Galea Games

Bacheloroppgave ved OsloMet
Institutt for informasjonsteknologi
Teknologi, Kunst og Design (TKD)

Gruppe 11

Sigurd Øvre Bjørndal
Benjamin Alexander Skjerve Jørgensen
Magomed Musajevitsj Khatsjukajev
Felix Skau Leypoldt

OSLOMET

OSLO METROPOLITAN UNIVERSITY
STORBYUNIVERSITETET

Vedleggsliste

Link til GitHub-repository:

https://github.com/s333933/GaleaGameDemo-1

Vedlegg 1: Prosessrapport

Innholdsfortegnelse

1. Forord

Denne prosessrapporten tar for seg refleksjoner om resultatet, og gir en begrunnelse for de valgene som ble tatt under utviklingen av den webbaserte spillapplikasjonen Galea Games. Prosjektoppgaven var ledet og utlevert av Lars Føleide, daglig leder for Enry AS og Galea AS. Prosjektet var internt veiledet av Henrik Lieng, førsteamanuensis, med tilhørighet til fakultet for teknologi, kunst og design, OsloMet - storbyuniversitetet.

Før valg av oppdragsgiver, var det bestemt at prosjektgruppen ønsket å jobbe med utvikling av en applikasjon. Prosjektgruppen kontaktet potensielle oppdragsgivere for bachelorprosjektet, og valget falt på Galea AS, som ønsket et moderne grunnlag for en webbasert spillapplikasjon. Etter et møte med daglig leder var det satt et mål om å nå frem til personer uten noen spesiell kunnskap eller erfaring med aksjemarkedet. Med Galea AS ble muligheten åpnet opp for å anvende kunnskapen og erfaring prosjektgruppen hadde tilegnet seg etter nesten fullført bachelorgrad. Dette inkluderer, men er ikke begrenset til programutvikling, systemutvikling, menneske-maskin-interaksjon, databaser, webprogrammering og testing av programvare.

Vi vil takke Lars Føleide og hele Galea AS for denne spennende muligheten samt god veiledning gjennom prosjektet. I tillegg, vil vi takke Boning Feng, ansvarlig for bachelor prosjektfaget og intern veileder Henrik Lieng for gode råd og veiledning under prosjektet.

Lærdommen og erfaringen prosjektgruppen har tatt med seg fra dette bachelorprosjektet har vært meningsfullt. Alle medlemmene i gruppen setter stor pris på muligheten for å skrive bachelorprosjekt, selv i en krevende tid. Denne lærdommen og erfaringen vil prosjektgruppen ta med seg videre i livet og få god nytte av, både i videre utdanning og arbeidslivet.

2. Introduksjon

Denne rapporten går gjennom våre refleksjoner om ideer, valg og omstendighetene som påvirket dette. Prosjektperioden besto av tilegning av nye programmeringsspråk og systemer, utvikling i full-stack og utøve tilegnet informasjon fra tidligere fag i praksis. Denne kunnskapen og erfaringen har vært grunnlaget for utviklingen av applikasjonen, Galea Games.

3. Planlegging og metode

Første steg i prosessen var å sette seg inn i de nye språkene og systemene som oppdragsgiver hadde bestemt på forhånd. Språkene og systemene som skulle benyttes var ReactJS, Redux og Firebase. Oppdragsgiver anbefalte og anskaffet Udemy, som er en online læringsplattform med over 130 000 kurs innenfor ulike fagfelt (Udemy, 2021). For prosjektet var det gitt tilgang til kurs for å bygge opp grunnlag for ReactJS, Redux og Firebase. Selv om Udemy ga et godt grunnlag for dette, var tilegningsprosessen ikke lineær, og gruppen ble nødt til å benytte alternative læringsmetoder slik som prøve- og feilemetoden. Dermed ble utviklingsprosessen ujevn basert på tid- og ressursfordeling.

Videre, ble det prosjektgruppen hadde lært i Udemy kurset anvendt i en praktisk øvelse fra oppdragsgiver. Prosjektgruppen fikk i oppgave å bruke ReactJS for å lage en enkel to-do-list applikasjon, som skulle liste opp selvvalgte gjøremål og påminnelser. Dette var en innføring i å utvikle i React og hva dette Javascript-biblioteket hadde å tilby.

I neste steg var det viktig opprette en struktur for hvordan spillapplikasjonen skulle se ut og hvilke funksjonaliteter skulle være inkludert. For dette var det hensiktsmessig å lage en LoFi prototype som visualiserte design og funksjonalitet på et primitivt stadiet. Prototypen ble benyttet som grunnlag for å vise oppdragsgiver mulig design og funksjonalitet, og dermed å få innspill og kommentarer på forbedringer eller endringer til spillapplikasjonen.

Etter LoFi prototypen var ferdigstilt, var det formålstjenlig å starte utvikling av selve spillapplikasjonen. Oppdragsgiveren ga ingen krav for å lage en HiFi prototype, slik at arbeidet med full-stack utviklingen kunne komme i gang umiddelbart. Arbeidet for utviklingen var delt i sprinter, som brukte Scrum-metodikk. Dette innebærer at prosjektgruppen benyttet en modifisert variant av Scrum, hvor hver fredag ble det holdt møter med oppdragsgiver. På disse møtene ble påbegynt og ferdigstilt funksjonalitet gjennomgått med oppdragsgiver slik at hele utviklingsteamet fikk en god forståelse av både behov og målsetting. Dette detaljeres ned til konkrete arbeidsoppgaver. I møtene ble det også spesifisert hvilke funksjoner av spillapplikasjonen som var planlagt, påbegynt og ferdigstilt. Vanligvis i Scrum blir det benyttet en tavle med kolonner som holder oversikt på dette (Nes, 2019). Derimot, i vårt tilfelle ble det tatt notater under hvert møte som la fram informasjon om hvilke funksjonalitet som var planlagt, påbegynt og ferdigstilt (*Vedlegg 4: Notater fra alle møter med arbeidsgiver)*.

4. Utviklingsprosessen

Utviklingen til prosjektet har vært delt opp i sprinter, hvor hver sprint har fokus på en eller flere spesifikke funksjonaliteter i spillapplikasjonen. Eksempelvis, startet første sprint med utvikling av en nettside for Galea Games. Nettsiden gir oversikt over prosjektgruppens dokumenter, slik som prosjektskisse, forprosjektrapport og den endelige sluttrapporten.

Innenfor dette har det vært egne utfordringer for hver enkelt sprint. Dette har eksempelvis vært leveransen av Firebase autentiseringsmetoder for innlogging til spillapplikasjonen. Primært dreier det seg om feil benyttelse av ruting i React-Router delen av spillapplikasjonen. Denne delen sørger for at navigering er forhåndsbestemt før spillapplikasjonen presenteres for brukeren. Det har derfor vært en grunnleggende del av spillapplikasjonen som må være implementert riktig, slik at spillapplikasjonen tjener sitt formål. Når ruting-delen av spillapplikasjonen var ferdigstilt på riktig måte, var det mulig å autentisere

brukere og gi tilgang til private ruter som kun var tilgjengelig for den gjeldende brukeren. Problemet lå ved omdirigering etter at brukeren enten logget inn med Facebook, Google eller e-post. Med det tidligere oppsettet vi hadde, uten en ruting, ville brukeren bli omdirigert til ikke-predefinerte ruter av spillapplikasjonen. Ved å løse dette var det mulig å begrense brukertilgangen til et predefinert sett med ruter (React, 2021).

Ettersom at flere av betingelsene til oppbygging og funksjon av spillapplikasjonen var forhåndsbestemt av oppdragsgiver, slik som ReactJS, Redux og Firebase, var det lite rom for gruppen å velge alternative løsninger. Til tross for dette var det beleilig å benytte seg av Python med en Flask-server for å kjøre deler av backend for spillapplikasjonen. Grunnen bak valget av Python og Flask var fordi JavaScript og ReactJS i seg selv ikke hadde de nødvendige funksjonene for å kjøre skript i backend via en server (Jayathilaka, 2018). Dette førte derimot til flere faglige utfordringer, fortrinnsvis opplæring av Python og Flask. I tillegg var det andre utfordringer som å få ReactJS til å kommunisere med scriptet i Flask-serveren. Dette var løst ved å sende GET-spørringer til Flask-serveren, som da initierte et Python-skript som fylte Firebase-databasen med et datasett for spillet.

Gjennom utviklingen av prosjektet har prosjektgruppen hatt god kommunikasjon med oppdragsgiver. I startfasen var det en strengere og mer profesjonell stemning, men ettersom både oppdragsgiver og prosjektgruppen ble bedre kjent med hverandre ble også stemningen mer avslappet. Dette førte til en enklere kommunikasjon med oppdragsgiver, som igjen førte til bedre samarbeid.

5. Forprosjektrapport (Kravspesifikasjon) og dens rolle

I vedlagt dokumentasjon (Vedlegg 2: Forprosjekt (Kravspesifikasjon)), er det spesifisert mål og rammebetingelser for prosjektoppgaven. I løpet av prosjektet, har det vært noen få endringer, slik som tillegget av Python som

programmeringsspråk og server-oppsett. I forprosjektrapporten er det spesifisert mål og rammebetingelser for prosjektoppgaven. I løpet av prosjektet, har det vært noen få endringer, slik som tillegget av Python som programmeringsspråk og server-oppsett. Når det kommer til design og implementering så hadde kravspesifikasjonene mye å si fordi den var allerede satt av arbeidsgiver. Ettersom at arbeidsgiver hadde satt grundige krav for spesifikke deler av mål og rammebetingelser for spillapplikasjonen, ble den brukt som en blåkopi.

Til tross for det som ble satt som krav i rammebetingelsene i det vedlagte dokumentet, var ikke nødvendigvis alt benyttet. Grunnlaget er at verktøyene, slik som CircleCi, Expo og nTask ikke fant noen nytte under utviklingen av prosjektet. Målet for testing var derimot rettet til brukertesting, med fokus på funksjonalitet og forståelse for brukeren. I tillegg, hadde prosjektgruppen planlagt å benytte seg av et kanban-board. Ettersom, notater ble tatt og gruppen fikk tilgang til disse etter ukentlige møter, viste det seg å ikke være nødvendig.

Sluttproduktet er til dels i samsvar med de målene som ble satt i forprosjektrapporten. Hvor det har vært avvik, er med rangering og nivåinndeling basert på prestasjon, oversikt over venners prestasjoner og implementere funksjonaliteten til å spille med live datasett uthentet fra NASDAQ. Grunnen bak dette har vært tidsprioriteringer av andre funksjonaliteter, som å gi bruker mulighet til å logge inn med støttede Firebase autentiseringsmetoder og spille med et predefinert datasett. I tillegg, på grunn av de samme prioriteringene, har det vært mindre fokus på et mobilt oppsett til spillapplikasjonen, og derfor er dette ikke ferdigstilt for mobile enheter. Selv om prioriteringen har vært lav for dette, er det fortsatt mulige å benytte spillapplikasjonen på mobile enheter. Derimot er disse kravene kun en pekepinn som ble satt før prosjektets start og arbeidsgiver konstatert at målet har vært utvikle spillapplikasjonen mest mulig innen tidsrammen.

For å videre kunne nå de målene som er satt i forprosjektrapporten, har det vært underliggende mål som først må være på plass før en kan gå videre med eksempelvis et live datasett. Ved å bygge fundamentet for datahåndteringen til spillapplikasjonens primære spillfunksjon og oppdatering av graf med et datasett, vil det være mulig å vite hvordan spillapplikasjonen håndterer data, både predefinert og live. Dette tilgjengeliggjør videreutviklingen av spillets live funksjonalitet. Årsaken bak dette er fordi API-er som Quandl, anbefalt og ønsket av arbeidsgiver, krever betaling av bruk. Prosjektgruppen ønsket å forsikre seg om å unngå unødvendige store kostnader av API-bruk. For at dette skulle implementeres, måtte arbeidsgiver stå for disse kostnadene. I tillegg, var det manglende dokumentasjon for gjeldende API, noe som igjen skapte usikkerhet ved bruk. Realiseringen av innhenting av live datasett, er en oppgave for seg selv, og vil kreve tid og ressurser for å oppfylles slik at det dekker spillapplikasjonens endelige mål.

6. Prinsipper, bibliotek og rammeverk

ReactJS er et JavaScript-bibliotek som benyttes for å bygge brukergrensesnitt. ReactJS benyttes som oftest til applikasjoner som sikter etter applikasjoner som benytter én side (Flavio Copes, 2018). Hensikten til ReactJS er å være et raskt JavaScript bibliotek for utviklingen av webapplikasjoner (React, 2019). Andre sider som benytter seg av biblioteket til ReactJS er for eksempel, Facebook, Dropbox, Codecademy, Airbnb, Netflix, og mange flere (Węglarz, 2020).
Før vi kunne gå videre med å utvikle applikasjonen med ReactJS, måtte gruppen ta noen innføringskurs på basis av ingen tidligere erfaring. Som nevnt tidligere i «Planlegging og metode» ønsket arbeidsgiver at prosjektgruppen skulle ta et Udemy kurs, kalt «Build an app with React, Redux and Firestore from scratch» (Cummings, 2021). Dette var fordi at arbeidsgiver ville at i løpet av de første ukene skulle prosjektgruppen gjøre seg kjent med språket og verktøyet.

Dette kurset varer totalt 23 timer med 224 foredrag om hvordan man starter å bruke ReactJS fra utvikling av en applikasjon til publisering. Opplæringen inkluderte temaer som, React Concepts, Thinking in React, Routing in React, Introduction to Redux, Asynchronous Code, Introduction to Firestore, Firebase Authentication og Firestore Security Rules (Cummings, 2021). I tillegg har prosjektgruppen gått gjennom andre temaer som Create React App, beskyttelse av API-nøkler, eksternt vindu for innlogging, Single Responsibility Operations, Firebase som en tjeneste og service workers.

6.1. React Concepts

Hvorfor velge React? Store bedrifter, som nevnt tidligere bruker denne teknologien. React er raskt, fordi når en lagrer og oppdaterer dokumentet, sendes det til Virtual Document Object Model (VDOM), og så til Real Document Object Model (Real DOM).

Virtual Document Object Model (VDOM) er et programmeringskonsept der en ideell eller «virtuell» representasjon av et brukergrensesnitt holdes i minnet og synkroniseres med det den «virkelige» DOM av et bibliotek som ReactDOM. Denne prosessen kalles forsoning på norsk og *reconciliation* på engelsk. (React, 2019). ReactJS bruker VDOM for å forbedre ytelsen. Den bruker det observerbare for å oppdage endringer i tilstanden. ReactJS bruker en effektiv diff-algoritme for å sammenligne versjoner av virtuell DOM. En diff-algoritme får et samlet output av to forskjellige input. (Ravichandran, 2018) Deretter sørger den for at oppdateringer blir sendt til den virkelige DOM for gjengivelse av brukergrensesnittet. (React, 2019)

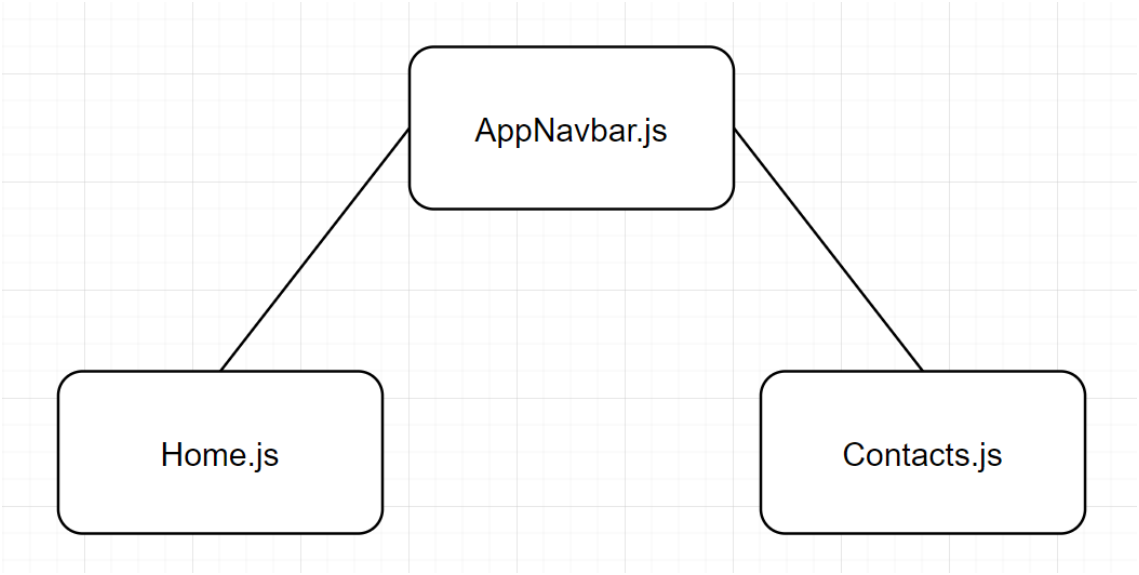
6.2. Thinking in React

ReactJS tillater utviklingsprosessen å være skalerbar, slik at selv små prosjekter kan videreutvikles til større og mer omfattende prosjekter. I tillegg til dette, gir det utvikleren muligheten til å bryte ned komplekse komponenter og gjenbruke koden for de samme komponentene til andre prosjekter. Dette

danner et økosystem for smidig utvikling, hvor utviklere kan enkelt og intuitivt produsere kode for webapplikasjoner med gjenbrukbar kode (React, 2019)

6.3. Single Responsibility Principle

Single Responsiblity Principle dreier som å kun tildele ett ansvar for hver gjeldende klasse (Oloruntoba, 2020). Som nevnt over i **Thinking in React** kan man bruke komponentene på nytt ved at man isolerer hver del av systemets egenskaper. Man kan da for eksempel ha en hjemmeside «Home.js» som har til hensikt å fungere som kun én hjemmeside. Videre kan man ha en komponent «AppNavbar.js» som vil kun igjen fungere som en navigeringsmeny. Skal man da opprette en ny side med «Contacts.js» som inneholder kontakter, kan man gjenbruke «AppNavbar.js» til denne siden. Eksempel på dette nedenfor.



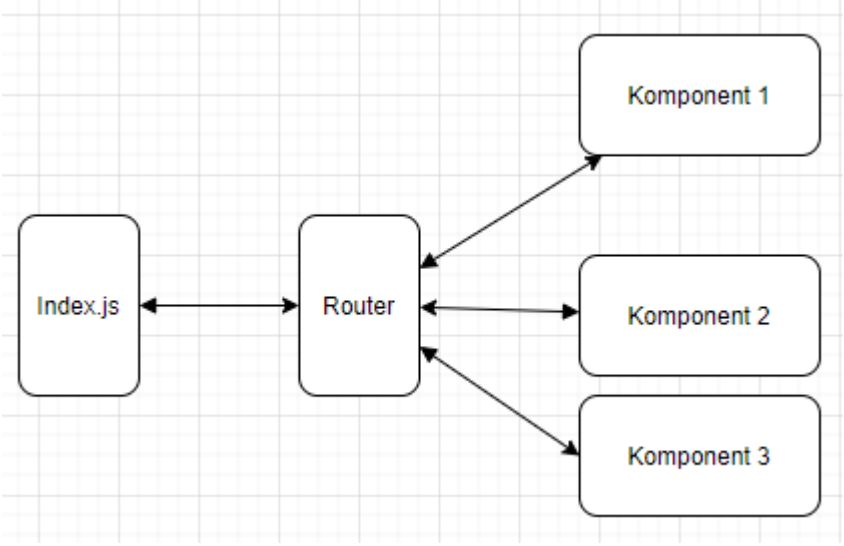
Figur 1 - Eksempel på selvstendige og gjenbrukbare komponenter i form av abstrakt modellering av JavaScript-moduler ved bruk av Draw.io

Hver komponent har til hensikt å gjøre kun én og bare én ting. Dette gjør det mulig å gjenbruke koden ettersom de er uavhengige av hverandre. «Software reuse is considered as the key to a successful software development because of its potential to reduce the time to market, increase quality and reduce costs. This increase in demand made the software organizations to envision the use of

software reusable assets which can also help in solving recurring problems.» (Bhargava Mithra Konda, 2010)

6.4. Routing in React

React Router er standard-biblioteket innenfor ReactJS som holder styr på rutingen til brukergrensesnittet i henhold til URL-lenkene (Rutene). Routing har en enkel API (Application Programming Interface) med funksjonalitet som Lazy Code Loading, Dynamic Route og Transition Handling bygget inn (freeCodeCamp, 2016). Lazy Code Loading - er praksisen med å vente med å laste eller initialisere koden fram det faktisk er nødvendig for en ønsket operasjon. Dette er for å spare ressursene og forbedre ytelsen til applikasjonen (Imperva, u.d.). Eksempel på utkast til React Routing er vist nedenfor.



Figur 2 - Modellering av utkast til React Routing laget i Draw.io

Dynamic Route er ruter som applikasjonen kan navigeres til ved initialisering av selve applikasjonen. Ruter utenfor disse prefiksene rutene er ikke mulig å havne i. Dette er for å ha et kontrollert miljø for applikasjonen. Routing må planlegges i forhold hvor i applikasjonen brukeren skal sendes til (React Router, u.d.). Transition Handling, er et lite React bibliotek som håndterer overgangen

mellom komponenter, Redux tilkoblede komponenter og Ruter og andre hendelser på en oversiktlig måte (npm, u.d.).

6.5. Introduction to Redux

Redux er en state container, som brukes i forbindelse med ReactJS for å opprettholde at klient, server og utviklingsmiljøet er konsistent med hverandre (Redux, u.d.). Betydningen dette har er at et globalt state kan holde hele applikasjonens tilstand på ett sted (Redux, u.d.).

6.6. Asynchronous code

Eller asynkron kode, er kode som kjører med to betingelser. Det kan automatisk returnere et løfte og kan bruke kommandoen «await», som kan avvente en betingelse. En asynkron funksjon returnerer løfter som løses med funksjonens returverdi eller avvises med at det dukker opp et unntak som ikke er håndtert (Mozilla, 2021).

6.7. Introduction to Firestore

Cloud Firestore er en NoSQL dokumentdatabase, som er bygget for automatisk skalering og fleksibilitet. En NoSQL-database som benyttes i Firestore-grensesnittet har mange av de samme funksjonene som en tradisjonell SQL-database. I motsetning til en SQL-database er det ingen tabeller eller rader. I stedet lagrer du data i dokumenter, som er organisert i kolleksjoner (Google, 2021).

Firestore er optimalisert for lagring av store kolleksjoner med små dokumenter. Alle dokumenter må lagres i kolleksjoner. Dokumentene kan inneholde underkolleksjoner og *Nested Objects* (Google, 2021). Både Nested Objects og underkolleksjonene kan inneholde primitive verdier som strenger eller komplekse objekter som lister (Google, 2021).

6.8. Firebase Authentication

Autentisering i Firebase tillater en brukers informasjon og data å bli lagret på en sikker måte i skyen. Firebase leverer løsninger for back-end tjenester, intuitiv SDKs (Software development kit) og biblioteker for brukergrensesnitt. I tillegg, støtter denne autentiseringen innloggingsmetoder som for eksempel Google, Facebook, telefonnummer, e-post og passord (Google, 2021).

Firebase samler nødvendig informasjon som for eksempel brukernavn og passord fra brukeren. Denne informasjonen brukes for å danne en tilgangstoken som videresendes til Firebase Authentication SDK, som verifiserer informasjonen. En respons blir tilbakesendt til brukeren dersom informasjonen er riktig og gir da tilgang for gjeldende bruker. Brukere som er tildelt tilgang vil være i stand til å lese og skrive til Real-Time databasen eller Firestore databasen til Firebase (Google, 2021).

6.9. Securing the application

En sikker applikasjon vil avgrense hva hver enkelt bruker kan få tilgang til. Ved hjelp av React Router kan man begrense tilgang til ulike ruter. Vi kan ha public routes som alle har tilgang til, private route som bare autentiserte brukere har tilgang til. Til slutt kan man ha restricted routes som er avgrenset slik at ikke-autentiserte brukere ikke får tilgang (Singh, 2020). For å forklare videre, så vil man ikke vise innloggingssiden før etter brukeren har logget inn til tjenesten. Hvis en autentisert bruker går til innloggingssiden, vil brukeren omdirigeres til en allerede innlogget side.

6.10. Firestore security rules

Med Cloud Firestore Security Rules kan man fokusere på å bygge en brukeropplevelse, uten å måtte tenke på administrering av infrastruktur eller skrive autentiserings og autorisasjonskode på serversiden. Sikkerhetsregler gir tilgangskontroll og datavalidering i et enkelt men uttrykksfullt format. For å videre bygge rolle- og brukerbaserte tilgangssystemer som holder brukernes

data sikre, må man benytte seg av Firebase Authentication med Cloud Firstore Security Rules (Google, 2021).

Alle sikkerhetsregler for Cloud Firestore består av «match statements», som identifiserer dokumenter i databasen, og tillater uttrykk som styrer tilgang til disse dokumentene (Google, 2021).

For at brukerne i appen ikke skal ha tilgang til undersider de ikke er autoriserte til å komme seg inn på ble det brukt protected routes. I appen sitt tilfelle ble dette brukt på «Scorepage.js» siden en bruker av appen får tilgang til å se scorepage-siden når levelet er fullført. Om spillet ikke er fullført, så er det ikke noe scorepage å vise, dermed er ikke brukeren autorisert til å få tilgang til denne siden.

På Firestore er det også mulig å strukturere og legge til betingelser til appens Cloud Firestore Security Rules (Hreniak, 2019). Betingelsene i appen sin Cloud Firestore Security Rules sa at brukere hadde lese og skrive rettigheter så samt dette ble gjort før den 28.06.2021. Dette ble gjort fordi det er en testserver og Firestore er da åpen for alle forespørsler og brukere innen nevnt tidsramme. Dette måtte endres, nemlig fordi denne tidsperioden gikk ut.

6.11. Beskyttelse av API-nøkler

Å beskytte API-nøkler ved å skjule dem fra offentligheten gjør det mulig å minimere tilgangen til gjeldende nøkler. Ved å ikke sikre tilgangen til API-nøklene kan det føre til at uautoriserte parter begår ondsinnete handlinger som å overbelaste databasen og eventuelt føre til kostnader dersom Firebase databasen er under et betalt medlemskap. En form for beskyttelse er å ignorere API-nøklene i et repo på GitHub med en gitignore-fil. Denne filen sjekker hva som må ignoreres før et repo oppdateres eller lastes opp til GitHub (git, u.d.). I henhold til spillapplikasjonen ble gitignore ikke benyttet fordi det er tre ulike parter, prosjektgruppen, Galea AS og OsloMet, som må ha tilgang til dette.

6.12. Eksternt vindu for innlogging

Ved å bruke Firebase er det mulig å ta i bruk autentiseringsmetoder slik som Google og Facebook. Ved bruken av disse i en webapplikasjon vil det presenteres for brukeren som et eksternt vindu med Google eller Facebook sine innloggingssider med felt for brukernavn/epost/telefonnummer og passord (Google, 2021).

6.13. Create React App

Create React App er et enkelt miljø å starte i om man er ny i ReactJS og det er den beste måten å starte å lage en ny Single-page Application (React, 2021). Det Create React App gjør er å sette opp et utviklingsmiljø der vi hadde mulighet til å bruke de nyeste JavaScript egenskapene, samt at den optimaliserer appen for produksjon. Create React App håndterer ikke noe backend, istedenfor lager det et oppsett for frontend-et av applikasjonen ved en enkel kommando i terminalen, som er «npx create-react-app my-app» (React, 2021).

6.14. Service Workers

Ved å bruke Create React App blir det også automatisk brukt service workers. En service worker er et script som blir kjørt i bakgrunn, uavhengig av en webside. Og det blir gjerne brukt til oppgaver som ikke trenger en webside eller bruker interaksjon. Eksempler på arbeidsoppgaver kan være push notifikasjoner og bakgrunn synkronisering (Gaunt, 2020).

6.15. Firebase som en tjeneste

Firebase er en Backend-as-a-service plattform som ga oss tjenester som realtime database, skylagring og autentiseringsmuligheter ved blant annet Facebook og Google (Esplin, 2016). Ved å bruke Firebase hosting ble det enklere for oss som utviklere å få infrastruktur og oversikt over appen. Fordelene med dette var blant annet data som var lastet opp i Firebase ble alltid

hentet kjapt og bruk av autentisering og innlogging via Facebook og Google, som var påkrevd av arbeidsgiver, ble enklere.

7. Resultatet

Bachelorprosjektet har gitt prosjektgruppen et innblikk i hvordan det er å jobbe som full-stack utviklere i et start-up selskap. Videre, har den faglige kompetansen gruppen allerede har tilegnet seg blitt forsterket og brukt i praksis i utviklingsprosessen. Eksempelvis, har tilbakemeldingene fra arbeidsgiver og de ukentlige møtene gitt oss innsikt til hva som kunne forbedres og hva som var av tilfredsstillende kvalitet. I tillegg, ga arbeidsgiver forslag til hvilke nye arbeidsoppgaver gruppen burde fokusere på. Dette ga prosjektgruppen erfaringer med Scrum-metodikk, sprinter, brukertesting, prototyping og planlegging. Følgelig, har erfaringer fra dette blitt benyttet under utviklingsprosessen og vil være nyttig erfaring å ta med seg videre.

Resultatet av bachelorprosjektet ga arbeidsgiver et godt utgangspunkt å bygge videre på. Bakgrunnen for dette var at arbeidsgiver ønsket å få en mer moderne vri på deres tidligere versjon av spillapplikasjonen. Ved å benytte de rammeverkene og bibliotekene fastsatt av arbeidsgiver, i tillegg til egne valg fra prosjektgruppen, har det blitt et sammensatt resultat av arbeidsgivers ønsker og visjon, og prosjektgruppens arbeid med dette.

Resultatet av funksjonaliteten til spillapplikasjonen, har vært et system som tillater flere autentiseringsmetoder, muligheten til å spille og få en score basert på gjeldende spill. Utover dette, gir spillapplikasjonen brukeren en økt som er isolert med vedvarende innlogging. Spillapplikasjonen gir også brukeren muligheten til å tilbakestille passord om dette er glemt. Dermed, ut fra funksjonaliteten som er implementert gir dette arbeidsgiver et utgangspunkt som har utviklingspotensiale. Resultatmessig, ble arbeidsgiver imponert over hva produktgruppen hadde fått til under utviklingsprosessen og kunne se for

seg hvordan de skulle fortsette å videreutvikle spillapplikasjonen (Vedlegg 9: Attest fra Lars Føleide (Galea AS)).

8. Konklusjon

Først og fremst, med tanke på at prosjektgruppen hadde felles mål og interesser for prosjektet har dette ført til et samarbeid alle på gruppen kunne ta nytte av. Av den grunn har det vært hensiktsmessig for gruppen å samarbeide på en intuitiv måte. Dette innebærer at planlegging, testing og arbeidsmetodikk har ligget i kjernen av samarbeidet. Videre, kan man bemerke at det er vanskelig å forutsi hvordan et samarbeid vil fungere, i tillegg er det krevende å fordele arbeidsoppgaver på en slik måte at det blir rettferdig når gruppen består av fire medlemmer. Det har derfor vært viktig med planlegging i forkant av hver sprint, slik at arbeidsoppgaver blir fordelt ut ifra kompetansen hvert gruppemedlem besitter.

For det andre, har utviklingen av prosjektet bestått av utviklingssteg som enkelte på gruppen er bedre tilpasset for enn andre. Ettersom at gruppen har gått ulike studieretninger, Anvendt Datateknologi og Informasjonsteknologi, har gruppen ikke bare lært av prosjektet, men også av hverandre. Videre, med tanke på at gruppen hadde lite kunnskap om de gitte rammeverkene og bibliotekene arbeidsgiver hadde utvalgt, startet samtlige på gruppen med et likt utgangspunkt. Dette førte til at store deler av prosjektet har vært tidskrevende og det har dermed vært en trinnvis læringskurve. Dermed måtte gruppen kombinere de faglige emnene som hadde blitt lært under studietiden for å nå et endelig resultat. Dette inkluderer, men er ikke begrenset til programutvikling, systemutvikling, menneske-maskin-interaksjon, databaser, webprogrammering og testing av programvare.

Avslutningsvis, er spillapplikasjon som er utviklet ikke ferdigstilt og vil fremover bli videre utviklet av Galea AS. Dette innebærer videreutvikling av de ulike nivåene som gjenstår, profilside som gir brukeren tilgang til tidligere spilldata,

muligheten til å endre passord i profilsiden, utfordre venner til spill og sammenlikne spilldata med venner eller andre spillere. Slik spillapplikasjonen er nå, når det delvis målet om være underholdende og lærerikt, men er avhengig av flere nivåer for å skape en varierende og underholdende spillopplevelse for brukeren. Planen videre for spillapplikasjonen er at det skal videreutvikles på en slik måte at det kan tjene sitt mål om å være et underholdende og lærerikt alternativ for å tilegne seg kunnskap om finans og aksjemarkedet.

Referanseliste

- Bhargava Mithra Konda, K. K. (2010). *A Systematic Mapping Study on Software Reuse.* Ronneby: Blekinge Institute of Technology.
- Cummings, N. (2021, Januar 1). *Build an app with React, Redux and Firestore from scratch*. Hentet fra Udemy: https://www.udemy.com/course/build-an-app-with-react-redux-and-firestore-from-scratch/
- Esplin, C. (2016, Oktober 24). *What is Firebase?* Hentet fra Medium: https://howtofirebase.com/what-is-firebase-fcb8614ba442
- Flavio Copes. (2018, November 11). *Flavio Copes*. Hentet fra What is a Single Page Application?: https://flaviocopes.com/single-page-application/
- freeCodeCamp. (2016, April 5). *Beginner's Guide to React Router*. Hentet fra freeCodeCamp: https://www.freecodecamp.org/news/beginner-s-guide-to-react-router-53094349669/
- Gaunt, M. (2020, Juli 24). *Service Workers: An Introduction*. Hentet fra Google - WebFundamentals: https://developers.google.com/web/fundamentals/primers/service-workers
- git. (u.d.). *gitignore*. Hentet fra git: https://git-scm.com/docs/gitignore
- Google. (2021, Mai 20). *Cloud Firestore*. Hentet fra Firebase: https://firebase.google.com/docs/firestore
- Google. (2021, Mai 20). *Cloud Firestore Data model*. Hentet fra Firebase: https://firebase.google.com/docs/firestore/data-model
- Google. (2021, Mai 11). *Firebase Authentication*. Hentet fra Firebase: https://firebase.google.com/docs/auth
- Google. (2021, Mai 20). *Get started with Cloud Firestore Security Rules* . Hentet fra Firebase: https://firebase.google.com/docs/firestore/security/get-started
- Hreniak, K. (2019, Mars 21). *Basic examples of using Cloud Firestore Security Rules*. Hentet fra Medium: https://khreniak.medium.com/cloud-firestore-security-rules-basics-fac6b6bea18e
- Imperva. (u.d.). *Lazy Loading*. Hentet fra Imperva: https://www.imperva.com/learn/performance/lazy-loading/
- Jayathilaka, H. (2018, Desember 18). Firebase: Developing a Web Service with Admin SDK, Flask and Google Cloud. Jayathilaka, Hiranya. Hentet fra https://medium.com/google-cloud/firebase-developing-a-web-service-with-admin-sdk-flask-and-google-cloud-6fb97eb38b80
- Mozilla. (2021, Mai 12). *Making asynchronous programming easier with async and await*. Hentet fra Mozilla: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Async_await
- Nes, S. M. (2019, Mai 16). *En kort introduksjon til Scrum*. Hentet fra Visma: https://www.visma.no/blogg/en-kort-introduksjon-til-scrum/

- npm. (u.d.). *React Transition Handler*. Hentet fra npm: https://www.npmjs.com/package/react-transition-handler
- Oloruntoba, S. (2020, Desember 17). *SOLID: The First 5 Principles of Object Oriented Design*. Hentet fra DigitalOcean: https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design
- Ravichandran, A. (2018, Desember 3). *React Virtual DOM Explained in Simple English*. Hentet fra programmingwithmosh: https://programmingwithmosh.com/react/react-virtual-dom-explained/
- React. (2019, August 6). *Thinking in React*. Hentet fra ReactJS: https://reactjs.org/docs/thinking-in-react.html
- React. (2019, Februar 7). *Virtual DOM and Internals*. Hentet fra ReactJS: https://reactjs.org/docs/faq-internals.html
- React. (2019, Februar 7). *Virtual DOM and Internals*. Hentet fra ReactJS: https://reactjs.org/docs/faq-internals.html
- React. (2021, Februar 14). *Create a New React App*. Hentet fra ReactJS: https://reactjs.org/docs/create-a-new-react-app.html
- React. (2021, Mai 1). *Quick Start*. Hentet fra React Router: https://reactrouter.com/web/guides/quick-start
- React Router. (u.d.). *Philosophy* . Hentet fra React Router: https://reactrouter.com/web/guides/philosophy
- Redux. (u.d.). *Getting Started with Redux* . Hentet fra Redux: https://redux.js.org/introduction/getting-started
- Redux. (u.d.). *Redux Fundamentals, Part 3: State, Actions, and Reducers*. Hentet fra Redux: https://redux.js.org/tutorials/fundamentals/part-3-state-actions-reducers
- Singh, K. P. (2020, Oktober 30). *Private, Public and Restricted routes in React*. Hentet fra DEV Community: https://dev.to/karanpratapsingh/private-public-and-restricted-routes-in-react-42ff
- Udemy. (2021, Mai 12). *Udemy*. Hentet fra https://www.udemy.com/
- Węglarz, R. (2020, 11 20). *9 Companies that Use React*. Hentet fra Droptica: https://www.droptica.com/blog/9-companies-use-react/

Figurliste

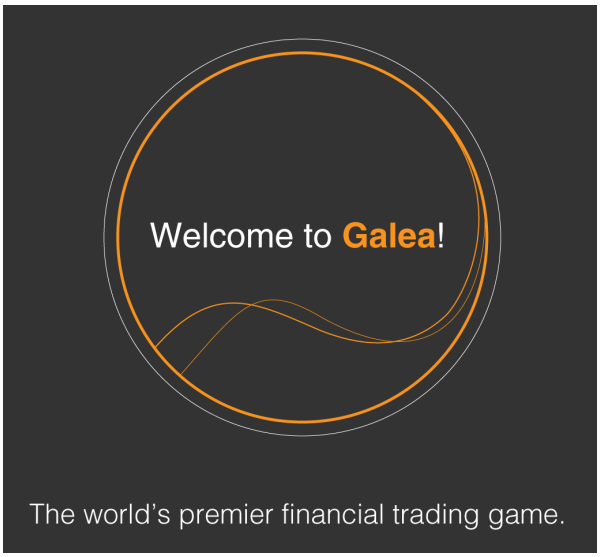
Vedlegg 2: Forprosjekt (Kravspesifikasjon)

Galea Games – Finansspill

Benjamin A. S. Jørgensen
Magomed M. Khatsjukajev
Felix S. Leypoldt
Sigurd Ø. Bjørndal

13, Januar, 2021

Original utgave:
Forprosjekt.pdf



1 Presentasjon

1.1 Oppdragsgiver

Våres oppdragsgiver er følgende:
Bedrift: Galea AS
Adresse: Klæbuveien 52, H0208 7030 Trondheim

1.2 Ekstern mentor

Meeyad Shabab
E-post: mmshabab90@gmail.com
Tlf: +4746364590

1.3 Kontaktperson

Våres prosjektgiver, veileder og kontaktperson for dette prosjektet er:
Lars Føleide, CEO
Tlf: +47 98 45 44 99
E-post: Lars@Galea.com
Adresse: Klæbuveien 52, H0208 7030 Trondheim

1.4 Intern veileder

Tildelt veileder fra OsloMet
Henrik Lieng
E-post: Henrik.Lieng@oslomet.no
Tlf: +47 67 23 87 58

1.5 Gruppemedlemmer

Benjamin A. S. Jørgensen s333933@oslomet.no +47 94817017
Magomed M. Khatsjukajev s334001@oslomet.no +47 46249540
Sigurd Ø. Bjørndal s333741@oslomet.no +47 99338126
Felix S. Leypoldt s333743@oslomet.no +47 95098473

1.6 Oppgaven

Oppgaven går ut på å utvikle et webbasert finansspill ved bruken av eksisterende verktøy som JavaScript rammeverk. I hovedsak React, som utviklingsgrunnlag til denne applikasjonen. Spillet skal bli utviklet slik at live data fra eksisterende børsmarked, som NASDAQ-børsen blir hentet ut til applikasjonen. Hovedfokus innledningsvis bør være NDX (NASDAQ-100). En indeks av de 100 største selskapene på NASDAQ børsen. Utover dette kan det være aktuelt å bruke prisdata for APPL (Apple) og TSLA (Tesla), og andre indekser som Dow Jones og SP500.

2 Sammendrag

Fokuset til Galea Games er å hjelpe gjennomsnittspersonen til å bli mer kunnskapsrik innenfor aksjer og finans. Dette gjøres ved at brukeren lærer på en underholdende måte, i form av et spill. Spillet vil i hovedsak bli laget i React/React Native og Firebase, og vil derfor være mulig å spille på hvilken som helst plattform uavhengig av operativsystem.

3 Dagens situasjon

Lars Føleide etablerte hovedkonseptet i 2012 under navnet Galea Games. Aksjeselskapet Galea AS ble opprettet sommeren 2020. Man kan lese mer på nettsiden deres http://www.galea.com. Galea har bestemt fra begynnelsen av våres utviklingsprosess at de ønsker en webapplikasjon. I tillegg til dette vil de at vi skal kunne benytte React Native for å tilgjengeliggjøre det på smarttelefoner også. Oppretterne av selskapet har tidligere startet Enry AS. Mens Enry videreutvikles, så allokeres også ressurser til å veilede våres gruppe til å utvikle et fungerende finansspill. Galea AS er fortsatt under oppstart, slik at deres bakhistorie er begrenset.

4 Mål for applikasjon

- Bruker skal ha mulighet til å logge inn med støttede Firebase autentiseringsmetoder
- Oversikt over egen profil.
- Rangering/nivåinndeling basert på prestasjon.
- Oversikt over venners prestasjoner
- Implementere funksjonaliteten til å spille med live datasett uthentet fra NASDAQ.

5 Rammebetingelser

- Benytte React for webutviklingen av applikasjonen og deretter porte dette til React Native.
- Benytte React Native for at applikasjonen kan brukes på flere plattformer
- Gruppemedlemmene skal bruke Scrum for daglige møter og Kanban for utvikling av applikasjonen.
- Applikasjon vil lastes opp og dokumenteres på Github.
- Applikasjon, utvikling og dokumentasjon skal være ferdigstilt 24.05.2021.
- Sluttpresentasjonen skal være klar uke 22-23.

Front-end teknologi som skal benyttes:

- **React**: React er et JavaScript bibliotek for å lage/bygge brukergrensesnitt. Det vedlikeholdes av Facebook og et fellesskap av ulike utviklere og selskaper. React kan brukes som et utgangspunkt i utviklingen av enkelt-side eller mobil applikasjon, da det er optimalisert for å raskt hente skiftende data som må registreres.
- **Redux**: Redux er et open-source JavaScript-bibliotek for å administrere applikasjonsstatus. Det brukes ofte med biblioteker som React eller Angular for å bygge brukergrensesnitt
- **Javascript**: JavaScript er et høynivå-programmeringsspråk. Sammen med HTML og CSS er det en av grunnsteinene i moderne web-utvikling, og alle moderne nettlesere kan kjøre JavaScript-programmer uten utvidelser. Språket kan brukes både til komplekse webapplikasjoner og som et enklere skriptspråk.

Back-end teknologi som skal benyttes:

- **Firebase**: Firebase is a NoSQL database that stores and syncs data in realtime (a real-time document store)

Verktøy:

- **GitHub**: GitHub er en kodevertsplattform for versjonskontroll og samarbeid. Det lar deg og andre jobbe sammen om prosjekter hvor som helst.
- **Visual Studio Code**: Visual Studio Code er en gratis kildekode-editor laget av Microsoft for Windows, Linux og macOS. Funksjonene inkluderer støtte for feilsøking, syntaksfremheving, intelligent kodefullføring, utdrag, kodefakturering og innebygd Git.
- **nTask**: En kalender/planleggingsverktøy, som blir brukt til å planlegge når ting skal gjøres.
- **CircleC**i: Verktøy som binder sammen GitHub og forenkler automatisering av tester.
- **Expo**: Verktøy som benyttes sammen med React Native som forenkler testing av applikasjonen på mobile enheter.

6 Løsninger/Alternativer

Arbeidsgiver har bestemt at applikasjonen skal utvikles i React/React Native som en webapplikasjon slik at den støtter flere plattformer. Videre, har Galea AS foreslått at vi skal bruke Firebase for å kunne hente ut data fra aksjer i sanntid. Arbeidsmetodikken velger vi selv som en gruppe og har kommet fram

til å benytte Scrum for daglige møter og oppdateringer. Kanban for utvikling av funksjoner.

Scrum - er et rammeverk for arbeidsmetodikk hvor man leverer korte iterasjoner i utviklingsforløpet. Scrum er hovedsakelig små selvstyrte lag. Vi utvikler i kort sikt, vanligvis med en fast lengde på en måneds tid. Fokuset blir å levere den viktigste funksjonaliteten først og deretter supplementerende funksjonaliteter i etterkant. Selv om Scrum er bygd opp på en fast prosedyre, vil vi gi rom for fleksibilitet til prosjektets utvikling.

Kanban - er et annet rammeverk for arbeidsmetodikk hvor man ikke starter på nytt arbeid før tidligere arbeid er regnet som ferdigstilt. Det vil gjelde pågående arbeid, slik som funksjonalitet som ikke er ferdig.

7 Analyse av virkninger

Vi tenker å ha daglige standup møter for oppdatering på framgangsprosessen hvert enkelt individ har. Dette innebærer korte møter på rundt 10-15 minutter for å gi en innføring på hva de har gjort forrige dagen og hva de skal gjøre i den påbegynte dagen. Slike møter vil også innebære utfordringer og hvordan man tenker å løse disse. Eventuelt spørringer om hjelp fra andre og hvordan en skal handtere visse utfordringer. Under Scrum sprintene tenker vi å ha et Kanban Board som gir en ekstra form for oversikt over hva som skal gjøres under sprintene. Ut ifra resultatet fra sammenlikningen har vi bestemt at vi skal ha en sammenslåing av Scrum og Kanban.

Tabell 1 - Skjematisk visning av fordeler og ulemper ved valgte arbeidsmetoder

Arbeids-metode	Fordeler	Ulemper
Scrum	• Ved Scrum blir tiden brukt effektivt, ved blant annet korte Scrum møter hver dag og sprinter. • Det er også lettere å endre på faktorer som ikke fungerer så bra underveis ved sprinter.	• Sjansen for at prosjektet feiles er høy om personene ikke er veldig samarbeidsvillige eller motiverte. • Kan oppstå tilfeller hvor Scrum fører til ukontrollert vekst i et prosjekt. Dette oppstår når omfanget av et prosjekt ikke er riktig definert, dokumentert eller kontrollert. • Scrum-rammeverket er krevende i større team og rammeverket kan bare lykkes med erfarne teammedlemmer.
Kanban	• God fleksibilitet og med fokus på kontinuerlig levering. • Betraktelig reduksjon av bortkastet arbeid /bortkastet tid. Dette fører til økt produktivitet / effektivitet. Dette er med på å øke teammedlemmenes evne til å fokusere.	• Ingen sprinter i Kanban slik det er i Scrum, derfor viktig at man har et godt forhold til alle de ulike tidsfristene. • Ved Kanban er det viktig at man ikke overkompliserer tavlen, da den skal være lett å tyde og skape en enkel oversikt over gjøremålene.

Tabell 2 - Skjematisk visning av fordeler og ulemper ved mulige utviklingsverktøy (Frontend)

Arbeids-metode	Fordeler	Ulemper
React/React-Native	• React Native sine moduler kommuniserer direkte med iOS, Android og andre enheter spesialtilpasser etter hvilke enheter man benytter. • Det finnes innebygget debugging i verktøyet for feilretting • React Native bruker ReactJS som JavaScript-bibliotek, så det har alle fordelene. For å opprette en React Native-app på tvers av plattformer, trenger ikke utviklere å kjenne språket til den opprinnelige plattformen.	• Reloading-issues • Inkompatible funksjoner gjennom libraries og forskjellige versjoner av React Native • Emulator problemer • Problemer med React-navigasjon • Må ofte re-installere pakker (packages) • Andre feil
Bootstrap	• Færre cross-browser feil • Et jevnt rammeverk som støtter store deler av alle nettlesere og CSS-kompabilitetsrettinger • Responsive strukturer og stiler • Flere JavaScript-plugins ved hjelp av jQuery • God dokumentasjon og samfunnstøtte • Massevis av gratis og profesjonelle maler, WordPress-temaer og plugins.	• Kan ende opp med å bruke mye tid på å tilpasse design, hard-koding. • Videre tilpasning av nettsiden er tuklete og tidkrevende • Stilene er omfattende og kan føre til mye HTML utdata som ikke er nødvendig • JavaScript er knyttet til jQuery og er ett av de vanligste bibliotekene, dette etterlater fleste plugins ubrukt • Ikke optimalt for wen + mobileenheter på lik måte som React/React-Native

Tabell 3 - Skjematisk visning av fordeler og ulemper ved mulige utviklingsverktøy (Backend)

Arbeids-metode	Fordeler	Ulemper
Firebase	• Firebase er eksplisitt laget for mobilapplikasjoner, som betyr at mange av funksjonene og brukergrensesnittet er for dette bruksområdet • Firebase er både en database og utviklingsverktøy som inkluderer hosting, autentisering og analysering	• Bruker JSON-filer som er vanskelig å migrere til tradisjonelle SQL-databaser. • Manglende filtreringsegenskaper for live-databaser. • Bruker JSON for lagring av hele databasen, som hindrer at man kan introdusere relasjoner mellom forskjellige datafiler.
Mongo DB	• Fullverdig databasesystem med funksjonalitet som hosting automatiserte operasjoner og autentisering • MongoDB er utviklet for enkel oppskalering • Enkel Query-syntax.	• Slik som Firebase, bruker MongoDB JSON-filer som er vanskelig å migrere til tradisjonelle SQL-databaser. • Mangler funksjoner for analyse av data. • Overholder ikke kravene for ACID (atomicity, consistency, isolation, durability).

Vedlegg 3: Brukertester

Disse kriteriene blir benyttet for å vurdere hva som må prioriteres ved utviklingen av applikasjonen. Denne fargekodingen er benyttet på notatene fra brukertestene under.

1. Kritisk	Viktig funksjonalitet som må implementeres i sluttproduktet.
2. Moderat	Funksjonalitet som kan implementeres ved overskudd av tid. Eller generell tilbakemelding.
3. Godkjent	Godkjente funksjonalitet/håndtering
4. Ingen feil	Annet/kommentar

Brukertest 1:

Testperson 1 (20-30 år)

Hovedflyt 1: Registrere en bruker med e-post og passord og logge inn		
Brukstilfelle: Innlogging/registrering ved e-post		
Tilleggsinfo: Her må bruker følge regex-en til programmet og feilhåndteringen. Feilhåndteringen implementert er sjekk på lengde av passord (minst 8 tegn), en regex håndtering der @ er forventet i String input. Feilhåndteringen sjekker om alle felter også er utfylt.		
Brukerhistorie	**1**	Brukeren er allerede på innloggingsiden til applikasjonen. Brukeren navigerer til registreringsskjemaet for e-post.
	2	Fyller ut feltene: Display Name, Email, Password, Confirm Password for å opprette en ny bruker
	3	Trykker på knappen «Submit» og blir sendt til startsiden.
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Registrering av bruker og innlogging går fint • Brukeren skrev et for kort passord og måtte skrive inn på nytt. Forsto uten veiledning.	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 5: Logge inn med registrert bruker		
Brukstilfelle: Innlogging med registrert bruker		
Tilleggsinfo: Her må bruker logge inn med nylig registrert bruker som er lagret i databasen.		
Brukerhistorie	**1**	Brukeren trykker på «Login» for å logge inn på allerede eksisterende bruker.
	2	Fyller ut feltene «Email» og «Password»
	3	Trykker på knappen «Submit» og blir sendt til startsiden.
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Innlogging av bruker går fint • Forventet standard til innlogging	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 9: Logge inn med Facebook		
Brukstilfelle: Innlogging Facebook		
Tilleggsinfo: Her må bruker logge inn med Facebook bruker. Dersom brukeren ikke har det kan det oppstå at de må opprette en Facebook bruker. På forhånd er det forsikret at testbruker har Facebook konto.		
Brukerhistorie	**1**	Brukeren trykker på «Login with Facebook» for å logge inn med Facebook.
	2	Brukeren kommer til innlogget side
	3	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Innlogging av bruker går fint • Brukeren måtte logge inn på egen Facebook bruker fordi h*n var utlogget. Et popup vindu med Facebook autentisering poppet opp.	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 12: Logge inn med Google		
Brukstilfelle: Innlogging Google		
Tilleggsinfo: Her må bruker logge inn med Google bruker. Dersom brukeren ikke har det kan det oppstå at de må opprette en Facebook bruker. På forhånd er det forsikret at testbruker har Google konto.		
Brukerhistorie	**1**	Brukeren trykker på «Login with email» for å komme til steget med muligheten til å logge inn med Google.
	2	Trykke på «Login with Google»
	3	Brukeren kommer til innlogget side
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Brukeren forsto ikke hvor innloggingsknappen var til Google umiddelbart. Mente den burde være tilgjengelig sammen med Facebook knappen. Den er litt skjult. • Brukeren var innlogget på Google bruker og ble tatt umiddelbart til innlogget side	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Brukertest 2:

Testperson 1: (20-30 år)

Hovedflyt 1: Brukeren starter gjennomgangen av spillets gang.		
Brukstilfelle: Innføring i spillet via opplæringsmodul		
Tilleggsinfo: Her er bruker allerede innlogget og skal benytte seg av opplæringsmodul for å få en innsikt i hvordan man skal spille.		
Brukerhistorie	**1**	Brukeren trykker «View Tutorial»
	2	Brukeren observerer det som skjer på skjermen og leser instruksene. Når første steg er ferdig, trykker brukeren på «Next»
	3	Brukeren blir presentert andre steg i spillets gang. Der en bruker blir introdusert til knappene «HIGH» og «LOW» som er en sentral del av spillet. Og funksjonaliteten
	4	Brukeren blir informert om nedtelleren som viser hvor mye tid spilleren har igjen for dette spillet. Når tredje steg er ferdig, trykker brukeren på «Submit» eller trykker seg ut av det interaktive vinduet

Notater	**Fokus for testansvarlige**
• Brukeren synes opplæringsmodulen var nyttig. • Testeren var klar til å prøve spillets funksjonalitet. • Ingen spørsmål	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 5: Brukeren starter spillet		
Brukstilfelle: Spille et spill på aggregert data		
Tilleggsinfo: Her er bruker allerede innlogget og skal benytte av spillfunksjonalitet som eksisterer.		
Brukerhistorie	**1**	Etter endt innføring av spillets gang. Trykker spilleren videre på «Start Game»
	2	Brukeren gjetter når dataene/aksjen når det høyeste punktet.
	3	Brukeren gjetter når dataene/aksjen når det laveste punktet.
	4	Etter endt tid blir brukeren presentert resultatene fra spillet.

Notater	**Fokus for testansvarlige**
• Testbrukeren trykket tidlig på «HIGH» knappen og låste svaret sitt. Dette kan ikke endres når det er først gjort. • Testbrukeren synes det var et interessant konsept. • Testbrukeren synes det var kjedelig presentering av resultater. Men fortsatt forståelig.	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Brukertest 1:

Testperson 2 (20-30 år)

Hovedflyt 1: Registrere en bruker med e-post og passord og logge inn		
Brukstilfelle: Innlogging/registrering ved e-post		
Tilleggsinfo: Her må bruker følge regex-en til programmet og feilhåndteringen. Feilhåndteringen implementert er sjekk på lengde av passord (minst 8 tegn), en regex håndtering der @ er forventet i String input. Feilhåndteringen sjekker om alle felter også er utfylt.		
Brukerhistorie	**1**	Brukeren er allerede på innloggingsiden til applikasjonen. Brukeren navigerer til registreringsskjemaet for e-post.
	2	Fyller ut feltene: Display Name, Email, Password, Confirm Password for å opprette en ny bruker
	3	Trykker på knappen «Submit» og blir sendt til startsiden.
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Registrering av bruker og innlogging går fint • Testbrukeren skrev inn forskjellige passord. Feilhåndteringen plukket det opp og ble informert med feilmelding.	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 5: Logge inn med registrert bruker		
Brukstilfelle: Innlogging med registrert bruker		
Tilleggsinfo: Her må bruker logge inn med nylig registrert bruker som er lagret i databasen.		
Brukerhistorie	**1**	Brukeren trykker på «Login» for å logge inn på allerede eksisterende bruker.
	2	Fyller ut feltene «Email» og «Password»
	3	Trykker på knappen «Submit» og blir sendt til startsiden.
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Testbrukeren hadde glemt passordet til brukeren • Heldigvis for denne testbrukeren var funksjonaliteten ved tilbakestilling av passord implementert	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet
• Mottok en mail på e-post hvor h*n kunne skrive inn et nytt passord.	

Hovedflyt 9: Logge inn med Facebook		
Brukstilfelle: Innlogging Facebook		
Tilleggsinfo: Her må bruker logge inn med Facebook bruker. Dersom brukeren ikke har det kan det oppstå at de må opprette en Facebook bruker. På forhånd er det forsikret at testbruker har Facebook konto.		
Brukerhistorie	**1**	Brukeren trykker på «Login with Facebook» for å logge inn med Facebook.
	2	Brukeren kommer til innlogget side
	3	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Innlogging av bruker går fint	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet
• Allerede innlogget på Facebook. Testbrukeren satt pris på at han slapp å huske noe passord ved bruk av Facebook.	

Hovedflyt 12: Logge inn med Google		
Brukstilfelle: Innlogging Google		
Tilleggsinfo: Her må bruker logge inn med Google bruker. Dersom brukeren ikke har det kan det oppstå at de må opprette en Facebook bruker. På forhånd er det forsikret at testbruker har Google konto.		
Brukerhistorie	**1**	Brukeren trykker på «Login with email» for å komme til steget med muligheten til å logge inn med Google.
	2	Trykke på «Login with Google»
	3	Brukeren kommer til innlogget side
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Brukeren fant innloggingsmuligheten til Google. Men kommenterte at den ikke var like tilgjengelig som Facebook. • Brukeren var innlogget på Google bruker og ble tatt umiddelbart til innlogget side	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Brukertest 2:

Testperson 2 (20-30 år)

Hovedflyt 1: Brukeren starter gjennomgangen av spillets gang.		
Brukstilfelle: Innføring i spillet via opplæringsmodul		
Tilleggsinfo: Her er bruker allerede innlogget og skal benytte seg av opplæringsmodul for å få en innsikt i hvordan man skal spille.		
Brukerhistorie	**1**	Brukeren trykker «View Tutorial»
	2	Brukeren observerer det som skjer på skjermen og leser instruksene. Når første steg er ferdig, trykker brukeren på «Next»
	3	Brukeren blir presentert andre steg i spillets gang. Der en bruker blir introdusert til knappene «HIGH» og «LOW» som er en sentral del av spillet. Og funksjonaliteten
	4	Brukeren blir informert om nedtelleren som viser hvor mye tid spilleren har igjen for dette spillet. Når tredje steg er ferdig, trykker brukeren på «Submit» eller trykker seg ut av det interaktive vinduet

Notater	**Fokus for testansvarlige**
• Hadde kjennskap til aksjer og hvordan det fungerte. Svippet fort gjennom gjennomgangen av spillet. • Hadde vært nyttig med premarked-data for denne aksjen og informasjon om hvilken aksje det er. • Ingen problemer med å forstå hva h*n skulle gjøre	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 5: Brukeren starter spillet		
Brukstilfelle: Spille et spill på aggregert data		
Tilleggsinfo: Her er bruker allerede innlogget og skal benytte av spillfunksjonalitet som eksisterer.		
Brukerhistorie	**1**	Etter endt innføring av spillets gang. Trykker spilleren videre på «Start Game»
	2	Brukeren gjetter når dataene/aksjen når det høyeste punktet.
	3	Brukeren gjetter når dataene/aksjen når det laveste punktet.
	4	Etter endt tid blir brukeren presentert resultatene fra spillet.

Notater	**Fokus for testansvarlige**
• Testbrukeren trykket rett under det høyeste punktet til aksjen ved å trykke på «HIGH» knappen og låste svaret sitt. • Gleder til å seg til å se videreutviklingen av spillet. • Testbrukeren synes det var oversiktlig brukergrensesnitt • Ville hatt HIGH og LOW knappene i en egen boks. Ikke over og under der aksjene blir presentert.	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Brukertest 1:

Testperson 3 (30-40 år)

Hovedflyt 1: Registrere en bruker med e-post og passord og logge inn		
Brukstilfelle: Innlogging/registrering ved e-post		
Tilleggsinfo: Her må bruker følge regex-en til programmet og feilhåndteringen. Feilhåndteringen implementert er sjekk på lengde av passord (minst 8 tegn), en regex håndtering der @ er forventet i String input. Feilhåndteringen sjekker om alle felter også er utfylt.		
Brukerhistorie	**1**	Brukeren er allerede på innloggingsiden til applikasjonen. Brukeren navigerer til registreringsskjemaet for e-post.
	2	Fyller ut feltene: Display Name, Email, Password, Confirm Password for å opprette en ny bruker
	3	Trykker på knappen «Submit» og blir sendt til startsiden.
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Registrering av bruker er litt problematisk.	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet
• Testbrukeren registrerte feil e-post adresse. Ble likevel innlogget. Kunne ikke endre e-post adresse tilknyttet brukeren. Mulig å implementere «Confirm E-mail» før brukeren blir opprettet.	
• Fikk opprettet bruker på riktig e-post addresse til slutt	

Hovedflyt 5: Logge inn med registrert bruker		
Brukstilfelle: Innlogging med registrert bruker		
Tilleggsinfo: Her må bruker logge inn med nylig registrert bruker som er lagret i databasen.		
Brukerhistorie	**1**	Brukeren trykker på «Login» for å logge inn på allerede eksisterende bruker.
	2	Fyller ut feltene «Email» og «Password»
	3	Trykker på knappen «Submit» og blir sendt til startsiden.
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Innlogging til applikasjon var uten problemer	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 9: Logge inn med Facebook		
Brukstilfelle: Innlogging Facebook		
Tilleggsinfo: Her må bruker logge inn med Facebook bruker. Dersom brukeren ikke har det kan det oppstå at de må opprette en Facebook bruker. På forhånd er det forsikret at testbruker har Facebook konto.		
Brukerhistorie	**1**	Brukeren trykker på «Login with Facebook» for å logge inn med Facebook.
	2	Brukeren kommer til innlogget side
	3	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Innlogging av bruker går fint • Veldig simpel innloggingsmetode satt h*n pris på	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 12: Logge inn med Google		
Brukstilfelle: Innlogging Google		
Tilleggsinfo: Her må bruker logge inn med Google bruker. Dersom brukeren ikke har det kan det oppstå at de må opprette en Facebook bruker. På forhånd er det forsikret at testbruker har Google konto.		
Brukerhistorie	**1**	Brukeren trykker på «Login with email» for å komme til steget med muligheten til å logge inn med Google.
	2	Trykke på «Login with Google»
	3	Brukeren kommer til innlogget side
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Brukeren fant innloggingsmuligheten til Google. Ingen kommentar.	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet
• Brukeren husket ikke passordet til Google-brukeren sin som ikke var innlogget.	
• Fikk tilbakestilt passordet til Google-brukeren og logget inn til applikasjonen	

Brukertest 2:

Testperson 3 (30-40 år)

Hovedflyt 1: Brukeren starter gjennomgangen av spillets gang.		
Brukstilfelle: Innføring i spillet via opplæringsmodul		
Tilleggsinfo: Her er bruker allerede innlogget og skal benytte seg av opplæringsmodul for å få en innsikt i hvordan man skal spille.		
Brukerhistorie	**1**	Brukeren trykker «View Tutorial»
	2	Brukeren observerer det som skjer på skjermen og leser instruksene. Når første steg er ferdig, trykker brukeren på «Next»
	3	Brukeren blir presentert andre steg i spillets gang. Der en bruker blir introdusert til knappene «HIGH» og «LOW» som er en sentral del av spillet. Og funksjonaliteten
	4	Brukeren blir informert om nedtelleren som viser hvor mye tid spilleren har igjen for dette spillet. Når tredje steg er ferdig, trykker brukeren på «Submit» eller trykker seg ut av det interaktive vinduet

Notater	**Fokus for testansvarlige**
• Lite kjennskap til aksjer. • Forsto hensikten ved spillet etter oppfølgingsmodul	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 5: Brukeren starter spillet		
Brukstilfelle: Spille et spill på aggregert data		
Tilleggsinfo: Her er bruker allerede innlogget og skal benytte av spillfunksjonalitet som eksisterer.		
Brukerhistorie	**1**	Etter endt innføring av spillets gang. Trykker spilleren videre på «Start Game»
	2	Brukeren gjetter når dataene/aksjen når det høyeste punktet.
	3	Brukeren gjetter når dataene/aksjen når det laveste punktet.
	4	Etter endt tid blir brukeren presentert resultatene fra spillet.

Notater	**Fokus for testansvarlige**
• Synes knappene var lette å bruke, ikke så veldig komplisert spill. • Synes det var fint man kunne se hva man hadde valgt som høyeste og laveste punkt • Simpelt brukergrensesnitt • Morsomt med konfetti når resultatene kom opp.	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Brukertest 1:

Testperson 4 (20-30 år)

Hovedflyt 1: Registrere en bruker med e-post og passord og logge inn		
Brukstilfelle: Innlogging/registrering ved e-post		
Tilleggsinfo: Her må bruker følge regex-en til programmet og feilhåndteringen. Feilhåndteringen implementert er sjekk på lengde av passord (minst 8 tegn), en regex håndtering der @ er forventet i String input. Feilhåndteringen sjekker om alle felter også er utfylt.		
Brukerhistorie	**1**	Brukeren er allerede på innloggingsiden til applikasjonen. Brukeren navigerer til registreringsskjemaet for e-post.
	2	Fyller ut feltene: Display Name, Email, Password, Confirm Password for å opprette en ny bruker
	3	Trykker på knappen «Submit» og blir sendt til startsiden.
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Registrering av bruker var simpelt • Ingen problemer	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 5: Logge inn med registrert bruker		
Brukstilfelle: Innlogging med registrert bruker		
Tilleggsinfo: Her må bruker logge inn med nylig registrert bruker som er lagret i databasen.		
Brukerhistorie	**1**	Brukeren trykker på «Login» for å logge inn på allerede eksisterende bruker.
	2	Fyller ut feltene «Email» og «Password»
	3	Trykker på knappen «Submit» og blir sendt til startsiden.
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Innlogging til applikasjon var uten problemer	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 9: Logge inn med Facebook		
Brukstilfelle: Innlogging Facebook		
Tilleggsinfo: Her må bruker logge inn med Facebook bruker. Dersom brukeren ikke har det kan det oppstå at de må opprette en Facebook bruker. På forhånd er det forsikret at testbruker har Facebook konto.		
Brukerhistorie	**1**	Brukeren trykker på «Login with Facebook» for å logge inn med Facebook.
	2	Brukeren kommer til innlogget side
	3	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Innlogging av bruker går fint	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet
• Bra med mange innloggingsalternativer, fordi det er ikke alle som gidder å registrere seg over alt og huske på passord.	

Hovedflyt 12: Logge inn med Google		
Brukstilfelle: Innlogging Google		
Tilleggsinfo: Her må bruker logge inn med Google bruker. Dersom brukeren ikke har det kan det oppstå at de må opprette en Facebook bruker. På forhånd er det forsikret at testbruker har Google konto.		
Brukerhistorie	**1**	Brukeren trykker på «Login with email» for å komme til steget med muligheten til å logge inn med Google.
	2	Trykke på «Login with Google»
	3	Brukeren kommer til innlogget side
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Ingen problemer	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Brukertest 2:

Testperson 4 (20-30 år)

Hovedflyt 1: Brukeren starter gjennomgangen av spillets gang.		
Brukstilfelle: Innføring i spillet via opplæringsmodul		
Tilleggsinfo: Her er bruker allerede innlogget og skal benytte seg av opplæringsmodul for å få en innsikt i hvordan man skal spille.		
Brukerhistorie	**1**	Brukeren trykker «View Tutorial»
	2	Brukeren observerer det som skjer på skjermen og leser instruksene. Når første steg er ferdig, trykker brukeren på «Next»
	3	Brukeren blir presentert andre steg i spillets gang. Der en bruker blir introdusert til knappene «HIGH» og «LOW» som er en sentral del av spillet. Og funksjonaliteten
	4	Brukeren blir informert om nedtelleren som viser hvor mye tid spilleren har igjen for dette spillet. Når tredje steg er ferdig, trykker brukeren på «Submit» eller trykker seg ut av det interaktive vinduet

Notater	**Fokus for testansvarlige**
• Moderat kjennskap til aksjer og aksjehandel • Det var hensiktsmessig å gå gjennom de sentrale funksjonene til spillet. Enkelt og greit oppsett.	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 5: Brukeren starter spillet		
Brukstilfelle: Spille et spill på aggregert data		
Tilleggsinfo: Her er bruker allerede innlogget og skal benytte av spillfunksjonalitet som eksisterer.		
Brukerhistorie	**1**	Etter endt innføring av spillets gang. Trykker spilleren videre på «Start Game»
	2	Brukeren gjetter når dataene/aksjen når det høyeste punktet.
	3	Brukeren gjetter når dataene/aksjen når det laveste punktet.
	4	Etter endt tid blir brukeren presentert resultatene fra spillet.

Notater	**Fokus for testansvarlige**
• Rød representerer laveste punkt (LOW), Grønn representerer høyeste punkt (HIGH) • Fint med horisontale linjer i farger som viser på grafen hva du har valgt av LOW og HIGH • Simpelt brukergrensesnitt • Kunne hatt mer interaksjon i spillet	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Brukertest 1:

Testperson 5 (40-50 år)

Hovedflyt 1: Registrere en bruker med e-post og passord og logge inn		
Brukstilfelle: Innlogging/registrering ved e-post		
Tilleggsinfo: Her må bruker følge regex-en til programmet og feilhåndteringen. Feilhåndteringen implementert er sjekk på lengde av passord (minst 8 tegn), en regex håndtering der @ er forventet i String input. Feilhåndteringen sjekker om alle felter også er utfylt.		
Brukerhistorie	**1**	Brukeren er allerede på innloggingsiden til applikasjonen. Brukeren navigerer til registreringsskjemaet for e-post.
	2	Fyller ut feltene: Display Name, Email, Password, Confirm Password for å opprette en ny bruker
	3	Trykker på knappen «Submit» og blir sendt til startsiden.
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Skjønte ikke hva «Display Name» skulle være	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet
• Hadde glemt å fylle ut passordet sitt 2 ganger	
• Registrering av bruker gikk fint	

Hovedflyt 5: Logge inn med registrert bruker		
Brukstilfelle: Innlogging med registrert bruker		
Tilleggsinfo: Her må bruker logge inn med nylig registrert bruker som er lagret i databasen.		
Brukerhistorie	**1**	Brukeren trykker på «Login» for å logge inn på allerede eksisterende bruker.
	2	Fyller ut feltene «Email» og «Password»
	3	Trykker på knappen «Submit» og blir sendt til startsiden.
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Innlogging til applikasjon var uten problemer. • Nettleseren logget han inn automatisk når det ble lagret i Google Chrome	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 9: Logge inn med Facebook		
Brukstilfelle: Innlogging Facebook		
Tilleggsinfo: Her må bruker logge inn med Facebook bruker. Dersom brukeren ikke har det kan det oppstå at de må opprette en Facebook bruker. På forhånd er det forsikret at testbruker har Facebook konto.		
Brukerhistorie	**1**	Brukeren trykker på «Login with Facebook» for å logge inn med Facebook.
	2	Brukeren kommer til innlogget side
	3	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Innlogging av bruker går fint • Allerede innlogget på Facebook	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Hovedflyt 12: Logge inn med Google		
Brukstilfelle: Innlogging Google		
Tilleggsinfo: Her må bruker logge inn med Google bruker. Dersom brukeren ikke har det kan det oppstå at de må opprette en Facebook bruker. På forhånd er det forsikret at testbruker har Google konto.		
Brukerhistorie	**1**	Brukeren trykker på «Login with email» for å komme til steget med muligheten til å logge inn med Google.
	2	Trykke på «Login with Google»
	3	Brukeren kommer til innlogget side
	4	Trykker på «Logout» knappen nederst i navigeringmenyen til venstre

Notater	**Fokus for testansvarlige**
• Ingen problemer • Allerede innlogget på Google-brukeren	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Brukertest 2:

Testperson 5 (40-50 år)

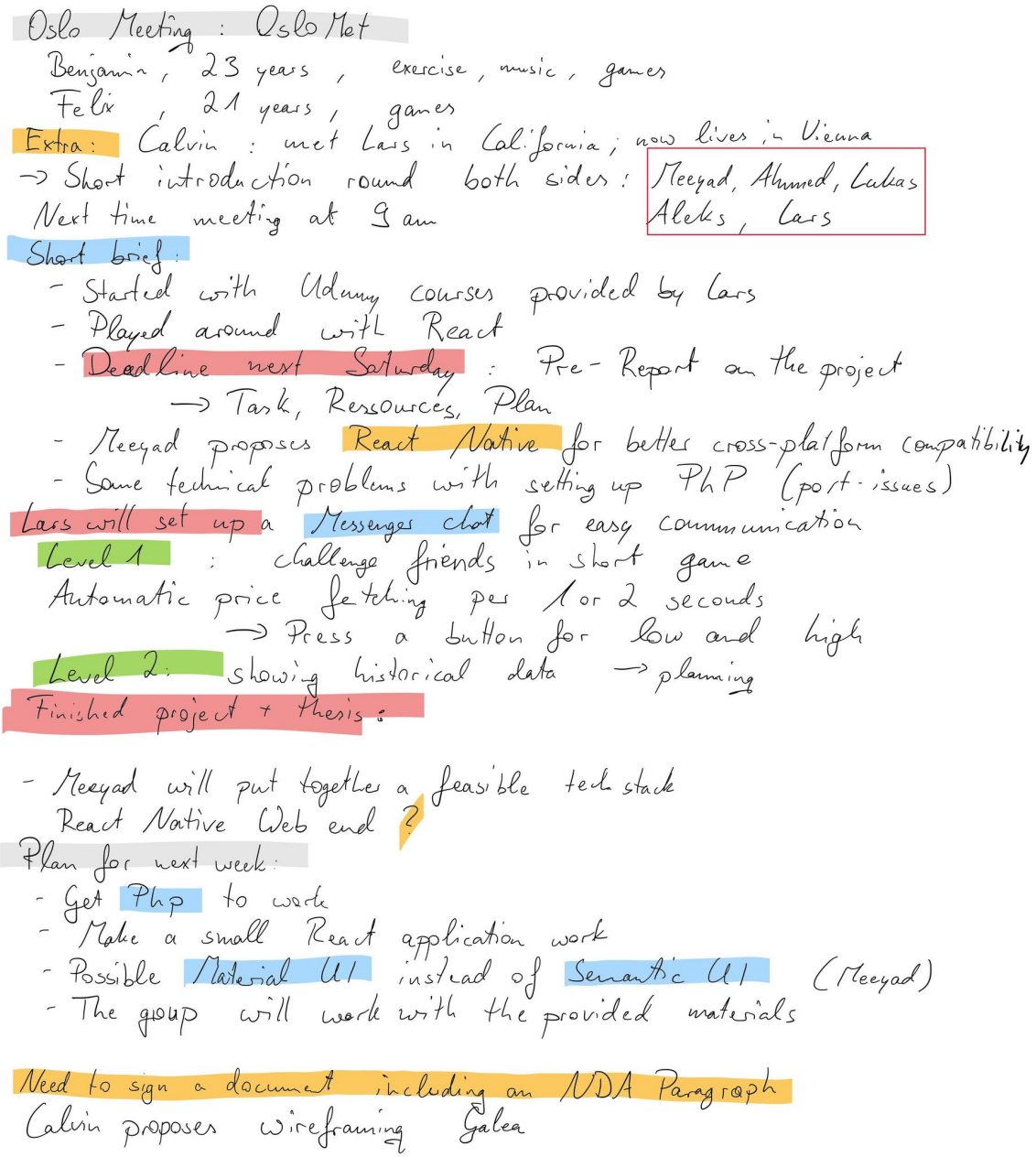
Hovedflyt 1: Brukeren starter gjennomgangen av spillets gang.		
Brukstilfelle: Innføring i spillet via opplæringsmodul		
Tilleggsinfo: Her er bruker allerede innlogget og skal benytte seg av opplæringsmodul for å få en innsikt i hvordan man skal spille.		
Brukerhistorie	**1**	Brukeren trykker «View Tutorial»
	2	Brukeren observerer det som skjer på skjermen og leser instruksene. Når første steg er ferdig, trykker brukeren på «Next»
	3	Brukeren blir presentert andre steg i spillets gang. Der en bruker blir introdusert til knappene «HIGH» og «LOW» som er en sentral del av spillet. Og funksjonaliteten
	4	Brukeren blir informert om nedtelleren som viser hvor mye tid spilleren har igjen for dette spillet. Når tredje steg er ferdig, trykker brukeren på «Submit» eller trykker seg ut av det interaktive vinduet

Notater	**Fokus for testansvarlige**
• Kjennskap til aksjer og aksjehandel • Skulle ønske det var mulighet for norsk språkvalg.	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

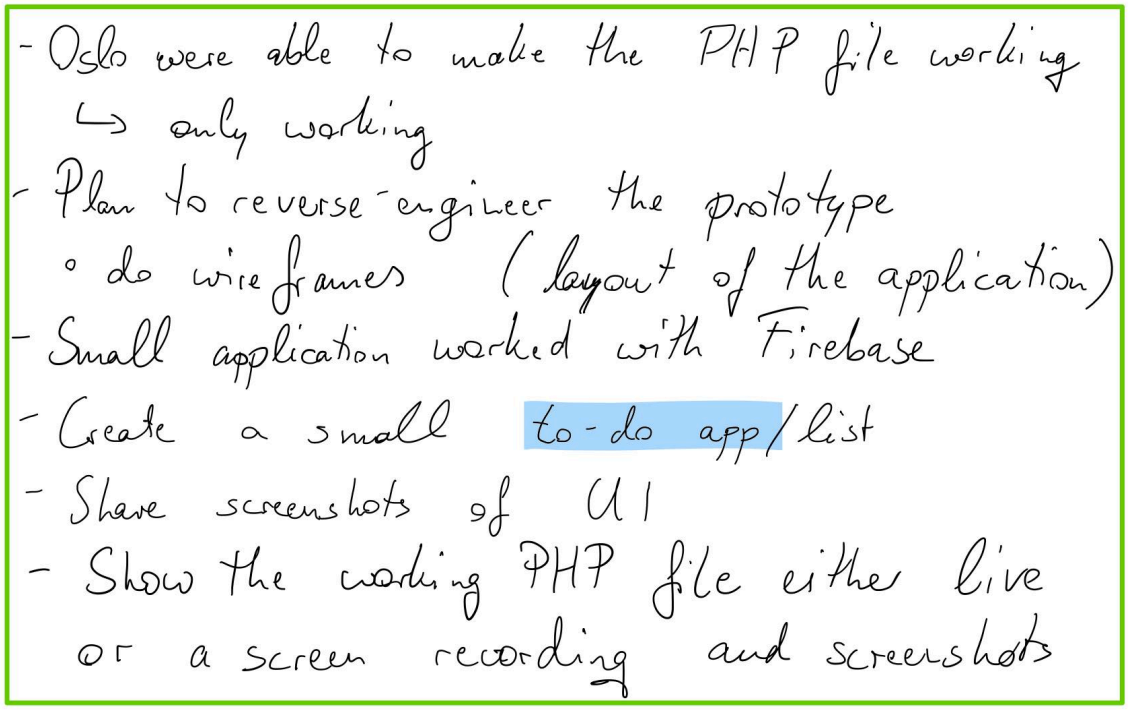
Hovedflyt 5: Brukeren starter spillet		
Brukstilfelle: Spille et spill på aggregert data		
Tilleggsinfo: Her er bruker allerede innlogget og skal benytte av spillfunksjonalitet som eksisterer.		
Brukerhistorie	**1**	Etter endt innføring av spillets gang. Trykker spilleren videre på «Start Game»
	2	Brukeren gjetter når dataene/aksjen når det høyeste punktet.
	3	Brukeren gjetter når dataene/aksjen når det laveste punktet.
	4	Etter endt tid blir brukeren presentert resultatene fra spillet.

Notater	**Fokus for testansvarlige**
• Skulle ønske aksjene ble presentert midt i synsvinkelen • Ikke like tydelig å skjønne når aksjen går opp og når den går ned. • Simpelt brukergrensesnitt • Morsomt og mye potensiale	• Misforståelser? • Usikkerhet? • Feiltolkning • Forventet resultat på funksjonalitet

Vedlegg 4: Notater fra alle møter med arbeidsgiver


Oslo Meeting : Oslo Met
Benjamin, 23 years, exercise, music, games
Felix, 21 years, games
Extra: Calvin: met Lars in California; now lives in Vienna
→ Short introduction round both sides: Meeyad, Ahmed, Lukas, Aleks, Lars
Next time meeting at 9 am
Short brief:
- Started with Udemy courses provided by Lars
- Played around with React
- Deadline next Saturday: Pre-Report on the project
→ Task, Ressources, Plan
- Meeyad proposes React Native for better cross-platform compatibility
- Some technical problems with setting up PhP (port-issues)
Lars will set up a Messenger chat for easy communication
Level 1: challenge friends in short game
Automatic price fetching per 1 or 2 seconds
→ Press a button for low and high
Level 2: showing historical data → planning
Finished project + thesis:
- Meeyad will put together a feasible tech stack
React Native Web end ?
Plan for next week:
- Get Php to work
- Make a small React application work
- Possible Material UI instead of Semantic UI (Meeyad)
- The group will work with the provided materials
Need to sign a document including an NDA Paragraph
Calvin proposes wireframing Galea

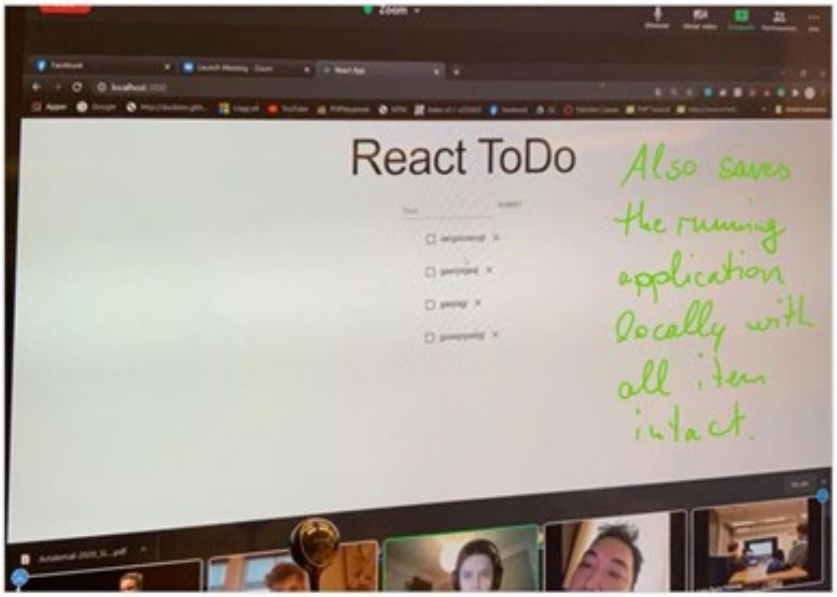
Første møte med arbeidsgiver



Andre møte med arbeidsgiver

Oslo:

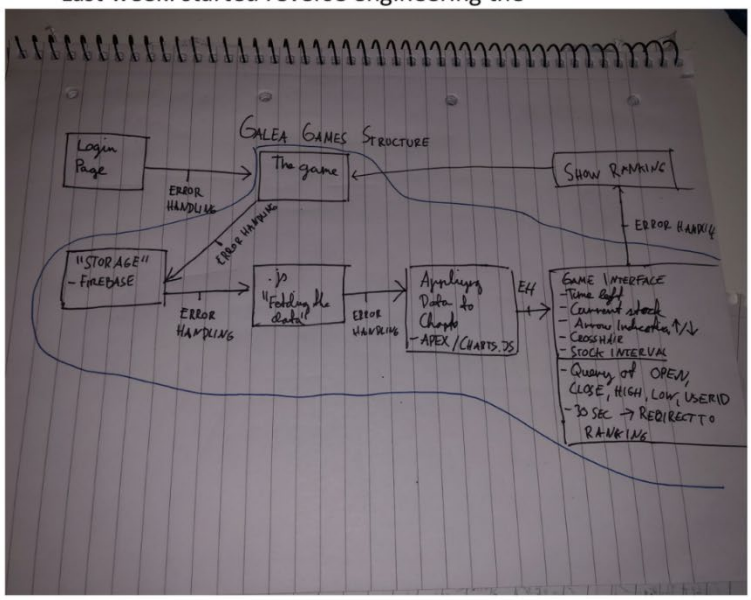
- Application is running and now has a tutorial
- Sharing screen for a closes look
- Some compiling errors along the way, but all have been resolved



Tredje møte med arbeidsgiver

Oslo Team

- Demo was based on PHP
- Last week: started reverse engineering the



- Small application showing live data from the stock market.
- ReCharts (recommended by Meeyad): intuitive library based on Chart.js
- API should be used through services (Redux/other store management) to avoid code duplications.
  - Now you load all of the data, instead of producing a feed.
- Meeyad will propose a re-design of the above picture
- Next week:
  - Trying out libraries can be continued
  - Now need to get up the wire framing — to get the design ready
  - Makes it easier to program when the wireframe is finished.
  - Easy: Pen and Paper
  - Wireframing tools: Adobe XD (student license), Balsamiq, Proto.io, Draw.io

  1. Login page
  2. Preparation page (?) for level 2
  3. PHP feature represents the actual application page
     Seperate Wireframe setups for level 1 and level 2.
     Every action needs a wireframe output.
     Level 1 is the hook (for ADHD)
     Level 5 is the ultimate end which includes preparation of participants.

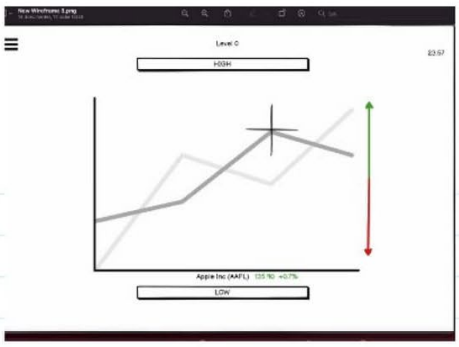
- Questions can be posed directly on the Messenger chat.

Fjerde møte med arbeidsgiver

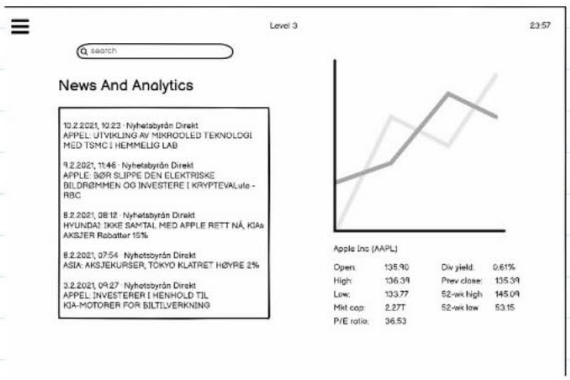
Oslo team - Galea

Wireframe made

Level 1



Level 3 - news & analysis

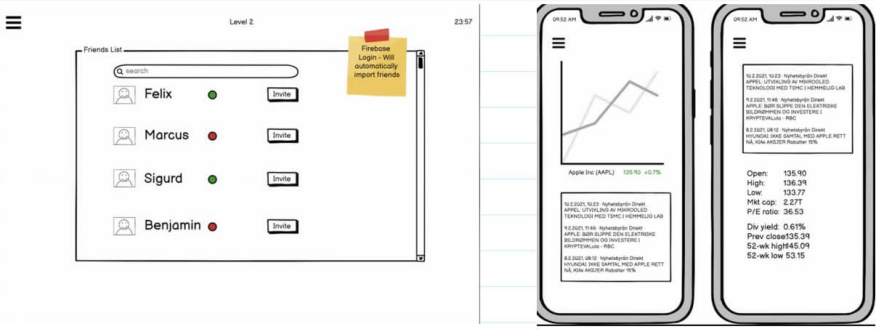


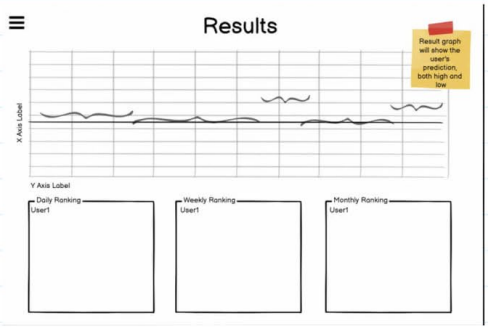
- Balance between enough prediction variables and too many
- Can start with high & low
- Introduction page proposed

- Meeyad: Need regiter button on first page & an info page for every level

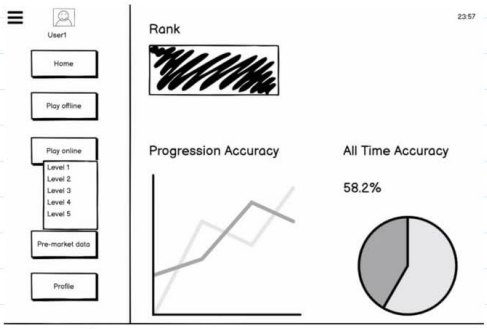
  - Static or pop up
  - Pop up instruction button

- Lars: first time - tutorial (short & sweet)
- Should have the opportunity to view the tutorial later if needed





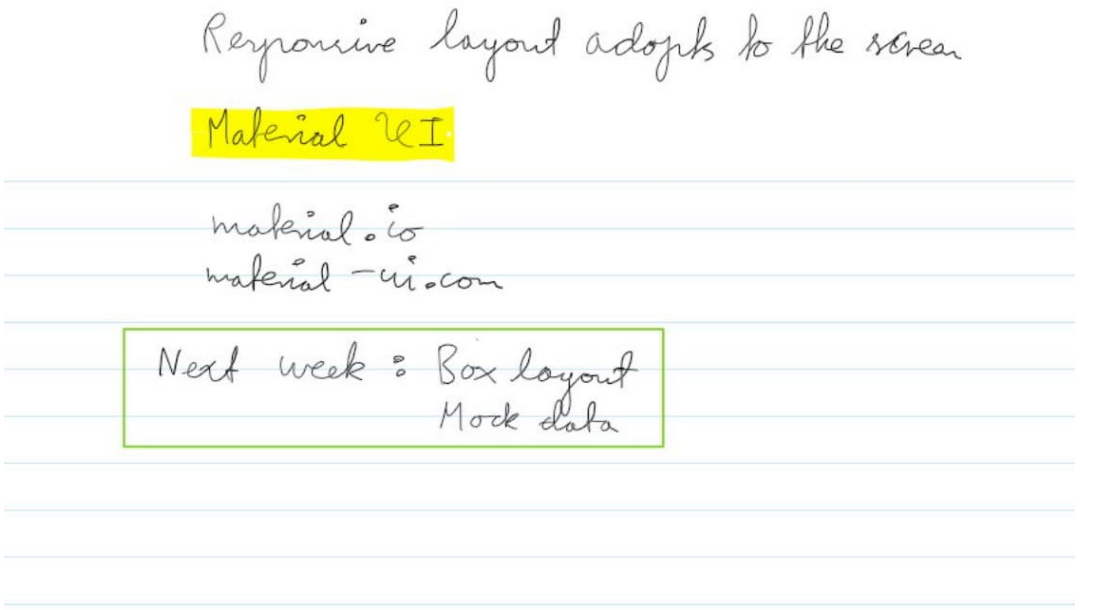
- Invite friends, can switch between global and friends → split screen



- User homepage
- Game stats

- Include real time data at level 5

Wireframe - feedback
Fix tutorial slides
Some aspects on online/offline aspects require more work
Mockaroo: create mock up structure



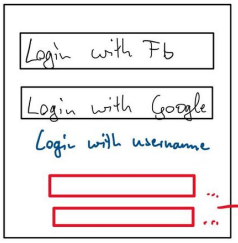
Femte møte med arbeidsgiver

Oslo:

Sigurd is reporting for the group

Oslo corona rules are still the same: closed theatres and gyms; campus opened

- Demonstration of the working login: Good shit!
- Picture of the authentification process in the chat
- Possibility to set the SoMe-login as first choice



Check Foodora login as an example

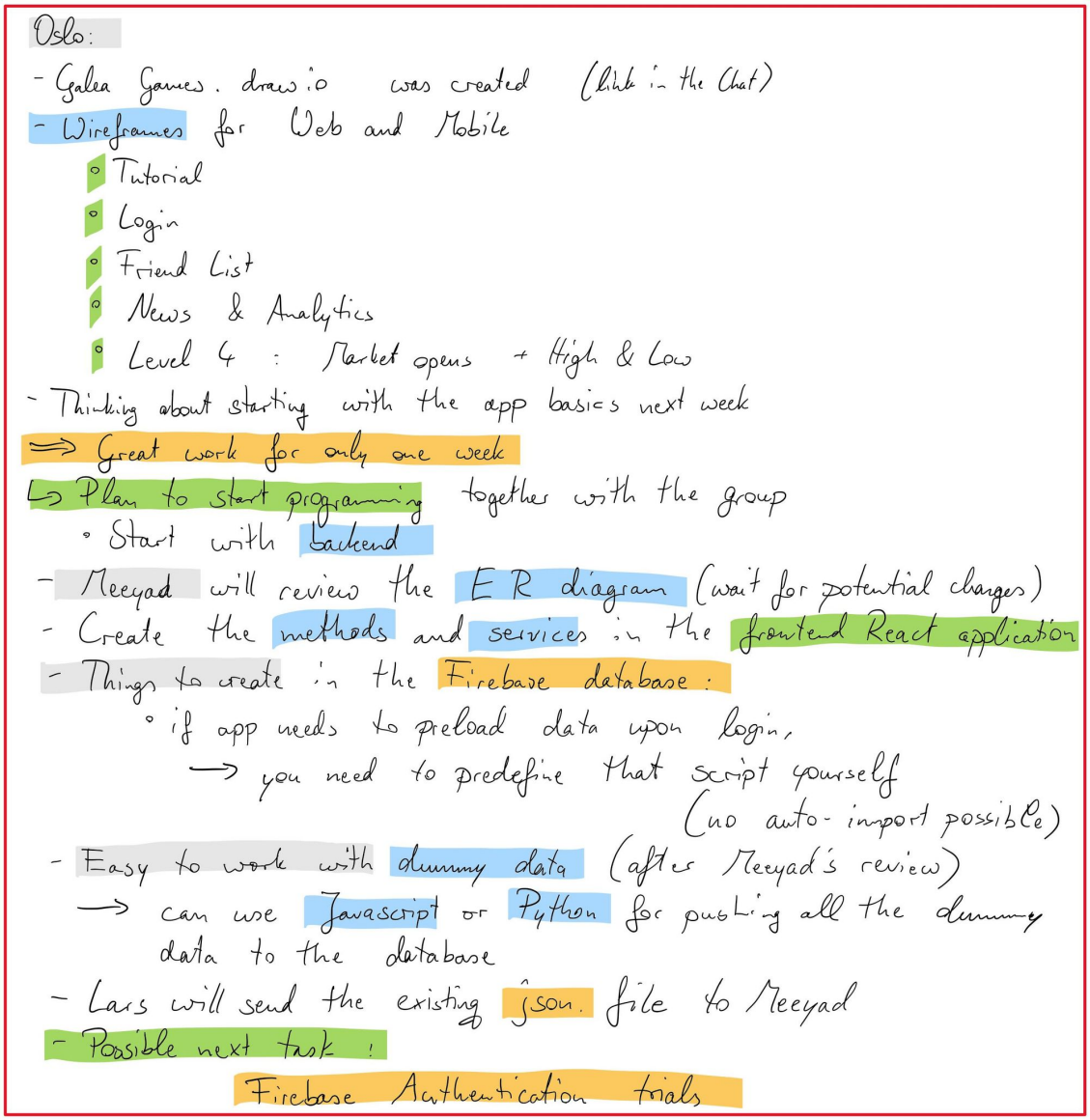
hidden at first

- What's up next?
  - Balsamiq cloud?
  - Logo discussion: Colors: old orange on dark grey background
    ↳ Calvin gets feedback
  - PHP got working: possible to recreate this in Firebase?
  - Data from the SQL should be created into a JSON file which can be integrated into a Javascript graphing tool with buttons
  - Score calculation at the end of the game

→ This is the plan. Input from Neeyad needed

  - Look into Apex Charts also

Sjette møte med arbeidsgiver



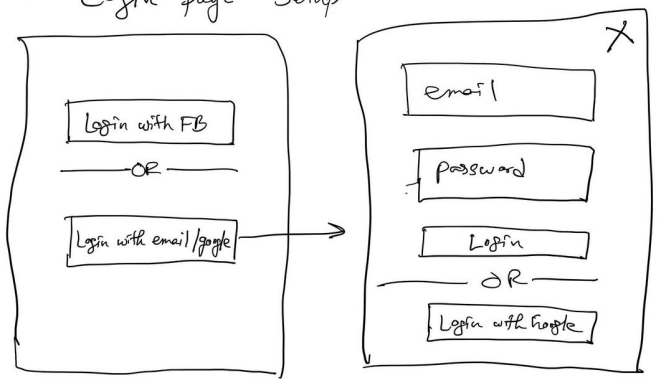
Syvende møte med arbeidsgiver

Oslo:

- Have been working on the report and the thesis itself
- Changed the login a bit
  - Buttons in the front are changed
  - not yet optimized perfectly
- Oslo lockdown is serious
- Goal with the login is to have Facebook login only!
  - ↳ People should have to click to see Google login and e-mail
  - No difference between Sign in and Sign up

→ Goal to have this ready for next week

→ Login page setup



- SQL to JSON transfer? → Needs more time
- Sprint planning focus last week
- Sprint planning spreadsheet will be shared in the chat
- User stories should be created as well
- Experience with Unity? None Lars is looking into it
- Meeyad looks at Angular and the Unreal Engine
- Facebook uses fail-fast

Åttende møte med arbeidsgiver

Oslo: Benjamin + Magomed + Sigurd

- Worked on login (a lot of work)

→ not yet quite finished as they wanted → still working on it

- Switching to routing browser

→ Looking to finish the login for next week

- The browser might not understand the .js file

→ Fix: enable routing in the browser to understand which file to display

- Planning to restructure the code to make it less messy
- There might be issues along the way, if they don't "redux" the code
- Tag Neeyad the repo
- There is a plan to solve the problems (Tutorials + Neeyads help)
- On Schedule (for now looking good)

- Main purpose of redux in this project

store management system for this application

One-stop-point:

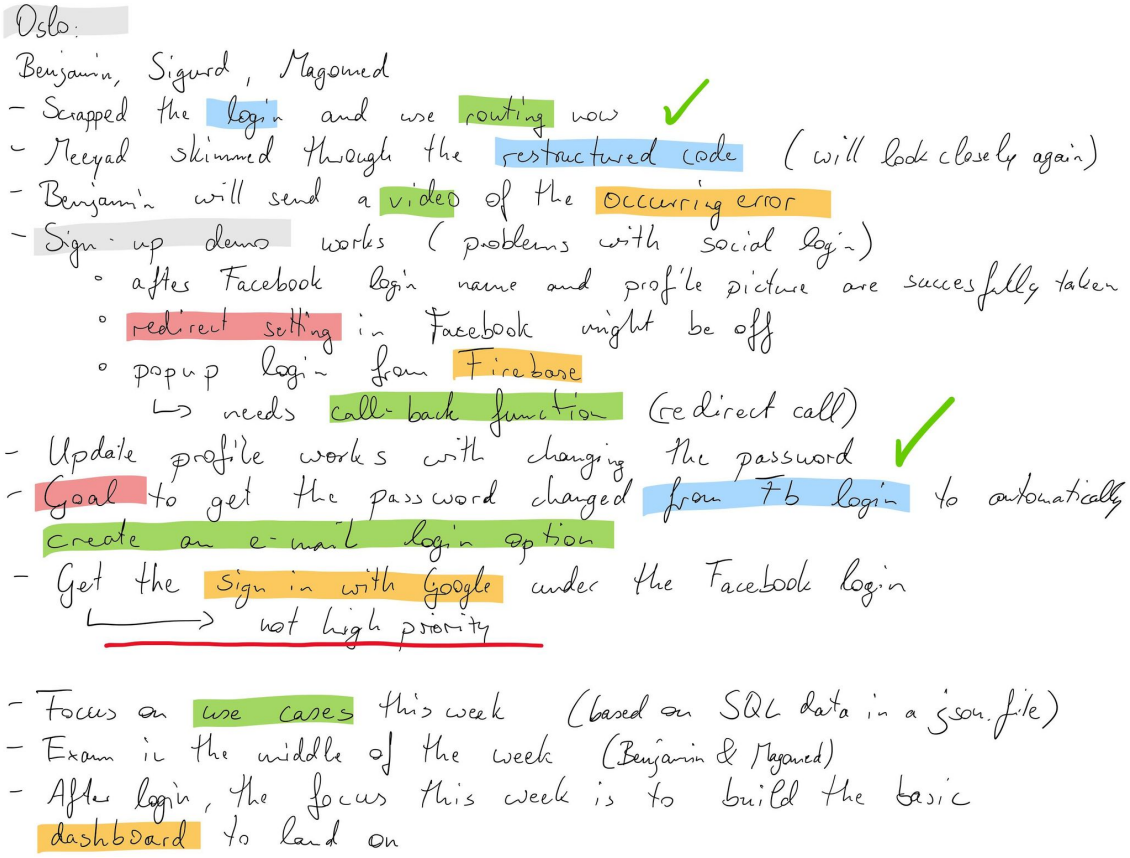
1. get data from database
2. manipulate
3. send back to database

-> Prevents code duplication → easier to scale

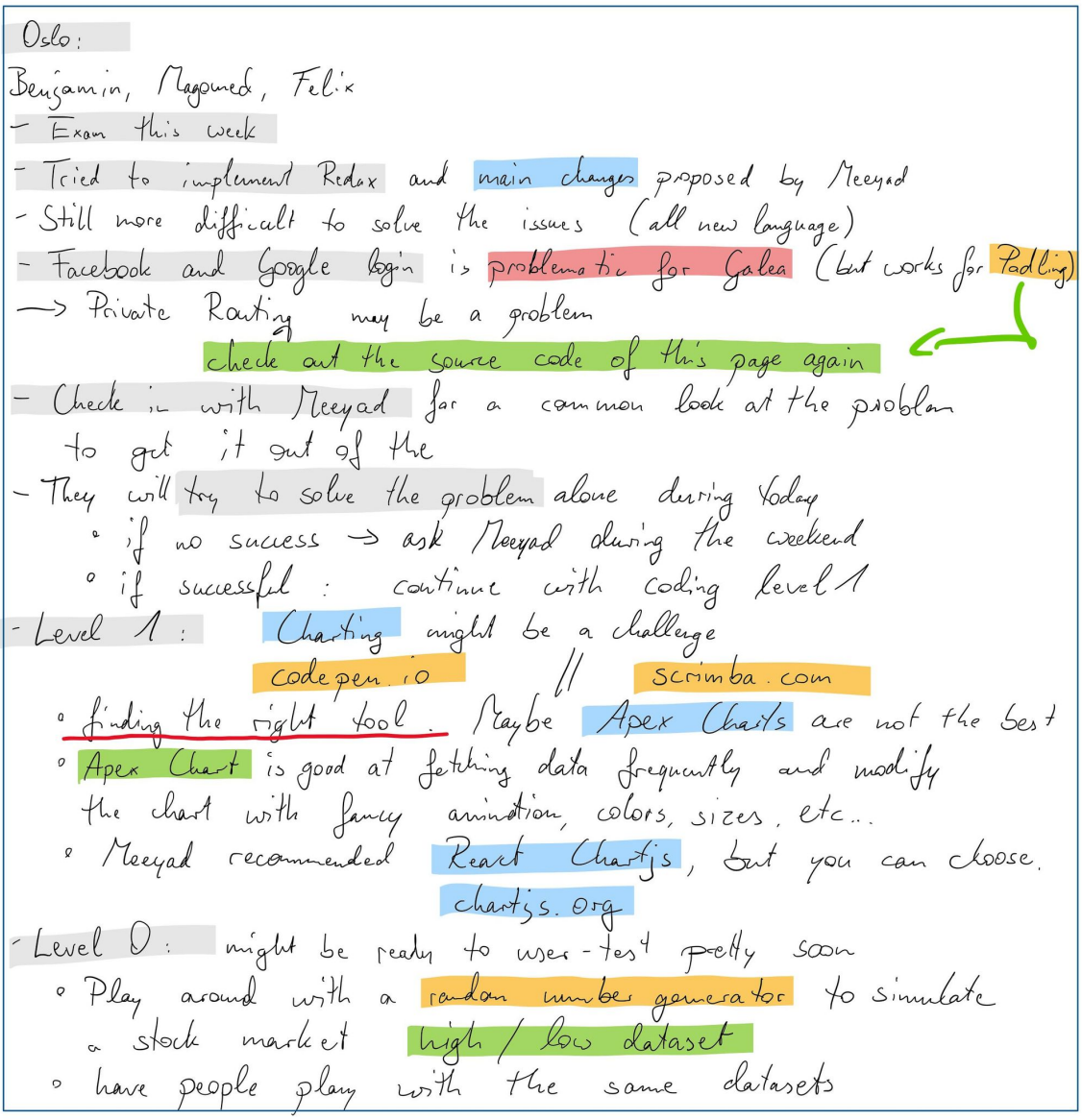
reduces resources + errors

easy to debug → reduces Javascript code

Niende møte med arbeidsgiver



Tiende møte med arbeidsgiver

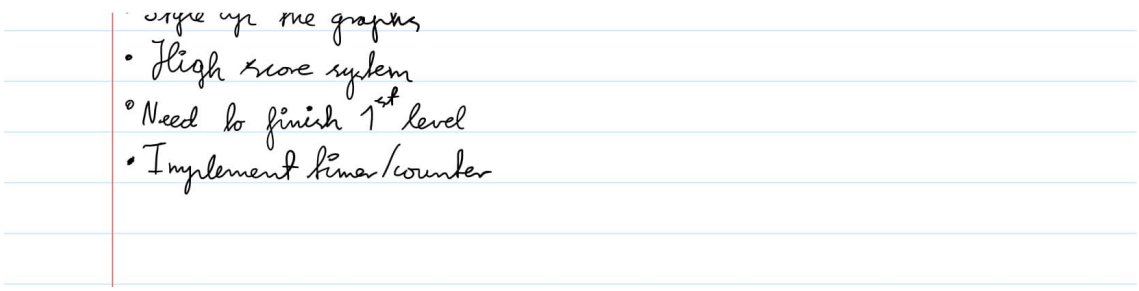


Ellevte møte med arbeidsgiver

Meeting notes 9.4

fredag 9. april 2021 08:50

- Fixed login pages with help from Meeyad
  - Facebook
  - Email
- Managed to feed in local data
  - Recharts
  - (Apex has problems with responsiveness)
  - chart.js (need to tweak some things to use)
  - Able to see all levels, but only one is accessable
  - High & low buttons @ the bottom
- Button code from material-ui.com
- Meeyad added zooming functions in the graph code
  i.e.o.t. adjust the range of x- and y-axes
- Store data in Firestore
- Will be working with improving the graph animation
- Quandl
- Array-based data pushing
- Avoid min/max functions to remove "jumps"
- Zoom needs to fit data variation to get good animations
- Candle stick graphs
- Test pushing data every minute (not ideal, every second is preferred)
- 2 sec loading limit of page
- Google Finance, CMC Markets
- Finalize use cases
- Random function => lower & upper bound
- Demonstration
- Style up the graphs
- High score system



Tolvte møte med arbeidsgiver

Benjamin, Sigurd, Felix

- Python Flask Server
- Server pushes dataen → for å unngå at man må gjøre det manuelt
- Graphen som vises kjører en snål animasjon
  → Må jobbes med animasjonen (evtl. slå av)
  → Sjekk recharts G,F
- High / Low knapp lager en horisontal linje
  grønn / rød
- Målet: level 0 må være ferdig (inkl. polish)
  vise Highscore til innlevering
  → inkl. flere datasett

→ Datasettene kan kjøres mer varient: angi range og få inn sannsynlighet for bedre forutsigbarhet

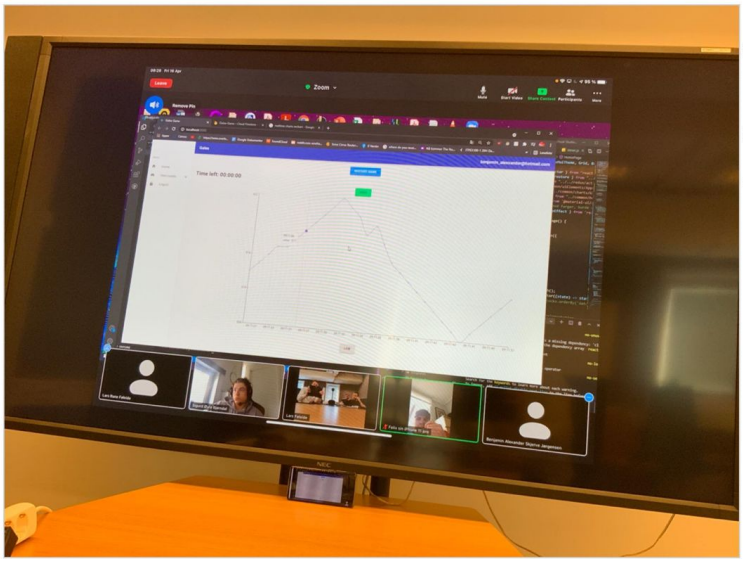
⇒ Viktig å lage ferdig spillet som kan livets rett etter innleveringen.

1. Alternativ animasjon
2. High / Low linje
3. Pushe Highscore
4. Generer ny datasett

... kanskje

Neste uke

... om det er tid



- Be Meeyad komme med innspill
- Google graf-eksempler
- Dash Plotly

Trettende møte med arbeidsgiver

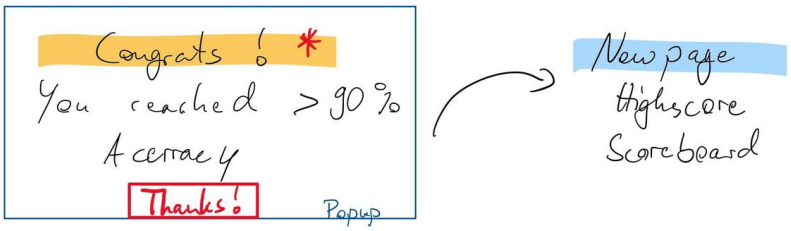
Eksempel på hvordan et intervall i et spill kan utfolde seg fra arbeidsgivers synspunkt.

Benjamin, Felix, Magomed, Sigurd

Showcasing

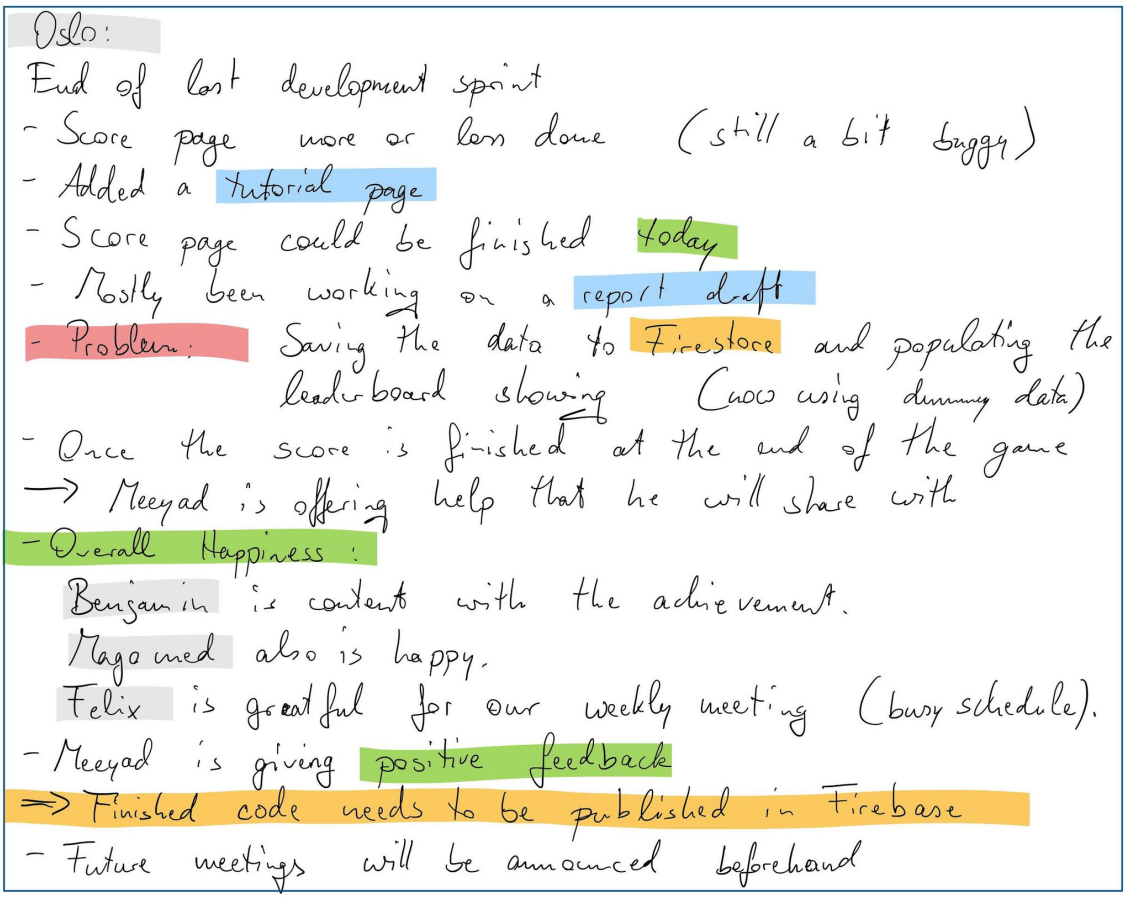
Animation improved ✓

- Percentage accuracy calculated of total values; consider Max/Min values as the absolute borders to calculate percentage off
- Colors: lighter gradient on button colors
- Show the results in a new page / or a popup



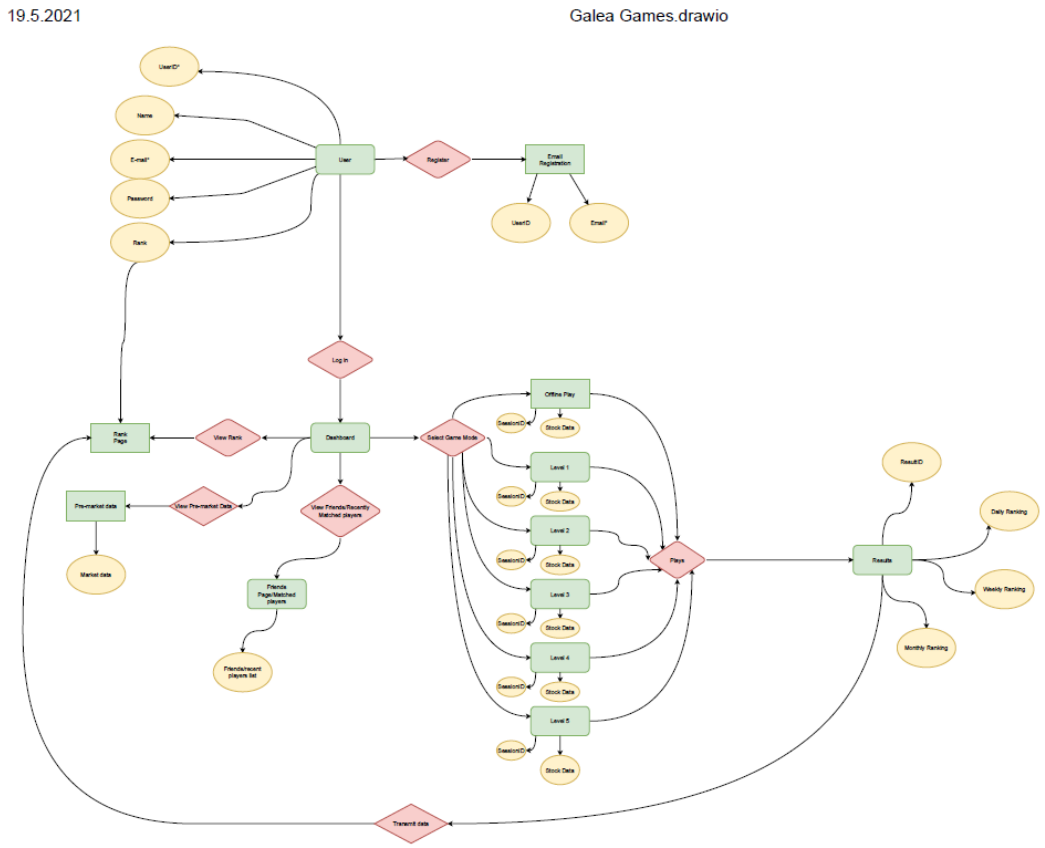
- Game end could be the trigger needed
- Save scores in matches in Firestore
  - link the documents
  - users only have access directly connected to that user
  - link user collection with score collection
- Candlestick graph should be introduced in Level 2
- Focus for this last week: Highscore (all time)
  → Think about an absolute Leaderboard (Top 10 Players)
  - Own Score is a must
- Randomize the data that shows up in each game
  rand int (upper, lower limit)
  → find a linear random number generator
- Think about a sharing-link to share the same dataset to a friend (low priority) Possible challenge a friend
- Possible to host on Firebase at the click of a button
  ⟶ React Firebase Hosting
- Lars gives access to the Envy Firebase for deployment

Fjortende møte med arbeidsgiver



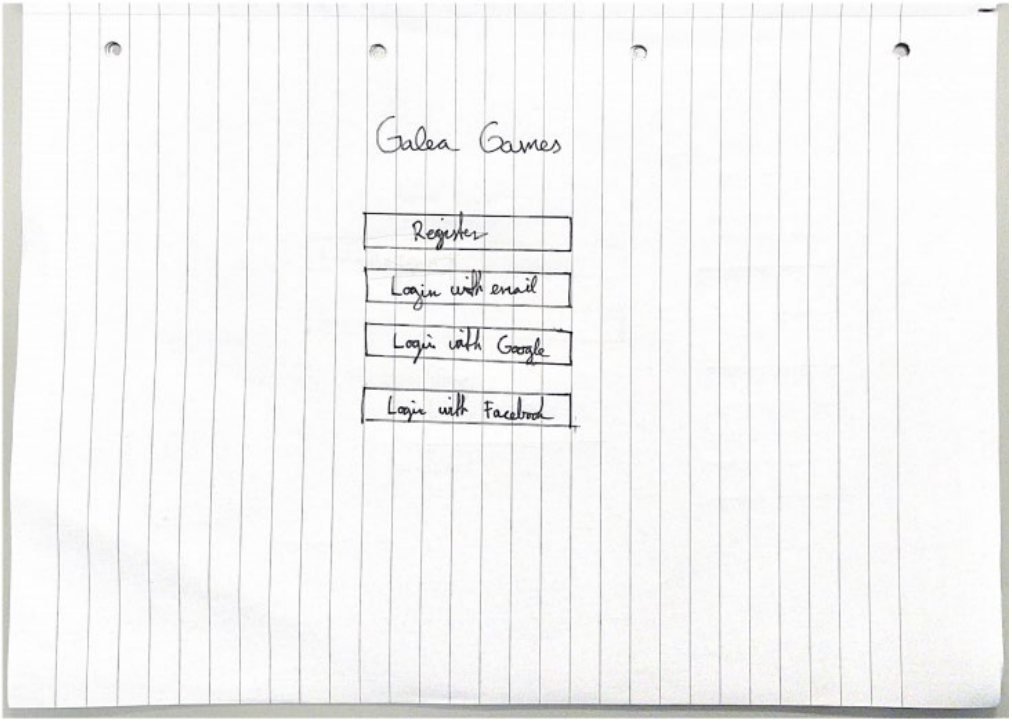
Femtende og siste møte med arbeidsgiver

Vedlegg 5: Flytskjema for applikasjon

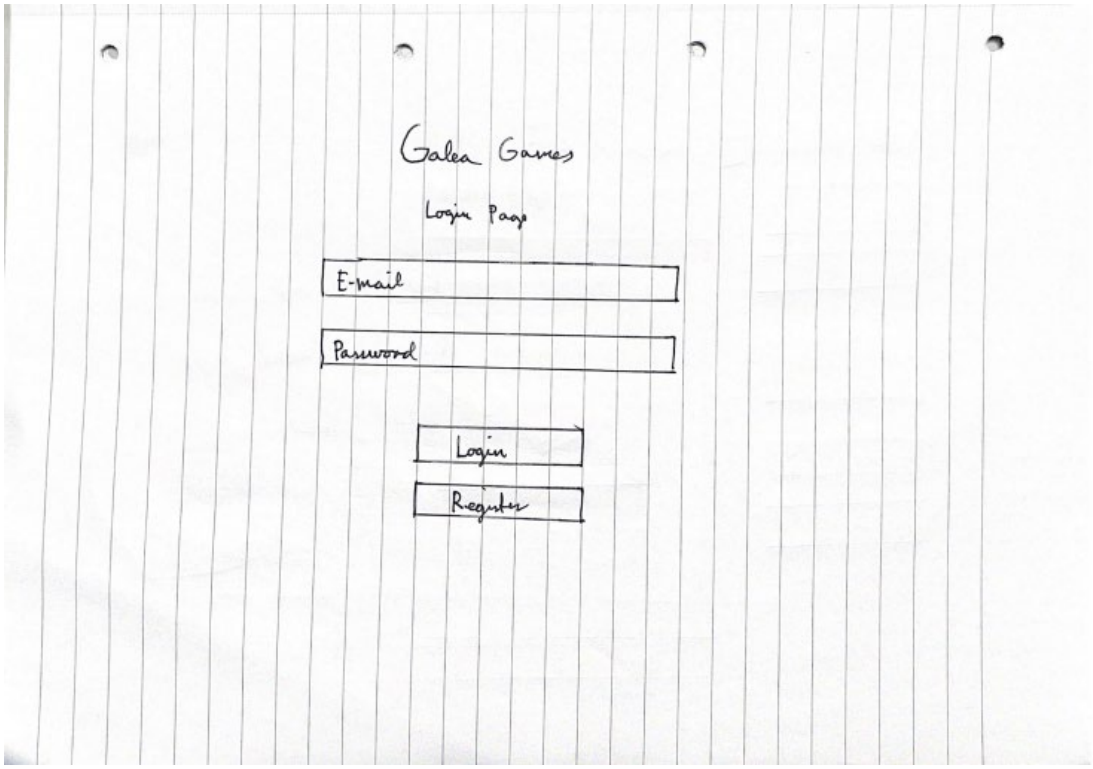


Full oppløsning: [Galea Games.drawio]

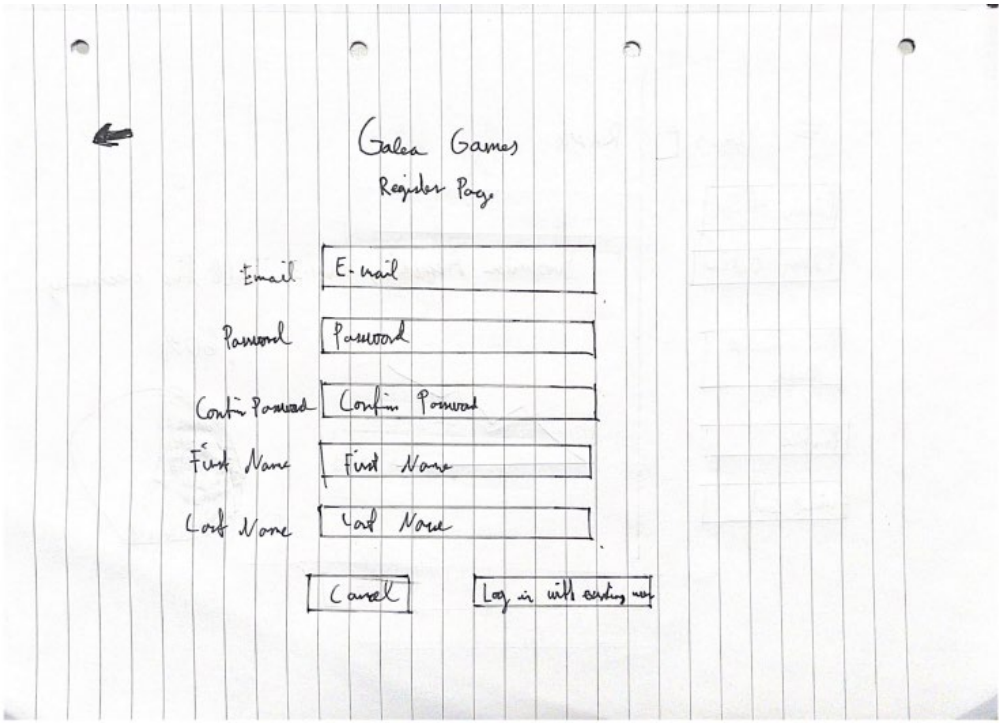
Vedlegg 6: LoFi-prototype (skisser)



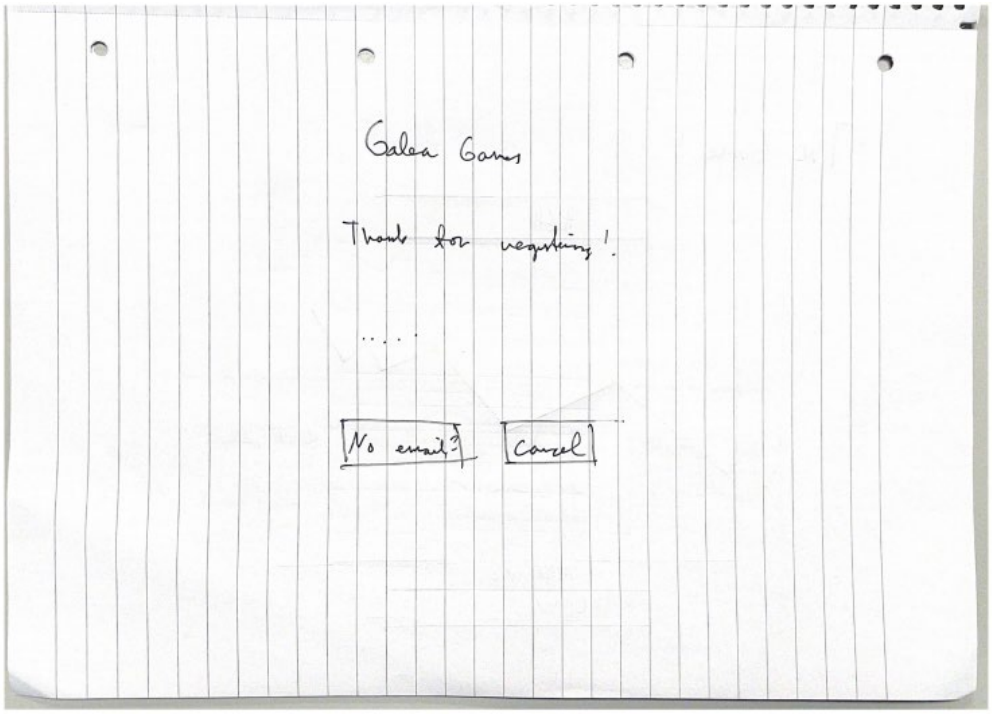
Skissering av innloggingside (Registrer, Google, E-mail, Facebook)



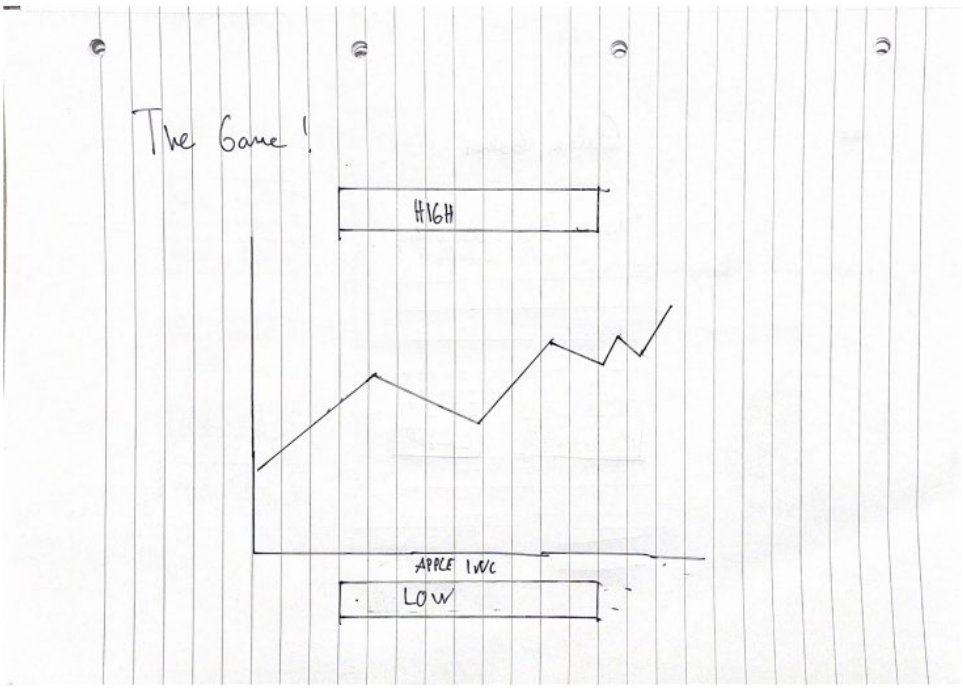
Skissering av innlogging e-mail og passord registrering



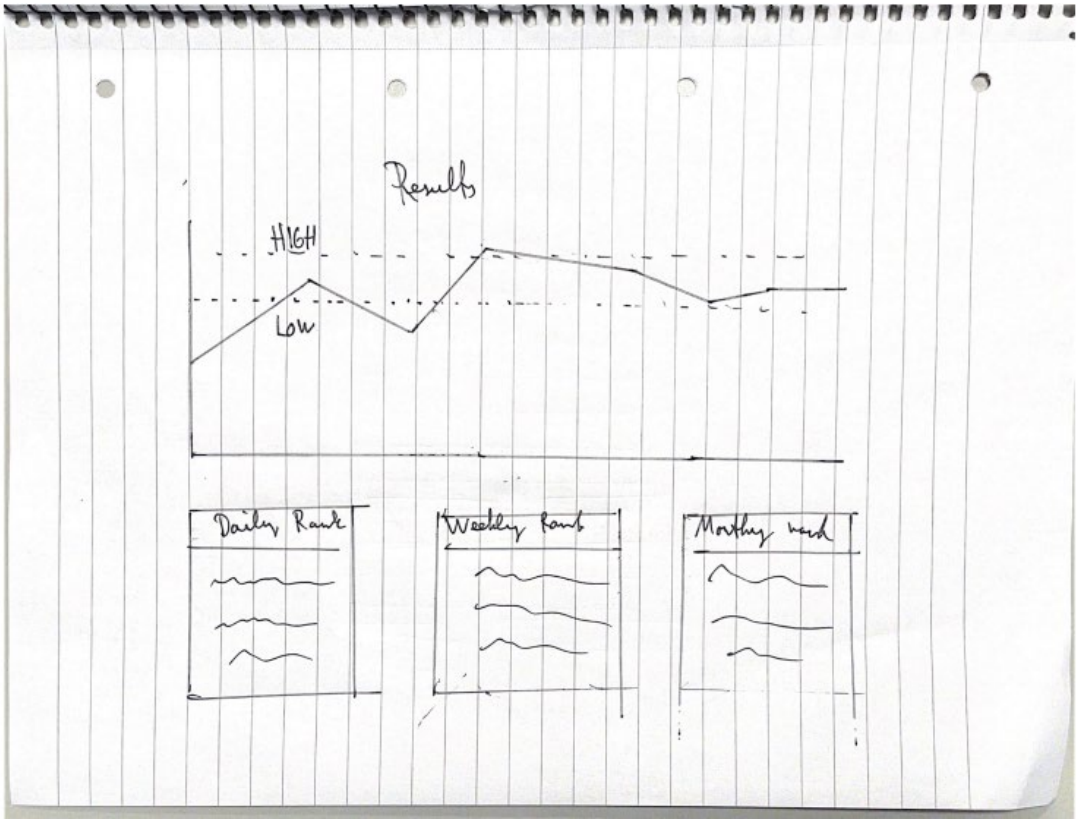
Skissering av registreringsside



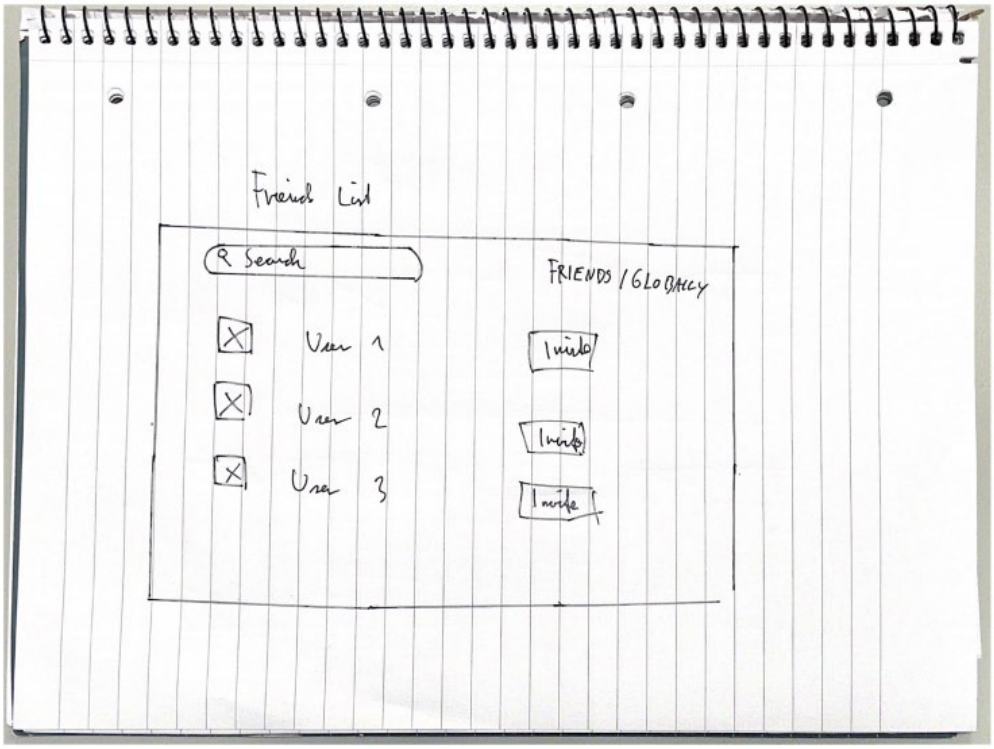
Skissering av fullført registrering



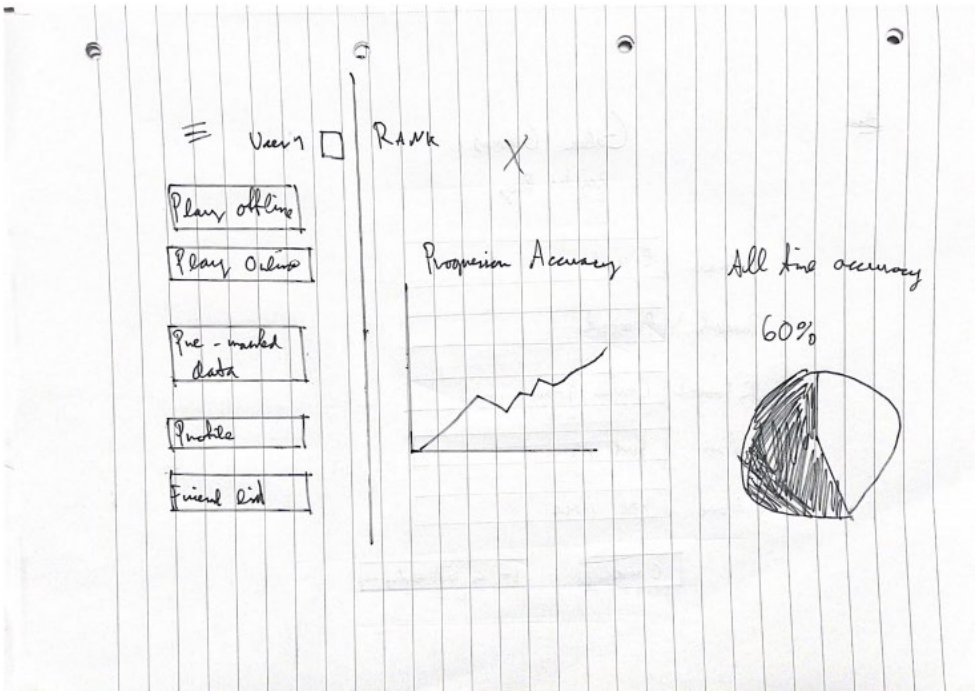
Skissering av hendelsesforløpet til et spill i Galea Games



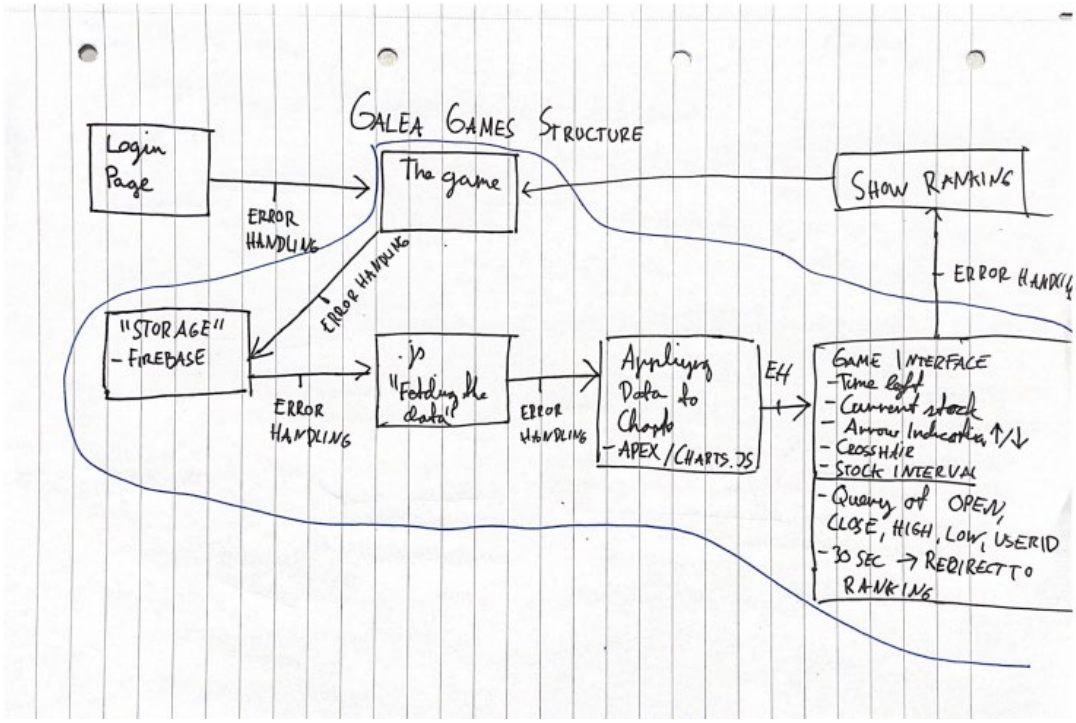
Skissering av resultater etter et spill



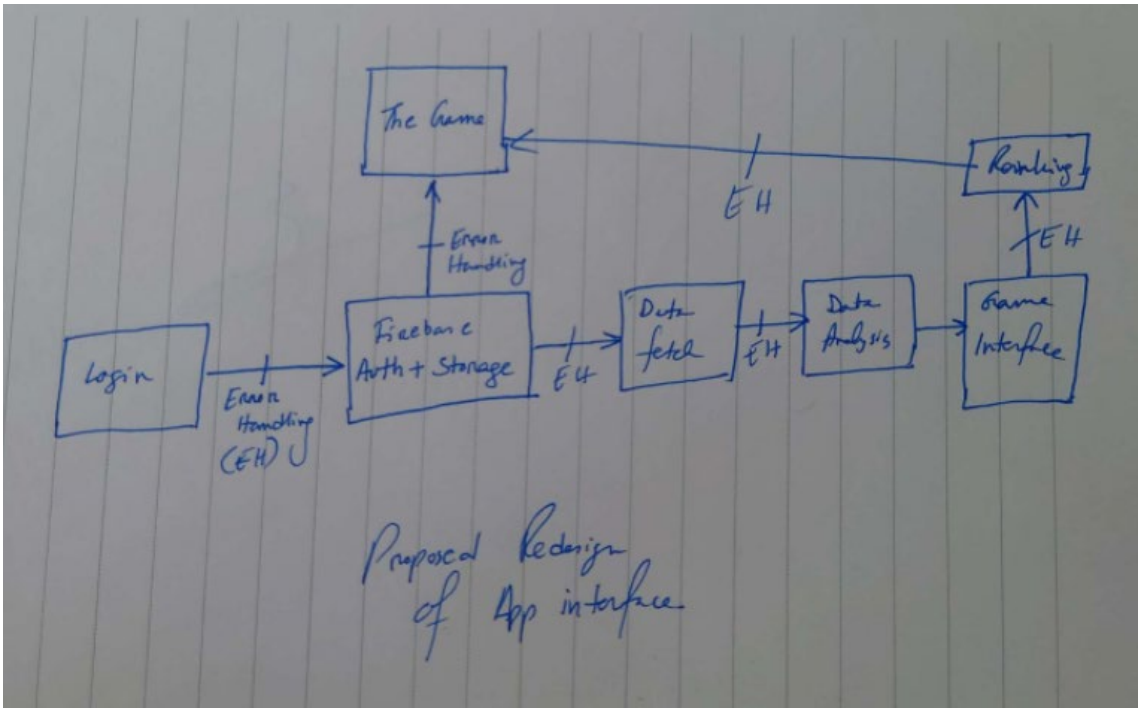
Skissering av venneliste i spillet



Skissering av «Brukerprofil/Min profil» i spillet



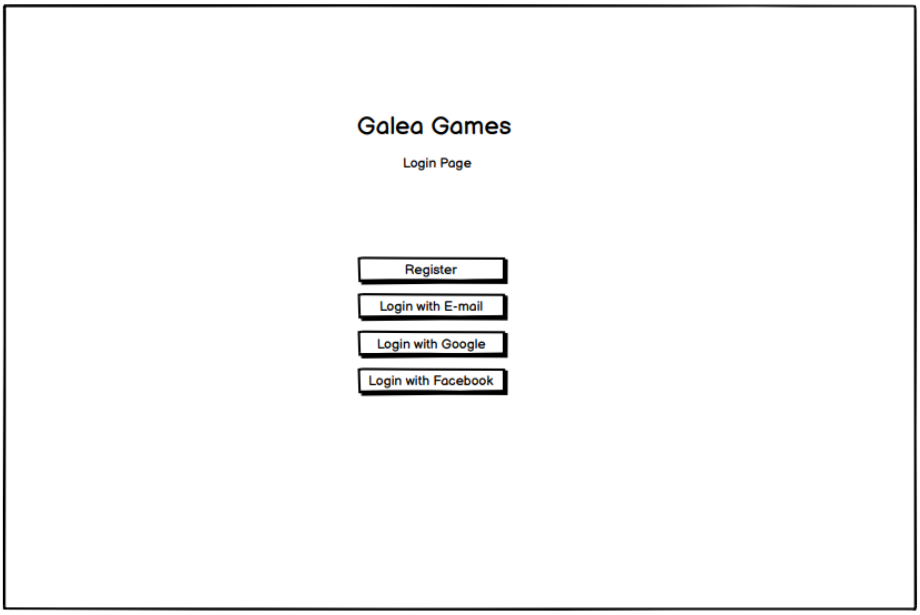
Skissering av databehandling i applikasjon. Alle skisseringer er tidlig utkast av hvordan tenkt arkitektur skal være. Ingen av disse er finaliserte og representerer sluttproduktet.



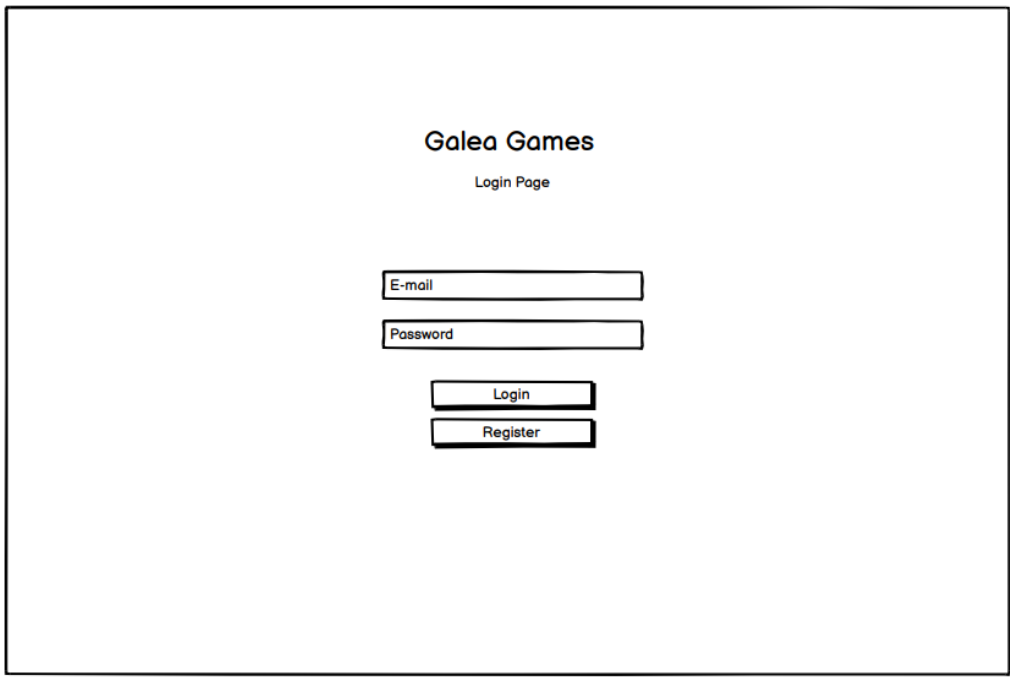
Revidert utkast av databehandling

Vedlegg 7: LoFi-prototype (Balsamiq)

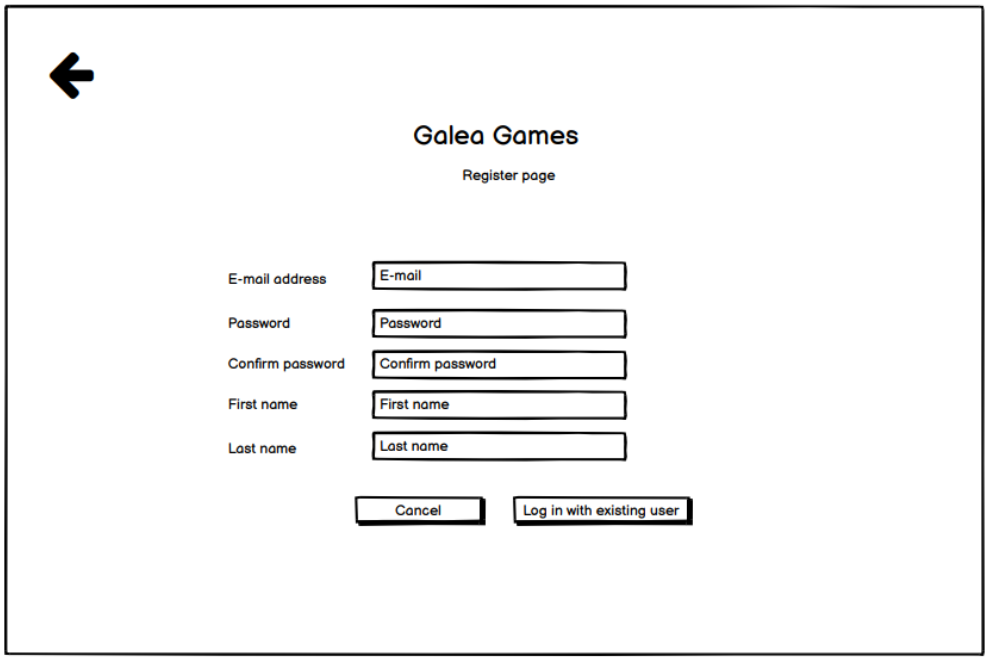
Velkomstside Galea Games



Innloggingsalternativer

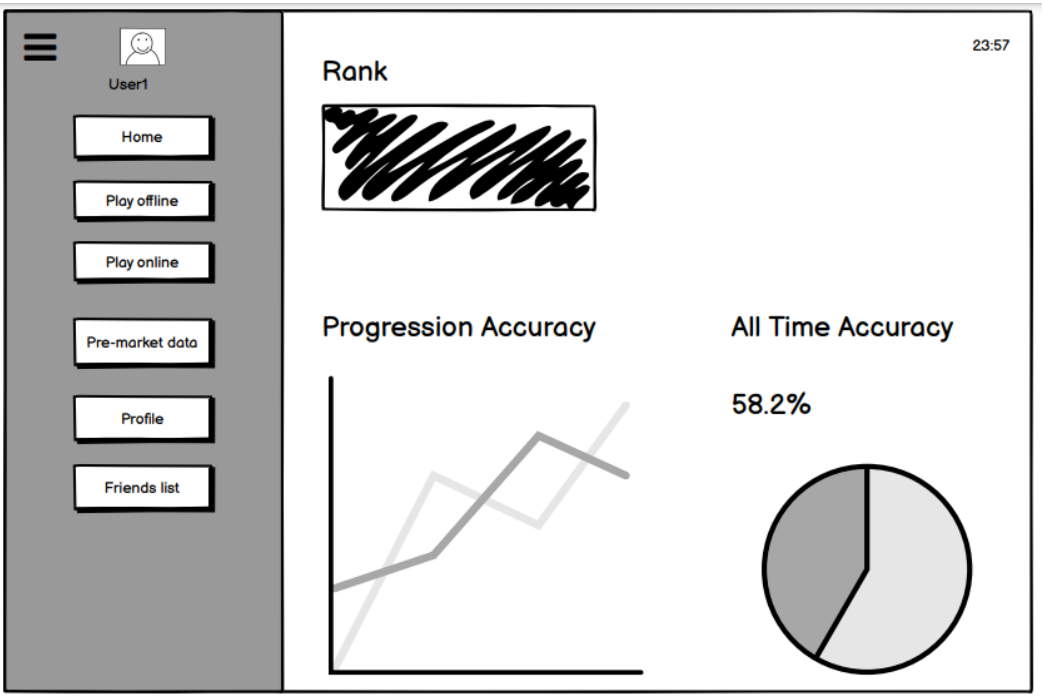


Innloggingsside (E-post og passord)

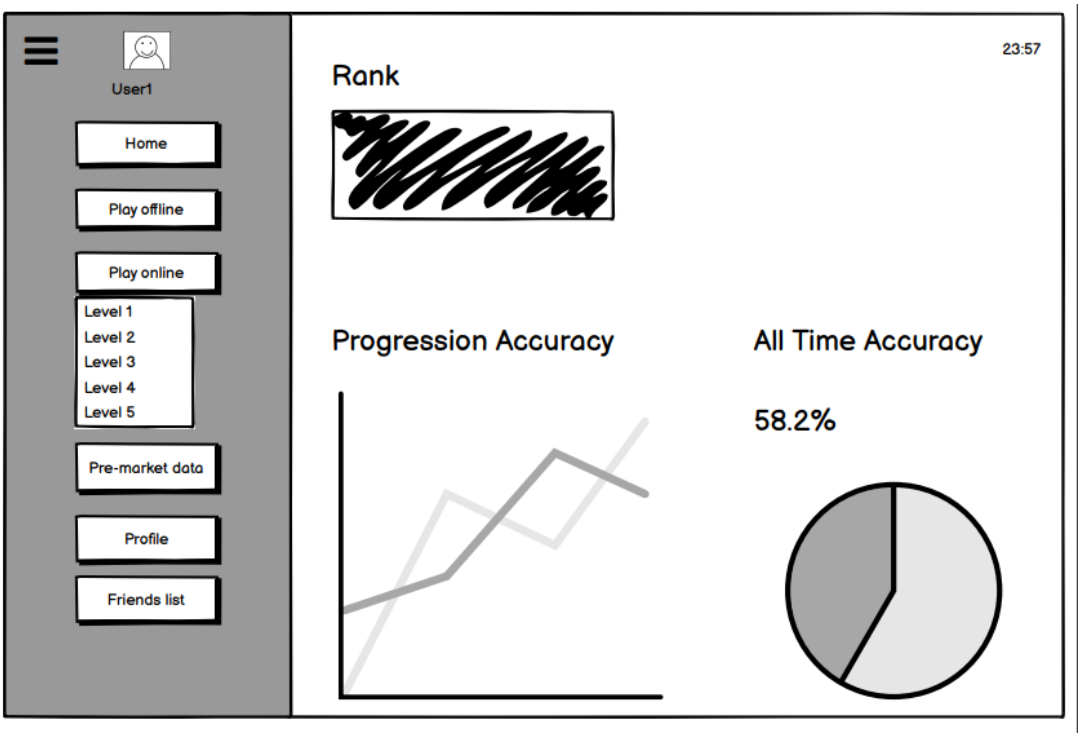


Registreringsskjema

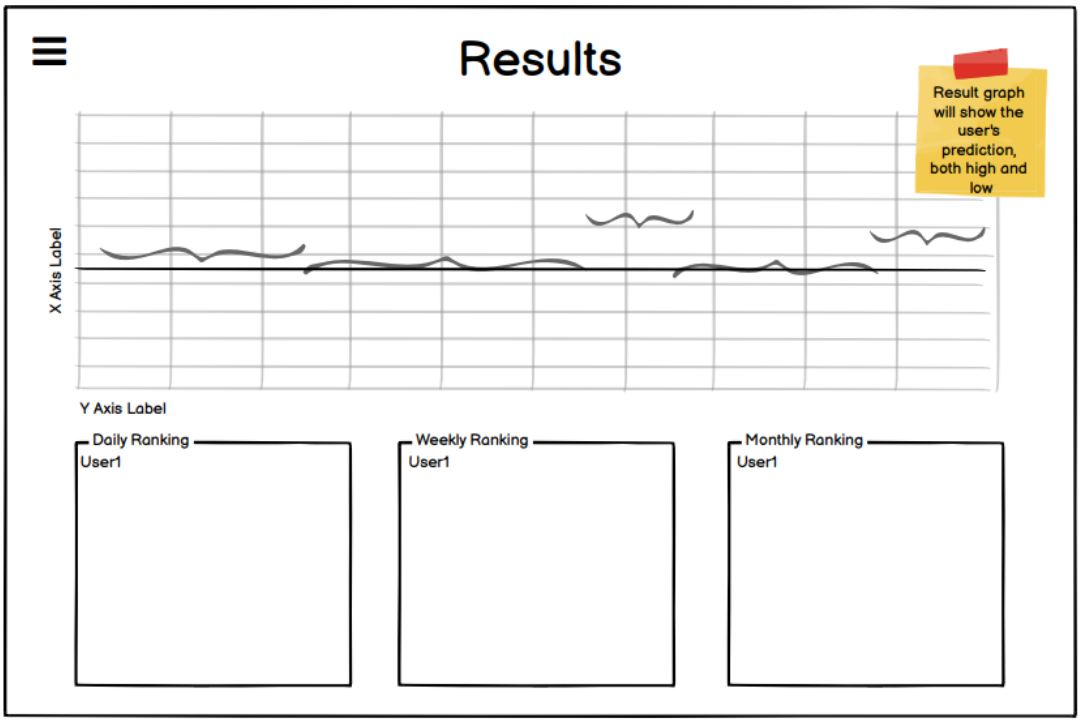
Fullført registrering melding



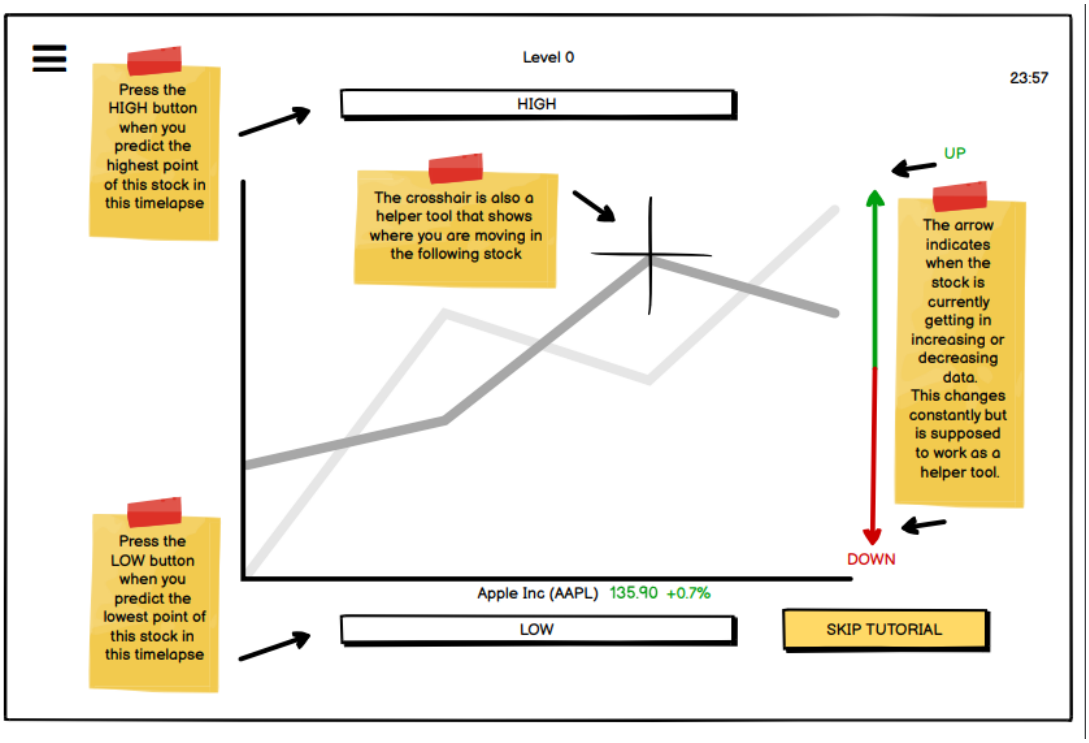
Min profil



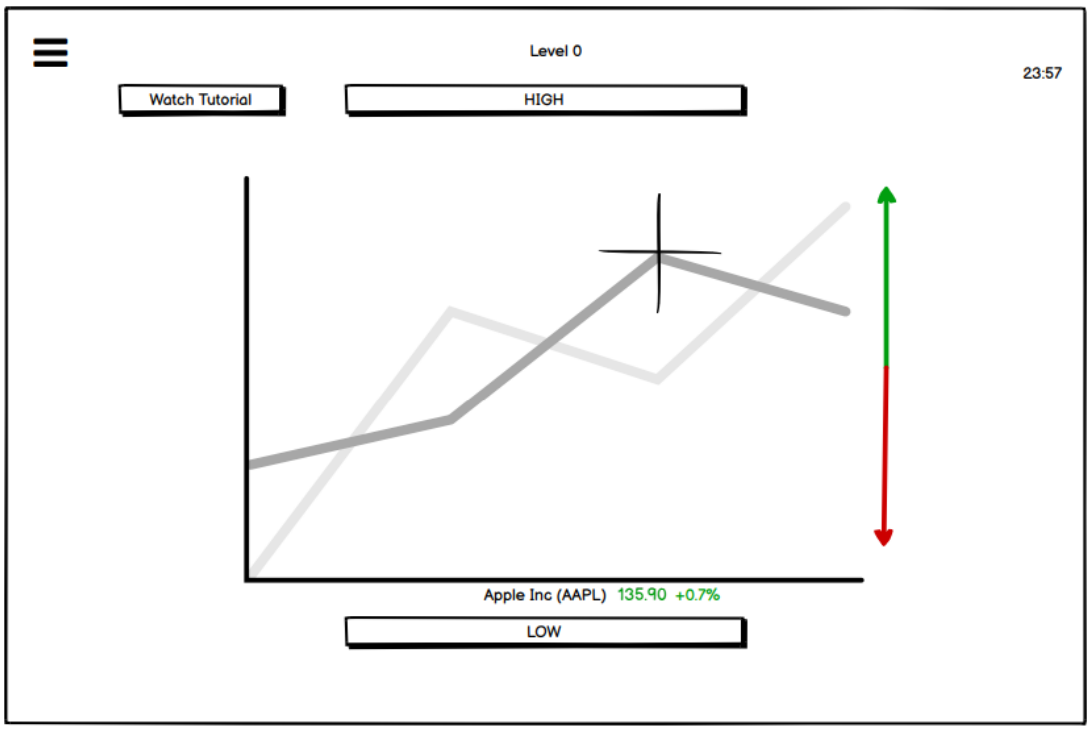
Min profil med nedtrekksmeny



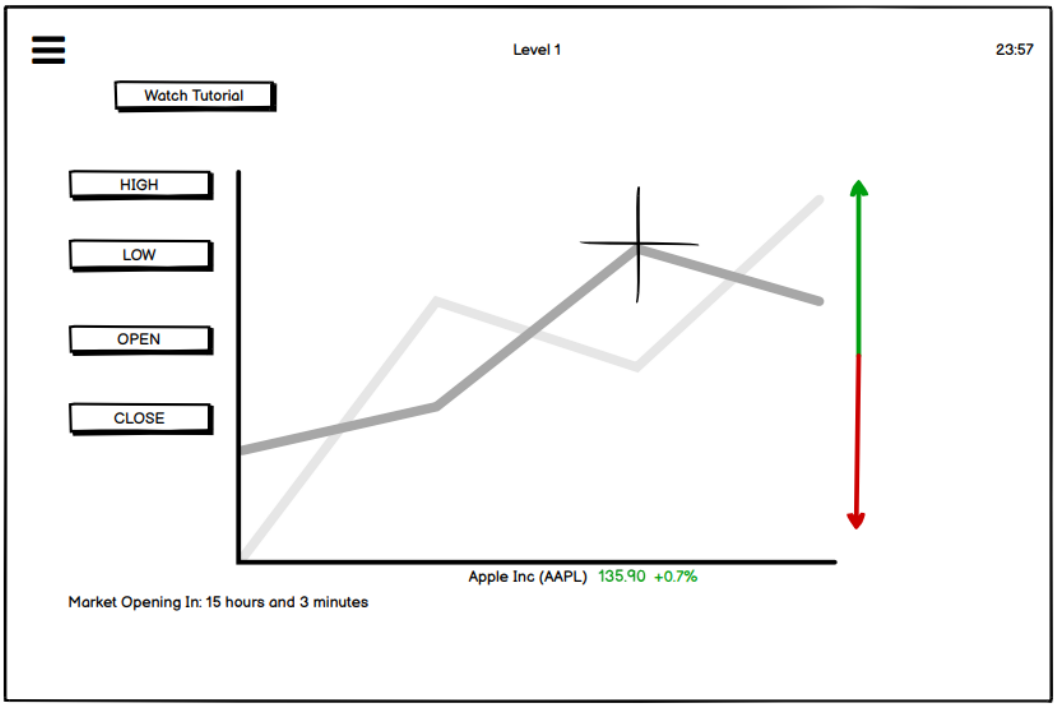
Resultater



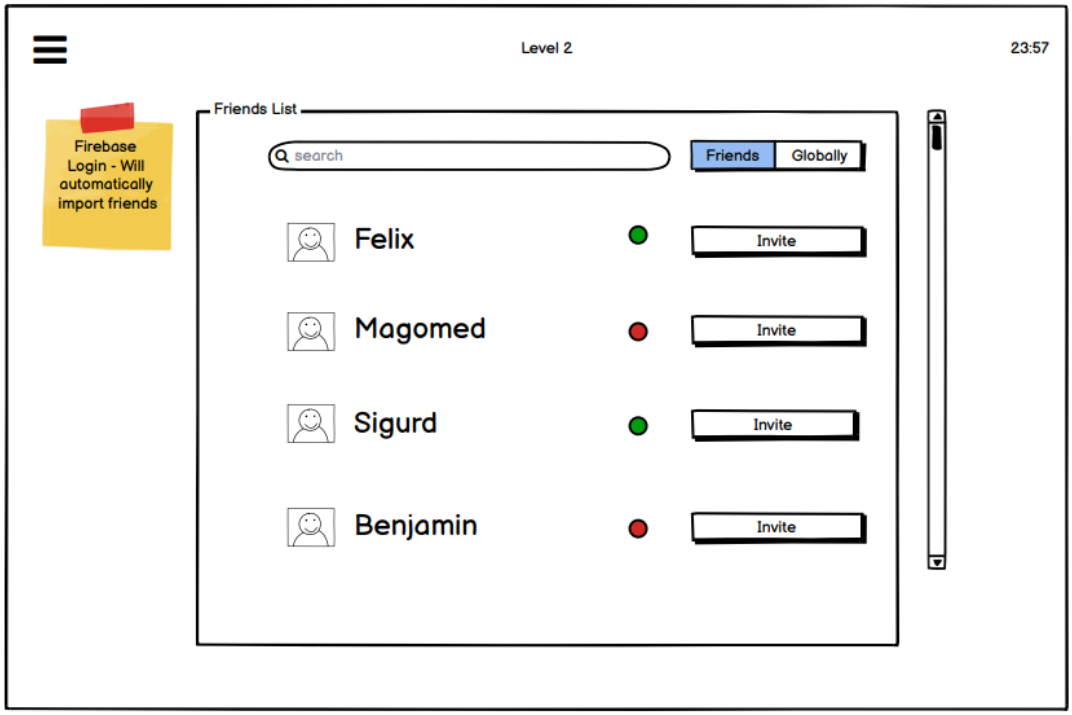
Level 0 av spillet med tutorial



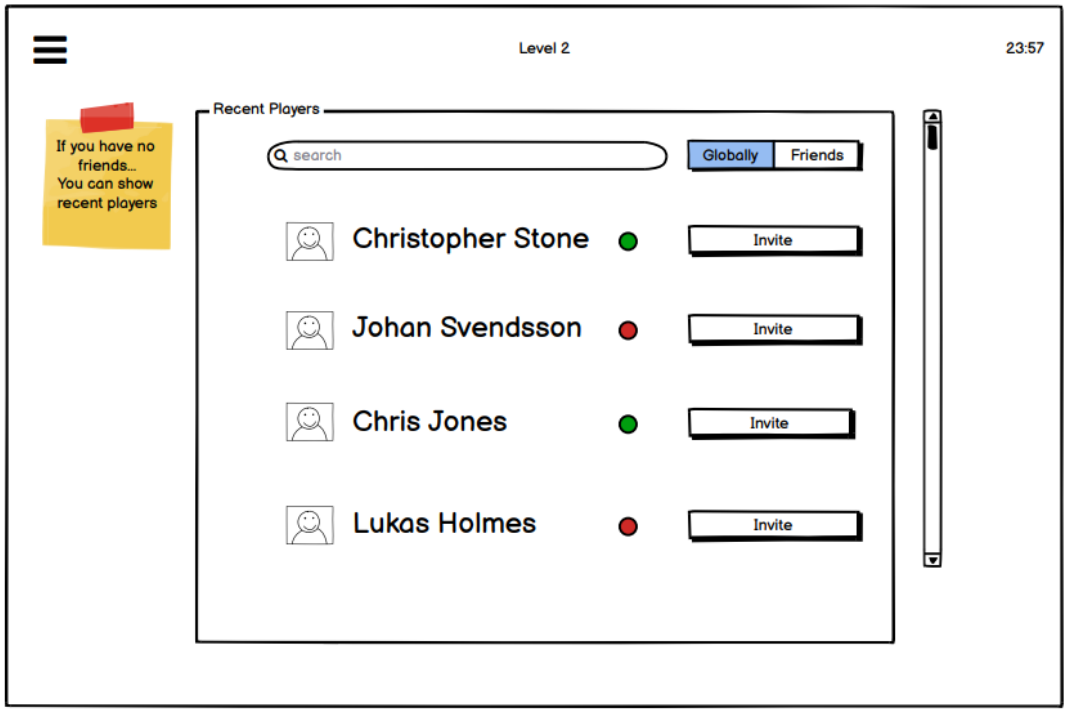
Level 0



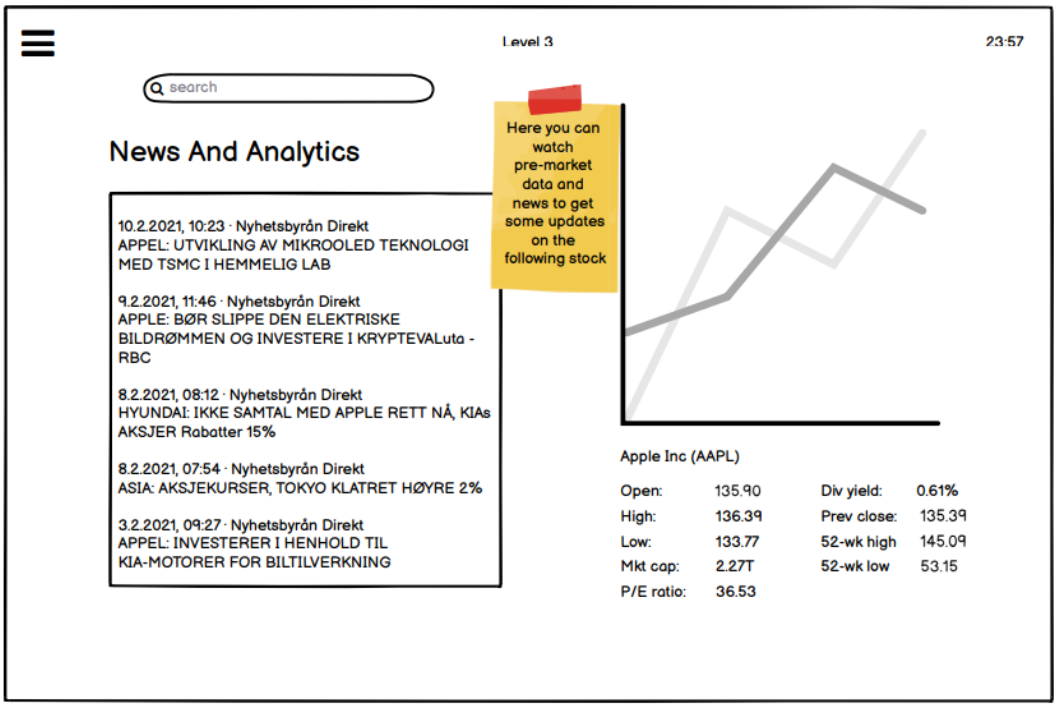
Selve spillet (Level 1)



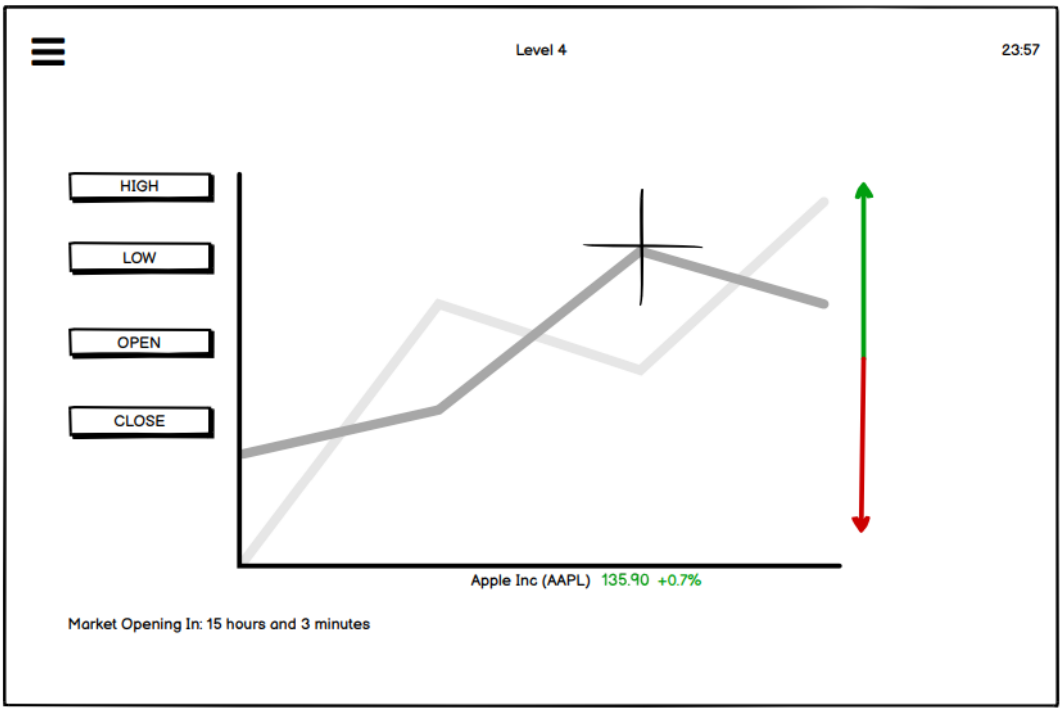
Venneliste (Level 2)



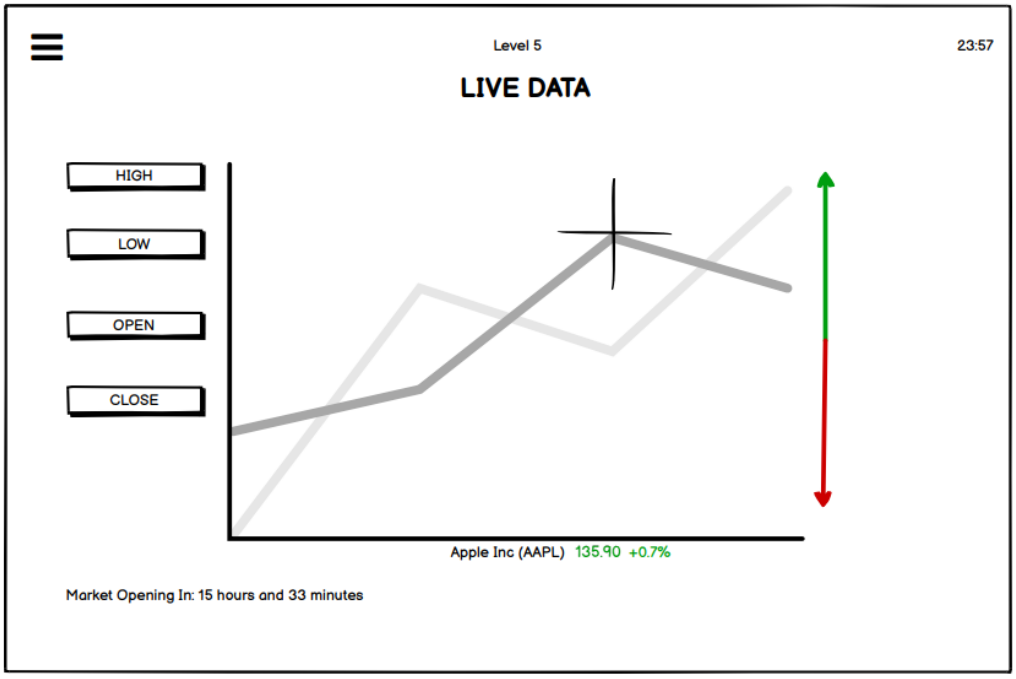
Spillere du har spilt mot (Level 2)



Pre-market (Level 3)



Level 4

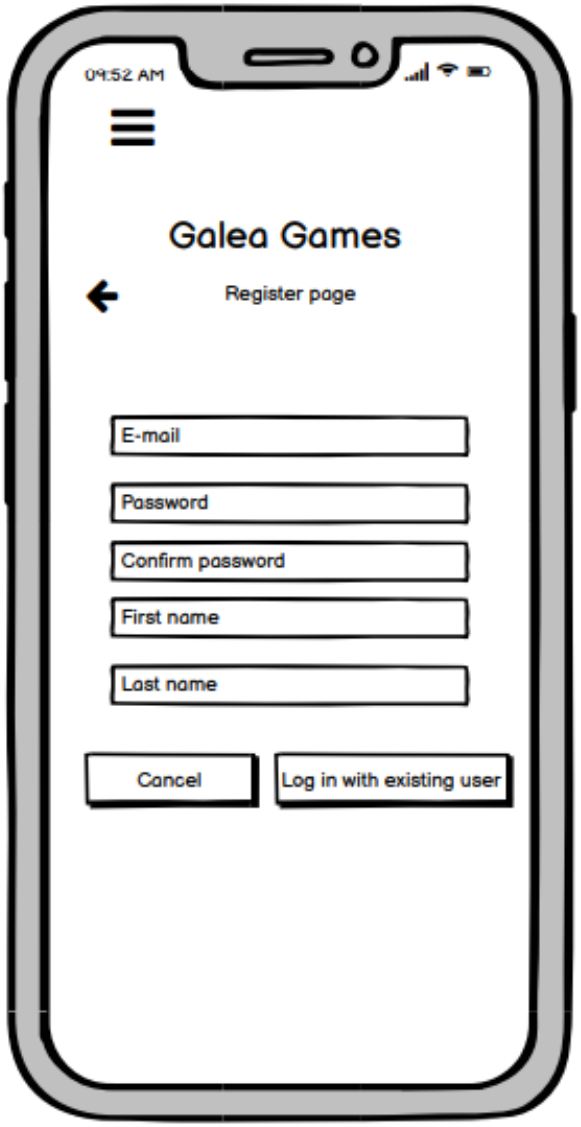


Live data (Level 5)

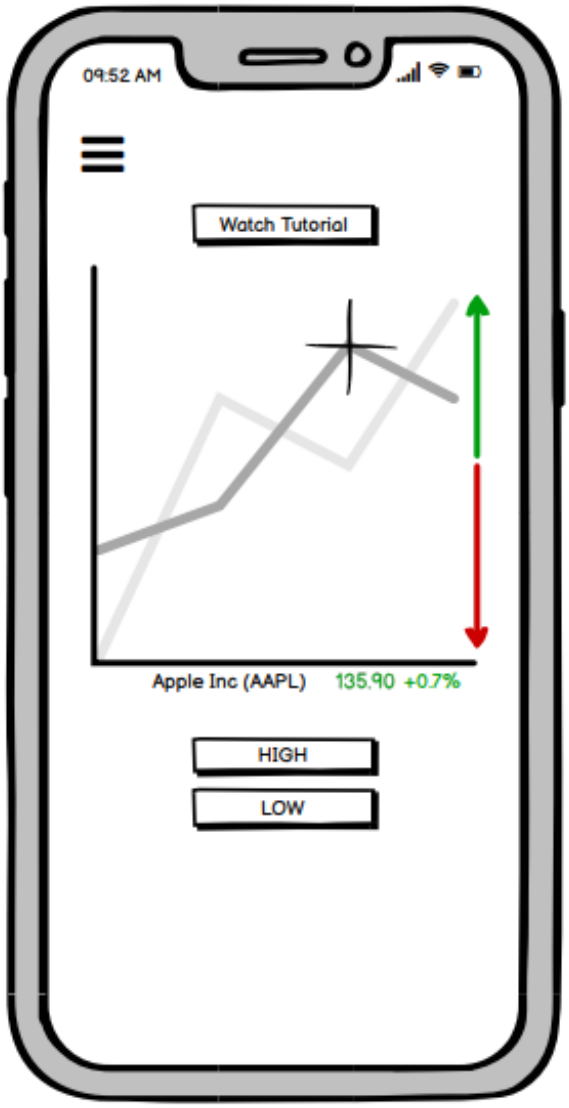
HiFi-skisser på mobile enheter:

Innloggingsalternativer mobil

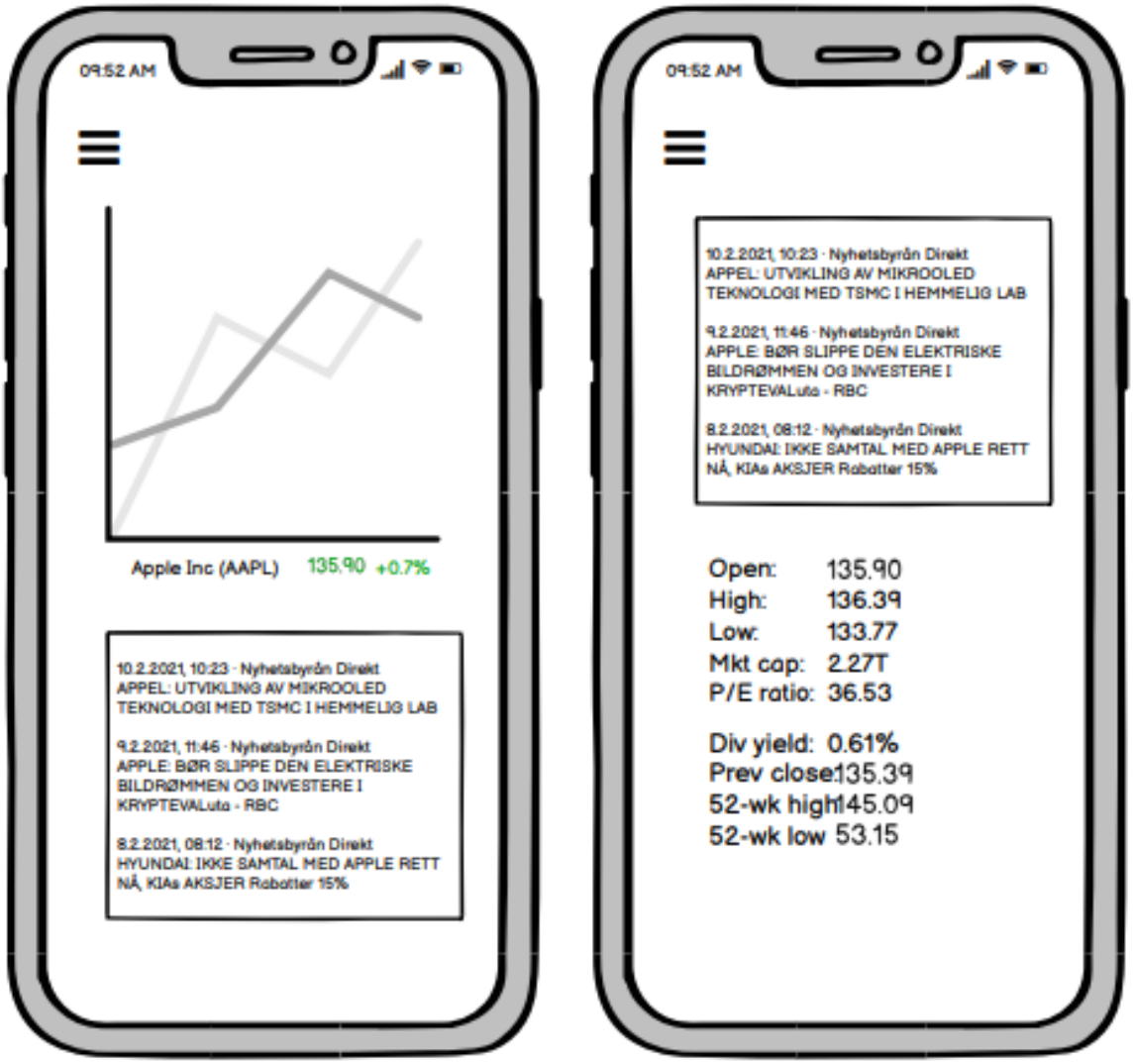
Innlogging e-post og passord mobil



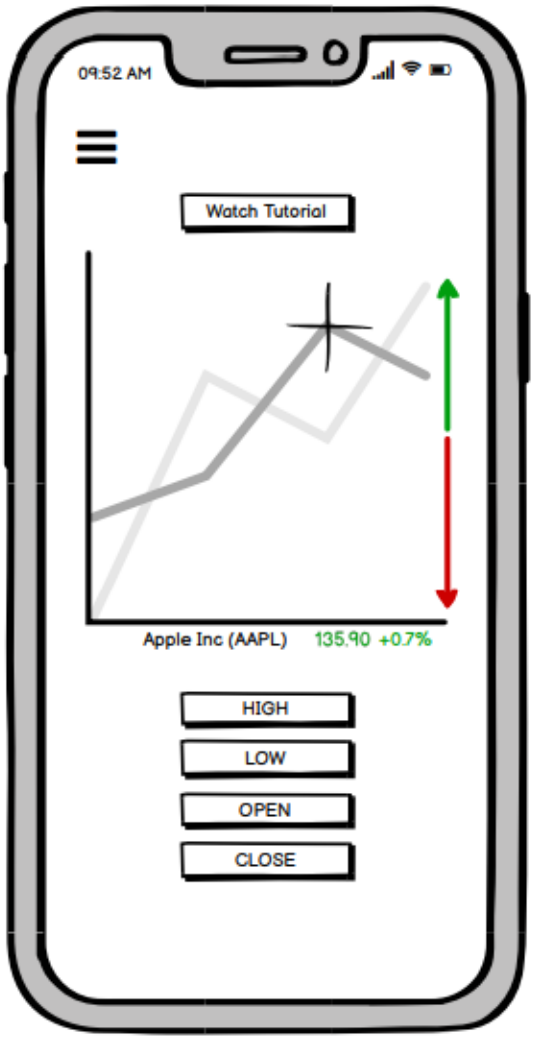
Registreringsskjema mobil



Spillets gang på mobil (High og Low)



Pre market data på mobil



Spillets gang (High, Low, Open, Close) på mobil

Vedlegg 8: Oppsett av Galea Games applikasjon

Dokumentasjon for oppsett av Galea Games React applikasjon

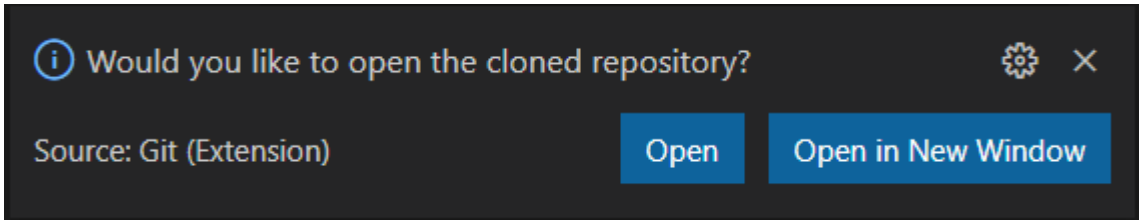
Ved oppsett Galea Games applikasjonen, er det nødvendig å benytte et Integrated Development Environment (IDE) eller lignende. Anbefalt IDE vil være en som støtter kommandoer i terminal og JavaScript. Det vil også være nødvendig å ha mulighet til å klone repo-et fra GitHub. Derfor er det anvendelig å bruke et IDE slik som Visual Studio Code, WebStorm IDE eller Atom. Ved utviklingen av applikasjonen var det benyttet Visual Studio Code, som er en gratis open-source IDE utviklet av Microsoft. Neste avsnitt tar forstand til at du som skal sette opp applikasjonen har Visual Studio Code installert.

Følgende steg beskriver innhenting av prosjektmappen til applikasjonen og installasjon av nødvendig moduler og avhengigheter som kreves for at applikasjonen starter:

Alternativ 1 - Kloning fra GitHub, installasjon og oppstart:

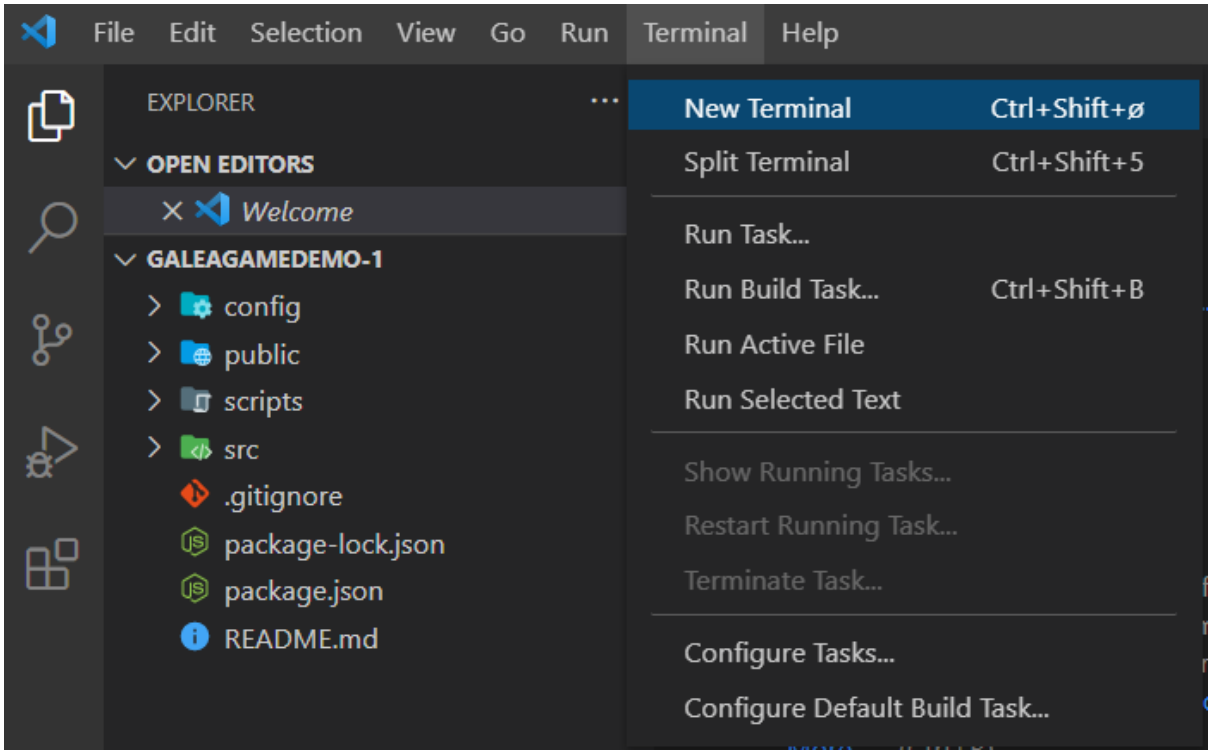
1. Installer «GitHub Pull Requests and Issues extension»
2. Etter installasjon logg inn på GitHub kontoen din ved å trykke på «Log In»-knappen i «GitHub»-siden i sidepanelet (eller opprett en konto dersom du ikke har det fra før av)
3. Dersom du befinner deg på «Source Control» har du valgmulighetene «Open Folder» og «Clone Repository». Velg Clone Repository. https://code.visualstudio.com/docs/editor/github
4. Lim inn lenken til prosjektet: https://github.com/s333933/GaleaGameDemo-1
5. Velg hvor på datamaskinen du vil lagre prosjektmappen.

6. Etter valg av plassering til prosjektmappen, vil repo-et laste ned og du vil få en spørring «Would like to open the cloned repository?». Dersom du ønsker repo-et i samme vindu trykk «Open» eller hvis du ønsker å åpne i et nytt vindu, trykk «Open in New Window» (*Figur 1*).

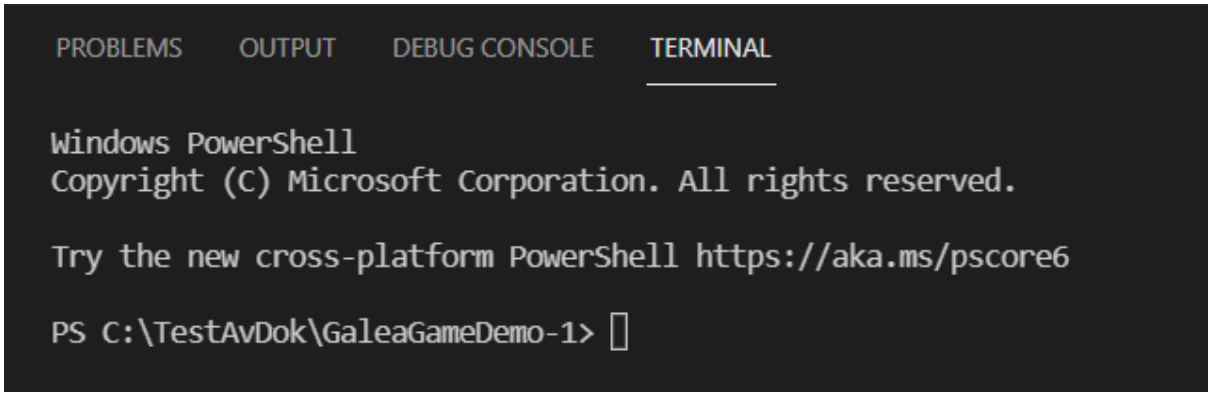


Figur 1 – Visual Studio Code Spørring

7. Naviger deg fram til «Terminal» (*Figur 2*), trykk på «New Terminal»
8. Et nytt terminal-vindu vil åpnes. Sørg for at du er i repo-ets mappe. Dette vil eksempelvis se slik ut: \Installasjonsmappe\GaleaGameDemo-1> (*Figur 3*)
   Her må man skrive «npm install» for å installere applikasjonen. (*Figur 4*) (Dette inkluderer moduler og avhengigheter for å starte applikasjonen).
9. Etter installeringen er ferdig, skriv «npm start» og trykk enter (*Figur 5*)
10. Etter at kommandoen er startet, vil en lokal server med adresse «localhost:3000» opprettes og åpnes automatisk i datamaskinens standard nettleser.



Figur 2 – Skjermbilde av Visual Studio Code – Terminal -> New Terminal



Figur 3 – Terminalvindu Visual Studio Code

Figur 4 – Kommando «npm install» i terminal

Figur 5 – Kommando «npm start» i terminal

Oppsett av Flask-server i Python

For å benytte seg av spillet og dens fulle funksjonalitet, må man først sette opp en Flask-server ved å kjøre et Python-skript som sørger for innhenting av predefinert data. Dataene sendes til Firestore (databasen) som applikasjonen er koblet opp mot. Oppkoblingen til databasen er ved hjelp av en API-nøkkel ved navn **service_key.json** som også befinner seg i samme mappe som Python-skriptet. Dataene sendes først når man trykker på «Start Game» i applikasjonen og når Flask-serveren kjører.
Python-skriptet befinner seg i prosjektmappen til applikasjonen, skriptet ligger under src\realtimefb**firestore.py**. Det finnes også alternativt skript med et annet datasett. src\realtimefb**firestoreanother.py**

Følgende steg beskriver oppsett av Flask-server til applikasjonen som kreves for at applikasjonen fungerer etter hensikt:

1. Terminal -> New Terminal, dersom en terminal ikke er allerede oppe
2. Videre må man navigere seg til mappen firestore.py befinner seg under. I terminalen skriver man «cd src\realtimefb» + enter (*Figur 6*)

```
PS C:\TestAvDok\GaleaGameDemo-1> cd src\realtimefb
```

Figur 6 – Kommando «cd sr\realtimefb» i terminal

3. Når man befinner seg i realtimefb-mappen, starter man skriptet ved å skrive «py firestore.py» + enter (*Figur 7)*

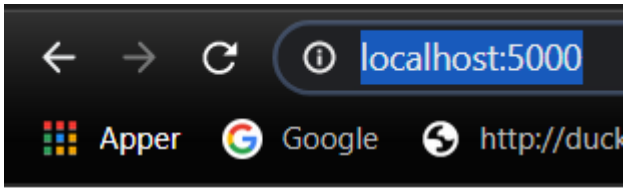
```
PS C:\TestAvDok\GaleaGameDemo-1\src\realtimefb> py firestore.py
```

Figur 7 – Kommando «py firestore.py» i terminal

```
PS C:\TestAvDok\GaleaGameDemo-1\src\realtimefb> py firestore.py
 * Serving Flask app "firestore" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 217-563-562
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figur 8 – Vellykket initialisering av Flask-server i terminal

4. Initialiseringen av Flask-serveren er vellykket når du får følgende melding (*Figur 8*) og at ved URL adressen localhost:5000 viser «OK» eller «Test» (*Figur 9*)



OK

Figur 9 – Hosting av lokal Flask-server på localhost:5000

Kommentar: Når spillet restartes, ved å trykke på «Restart» i applikasjonen, stopper Python-skriptet og Flask-serveren stoppes, dataene som er lagret i databasen fjernes. For å kjøre spillet på nytt må man derfor starte serveren på nytt. (Figur 7)

Alternativ 2: Installasjon og oppstart fra mappe

1. File -> Open Folder - Velg prosjektmappen «GaleaGameDemo-1»
2. Terminal - > New Terminal, sørg for at du står i prosjektmappen/repo-et i terminalvinduet. (Figur 2)
3. Her må man skrive «npm install» for å installere applikasjonen. (Figur 4) (Dette inkluderer moduler og avhengigheter for å starte applikasjonen).
4. Etter installeringen er ferdig, skriv «npm start» og trykk enter (Figur 5)
5. Etter at kommandoen er startet, vil en lokal server med adresse «localhost:3000» opprettes og åpnes automatisk i datamaskinens standard nettleser.

Andre kommentarer eller problemer?

Dersom det dukker opp problemer med å starte Python-skriptet og kjøre Flask-serveren må man installere Python på maskinen eller på Visual Studio Code.

https://code.visualstudio.com/docs/python/python-tutorial

Installere på MacOS og Windows:

https://www.python.org/downloads/

Vedlegg 9: Attest fra Lars Føleide (Galea AS)

ATTEST

Benjamin A. S. Jørgensen, Magomed M. Khatsjukajev, Sigurd Ø. Bjørndal og Felix S. Leypoldt har deltatt i et samarbeid med Galea AS i perioden 1/1-2021 til 30/4-2021. Samarbeidet har vært i form av selvstendig arbeid i en prosjektgruppe for et bachelorprosjekt ved OsloMet - Fakultet for teknologi, kunst og design (TKD).

Deres oppgaver har bestått av ukentlige leveringer (sprinter) og møter med Galea AS hver fredag - utenom avtalt avspasering. Forhåndsdefinerte mål og krav er dokumentert i prosjektrapporten, Galea Games – Finansspill.

Galea AS er svært fornøyd med prosjektgruppens innsats, leveringer og sluttprodukt.

Med vennlig hilsen,

Trondheim, 25. mai 2021

Lars Føleide

Daglig leder for Galea AS

Klæbuveien 52, 7030 Trondheim

Tlf: 98 45 44 99