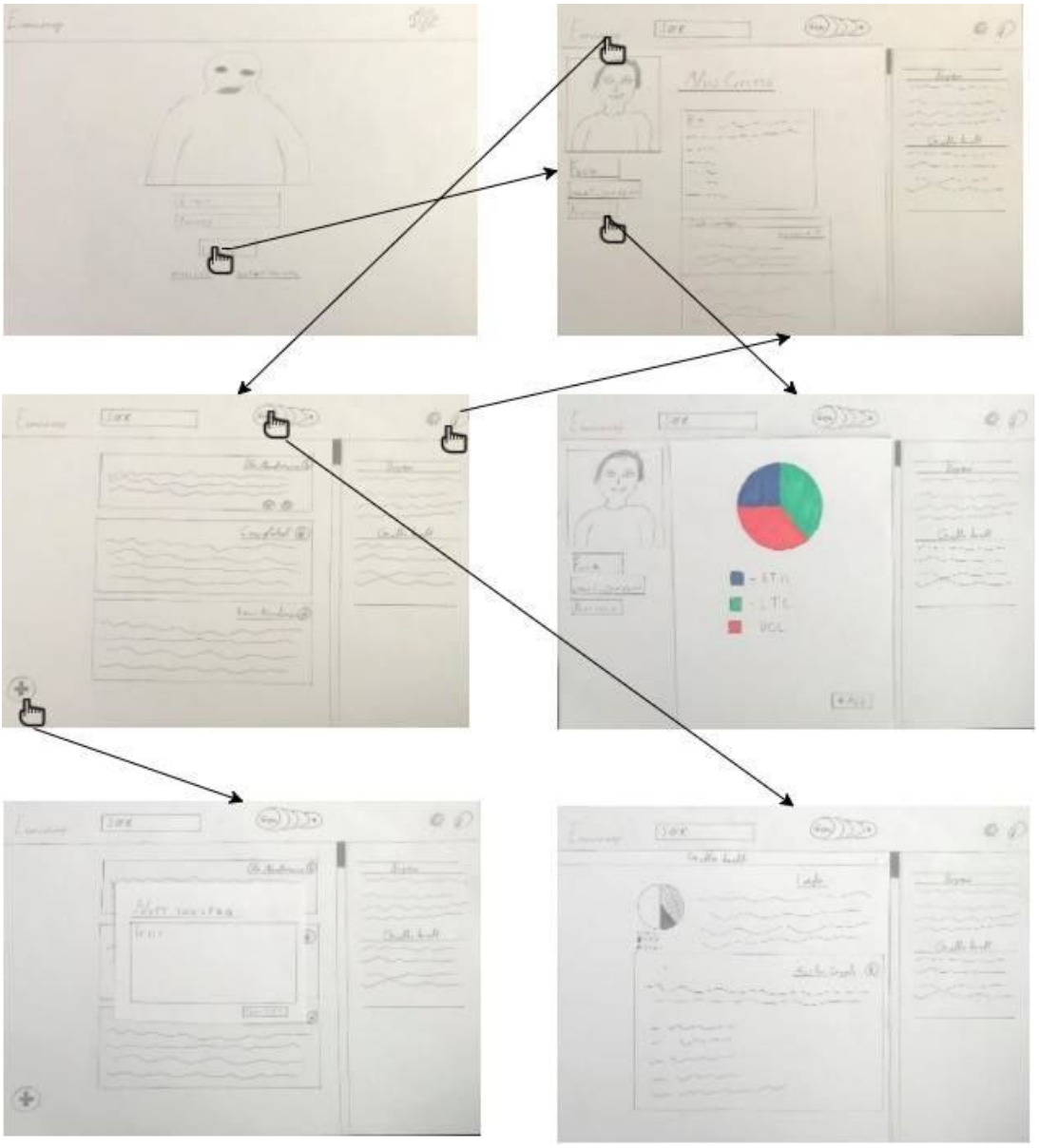
For å lage en oversikt over alle sidene som skal være med i systemet har vi laget et sitemap. Her har vi også linket relasjoner mellom sidene. Etter som prosjektet vårt er delt opp i to faser, og bachelor prosjektet tar for seg fase 1, har vi sirklet ut sidene som hører til denne fasen av prosjektet.



Figur 12, Sitemap

9.7 Wireframe

Wireframe gir også en oversiktlig visualisering av systemet. I forhold til sitemap gir wireframe også et screenshot av siden, det linker relasjoner mellom sites og det viser hvilken knapp som oppretter linkene. Vår wireframe er utviklet ut fra papirprototypen, som var vårt første utkast av design.

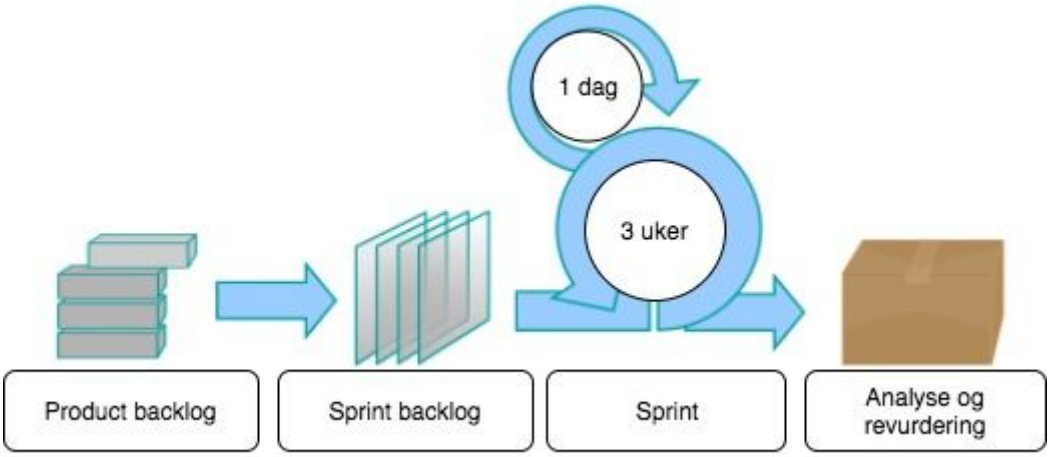


Figur 13, Wireframe

10.0 Prosjektgjennomførelse

Vi har som nevnt valgt å gjennomføre prosjektet i sprinter der vi utvikler, tester, analyserer og revurderer for hver eneste sprint. Vi har totalt gjennomført 6 sprinter når denne rapporten skrives.
I alle sprintene har vi tatt med hvor mye tid vi har brukt på hver oppgave i sprint-backlogen, men dette er ikke alle timene vi har brukt den sprinten. I tillegg til timene som står i backlogen har det blitt brukt mye tid på diverse gruppemøter (daily scrum, sprint planning, sprint review, sprint retrospektiv, styringsgruppemøte) og forelesninger. Vi estimerer at det er brukt i snitt 30 timer på dette i hver sprint fra og med sprint 4. I de tre første sprintene vil dette tallet være en del lavere da de bare var en uke lange.

Sprintoversikt:



Figur 14, Sprintoversikt

10.1 Pre-sprint

Vår pre-sprint startet ved semesterstart tidlig i januar. Da hadde vi allerede brukt juleferien på å utforske programmeringsspråk og andre verktøy som skulle brukes til utvikling av systemet. Ved starten av semesteret gikk vi først i gang med kompetansebygging innen programmeringsspråkene vi hadde valgt. Dette var en omfattende prosess, da et nytt programmeringsspråk ikke læres over natten. Det ble en hard start på semesteret der samtlige gruppemedlemmer brukte langt flere timer enn estimert arbeidsmengde. Vi hadde også dannet et bilde av at vi måtte kunne språkene godt før vi begynte med utvikling. Vi ser i ettertid at vi kunne hatt en langt mer "learning by doing" tilnærming, og startet tidligere med utvikling av systemet. Pre-sprinten inneholdt også en del planlegging, hvor vi laget en overordnet prosjektplan. Deretter laget vi rikt bilde, førsteutkast av brukerhistorier, klassediagram og definerte metodikken vi skulle bruke videre i prosjektet. Dette var en utfordrende prosess, ettersom vår kunnskap og erfaring innen dette feltet hadde sine begrensninger.

Pre-sprinten ble litt lenger enn planlagt fordi det var utfordrende å bli enige om metodikken. Etter mye diskusjon og research bestemte vi oss for å følge en hybrid av Lean startup og Scrum.

Etterhvert som vi kom i gang med arbeidet fant vi ut at vi måtte endre litt på kravene til vår MVP, slik at vi kunne komme i mål iløpet av bacheloren. Vår MVP var i utgangspunktet en fungerende plattform, der brukere kunne lage profiler, opprette en portfolio og dele informasjon med andre brukere. I dette skulle vi også implementere Steemit sin Smart Media Token, for å løse transaksjoner og veksling av valuta. Etter litt tid med kompetansebygging og diskusjon kom vi fram til at dette var litt for ambisiøse krav for en periode på fem måneder. Vi bestemte oss dermed for å dele prosjektet opp i to faser, og ferdigstille fase 1 som bacheloroppgave. I denne fasen valgte vi å ha hovedfokuset på design, og lage frontend til den sosiale delen av plattformen. Dette er også noe vi i har revurdert flere

ganger, da vi hele tiden har tilegnet oss ny kunnskap som har gitt oss mer innsikt om hva som er realistisk å oppnå.

10.2 Sprint 1-3. (12.02 - 02.03)

Sprint 1-3 var korte sprinter vi brukte til å teste vår smidige metodikk. Vi startet med sprinter på 1 uke, og bestemte oss for å prøve det i tre uker for så å vurdere om det var passelig lengde på iterasjonene. Da disse sprintene ble korte, og hadde noe likt innhold, har vi valgt å omtale dem i samme kapittel. Sprint planning ble gjort torsdagene uken før sprinten skulle starte. Vi hadde sprint planning, sprint retrospective, sprint review og daglige scrum møter i hver sprint. Etter et par sprinter begynte vi å se svakheter med å jobbe med slike korte sprinter. Vi brukte veldig mange timer på møter, og satt da igjen med færre arbeidstimer per uke. Det var heller ikke nødvendig å ha disse møtene så ofte, da hele gruppen sitter samlet og arbeider sammen hver dag. Dermed bestemte vi oss for å utvide lengden på sprintene til tre uker, fra og med sprint 4.

I disse sprintene tok vi i bruk styringsverktøyet Jira, noe som førte til litt startvansker. Dette var et nytt verktøy for hele gruppen, og vi måtte ta en "learning by doing" framgang. Jira fungerer som en backlog, og en grafisk framstilling av når oppgavene blir gjort og hvem som har ansvar for dem. Oppgavene i sprint 1-3 besto hovedsakelig av kompetansebygging, og vi arbeidet med design. Etter planen skulle vi være ferdige med design, og starte med koding innen 23.februar. Vi arbeidet godt, og startet med koding 26. Februar. Videre i sprinten jobbet vi parallelt med design og programmering. Designet ble nesten helt ferdig etter planen.

Vi fikk en god start på programmeringen. Erik og Oscar begynte å utvikle HTML sidene til fase 1, da dette var basis for å implementere brukerhistorie 1. Gruppen var veldig fornøyde med resultatet, og dermed fikk Erik og Oscar hovedansvaret for å jobbe videre med utvikling av den statiske siden. Samtidig jobbet Sebastian med å utforske React og React-Router-Dom.

Sprint planning sprint 1:
I denne sprinten fokuserte vi på å forbedre brukerhistoriene og prioritere dem i MoSCoW. Vi ville også lage papirprototype, testscenarier til papirprototypen og på slutten av sprinten teste denne. For å forberede oss på uforutsette risikoer som kunne oppstå ville vi også lage en risikoanalyse.

Sprint review sprint 1:
Vi ble ferdig med alle oppgavene denne sprinten, og er godt forberedt til neste sprint.

- MoSCoW: Vi MoSCoW prioriterte alle brukerhistoriene.
- Sitemap: Vi laget en oversikt over alle sidene i systemet vårt. Dette gjorde det lettere å utforme en papirprototype.
- Papirprototype: Vi laget en papirprototype vi ble veldig fornøyd med. Dette ble første utkast av designet vårt. Vi testet denne på en medstudent på IT studiet, og fikk mange nyttige tilbakemeldinger på utseende og funksjon.
- Risikoanalyse: Utarbeidet en god risikoanalyse etter mal fra foreleser i IS-304.

Sprint retrospective sprint 1:

Det har vært en grei sprint, tatt i betraktning at dette var vår første sprint synes vi den har gått bra. Vi samarbeider godt. Ola jobber med å innta rollen som scrum master. Vi ser at det kan hjelpe til å få enda bedre struktur i scrum møtene. Vi kan bli bedre på å holde de daglige scrum møtene korte og saklige Vi ser også at det går mye tid til møter på en uke.

Sprint planning sprint 2:
I denne sprinten ville vi fortsette med å utvikle designet. Vi fikk gode resultater fra papirprototype testen i sprint 1, og ville bruke disse resultatene til å oppdatere designet og utføre en ny runde med test av papirprototype. Vi brukte også designet fra papirprototypen til å lage wireframe.

Sprint review sprint 2:
Oppgavene i denne sprinten var enkle å estimere tid på, da vi har gjort tilsvarende oppgaver tidligere. Det gjorde at vi hadde god kontroll på arbeidet denne uken, og ble ferdig med alle oppgavene vi hadde hentet inn i sprint backloggen.

- Papirprototype: Vi tok resultatene fra sprint 1, forbedret papirprototypen og utførte ny test.
- Wireframe: Vi brukte sitemap og papirprototype til å lage wireframe.
- Rikt bilde: Vi forbedret vårt rike bilde. Oppsettet i den første utkasten var litt uforståelig, og det var ikke helt klart hva verdien av produktet vårt var.

Sprint retrospective sprint 2:
Vi synes denne sprinten gikk bra. Oppgavene i denne sprinten var enkle å estimere tid på, og det gjorde at vi hadde god kontroll hele sprinten. Vi synes fortsatt sprinten blir kort, og at mye tid da går til scrum møter. Vi vurderer om vi skal utvide lengden på sprinten.

Sprint planning sprint 3:
Målet for denne sprinten var å videreutvikle designet, og klargjøre for å starte utvikling 23.februar. Før vi kunne begynne å programmere måtte vi sette opp git/github, så alle kunne arbeide på samme repository. Designet skulle nå digitaliseres med Adobe Xd, etter resultatet fra papirprototypene i sprint 1 og 2.

Sprint review sprint 3:
I denne sprinten fikk vi ikke ferdigstilt alle oppgavene, da utviklingsoppgaven ble begynt på rett før sprint slutt.

- Design/Mockups: Vi laget et designutkast etter resultatene fra papirprototypen.
- Git/Github: Vi opprettet en repository på github og installerte git på alle maskinene.
- Utvikling: Denne videreføres til sprint 4.

Sprint retrospective sprint 3:
Vi er fornøyd med resultatet av denne sprinten, fordi vi fikk en veldig godt start på programmeringen. Grunnen til at utviklingsoppgaven ikke ble ferdigstilt var at den ble startet rett før sprinten var ferdig. Dette var forventet. Vi har i løpet av de tre første sprintene kommet frem til at det er for korte iterasjoner, og vi vil fra neste sprint utvide til tre ukers sprinter. Da vil vi få mer arbeidstid kontra møtetid. Vi sitter også samlet hver dag, vi ser derfor på det som unødvendig med så mange møter.

Sprint 1-3 backlog

Oppgaver	**Tid estimert**	**Tid brukt**	**Ansvarlig**	**Status**
Sprint 1				
MoSCoW	10t	12t	Alle	Ferdig
Lage papirprototype	9t	8t	Ola, Erik	Ferdig
sitemap	3t	5t	Oscar, Erik	Ferdig
Risikoanalyse	5t	6,5t	Alle	Ferdig
Lage testscenarier til papirprototype	2t	2t	Ola	Ferdig
Papirprototype test runde 1.	8t	12t	Ola, Erik	Ferdig
Sprint 2				
Forbedre rikt bilde	2t	2,5t	Ola	Ferdig
Forbedre papirprototype.	4t	5t	Ola, Erik	Ferdig
Papirprototype test runde 2	10t	8,5t	Ola, Erik	Ferdig
Wireframing	14t	10t	Sebastian, Ola	Ferdig
Sprint 3				
Design/mockups	50t	57t	Alle	Ferdig
Sette opp git/github	24t	26t	Alle	Ferdig
Programmere profilkort (HTML,CSS)	103t	25t	Oscar, Erik	Påbegynt

Tabell 7, backlog sprint 1-3

10.3 Sprint 4 (05.03.18 - 23.03.18)

Sprint planning:

I denne sprinten ville vi ha fokus på programmeringen. I forrige sprint ble vi ferdig med alle oppgaver utenom profilsiden, derfor ville vi fortsette med det i denne sprinten. Deretter vil vi begynne å programmere feed og login side. Vi ville også gå gjennom brukerhistorier og MoSCoW prioriteringen for å se om vi kunne gjøre forbedringer eller endringer ettersom vi har tilegnet oss ny kunnskap som kan sette ting i nytt lys.

Sprint backlogen for sprint 4 så slik ut:

Oppgaver	Tid estimert	Tid brukt	Ansvarlig	Fullført.
Sprint 4				
Programmere profilkort (HTML,CSS)	103t	40t	Oscar, Erik	Påbegynt
Programmere feed (HTML,CSS)	117t	31t	Oscar, Erik	Påbegynt
HTML login	108t	78t	Oscar, Erik	Påbegynt
Forbedring/utvidelse av brukerhistorier.	9t	10t	Erik, Ola, Oscar	Ferdig
Forbedre/utvidelse av MoSCoW	3t	6t	Alle	Ferdig

Tabell 8, backlog sprint 4

Denne sprinten ble kortere enn vi ønsket på grunn av at samtlige gruppemedlemmer hadde eksamen i et annet fag i denne tidsperioden. Vi ble enige at vi skulle få lov til å fokusere på denne eksamen noen dager i forveien, dette gjorde at vi mistet noen dager med arbeid før påsken. Vi mistet også litt arbeidstid i påsken, da samtlige gruppemedlemmer var hjemme hos sin familie. Selv om gruppen var splittet geografisk klarte vi å holde prosjektet i gang, og arbeidet hver for oss. Vi gjennomførte da statusmøtene over videokonferanse.

Vi fortsatte i denne sprinten å programmere HTML sidene og implementerte designet med CSS etter designskissene vi hadde laget i Adobe XD. I denne sprinten laget vi også et HTML login form. Vi var veldig fornøyd med implementeringen av designet, da dette ble så godt som nøyaktig slik vi skisset det i Adobe XD. Vi jobbet også jevnt med kompetansebygging i React, som skal brukes i sluttproduktet.

Sprint review:

- Programmer profil : oppsettet er ferdig, vi deler den videre opp i mindre programmeringsoppgaver til de neste sprintene.
- Programmer feed: oppsettet er ferdig, vi deler den videre opp i mindre programmeringsoppgaver til de neste sprintene.
- Forbedring av brukerhistorier: vi gjorde noen små endringer på eksisterende brukerhistorier og la til noen nye.
- Forbedring av MoSCoW: MoSCoW prioriteringen har ikke endret seg siden første gang vi gjorde den, men vi har prioritert de nye brukerhistoriene som kom inn.

Vi fikk gjort oss greit ferdig med alle oppgavene, utenom programmeringen som vi videre delte opp i mindre oppgaver som “lag innleggsfunksjon” osv… Vi har innsett at å bare ha “programmer profil” blir en altfor omfattende oppgave.

Sprint retrospektiv:

Dette er første sprinten etter vi bestemte oss for å utvide lengden på sprintene, og vi er enig om at det har fungert veldig bra. Vi merker at vi har spart tid på å ikke ha så mange møter, og vi synes det er bedre å kunne planlegge for lenger enn bare en uke.

Det har ikke vært noen store problemer under denne sprinten. Vi måtte bruke en del tid på lesing til eksamen som ellers kunne gått til jobbing med bachelor, men dette hadde vi på forhånd tatt med i planleggingen, så det ble ikke noe stort problem.
Vi vil fortsette med samme lengde på sprintene ettersom vi synes dette fungerte veldig bra i denne.

10.4 Sprint 5 (03.04.18 - 21.04.18)

Sprint planning:
Etter å ha vurdert resultatene fra forrige sprint bestemte vi oss for å fortsette programmeringen av frontend; gjøre ferdig profilkort på profilside og begynne å programmere et element for innlegg. parallelt med dette ville vi begynne på backend med oppsett av database og server.
Sprint backlogen for sprint 5 så slik ut:

Oppgaver	**Tid estimert**	**Tid brukt**	**Ansvarlig**	**Status**
Programmere innleggsfunksjon	72t	91t	Erik	Påbegynt
Ferdigstille profilkort	30t	27t	Oscar	Ferdig
GraphQL server	94t	88t	Sebastian	Ikke fullført
Prisma	72t	62t	Sebastian	Ikke fullført

Tabell 9, backlog sprint 5

I denne sprinten merkes det at gruppens medlemmer har begynt å bli komfortabel med utviklingsmetodikken, og det er god flyt i arbeidetsdagene. Vi lå også i denne sprinten godt ann i forhold til prosjektplanen, og frontend-delen til den den sosiale plattformen begynte å bli ferdig. Den neste delen i utviklingen ble som forventet den mest utfordrende. Her prøvde vi å finne ut hvordan vi kunne bruke GraphQL til å sette opp webserver, og koble opp en database ved bruk av Prisma DB. Grunnet økning i vanskelighetsgrad bestemte vi oss for å prøve peer programming. Hele gruppen var aktiv i denne prosessen og vi begynte å danne oss et godt bilde av hvordan vi skulle få dette til.

sprint review:

- Programmere innlegg funksjon: er påbegynt, møtte på en feil som vi enda ikke har klart å fikse.
- ferdigstille profile card: ble fullført uten at vi støtte på noen store problemer, resulatet er tilfredsstillende.
- GraphQl: påbegynt, men var ikke tilstrekkelig med tid til å kunne gjøres ferdig i løpet av en sprint. Arbeidet flyttes videre til en senere sprint.
- prisma DB: påbegynt, men var ikke tilstrekkelig med tid til å kunne gjøres ferdig i løpet av en sprint. Arbeidet flyttes videre til en senere sprint.

Sprint retrospektiv:

I denne sprinten hadde vi alt fokuset på programmering. Vi har merket at det er veldig vanskelig å gjøre gode estimeringer av programmeringsoppgaver vi ikke har noen tidligere erfaring med. Dette er fordi det fort kan oppstå ukjente feil eller at vi får feilmeldinger vi ikke har vært borti før. I retrospekt ser vi at vi burde estimert mer tid på oppgaver vi ikke har noen erfaring med slik at vi ikke blir altfor langt bak skjema dersom det skulle oppstå en feil.
Vi kom godt i gang med koding av backend med GraphQL og PrismaDB men oppgavene ble for store til at vi fikk fullført de.
Vi har brukt mye peer programming denne sprinten og det synes vi har fungert veldig bra da det gjør at alle får innsikt i hvordan koden er satt opp og at alle kan være med å hjelpe til med kodingen, dette har vært spesielt nyttig når det har oppstått feil i koden.

10.5 Sprint 6 (23.04.18 - 16.05.18)

I denne sprinten har fokuset blitt rettet mot rapporten. Vi måtte samle dokumentasjonen vi har laget gjennom semesteret, og sette det sammen til en helhetlig rapport. Fullstendig liste over oppgaver til rapportskriving ligger i vedlegg.

11.0 Refleksjon

11.1 Prosjektgjennomføring

Selve prosjektgjennomføringen har vært en omfattende prosess vi har lært mye av. Vi startet med å lage en overordnet prosjektplan med mål for fremgang, men vi oppdaget tidlig at vi hadde satt litt vel ambisiøse mål for prosjektet og fremgangen. Dette justerte vi flere ganger slik at vi til slutt satt igjen med realistiske mål som fortsatt var ambisiøse og utfordrende, men som vi som gruppe mente vi kunne klare å nå. Her har vi lært mye om hva som er realistisk å oppnå innen gitte tidsrammer, og hvor viktig det er å gjøre tilstrekkelig research på forhånd for å kunne planlegge og sette realistiske mål.

I pre-sprinten hadde vi en lang periode med kompetansebygging før vi begynte med utvikling. Dette er fordi vi hadde dannet et bilde av at vi måtte kunne programmeringsspråkene godt før vi begynte med utviklingen av vårt system. I ettertid ser vi at vi kunne med fordel ha gått for en mer "learning by doing" fremgangsmåte, slik at vi tidligere kunne gått i gang med utviklingen av prosjektet.

Vi har gjennom prosjektet arbeidet etter en smidig prosess med en inkrementell planlegging. Hensikten med dette er at det skal være lettere å kunne tilpasse oss miljøet og markedet. Utviklingsprosjekter vil som oftest i praksis være en kombinasjon av plandrevet og smidige prosesser. En plandrevet prosess er når alle aktivitetene er planlagt på forhånd og fremdrift blir målt mot denne planen. Vi har gjennom prosjektet satt overordnede mål vi har målt fremdrift etter. Det blir derfor feil å si at vi kun har utnyttet en smidig prosess.

Under vår inkrementelle utviklingsprosess har vi prøvd å sammenflette spesifikasjon, utvikling og validering. Dette er både plandrevet og smidig. Fordeler med en inkrementell utvikling er at det reduserer kostnader ved endring av krav. Det er også enklere å få tilbakemelding på utvikling som er gjort underveis. Men det viktigste er å få på plass fungerende deler av systemet som kunden kan bruke.

En ulempe er at prosessen ofte ikke er like synlig. Ved større prosjekter kan strukturen i systemet lettere forringes for hver gang et inkrement blir lagt til.

11.2 Lean og Scrum

Lean og Scrum ble ikke kombinert helt slik vi hadde sett for oss. Tanken om en hybrid mellom disse var god, men ut fra rammene i dette prosjektet ble det mer naturlig å jobbe med ren Scrum. Lean gir mer overordnede retningslinjer, som ofte gjelder for større organisasjoner, i forhold til Scrum som påvirker mindre team. Fordelene vi hadde planlagt å utnytte fra Lean var best practises som ga feedback fra sluttbruker fortløpende. I ettertid ser vi at i fase 1 av utviklingen vår kommer vi ikke langt nok til å utnytte disse godene fra Lean. Lean elementene vil da egentlig ikke bli særlig involvert i prosjektet før fase 1 er ferdigstilt. Det vil si at dette semesteret har vi i realiteten bare fulgt en iterativ utviklingsprosess med scrum. Vår metodologi ble da ikke helt som vi i utgangspunktet hadde sett for oss, men vi er fornøyd med hvordan vi faktisk utførte prosjektet.

11.3 Kommunikasjon

I dette prosjektet har vi hele tiden hatt god kommunikasjon mellom oss. Vi har startet hver arbeidsdag med et kjapt statusmøte/daily scrum, hvor vi har fortalt hva vi gjorde dagen før, hva vi skal gjøre i dag og tatt opp eventuelle problemer. Ingen har vært nølende når det kommer til å uttrykke sin egen mening og vi har gjort vårt beste for at alle meninger har blitt hørt og diskutert.

I rapporten nevnte vi at vi brukte Slack som et kommunikasjonsverktøy, men vi har ikke brukt det like aktivt som vi hadde tenkt. Internt i gruppen brukte vi facebook messenger som vår hovedplattform for kommunikasjon, Slack ble brukt når det var noe som var interesse for både oss og veileder. Kommunikasjonen med veileder som avtale om møter og spørsmål ble gjort over mail, noe som også er veldig normalt for oss og bruke. I ettertid ser vi at vi kunne kuttet ut slack og bare brukt Messenger, skype og mail.

11.4 Styringsverktøy

Vi har gjennom dette prosjektet prøvd flere styringsverktøy, til sammen har vi testet fire forskjellige. Vi har brukt mye tid på å teste ut de forskjellige verktøyene som ellers kunne vært brukt på andre oppgaver. I ettertid ser vi at vi burde ha tatt en beslutning om verktøy tidligere og holdt oss til dette.Hele gruppen mener at hvis vi skulle gjøre dette prosjektet på nytt ville vi ha valgt å bruke Google sheet som både produkt og sprint-backlog, siden dette er det som har gitt mest verdi.

11.5 Tidsbruk

Vi har lært mye om planlegging og estimering av tid. Det er veldig mange ting som kan gå galt under et utviklingsprosjekt som hindrer fremgang eller fører til at tidsfrister ikke blir overholdt. For å sikre at prosjektet ble gjennomført på en best mulig måte, begynte vi med å etablere en tidsplan med dato for ferdigstillelse av prosjektet og viktige milepæler underveis. Dette ble gjort for at fremdriften i prosjektarbeidet ble målbar. Vi har gjennom hver Sprint planlagt på et overordnet nivå for å få et overblikk over oppgavene som skal fullføres. Deretter har vi gått nærmere inn på detaljene i hver

enkelt oppgave og estimert hvor mye tid det vil ta å fullføre oppgaven. Ettersom vi har lite erfaring og kunnskap fra lignende prosjekter har det tidvis vært vanskelig å gi et realistisk estimat. Vår løsning på dette har vært å bruke *Planning Poker,* som er en estimering og planleggingsteknikk. Dette har vært en effektiv måte å tilrettelegge for koordinering og for å skape en felles oppfattelse av oppgavenes størrelsesorden, som vi synes har fungert veldig bra.

Hele prosessen med å planlegge og estimere tidsbruk har vært krevende, men gitt oss mye verdifull erfaring som vi kan ta med inn i arbeidet videre. Hvis vi skulle startet det samme prosjektet i dag med tilsvarende kriterier ville arbeidsflyten vært bedre og ville ikke brukt like mye tid på å løse problemer.

11.6 Håndtering av utfordringer

Det har vært svært utfordrende å utvikle et system i et programmeringsspråk vi ikke har tidligere erfaring med. Spesielt i språk som JavaScript, som ikke er strict. Små feil i syntax, serverkonfigurasjon og andre generelle bugs har plutselig stoppet opp deler av utviklingen i timesvis. Vi er heldige som har fått arbeide i samme rom på UIA Nyskapning under hele bacheloren, og når bugs oppsto løste vi de fortløpende i felleskap ved bruk av peer programming. I de tilfellene arbeidet har stoppet opp på grunn av uenigheter, har vi funnet frem vår konfliktløsningsstrategi som nevnt tidligere i rapporten. Konfliktløsningsstrategien vår har fungert godt, vi tror at den har vært med på å holde vennskapet i gruppen vedlike, og en god flyt i arbeidet.

Å være et i et team hvor alle i utgangspunktet er venner, og kjenner hverandre fra før, kan være en fordel og en ulempe. Fordelene er at vi vet veldig godt hva hverandres kvaliteter er, og hvordan vi skal samhandle med hverandre. Det ligger også en ansvarsfølelse i bunnen, der alle i gruppen ønsker hverandre godt, og en god karakter på prosjektet. Ulempene kan derimot komme når det oppstår uenigheter og misnøye. Det er også lettere for at det oppstår avsporinger under arbeidstiden, og at "jobben" blir tatt med opp i sosiale settinger. Dette er noe vi reflekterte over og formet en enighet rundt allerede da vi dannet gruppen. Vi ble enige om å holde et profesjonelt forhold til prosjektet, og prøve å spille på de positive sidene ved at vi kjenner hverandre godt fra før.

11.7 Samarbeid med klient

Dette prosjektet skiller seg fra de fleste andre bachelor prosjektene da vi opererer som vår egen arbeidsgiver. For å sørge for at arbeidsoppgaver blir utført, og at gruppens medlemmer jobbet mot det overordnede målet, valgte vi å rullere på rollen som produkteier. Alle gruppens medlemmer har hatt hver sin periode som prosjektleder. Vi så i utgangspunktet på å være vår egen arbeidsgiver som en stor utfordring, da det krever veldig mye selvdisiplin, men det åpnet også for store muligheter. Vi fikk muligheten til å arbeide med noe vi har en genuin interesse for. Etter som engasjementet for prosjektet er stort hos alle på gruppen har det ikke vært noe problem å holde motivasjonen oppe. Det hadde til tider vært fint å hatt en arbeidsgiver vi kunne støttet oss på når ting stopper litt opp. Men vi synes totalt sett at vi har klart å stå på egne ben, og vi er veldig glad for at vi har fått innblikk i hvordan det er å drive et systemutviklings-team.

11.8 Samarbeid med veileder

Vår veileder i dette prosjektet har vært Janis Gailis. Janis har oppholdt seg i utlandet under store deler av semesteret, men dette har overhodet ikke vært noen utfordring da han har vært tilgjengelig på mail og videokonferanse til enhver tid. Janis har vært til god hjelp, og stilt kritiske spørsmål under hele prosjektet som har gjort at vi har opprettholdt høye krav til oss selv og produktet. Det eneste vi angrer

på angående vår veileder er at vi ikke tok han i bruk mer i starten av semesteret. Hadde vi gjort det kunne vi mest sannsynlig fått en lettere oppstart.

12.0 Videre utvikling

Vi har tidligere i rapporten nevnt at vi kun har arbeidet med fase 1, og det gjenstår mye før vi lanserer en alpha versjon. Vi har kommet langt på front-end og jobbet mye med kompetansebygging og verktøyvalg for den neste fasen av prosjektet. I dette avsnittet vil vi beskrive verktøy og teknologier vi skal bruke for å ferdigstille det fullverdige produktet. Selv om dette er en bachelorgruppe, er alle medlemmene enige om å fortsette videre utvikling av produktet fremover. Vi håper å kunne lansere produktet vårt i løpet av 2019.

13.0 Konklusjon

Målet for dette prosjektet var å få på plass fase 1, den sosiale delen av Ecoinomy. Dette har vi etter beste evne prøvd å få til med tiden og ressursene vi har hatt til rådighet. Vi har med utgangspunkt i analysen vi gjennomførte i starten av prosjektet gjort flere sentrale beslutnigner. Vi har blant annet valgt en hybrid mellom Lean startup og Scrum som metode. Det har også vært en del "trade offs" for å holde prosjektet innenfor rammene vi opererer i, samt for å kunne skape noe av verdi. Disse "trade offsene" har mest dreid seg om å redusere omfanget av prosjektet slik at vi kan få gjort de viktigste oppgavene tilfredsstillende innen tidsfristen.

Prosjektgjennomføringen har foregått i sprinter på tre uker (utenom sprint 1-3 som var på en uke per). I disse sprintene har vi gjennomført utvikling, test, analyse og revurdering for hver iterasjon. Slik har vi kontinuerlig revurdert og gjort nødvendige endringer på produktet gjennom hele prosjektet. Vi har ikke klart å bli ferdig med alt vi planla, blant annet på grunn av uforutsette feil og kompleksiteten i det vi lager. Men vi har fått ferdig en statisk Frontend og kommet i gang med utviklingen av backend, som vi også vil fortsette med senere.

Vi har gjennom dette prosjektet prøvd og feilet mye, dette gjelder fra verktøy vi har brukt til prosjektstyring, lengdene på sprintene, omfanget i rapporten mm. Men hver gang vi har feilet har vi blitt litt klokere og gått tilbake å sett hva som var galt for å så gjøre forbedringer. Et av sitatene vi har brukt som inspirasjon er et Ola lærte av sin norsklærer; "fail, fail again, fail better"(sitatet stammer fra navnet på en bok, original sitatet er: Try again, fail again, fail better.). I dette sitatet legger vi at det er ikke noe gale i å feile så lenge man forstår hvorfor og lærer av det. Alt i alt har vi lært utrolig mye av å gjennomføre denne bacheloren og vi er mye bedre rustet til å gjennomføre et slikt prosjekt igjen om muligheten skulle vise seg.

14.0 Kilder

Airbnb. (u.d.). *GitHub.* Hentet fra airbnb/javascript: https://github.com/airbnb/javascript/tree/master/react

Apollo. (u.d.). *Apollo*. Hentet fra Query Batching | Apollo Engine: https://www.apollographql.com/docs/engine/query-batching.html

Apollo. (u.d.). *Apollo.* Hentet fra Apollo GraphQL: https://www.apollographql.com

Apollo-server. (u.d.). *GitHub*. Hentet fra apollographql/apollo-server: https://github.com/apollographql/apollo-server

Blank, S. (2018, Februar 9). *https://hbr.org/2013/05/why-the-lean-start-up-changes-everything*. Hentet fra Why the Lean Start-Up Changes Everything: https://hbr.org/2013/05/why-the-lean-start-up-changes-everything

Byron, L. (2018, Januar). *npm*. Hentet fra dataloader: https://www.npmjs.com/package/dataloader

Club, B. V. (2016, Februar 9). *YouTube*. Hentet fra "The Lean Startup" by Eric Ries - BOOK SUMMARY: https://www.youtube.com/watch?v=sobxOzRjAGg

conferences, G. (2013, April 3). *YouTube*. Hentet fra GOTO 2012 • Lean Startup: Why it Rocks far more than Agile Development • Joshua Kerievsky: https://www.youtube.com/watch?v=V5p8m1IjJoA

Entrepreneurs, G. f. (2014, Juni 17). *YouTube*. Hentet fra Lean Startup Meets Design Thinking: https://www.youtube.com/watch?v=bvFnHzU4_W8

felixge. (2017, januar 22). *GitHub.* Hentet fra felixge/node-style-guide: https://github.com/felixge/node-style-guide

graphql-yoga. (u.d.). *GitHub*. Hentet fra prismagraphql/graphql-yoga: https://github.com/graphcool/graphql-yoga

Haverbeke, M. (2015). *Eloquent JavaScript: a modern introduction to programming.* No Starch Press.

HowGraphQL. (u.d.). *GraphQL Core Concepts Tutorial*. Hentet fra How to GraphQL - The Fullstack Tutorial for GraphQL: https://www.howtographql.com

https://www.rga.com. (2014, Mai 1). Hentet fra R/GA Transformation at speed: https://www.rga.com/futurevision/pov/can-qa-defy-the-law-of-the-iron-triangle

Høiseth, Y. (u.d.). *eZ Publish Platform, CXM & CMS*. Hentet fra Smidig vs. fossefall: fordeler og ulemper: https://ez.no/no/Blogg/Smidig-vs.-fossefall-fordeler-og-ulemper

International Scrum Institute. (u.d.). *What is a Sprint? - International Scrum Institute*. Hentet fra Scrum Burndown Chart: https://www.scrum-institute.org/Burndown_Chart.php

Jenkins. (u.d.). *Blue Ocean*. Hentet fra Jenkins: https://jenkins.io/

Johnson, J. (2008, Juli 21). *LinkedIn SlideShare*. Hentet fra Psych 101: The Psychological Basis for UI Design Rules: https://www.slideshare.net/guest45d695/jeff-johnson-psych-101-the-psychological-basis-for-ui-design-rules-522929/19

Lean Enterprice Institute. (u.d.). *Lean.org*. Hentet fra What is Lean?: https://www.lean.org/WhatsLean/

Microsoft. (u.d.). *Project*. Hentet fra Prosjekttrekanten: https://support.office.com/nb-no/article/prosjekttrekanten-8c892e06-d761-4d40-8e1f-17b33fdcf810

Mountain Goat Software. (u.d.). *Mountain Goat Software*. Hentet fra Release Burndown Chart: https://www.mountaingoatsoftware.com/agile/scrum/scrum-tools/release-burndown

Muller, K. (2015, November 20). *Medium*. Hentet fra Om smidig vs fossefall – Kjartan Rekdal Müller – Medium: https://medium.com/@kjartanmuller/historikken-bak-smidig-vs-fossefall-796a6eea1a9d

Nordbø, T. (2017). *Introduksjon til interaksjonsdesign.* Universitetsforl.

Norman, D. (2004). *Emotional Design: Why We Love (or Hate) Everyday Things* . Basis Books.

O'Reilly. (2017). *Designing data-intensive applications: the big ideas behind reliable, scalable, and maintainable systems* . Oreilly & Associates Inc.

Patel, N. (2014, Juni 24). *Entrepreneur.* Hentet fra The Psychology of Instant Gratification and How It Will Revolutionize Your Marketing Approach: https://www.entrepreneur.com/article/235088

Prettier. (u.d.). *Prettier Blog ATOM.* Hentet fra Prettier · Opinionated Code Formatter: https://prettier.io/

Prisma. (u.d.). *GitHub*. Hentet fra prismagraphql/prisma: https://github.com/graphcool/prisma#architecture

Rekhi, S. (2017, Januar 23). *Medium*. Hentet fra Don Norman's Principles of Interaction Design – Sachin Rekhi – Medium: https://medium.com/@sachinrekhi/don-normans-principles-of-interaction-design-51025a2c0f33

Rossen, E. (2009, Februar 14). *Store norske leksikon*. Hentet fra Store norske leksikon: https://snl.no/CSS

Shalloway, A. (2009, September 28). *Net Objectives.* Hentet fra Comparing Scrum to Lean - Not Quite Apples to Oranges: http://www.netobjectives.com/blogs/comparing-scrum-lean-not-quite-apples-oranges

StoryboardItsBetter. (2013, Oktober 22). *YouTube*. Hentet fra Storyboard of The Lean Startup Introduction: https://www.youtube.com/watch?v=jBlrLqsjIDw

Talks at Google. (2018, April 7). *YouTube*. Hentet fra Eric Ries: "The Lean Startup" | Talks at Google: https://www.youtube.com/watch?v=fEvKo90qBns

The Lean Startup. (u.d.). *The Lean Startup | Methodology*. Hentet fra Methodology: http://theleanstartup.com/principles

Wang, A. I. (u.d.). *ntnu.no.* Hentet fra Systemutvikling: https://www.ntnu.no/wiki/download/attachments/84874858/SoftwareEngineering.pdf?version=1&modificationDate=1479208388545&api=v2

Wikipedia. (2018, Mai 8). Hentet fra Project management triangle: https://en.wikipedia.org/wiki/Project_management_triangle

Wikipedia. (2018, Mai 12). Hentet fra Wikipedia: https://en.wikipedia.org/wiki/Waterfall_model

Wikipedia. (2018, Mai 12). Hentet fra Proxy server: https://en.wikipedia.org/wiki/Proxy_server

Wikipedia. (2018, Februar 17). *Wikipedia*. Hentet fra Burn down chart: https://en.wikipedia.org/wiki/Burn_down_chart

Wikipedia. (2018, Mai 12). *Wikipedia*. Hentet fra Cache (computing): https://en.wikipedia.org/wiki/Cache_(computing)

Øritsland, T. A. (2005, September 20). *ntnu.* Hentet fra http://www.idi.ntnu.no/emner/it3402/F7GestaltLayout.pdf

15.0 Vedlegg

Gruppeevaluering

Samarbeidet i gruppen har vært veldig bra, og når det har oppstått uenigheter har vi anvendt konfliktløsningsstrategien. Vi satt også opp en kommunikasjonsplan som har hjulpet oss med å holde arbeidet profesjonelt og skilt det fra privatlivet. Alt i alt er gruppen veldig fornøyd med hvordan prosjektet har gått og vi gleder oss til å jobbe videre med det etter bacheloren.

Selvevaluering

Oscar Lindberg:

Arbeidet med dette prosjektet kommer fra gruppemedlemmenes genuine interesse for blockchain teknologi, og det har derfor vært lett å engasjere seg for temaet. Vi har tilegnet oss mye ny kunnskap rundt området, og vi har fått mulighet til å sette en drøm og en ide ut i virkelighet. Samarbeidet på gruppen har fungert veldig bra, da vi alle kjenner hverandre godt fra før. Jeg er også veldig stolt av egen innsats, da jeg står for at jeg har arbeidet etter beste evne under hele prosjektet. Jeg har hatt mest arbeid med utvikling av frontend delen, og design, dette har passet meg godt da det er noe jeg ser for meg å arbeide med i fremtiden. Jeg har lært veldig mye av dette semesteret som jeg vil ta med meg ut i arbeidslivet. Det har vært lærerikt å se hvordan flere av kursene gjennom bacheloren blir kombinert i bacheloren. Det har gitt meg god forståelse av helheten i et utviklingsprosjekt.

Erik Haaland:

Jeg har vært med på stort sett alle delene av prosjektet. Noe av det jeg har tatt spesielt ansvar for er testing av prototype, oppsett av rapport og rapportskriving, og programmering av frontend sammen med Oscar. Vi har alle hatt en tilnærmet like stor rolle i nesten alle aspektene ved prosjektet, så arbeidsfordelingen har vært veldig jevn. Dette har gjort at jeg har blitt utfordret på mange forskjellige områder, samtidig som jeg har fått et godt overblikk over helheten i prosjektet. Jeg har lært utrolig mye av å gjennomføre dette prosjektet og fått erfaringer og kunnskap jeg vil ta med meg videre.

Sebastian Nomme:

Dette har vært et ufattelig spennende og lærerikt prosjekt å arbeide med. Jeg føler at jeg har fått en mye bedre forståelse for aspektene ved systemutvikling og gjennom prosjektet har jeg funnet ut av at jeg kunne tenke meg å arbeide som systemutvikler på fulltid. Jeg har lært enormt mye om ulike web teknologier og hvordan de henger sammen. Det er omfattende emner med mye kompleksitet. Vi har gjennom prosjektet arbeidet sammen om å løse oppgavene og jeg har tatt del i nesten alt som har blitt gjort i prosjektarbeidet. Hvis jeg skal spesifisere vil jeg påstå at min arbeidsinnsats heller mest mot den

tekniske delen av prosjektet samt designarbeid. Helhetlig har det vært en veldig verdifull erfaring som jeg vil kunne dra nytte av i arbeidslivet og videre studier.

Ola Skarpmoen Eriksen:

Bachelor prosjektet har vært et morsomt og lærerikt reise. Jeg tar med meg mye kunnskap som jeg har lært gjennom hele bachelorløpet og dette prosjektet. Min forståelse for systemutvikling har forandre jevnt over hele prosjektet, og har lært noe hele tiden. Personlig har jeg fått en interesse for Scrum og administrering, siden vi hadde noen problemer med dette. I dette prosjektet har jeg vært innom alle de forskjellige delene, med større fokus på design og administrasjon. Hele gruppen har jobber bra sammen, og laget en rapport jeg er stolt av. Erfaringen jeg har fått under dette prosjektet, har gjort at jeg planlegger å ta master og vil jobbe mer med de administrative oppgavene.

Prosjektplan

Start: 15.jan - slutt: 16.mai

Januar:

Oppstart/ Planlegging/Analyse.
Stort fokus på kompetansebygging:

- HTML
- CSS
- Javascript
- Git

Brukerhistorier.
Rike bilder.
Utvikle design; wireframes og mockups.

Februar:

Tidsestimering til backlogg; planning poker. (www.planningpoker.com) Definere små oppgaver.
Ferdigstille design; wireframes og mockups.
16. Februar: Styringsgruppemøte.
Videre kompetansebygging;

- React.JS
- Node.JS
- Mongo
- GraphQL

23. Februar: Begynne utvikling.

Mars:

23. Mars: Være ferdig med den sosiale funksjonen på nettsiden.
Planleggingsmøte: Hvilken vei videre skal vi ta: portefølje app, portefølje eller Steem integrasjon?

Utvikle plattform.
Starte rapportskriving.

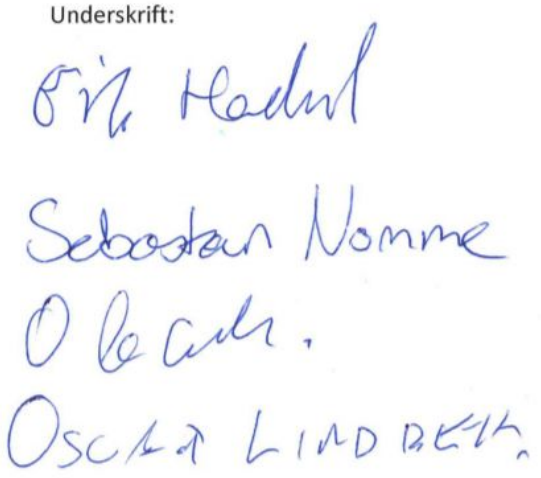
April:
Utvikle plattform.
Rapportskriving.

Mai:
Ferdigstille plattform.
Ferdigstille rapport.

kommentar fra oppdragsgiver:
Vi er som sagt vår egen arbeidsgiver, derfor vil vårt syn om arbeidet også ha kommet frem fra rapporten. Men om vi skulle gitt en liten kommentar til oss selv fra et objektivt synspunkt ville vi sagt dette:
Gruppen har arbeidet strukturert og godt. Prosjektet ble ikke helt ferdig, men det er blitt et godt utgangspunkt for videre utvikling.

Generelt er oppdragsgiver godt fornøyd med prosjektet.

Underskrift:



Rullering av produkteier i prosjektet

Siden vi ikke har en arbeidsgiver gjennom semesteret, og arbeider med vårt eget prosjekt, vil vi rullere på rollen som produkteier. Ved å gjøre det slik vil alle få litt erfaring fra å "lede" et prosjekt, og vi får fylt rollen som produkteier ved diverse møter.

Uke 1-5: Oscar
Uke 6-10: Erik
Uke 11-15: Ola
Uke 16-20: Sebastian

Gruppekontrakt

Gruppe medlemmer:

- Erik Haaland
- Ola Skarpmoen Eriksen
- Oscar Lindberg
- Sebastian Nomme

Mål.

Alle medlemmer i gruppen har ambisjoner for prosjektet og ønsker en karakter i det øvre-sjiktet. Det er forventet at alle bidrar etter beste evne og kommer forberedt til gruppemøte og forelesninger. Alle må bidra for at vi skal nå målet og vi tolererer ikke at noen loffer. Gruppen forventer å jobbe den tiden det vil ta og oppnå det resultatet vi har blitt enig om.

Gruppe regler:

1. Medlemmer skal møte punktlig på avtalte møter. Gruppen krever at alle medlemmer er tilstede på minimum 75% av alle møter.
2. Alle skal si ifra hvis det er noe de trenger hjelp med.
3. Man skal si ifra i god tid hvis man ikke kan komme, eller kommer for sent.
4. Alle I gruppen har like mye rett til å si sine meninger, og alle meninger skal bli hørt på av alle, for å unngå konflikt og for at alle skal bli hørt.
5. Alle må godta at ekstra møter kan forekomme hvis det blir nødvendig.
6. Grove brudd på gruppens regler kan føre til utkastelse.

Underskrift:

X _____
Erik Haaland

X _____
Ola Skarpmoen Eriksen

X _____
Oscar lindberg

X _____
Sebastian Nomme

Product backlog

User Story	ToDo	Tid estimat
Som en bruker vil jeg kunne skrive og publisere innlegg, slik at jeg kan dele informasjon med andre.	Autentisere backend	86t
	Opprette bruker modell	5t
	Autentisering front-end	35t
	Programmere feed.	117t
	lage post funksjon.	50t
	Programmere innleggsfunksjon	72t
	Designe feed	59t
	Sette opp git/github	23t
Som en bruker ønsker jeg en egen profilside, for å ha all informasjon om meg på et sted.	programmere profilkort.	103t
	Laste inn brukerens poster	64t
	Designe profil	50t
Som en bruker vil jeg kunne 'upvote' andre sine innlegg, for å gi noe tilbake for nyttig informasjon og kvalitetssikre innlegg.	Lage like funksjon.	23t
	laste opp tre siste som har likt innlegget.	37t
	Design.	5t
Som en bruker vil jeg ha muligheten til å grafisk fremstille mine investeringer av kryptovaluta, for å ha bedre oversikt over mine midler.	Lage en grafisk funksjon.	45t
	Hente inn live data for krypovaluta.	86t
	Legge inn data.	53t
	Design portfolio	23t
	Begregning av verdi.	9t
Som en bruker vil jeg ha mulighet til å dele min portefølje, for å få tilbakemelding fra andre brukere.	Hente ut grafisk fremstilling, integrere i post. (snapshot)	21t
	Tilgang til andre sin profil.	43t

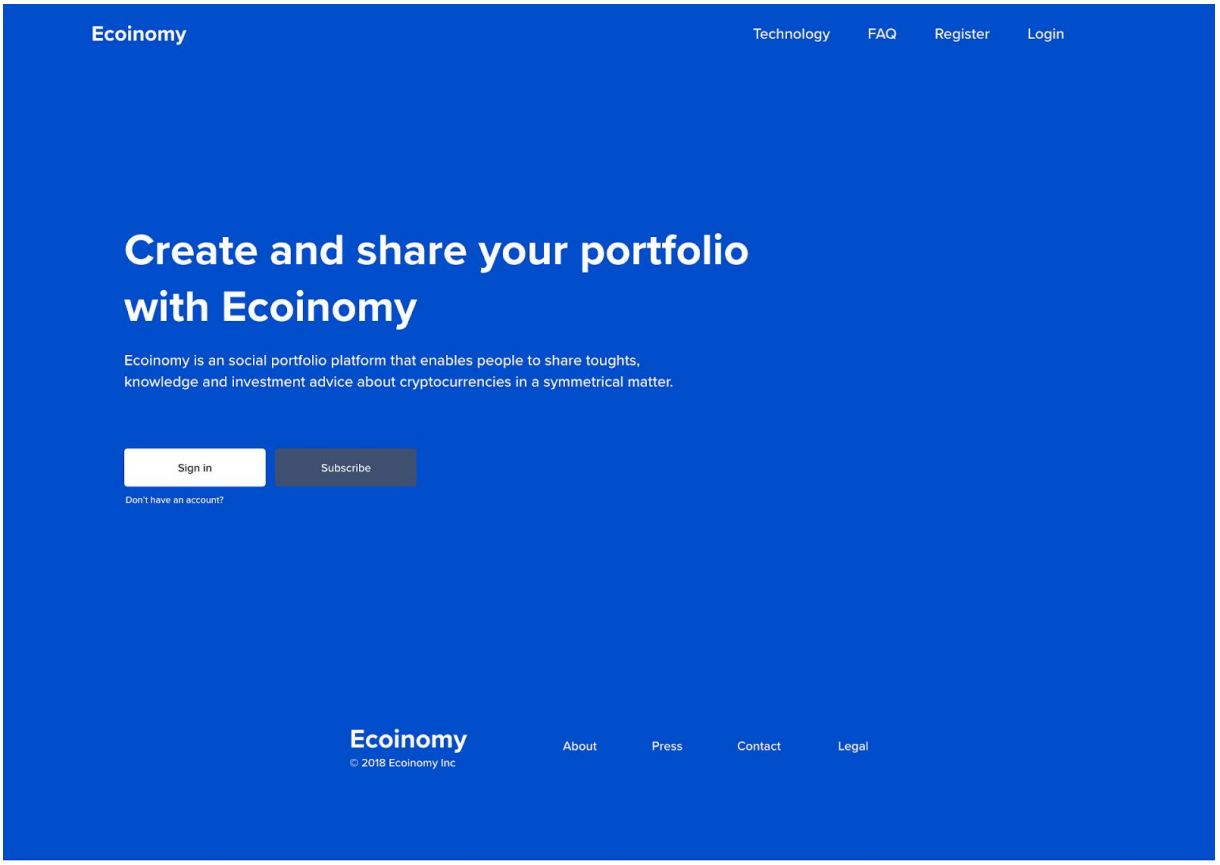
Som en bruker ønsker jeg å kunne se andre sine investeringer slik at jeg kan få inspirasjon og råd til fremtidige kjøp.		
Som en bruker vil jeg kunne lage en gruppe, for å knytte nettverk og samarbeide med andre brukere.	Lage gruppe funksjonen.	16t
	integrere feed funksjon.	41t
Som en bruker vil jeg kunne bli medlem av grupper, for å knytte nettverk og samarbeide med andre brukere.	Sende forespørsle/invitere.	13t
Som en bruker vil jeg kunne rapportere upassende og støtende innlegg, for å sikre egen trivsel på nettsiden.	Lage rapporter knapp.	5t
	flag funksjon.	14t
Som en bruker vil jeg kunne rapportere andre brukere, fordi de kan dele støtetene innhold som ikke er i tråd med retningslinjene.	Lage knapp.	1t
	Flag funksjon.	14t
Som en bruker vil jeg at innhold som har blitt besvart ikke slettes fordi jeg har bidratt økonomisk eller med støtte med mindre det bryter med rettningslinjene.	Fjerne eierskap til post.	
Som bruker vil jeg ha en chattefunksjon for at jeg skal kunne konversere med andre brukere i et lukket miljø.		
Som administrator ønsker jeg at hver bruker skal ha en unik ID slik at det er lett å skille de fra hverandre og å behandle	Lage funksjon som setter en unik ID på ny bruker.	3t
	HTML login.	108t
	GraphQL server	94t

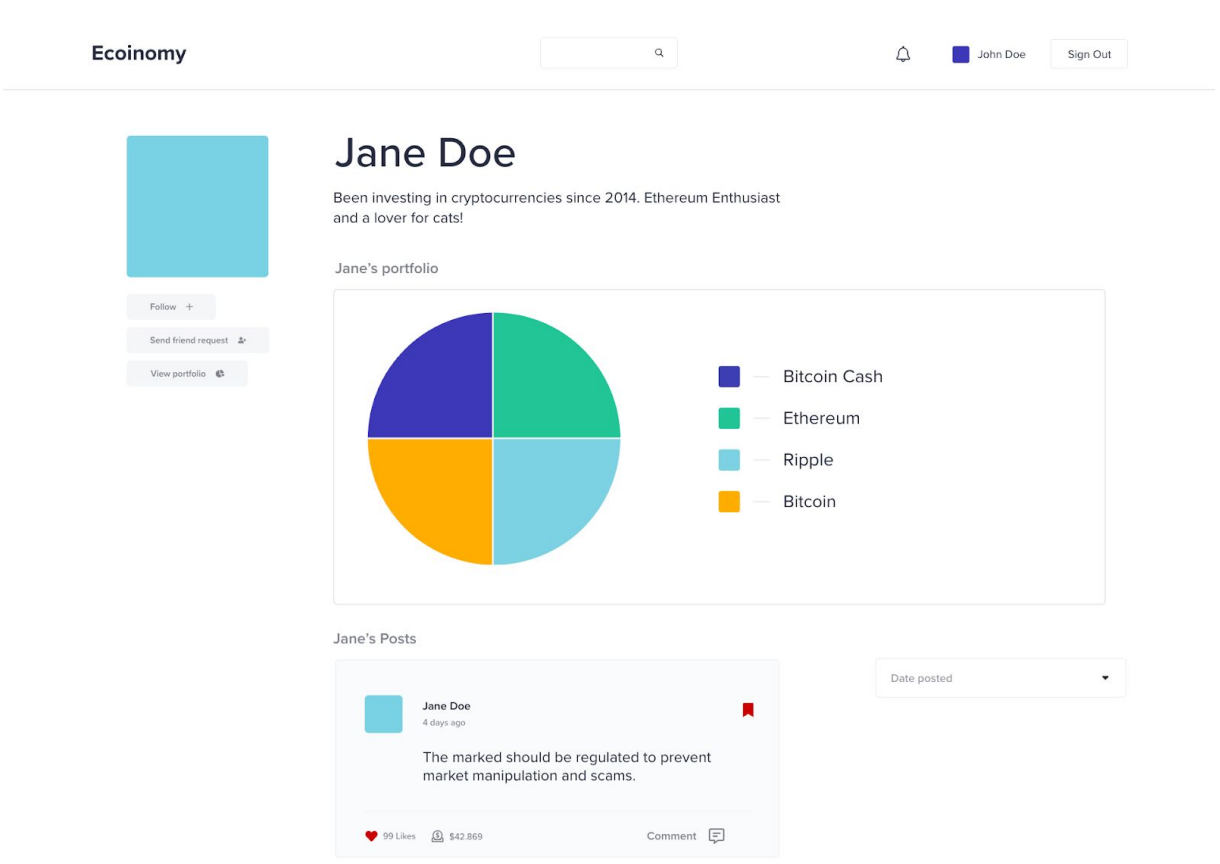
riktig bruker.	Prisma	75t
	Lage administrator bruker.	14t
Som administrator vil jeg kunne slette innlegg, for å sikre kvalitet på siden.		
Som administrator vil jeg kunne utestenge brukere ved brudd på regler, for å opprettholde saklighet.		
Som en administrator vil jeg kunne slette grupper og innhold, for å hindre aktiviteter som er brudd på retningslinjer.		
	MoSCow	10t
	Lage papirprototype	10t
	Lage testscenarier til papirprototype	2t
	Prototype test nr.1	9t
	Forbedre prototype	4t
	Prototype test nr.2	11t
	Sitemap	4t
	Rikt bilde.	6t
	Forbedre rikt bilde	4t
	Wireframe	14t
	Risikonanlyse	5t
	Brukerhistorier	21t
	Forbedre brukerhistorier	9t
	Forbedre/utvidelse av MoSCoW	3t
Analyse		

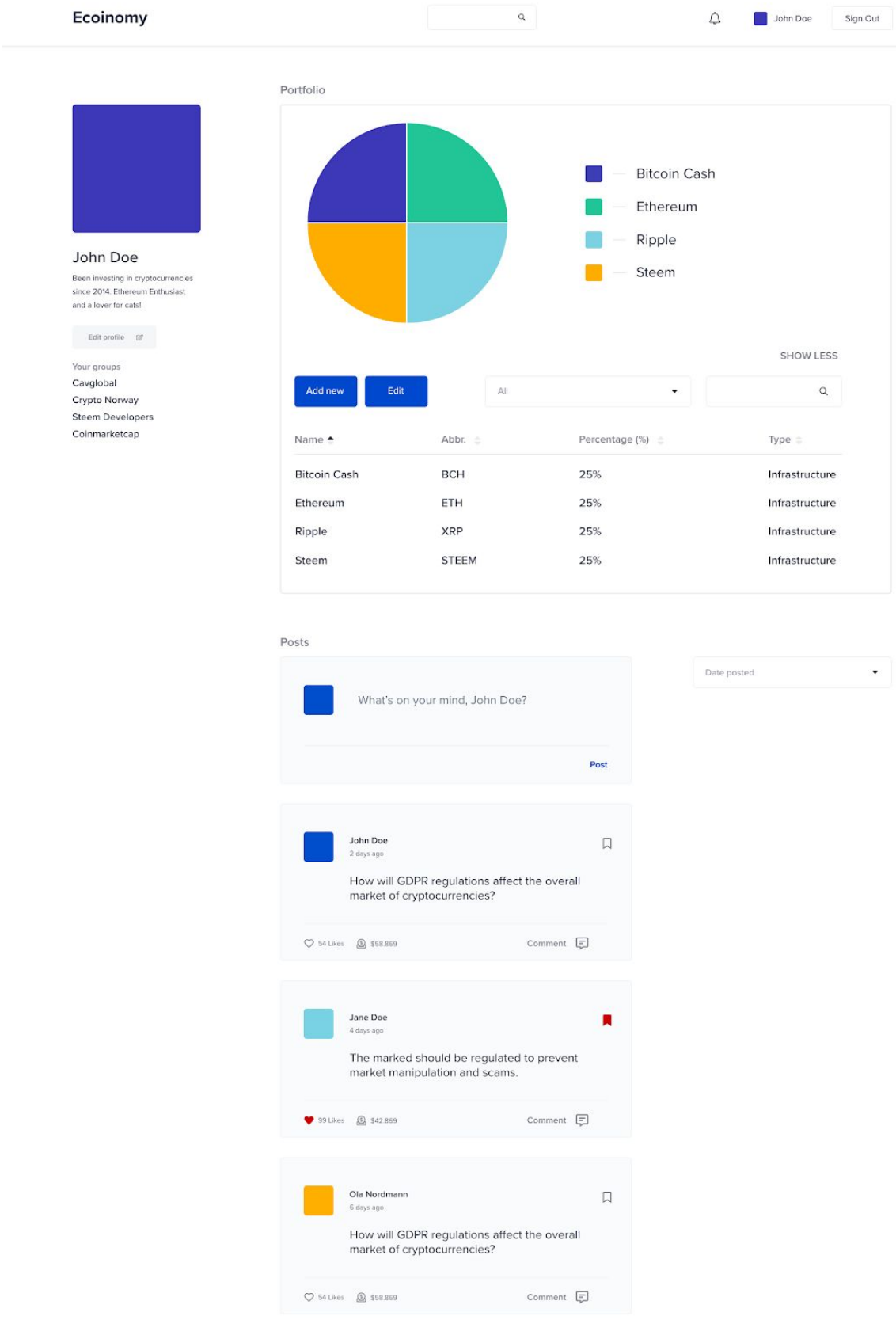
Oppgaver til rapportskriving

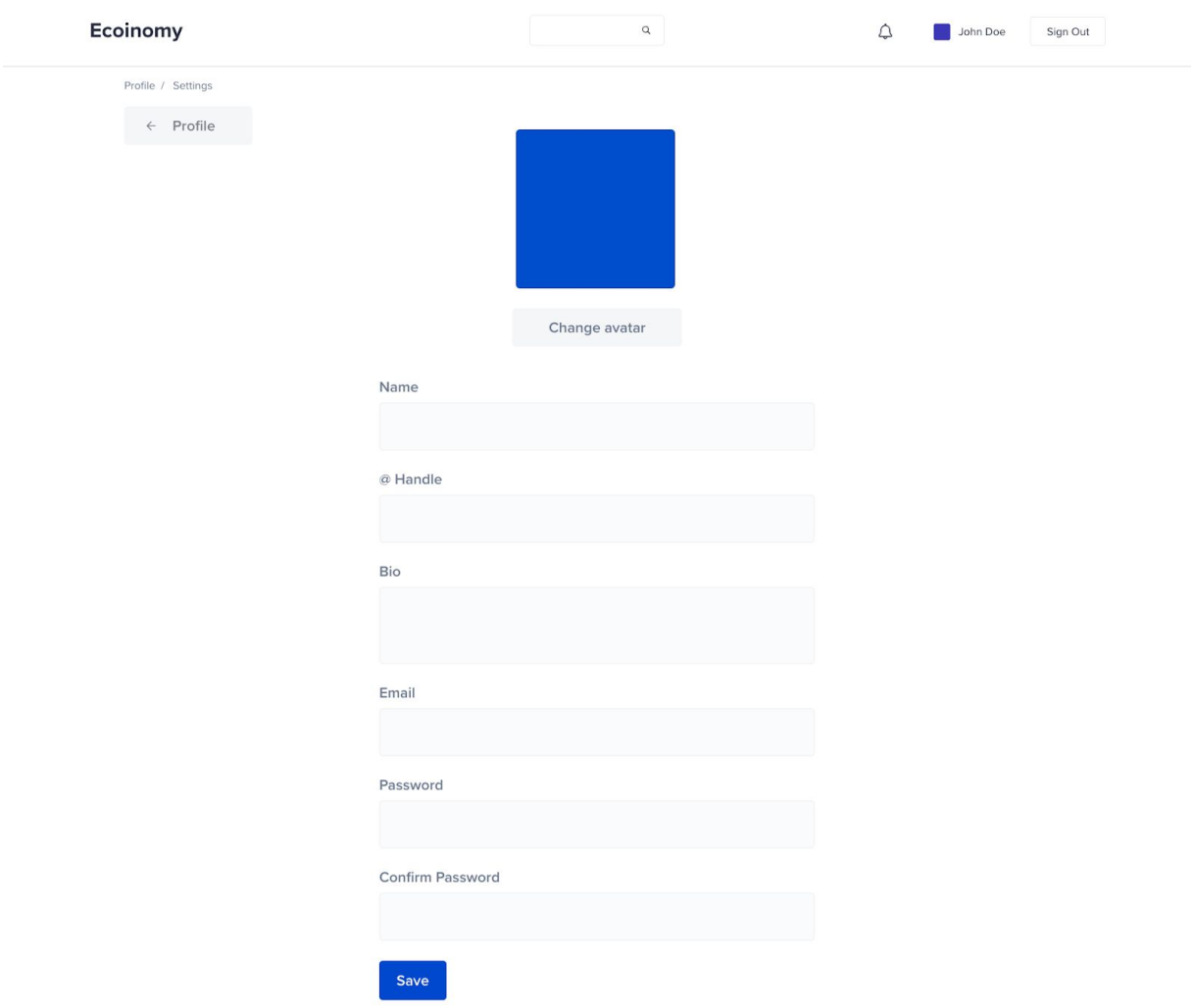
oppgaver til Rapport:	skrive om Metodologi, hva har vi brukt og hvorfor.
	skrive om Kvalitet, hvordan har vi definert det i vårt prosjekt.
	skrive om prosjekt styring
	forklare hvilken Teknologi vi bruker, og hvorfor.
	skrive om Standarer (dokumentasjon, design, koding etc)
	Skrive om hvordan sprinter. (gjennomføring/hoveddel)
	skrive om Design. Hvilken prinsipper har vi brukt.
	skrive om Rikt bilde
	skrive om Papir prototype og testing.
	skrive om styringsverktøy, hvem har vi brukt og hvorfor
	skrive om gruppen, liten forklaring på hvem vi er.
	Skrive om backlog, og hvordan vi har ført timer.
	skrive om vår MOSCOW tabell, og bruker historier.
	skrive inntroduksjon.
	skrive forord. (takke janis, nyskapning etc)
	skrive refleksjon.

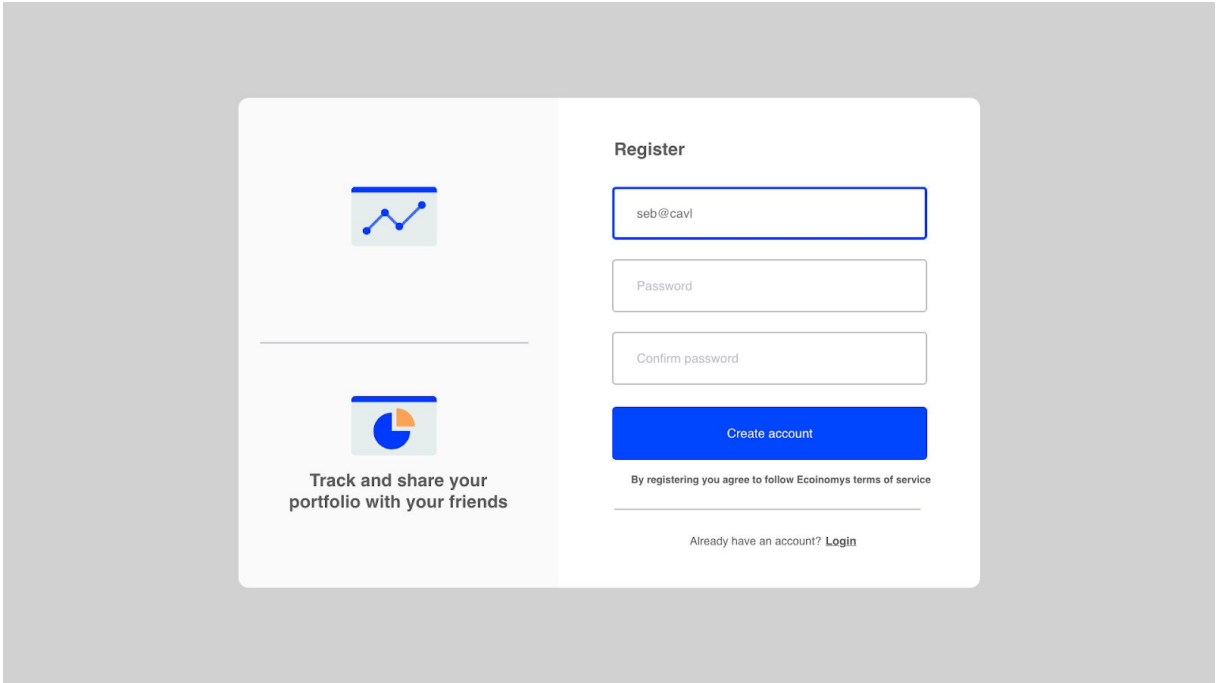
Mockups i Xd:


Ecoinomy
Technology
FAQ
Register
Login
Create and share your portfolio with Ecoinomy
Ecoinomy is an social portfolio platform that enables people to share toughts, knowledge and investment advice about cryptocurrencies in a symmetrical matter.
Sign in
Subscribe
Don't have an account?
Ecoinomy
© 2018 Ecoinomy Inc
About
Press
Contact
Legal


Ecoinomy
John Doe
Sign Out
Jane Doe
Been investing in cryptocurrencies since 2014. Ethereum Enthusiast and a lover for cats!
Follow
Send friend request
View portfolio
Jane's portfolio
Bitcoin Cash
Ethereum
Ripple
Bitcoin
Jane's Posts
Date posted
Jane Doe
4 days ago
The marked should be regulated to prevent market manipulation and scams.
99 Likes
$42.869
Comment


Ecoinomy
John Doe
Sign Out
John Doe
Been investing in cryptocurrencies since 2014. Ethereum Enthusiast and a lover for cats!
Edit profile
Your groups
Cavglobal
Crypto Norway
Steem Developers
Coinmarketcap
Portfolio
Bitcoin Cash
Ethereum
Ripple
Steem
SHOW LESS
Add new
Edit
All
Name
Abbr.
Percentage (%)
Type
Bitcoin Cash BCH 25% Infrastructure
Ethereum ETH 25% Infrastructure
Ripple XRP 25% Infrastructure
Steem STEEM 25% Infrastructure
Posts
Date posted
What's on your mind, John Doe?
Post
John Doe
2 days ago
How will GDPR regulations affect the overall market of cryptocurrencies?
54 Likes
$58.869
Comment
Jane Doe
4 days ago
The marked should be regulated to prevent market manipulation and scams.
99 Likes
$42.869
Comment
Ola Nordmann
6 days ago
How will GDPR regulations affect the overall market of cryptocurrencies?
54 Likes
$58.869
Comment


Ecoinomy
John Doe
Sign Out
Profile / Settings
Profile
Change avatar
Name
@ Handle
Bio
Email
Password
Confirm Password
Save


Register
seb@cavl
Password
Confirm password
Create account
By registering you agree to follow Ecoinomys terms of service
Already have an account? Login
Track and share your portfolio with your friends