Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

Nettbasert testing av kodeferdigheter for rekrutterings formål

Systemdokumentasjon

Versjon <2.0>

Dokumentet er basert på Systemdokumentasjon utarbeidet ved NTNU. Revisjon og tilpasninger til bruk ved IDER, DATA-INF utført av Carsten Gunnar Helgesen, Svein-Ivar Lillehaug og Per Christian Engdal. Dokumentet finnes også i engelsk utgave.

REVISJONSHISTORIE

Dato	Versjon	Beskrivelse	Forfatter
24.02.2022	1.0	Første iterasjon	Ruben, Svein Ove
17.5.2022	2.0	Ferdigstille systemdokumentasjon	Ruben, Svein Ove

Innholdsfortegnelse

1 INNLEDNING

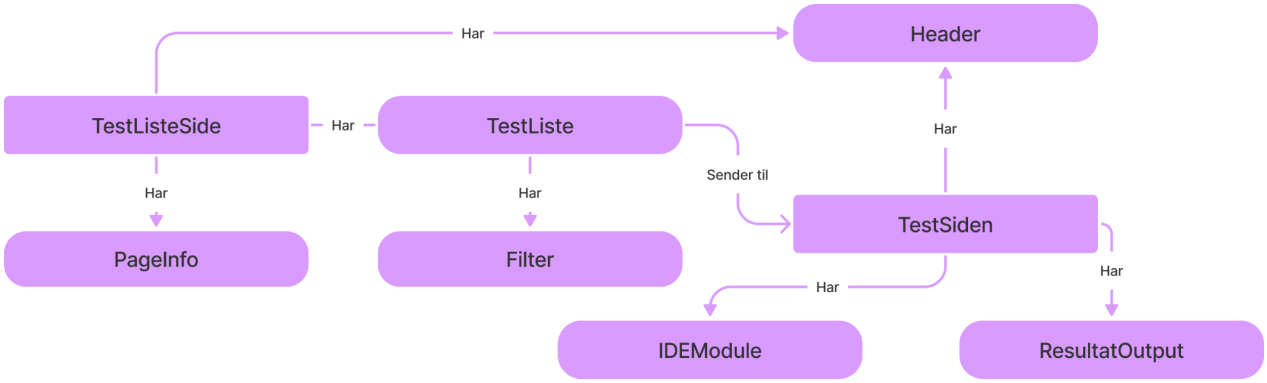
Systemdokumentasjonen er laget for å gi en dypere innsikt i designet og arkitekturen til web-applikasjonen som er laget i dette bachelorprosjektet. Hovedpunktene som blir beskrevet er arkitektur, struktur, sikkerhet og oppsett og kjøring

Noen av kapitlene er ikke relevant for det leverte test systemet, men heller når det er videreutviklet og ferdig. Gjennom samtaler med utviklere hos WA er en mulig løsning funnet på noen av disse områdene, og er da presentert i kapitlene det gjelder.

Noen av modellen som er presentert her er ganske store, og det er vanskeligheter å lese all teksten på noen av disse. For dem det gjelder er det en forstørret versjon i vedlegg der teksten er klarere.

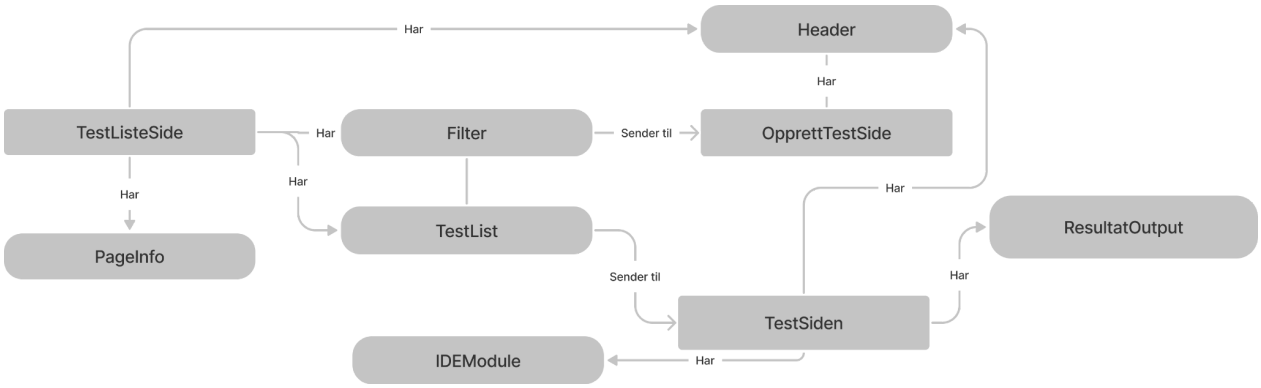
2 ARKITEKTUR

Figur 2.1 viser oppsettet og de viktigste komponentene til kandidatsidene. Domenemodellen ble laget i oppstartsfasen av prosjektet for å enklere se sammenhengen mellom sidene.



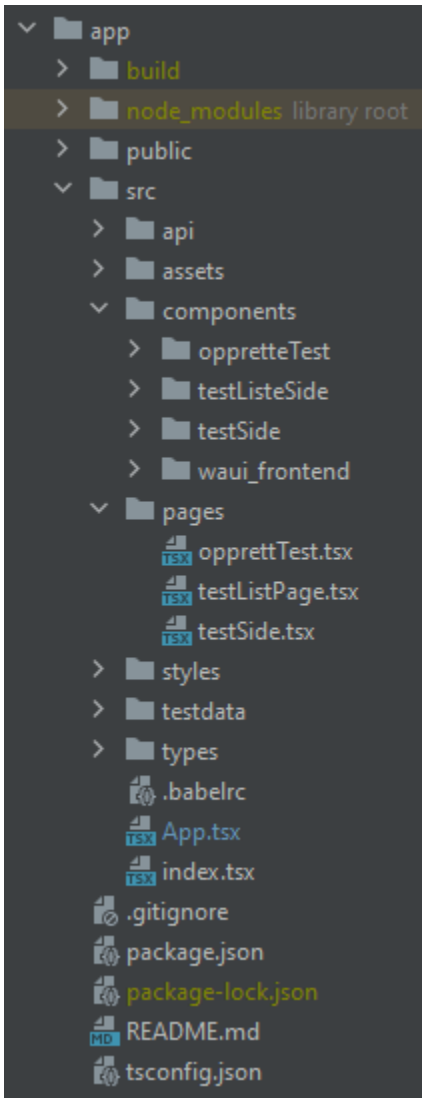
Figur 2.1: Domenemodell for kandidatsidene

Figur 2.2 viser oppsettet for kundesidene. Det er mange felleskomponenter mellom kundesidene og kandidatsidene, og hovedforskjellen er at kunder kan lage tester og ikke ta tester.



Figur 2.2.: Domenemodell for kundesider

3 PROSJEKTSTRUKTUR



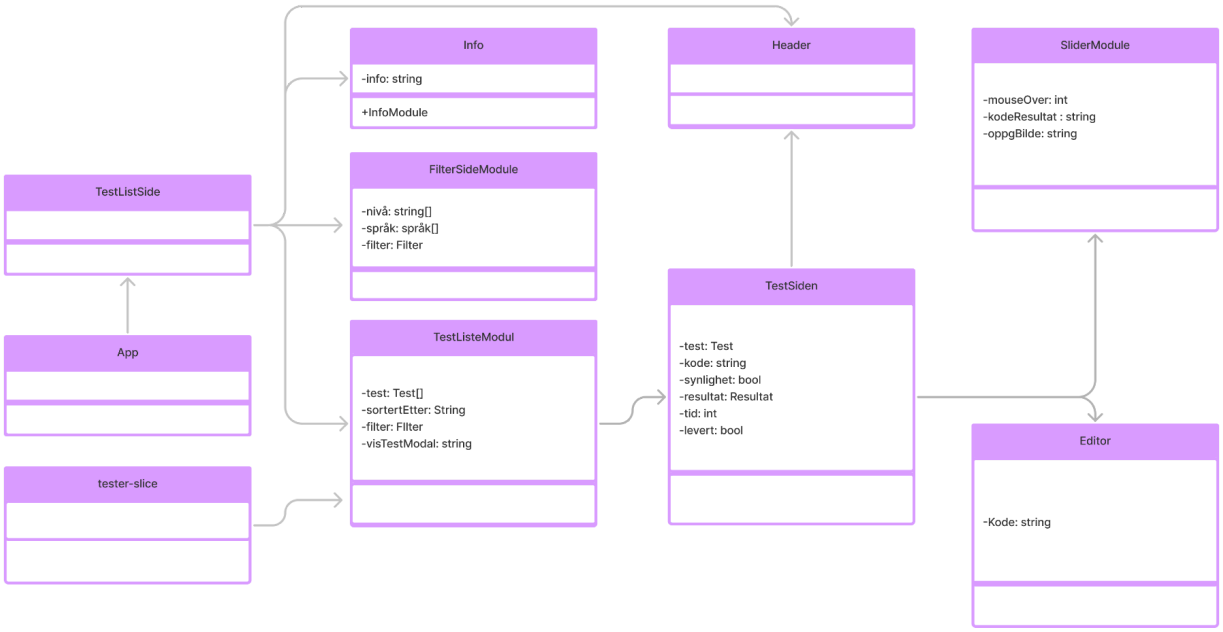
Bilde 3.1: Skjermbilde av katalogstrukturen i prosjektet

Bilde 3.1 viser mappestrukturen til web applikasjonen, og viser organiseringen for disse. F.eks har hver side en egen mappe for komponentene den bruker.

Før starten av utviklingen ble det bestemt av å bruke norske navn gjennom utviklingen, på grunn av rapporten skulle skrives på norsk. Dette skapte derimot problemer etter hvert som medlem kodet da vane førte til at en del av navnene ble skrevet på engelsk. Det skapte et system der det er noe norske og noe engelske navn på variabler, metoder og klasser. Dette ble oppdaget så sent at selv om noen deler er rettet på, så er det fortsatt blanding av norsk og engelsk i koden.

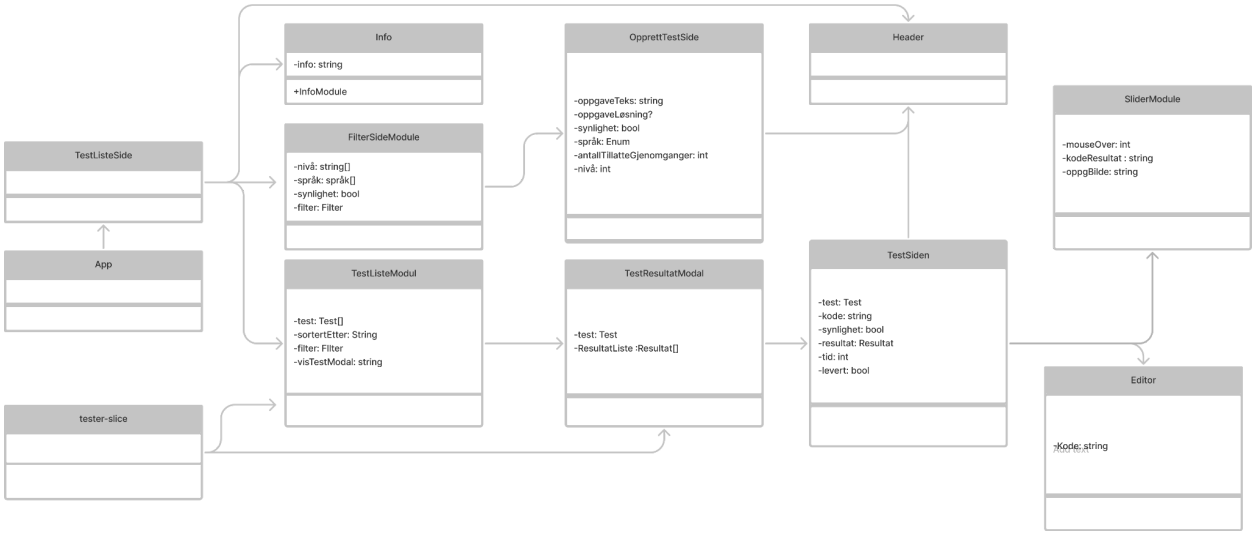
4 KLASSEDIAGRAM

Figur 4.1 viser kandidatsidene med et klassediagram, og et forslag til hvilke verdier som trengs i hver komponent.



Figur 4.1: Klassediagram til kandidatsidene

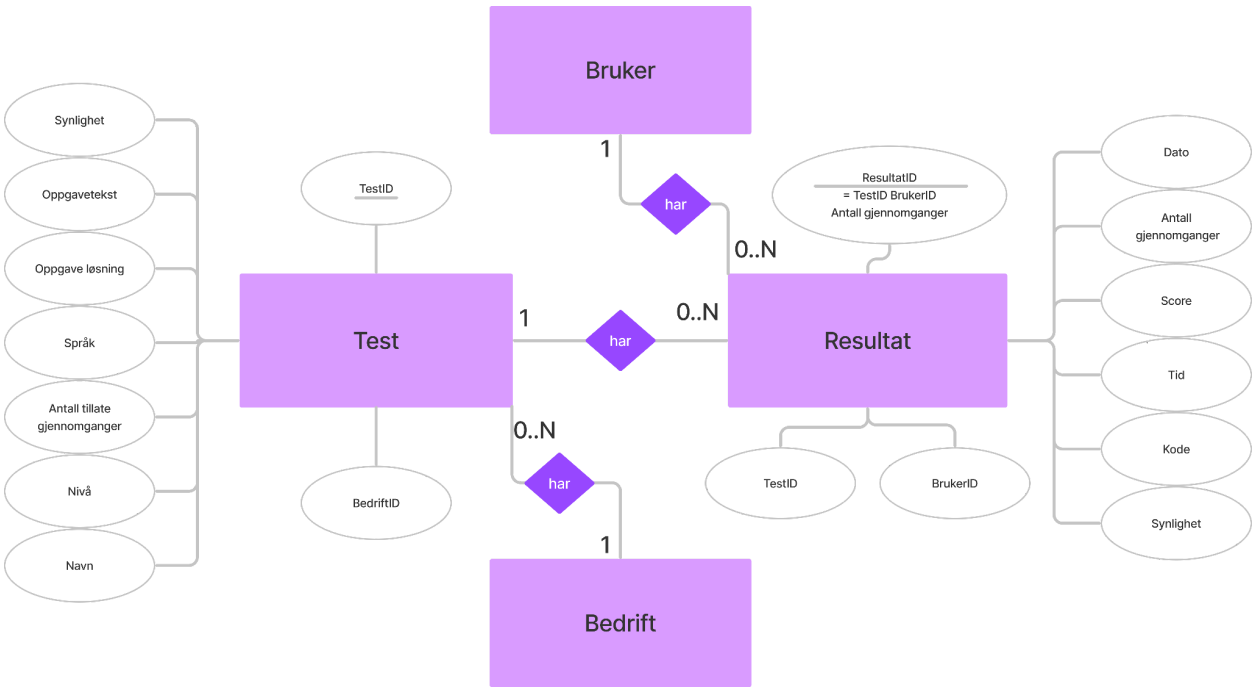
Figur 4.2 viser klasseridagrammet for kundesidene. Disse skiller seg fra kandidatsidene med at en kan lage nye tester, flere filter muligheter og at en kan se resultater istedenfor å ta en test.



Figur 4.2: Klassediagram til kundesidene

5 DATABASEMODELL

I starten av prosjektet ble det designet en database for å kunne lagre tester og resultater. Den leverte løsningen har ikke implementert noe database, men figur 5.1 er brukt for å lage objekter som er brukt i web-applikasjonen.



Figur 5.1: ER-diagram for databasen

Figur 5.1 har to hoved tabeller i seg, en for tester og en for resultater. Hver test har en kobling til bedriften som laget den, og resultater til denne testen.

Resultat tabellen har en kobling til testen den er resultat for, og brukeren som har tatt testen. IDen til resultatet er en kombinasjon av testID, brukerID og hvilke gjennomgang resultatet er fra. Gjennomganger er med for å gjøre det mulig å lagre alle resultatene på tester som kan gjennomføres flere ganger.

6 SERVER-TJENESTER

Det er to servertjenester i det ferdige produktet. Den ene vil fungere som en vanlig API, og vil bli bygget på den eksisterende APIen som er på wa.works plattformen. Denne vil håndtere all datahenting og lagring som kommer fra testsystemet.

Den andre servertjenesten er for å validere koden som blir sendt inn fra kandidatene. Når en kandidat leverer oppgaven sinn blir den kjørt og vurdert i nettleseren. Problemet er at kandidaten kan endre på koden i ettertid for å få et bedre resultat, og derfor må resultatet valideres i back-end også. Grunnen til at dette blir gjort i en egen maskin er for å sikre mot dataangrep med å kjøre ukjent kode fra kandidatene.

7 SIKKERHET

Når tekniske tester blir utført, må koden kjøres for å validere om den fungerer eller ikke. Dette kan skape situasjoner hvor kandidatene som skrive inn kode kan prøve å få tilgang til data fra wa.works plattformen. Dette er en grunn til å kjøre koden i nettleseren, men som sagt i kapittel 6 må resultatene valideres.

Dette er utenfor testsystemte som blir levert, men gjennom samtaler med oppdragsgiver er det funnet noen mulige løsniger:

- Ha en egen masking/API som håndterer all resultat validering. Denne vil da ikke ha tilgang til noe data, og kan kun legge til resultater.
- Begrense hvilke biblioteker som er tilgjengelig for koden som blir kjørt. Kan da sperre biblioteker som trengs for å hente data og slikt.

Når test systemet blir implementert vil den også kjøres innenfor wa.works plattformen, og vil da arve alle sikkerhetstiltak som allerede er implementert der.

8 INSTALLASJON OG KJØRING

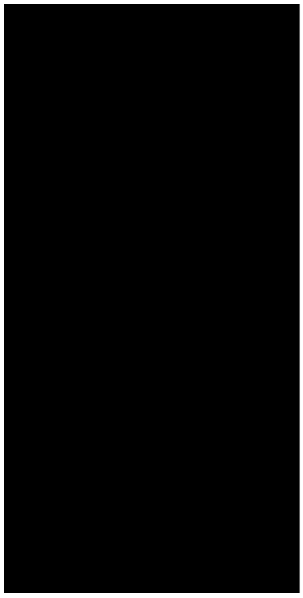
For å kjøre applikasjonen må Node være installert på maskinen.

Første gang React applikasjonen starter må Node biblioteker bli installert. Åpne et command vindu i prosjektmappen, og kjør kommandoen npm install

Når den er ferdig kan du videre kjøre kommandoen npm start. Web serveren vil da starte lokalt på maskinen, og den er tilgjengelig på http://localhost:3000. For å bytte mellom kundesider og kandidatsider trykker en på Ninyo oppe til venstre på siden.

9 KONTINUERLIG INTEGRASJON OG TESTING

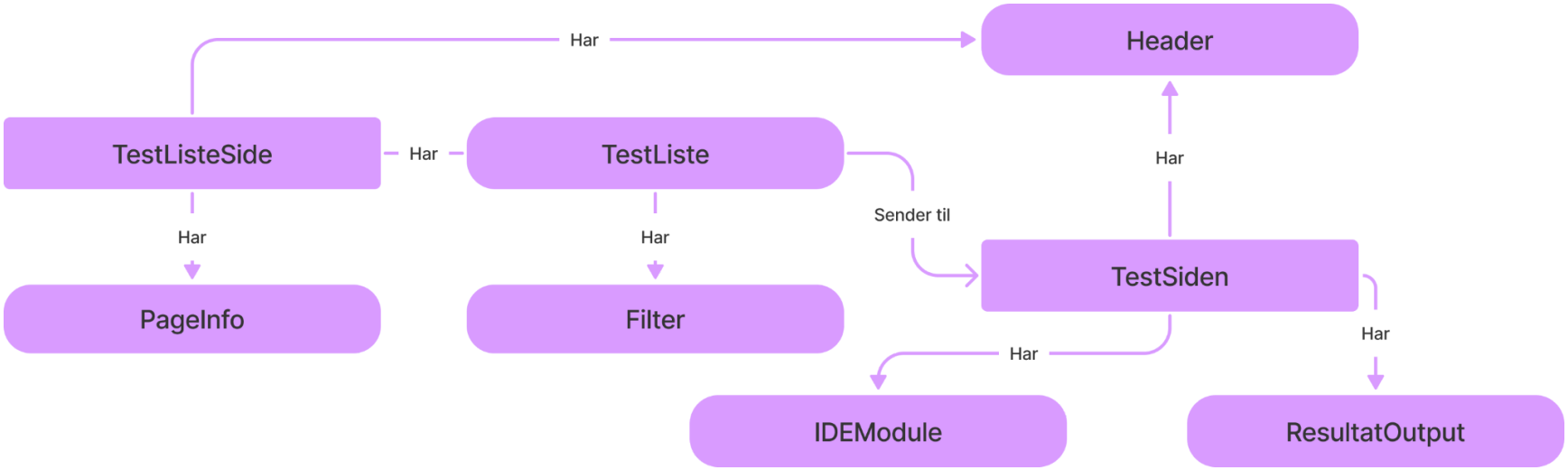
Testsystemet har ikke mange store funksjonaliteter, og det er derfor ikke laget noen automatiske tester. Hovedsakelig er det bare bildesammenligningen som manuelt har blitt testet. For å teste denne ble det laget en egen test som er enkel å teste om riktig % blir returnert.



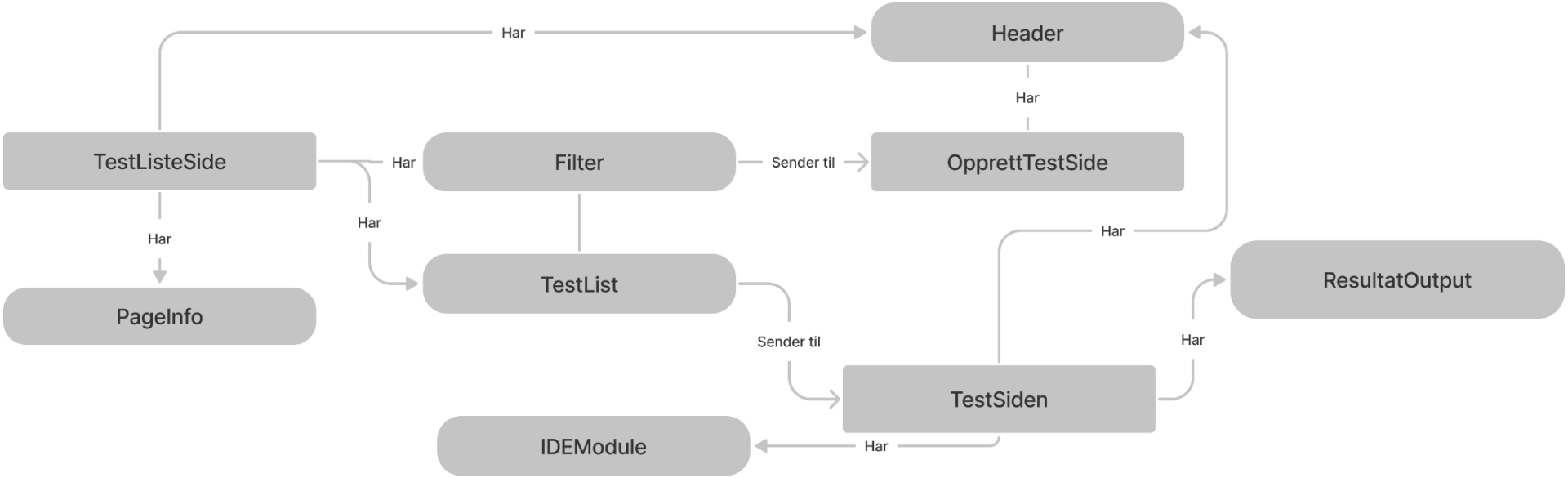
Bilde 9.1: Bildet som ble brukt til å teste bildesammenligning. 50% svart og 50% hvit.

Bilde 9.1 er bildet som ble brukt for å teste resultatene som ble gitt etter bildesammenligningen. En test som ble gjort var å levere et bilde som var svart i øverste halvdel, og hvit i nederste halvdel. Hvis bildesammenlignigen er riktig skal den da returnere at 50% er riktig.

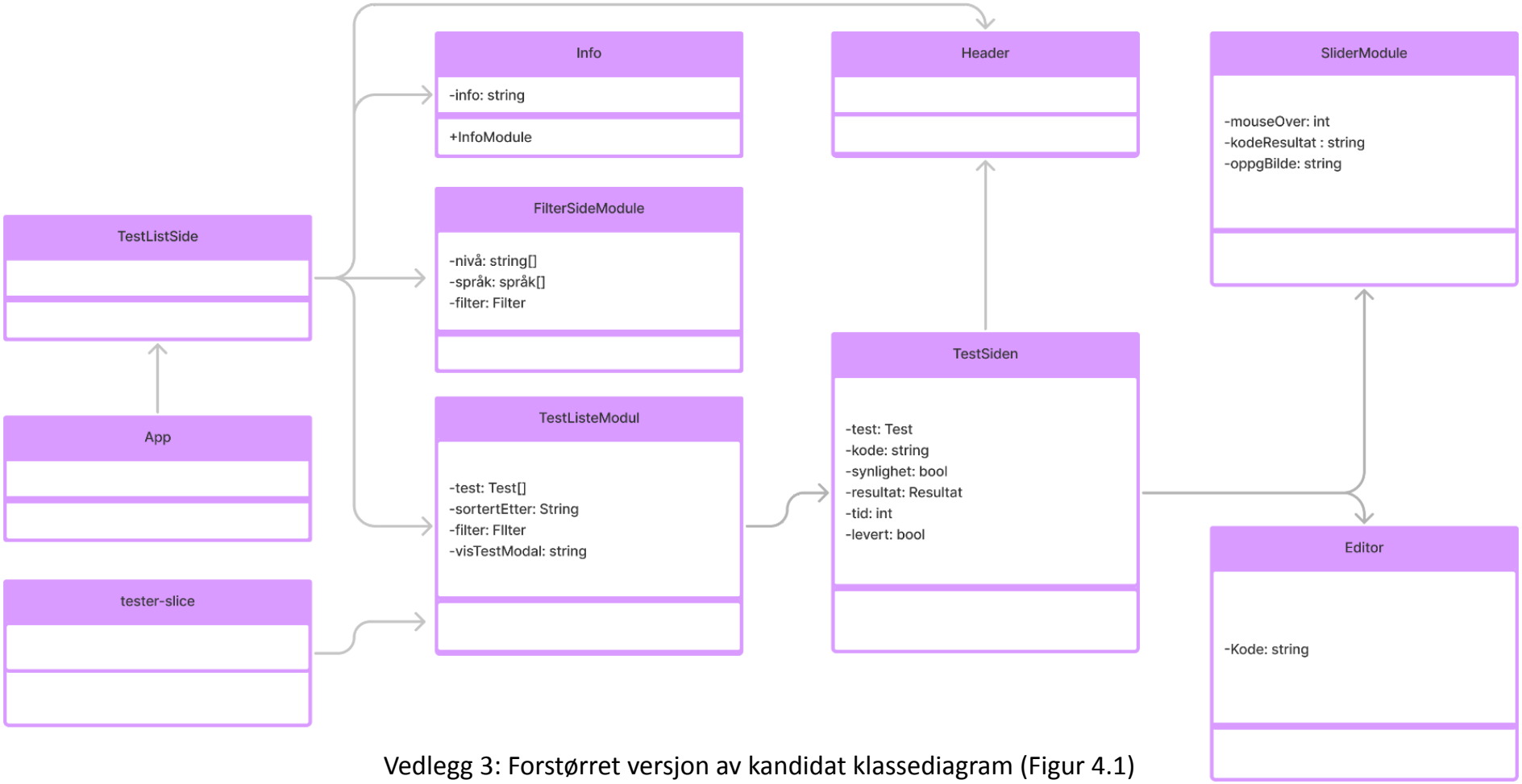
VEDLEGG:



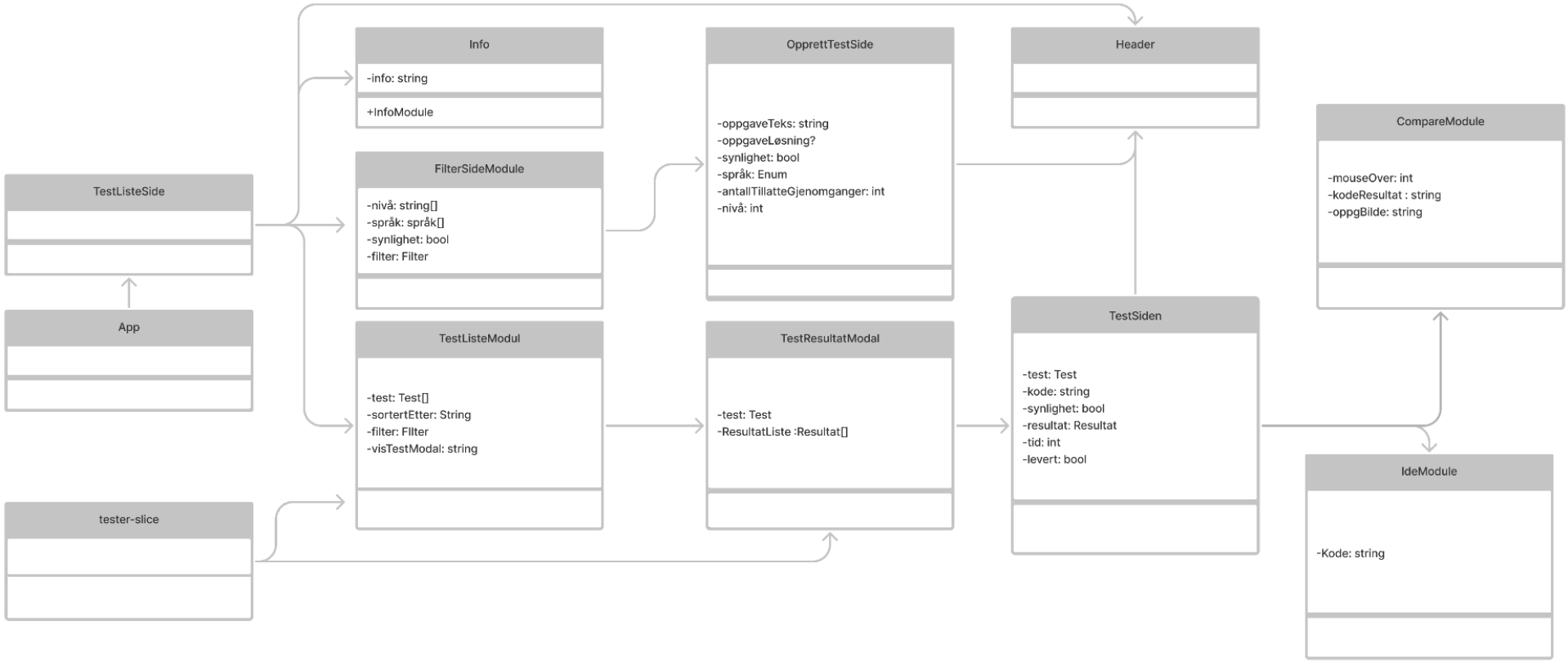
Vedlegg 1: Forstørret versjon av domenemodell for kandidatsidene (Figur 2.1)



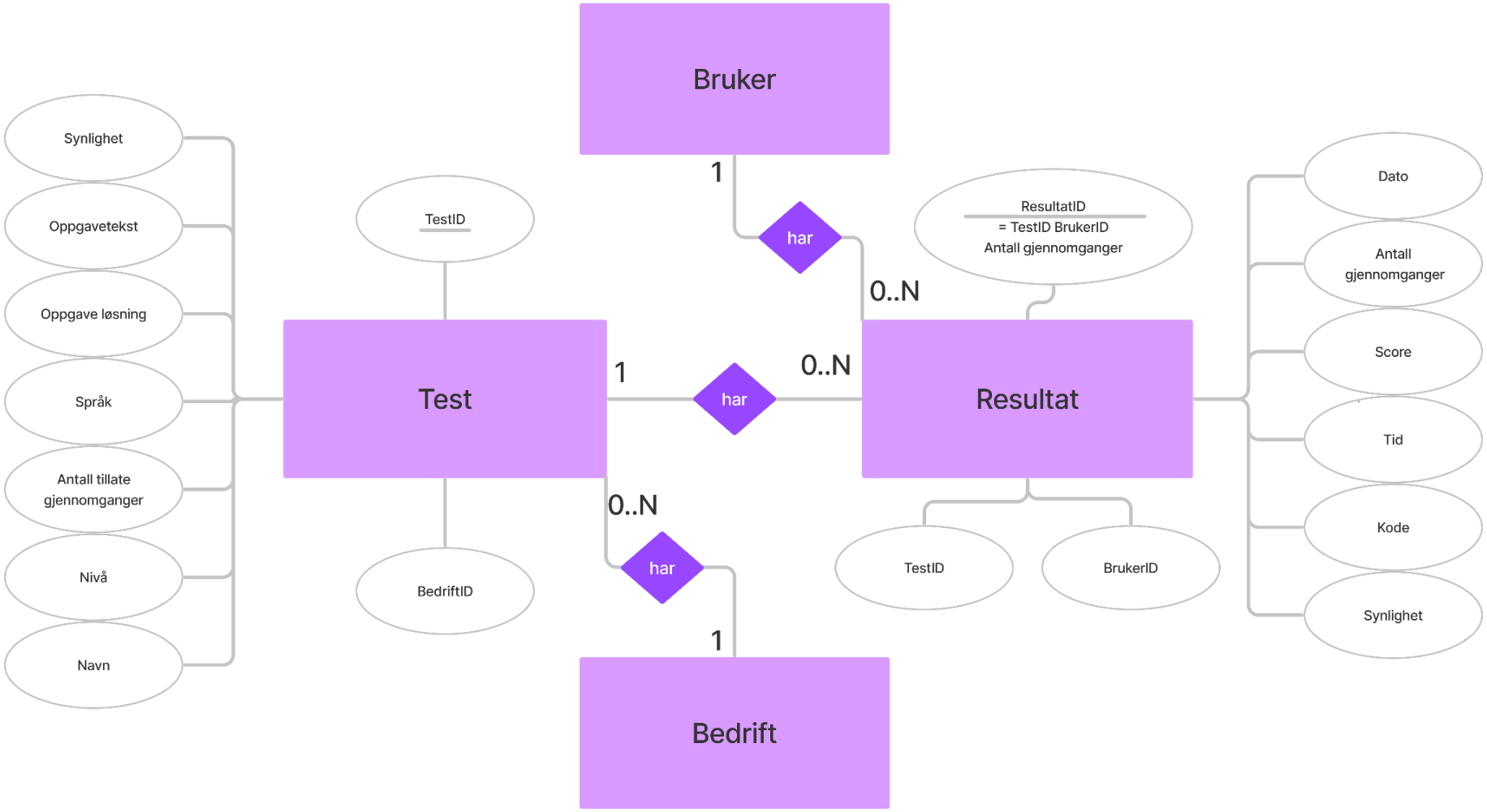
Vedlegg 2: Forstørret versjon av domenemodell for kundesidene (Figur 2.2)



Vedlegg 3: Forstørret versjon av kandidat klassediagram (Figur 4.1)



Vedlegg 4: Forstørret versjon av kunde klassediagram (Figur 4.2)



Vedlegg 5: Forstørret versjon av ER-diagram for databaseopsett (Figur 5.1).