Høgskulen på Vestlandet

Bacheloroppgave

Nettbasert testing av kodeferdigheter for rekrutterings formål

Web based assessment of coding skills for recruiting purposes

Svein Ove Surdal & Ruben Aadland

DAT191 Bacheloroppgave
Fakultet for ingeniør- og naturvitskap
Institutt for datateknologi, elektroteknologi og realfag

Veileder: Rogardt Heldal
Oppdragsgiver: Wide Assessment AS
Innleveringsdato: 23.05.2022

Vi bekrefter at arbeidet er selvstendig utarbeidet, og at referanser/kildehenvisninger til alle kilder som er brukt i arbeidet er oppgitt, *jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 10.*

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap
Institutt for datateknologi, elektroteknologi og realfag

TITTELSIDE FOR HOVEDPORSJEKTET

Rapportens tittel: Nettbasert testing av kodeferdigheter for rekrutterings formål	Dato: 23.5.2022
Forfatter(e): Ruben Aadland og Svein Ove Surdal	Antall sider u/vedlegg: 43
	Antall sider vedlegg: 58
Studieretninger: Dataingeniør	Antall disketter/CD-er: 0
Kontaktperson ved studieretning: Rogardt Heldal	Gradering: Ingen
Merknad: Ingen	

Oppdragsgiver: Wide Assessment AS	Oppdragsgivers referanse:
Oppdragsgiver kontaktperson: Andreas Hammerbeck	Telefon: 97661466

Sammendrag:
Bachelor rapporten omhandler planlegging og utvikling av et testsystem, for testing av CSS nivået til jobbsøkere. Testsystemet skal brukes opp mot rekrutterings formål og er utviklet for Wide Assessment AS. Systemet som er utviklet er en MVP, og er ment som et utgangspunkt til å videreutvikle til et ferdig system. Testsystemet skal til slutt integreres i Wide Assessment sin rekrutteringsplattform wa.works.

Stikkord:

Tekniske tester Bildesammenligning	Rekruttering CSS	React

Høgskulen på Vestlandet, Fakultet for ingeniør- og naturvitskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

FORORD

Denne rapporten går gjennom bachelor prosjektet Nettbasert testing av kodeferdigheter for rekrutterings formål. Formålet med rapporten er å beskrive utviklingen av CSS testsystemet utviklet for Wide Assessment AS. Prosjektet er gjennomført av Svein Ove Surdal og Ruben Aadland.

Vi ønsker å takke Wide Assessment for oppgave og at vi har blitt så godt tatt imot av dere. En spesielt takk går til Andreas Hammerbeck og Sindre Andreassen som alltid har en åpen dør og hjulpet med et smil.

Takk til Rogardt Heldal for den støtten og hjelpen du har gitt oss under hele prosjektet. Selv med travle dager har du satt av møter, lest gjennom rapporter og gitt tilbakemeldinger. Takk skal du ha.

Vi vil og rette en takk til de andre foreleserne vi har hatt gjennom disse årene. Som på tross av covid har lært oss det vi trengte for bachelor prosjektet og arbeidslivet.

INNHOLDSFORTEGNELSE

ORDLISTE

Iframe: Inline frame brukes for å vise et dokument i et HTML dokument. Eks: En nettside inne i en annen nettside (W3schools, u.å.).

React: Et JavaScript bibliotek for å lage "single page applications" (React, u.å.).

JavaScript: Programmeringsspråk brukt i nettsideutvikling (Mozilla, u.å.).

Props: Data sendt fra en React komponent til en annen.

Typescript: Et programmeringsspråk bygget på javaScript, men med typer (Typescriptlang, u.å.).

Visual Studio Code: Et utviklingsmiljø for programvareutvikling.

Ninyo: Maskoten til Wide Assessment AS.

Redux: Redux er en state manager for Javascript applikasjoner (Redux, u.å.).

Redux store: Komponent i redux som holder forskjellige state verdier. Ofte brukt for å dele data fra API mellom komponenter i en web-applikasjon (React-redux.js, u.å.).

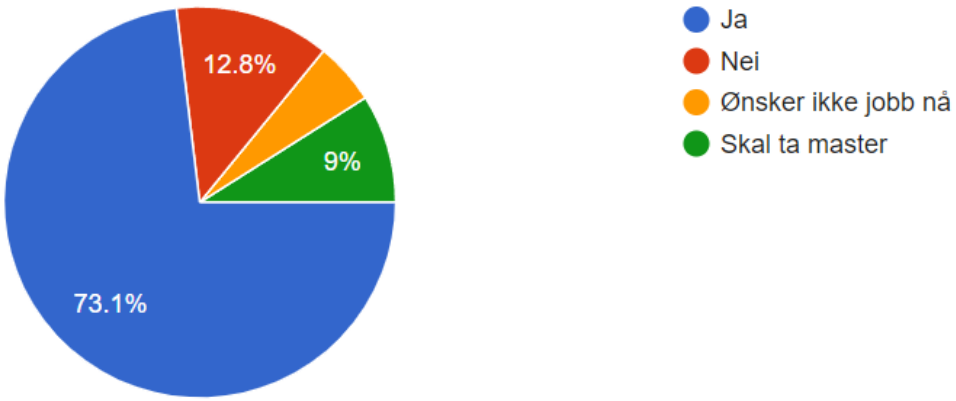
1 INNLEDNING

1.1 KONTEKST

Samfunnet er i en stadig økende grad av digitalisering og behovet for kompetanse innenfor IKT er økende. Basert på tall fra Manpowers skal over 60% av it-bedriftene de har kontakt med bemanne opp i starten av 2022 (Experis 2022).

I 2021 gjorde NAV en bedriftsundersøkelse hvor de blant annet sammenlignet behovet for arbeidskraft med ledig arbeidskraft (Gjerde 2021 p. 6). Her fant de at behovet for arbeidskraft innenfor IKT er nær den ledige arbeidskraften. Dette kan tyde på lite rom for vekst i markedet.

For studenter ved data- og informasjonsteknologi studiet ved HVL ble det sendt ut er spørreskjema. Skjemaet har pr. 10.5.2022 blitt besvart av 78 studenter som er ferdig med bachelorgraden til sommeren. Bilde 1.1 viser resultatene, og viser at det bare er 12,8% som ikke har skaffet seg jobb pr. 10.05.2022.



Bilde 1.1 Undersøkelse om hvor mange som har fått jobb. Bilde hentet 11.05.22.

Økende bemanning, lav ledighet og undersøkelsen viser at det potensielt er vanskelig for bedrifter å finne de riktige personene til stillingene de har. Dette kan skape flere problemer. Dersom et selskap ikke klarer å fylle en stilling kan de potensielt miste kunder. Det kan også skape feilrekruttering, og det kan være dyrt for bedriften (HRmagasinet 2018; Berggren, 2020). Rekrutteringsverktøy kan hjelpe bedrifter å gjøre bedre valg ved å gi dem oversikt over hva kandidaten kan. De kan og gi bedrifter muligheten til å ansette kandidater som ikke har den nødvendige utdanning, men den nødvendige kompetansen for den relevante stillingen.

1.2 PROSJEKTEIER

Prosjekteier er Wide Assessment AS, og er heretter kalt WA. WA ble grunnlagt i 2016, og har som mål å forenkle rekrutteringsprosessen i IT-industrien. Dette blir gjort med wa.works som er rekruteringsplattformen utviklet av WA. Her kan jobbsøkere lage en skill basert CV sammen med tidligere arbeidserfaring og utdanning. Bedrifter, heretter kalt kunder, kan da søke etter kandidater basert på hvilke egenskaper de ser etter.

Da den første versjonen av wa.works ble lansert var det meste bygget av studenter, og mye av dagens system er fortsatt det. Det er også årlig inne studenter fra HVL i praksis som videreutvikler eksisterende eller nye systemer. Mange av studentene fra praksis får også tilbud om å gjennomføre bacheloren hos WA.

1.3 MOTIVASJON

Målet med wa.works er å gjøre rekrutteringsprosessen enklere, og en viktig del av dette er å gjennomføre tekniske tester. Tekniske tester blir ofte brukt av IT-industrien for å sjekke ferdighetene til jobbsøkere. Testene er en viktig del av rekrutteringsprosessen, men krever ofte mye arbeid å få gjennomført (Mæland, 2020; Berg, 2019; Talentech u.å.). For jobbsøkere er det ofte tungvint å gjennomføre mange tekniske tester gjennom flere intervjuprosesser, men med en felles testplattform kan det redusere antall tester som må gjennomføres. Ønsket er at tekniske tester kan bli gjort på wa.works for å videre forenkle rekrutteringsprosessen.

1.4 PROBLEMBESKRIVELSE OG MÅL

WA videreutvikler kontinuerlig wa.works plattformen, og nå er de kommet til neste steg i prosessen for å forenkle rekrutteringsprosessen. I dagens system kan kandidater legge til ferdigheter og nivået på denne, men det er noen svakheter i dette systemet. For kandidaten kan det være vanskelig å vite hvilke nivå de har i ferdigheten uten å ha noe å sammenligne seg med. For kunden kan det også være vanskelig å stole blindt på ferdighetsnivåene som kandidaten har gitt seg selv.

Målet med prosjektet er å lage en testplattform der kandidater kan teste kode-ferdighetene sine. Det ferdige systemet skal kunne teste kandidater i flere forskjellige programmeringsspråk, men i første versjon skal det kun støtte CSS.

> Målet er å lage en plattform hvor kandidater tester ferdighetene sine i CSS, og får en vurdering som kan brukes til å fastslå kandidatens nivå.

1.5 OPPBYGGING AV RAPPORTEN

Rapporten består av 7 kapitler som beskriver arbeidet for å lage en testplattform for Wide Assessment.

I kapittel to blir prosjektbeskrivelsen gjennomgått, og det blir fastsatt hva kravene, avgrensningene og ressursene er for dette prosjektet

Kapittel tre beskriver utformingen av prosjektet, og hvilken løsning som blir laget til WA. Det blir også fastsatt hvilke utviklingsmetode som blir brukt, og hva evalueringsplanen er til prosjektet.

Testplattformen blir detaljert beskrevet i kapittel fire. Her er design, levert løsning, og de viktigste funksjonene beskrevet i detalj.

Det femte kapittelet beskriver hva metoden er for å evaluere plattformen, og hva evalueringsresultatene er.

Diskusjonen av prosjektet er i kapittel seks, og har med refleksjoner rundt arbeidet som er gjort.

Til slutt blir konklusjonen og planen for videre arbeid presentert i kapittel syv.

2 PROSJEKTBESKRIVELSE

Dette kapittelet beskriver selve prosjektet som skal gjennomføres. Her blir den praktiske bakgrunnen presentert, hvordan prosjektet er avgrenset og hvilke ressurser som var tilgjengelig.

2.1 PRAKTISK BAKGRUNN

2.1.1 Tidligere arbeid

Arbeidet i denne bacheloroppgaven er en ny funksjon til WA sin plattform wa.works. Med denne ønskes det å skape et mer sikkert nivå system for kandidater og kunder. Noe innledende brainstorming ble gjort sammen med WA, men gruppen stod ganske fri i måten en ønsket å lage systemet. WA ønsker å utvikle dette konseptet, og rapporten vil omhandle utviklingen av et minste brukbare produkt (MVP). For at testplattformen skal vise tilhørighet til wa.works, vil deler av styling og komponenter fra wa.works bli brukt i systemet.

2.1.2 Initielle krav

Gjennom den utdelt oppgaven og samtaler med WA CTO Andreas ble det bestemt noen initielle krav for testplattformen (se oppgavebeskrivelsen i vedlegg):

1. Brukere skal kunne velge en test ut fra en liste med tester
2. Når en test starter, skal brukeren kunne skrive inn kode for å løse oppgaven
3. Når oppgaven blir levert skal resultatet kompileres, testes, og få en poengscore
4. Etter å ha gjennomført testen skal resultatet og koden lagres for bruk i rekrutteringssystemet.
5. Kunder skal kunne lage egne tester.

2.1.3 Initiell løsnings-idé

I prosjektoppgaven skriver WA om hva bakgrunnen til prosjektet er (se oppgavebeskrivelsen i vedlegg). De ønsker å gjøre prosessen ved å gå fra kandidat til ansatt så enkel som mulig. Derfor ønsker de at kandidater kan gjennomføre kodeoppgaver på plattformen deres for flere grunner:

- Ved hjelp av tester kan kandidatene få oversikt over kunnskapsnivået sitt, og se hvordan de ligger an i forhold til andre.
- Kunder kan enklere finne den kandidaten som er rett for deres stilling.
- Mulighet for enkel testing av kandidater i en intervju prosess.

Testsystemet er ment for å styrke wa.works, og støtter ønsket til WA om å gjøre rekruttering enklere i IT-bransjen.

2.2 AVGRENSNINGER

Bachelor prosjektet går ut på å lage en testplattform for å kunne teste kodeferdighetene til kandidatene. WA ønsker at tester for JavaScript eller CSS blir implementert, og det er tenkt at systemet skal utvides med flere språk senere. Prosjektet er begrenset til implementering av tester for CSS. Etter samtaler med WA er det bestemt å ikke lagre dataen i databasen. Dette er for å kunne fokusere på testplattformen og funksjonene som har høyere prioritering.

I utviklingen av testplattformen er det også begrensninger innenfor valg av teknologier. Valgte teknologier blir presentert i kapittel 3.2 og 3.3

2.3 RESSURSER

Hos WA er det tildelt eget kontor for arbeid med bacheloroppgaven. Her har gruppen hele tiden tilgang til utviklerne hos WA, og døren deres var alltid åpen for spørsmål. Det meste av utviklingen i løpet av bachelorprosjektet er blitt gjennomført på kontoret til WA.

Ut fra teknologivalget som blir gjort kan mye av kode og styling fra wa.works bli gjennbrukt i testsystemet. Dette gjør det enklere og mer effektivt å lage et produkt som ligner på wa.works.

3 UTFORMING AV PROSJEKTET

Dette kapittelet beskriver forskjellige løsninger som er tilgjengelig, og hvilken av disse som er valgt for å lage testplattformen. Deretter går delkapitlene inn på hvilke verktøy og metodikk som er blitt brukt for å løse prosjektet. Til slutt blir evalueringsplanen til prosjektet kort introdusert.

3.1 FORSLAG TIL LØSNING

Delkapitlene under går inn på forskjellige løsninger til de initielle kravene som er satt i kap. 2.1.2.

3.1.1 Alternativ 1: Integrert i eksisterende system

Første alternativ er å lage et testsystem som er integrert i den eksisterende rekrutteringsplattformen til WA. Dette vil da fungere ved å legge til en ny side som kandidater og kunder kan komme seg til. Ønsket for testsystemet er da at stylingen ligner den som allerede er brukt wa.works. Med denne løsningen vil kandidater og kunder først logge inn til sine respektive sider, og deretter komme seg inn på testsystemet for å ta eller opprette tester.

Løsning en er integrert i det eksisterende systemet til WA, og vil derfor bruke mange av de samme teknologiene. Dette gjør det enklere å videreutvikle og vedlikeholde systemet for ansatte hos WA, etter som de er kjent med teknologiene som blir brukt.

3.1.2 Alternativ 2: Dedikert nettside

Andre alternativ er en egen nettside som er frittstående og ikke integrert direkte i det eksisterende rekrutteringssystemet. Her vil det være muligheter å ta tester for kandidater og opprette tester for firma.

Med en dedikert nettside for testsystemet er det større frihet til å velge styling og teknologier som blir brukt.

Hoved ønsket til WA er at systemet skal kunne brukes opp mot deres rekrutteringssystem, og brukere på testsiden må da være oppkoblet mot wa.works brukere eller dele brukere mellom plattformene.

3.1.3 Alternativ 3: App

Tredje alternativ er en mobil app. Dette kan være en native Android eller IOS app, eller en hybridløsning for begge systemene. Testplattformen trenger ikke tilgang til mobil-sensorer, og en hybrid løsning vil derfor fungere best.

Appen vil kun være for gjennomføring av tester, og det trengs en separat løsning for å kunne lage tester. Dette kan bli gjort på lignende måte som alternativene presentert over.

Som nevnt i alternativ to er et av ønskene at testsiden skal kunne knytte tester og resultater opp mot den eksisterende rekrutteringsplattformen. Et lignende innloggingssystem er da nødvendig, og vil ligne det presenterte i kapittel 3.1.2 Løsning 2: dedikert nettside.

3.1.4 Diskusjon av alternativene

Fordeler med det første alternativet er at den er naturlig oppkoblet mot profilene til kunder og kandidater. Dette er enklere rent praktisk ettersom alt er på samme sted, og kan bruke eksisterende komponenter og styling. Et problem kan være sikkerhet rundt å kjøre kandidaters kode på wa.works plattformen, men mulige tiltak for dette er beskrevet i systemdokumentasjon, kapittel 7.

Andre alternativ vil gi større frihet for styling og valg av teknologier som blir brukt. Dette kan ha fordeler som å velge det som egner seg bedre enn det som blir brukt på den eksisterende rekrutteringsplattform. I ettertid vil det derimot bli vanskeligere å integrere og vedlikeholde testsystemet ut fra teknologivalget.

Tredje alternativ har ingen store fordeler ovenfor de to andre løsningene. En native applikasjon kan få dypere tilgang til maskinvaren, men for testsystemet er ikke disse dype funksjonene nødvendig.

3.2 VALGT LØSNING

Det beste alternativer for å løse de initielle kravene for prosjektet er den første. Dette valget er blitt presentert for oppdragsgiver, og gjennom samtaler med dem er det kjent at testsystemet skal integreres i det eksisterende rekrutteringssystemet. Dette alternativet har en rekke fordeler ovenfor de andre løsningene:

- Den tillater at alt er på samme sted, noe som vil være mer intuitivt for en person som er vant med wa.works.
- Det er koblet opp mot kandidater/selskaper, og databehandling blir da enklere.
- Lettere for kandidater og kunder å ta i bruk systemet ettersom det er lettere tilgjengelig.

Med bruk av de samme teknologiene som WA bruker blir det da enklere å integrere systemet, og lettere å videreutvikle/vedlikeholde.

3.3 VALG AV VERKTØY

På bakgrunn av den valgte løsningen vil mye av de sammen verktøyene og teknologiene fra wa.works bli brukt i prosjektet. Dette betyr at web-applikasjonen blir bygget i React og Typescript.

Under utvikling blir data lagret lokalt, men i et ferdig produkt blir den integrert i det eksisterende database systemet.

Til utvikling blir Visual Studio Code brukt pga. tidligere kjennskap. For planlegging er Pivotal Tracker og Trello brukt, og for design av ulike modeller er Figma og Lucidcharts brukt.

For å dele koden mellom gruppemedlemmene og oppdragsgiver ble GitHub brukt. Dette er et privat repository som er knyttet til WA bedriften.

3.4 PROSJEKTMETODIKK.

3.4.1 Utviklingsmetodikk

Gjennom hele prosjektet har medlemmene i gruppen faste møter to ganger i uken. På møtene går en gjennom hva som er gjort siden forrige møte, og hva som skal gjøres videre.

Det er satt opp faste møter med veileder på fredager. Hvis det er ønsket/nødvendig blir også oppdragsgiver invitert med på dette møtet. I løpet av møtet blir modeller, ideer eller spørsmål tatt opp. Store deler av utviklingen skjer på kontoret til WA, og det er der kontinuerlig og god kommunikasjon med CTO Andreas.

Utvikling skjer iterativt og det er tatt inspirasjon fra Scrum (Scrum.org n.å.). Som i Scrum er det laget en todoliste i Pivotal Tracker med arbeidsoppgaver som er sortert etter prioritet. Fra den prioriterte listen velges det ut oppgaver som til sammen anslås å ta en uke å fullføre, tilsvarende en sprint på en uke. Dersom alt blir ferdig kan en velge flere arbeidsoppgaver, og oppgaver som ikke er ferdig blir overført til neste sprint. Gjennomgangen av oppgavene som er gjort i løpet av uken og valg av nye oppgaver gjøres på slutten av hver uke. Produktet presenteres ved jevne mellomrom til WA for raskt å kunne tilpasse videre utvikling, tilsvarende en Sprint review. Derimot på grunn av kun to utviklere i teamet så er det fjernet noen deler fra scrum metodikken. Det er ikke satt noen scrum master, og daglige scrum møter blir ikke holdt hver dag. Utviklingen skjer nesten eksklusivt ved å jobbe sammen, og samtaler skjer derfor naturlig underveis. Etter som det utvikles iterativt og listen over hva som skal gjøres er prioritert, så vil alltid det mest kritiske gjøres først. På denne måten passer en på at selv om det er tids problemer på slutten, så vil de mest kritiske delene være fullført. Dette er grunnen til at gruppen har valgt å jobbe på en iterativ måte istedenfor fossefallsmetoden.

3.4.2 Prosjektplan

For generell planlegging av prosjektet har det blitt benyttet Gannt-diagram (Figur 3.4 i vedlegg). Diagrammet har to deler: en for utvikling, og en for rapporter og støttedokumenter. Den valgte utviklingsmetoden og usikkerhet rundt arbeidsmengden til deler av utviklingen gjorde det vanskelig å sette det i Gannt-diagrammet. Utviklingsarbeidet ble fortsatt tatt med i Gannt diagrammet, men mer som en overordnet plan. Utenom Gannt er det også laget en intern plan ved hjelp av

PivotalTracker. På grunn av agilt arbeidsmetode og at noen funksjoner har tatt mye mer tid enn ventet, har planen vært i konstant endring gjennom utviklingstiden.

3.4.3 Risikovurdering

Det ble tidlig i prosjektet laget en enkel risikovurdering for de vanligste og vanskeligste problemene som kunne oppstå i løpet av bachelorprosjektet. Problemene som ble funnet er i tabell 3.4.1 og satt inn sammen med problem detaljer, mulige tiltak og produkt av konsekvensene.

Tabell 3.4.1: Risikovurdering for bachelorprosjektet

	Hendelse/ Risiko	Årsak	Sannsynlighet	Konsekvens	Risiko-produkt	Tiltak
1	Bilde Sammenligning	Får ikke sammenlignet oppgave bildet og kode bildet	Høy (4)	Svært høy (5)	20	Høyt prioritert del av utviklingen må startes tidlig i utviklingsprosessen.
2	Få tak i testere (kandidater)	Dårlig tid / får ikke tak i nok interesserte	Høy (4)	Høy (4)	16	Starte tidlig med å finne potensielle testere
4	Kode input fungerer ikke (visuelt)	Klarer ikke utvikle systemet	Middels (3)	Middels (3)	9	Høyt prioritert del av utviklingen må startes tidlig i utviklingsprosessen.
5	Få ikke ferdiglaget en MVP	Trenger flere funksjoner eller oppfyller ikke alle krav	Middels (3)	Lav (2)	6	Jobbe agilt, kjøre tester og vise frem iterasjoner av produkter til oppdragsgiver
6	Sykdom	Sykdom	Middels (3)	Lav(2)	6	Følge FHI sine anbefalinger

7	Systemet blir ikke tatt i bruk	Systemet fungerer ikke, ineffektivt eller lite brukervennlig	Svært lav (1)	Svært høy (5)	5	Jobbe agilt, kjøre tester og vise frem iterasjoner av produkter til oppdragsgiver

3.5 EVALUERINGSPLAN

Evalueringen er viktig for å fastsette om det endelige produktet oppfyller alle kravene som er satt i starten av arbeidet. For å evaluere dette prosjektet vil gruppen benytte seg av bruker tester, samtaler med prosjekteier og en presentasjon av slutt produktet for prosjekteier. Bruker testene vil bestå av noen spørsmål til tester samt at testerne gjennomfører en test i systemet. Produkteiers evaluering vil være i form av kontinuerlig kommunikasjon og tilbakemeldinger, og til slutt ha et møte for å presentere det leverte produktet. I møtet blir alle funksjonaliteter presentert, og tilslutt er det en gjennomgang av koden for å sjekke kvalitet og mulighetene for videreutvikling. Detaljert gjennomgang av evalueringen og resultatet av evalueringen kan ses i kapittel 5 Resultat.

4 DETALJERT DESIGN

Dette kapitlet gjennomgår designet av produktet i mer detalj. Først vises det initielle wireframe designet, og så presenteres de ferdiglagde sidene. Etter dette blir de viktigste funksjonene i testsystemet detaljert beskrevet og fremvist.

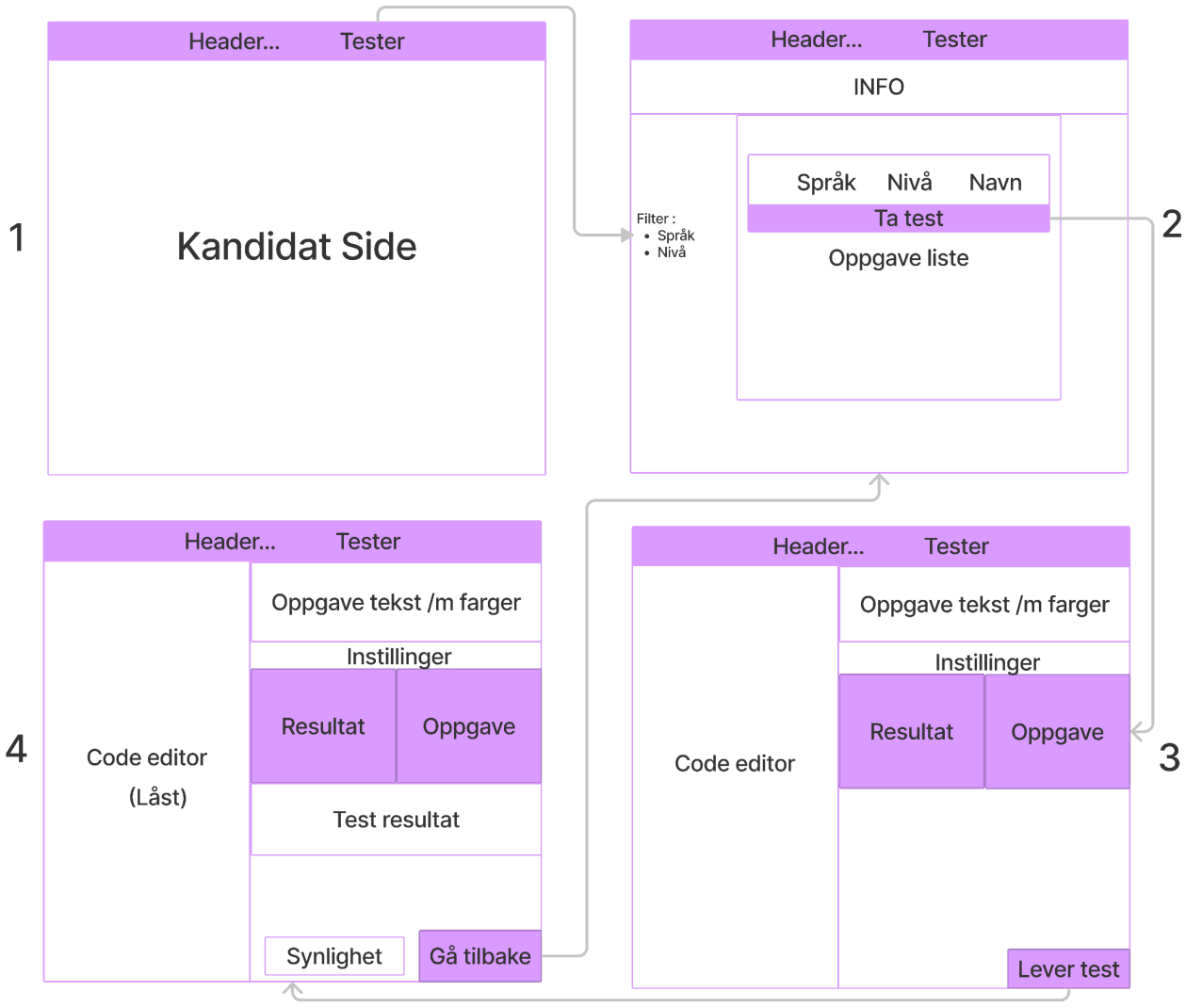
Noen av modellene og bildene fremvist i kapittel 4 har litt for liten tekst til å enkelt lese. For dem det gjelder er det vedlagt en forstørret versjon i vedlegg.

4.1 Wireframe design

Fra de initielle kravene er det forskjellige funksjonaliteter som er rettet mot om det er en kunde eller kandidat som bruker siden. Det er derfor valgt å lage to forskjellige sett med sider for å vise de forskjellige perspektivene. For å designe disse sidene ble det laget wireframe skisser for alle sidene, og piler for å vise navigasjonen mellom disse.

4.1.1 Design av kandidatsidene

Figur 4.1.1 viser wireframe designet til kandidat sidene. Hver side er en separat firkant, og i gjennomgangen av designet referer de forskjellige tallene til sidene i figur 4.1.1. Videre i kapittelet vil sidene i figuren bli referert etter hvilke side nr. de har.



Figur 4.1.1: Wireframe av kandidatsidene.

Den første siden viser hvordan en kommer fra wa.works plattformen til testsystemet gjennom navigasjonsmenyen på toppen.

Den andre siden er hovedsiden for testsystemet. Dette er siden som har listet opp alle de forskjellige testene en kandidat kan ta. Hvert element i listen viser en test sammen med relevant informasjon om testen. Rett under navigasjonsmenyen er det en liten infoboks som forteller hvordan testsystemet fungerer. På den venstre siden er det en filter komponent. Med denne kan kandidater filtrere listen etter kodespråk eller nivået på testene.

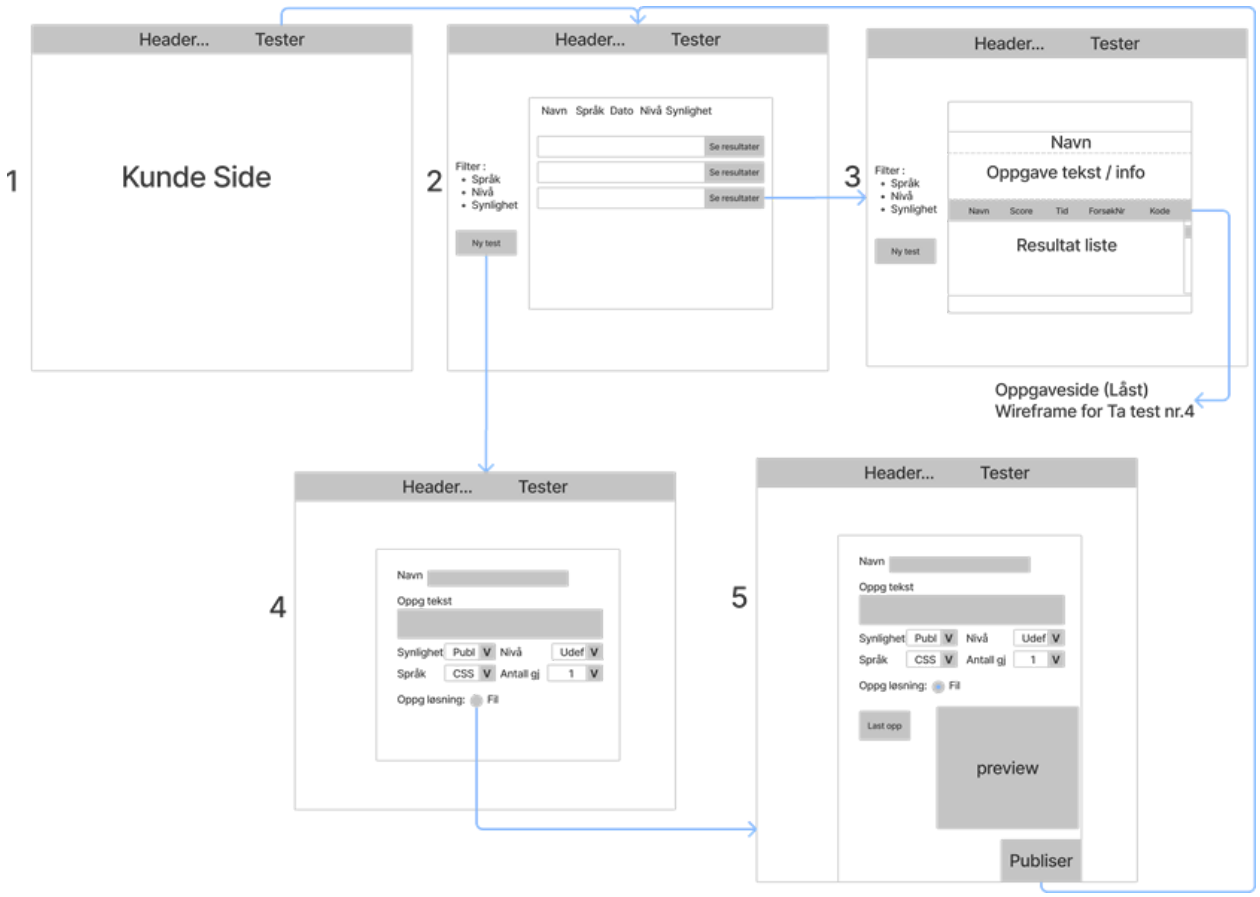
Etter å ha valgt på en test fra test listen kommer kandidaten til side tre. Dette er siden hvor kandidater kan gjennomføre den valgte testen. På venstre delen av siden er det en editor som kan blir brukt for å skrive inn kode. Når koden blir skrevet inn så vil resultatet vises live i resultatvinduet.

Etter å ha levert testen kommer en til side fire. Dette er ikke en ny side i seg selv, men en utvidelse av side tre. Nå som testen er levert blir kode editoren låst. Deretter blir likheten mellom oppgavebildet og den leverte løsningen funnet. Resultatet fra testen vises under bildet sammen med antall tegn og tiden som er brukt. Før en går tilbake til hovedsiden kan en velge om testresultatet skal være offentlig eller privat. Om den er offentlig kan kunden som eier testen se hvem som har fått denne score, og kan koble resultatet til en wa.works profil.

Testsiden som er vist er spesifikt designet for CSS oppgaver, og hadde ikke fungert for f.eks. Javascript tester. Siden er designet modulært, og dette gjøre det enkelt å bytte hvilke “evalueringskomponent” som blir vist. For Javascript kunne f.eks. de forskjellige bildene endres til å vise en konsoll som viser resultatet etter å ha kjørt koden.

4.1.2 Design av kundesidene

For kundesidene ble det også laget en wireframe for å utforske designet til sidene. Som i wireframen til kandidatsidene er figur 4.1.2 delt inn i flere firkanter som representerer de forskjellige sidene, og disse kommer heretter til å bli referert etter hvilke side nr. de har.



Figur 4.1.2: Wireframe av kundesidene.

Som med figur 4.1.1 er den første siden i wireframen fra figur 4.1.2 en peker til testsystemet fra den eksisterende wa.works plattformen. Dette er gitt at en er logget inn som en kunde, og ikke som en kandidat.

Etter å ha navigert seg til testsiden med navigasjonsmenyen, kommer kunden til en liste av tester det innloggede selskapet har publisert. På samme måte som på kandidatsiden kan disse testene filtreres ut fra nivået og kodespråket som er i bruk. Kunder kan også filtrere etter synligheten på oppgaven, og under filteret er det en knapp for å legge til nye tester.

Når en velger en test fra listen på side 2 blir en sendt til side 3. Dette er ikke en ny side i seg selv, men en modul som vises over den eksisterende siden. Modulen viser den fullstendige informasjonen til oppgaven, og under er det listet opp resultatene til de forskjellige kandidatene som har tatt testen. Resultater fra kandidater som ikke ønsker at

resultatet synlig dukker også opp her, men resultatet er anonymisert og kunden kan bare se resultatet. Videre fra dette så kan kunden trykke seg inn på hver enkelt test for å se hele løsningen til kandidaten. Kunden ser da side 4 i figur 4.1.1, men med små endringer som f.eks. at en ikke kan endre synligheten til testen. Hvis testen selv ikke er synlig så er det muligheter her til å dele testen med en link. Dette er også mulig for tester som er synlige.

Tilbake til hovedsiden kan kunder lage nye tester ved å trykke på «Ny Test» knappen under filteret. Dette sender deg videre til side 4, som har input for all informasjonen som er nødvendig for å lage en ny test. Ut fra hvilke språk som er valgt blir kravene for løsningen endret. For en CSS-test vil oppgaven være et bilde av det som skal bli replikert, men for Javascript vil dette f.eks. være ulike tester for å se om oppgaven er løst.

Når oppgave bildet er valgt vil bildet bli vist som en bekreftelse før oppgaven blir lagret. Dette vises i side 5, som er en dynamisk endring fra side 4. Når oppgaven blir publisert blir kunden sendt tilbake til listen av tester, og kan da se statistikken for den nye testen.

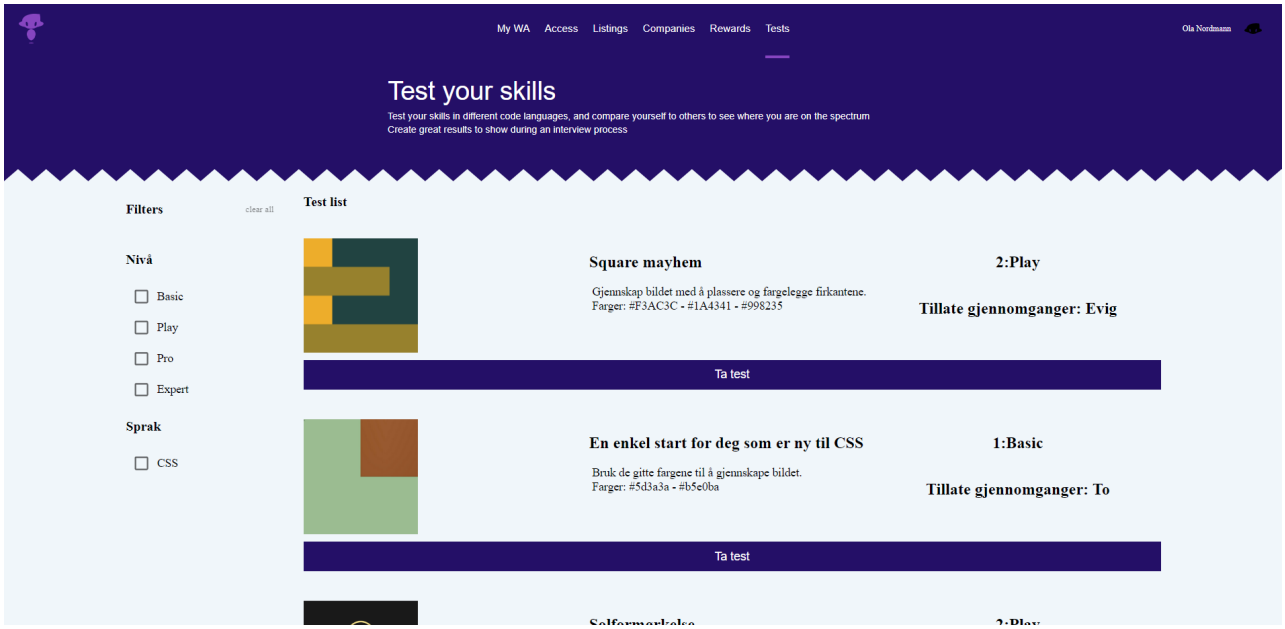
4.2 Sider

En viktig grunn for teknologi valgene som ble diskutert i kapittel 3.3 var at eksisterende komponenter fra wa.works plattformen kunne brukes for å lage testplattformen. Dette vises i de aller fleste sidene som er laget for testplattformen, og fungerer som en demonstrasjon at mange komponenter kan gjenbrukes for å lage testsystemet.

Felles for alle sidene er navigasjonsmenyen på toppen, og ut fra hvem som er logget inn skifter den hvilke linker som er tilgjengelig på toppen. Det er også felles hvilket tema som blir brukt på sidene. Det er et spesifikt tema for kandidater og kunder som er hentet fra wa.works plattformen.

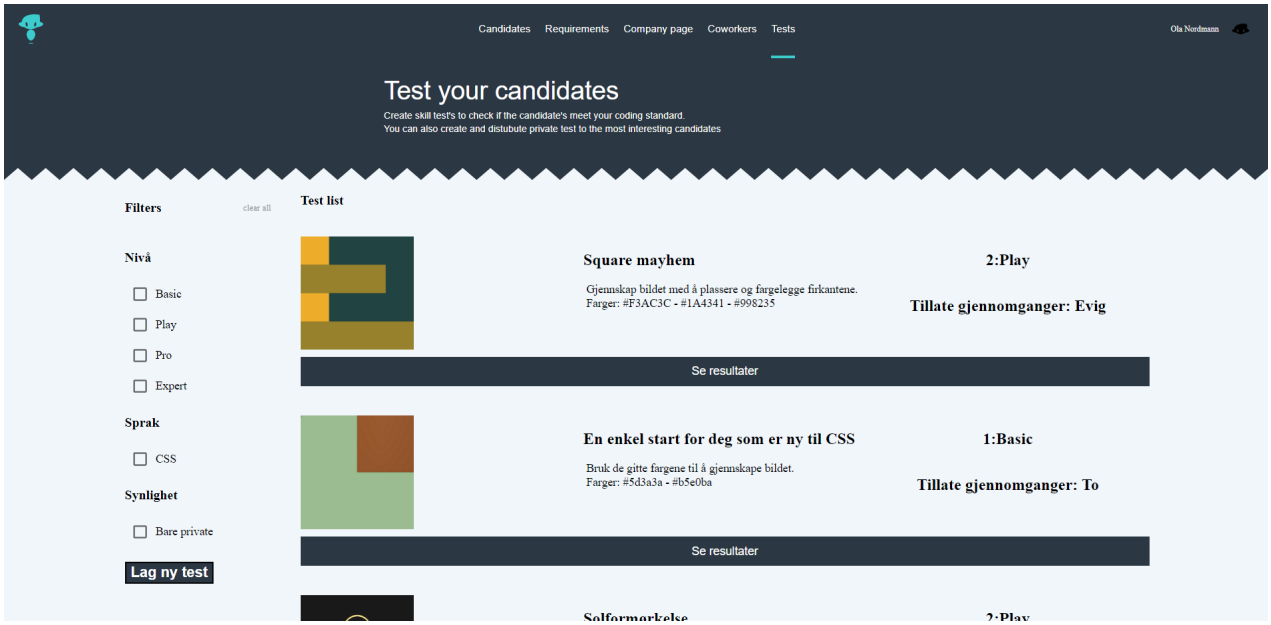
4.2.1 Test liste

Bilde 4.2.1 er et skjermbilde av kandidatsiden hvor tester kan bli valgt, og videre i kapittelet blir funksjonaliteten beskrevet.



Bilde 4.2.1: Skjermbilde av siden som viser alle testene.

Test liste siden er en viktig del av testsystemet, og er hovedmåten kandidater finner og begynner tester fra. Designet av denne siden har tatt stor inspirasjon fra en eksisterende side på wa.works plattformen, og det er gjenbrukt komponenter fra denne.

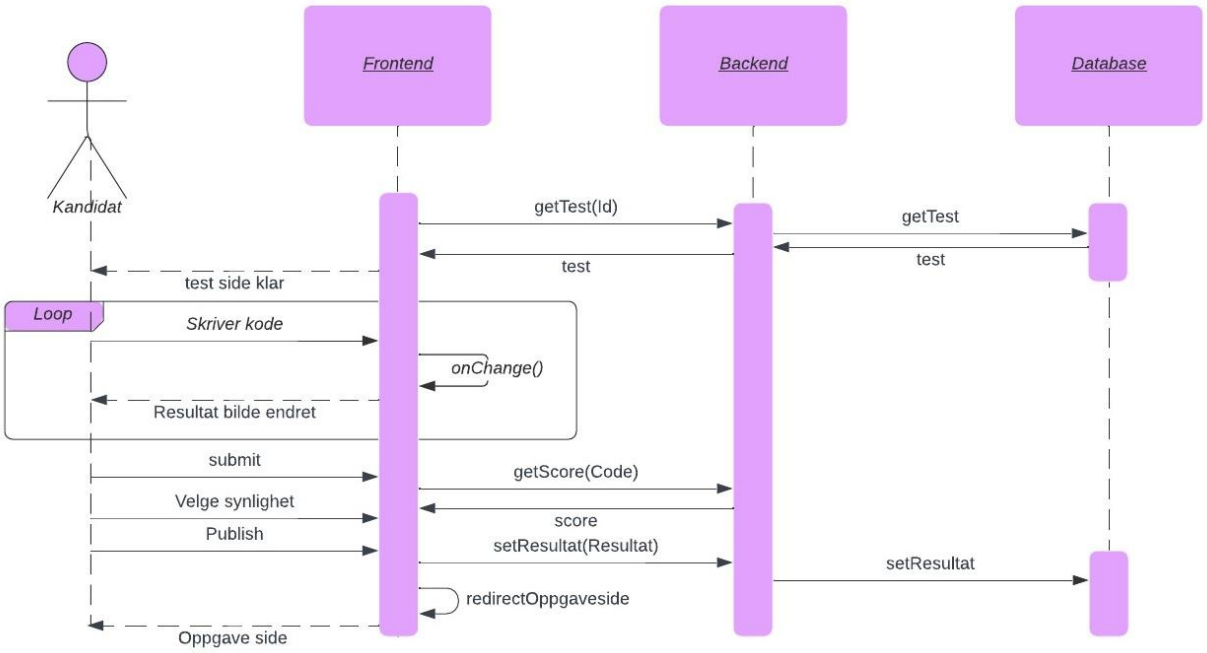


Bilde 4.2.2: Skjermbilde av tester som en kunde har publisert.

Sammenligner du kandidatsiden med kundesiden (bilde 4.2.2), er likheten mellom dem veldig stor. Det er også enkelt å se at kundesiden ikke ligner på konseptet som ble introdusert i wireframe designet som ble beskrevet i kapittel 4.1.2. Dette er et designvalg som ble gjort for å kunne gjenbruke komponentene som ble laget for kandidatsiden. Faktisk er kandidatsiden og kundesiden den samme, men temaet og litt av teksten er forskjellig. Dette er basert på hvilke «context» som er valgt i applikasjonen, og dette systemet er beskrevet i mer detalj i kapittel 4.3.

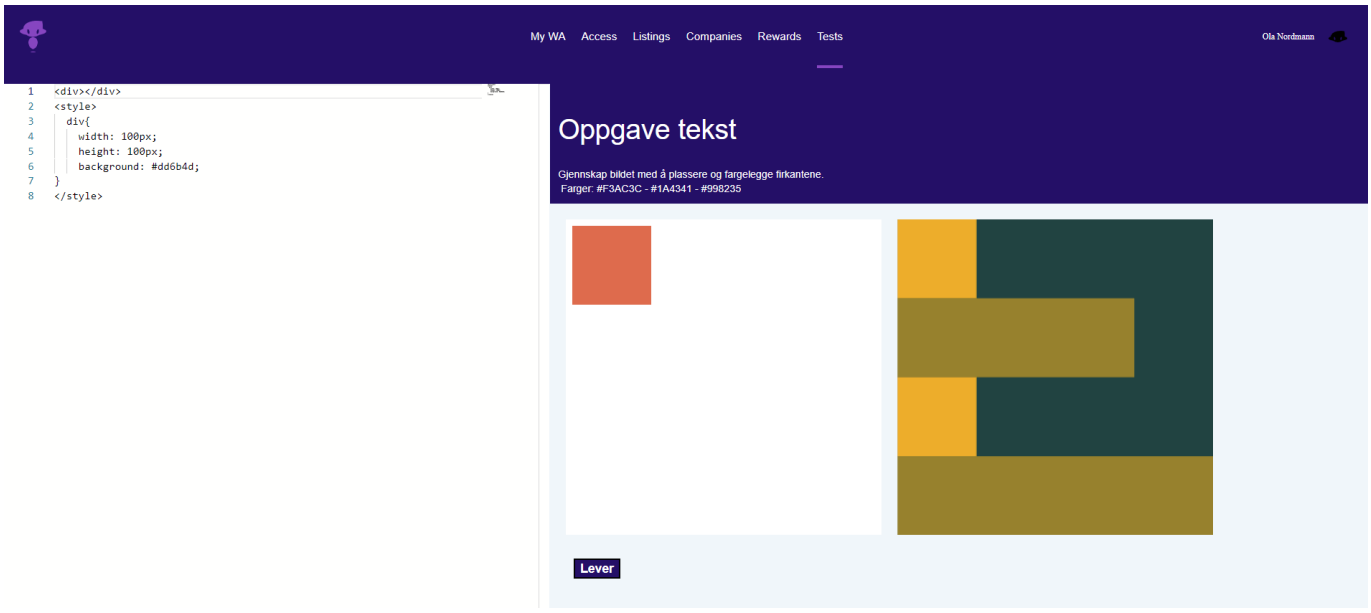
4.2.2 Test

Sekvensdiagrammet i figur 4.2.1 viser hva som skjer etter at det er blitt valgt en test fra test listen. I dette kapitlet går vi gjennom det å gjennomføre en test. Merk at figur 4.2.1 også viser kallene som skulle blitt gjort til APIet og databasen, men dette er ikke blitt implementert i det leverte systemet. Istedenfor dette er det brukt en «store» som inneholder dataen som skulle kommet fra de forskjellige API-kallene. Denne lagringen er videre beskrevet i kapittel 4.5.



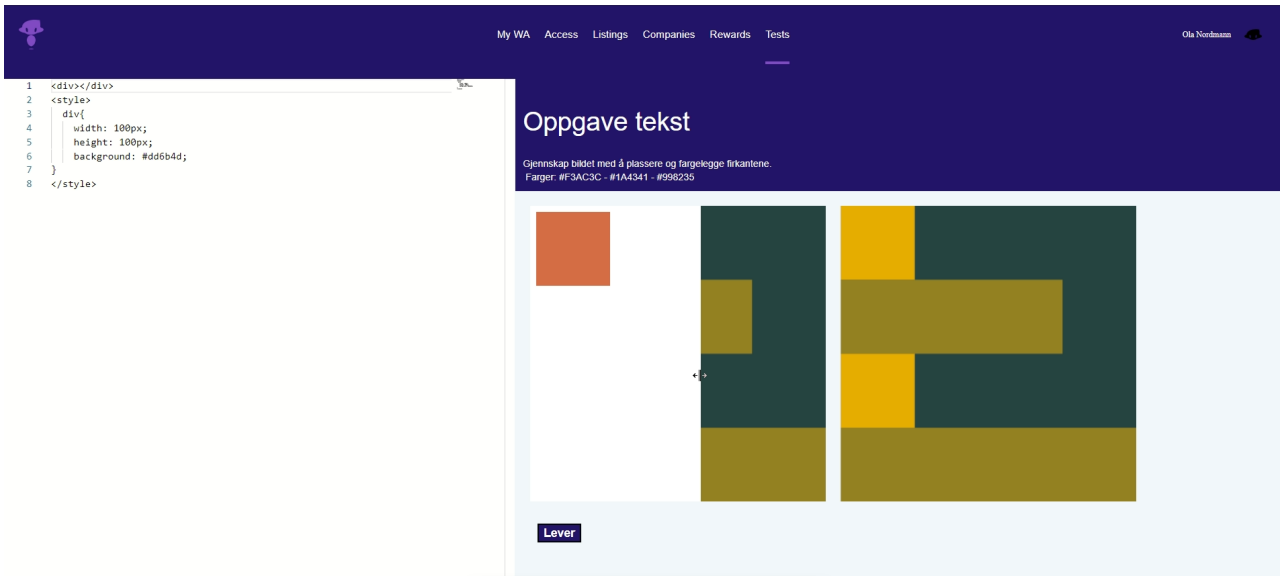
Figur 4.2.1: Sekvensdiagram for å gjennomføre en CSS-test

Eksemplene videre i kapitlet og skjermbildene i kapitlet er etter at en kandidat har valgt den øverste testen sett i bilde 4.2.1.



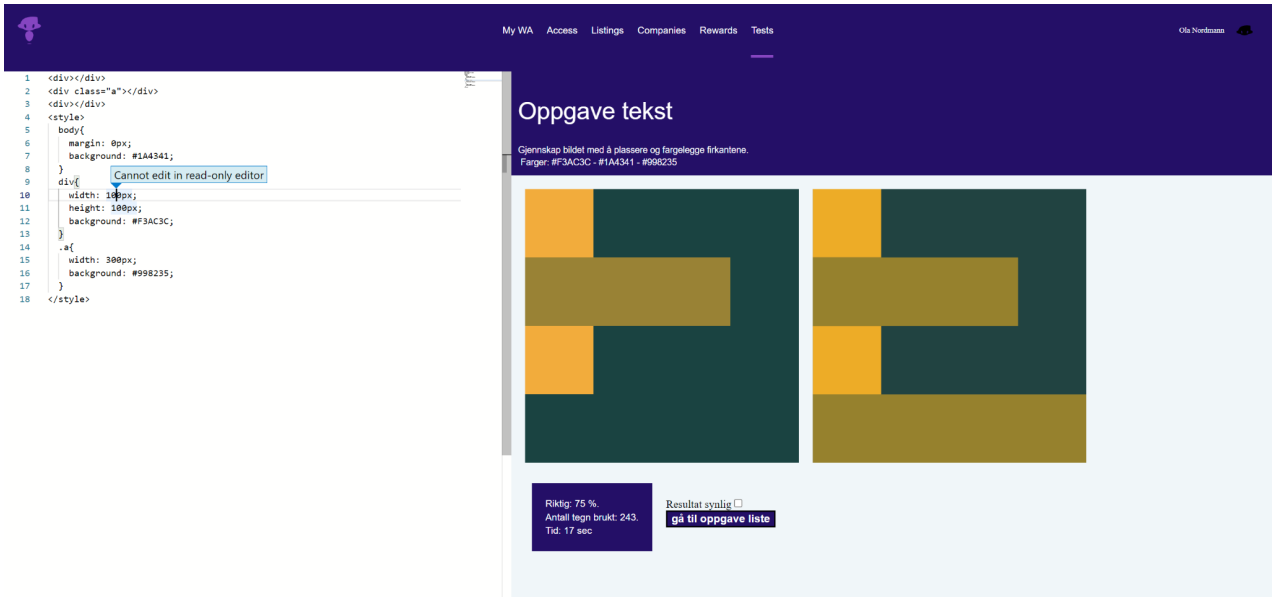
Bilde 4.2.3: Skjermbilde av en nylig startet test.

Bilde 4.2.3 viser test siden rett etter at kandidaten er sendt hit. Til venstre er kode editoren med noe CSS kode. Startkoden er lagt til for å demonstrere hvordan systemet fungerer. Til venstre for oppgavebildet vises resultatet fra koden som blir sendt inn fra editoren. Kode editoren er et node-bibliotek, og tilpassingen og koden blir mer beskrevet i kapittel 4.4.1.



Bilde 4.2.4: Skjermbilde av testsiden mens slideren blir brukt.

Når testen blir utført kan det til tider være vanskelig å se om forskjellen mellom koderesultatet, og oppgavebildet som skal gjenskapes. For å gjøre dette enklere er det derfor implementert en måte å «slide» oppgavebildet over koderesultatet. I bilde 4.2.4 er dette demonstrert, og forskjellen mellom resultatet og oppgavebildet kan lettere bli sett. Funksjonaliteten, designet og koden til slideren er detaljert beskrevet i kapittel 4.4.2.



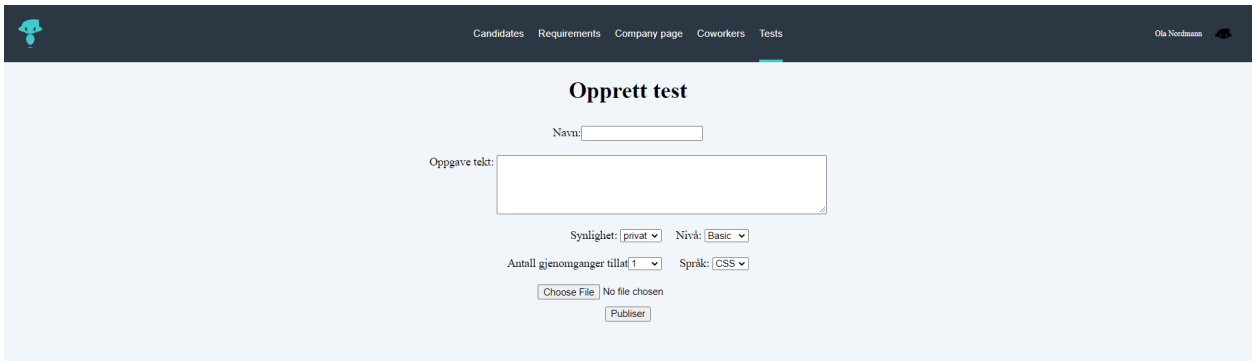
Bilde 4.2.5: Skjermbilde av testsiden, etter at testen er levert.

Når kandidaten er fornøyd med oppgaven kan den bli levert. Etter dette vises resultatet av bilder sammenligningen, tiden og antall tegn brukt under bildene. Bildesammenligningen blir detaljert beskrevet i kapittel 4.4.3.

Nå som testen er levert er det ikke lenger mulig å endre på koden. Dersom det blir gjort forsøk på endringer vil det komme opp en info-melding, og editoren godtar ingen endringer.. Hvordan editoren gjør denne endringen blir beskrevet i kapittel 4.4.1.

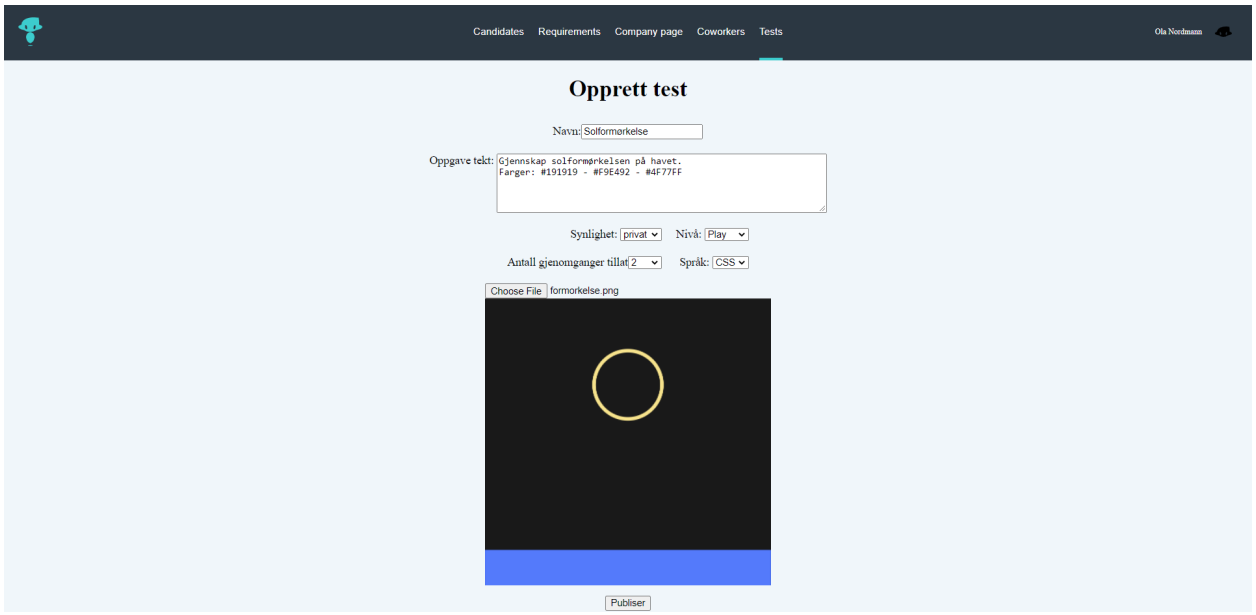
4.2.3 Lag ny test

En viktig funksjon for kunder er å kunne lage egendefinerte tester. Bilde 4.2.6 viser siden for dette.



Bilde 4.2.6: Skjermbilde av ny test siden.

På denne siden kan en sette inn all informasjonen som trengs for å lage en ny test. En av disse er å velge hvilket testspråk som skal brukes. Ut fra hva som blir valgt her vil den forventede løsning også bli endret. I bilde 4.2.6 er CSS valgt, og det er da et bilde som skal bli lastet opp.



Bilde 4.2.7: Skjermbilde av en utfylt ny test side.

Bilde 4.2.7 viser det utfylte skjemaet for å kunne lage en ny test. Her er det blitt lagt til et nytt bilde, ved hjelp av en input tag av type file. Dette er et bilde som blir vist frem som en preview. Når denne testen blir levert blir all informasjon lagret i et nytt test objekt, inkludert bildet som blir lagret som en string se Bilde 4.2.8 under.

```
const reader = new FileReader();
reader.addEventListener( type: "load", listener: ()=>{
    bilde = reader.result! as string;
```

Bilde 4.2.8 opplasting av bilde

4.3 Context

Tidligere i kapittel fire har vi referert til «context» i form av å bytte mellom kandidat- og kundesidene. Context er en string som blir sendt til de forskjellige komponentene som er avhengig av å vite hvilke type bruker som er logget inn. F.eks. trenger navigasjonsmenyen å vite om det er en kandidat eller kunde som er logget på siden.

Bilde 4.3.1 viser hvordan denne får context verdien for å kunne bestemme hvilke tema og linker som skal vises.

```
<Menu context={context} />
```

Bilde 4.3.1: Meny komponenten blir gitt en context prop

I testsystemet fungerer context som en midlertidig funksjon for å bytte kandidat og kunde. Når det senere blir integrert vil context fortsatt bli brukt, men er da støttet av innlogging. Så en kandidat kan ikke manipulere context verdien for å få tilgang til kunde funksjoner.

4.4 Funksjoner

Delkapitlene under går inn på forskjellige funksjoner som er presentert, men som ikke er beskrevet i detalj tidligere i kapitlene.

4.4.1 Kode editoren

For å gjøre utviklingen enklere ble tidlig bestemt at det var urealistisk å lage en egen kode editor, og det ble valgt å finne en open-source komponent som kunne brukes. Tilslutt ble node-biblioteket Monaco valgt. Denne er utviklet av Microsoft (Microsoft, u.å.). Dette er den samme IDE-editoren som blir brukt i VS Code, og har god støtte for språk som kan brukes og den har IntelliSense. Monaco er under MIT lisens, og er derfor fri til å bruke i dette prosjektet og videre av oppdragsgiver (Opensource.org, u.å).

```
<Editor
  height="90vh"
  defaultLanguage={props.sprok}
  defaultValue={defaultValue}
  options={{readOnly: props.levert}}
  onChange={handleEditorChange}
/>
```

Bilde 4.4.1: Editor komponenten fra testsiden.

Bilde 4.4.1 vises konfigurasjonen til Monaco-editoren. DefaultValue er verdien til startkoden som vist i bilde 4.2.3.

Koden som blir skrevet inn i editoren blir behandlet med onChange metoden som er definert i Monaco konfigurasjonen. Denne metoden sender koden fra editoren til resultatbildet som viser resultatet. Det blir brukt en IFrame tag for å vise resultatet fra koden som blir sendt inn.

4.4.2 Bilde slider

Som vist i bilde 4.4.2 er det en slider på resultatbildet for å sammenligne oppgavebildet og resultatbildet. Dette blir gjort ved at resultatbildet blir fjernet fra høyre til venstre ut fra hvor musepekeren er på bildet. Dette kan bli gjort fordi under resultatbildet ligger oppgavebildet, og det kommer frem når resultatbildet blir kuttet vekk.

```
const mouseOver = (e:any) =>{
    const cutAt :number = e.clientX - e.target.getBoundingClientRect().left;
    setEndring(cutAt);
}
```

Bilde 4.4.2: Metoden som blir kjørt med mouse-over på resultatbildet .

Variabelen cutAt har den utregnede verdien til hvor slideren skal vise oppgavebildet til. Denne verdien blir så satt inn i setEndring som oppdaterer state-verdien. Denne state

oppdateringen gjør at komponenten blir tegnet på nytt, og derfor kan vi se bilde bevege seg.

4.4.3 Bildesammenligning

Sammenligning av resultatbildet og oppgavebildet er en viktig del for å vurdere koden som blir sendt inn. For å sammenligne bildene er PixelMatch blitt brukt (mapbox, u.å). PixelMatch er under ISC lisens, og er derfor også fri til å bli brukt (Opensource.org, u.å). Pikslene fra resultatbildet og oppgavebildet blir sammenlignet med hverandre, og returnerer hvor mange av disse som er like. Siden bildene er 400x400, er en perfekt score 160 tusen. Resultatet som blir gitt til brukeren er en prosent av hvor mange av pikslene som er riktige.

```
const hoyde = 400;
const lengde = 400;

const img1Context = imgCanvas.getContext( contextId: "2d")!;
const img2Context = imgCanvas2.getContext( contextId: "2d")!;
const imgCanvas3 = document.createElement( tagName: "canvas");
const diffContext = imgCanvas3.getContext( contextId: "2d")!;

const img1 = img1Context.getImageData( sx: 0,  sy: 0, lengde, hoyde);
const img2 = img2Context.getImageData( sx: 0,  sy: 0, lengde, hoyde);
const diff = diffContext.createImageData(lengde, hoyde);

const num = Pixelmatch(img1.data, img2.data, diff.data, lengde, hoyde);
```

Bilde 4.4.3: Koden som sammenligner bildene

Bilde 4.4.3 viser koden som brukes for å sammenligne bildene. For å sammenligne bildene ved hjelp av Pixelmatch må riktig data først hentes fra bildet. Dette gjøres ved å bruke html2canvas som er under MIT License (Von Hertzen, u.å; Opensource.org, u.å.). Iframe elementet må hentes ut først ettersom html2canvas i utgangspunktet ikke kan hente data fra iframe, se kode på Bilde 4.4.4. Canvas elementet er hentet ved hjelp av html2canvas, og har da den nødvendige dataen for å sammenligne bildene.

```
let iframeDocument = document.getElementsByTagName( qualifiedName: "iframe")[0]
    .contentDocument?.documentElement!;
html2canvas(iframeDocument).then(imgCanvas => {
```

Bilde 4.4.4 henter ut HTMLElementet til iframe for å bruke i html2canvas

For kodetestene er det alltid en risiko ved å kjøre ukjent kode på maskinen. Med CSS er det ikke så enkelt å utføre et dataangrep, men i fremtiden med andre kodespråk kan dette bli et problem. Det er derfor valgt å regne ut resultatet av oppgaven i front-end, og validere dette i back-end. Det er fortsatt noen sikkerhetsrisikoer med å gjøre dette, og disse blir videre diskutert i systemdokumentasjon, kapittel 7.

4.5 Intern lagring og API

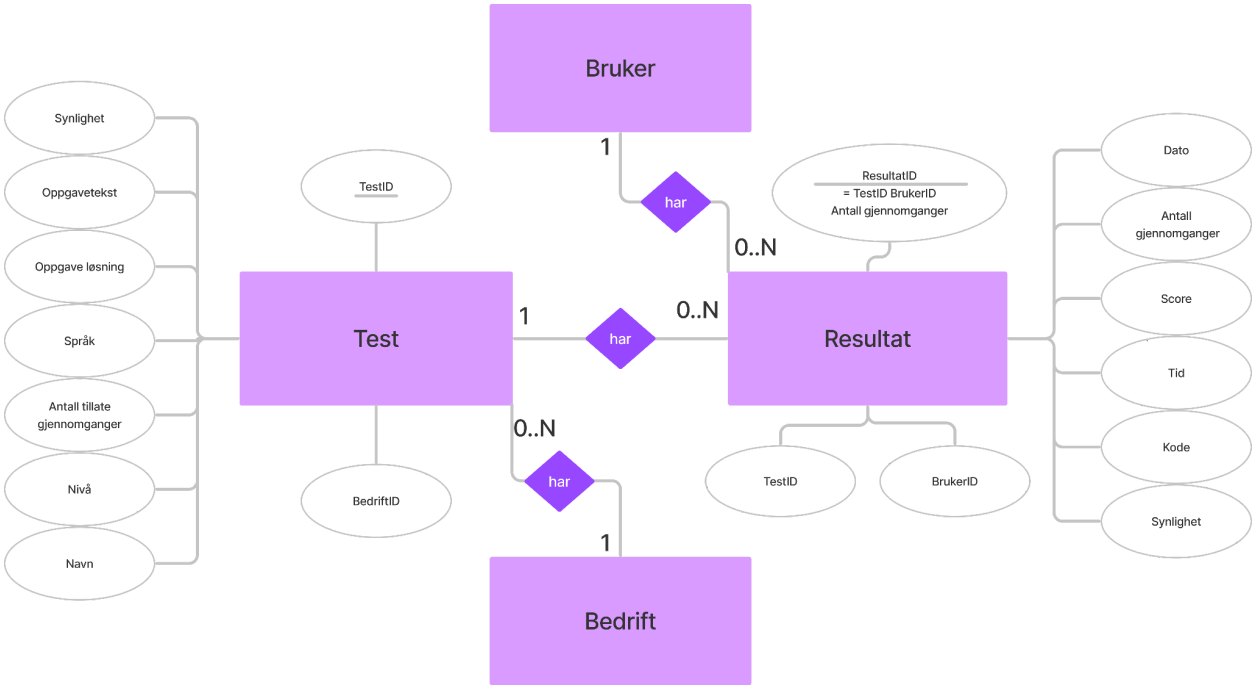
For intern lagring er Redux brukt. Redux er en state manager for JavaScript programmer, og det er her de forskjellige verdiene som blir hentet fra API-en blir lagret. I det leverte produktet er det ikke noen API å koble til, så applikasjonen kjører med lagrede standardverdier for tester. Dette fungerer fint for å teste applikasjonen, og det er enkelt å utvide redux storen til å gjøre et API kall for å skaffe dataene. Bilde 4.5.1 viser konfigurasjonen av Redux som er brukt i applikasjonen.

```
export const store = configureStore( options: {
    reducer: {
        [testResultSlice.reducerPath] :testResultSlice.reducer,
        testList: testListSliceReducer
    },
    middleware :getDefaultMiddleware => {
        return getDefaultMiddleware()
            .concat(
                testResultSlice.middleware,
            );
    }
});
```

Bilde 4.5.1: Konfigurasjonen av Redux store

4.6 Database design

Den leverte løsningen har ikke database tilkobling, men det ble tidlig i prosjektet laget et ER-diagram(figur 4.6) som skulle brukes for å lage databasen. På grunn av omprioriteringer ble denne aldri laget, men er blitt brukt internt til å lage test- og resultat objekter. Dette gjør det enklere i fremtiden å integrere databasen siden applikasjonen er designet rundt tabellene som er i databasen.



Figur 4.6: ER-diagram for databasen

5 RESULTATER

Dette kapitlet går inn på hvordan prosjektet blir evaluert, og resultatet av evalueringen. Deretter går det i dybde på resultatet og gjennomførelsen av prosjektet.

5.1 Evalueringsmetode

For å evaluere om det leverte produktet har oppnådd de initielle kravene som ble satt i kapittel 2.1.2 må produktet bli vurdert på forskjellige måter. Delkapitlene under beskriver hvilke metoder som er brukt for å sjekke dette.

5.1.1 Evalueringsmetode for brukere

Evalueringen fra brukere ble gjennomført ved at brukeren først ble forklart hvordan wa.works fungerer, og testerne besvarte et spørsmål før de begynte:

- Hvilket nivå vil du si at du har i CSS?

Bruker fikk så velge test fra testlisten. Etter gjennomført test svarte bruker på disse spørsmålene:

- Hvordan opplevde du testen?
- Ville et slikt system kunne hjelpe deg til å enklere finne nivået ditt i CSS og andre programmeringsspråk?
- Ville du endret CSS nivået ditt nå?

5.1.2 Evalueringsmetode for WA

Gjennom hele prosjektet har det vært kontinuerlig kommunikasjon med prosjekteier, men prosjektet i sin helhet har ikke blitt presentert før evalueringen. Vurderingen av det ferdige produktet ble gjort i form av et møte hos WA, hvor produktet ble vist frem i sin helhet. Etter dette blir også kildekoden gjennomgått for å vise frem hvordan de viktigste funksjonene fungerer. Etter denne evalueringen svarte prosjekteier på disse spørsmålene:

- Oppfyller produktet kravene til oppdragsgiver?
- Er oppgaven fullført?
- Enig med avvik fra original plan?
- Hva er planen for prosjektet videre?
- Generelt tilbakemeldinger på oppgaven, og hva som kunne vært gjort annerledes.

5.2 Evalueringsresultat

I dette kapittelet blir resultatet fra de forskjellige evalueringsmetodene presentert.

5.2.1 Evalueringsresultat for Brukere

Antallet brukere som testet systemet var svært få. Mer detaljer om dette i kapittel 6 Diskusjon. Det ble til slutt gjennomført test med kun tre brukere. Alle brukerne kjente allerede til systemet til WA. Fra samtlige brukere kom det fram at de ikke følte seg trygge på CSS og hadde jobbet svært lite med dette.

Resultatet fra evalueringen var positiv og alle brukerne så for seg at det ville være enklere å velge et nivå etter testing på denne måten. Det var derimot ingen av brukerne ville endret nivået sitt etter test. At ingen av testerne ville endret nivået etter test kan skyldes at de hadde lite erfaring med CSS, og var klar over dette på forhånd.

En av testerne kommenterte at for å hjelpe opp mot vurdering av nivå, så måtte han nok ha tatt ganske mange tester. Og at testene derfor måtte vært interessante nok til å motivere han til å gjøre dette. Det ble også nevnt at å sammenligne resultater med andre kan hjelpe å fastsette nivået sitt.

5.2.2 Evalueringsresultat for WA

Resultatet av evalueringen til oppdragsgiveren gir et grunnlag for å si om applikasjonen oppfyller kravene som ble satt i starten av prosjektet. Gjennom evalueringsmøtet ble det raskt fastsatt at de aller fleste krav var fullført, men at muligheten for lagring av resultater ikke var ferdig. Gjennom tidligere samtaler med oppdragsgiver har det vært enighet i å ha et hovedfokuset var på front-end løsningen. Det ble konkludert med at oppgaven var gjennomført, og at den oppfyller ønskene til oppdragsgiver.

Et annet krav hvor løsning ble endret underveis var hvordan poeng blir utregnet. Løsningen for dette blir videre diskutert i kap. 6 Diskusjon.

I evalueringsmøtet ble flere positive tilbakemeldinger gitt. F.eks:

- Når oppgaven er levert kan en se tid og antall tegn som ble brukt.
- Slider funksjonen for å sammenligne resultat bilde som brukeren lager og oppgave bildet.
- At applikasjonen var stylet til å ligne den eksisterende plattformen deres wa.works.

Noen tilbakemeldinger på mulige forbedringer på det leverte produktet var:

- Ønske om å ha en start knapp på test siden for å kunne starte testen. Ikke bare automatisk startet med en gang en er inne på siden.
- Vise en klokke med brukt tid under hele gjennomføringen av testen, og ikke bare når oppgaven er levert.

WA har planer om å videreutvikle ideen med flere språk, og tilslutt implementere dette i den eksisterende rekrutteringsplattformen wa.works.

5.3 Prosjektresultat

Prosjektet har resultert i et fungerende testsystem for CSS, men uten lagring i database. Systemet viser lister over tester, og brukerne av systemet kan opprette og gjennomføre tester for CSS. Når en person har gjennomført en test får han/hun ut brukt tid og poeng. Visuelt er siden satt opp ganske likt som eksisterende system hos WA. Kode editoren som er brukt kan enkelt brukes i videre utviklingen hos WA, ettersom den kan brukes til flere språk. Resultatet har noen mangler innenfor lagring av data siden databasen ikke er satt opp. Selv om databasen ikke er satt opp er det blitt laget et ER diagram (Figur 4.6) og sekvensdiagrammer (Figur 4.2.1, Figur 5.2) som viser hvordan databasen skal settes opp. Selv om brukertesting var svært begrenset, så er det en indikasjon på at et testsystem kan hjelpe kandidatene på wa.works.

5.4 Prosjektgjennomføring

Sett fra gruppens perspektiv er prosjektet en suksess. Produktet er et bra grunnlag for videreutviklingen som er planlagt fra WAs side. Samarbeid har fungert bra innad i gruppen og med oppdragsgiver. Det var derimot flere funksjonaliteter som originalt var tenkt å bli implementert som ikke ble det. Gruppens erfaring med så store prosjekt som dette var svært begrenset og noen elementer tok mye lenger tid enn forventet. Mer om dette i kap.6 Diskusjon. Det har vært svært vanskelig å få tak i testere. Noe som ender opp med at det er vanskelig å konkludere om systemet vil kunne hjelpe å vurdere kandidaters nivå.

6 DISKUSJON

Kapittelet presenterer og diskuterer de viktigste delene av prosjektet. Dette for å gi en dypere innsikt i hvordan bachelor prosjektet har blitt gjennomført, og ulike vanskeligheter som har oppstått.

Et problem som oppstod var å finne ut hvordan resultatet for en test skulle beregnes. Den originale planen var at brukeren skulle få tildelt en score basert hvor mange ord som ble brukt for å løse oppgaven, og sammenligningen mellom resultat- og oppgavebildet. Problemet med dette er hovedsakelig hvordan en regner på antall tegn/ord som ble brukt. Dersom vi regner scoren ut fra mengden tegn, blir resultatet potensielt kode som er lite lesbar. Regner man på antall ord kan en bruker bare fjerne alle mellomrom og ha alt på en linje for å få full score, uavhengig av mengden brukte ord. Begge disse resultatene er noe en ikke ønsker i rekrutteringssammenheng. Det ble derfor bestemt å vise tid, antall tegn og prosent score separat fra hverandre og ikke som en utregnet score. På denne måten kan kunden se koden for å vurdere resultatet selv. I evalueringen av prosjektet ble det foreslått å bruke minify for så å regne ut antall tegn i koden, men dette har ikke blitt implementert eller testet. For tester i andre språk kan det være lurt å se på tiden det tar for koden å kjøre, men dette er ikke mulig å gjøre i dette tilfellet.

Det var noen aspekt med oppgaven som tok mer tid enn forventet. Spesielt koden for å sammenligne de to bildene tok mye tid. Først måtte en finne en metode for å sammenligne to bilder som returnerte et nummer som kunne brukes videre til å regne ut score. Etter en del arbeid ble det bestemt å bruke pixelmatch. Utfordringen videre var å få konvertert bildene til riktig format for bruk i pixelmatch. Oppgave bildet kunne konverteres på flere måter, fra github siden til pixelmatch var følgende måte vist (bilde 6.1 i vedlegg). Resultat bildet var et større problem etter som det i utgangspunktet ikke var et bilde, men en iframe. Etter mye tid ble en løsning tilslutt funnet. Løsningen var å bruke html2canvas. Html2canvas gjør om elementene fra resultat bilde og oppgave bildet til canvas elementer. Dette gjør at løsningen ikke er avhengig av filformatet til oppgavebildet eller plasseringen av bildene i nettleseren. Løsningen gjør nettsiden enkel å vedlikeholde og koden kan enkelt brukes andre steder om dette skulle være ønskelig.

Sammenligningen av bildene var helt essensielt for testsystemet, og det hadde ikke fungert uten denne funksjonen. Siden bildesammenligningen tok mer tid enn antatt gjorde det at andre ting ble nedprioritert.

På grunn av nedprioriteringer endte det opp med å begrense det leverte produktet. Nesten alle kravene som ble satt i starten er oppfylt, men det er noen interne mål som ikke er det. I kapittel 4.1.2 ble designet for kundesidene presentert, og denne har noen funksjoner som ikke var en del av kravene. Disse funksjonene ble ikke prioritert ovenfor de originale kravene, men hadde vært grei å ha for å nærmere vise hvordan testsystemet kan bli. De forskjellige endringene i planene er diskutert med produkteier underveis, og de syntes vi har prioritert riktig med det vi har fått gjort. Ved sluttevalueringen savnet de hovedsakelig lagring av resultater. Mye av arbeidet for dette er allerede gjort, og det var ikke høyt prioritert.

Visjon- og kravdokumentet ble laget tidlig i prosjektet for å fastsette hva som skulle gjøres. Nå som prosjektet er ferdig, kan en se at ikke alt i disse dokumentene stemmer med det som er levert. Som nevnt tidligere har det vært litt tidspress, og dette har forårsaket noen endringer i hva som ble levert. Både krav- og visjonsdokumentet fungerte bra for å starte prosjektet, men er ikke egnet bra for å vurdere hva prosjektet er nå. De egner seg derimot bra for å se det originale kravet og visjonet for prosjektet. F.eks. er visjonsdokumentet skrevet som at testplattformen er integrert i wa.works, og noen av de funksjonelle egenskapene som er beskrevet er avhengig av dette. Wa ønsker derimot ikke å integrere systemet før det er lagt til flere testbare språk. Merk at modellene som er presentert i kravdokumentet er fortsatt oppdatert ettersom de er brukt andre steder i prosjektet.

Tidsbegrensninger skapte også problemer rundt testing. Siden sammenligningen er så kritisk, kunne ikke systemet testes før den var implementert. Et annet problemet er at systemet er laget for å teste personer som kan kode. De personene som gruppen kjenner til som kan kode er svært travle med egen bacheloroppgave og eksamener. Dette har ført til for få testere og testerne som gikk med på å delta hadde lite erfaring med CSS. Med mer tid hadde det vært interessant å se utviklingen etter systemet var implementert på nettsiden. Med intervjuer av kandidater og kunder før og etter implementeringen hadde en kunne evaluert om systemet blant annet hjelper mot feilrekruttering.

Designet av nettsiden ble tidlig bestemt og presentert til oppdragsgiver, og en viktig del av dette var at den lignet på wa.works. Grunnen til dette er for å kunne gi WA muligheten til å se produktet i den settingen det er tenkt. Dette har resultert i at testsystemet bruker mange komponenter fra wa.works plattformen.

Sett i ettertid var valget om å bruke WA komponenter både bra og dårlig. Den positive siden er at det var veldig enkelt å gjenskape utseende til wa.works plattformen. Problemet var at komponentene i bruk er veldig komplekse, og det var ikke enkelt å overføre dem til testsystemet. Navigasjonsmenyen vi bruker er tatt direkte fra wa.works, men bare det som er nødvendig for å gjenskape utseende. I ettertid kunne det vært enklere å gjenskape, enn å tilpasse de eksisterende komponentene. Dette skjedde med de fleste av komponentene som ble overført, og har muligens ikke spart noe tid totalt sett.

7 KONKLUSJON OG VIDERE ARBEID

7.1 Konklusjon

Produktet som er laget er en plattform hvor kandidater kan teste ferdigheter sine i CSS. Testen gir et resultat som kan brukes for å vurdere nivået kandidaten har i CSS.

Produktet har noen mangler når det gjelder lagring av data, men er et godt grunnlag for videre utvikling mot en fullstendig testplattform for å teste kandidater.

7.2 Videre arbeid

I evalueringsmøtet ble det diskutert hvordan prosjektet kommer til å bli tatt videre. Prosjektet som ble levert kommer til å bli integrert i den eksisterende rekrutteringsplattformen wa.works, og deretter skal flere programmeringsspråk bli implementert for kandidater å teste seg i.

- Oppretting av CSS tester, testing i CSS og valg av oppgaver er ferdig. Det er lite som må gjøres for å ta i bruk. Muligheter for videre utvikling er:
    - Oppkobling mot database.
    - Potensiell endring av score ved å legge til antall ord/tegn i utregningen.
    - Legge til at kandidat må bekrefte starting av test.
    - Legge til synlig tid under gjennomføring av test.
- Det meste av arbeidet er å legge til nye språk, men samme kode editor kan brukes. Muligheter for videre utvikling er:
    - Legge til input og output tester. Helst på en slik måte at en og samme test kan brukes for å teste flere språk.
    - Testing av tiden koden tar å kjøre for potensiell bruk i score.
    - Muligheter for enkle grafiske løsninger for å gjøre testen mer interessant å gjennomføre. Dette kan være enkle figurer som beveger seg i x og y retning.

REFERANSER

Berg, C, Azets, (2019), Tester i rekruttering: Slik gjør du det riktig, Hentet fra https://www.azets.no/blogg/tester-i-rekruttering/ [Lest 5.Mai.2022]

Berggren, Adresseavisen, (2020), Hva koster en feilansettelse?, Hentet fra, https://kundeservice.adressa.no/2020/12/08/hva-koster-en-feilansettelse/ [Lest 15.Mai.2022]

Experis ManpowerGroup, (2022), Aldri vært vanskeligere å få tak i riktig kompetanse, Hentet fra, https://www.experis.no/nb/aktuelt/articles/2021/12/22/meos-q1-2022 [Lest 6.Mai.2022]

Gjerde, A, (2021), NAVS BEDRIFTSUNDERSØKELSE 2021 [Internett], Tilgjengelig fra: Nav https://arbeidogvelferd.nav.no/journal/2021/2/m-5101/NAVs_bedriftsunders%C3%B8kelse_2021 [Lest 10.Mai.2022]

HRmagasinet, (2018), Svindyrt med feilansettelser, Hentet fra, https://www.hrmagasinet.no/adecco-feilansettelse-ledelse/svindyrt-med-feilansettelser/508507#:~:text=Det%20er%20vanskelig%20%C3%A5%20ansl%C3%A5,ligge%20mellom%20NOK%20700.00%20%E2%80%93%20900.000 [Lest 15.Mai.2022]

Mapbox, (u.å.), pixelmatch, Hentet fra https://github.com/mapbox/pixelmatc [Lest 3.Mars.2022]

Microsoft, (u.å.), monaco-edito, Hentet fra https://github.com/Microsoft/monaco-editor [Lest 2.Januar.2022]

Mozilla, (u.å.), JavaScript, Hentet fra https://developer.mozilla.org/en-US/docs/Web/javascript [Lest 19.Mai.2022

Mæland, J, UIA, (2020), Testverktøy i rekruttering, Hentet fra https://www.uia.no/student/uia-karriere/jobbsoek/testverktoey-i-rekruttering [Lest 5.Mai.2022]

Open Source Initiative, (u.å.), ISC License (ISC), Hentet fra https://opensource.org/licenses/ISC [Lest 23.Mars.2022]

Open Source Initiative, (u.å.), The MIT License, Hetet fra https://opensource.org/licenses/MIT [Lest 21.Mars.2022]

React, (u.å.), React, Hentet fra https://reactjs.org/[Lest 19.Mai.2022]

React-redux.js, (u.å.), React Redux, Hentet fra https://react-redux.js.org [Lest 19.Mai.2022]

Redux, (u.å.), Redux Hentet fra https://redux.js.org/ [Lest 19.Mai.2022]

Scrum.org, (u.å.), What is Scrum? Hentet fra https://www.scrum.org/resources/what-is-scrum [Lest 25.April.2022]

Talentech, (u.å.), Ferdighetstest ved rekruttering, Hentet fra, https://talentech.com/no/kunnskapsbank/rekrutteringshuben/ferdighetstest-ved-rekruttering/ [Lest 5.Mai.2022]

Typescriptlang, (u.å.), What is TypeScript?, Hentet fra https://www.typescriptlang.org

Von Hertzen, N, (u.å.), html2canvas, Hentet fra https://github.com/niklasvh/html2canvas [Lest 23.April.2022]

W3schools, (u.å.), HTML <iframe> Tag, Hentet fra https://www.w3schools.com/tags/tag_iframe.asp [Lest 19.Mai.2022]

VEDLEGG

Oppgavebeskrivelse

Testplattform (Wide Assessment AS)

Wide Assessment (WA) er et enkelt, rimelig og effektivt rekrutteringsverktøy for bedrifter i IT-bransjen. WA gjør noe ingen andre verktøy gjør i dag; det rangerer kandidater. Det vil si at uansett hvor stor søknadsbunke man står ovenfor, så vet man øyeblikkelig hvem man bør ta en samtale med. Dette er mulig fordi vi har laget en ny skill-basert CV som viser hva kandidaten kan utover stillingstitler og utdanningsgrader.

I tillegg har vi valgt å åpne databasene slik at arbeidsgiver slipper å begrense seg til hva man klarer å få inn av søkere selv. I WA kan man rangere alle kandidater i hele databasen etter sitt individuelle behov for f eks. en react-utvikler med C#-kompetanse. I denne felles rangeringen er andres kandidater i utgangspunktet anonyme, men bedriftene kan oppfordre gode matcher til å søke på sine stillinger. Store deler av plattformen er utviklet av studenter fra HVL gjennom praksis, bachelor oppgaver eller ansettelser etter fullført studier.

Plattformen lever på www.wa.works.

Oppgave

Bakgrunn for prosjektet

WA brukes i dag av flere bedrifter til å komme i kontakt med kandidater. Vi ønsker å gjøre prosessen fra å gå fra kandidat til ansatt så enkel som mulig. Vi ønsker å la kandidatene gjennomføre kodeoppgaver på plattformen vår for flere grunner:

- Få oversikt over kunnskapsnivået sitt og se hvordan man ligger ann i forhold til andre kandidater
- Felles tester som kan brukes hos flere bedrifter i en intervju prosess.
- Kunne gjennomføre kodekonkurranser

Beskrivelse av teknologier / metoder og annet som er tenkt brukt i prosjektet

Det programmeres i dag i React, TypeScript og C#(.Net Core), og vi har også noen node og gatsby tilleggstjenester. Vi bruker Pivotal Tracker for Kanban-board og kjører Scrum som metode.

Beskrivelse av arbeidsoppgaver for studentene i prosjektet

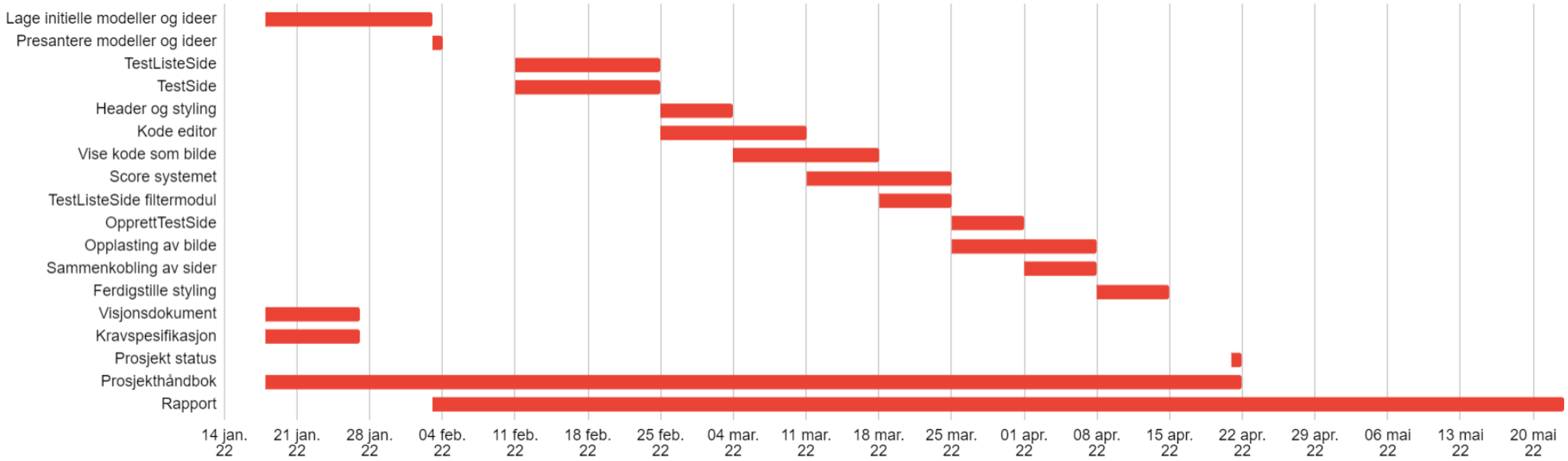
Arbeidsoppgavene i dette prosjektet er å utvikle en testplattform. Gruppen står fritt til å velge om testene skal være i JavaScript eller CSS.

- Brukeren skal kunne velge en test, ut av en liste med tester.
- Når en test starter, skal brukeren få opp et tekstinput felt hvor man kan skrive kode.
- Koden brukeren skriver skal kompileres og testes, her får man poeng basert på antall linjer, likhet, kjøretid osv
- Etter gjennomført test, skal resultatet lagres i en database.

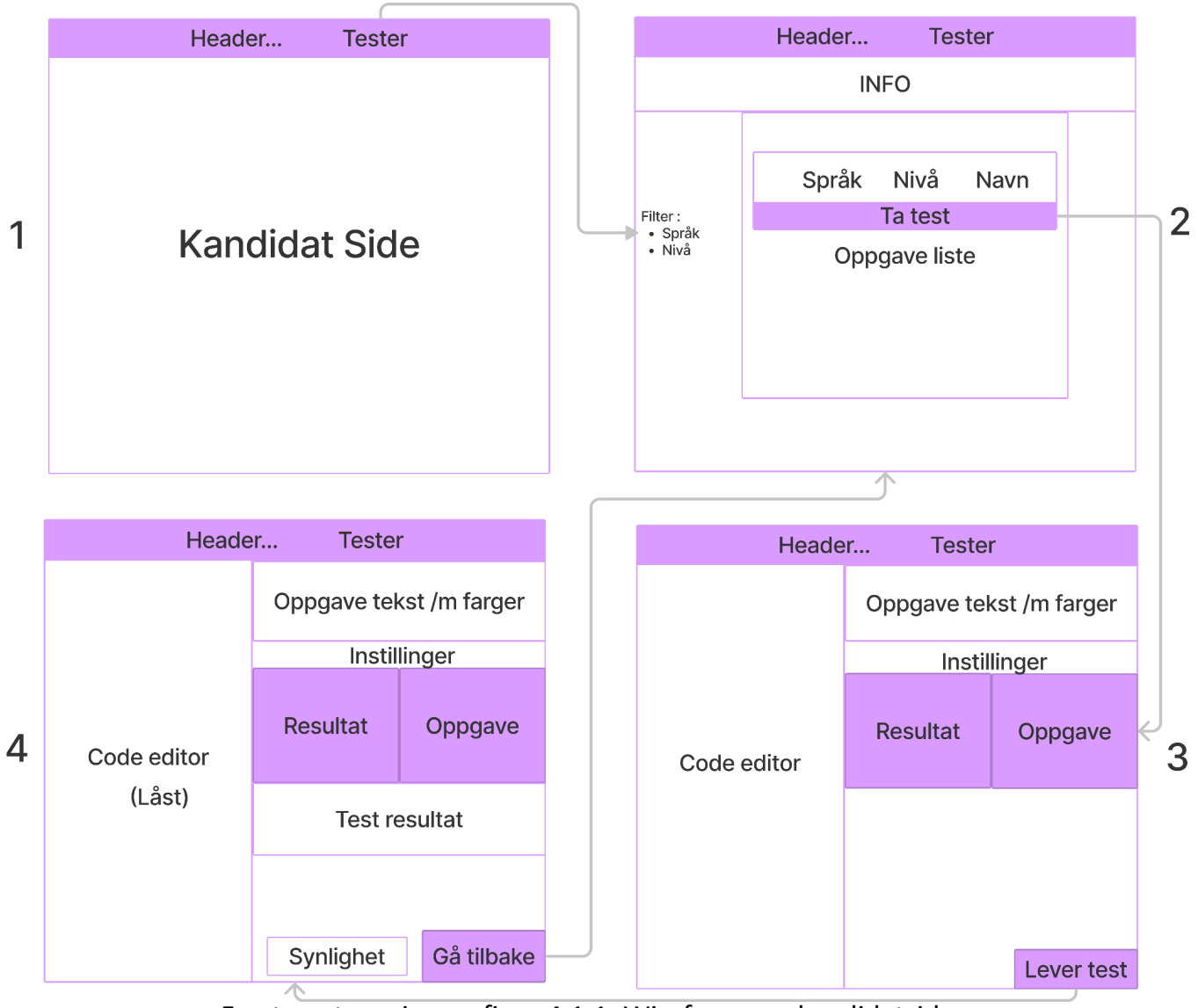
Hvorfor studentene bør velge akkurat dette prosjektet.

Man bør velge dette prosjektet på grunn av erfaringen vår med studentprosjekter. Vi har gode rutiner for å spekke oppgavene og kjøre slike prosesser i tett samarbeid med studentene og ser verdien av å invitere dem til å oppgradere våre originale planer. Dessuten er det en unik mulighet til å bli komfortabel med teknologi som er ettertraktet på markedet og få førstehåndskunnskap om hva som skal til for å sikre seg drømmejobben.

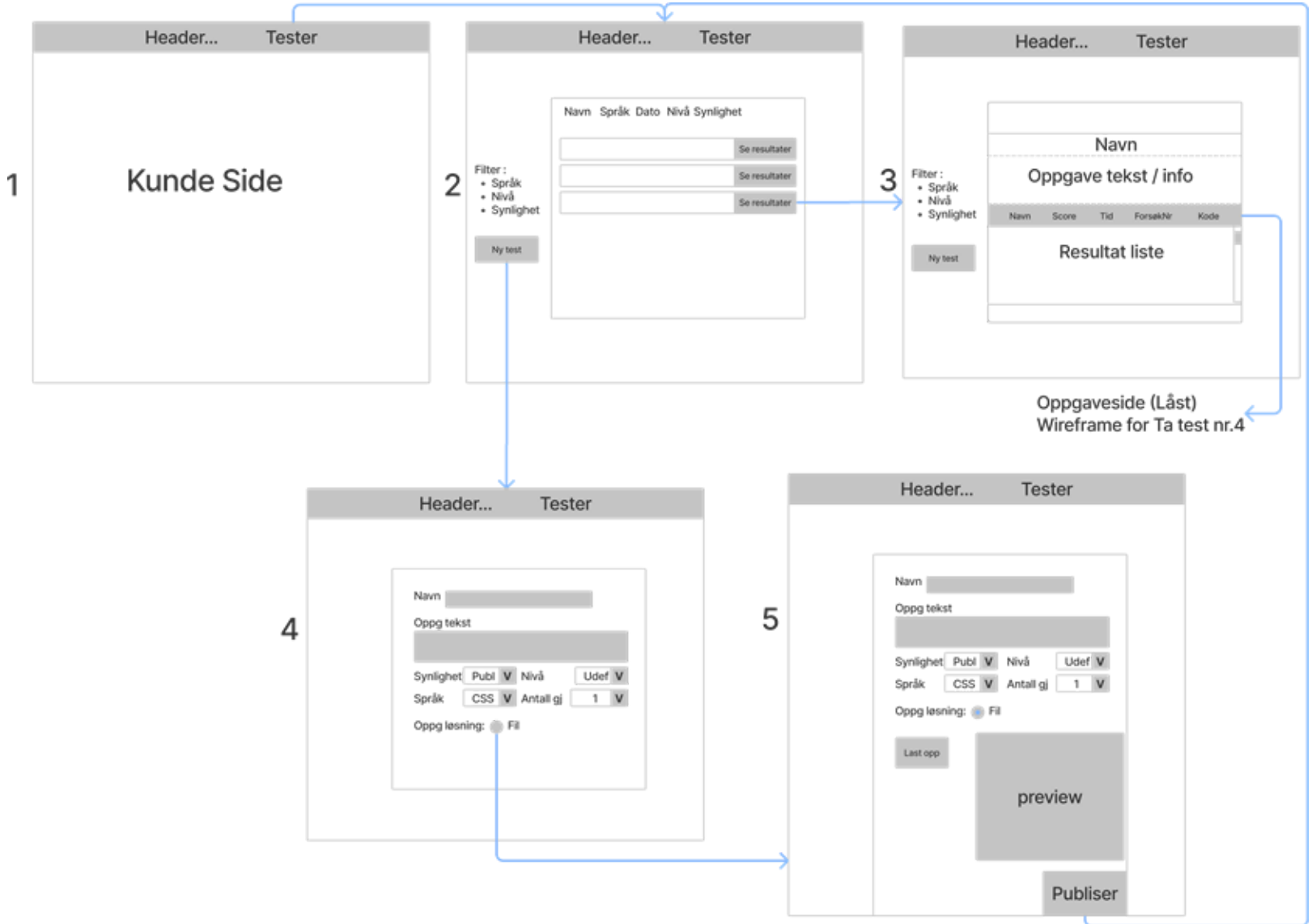
Bilder og figurer



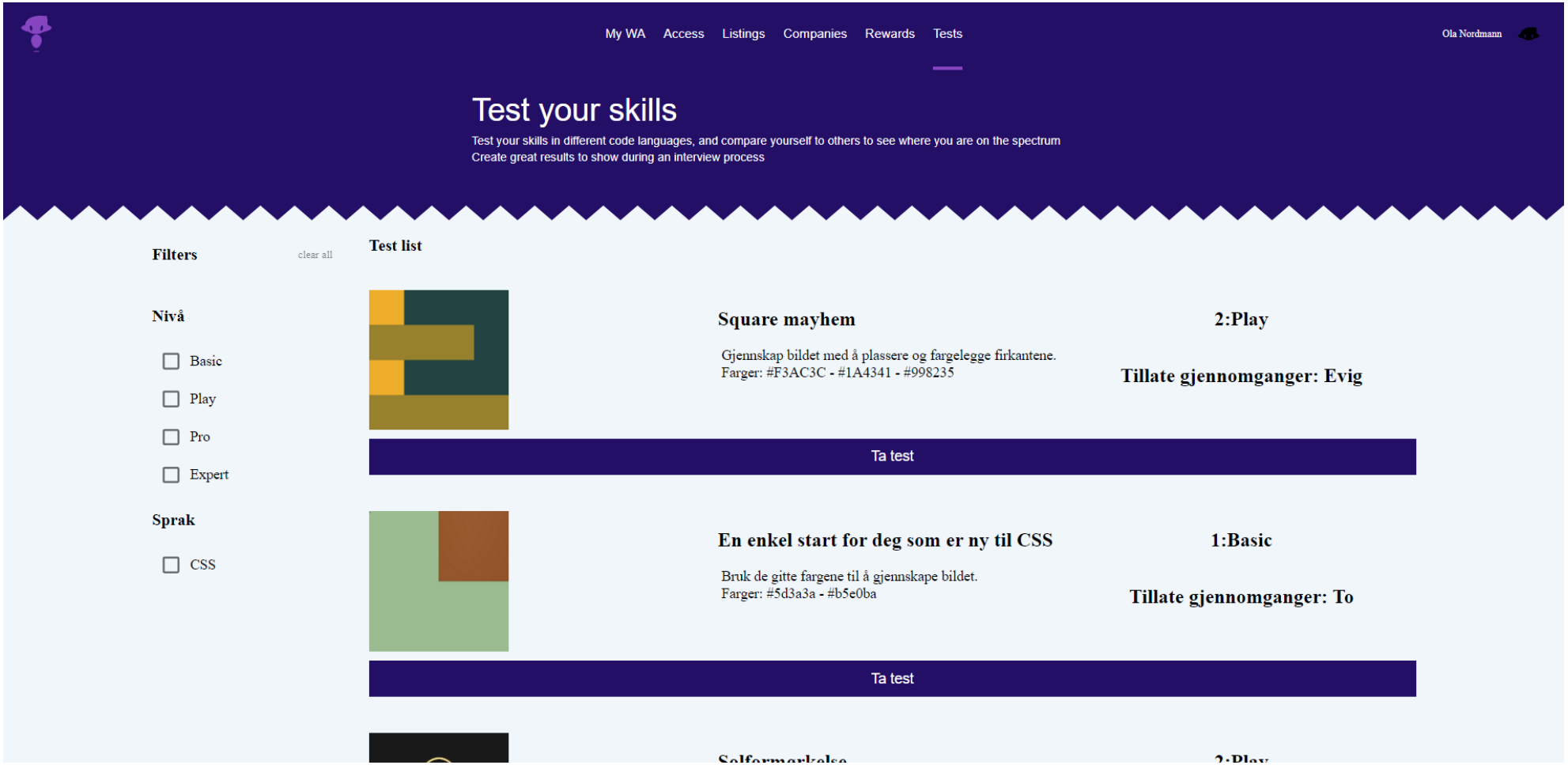
Figur 3.4 Gannt diagram



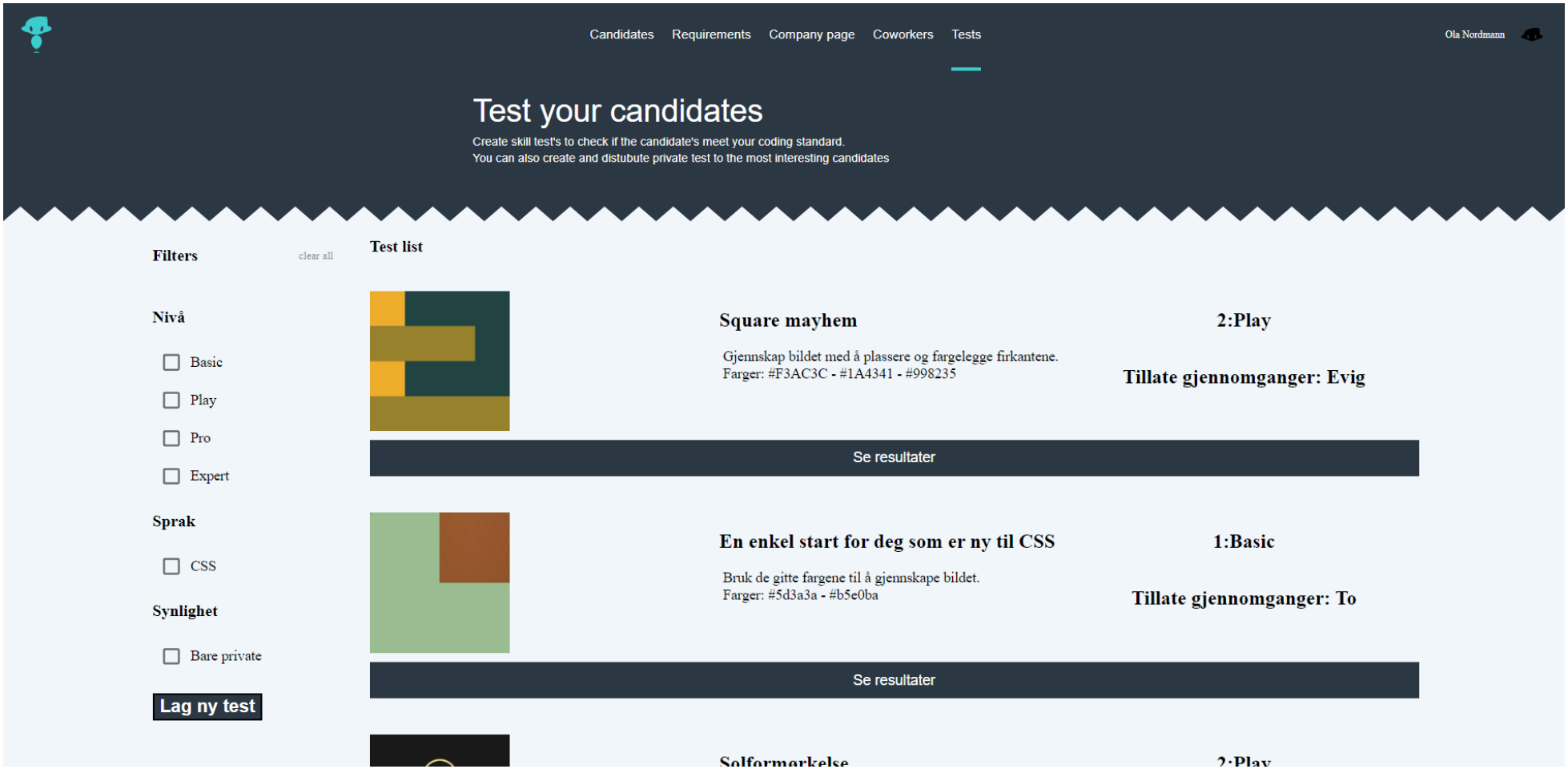
Forstørret versjon av figur 4.1.1: Wireframe av kandidatsidene.



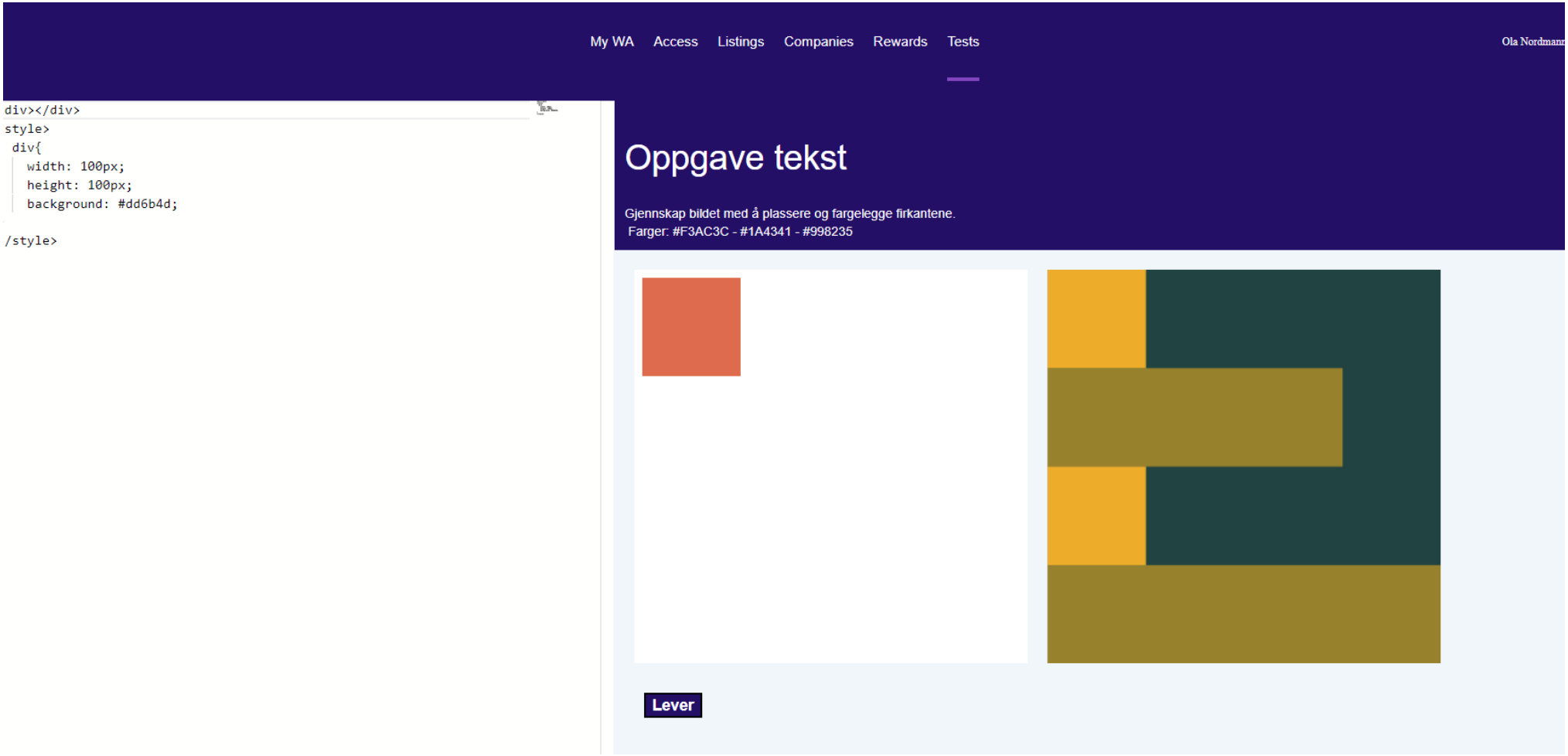
Forstørret versjon av figur 4.1.2: Wireframe av kundesidene.



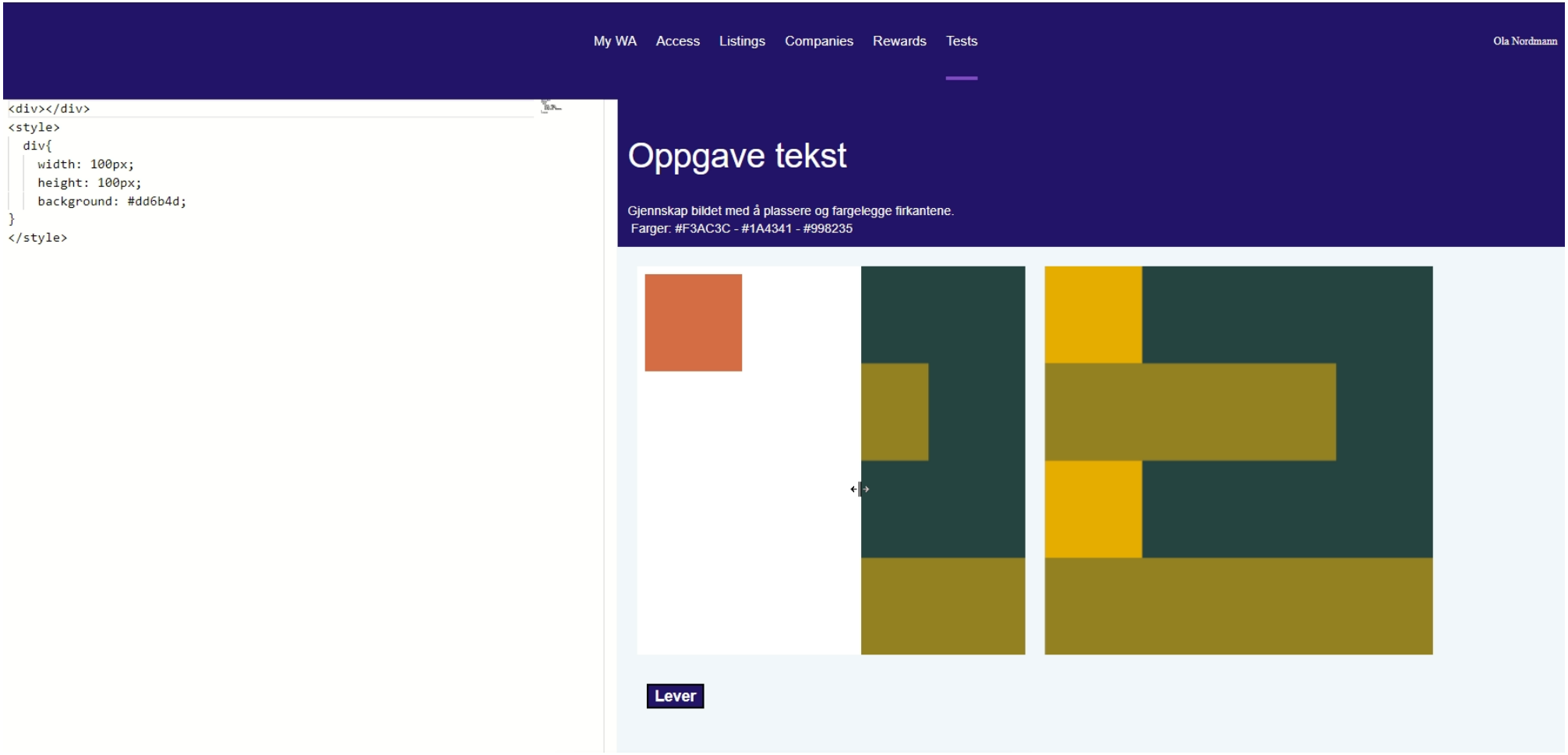
Forstørret versjon av bilde 4.2.1: Skjermbilde av siden som viser alle testene.



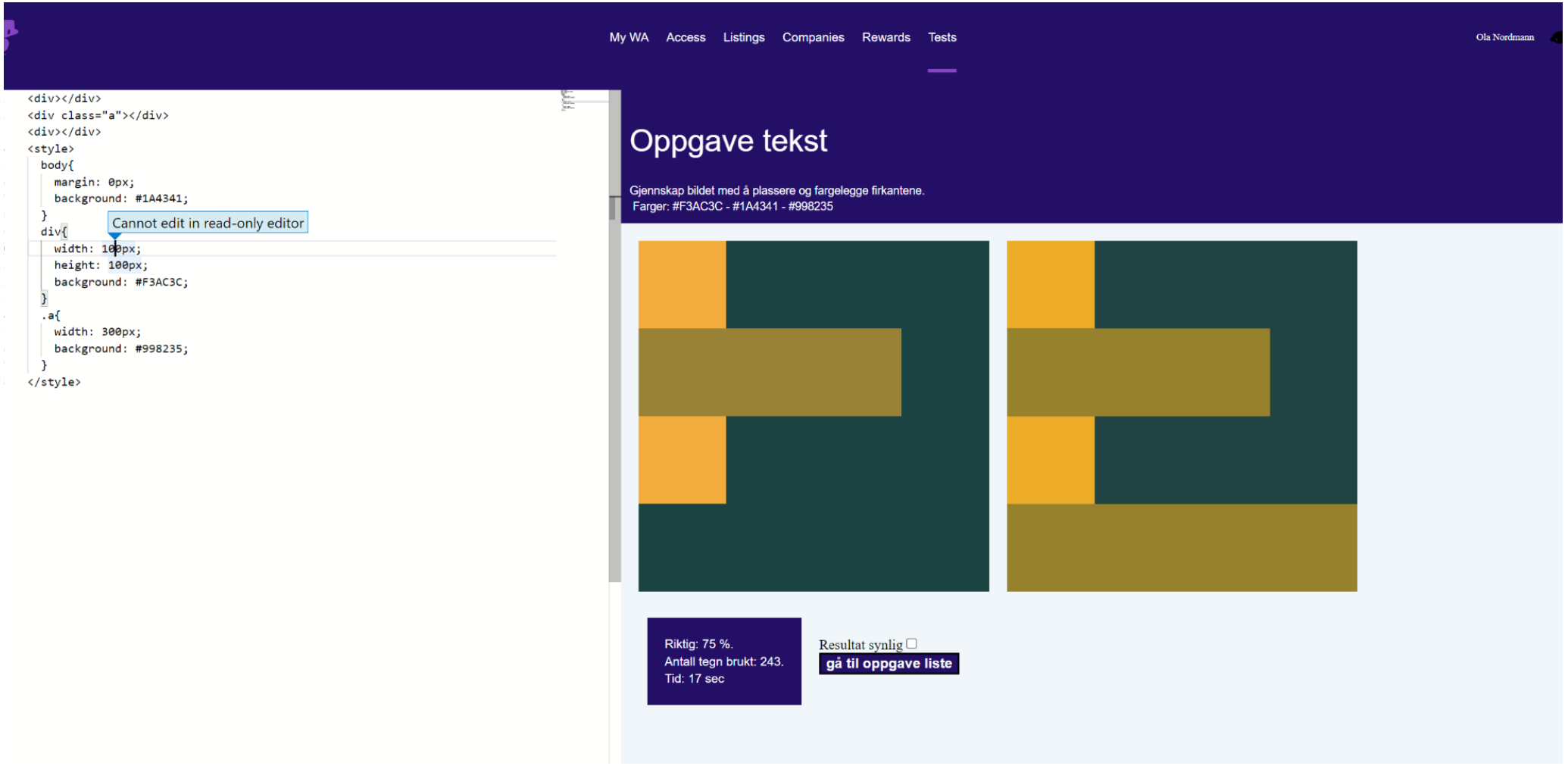
Forstørret versjon av bilde 4.2.2: Skjermbilde av tester som en kunde har publisert.



Forstørret versjon av bilde 4.2.3: Skjermbilde av en nylig startet test.

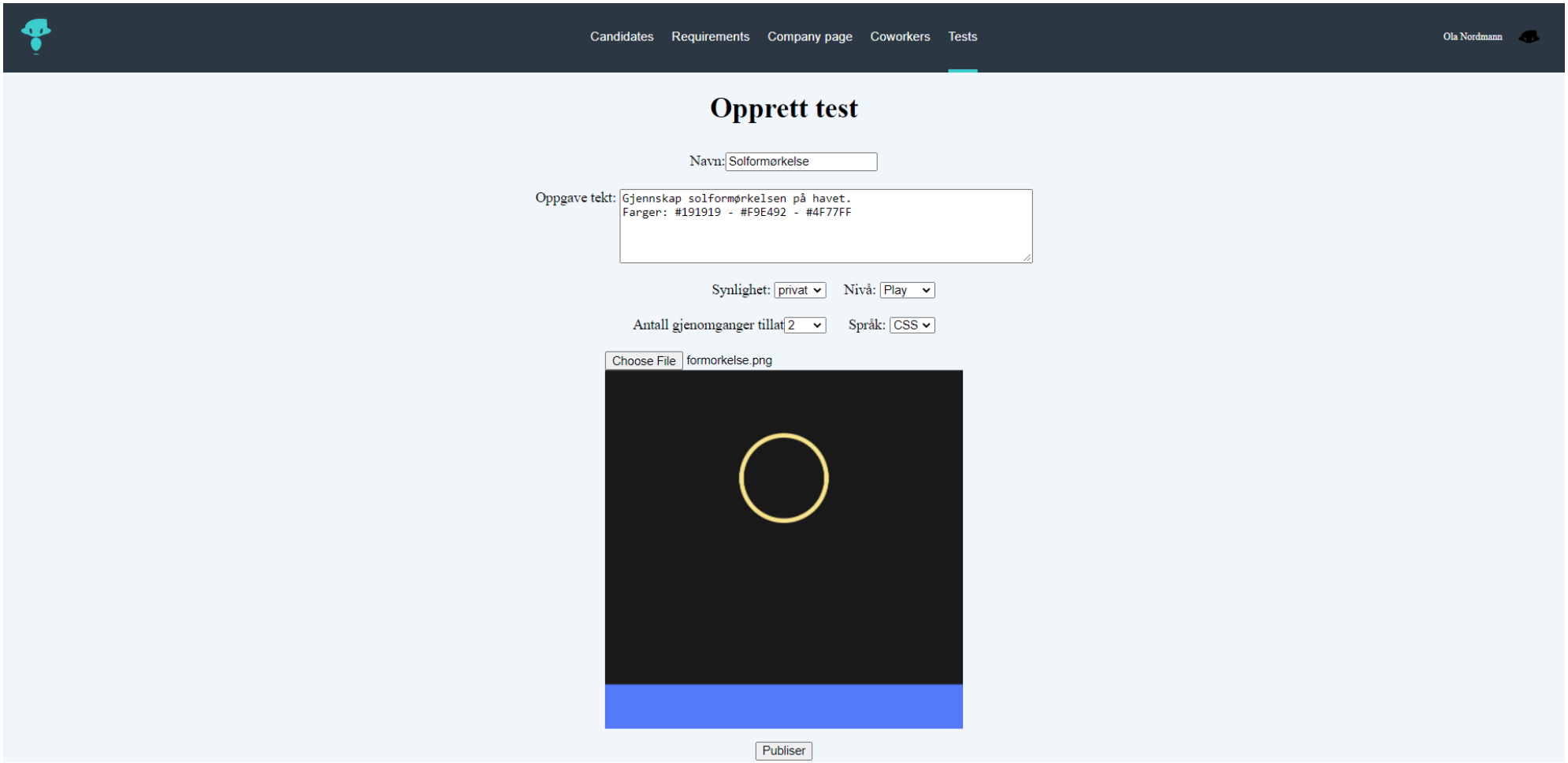


Forstørret versjon av bilde 4.2.4: Skjermbilde av testsiden mens slideren blir brukt.

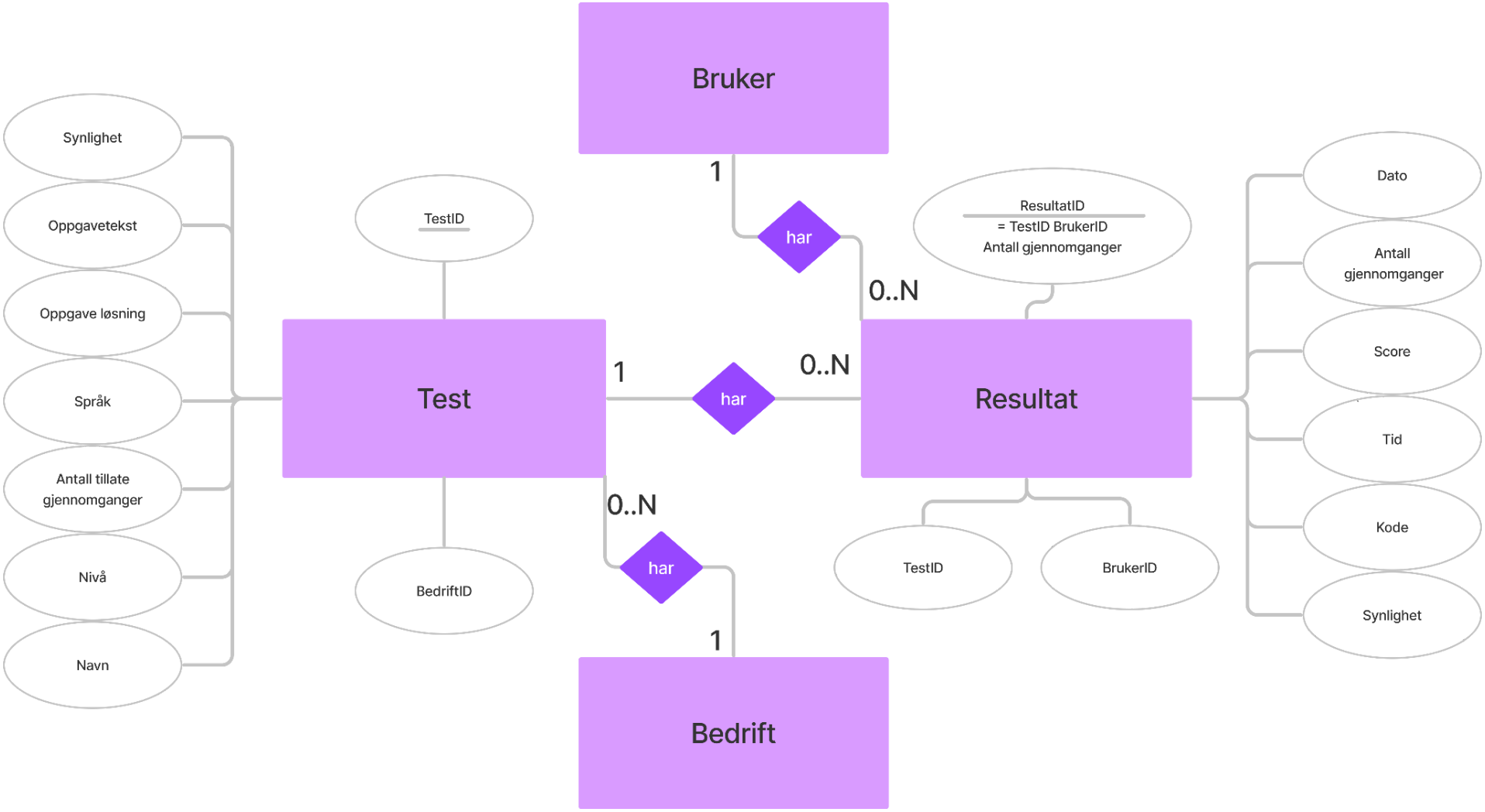


Forstørret versjon av bilde 4.2.5: Skjermbilde av testsiden, etter at testen er levert.

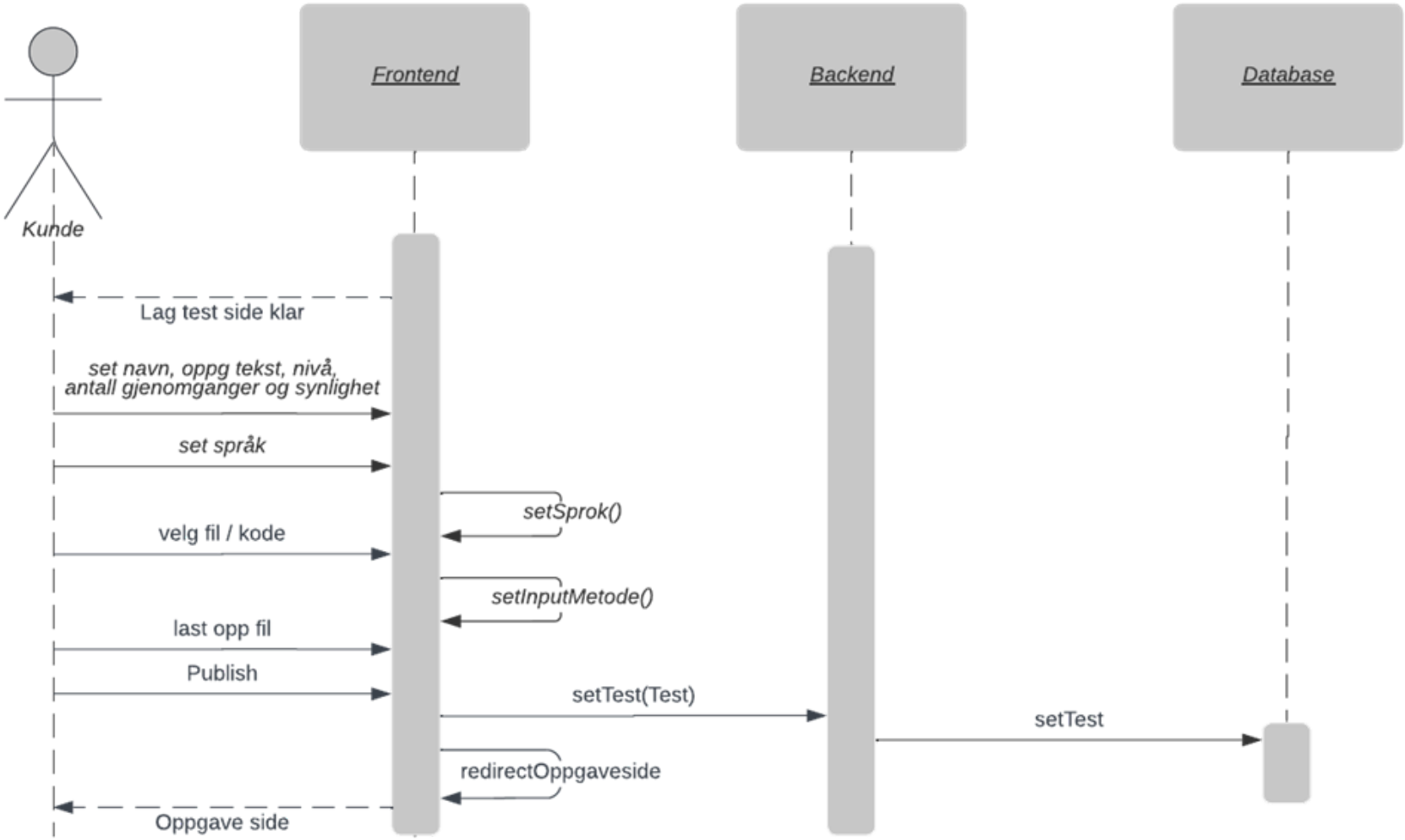
Forstørret versjon av bilde 4.2.6: Skjermbilde av ny test siden.



Forstørret versjon av bilde 4.2.7: Skjermbilde av en utfylt ny test side.



Forstørret versjon av figur 4.6: ER-diagram for databasen



Figur 5.2: Sekvensdiagram for å lage test

```
const fs = require('fs');
const PNG = require('pngjs').PNG;
const pixelmatch = require('pixelmatch');

const img1 = PNG.sync.read(fs.readFileSync('img1.png'));
const img2 = PNG.sync.read(fs.readFileSync('img2.png'));
const {width, height} = img1;
const diff = new PNG({width, height});

pixelmatch(img1.data, img2.data, diff.data, width, height, {threshold: 0.1});
```

Bilde 6.1 kode for konvertering av bilde tatt fra https://github.com/mapbox/pixelmatch [12.Mai.2022].